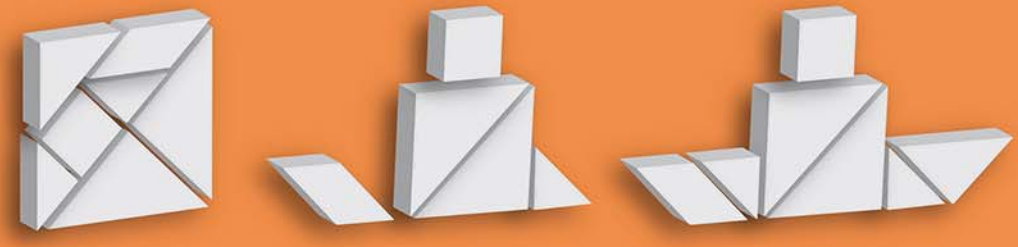
Владимир Дронов

HTML и CSS

33 урока для начинающих



Материалы на www.bhv.ru

bhv®

Владимир Дронов

HTML и CSS

33 урока для начинающих

Санкт-Петербург
«БХВ-Петербург»
2025

УДК 004.43+004.738.5
ББК 32.973.26–018.1
Д75

Дронов В. А.

Д75 HTML и CSS: 33 урока для начинающих. — СПб.: БХВ-Петербург, 2025. — 640 с.: ил.

ISBN 978-5-9775-2007-2

В книге 33 иллюстрированных урока, 50 практических упражнений по разработке веб-страниц и веб-сайтов разной сложности и 40 заданий для самостоятельной работы. Дано введение в веб-разработку, раскрыты основы HTML, CSS и работы в WWW. Объяснено, как с помощью HTML структурировать и форматировать текст, размещать на веб-страницах графические изображения, аудио- и видеоролики, таблицы, гиперссылки, веб-формы с элементами управления. Рассказано, как средствами CSS оформлять веб-страницы, текст, изображения, гиперссылки, веб-формы, элементы управления, задавать фон, местоположение элементов на страницах, создавать элементы непрямоугольной формы, применять фильтры, использовать двухмерные и трехмерные преобразования, создавать анимацию. Рассмотрены вопросы адаптации страниц под мобильные устройства.

Электронное приложение-архив на сайте издательства содержит коды всех примеров и учебных веб-сайтов.

Для начинающих веб-разработчиков

УДК 004.43+004.738.5
ББК 32.973.26–018.1

Группа подготовки издания:

Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Людмила Гауль*
Редактор	*Григорий Добин*
Компьютерная верстка	*Людмилы Гауль*
Корректор	*Светлана Крутоярова*
Оформление обложки	*Зои Канторович*

«БХВ-Петербург», 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-5-9775-2007-2

Оглавление

Предисловие

Эта книга — для начинающих разработчиков, пока еще «чайников» в *веб-верстке*, но желающих узнать на практике, как делаются веб-страницы и веб-сайты. В ней рассмотрены все необходимые инструменты, которыми должен владеть каждый уважающий себя веб-верстальщик.

Объем книги относительно невелик, материал можно освоить за полтора месяца. В итоге читатель научится верстать небольшие веб-сайты с привлекательным дизайном, который может удовлетворить потенциального заказчика. Вы получите практический опыт веб-верстки, узнаете, как применять актуальные веб-технологии для достижения нужного результата, и — кто знает — возможно, начнете карьеру профессионального веб-верстальщика. Более того, в процессе изучения книги вы самостоятельно создадите несколько сайтов, вполне пригодных для публикации в Сети и включения в портфолио.

На каких принципах основана эта книга?

Практика — вот девиз, которому следовал автор. Практика и необходимая теория, даваемая по ходу дела.

Книга построена на основе следующих принципов:

- материал разбит на 33 коротких наглядных иллюстрированных урока, в которых теория соседствует с практическими упражнениями;
- побольше иллюстраций: «лучше один раз показать, чем сто раз рассказать»;
- говоря о каком-либо инструменте, автор старается продемонстрировать (на конкретном примере), чем он может быть полезен;
- дается максимально полное описание всех актуальных веб-технологий, чтобы читатель смог выбрать из них наиболее подходящие в каждом конкретном случае.

  «За кадром» остались лишь средства, редко используемые и поддерживаемые не всеми основными веб-обозревателями (в число которых

входят Google Chrome, Mozilla Firefox, Microsoft Edge Chromium, Opera и Apple Safari);

- рассказ ведется простым человеческим языком, с минимумом специальной терминологии;
- и никакой «воды» — только по делу!

Что в ней описано?

В книге мы изучим все технологии, применяемые при разработке веб-страниц и веб-сайтов:

- HTML — язык, на котором пишутся сами веб-страницы. Именно с его помощью текст разбивается на абзацы и заголовки, создаются списки и таблицы, на страницы помещаются графические изображения, аудио- и видеоролики;
- CSS — язык, на котором описывается оформление веб-страниц: каким шрифтом нужно вывести текст абзацев, а каким — заголовков, каким фоном следует залить панель навигации, какой рамкой нужно ее окружить и как сделать так, чтобы при наведении курсора мыши на гиперссылку она плавно меняла цвет (т. е. создать анимацию);
- принципы функционирования Всемирной паутины (WWW): что такое клиент, сервер, протокол HTTP и страница по умолчанию.

Каждый урок содержит:

- теоретические части (в самом деле, куда же без теории?..);
- упражнения, выполняемые под руководством автора. Они представляют собой набор пронумерованных действий, которые необходимо выполнить для достижения поставленного результата. Как правило, каждое упражнение призвано закрепить знания, полученные из предыдущей теоретической части;
- самостоятельные упражнения — их следует выполнить вам, уважаемые читатели.

В процессе учебного курса мы создадим несколько веб-сайтов: придуманного автором суши-бара «Йокогама», видеокиоска, фотогалереи великой группы «The Beatles», а также несколько рекламных сайтов.

Книгу сопровождает файловый архив, содержащий необходимые для работы материалы, равно как и результаты выполнения всех упражнений (см. *приложение 4*). Архив доступен по ссылке **https://zip.bhv.ru/9785977520072.zip** и со страницы книги на сайте **https://bhv.ru/**.

Что нового?

За прошедшие с момента выхода предыдущего издания книги[1] годы в HTML и CSS появилось много нового. Вот наиболее значительные нововведения (табл. П.1).

Таблица П.1

Нововведение	Раздел, урок, где описаны
Группы колонок и колонки	*Разд. 8.7* и *16.2.5*
Спойлер	*Разд. 10.5* и *18.4*
Индикатор процесса	Разд. 10.6.1
Метр	Разд. 10.6.2
Переменные CSS	Разд. 12.3
Псевдокласс `:has()`	Разд. 13.8
Указание соотношения сторон в CSS	Разд. 15.6
Задание местоположения и размеров изображения и видео	*Разд. 17.3* и *17.4*
Угловой градиент	Разд. 19.2.3
Смешивание содержимого элемента и фона его родителя	Разд. 19.6
Приклеивающиеся элементы	Разд. 22.2.1.4
Сеточная разметка	Урок 24
Создание элементов произвольной формы	Разд. 26.2
Фильтры CSS	Урок 27
Трехмерные преобразования CSS	Урок 29
Анимация по фигуре	Разд. 30.5
Адаптация элементов под размеры их родителей	Разд. 31.4
Адаптивные изображения	Разд. 31.6

[1] Дронов В. HTML и CSS. 25 уроков для начинающих. СПб.: БХВ-Петербург, 2020.

Типографские соглашения

В книге будут часто приводиться форматы написания различных языковых конструкций HTML и CSS и примеры готового кода. Для наглядности при их написании использованы следующие типографские соглашения (в реальном коде они недействительны):

- HTML- и CSS-код набран моноширинным шрифтом. Примеры:

```
<h1>Суши-бар <em>Йокогама</em></h1>
<p><img src="images/sushi.jpg"></p>

h1 {
    font-size: 32pt;
    text-align: center;
}
```

- в угловые скобки (<>) заключаются наименования различных значений, которые дополнительно выделяются курсивом. В реальный код, разумеется, должны быть подставлены реальные значения. Например:

  `url(<`*`ссылка на сторонний файл`*`>)`

  Здесь вместо подстроки *ссылка на сторонний файл* должна быть подставлена реальная ссылка на сторонний файл;

- в квадратные скобки ([]) заключаются необязательные фрагменты кода. Например:

  <*имя создаваемого счетчика*> [<*изначальное значение*>]

  Здесь *изначальное значение* может указываться, а может и не указываться;

- вертикальной чертой (|) разделяются доступные для выбора значения, из которых в код можно подставить лишь одно. Например:

  `orientation: portrait|landscape`

  Здесь после двоеточия необходимо подставить либо `portrait`, либо `landscape`, но не то и другое одновременно и не какое-либо иное значение;

- слишком длинные, не помещающиеся на одной строке фрагменты кода автор разрывал на несколько строк и в местах разрывов ставил знаки ↳. Например:

```
content="width=device-width, initial-scale=1, ↳
         minimum-scale=0.5, maximum-scale=2">
```

  Приведенный код здесь разбит на две строки, но должен быть набран в одну. Символ ↳ при этом нужно удалить;

- троеточием (. . .) помечены фрагменты кода, пропущенные ради краткости. Пример:

```
<h1>Суши-бар <em>Йокогама</em></h1>
. . .
<p>Ждем вас!</p>
```

  Здесь весь код между двумя приведенными строками пропущен.

  Обычно такое можно встретить в исправленных впоследствии фрагментах кода — приведены лишь собственно исправленные строки, а оставшиеся неизмененными пропущены. Также троеточие используется, чтобы показать, в какое место должен быть вставлен вновь написанный код: в начало исходного фрагмента, в его конец или в середину, между уже присутствующими в нем строками;

- полужирным шрифтом выделен вновь добавленный код;
- зачеркиванием выделяется код, подлежащий удалению.

  Пример:

```
. . .
~~<tr><th>Суши</th><th></th></tr>~~
<tr><th colspan="2">Суши</th></tr>
. . .
```

  Здесь первая строка HTML-кода удалена, а вторая — добавлена.

ВНИМАНИЕ!

Все приведенные здесь типографские соглашения имеют смысл только в форматах написания различных языковых конструкций HTML и CSS. В реальном коде они не используются — за исключением знака ↳, троеточия, полужирного и зачеркнутого текста.

ЧАСТЬ I

НАЧАЛА ВЕБ-ВЕРСТКИ

- Наша первая веб-страница. Язык HTML.
- Оформление веб-страниц посредством стилей. Язык CSS.
- Как работает Всемирная паутина? Клиенты и серверы. Интернет-адреса.

Урок 1. Введение в язык HTML

Создание веб-страниц.
Теги и атрибуты тегов.
Вложенность тегов.
Специальные символы.
Графика.

Учиться веб-верстке лучше всего, выполняя какие-либо практические упражнения и попутно знакомясь с теорией. Так увлекательнее и доходчивее.

Мы поступим так же и будем изучать веб-технологии на примере создания веб-сайта гипотетического суши-бара «Йокогама».

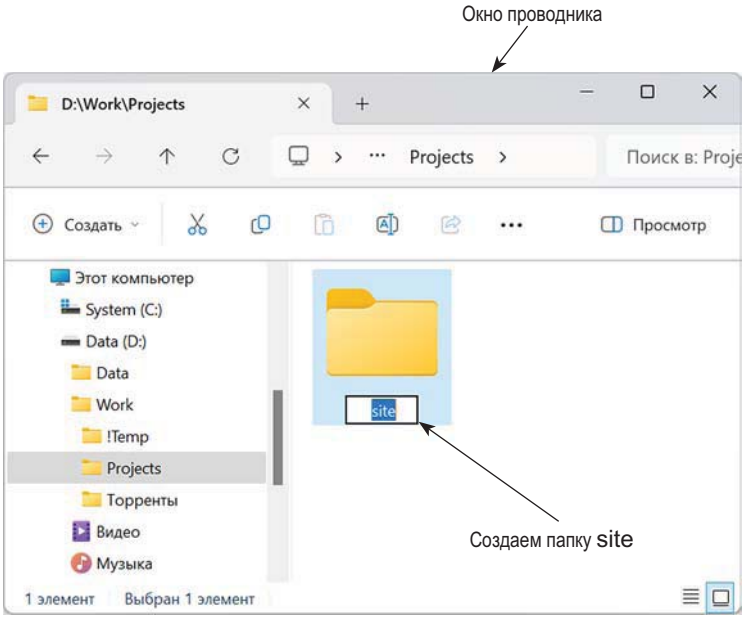
1.1. Упражнение. Первая веб-страница

Создадим нашу первую, совсем простенькую веб-страничку, включающую заголовок и несколько абзацев. На ее примере мы далее будем изучать принципы написания веб-страниц.

1. Откроем окно Проводника.

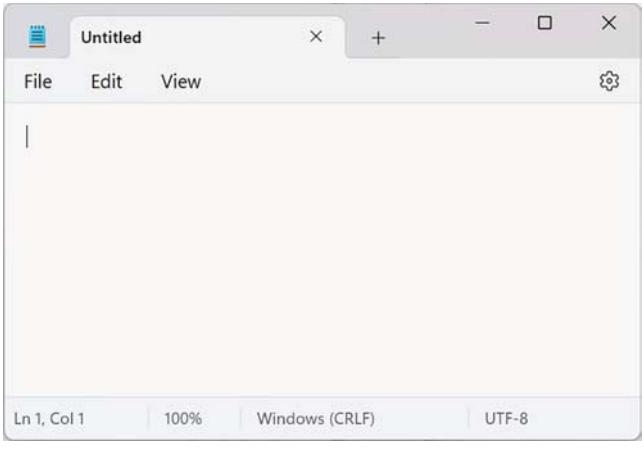
   Все файлы, входящие в состав веб-сайта: страницы, графические изображения, аудио-, видеоролики и пр., — должны храниться в особой папке или вложенных в нее подпапках.

2. Создадим папку, в которой будут находиться все файлы нашего будущего веб-сайта. Назовем ее site.



Веб-страницы представляют собой обычные текстовые файлы, только имеющие расширение html (а не txt). Их можно создавать в любом текстовом редакторе — например, в Блокноте[1], поставляемом в составе Windows 10/11.

3. Запустим Блокнот, выбрав в меню **Пуск** Windows 11 пункт **Все приложения** | **Блокнот** или в меню **Пуск** Windows 10 пункт **Стандартные** | **Блокнот**.

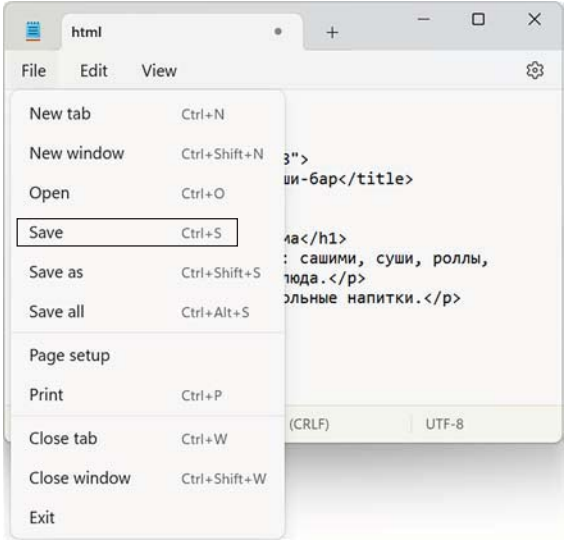


[1] Автор приносит свои извинения за показ в скриншотах английской версии Блокнота, где все пункты меню обозначены не по-русски.

4. Наберем в Блокноте следующий текст, стараясь не допускать ошибок:

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Йокогама: суши-бар</title>
    </head>
    <body>
        <h1>Суши-бар Йокогама</h1>
        <p>У нас вы найдете: сашими, суши, роллы,
           соусы, горячие блюда.</p>
        <p>Также - безалкогольные напитки.</p>
        <p>Ждем вас!</p>
    </body>
</html>
```

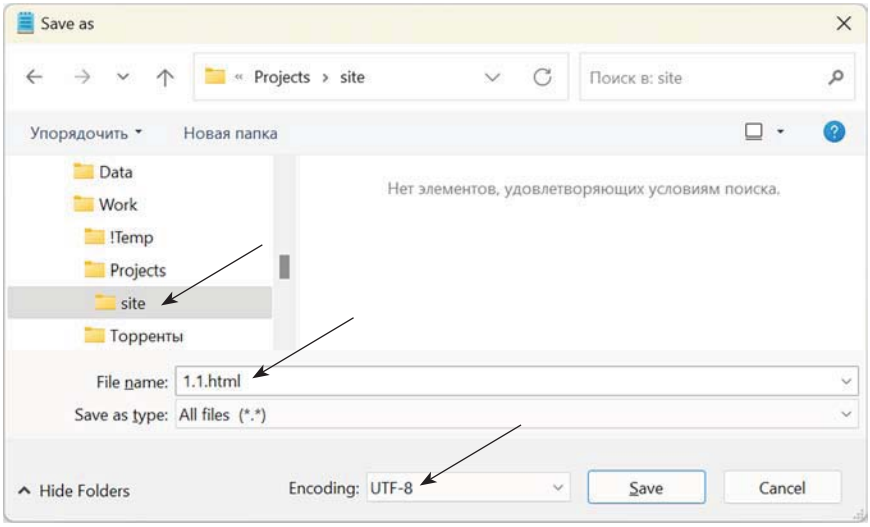
5. Сохраним набранный текст в файле. Это выполняется выбором в меню **File** пункта **Save** (**Файл** | **Сохранить**). Также можно просто нажать комбинацию клавиш <Ctrl>+<S>.



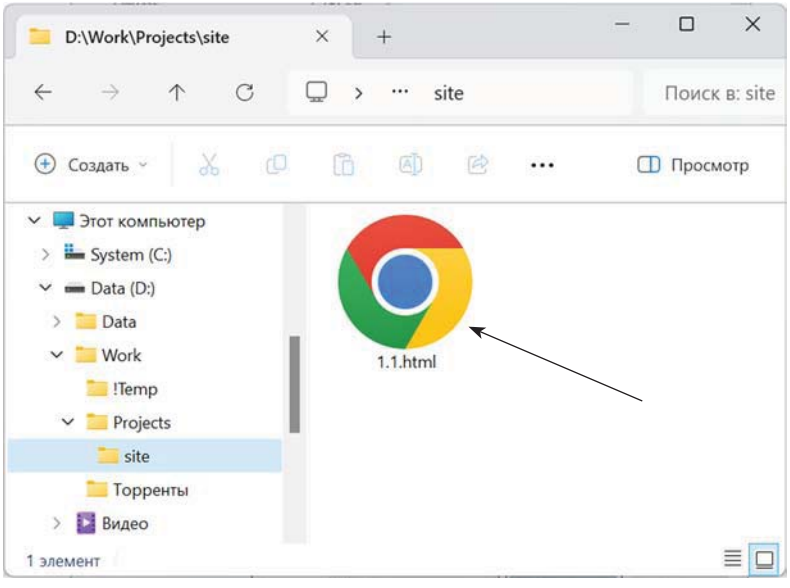
6. Укажем в качестве местоположения сохраняемого файла созданную ранее папку site. Дадим файлу имя 1.1.html, набрав его в поле ввода **File name** (**Имя файла**)[2]. В раскрывающемся списке **Encoding** (**Кодировка**)

[2] Здесь и далее в книге, если при описании какого-либо элемента управления указывается его название в английской редакции приложения, то за ним в скобках приводится это название по-русски.

выберем кодировку **UTF-8**. В раскрывающемся списке **Save as type** (**Тип файла**) выберем пункт **All files (*.*)** (**Все файлы (*.*)**), иначе Блокнот при сохранении добавит к введенному имени файла расширение txt. Для собственно сохранения файла нажмем кнопку **Save** (**Сохранить**).



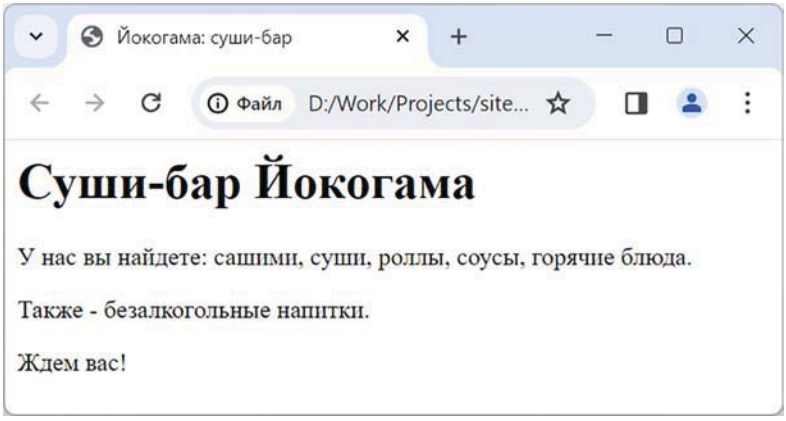
7. Откроем только что созданный файл 1.1.html[3] двойным щелчком мыши.



[3] Здесь и далее пути к файлам сайта указываются относительно папки, в которой находится содержимое сайта (в нашем случае — site).

✓ Результат

Веб-страница будет открыта в веб-обозревателе, зарегистрированном в системе как программа для открытия HTML-файлов по умолчанию. У автора книги это Google Chrome.



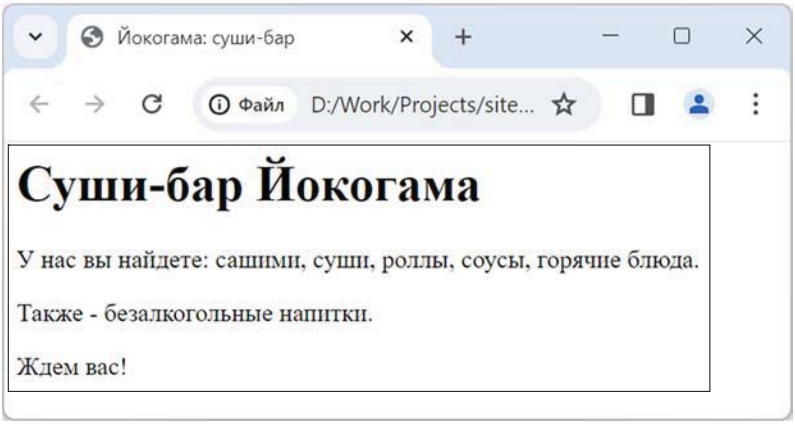
Оставим текстовый редактор с текстом страницы и веб-обозреватель, выводящий страницу, запущенными. Они пригодятся нам при выполнении *упражнения 1.3*.

1.2. Теги и атрибуты тегов. Язык HTML

Если рассмотреть набранный нами в Блокноте текст страницы...

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Йокогама: суши-бар</title>
    </head>
    <body>
        <h1>Суши-бар Йокогама</h1>
        <p>У нас вы найдете: сашими, суши, роллы,
           соусы, горячие блюда.</p>
        <p>Также - безалкогольные напитки.</p>
        <p>Ждем вас!</p>
    </body>
</html>
```

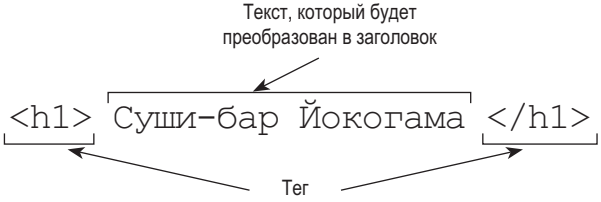
...нетрудно найти часть, создающую «видимое» содержание веб-страницы: заголовок и три абзаца (в приведенном тексте и на следующем рисунке она обведена рамкой).



Видно, что она состоит из фрагментов обычного текста и пометок в виде латинских букв и цифр, взятых в угловые скобки (выделены полужирным шрифтом):

```
<h1>Суши-бар Йокогама</h1>
<p>У нас вы найдете: сашими, суши, роллы, соусы,
   горячие блюда.</p>
<p>Также - безалкогольные напитки.</p>
<p>Ждем вас!</p>
```

Рассмотрим первую строку, создающую заголовок:



> ***Тег***
>
> Пометка, указывающая веб-обозревателю создать на основе помеченного ею текста определенный элемент страницы (в нашем случае — заголовок).

Тег `<h1>` создает заголовок первого уровня, которым предваряются самые крупные фрагменты текста (в нашем случае — это сама страница).

Тег `<h1>` — парный.

Парный тег

Состоит из открывающего тега, содержимого и закрывающего тега.

Содержимое тега

Фрагмент содержимого страницы, на который распространяется действие тега.

Открывающий тег

Отмечает начало содержимого тега.

Закрывающий тег

Отмечает конец содержимого тега.

Посмотрим на следующую строчку набранного нами текста:

```
<p>У нас вы найдете: сашими, суши, роллы, соусы,
   горячие блюда.</p>
```

Здесь используется тег `<p>`, который формирует на странице обычный абзац и, как и `<h1>`, является парным. В последующем тексте присутствуют еще два тега `<p>`, создающие остальные абзацы.

А что же остальной текст (здесь он также обведен рамкой)?

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Йокогама: суши-бар</title>
    </head>
    <body>
        <h1>Суши-бар Йокогама</h1>
        <p>У нас вы найдете: сашими, суши, роллы,
           соусы, горячие блюда.</p>
        <p>Также - безалкогольные напитки.</p>
        <p>Ждем вас!</p>
    </body>
</html>
```

Это тоже теги. Они структурируют текст и задают сведения о самой странице, используемые веб-обозревателем при ее выводе. Более подробно мы рассмотрим их на *уроке 4*.

А пока что отметим четыре примечательных тега. Во-первых, это парный тег <body>, в котором находятся все теги, создающие «видимое» содержание страницы (этот тег выделен полужирным шрифтом):

```
<body>
    <h1>Суши-бар Йокогама</h1>
    . . .
    <p>Ждем вас!</p>
</body>
```

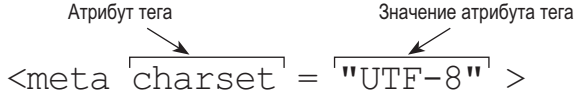
Тег <body> создает *секцию тела страницы*.

Во-вторых, парный тег <title>, который задает название страницы:

```
<title>Йокогама: суши-бар</title>
```

> ***Название веб-страницы***
>
> Выводится на корешке вкладки веб-обозревателя, на которой открыта страница, а также отображается в журнале и избранном веб-обозревателя.

В-третьих, это тег <meta>, в котором записана текстовая кодировка страницы:



Этот тег является одинарным и содержит атрибут.

> ***Одинарный тег***
>
> Не включает содержимого и закрывающего тега и фактически представляет собой лишь открывающий тег.
>
> ***Атрибут тега***
>
> Параметр элемента страницы, создаваемого тегом.
>
> ***Значение атрибута тега***
>
> Значение соответствующего параметра тега.

Тег <meta> служит для указания какой-либо из характеристик страницы (например, текстовой кодировки, в которой был сохранен ее текст). Атрибут тега charset, поддерживаемый этим тегом, задает наименование кодировки (в нашем случае — UTF-8).

В теге можно записать произвольное количество атрибутов, разделив их пробелами.

Наконец, в-четвертых, заслуживает внимания парный тег `<head>`, в котором записываются всевозможные сведения о странице, используемые веб-обозревателем при ее выводе, в частности, ее название (тег `<title>`) и текстовая кодировка (тег `<meta>`). Этот тег создает *секцию заголовка страницы*.

Свод описаний различных тегов и правил их применения составляет язык HTML.

> ***HTML***
>
> HTML (HyperText Markup Language, язык гипертекстовой разметки) — служит для написания содержания веб-страниц.
>
> Существуют пять разновидностей (версий) языка HTML. Текущей сейчас является версия 5 этого языка, окончательно утвержденная в 2014 году.

ВНИМАНИЕ!

Текст веб-страницы, написанный на языке HTML, носит название *HTML-кода*, или просто *кода*. Так мы и будем называть его в дальнейшем (чтобы не путать с текстом, который выводится на странице).

Всемирная паутина (WWW)

Годом рождения Всемирной паутины (WorldWideWeb, WWW) считается 1989-й. Именно тогда работавший в ЦЕРН британец Тим Бернерс-Ли разработал первую версию языка HTML, сетевой протокол HTTP, определяющий правила пересылки файлов, систему интернет-адресов, первый веб-обозреватель и первый веб-сервер httpd.

Интернет-стандартами заведует организация под названием World Wide Web Consortium (консорциум Всемирной паутины), сокращенно — WWWC или W^3C.

1.3. Упражнение. Выделение текста и списки

Первые два абзаца нашей страницы содержат перечень блюд, предлагаемых суши-баром «Йокогама». В таком виде он не очень смотрится. Давайте преобразуем его в полноценный список, пункты которого помечены маркерами (*маркированный список*). Кроме того, привлечем внимание посетителя нашего будущего сайта к тому факту, что упомянутое заведение предлагает также горячие блюда и безалкогольные напитки.

Начнем с создания маркированного списка.

1. Внесем в код страницы из файла 1.1.html следующие правки[4]:

```
<h1>Суши-бар Йокогама</h1>
<p>У нас вы найдете: сашими, суши, роллы,
   соусы, горячие блюда.</p>
<p>Также - безалкогольные напитки.</p>
<p>У нас вы найдете:</p>
<ul>
    <li>сашими,</li>
    <li>суши,</li>
    <li>роллы,</li>
    <li>соусы,</li>
    <li>горячие блюда,</li>
    <li>также - безалкогольные напитки.</li>
</ul>
<p>Ждем вас!</p>
```

Сам маркированный список создается с помощью парного тега <ul>. Отдельный пункт этого списка создается парным тегом <li>, помещенным в тег <ul>.

Теперь привлечем особое внимание посетителей к фразе «безалкогольные напитки» и, в меньшей степени, — к фразе «горячие блюда».

2. Исправим код страницы так:

```
. . .
<ul>
    . . .
    <li>горячие блюда,</li>
    <li><strong>горячие блюда</strong>,</li>
    <li>также - безалкогольные напитки.</li>
    <li>также - <em>безалкогольные напитки</em>.</li>
</ul>
. . .
```

Парный тег помечает свое содержимое как **очень важный текст**, выводимый полужирным шрифтом. А парный тег превращает содержимое в *важный текст*, который имеет меньшую важность, нежели созданный тегом , и выводится курсивом.

[4] Здесь и далее добавляемый и удаляемый код в примерах будет приводиться в соответствии с типографскими соглашениями, описанными в *предисловии*.

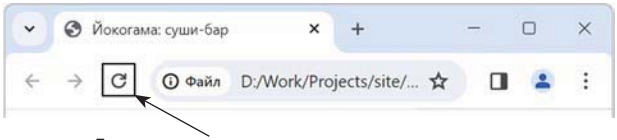
В последнем пункте списка присутствует дефис, но, согласно правилам русской орфографии, там следует использовать длинное тире. Кроме того, пробел, предшествующий длинному тире, рекомендуется сделать неразрывным.

> ***Неразрывный пробел***
>
> Пробел, по которому веб-обозреватель никогда не будет выполнять перенос строк. Визуально не отличается от обычного пробела.

3. Исправим код, создающий последний пункт списка, следующим образом:

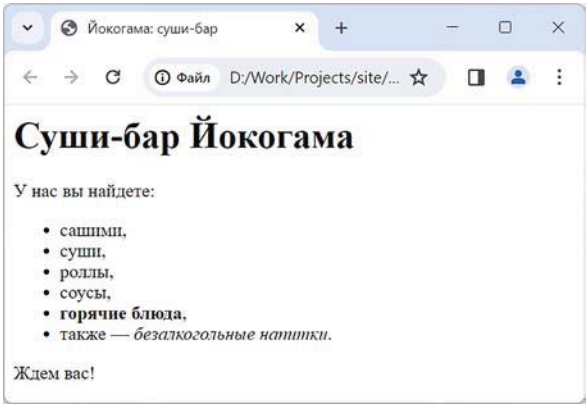
```
. . .
<li>также – <em>безалкогольные напитки</em>.</li>
<li>также &mdash; <em>безалкогольные напитки</em>.</li>
. . .
```

4. Сохраним исправленный файл.
5. Обновим открытую в веб-обозревателе страницу 1.1.html, чтобы увидеть результат.



Для этого щелкнем расположенную в панели инструментов кнопку Обновить (у разных программ она имеет разный вид) или просто нажмем клавишу <F5>.

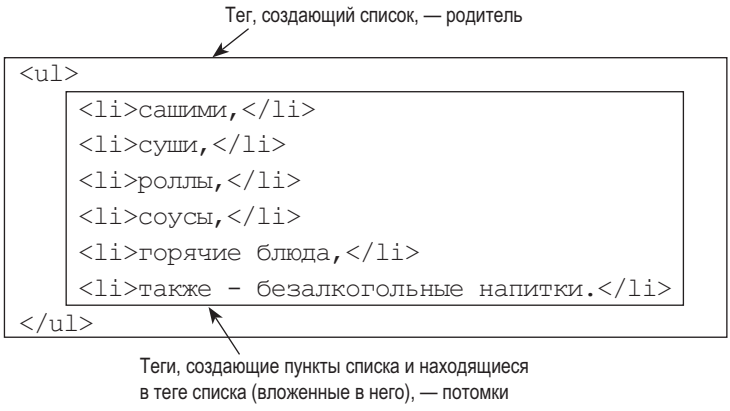
✓ Результат



Опять же, оставим редактор с кодом страницы и веб-обозреватель с самой страницей открытыми. Они понадобятся нам при выполнении *упражнения 1.5*.

1.4. Вложенность тегов. Родитель, соседи и потомки. Недопустимые и специальные символы

Снова взглянем на набранный ранее код...



Вложенный тег

Тег, являющийся частью содержимого другого тега.

Родитель

Элемент страницы, в который вложены другие элементы.

Потомок

Элемент, вложенный в другой элемент.

Элемент-потомок является частью элемента-родителя. Так, в нашем случае все пункты являются частью списка.

Любой потомок может являться родителем другого элемента. Например, предпоследний пункт списка является потомком списка и родителем очень важного текста (вложенность тегов обозначена вложенными друг в друга рамками):

```
<ul>
    <li>
        <strong>горячие блюда</strong>
    </li>
</ul>
```

Сосед

Элемент, имеющий того же родителя.

Например, соседями первого пункта списка являются второй и последующие пункты.

Вложенность тегов в HTML — явление обычное. Если вы посмотрите на полный HTML-код страницы, то увидите, что все теги, создающие «видимое» содержание страницы, вложены в парный тег `<body>`, а тот, в свою очередь, — в парный тег `<html>`. Два упомянутых тега структурируют код страницы и рассматриваются на *уроке 4*.

Теперь обратим внимание на код последнего пункта списка, в который мы ранее вставили неразрывный пробел и длинное тире:

```
<li>также     &mdash;  <em>безалкогольные напитки</em>.</li>
```

Специальные символы

Недопустимый символ

Символ, недопустимый в текстовом содержимом тегов. К таковым относятся, в частности, символы <, >, & и «, которые применяются при написании тегов.

Специальный символ

Последовательность символов, применяемая для вставки в HTML-код недопустимого символа или знака, который нельзя напрямую ввести с клавиатуры.

Как вы, уважаемые читатели, уже успели догадаться, неразрывный пробел обозначается специальным символом ` `, а длинное тире — специальным символом `&mdash;`. Другие наиболее часто применяемые специальные символы мы изучим на *уроке 4*.

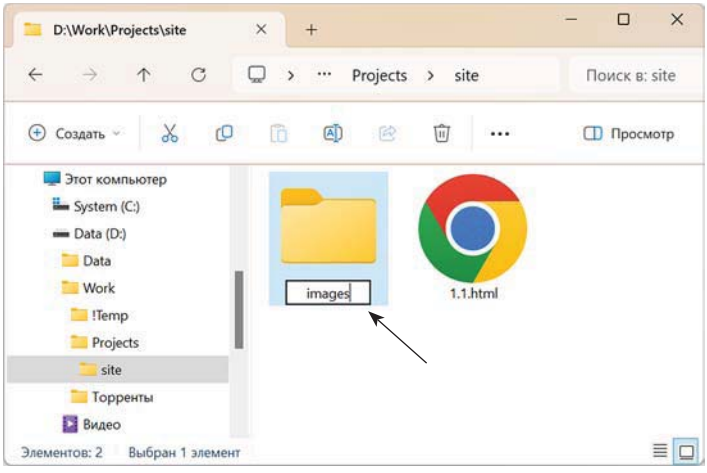
1.5. Упражнение. Интернет-графика

Что-то скучно выглядит наша страничка без картинок... Давайте поместим на нее парочку.

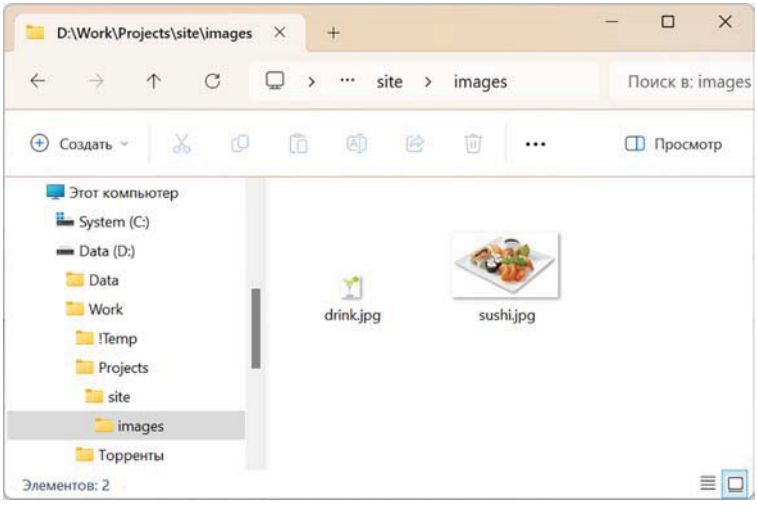
СОВЕТ

Файлы с графическими изображениями, помещаемыми на страницы веб-сайта, рекомендуется сохранять в отдельной папке, вложенной в папку с содержимым сайта.

1. Создадим в папке site папку images, в которую впоследствии поместим файлы с изображениями.



2. Найдем в папке 1\!sources сопровождающего книгу файлового архива (см. *приложение 4*) файлы sushi.jpg и drink.jpg с изображениями соответствующей тематики. Скопируем их в ранее созданную папку images.



3. Исправим код страницы из файла 1.1.html следующим образом:

```
<h1>Суши-бар Йокогама</h1>
<p><img src="images/sushi.jpg"></p>
<p>У нас вы найдете:</p>
<ul>
    . . .
```

```
<li>также &mdash; <em>безалкогольные напитки</em>.</li>
<li>также &mdash; <em>безалкогольные напитки</em>
    <img src="images/drink.jpg">.</li>
</ul>
<p>Ждем вас!</p>
```

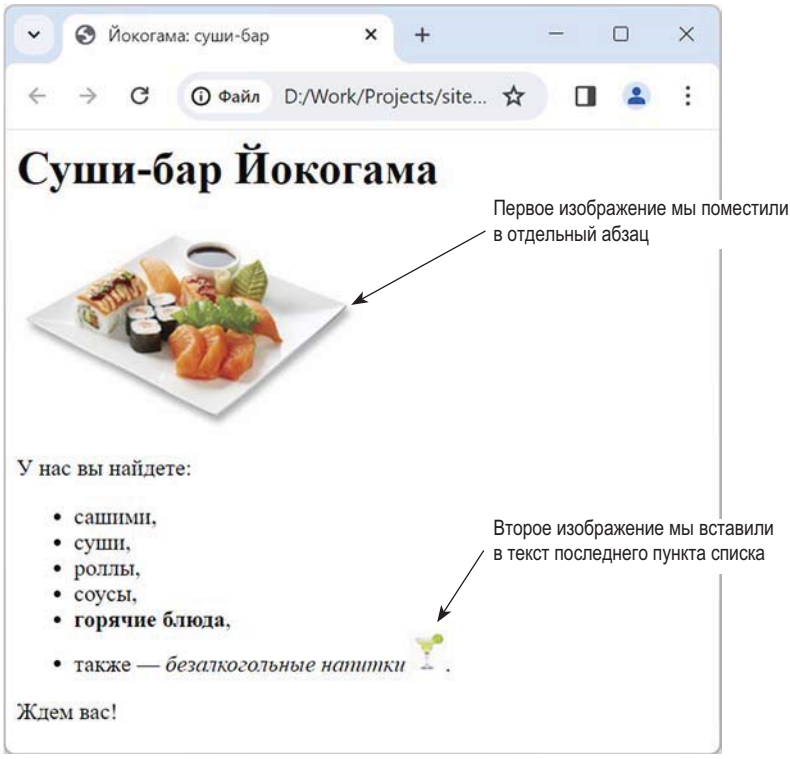
Для помещения на страницу графического изображения применяется одинарный тег `<img>`. В атрибуте `src` этого тега задается ссылка на файл, в котором хранится нужное изображение. Такой ссылкой может быть как интернет-адрес, так и путь к файлу (как в нашем случае). Подробнее ссылки на файлы мы рассмотрим на *уроке 4*.

4. Сохраним исправленный файл и обновим открытую в веб-обозревателе страницу, чтобы увидеть результат.

ВНИМАНИЕ!

В дальнейшем не забываем выполнять эти действия после каждого исправления кода. Автор более не будет напоминать об этом!

✓ Результат



1.6. Самостоятельное упражнение

Добавьте на созданную ранее веб-страницу 1.1.html:

- список подаваемых напитков (аналогичный списку блюд — см. *упражнение 1.3*);
- физический адрес суши-бара (используйте парный тег `<address>`).

✓ У вас должно получиться ⌄

Урок 2. Введение в язык CSS

Оформление веб-страниц.
Таблицы стилей.
Стили и атрибуты стилей.
Селекторы.
Наследование.
Каскадность.

2.1. Оформление по умолчанию

Посмотрим на нашу первую веб-страничку:

Сейчас она оформлена по умолчанию:

- заголовок выводится полужирным шрифтом с большим кеглем, который установлен веб-обозревателем;

> ***Кегль***
>
> Размер шрифта.

- абзацы и адрес выводятся шрифтом обычной насыщенности и небольшого кегля, также установленного веб-обозревателем;
- гарнитура шрифта для вывода текста выбирается веб-обозревателем;

- текст и изображение выравниваются по левому краю;
- элементы отделяются друг от друга по вертикали просветами.

Оформление по умолчанию весьма невыразительно и практически всегда не то, что нужно веб-верстальщику.

2.2. Упражнение. Оформление веб-страницы

Давайте оформим нашу страничку как нравится нам, а не веб-обозревателю:

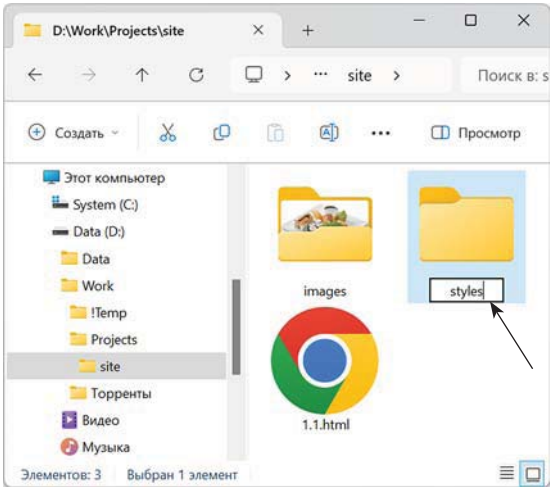
- для вывода текста используем шрифт Arial;
- для заголовка установим кегль шрифта, равный 32 пункта, и выравнивание по центру;
- для абзацев — кегль 14 пунктов;
- для списков — такой же кегль и маркер в виде квадратика.

1. Найдем в папке 1\s1.6 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

СОВЕТ

Файлы, содержащие правила оформления страниц, рекомендуется сохранять в отдельной папке.

2. Создадим в папке site папку styles, в которой позже сохраним файл с правилами оформления страницы.



3. Запустим Блокнот и наберем следующий код, стараясь не допускать ошибок:

```
body { font-family: Arial; }
h1 {
    font-size: 32pt;
    text-align: center;
}
p { font-size: 14pt; }
ul {
    font-size: 14pt;
    list-style-type: square;
}
```

4. Сохраним набранный код в файле в созданной ранее папке styles. Дадим файлу имя 2.1.css. В качестве кодировки укажем: **UTF-8**.

   Не забываем в раскрывающемся списке **Save as type** (**Тип файла**) указывать **All files (*.*)** (**Все файлы (*.*)**), чтобы Блокнот при сохранении не добавил к имени сохраняемого файла расширение txt.

5. Откроем файл 1.1.html и добавим в HTML-код страницы следующий фрагмент:

```
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="styles/2.1.css">
    <title>Йокогама: суши-бар</title>
</head>
```

6. Откроем страницу 1.1.html в веб-обозревателе.

✓ Результат ›

Оставим открытыми текстовый редактор с кодом и веб-обозреватель с выведенной страницей — они понадобятся нам для выполнения следующих упражнений.

2.3. Таблицы стилей, стили, атрибуты стилей, наследование и привязка. Язык CSS

Еще раз посмотрим на код, который мы сохранили в файле styles\2.1.css:

```
body { font-family: Arial; }
h1 {
    font-size: 32pt;
    text-align: center;
}
p { font-size: 14pt; }
ul {
    font-size: 14pt;
    list-style-type: square;
}
```

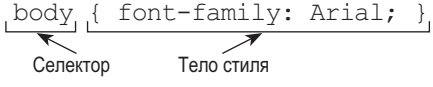
Это таблица стилей.

> ***Таблица стилей***
>
> Описание оформления различных элементов веб-страницы, выполненное в особом формате.

Наша таблица стилей состоит из четырех отдельных *стилей*, применяемых, соответственно, к секции тела страницы, заголовкам первого уровня, абзацам и спискам.

> ***Стиль***
>
> Описание оформления элемента или группы элементов страницы, удовлетворяющих определенному признаку (например, созданных одинаковыми тегами).

Рассмотрим первый стиль:

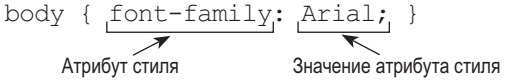


> ***Селектор***
>
> Указывает на элемент или группу элементов страницы, к которым будет применен стиль.
>
> ***Тело стиля (или описание стиля)***
>
> Собственно описание оформления, которое будет применено к элементам страницы, на которые указывает селектор.

Селектор `body` есть не что иное, как тег `<body>`, написанный без угловых скобок. Это селектор тега.

> ***Селектор тега***
>
> Обозначает все элементы страницы, созданные с применением заданного HTML-тега. Представляет собой тег, записанный без символов < и >.

Соответственно, рассматриваемый нами стиль будет применен к секции тела страницы (к тегу `<body>`).



> ***Атрибут стиля***
>
> Одна из настроек оформления: гарнитура шрифта, его кегль, выравнивание текста, форма маркера у пунктов списка и пр. Не путать с атрибутом тега!
>
> ***Значение атрибута стиля***
>
> Значение соответствующей настройки оформления.

Атрибут стиля `font-family` задает гарнитуру шрифта, которым будет выведен текст. В качестве значения этого атрибута задано название шрифта Arial.

Мы указали атрибут стиля `font-family` для тега `<body>` — родителя заголовка, абзацев, списков и адреса. Приведенные элементы, являющиеся потомками тега `<body>`, унаследуют стиль `font-family` из стиля, заданного у родителя. В результате весь текст на странице будет выведен шрифтом Arial.

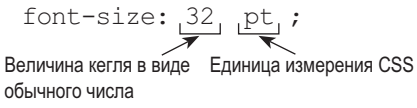
> ***Наследование атрибутов стиля***
>
> Автоматическое применение атрибута стиля, указанного у элемента-родителя, к элементам, являющимся его потомками.
>
> ***Наследуемый атрибут стиля***
>
> Атрибут стиля, который элементы-потомки наследуют у элемента-родителя.

Посмотрим на второй стиль:

```
h1 {
    font-size: 32pt;
    text-align: center;
}
```

В нем используется аналогичный селектор тега, но теперь указывающий на тег <h1>. Следовательно, этот стиль будет применен к заголовкам первого уровня.

Атрибут стиля font-size задает кегль шрифта.



Единица измерения pt — это типографские пункты. Остальные единицы измерения, поддерживаемые CSS, рассматриваются на *уроке 12*.

Атрибут стиля text-align определяет выравнивание текста. Его значение center задает выравнивание по центру.

Третий стиль:

```
p { font-size: 14pt; }
```

Он будет применен к абзацам — тегам <p> и задаст для них кегль шрифта, равный 14 пунктам.

Четвертый стиль:

```
ul {
    font-size: 14pt;
    list-style-type: square;
}
```

Он применится к маркированным спискам — тегам <ul>, укажет для их пунктов кегль шрифта 14 пунктов и маркер в виде закрашенного квадратика (его определит атрибут стиля list-style-type со значением square).

> ***CSS***
>
> CSS (Cascading Style Sheets, каскадные таблицы стилей) — язык, на котором пишется оформление веб-страниц, — таблицы стилей.
>
> Существуют три версии языка CSS. Наиболее актуальной версией сейчас является CSS 3. Ее спецификация была утверждена в 2011 году и до сих пор дополняется новыми инструментами.

ВНИМАНИЕ!

Текст таблицы стилей, написанный на языке CSS, веб-верстальщики называют *CSS-кодом*, или просто *кодом*, чтобы не путать его с обычным текстом. Так мы и будем называть его в дальнейшем.

Мы сохранили нашу первую таблицу стилей в отдельном файле, создав тем самым внешнюю таблицу стилей.

> ***Внешняя таблица стилей***
>
> Таблица стилей, хранящаяся в отдельном файле.

Чтобы оформление, записанное во внешней таблице стилей, было применено к странице, необходимо выполнить привязку таблицы стилей к этой странице.

> ***Привязка***
>
> Связывание внешней таблицы стилей с веб-страницей с целью применить к странице все стили, записанные в таблице стилей.

Привязка таблицы стилей к странице выполняется посредством одинарного тега `<link>`, который записывается в секции заголовка страницы (в теге `<head>`):

```
<link rel="stylesheet" href="styles/2.1.css">
```

Ссылка на файл с внешней таблицей стилей указывается в атрибуте `src` этого тега. А наличие в этом теге атрибута `rel` со значением `stylesheet` сообщает веб-обозревателю о том, что к странице привязывается именно таблица стилей.

Устаревшие HTML-теги

Ранее, до появления CSS, для оформления веб-страниц применялись особые HTML-теги. Так, для задания начертания, кегля и других параметров шрифта использовался парный тег `<font>`. Сейчас этот и другие теги схожего назначения удалены из спецификации HTML и веб-обозревателями более не поддерживаются.

2.4. Упражнение. Стилевые классы

Давайте сделаем следующее:

- зададим у последнего абзаца («Ждем вас!») шрифт с кеглем 24 пункта;
- выровняем изображение суши по центру.

Для этого придется создать еще два стиля. И мы сразу же столкнемся с проблемой: каждый из этих стилей должен быть применен только к определенному абзацу, не затрагивая остальных абзацев. Как это сделать? Использовав стилевые классы.

> ***Стилевой класс***
>
> Пометка, которая может быть указана у любого количества произвольных тегов с целью применить к ним какой-либо стиль. Представляет собой произвольную строку любых символов, за исключением пробелов.

Свой первый стилевой класс мы укажем у абзаца, в котором выводится изображение суши. Дадим этому стилевому классу имя `picture`. А позже напишем стиль, который сошлется на элемент с этим стилевым классом и задаст у него выравнивание по центру.

1. Внесем в HTML-код из файла 1.1.html следующее исправление:

```
. . .
<p><img src="images/sushi.jpg"></p>
<p class="picture"><img src="images/sushi.jpg"></p>
. . .
```

Стилевой класс, задаваемый у тега, указывается в атрибуте тега `class`.

Другой стилевой класс, с именем `greeting`, укажем у последнего абзаца. А потом напишем стиль, который для абзаца с этим стилевым классом укажет кегль 24 пункта.

2. Внесем в HTML-код из того же файла следующую правку:

```
. . .
<p>Ждем вас!</p>
<p class="greeting">Ждем вас!</p>
. . .
```

3. Добавим в таблицу стилей из файла styles\2.1.css стиль, содержащий селектор стилевого класса `picture` и задающий выравнивание по центру:

```
.picture { text-align: center; }
```

> ***Селектор стилевого класса***
>
> Ссылается на все элементы страницы со стилевым классом, записанным в селекторе. Представляет собой имя стилевого класса, предваренное точкой.

В стиле указан селектор `.picture` (имя соответствующего стилевого класса, предваренное точкой). Следовательно, стиль будет применен ко всем элементам страницы, помеченным стилевым классом `picture`. У нас такой элемент всего один — абзац с картинкой.

4. Добавим в таблицу стилей styles\2.1.css еще один стиль, который задаст для абзаца со стилевым классом `greeting` кегль в 24 пункта:

```
p.greeting { font-size: 24pt; }
```

В стиле применен комбинированный селектор `p.greeting`, представляющий собой комбинацию селектора тега `p` и селектора стилевого класса `.greeting`.

> ***Комбинированный селектор***
>
> Комбинация более простых селекторов (например, селектора тега и селектора стилевого класса). Указывает на элементы, удовлетворяющие одновременно всем записанным в нем простым селекторам.

Стиль с комбинированным селектором `p.greeting` будет применен только к абзацам со стилевым классом `greeting`, — то есть к абзацу «Ждем вас!».

5. Обновим страницу 1.1.html, открытую в веб-обозревателе.

✓ Результат ›

Оставим текстовый редактор и веб-обозреватель открытыми — упражнения в этом уроке еще не закончились.

2.5. Каскадность

Снова обратим внимание на последний абзац — с текстом «Ждем вас!» К нему фактически применены два стиля:

```
p { font-size: 14pt; }
p.greeting { font-size: 24pt; }
```

Первый стиль — с селектором тега `p`, задает кегль 14 пунктов, а второй — с селектором стилевого класса `.picture`, — кегль 24 пункта. Шрифтом какого кегля будет выведен текст абзаца?

Каждая из разновидностей селекторов, поддерживаемых языком CSS, имеет определенный *приоритет*. Приведем уже известные нам разновидности в порядке увеличения их приоритета:

- селектор тега;
- селектор стилевого класса;
- комбинированный селектор.

> ***Правило каскадности***
>
> Если два стиля задают разные значения для одной и той же настройки оформления, будет взято значение из стиля, чей селектор имеет больший приоритет.

В нашем случае значение кегля будет взято из второго стиля — с комбинированным селектором `p.greeting`, чей приоритет выше, чем у селектора тега из первого стиля.

2.6. Упражнение. Составные селекторы

Что-то у нас картинка суши маловата... Давайте ее увеличим! И не просто увеличим, а зададим для нее ширину, равную $^{2}/_{3}$ от ширины страницы.

Мы можем, как при выполнении *упражнения 2.4*, пометить тег `<img>` картинки каким-либо стилевым классом и написать стиль с селектором этого стилевого класса. А можем воспользоваться составным селектором.

> ***Составной селектор***
>
> Состоит из произвольного количества простых селекторов: селекторов тегов, селекторов стилевых классов, комбинированных селекторов и др. — отделенных друг от друга разделителями.
>
> Стиль всегда применяется к элементу, на который указывает последний из простых селекторов, входящих в составной селектор.
>
> ***Разделитель***
>
> Задает местоположение элемента, обозначенного селектором, который стоит справа от разделителя, относительно элемента, обозначенного селектором слева (например, правый элемент должен являться для левого элемента потомком, следующим соседом или др.).

Добавим в таблицу стилей из файла styles\2.1.css такой стиль:

```
.picture img { width: 67%; }
```

В стиле мы записали составной селектор `.picture img`, состоящий из двух селекторов и одного разделителя:

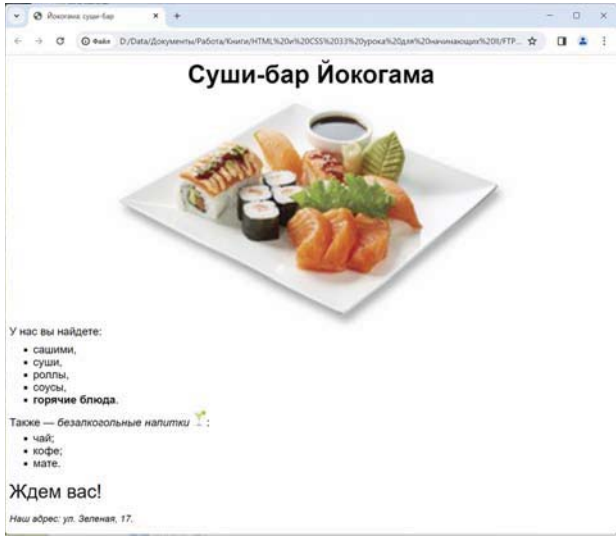
- селектор стилевого класса `.picture` — расположен слева, указывает на элемент со стилевым классом `picture`;
- селектор тега `img` — стоит справа, указывает на изображение;
- обычный пробел, отделяющий селекторы друг от друга, — «пустой» разделитель, или разделитель потомка.

> ***Разделитель потомка***
>
> Указывает, что элемент, обозначенный селектором справа, должен быть потомком элемента, обозначенного селектором слева.

Составные селекторы удобнее разбирать с конца. Так, составной селектор `.picture img` из нашего стиля укажет на изображение (селектор тега `img`), являющееся потомком (разделитель потомка) элемента со стилевым классом `picture` (селектор стилевого класса `.picture`). К этому изображению и будет применен стиль.

Атрибут стиля `width` задает ширину элемента страницы. Символ `%` обозначает проценты. В результате изображение суши будет иметь ширину в 67% (примерно $^{2}/_{3}$) от ширины страницы.

✓ Результат ⌄



Для достижения наилучшего эффекта следует увеличить ширину окна веб-обозревателя.

2.7. Самостоятельные упражнения

- Добавьте в уже существующий стиль, действующий на заголовки первого уровня, указание вывести заголовок шрифтом Verdana.
- Добавьте в таблицу стилей следующие стили:
  - задающий у адреса (тега `<address>`) выравнивание по правому краю. Используйте для этого значение `right` атрибута стиля `text-align`;

- задающий у фрагмента «горячие блюда», присутствующего в последнем пункте первого списка и заключенного в тег `<strong>`, курсивное начертание. Начертанием шрифта управляет атрибут стиля `font-style`. Значение `italic` делает шрифт курсивным.

✓ У вас должно получиться ⌄

Урок 3. Как работает WWW?

Клиенты и серверы.
Публикация веб-сайта.
Интернет-адрес.
Ошибка 404.

3.1. Клиенты и серверы

Все программы, применяемые для работы в Интернете, делятся на две группы:

Клиенты

Представляют пользователям информацию, отправленную им по сети программами-*серверами*.

В WWW программами-клиентами являются *веб-обозреватели*:

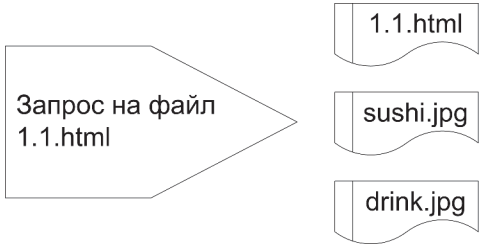
- Google Chrome;
- Mozilla Firefox;
- Microsoft Edge;
- Opera;
- Apple Safari и др.

Серверы

Хранят информацию, выполняют операции по ее обслуживанию и выдают программам-клиентам. В WWW программами-серверами являются веб-серверы:

- Apache HTTP Server (сокращенно — Apache);
- NGINX;
- Microsoft Internet Information Services и др.

Веб-обозреватель отправляет веб-серверу *клиентский запрос* (или просто *запрос*), в котором указывает путь нужного ему файла: веб-страницы, изображения, внешней таблицы стилей и пр.

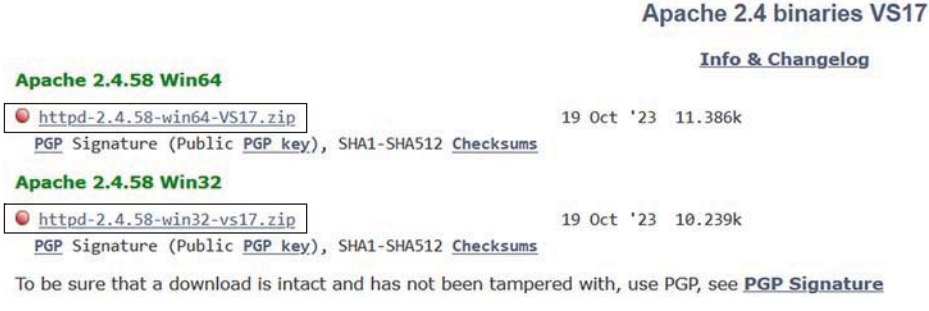


Веб-сервер, получив клиентский запрос, считывает с диска файл с указанным в запросе путем и отправляет веб-обозревателю в составе *серверного ответа* (или просто *ответа*).

3.2. Упражнение. Публикация веб-страницы на веб-сервере

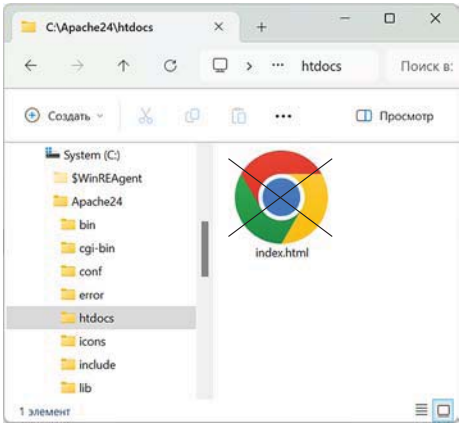
Установим на компьютер веб-сервер Apache и опубликуем на нем наш сайтик, пока состоящий всего из одной страницы.

1. Найдем в папке 2\s2.7 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.
2. Отправимся на страницу, с которой можно загрузить веб-сервер[1].
3. На открывшейся странице — найдем гиперссылки на файлы с редакциями дистрибутива веб-сервера: 32-разрядной и 64-разрядной. Все дистрибутивы Apache HTTP Server распространяются в виде архивов ZIP.

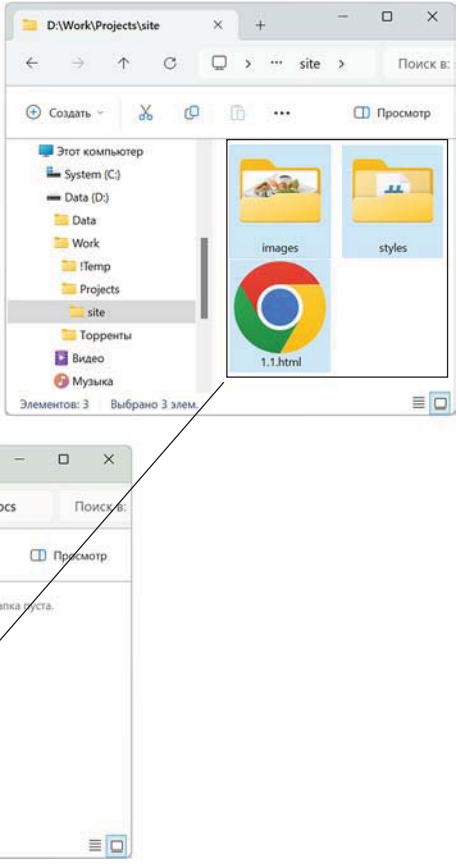


[1] См. **https://www.apachelounge.com/download/**.

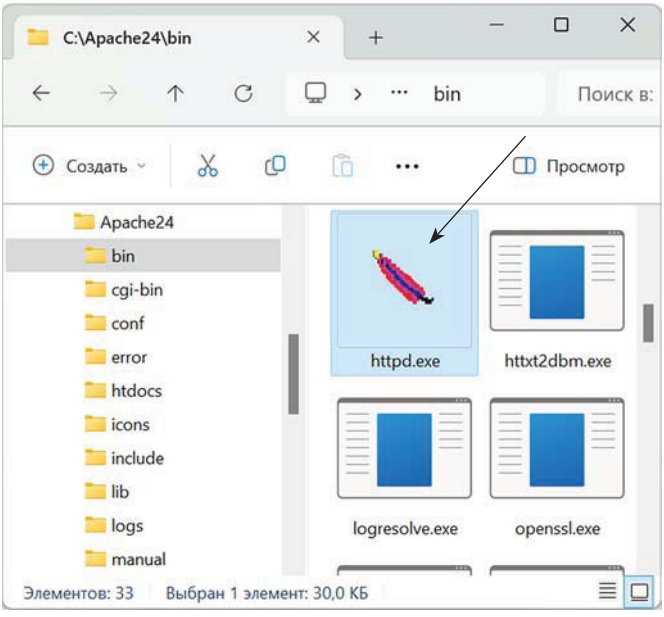
4. Загрузим 32- или 64-разрядную редакцию дистрибутива Apache — в зависимости от операционной системы нашего компьютера.
5. Распакуем загруженный архив в корневую папку диска C: — в результате на диске C: появится папка apache24.
6. Удалим файл index.html из папки C:\apache24\htdocs.



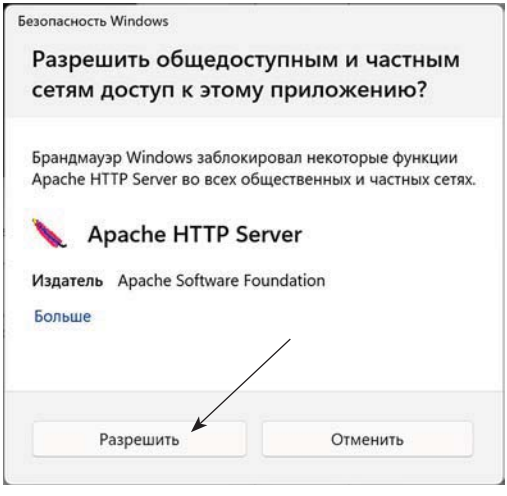
7. Скопируем все файлы, составляющие сайт, из папки site в папку C:\apache24\htdocs.



8. В папке C:\apache24\bin найдем файл httpd.exe (это исполняемый файл веб-сервера) и запустим его двойным щелчком мыши.



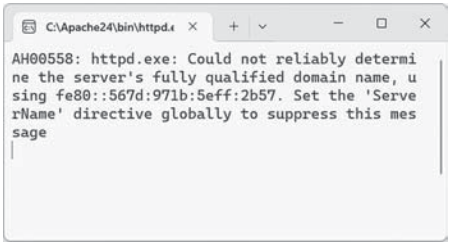
9. В появившемся на экране окне-предупреждении подсистемы UAC нажмем кнопку **Разрешить**, чтобы дать запущенному веб-серверу доступ к сети.



Если этого не сделать (или нажать кнопку **Отменить**), мы не сможем обратиться к веб-серверу даже с того же компьютера, на котором он запущен.

Запущенный Apache HTTP Server представляется на экране в виде черного или белого окна (цвет зависит от системных настроек — у автора книги

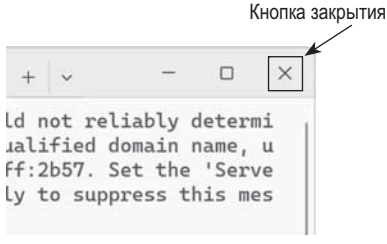
это окно белое), в котором выводятся всевозможные информационные сообщения, для нас не очень интересные.



Если Apache все равно не удается запустить, следует загрузить и установить библиотеки времени исполнения Microsoft Visual Studio 2017[2]. Кстати, ссылки на эти файлы можно найти на странице, с которой мы ранее загрузили веб-сервер.

10. Запустим веб-обозреватель и выполним переход по интернет-адресу **http://localhost/1.1.html** — веб-обозреватель загрузит нашу страницу с веб-сервера и выведет на экран. ❯

11. Завершим работу веб-сервера Apache, щелкнув на кнопке закрытия окна, которым он представляется на экране. После щелчка потребуется подождать секунду, пока программа не завершит все свои процессы.



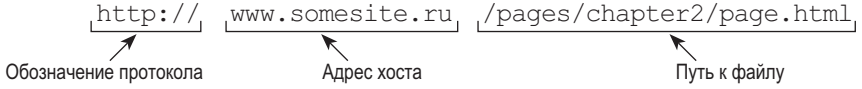
[2] Дистрибутивы этих библиотек находятся по адресам: **https://aka.ms/vs/17/release/VC_redist.x86.exe** (32-разрядная редакция) и **https://aka.ms/vs/17/release/VC_redist.x64.exe** (64-разрядная редакция).

3.3. Интернет-адрес и его составные части

Любой файл, входящий в состав какого-либо опубликованного в Интернете сайта, однозначно идентифицируется по его интернет-адресу.

> ***Интернет-адрес***
>
> Уникальный адрес файла, опубликованного в Интернете.

Интернет-адрес состоит из трех основных частей:



> ***Протокол***
>
> Своего рода «язык», на котором клиент и сервер «общаются» друг с другом. Включает в себя описание форматов клиентских запросов, серверных ответов и набор команд, посылаемых клиентом серверу в составе запросов и предписывающих тому выполнить определенные действия над файлами.
>
> ***Адрес хоста***
>
> Идентифицирует компьютер, который подключен к Интернету (хост) и на котором работает программа сервера. По этому адресу клиент отправляет клиентский запрос на получение файла.
>
> ***Путь к файлу***
>
> Непосредственно указывает на запрашиваемый файл, который хранится на сервере. Посылается клиентом серверу в составе запроса.

Рассмотрим эти составные части более подробно.

3.3.1. Протокол

В WWW применяются два протокола со следующими обозначениями:

- **http://** — HTTP (HyperText Transfer Protocol, протокол передачи гипертекста).

  Пересылает данные в открытом, незашифрованном виде. В настоящее время считается недостаточно защищенным и не рекомендован к применению, однако все еще широко используется.

- **https://** — HTTPS (HTTP Secure, защищенный HTTP).

  Пересылает данные в зашифрованном виде. В настоящее время рекомендован к применению во всех вновь разрабатываемых сайтах и постепенно набирает популярность.

Если обозначение протокола не указано, применяется HTTP.

3.3.2. Адрес хоста

Может быть указан в виде:

- доменного имени.

> ***Доменное имя***
>
> Представляет собой набор слов и аббревиатур, разделенных точками. Как правило, имеет осмысленный вид и поэтому удобно для запоминания.

Примеры доменных имен: **www.somesite.com**, **www.google.ru**, **bhv.ru**;

- IP-адреса.

> ***IP-адрес***
>
> Представляет собой набор цифр, вследствие чего труден для запоминания. Существуют две его версии: IPv4 и IPv6.
>
> ***IPv4***
>
> Состоит из четырех цифр. Может адресовать не более 4 294 967 296 компьютеров, что в настоящее время уже недостаточно.

Примеры: **197.243.34.79**, **203.9.56.193**.

> ***IPv6***
>
> Состоит из восьми шестнадцатеричных цифр. Может адресовать до 5×10^{28} компьютеров.

Пример: **2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d**.

При выполнении *упражнения 3.2*, открывая нашу страницу с веб-сервера, мы обращались к нашему собственному компьютеру — *локальному хосту*.

> ***Локальный хост***
>
> То же самый компьютер, на котором работает программа-клиент.

Для обозначения локального хоста могут быть использованы:

- доменное имя **localhost**;
- IP-адрес версии IPv4 **127.0.0.1**;
- IP-адрес версии IPv6 **0:0:0:0:0:0:0:1**.

Протокол IP, IP-адреса и доменные имена

В основе Интернета лежит низкоуровневый протокол *IP* (Internet Protocol, протокол Интернета). Протоколы HTTP и HTTPS базируются на нем.

➪ Протокол IP для идентификации хостов поддерживает только IP-адреса и не «понимает» доменных имен. Поэтому перед тем, как отправить запрос хосту, веб-обозреватель выясняет, какой IP-адрес связан с его доменным именем. Для этого он пользуется услугами особой интернет-службы под названием *DNS* (Domain Name System, система доменных имен).

3.3.3. Путь к файлу

Все пути к файлам, находящимся в составе сайта, отсчитываются от *корневой папки*.

> ***Корневая папка***
>
> Папка, в которой находятся все файлы и папки, составляющие опубликованный в Сети сайт. В случае веб-сервера Apache это папка C:\apache24\htdocs.

В путях к файлам в коде используются прямые (/), а не обратные (\) слеши.

Если путь к файлу не указан, используется путь / (прямой слеш).

Пути к файлам в HTML-тегах

Выполняя *упражнение 1.5*, в атрибуте `src` тега `<img>` мы указали не интернет-адрес файла, а только путь к нему. Как веб-обозреватель узнает, к какому хосту отправить запрос на получение этого файла?

В таком случае он поступает очень просто: шлет запрос к тому же хосту, с которого была загружена сама страница. В нашем случае это локальный хост **localhost**.

3.3.4. Примеры интернет-адресов и их обработка веб-обозревателем

Интернет-адрес		
Протокол	**Хост**	**Файл**
http://www.somesite.ru/pages/chapter2/main.html		
HTTP	**www.somesite.ru**	main.html из папки pages/chapter2, находящейся в корневой папке

http://localhost/1.1.html		
HTTP	Локальный хост	1.1.html, который хранится непосредственно в корневой папке
localhost/images/sushi.jpg		
HTTP*	Локальный хост	sushi.jpg из папки images, находящейся в корневой папке

* Поскольку в интернет-адресе отсутствует обозначение протокола, будет использован протокол HTTP.

3.4. Упражнение. Веб-страница по умолчанию

Если в пути, записанном в интернет-адресе, отсутствует имя файла (то есть путь указывает на папку), веб-сервер отправит файл *страницы по умолчанию*, хранящийся в этой папке.

> ***Веб-страница по умолчанию***
> Отправляется клиенту в том случае, если путь к файлу, полученный в составе запроса, не содержит имени файла и фактически указывает на папку. Файл веб-страницы по умолчанию должен иметь имя index.html.

1. Переименуем файл нашей страницы 1.1.html, опубликованной на веб-сервере (в папке C:\apache24\htdocs), дав ему имя index.html.
2. Запустим веб-сервер Apache, выполнив двойной щелчок на файле C:\apache24\bin\httpd.exe.
3. В веб-обозревателе выполним обращение по интернет-адресу **http://localhost/**.

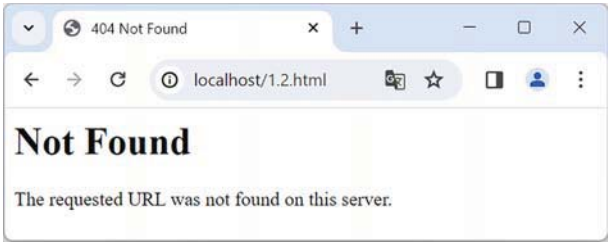
Получив от веб-обозревателя запрос с путем / (прямой слеш), указывающим на корневую папку, веб-сервер отправит находящийся в ней файл страницы по умолчанию — index.html.

> Путь к корневой папке сайта и имя файла страницы по умолчанию могут быть изменены в настройках веб-сервера.

3.5. Ошибка 404

Если веб-сервер получит в составе запроса путь к несуществующему файлу, он отправит запросившему его веб-обозревателю ответ с сообщением об *ошибке 404* («файл не найден»).

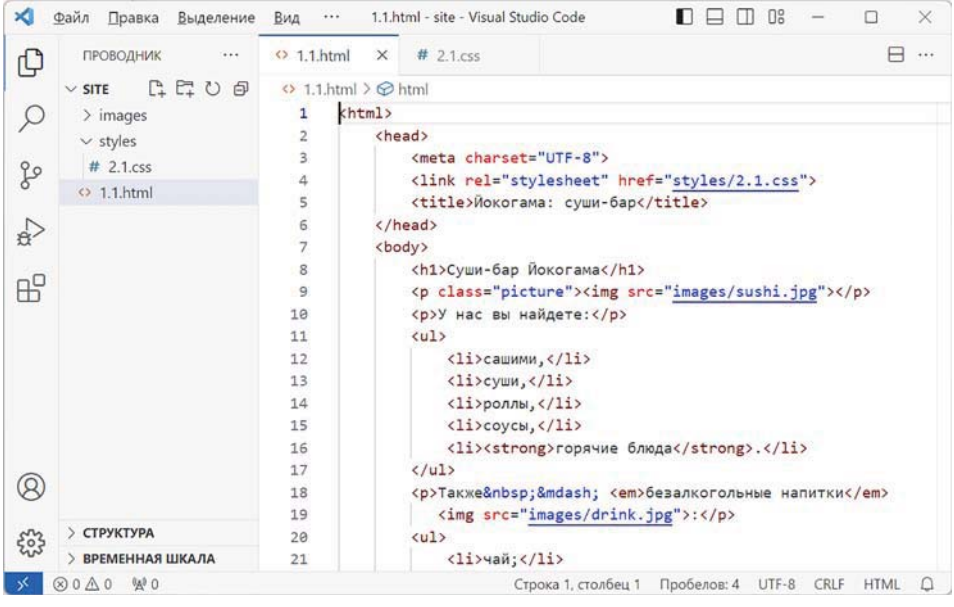
В ответ веб-обозреватель выведет соответствующее сообщение (у разных веб-серверов и веб-обозревателей оно может различаться).



Visual Studio Code

Блокнот — не лучший инструмент для веб-верстки. Он не выполняет ни автоматическое форматирование, ни синтаксическую подсветку кода, имеет довольно бедные средства для поиска и замены и вообще неудобен в работе. А Блокнот Windows 10 вообще позволяет открывать в одном экземпляре программы лишь один файл.

Всех этих недостатков лишен специализированный текстовый редактор для веб-верстальщиков и программистов Visual Studio Code. Его дистрибутивы, как 32-разрядный, так и 64-разрядный, а также документацию можно найти на сайте **https://code.visualstudio.com/**.



Окно текстового редактора Visual Studio Code с двумя документами, открытыми на разных вкладках

ЧАСТЬ II

HTML

- Абзацы, заголовки и списки.
- Семантическая разметка.
- Выделение фрагментов текста.
- Вставка изображений.
- Работа с мультимедиа.
- Создание таблиц.
- Гиперссылки. Связывание веб-страниц друг с другом.
- Веб-формы и элементы управления.

Урок 4. Основные понятия HTML

Правила набора HTML-кода.
Структура HTML-кода.
Указание ссылок на файлы.
Специальные символы HTML.
Комментарии HTML.

HTML — это язык, на котором пишется содержание веб-страниц. На *уроке 1* мы вкратце познакомились с ним. Пришла пора изучить его подробнее.

Практических упражнений на этом уроке не будет. В плане практики мы наверстаем все на последующих уроках.

4.1. Как сохраняются веб-страницы?

HTML-код страницы представляет собой обычный неформатированный текст, аналогичный хранящемуся в обычных текстовых файлах с расширением txt. HTML-код может быть набран в любом текстовом редакторе, предназначенном для работы с неформатированным текстом: Notepad, Блокноте, Visual Studio Code и т. п.

HTML-код практически всегда сохраняется в текстовой кодировке UTF-8. Благодаря этому появляется возможность использовать в тексте страниц любые символы, поддерживаемые этой кодировкой: буквы латиницы, кириллицы, в том числе и с диактрическими знаками, греческого алфавита, математические символы, стрелки, обозначения валют и многое другое.

Файлы с веб-страницами должны иметь расширение html.

Файлы со страницами, предназначенными для публикации в Интернете, должны иметь имена, содержащие только буквы латиницы, цифры, знаки дефиса и подчеркивания. Лишь эти символы допустимы в интернет-адресах.

- У файлов с веб-страницами можно указать расширение htm. Однако оно считается устаревшим.
- HTML-код можно набирать и в текстовых процессорах наподобие Microsoft Word. Однако это неудобно.

4.2. Правила набора HTML-кода

4.2.1. Написание тегов

Тег — это пометка, вставляемая в код страницы и предписывающая веб-обозревателю вывести в этом месте страницы элемент определенного типа. Теги бывают двух видов.

- *Парные теги* включают *открывающий тег*, помечающий начало содержимого создаваемого элемента, *закрывающий тег*, который отмечает конец содержимого, и само *содержимое*. Формат парного HTML-тега:

```
< <имя тега> > <содержимое тега> </ <имя тега> >
```

Открывающий тег — Закрывающий тег

  Открывающий и закрывающий теги должны содержать одинаковое *имя тега* — это станет признаком того, что они являются частью одного парного тега. Закрывающий тег отличается от открывающего лишь наличием символа слеша (/) после открывающей угловой скобки (<).

  Содержимым парного тега может быть текст, другие теги или комбинация текста и других тегов. Ряд парных тегов могут включать в качестве содержимого только другие теги, причем строго определенные (например, тег списка <ul> может включать только теги пунктов <li>). Существуют специфические парные теги, у которых содержимое не указывается.

- *Одинарные теги* записываются в формате:

```
< <имя тега> >
```

Имена тегов допускается набирать в произвольном регистре. Так, следующие три тега выведут три одинаковых адреса:

```
<address>Наш адрес: ул. Зеленая, 17.</address>
<ADDRESS>Наш адрес: ул. Зеленая, 17.</ADDRESS>
<AdDrEsS>Наш адрес: ул. Зеленая, 17.</aDdReSs>
```

Однако, согласно существующим соглашениям, имена тегов всегда набираются в нижнем регистре.

Между именем тега и закрывающей угловой скобкой (>) допускается произвольное количество *пробельных символов*.

> ***Пробельный символ HTML***
>
> Пробел, табуляция или разрыв строки. Веб-обозреватель трактует произвольное количество таких символов как единичный пробел.

Например, следующие три тега выведут три одинаковых абзаца:

```
<p>Ждем вас!</p>
<p      >Ждем вас!</p      >
<p
>Ждем вас!</p
>
```

ВНИМАНИЕ!

Пробельные символы между открывающей угловой скобкой, слешем и именем тега не допускаются. Веб-обозреватель не сможет правильно обработать следующий тег и выведет его в виде обычного текста:

```
<    p>Ждем вас!<    /    p>
```

Не забывайте ставить закрывающие теги!

Пропуск закрывающего тега в парном теге — как, например, здесь (был пропущен закрывающий тег `</h1>`):

```
<h1>Суши-бар Йокогама
<p>У нас вы найдете: сашими, суши, роллы,
   соусы, горячие блюда.</p>
```

может привести к тому, что веб-страница отобразится неправильно.

Поэтому никогда не забывайте ставить закрывающие теги.

Неопытные веб-верстальщики часто забывают поставить в закрывающем теге символ слеша, в результате чего у них получается еще один открывающий тег. Такую ошибку выявить довольно трудно, поэтому лучше проверять HTML-код непосредственно в процессе его набора.

4.2.2. Написание атрибутов тегов

Атрибут тега — это параметр элемента страницы, записанный непосредственно в создающем его теге. Формат написания атрибута тега и его значения:

<имя атрибута тега>="*<значение атрибута тега>*"

Значение атрибута тега берется в двойные кавычки.

Атрибут тега записывается непосредственно в теге после его имени через произвольное количество пробельных символов (пробелов, табуляций или разрывов строки). Так, следующие три тега выведут на страницу три одинаковые изображения:

```
<img src="images/drink.jpg">
<img         src="images/drink.jpg">
<img
     src="images/drink.jpg">
```

В случае парного тега атрибуты следует записывать *в открывающем теге* (тег с атрибутом подчеркнут):

```
<p class="greeting">Ждем вас!</p>
```

В теге можно записать любое количество атрибутов, разделив их произвольным количеством пробельных символов. Следующие три тега полностью идентичны:

```
<link rel="stylesheet" href="styles/2.1.css">
<link      rel="stylesheet"       href="styles/2.1.css">
<link      rel="stylesheet"
           href="styles/2.1.css">
```

Имена атрибутов тегов допускается набирать в произвольном регистре:

```
<p class="greeting">Ждем вас!</p>
<p CLASS="greeting">Ждем вас!</p>
<p cLaSs="greeting">Ждем вас!</p>
```

Однако, согласно существующим соглашениям, имена атрибутов тегов набираются в нижнем регистре.

> ***Атрибут тега без значения***
>
> Атрибут тега, не имеющий значения, записывается указанием лишь его имени. Применяется для включения какого-либо специфического режима отображения элемента.

В качестве примера можно привести атрибут без значения `readonly`, поддерживаемый тегом `<input>`, который создает поле ввода. Этот атрибут предписывает сделать поле ввода доступным только для чтения. Пример:

```
<input readonly>
```

- Как показывает практика, значение атрибута тега можно заключить и в одинарные кавычки:

  ```
  <img src='images/drink.jpg'>
  ```

  Более того, если значение атрибута не содержит пробелов, его можно записать вообще без кавычек:

  ```
  <img src=images/drink.jpg>
  ```

  Однако подобного рода код выглядит неаккуратно.

- В предыдущих версиях HTML имена тегов и их атрибутов набирались в верхнем регистре.

4.2.3. Как веб-обозреватель обрабатывает пробельные символы?

Веб-обозреватель обрабатывает пробельные символы по-разному в зависимости от того, в какой части HTML-кода они присутствуют:

- в обычном текстовом содержимом, между словами — последовательность из произвольного количества пробельных символов при выводе преобразуется в один пробел. Так, следующие три тега отобразят на странице три одинаковых абзаца, содержащие единственный пробел между словами:

  ```
  <p>Ждем вас!</p>
  <p>Ждем              вас!</p>
  <p>Ждем
                     вас!</p>
  ```

  Перенос выводимых строк, если они по ширине не помещаются в окне, веб-обозреватель выполняет самостоятельно;

- между открывающим или закрывающим тегом и содержимым — игнорируются. Следующие три тега создадут три одинаковых абзаца:

  ```
  <p>Ждем вас!</p>
  <p>         Ждем вас!          </p>
  <p>
      Ждем вас!
  </p>
  ```

- между отдельными тегами абзацев, заголовков, списков, адресов и прочих блочных элементов (элементов, выстраиваемых по вертикали друг под другом, подобно абзацам текста; подробности — в *разд. 5.4*) — игнорируются. Например, следующие три набора тегов выведут три одинаковых списка:

```
<ul>
    <li>чай;</li>
    <li>кофе;</li>
    <li>мате.</li>
</ul>
<ul>   <li>чай;</li>   <li>кофе;</li>   <li>мате.</li>   </ul>
<ul><li>чай;</li><li>кофе;</li><li>мате.</li></ul>
```

Благодаря такой особенности HTML, появляется возможность форматировать набираемый HTML-код для повышения его наглядности.

4.2.4. Форматирование HTML-кода при наборе

- Каждый HTML-тег набирается в отдельной строке:

```
<h1>Суши-бар Йокогама</h1>
<p class="picture"><img src="images/sushi.jpg"></p>
<p>У нас вы найдете:</p>
```

- Вложенные теги набираются с отступом слева, который традиционно составляет 2 или 4 пробела[1]:

```
<ul>
    <li>чай;</li>
    <li>кофе;</li>
    <li>мате.</li>
</ul>
```

Теги, вложенные во вложенный тег, набираются с отступом, кратным уровню вложенности:

```
<ul>
    <li>чай;</li>
    <li>кофе:
        <ul>
            <li>мокко;</li>
            <li>эспрессо;</li>
            <li>латте;</li>
```

[1] Автор книги предпочитает делать отступ в 4 пробела.

```
        </ul>
    </li>
    <li>мате.</li>
</ul>
```

- Слишком длинное текстовое содержимое тега разбивается на отдельные строки. Вторая и каждая последующая строки обычно начинаются с колонки, на которой находится первый символ первой строки. Пример:

```
<p class="dishes">У нас вы найдете:
                  сашими, суши, роллы,
                  соусы, горячие блюда.</p>
```

- Если текстовое содержимое имеет очень большую длину и отформатировано с применением других тегов (наподобие `<strong>` и `<em>`), открывающий и закрывающий теги набираются в отдельных строках, а содержимое набирается между ними с отступом слева (как вложенные теги). Пример:

```
<p>
    У нас вы найдете: сашими, суши, роллы, соусы,
    разнообразные <em>горячие блюда<em>.
</p>
```

- Слишком длинные теги разбиваются на отдельные строки. При этом вторая и каждая последующая строка обычно начинается с колонки, на которой находится первый символ первого атрибута тега. Пример:

```
<link rel="stylesheet"
      href="/styles/additional/special-styles.css">
```

- Очень длинные теги разбиваются на строки следующим образом:
  - открывающая угловая скобка и имя тега располагаются на отдельной строке;
  - атрибуты тега — на последующих строках, набранных с отступом слева (подобно вложенным тегам);
  - закрывающая угловая скобка располагается на последней строке без отступа слева.

  Примеры:

```
<link
    rel="stylesheet"
    href="/styles/additional/vendor/very-special-styles.css"
>
```

- Очень короткие теги допускается набирать в одну строку, вплотную:

  ```
  <ul><li>чай;</li><li>кофе;</li><li>мате.</li></ul>
  ```

  Или разделив пробелами — для наглядности:

  ```
  <ul> <li>чай;</li> <li>кофе;</li> <li>мате.</li> </ul>
  ```

  Это сделает HTML-код несколько компактнее.

Следует воздерживаться от вставки в код излишних пробелов и разрывов строк, поскольку они увеличивают объем страницы, замедляют ее пересылку по сети (особенно по медленным каналам связи) и лишь ухудшают читаемость кода.

Неизвестные теги и атрибуты тегов

В HTML-коде допускается использовать теги и атрибуты тегов, не поддерживаемые веб-обозревателем.

Содержимое тега, не поддерживаемого веб-обозревателем (неизвестного ему тега), выводится на странице в виде обычного текста без какого бы то ни было форматирования и выделения.

Неизвестный тег является встроенным элементом (такие элементы выстраиваются друг за другом по горизонтали, подобно словам в тексте, подробности — в *разд. 6.2*).

Так, содержимое неизвестного тега `<sometag>` будет выведено как часть абзаца, в который вложен этот тег:

```
<p>Это <sometag>неизвестный тег</sometag></p>
```

Атрибуты тегов, не поддерживаемые веб-обозревателем (*неизвестные атрибуты тегов*), игнорируются.

Например, неизвестный атрибут `theme`, присутствующий в теге абзаца, будет проигнорирован веб-обозревателем:

```
<p theme="light">Здесь есть неизвестный атрибут тега</p>
```

Неизвестные теги и атрибуты тегов, не поддерживаемые веб-обозревателями, тем не менее, могут обрабатываться другими программами (например, службами онлайн-перевода). Также их можно использовать для того, чтобы применить к их содержимому какой-либо стиль.

4.3. Структура HTML-кода: пролог, секции веб-страницы, метаданные

HTML-код страницы должен быть структурирован особым образом, чтобы веб-обозреватель «понял» его. В коде также следует записать сведения о странице, которые помогут веб-обозревателю вывести ее на экран.

Для структурирования кода и записи сведений о странице применяются особые теги. Рассмотрим их на примере кода нашей первой страницы 1.1.html (код, создающий «видимое» содержание, опущен для краткости):

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Йокогама: суши-бар</title>
    </head>
    <body>
        . . .
    </body>
</html>
```

- `<!doctype>` — одинарный тег, создающий *пролог*.

> ***Пролог***
>
> Обозначение языка, на котором написана веб-страница, и версии этого языка.

  Атрибут без значения `html`, помещенный в тег `<!doctype>`, обозначает язык HTML версии 5.

  Пролог должен располагаться в самом начале HTML-кода, на отдельной строке.

- `<html>` — парный тег, в который помещается весь остальной код страницы.
  - `<head>` — парный тег, создающий *секцию заголовка* страницы.

> ***Секция заголовка***
>
> Содержит метаданные веб-страницы и некоторые служебные теги.
>
> ***Метаданные***
>
> Данные, описывающие саму страницу: кодировку, название и др. Они не выводятся на экран, но используются веб-обозревателем при выводе страницы.

- `<meta>` — одинарный тег, хранящий одну единицу метаданных с техническими сведениями о странице.

  Атрибут `charset` этого тега задает кодировку, в которой сохранена страница (в нашем случае — UTF-8).

  Для указания прочих технических сведений применяются другие атрибуты тега `<meta>`. Они описаны в *разд. 33.1*.

- `<title>` — парный тег, указывающий название страницы, которое выводится на корешке вкладки веб-обозревателя.

В секции заголовка также записываются теги `<link>`, с помощью которых к странице привязываются внешние таблицы стилей.

- `<body>` — парный тег, создающий *секцию тела* страницы.

> ***Секция тела***
>
> Содержит HTML-код, который создает «видимое», непосредственно отображаемое на экране содержание страницы.

Допускается менять местами секции заголовка и тела страницы:

```
<html>
    <body>
        . . .
    </body>
    <head>
        . . .
    </head>
</html>
```

Однако делать так не принято.

4.4. Ссылки на файлы и их указание

Ссылка на файл, записанная, например, в атрибуте `src` тега `<img>` или в атрибуте `href` тега `<link>`, может представлять собой полный интернет-адрес или же только путь к файлу. Какую именно применить ссылку, зависит от того, в состав какого сайта входит файл.

4.4.1. Если файл входит в состав того же веб-сайта, что и сама веб-страница…

Задается путь к файлу. Для получения файла веб-обозреватель отправит запрос тому же веб-серверу, с которого была загружена страница.

Можно указать две разновидности пути:

- *абсолютный* — предваряется символом прямого слеша.

> **Абсолютный путь к файлу**
>
> Отсчитывается от корневой папки сайта. Обрабатывается самим веб-сервером, вследствие чего пересылается ему веб-обозревателем в составе запроса в неизменном виде.

Примеры:

- `<img src="/images/drink.jpg">`

  Указывает на файл drink.jpg из папки images, находящейся в корневой папке сайта;

- `<img src="/logo.png">`

  Указывает на файл logo.png, хранящийся непосредственно в корневой папке;

- `<img src="/images/dishes/sushi/chukka.png">`

  Указывает на файл с путем images/dishes/sushi/chukka.png, отсчитанным от корневой папки;

- *относительный* — *не* предваряется символом прямого слеша.

> **Относительный путь к файлу**
>
> Отсчитывается от папки, в которой хранится файл текущей страницы. Преобразуется веб-обозревателем в абсолютный путь к файлу, который и отправляется веб-серверу.

Пусть файл текущей страницы хранится на веб-сервере по абсолютному пути /pages/sushi/chukka.html. Тогда все относительные пути будут отсчитываться от папки /pages/sushi.

Примеры:

- `<img src="chukka-small.png">`

  Указывает на файл /pages/sushi/chukka-small.png;

- `<img src="additional/chukka-big-1.jpg">`

  Указывает на файл /pages/sushi/additional/chukka-big-1.jpg.

Если нужно обратиться к папке, в которую вложена папка с файлом текущей страницы (к папке предыдущего уровня вложенности), следует указать в начале относительного пути обозначение ../ (две точки и слеш). Пример:

```
<img src="../sushi-thumb.gif">
```

Указывает на файл /pages/sushi-thumb.gif.

Если нужно «подняться» на два или более уровня папок, обозначение ../ повторяется нужное количество раз.

Примеры:

- `<img src="../../logo.png">`

  Указывает на файл /logo.png (то есть находящийся непосредственно в корневой папке):

- `<img src="../../additional/telegram.gif">`

  Указывает на файл /additional/telegram.gif.

Относительный путь следует использовать для ссылок на файлы, хранящиеся в той же папке, что и файл страницы, во вложенной в нее папке или в папке предыдущего уровня вложенности. Во всех прочих случаях удобнее применять абсолютные пути.

4.4.2. Если файл входит в состав другого веб-сайта…

Указывается полный интернет-адрес файла:

```
<img src="http://www.othersite.com/images/img.jpg">
```

Абсолютные пути к файлам и веб-сервер

Если сайт предназначается для публикации на веб-сервере, в HTML-коде (например, в тегах <img>) можно указывать ссылки на файлы всех трех рассмотренных в *разд. 4.4* видов: абсолютные пути, относительные пути и полные интернет-адреса.

Однако если сайт предназначен для просмотра только с локального диска или, как вариант, просматривается в процессе разработки только с локального диска, абсолютные пути к файлам в коде его страниц использовать нельзя.

Абсолютные пути отсчитываются от корневой папки сайта, а их обработку выполняет веб-сервер, которому известно положение этой папки. Но если страницы сайта открываются непосредственно с локального диска, без участия веб-сервера, абсолютные пути не могут быть правильно обработаны. Дело в том, что ни веб-обозреватель, ни операционная система не «знают», где находится корневая папка, от которой следует отсчитывать абсолютные пути.

В результате, скорее всего, такие пути будут обрабатываться как относительные — отсчитываться от папки, в которой хранится файл текущей страницы. При этом помещенные на страницу рисунки не будут выведены на экран, а внешняя таблица стилей не будет применена.

4.5. Атрибуты тегов, поддерживаемые всеми тегами

Существуют атрибуты тегов, которые поддерживаются всеми тегами:

- `class` — задает стилевой класс у тега:

  ```
  <p class="greeting">Ждем вас!</p>
  ```

  Допускается задавать несколько стилевых классов, разделив их пробелами:

  ```
  <p class="greeting very-soon special">Ждем вас!</p>
  ```

- `id` — задает *якорь* (аналог стилевого класса, только указываемый у единственного тега; подробности — в *разд. 9.1.4*):

  ```
  <p id="sushi-list">Суши</p>
  ```

- `name` — задает *наименование* (уникальный идентификатор тега, используемый веб-обозревателем; подробности — в *разд. 9.4*):

  ```
  <map name="main-map"> . . . </map>
  ```

- `style` — задает встроенный стиль (встроенные стили рассматриваются в *разд. 11.2*);

- `lang` — указывает обозначение языка, на котором написано текстовое содержимое тега и его потомков. Русский язык имеет обозначение `ru`, английский — `en`[2]. Примеры:

  ```
  <h1 lang="ru">Суши-бар Йокогама</h1>
  <h1 lang="en">Yokogama sushi bar</h1>
  ```

  Обозначение языка, заданное у родителя, наследуется его потомками. Однако при необходимости можно указать обозначение языка в потомке — и оно перекроет заданное у родителя. Пример:

  ```
  <h1 lang="ru">
      <em>Суши-бар Йокогама</em>
      <em lang="en">(Yokogama sushi bar)</em>
  </h1>
  ```

  Как правило, обозначение языка, на котором набран весь текст страницы или его бóльшая часть, задается у тега `<html>`. А при необходимости вывести текст на другом языке обозначение этого языка указывается у тега, в котором записан иноязычный текст. Пример:

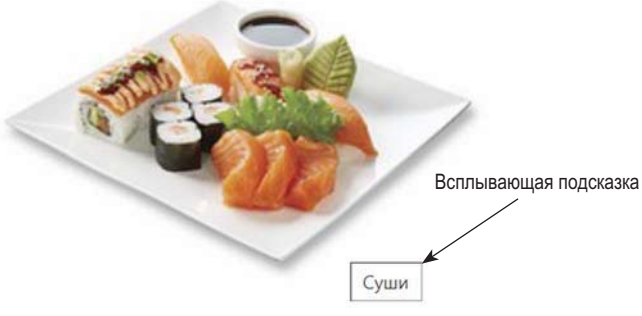
  ```
  <html lang="ru">
      . . .
      <body>
          <h1>Суши-бар Йокогама</h1>
  ```

[2] Полный список обозначений языков приведен на странице **http://programmerbook.ru/html/common-values/lang/**.

```
        <h1 lang="en">Yokogama sushi bar</h1>
        . . .
    </body>
</html>
```

Веб-обозреватель использует обозначение языка при выполнении переноса слов (как организовать такой перенос, рассмотрено в *разд. 14.4.4*).

- `hidden` — предписывает веб-обозревателю не выводить элемент на экран. Может быть использован как:
  - атрибут без значения — *полностью скрывает элемент*, в результате чего он не станет выводиться на странице и будет игнорироваться при выполнении поиска на странице средствами веб-обозревателя:
    ```
    <p hidden>Меня нет!</p>
    ```
  - обычный атрибут с указанием одного из следующих значений:
    - `""` (пустая строка) или `hidden` — полностью скрывает элемент:
      ```
      <p hidden="">Меня нет!</p>
      <p hidden="hidden">Меня тоже нет!</p>
      ```
    - `until-found` — *частично скрывает элемент*, при этом он не выводится на экране, однако принимается во внимание при поиске на странице средствами веб-обозревателя:
      ```
      <p hidden="until-found">Меня не видно, но можно найти.</p>
      ```
- `title` — задает текст всплывающей подсказки, которая выводится на экран при наведении на элемент страницы курсора мыши:
  ```
  <img src="images/sushi.jpg" title="Суши">
  ```



- `translate` — указывает, будет ли содержимое тега переводиться на другой язык службами онлайн-перевода наподобие Google Translate. Поддерживаются следующие значения:
  - «""» (пустая строка) или `yes` — переводить содержимое тега;
  - `no` — не переводить.

Если атрибут тега не указан, содержимое будет переводиться — как если бы этот атрибут был указан со значением `yes`. Говорят, что атрибут тега имеет значение по умолчанию `yes`.

> ***Значение по умолчанию***
>
> Значение атрибута тега, используемое, если этот атрибут не указан в теге.

Пример:

```
<p translate="no">Google Chrome</p>
```

4.6. Недопустимые символы HTML

Текстовое содержимое тегов можно набирать любыми символами, поддерживаемыми используемой кодировкой (например, UTF-8), — за исключением следующих четырех символов, которые задействуются при написании тегов и являются *недопустимыми*:

- `<` (знак «меньше») — открывающая угловая скобка;
- `>` (знак «больше») — закрывающая угловая скобка;
- `"` — прямая двойная кавычка;
- `&` — амперсанд.

Вставка любого такого символа в обычный текст приведет к тому, что веб-обозреватель не сможет правильно обработать такой текст и выведет его с искажениями.

Для вставки недопустимых символов в обычный текст следует применять специальные символы HTML.

4.7. Специальные символы HTML

Специальные символы (или *мнемоники*) — это последовательности символов, применяемые для вставки в текст символов, либо недопустимых в обычном тексте (наподобие `<` и `>`), либо тех, которые нельзя ввести с клавиатуры (в частности, длинного тире, угловых кавычек, букв с диакритическими значками, греческих букв и т. п.).

Список наиболее часто используемых специальных символов приведен в табл. 4.1[3].

[3] Полный список доступных в HTML специальных символов можно найти по адресу **https://symbl.cc/ru/html-entities/**.

Таблица 4.1

Знак	Специальный символ	Знак	Специальный символ	Знак	Специальный символ
Неразрывный пробел		"	"	&	&
<	<	>	>	—	—
«	«	»	»	§	§
“	“	”	”	½	½
…	…	°	°	∞	∞
©	©	®	®	™	™
✓	✓	✗	&cross	≠	≠
×	×	÷	÷	±	±
₽	₽	€	€	¢	¢
♪	♪	∞	∞	Табуляция	

4.8. Комментарии HTML

> ***Комментарий HTML***
>
> Фрагмент HTML-кода, не обрабатываемый веб-обозревателем и не выводящийся на экран. Применяется для внесения в код всевозможных заметок и примечаний.

Формат написания комментария:

```
<!-- <текст комментария> -->
```

Текст комментария может быть разбит на строки и иметь произвольный размер. Между *текстом комментария* и тегами <!-- и --> может присутствовать произвольное количество пробельных символов.

Короткие комментарии набираются в одну строку:

```
<!-- Выровнять этот заголовок по центру страницы -->
<h1>Суши-бар Йокогама</h1>
```

Если комментарий велик, открывающий `<!--` и закрывающий `-->` теги располагают на отдельных строках, а текст комментария указывают между ними с отступом слева:

```
<!--
    Выровнять этот заголовок по центру страницы и поместить под ним
    изображение суши из файла sushi.jpg
-->
<h1>Суши-бар Йокогама</h1>
```

Как еще можно использовать комментарии?

Мы можем поместить в комментарий часть HTML-кода страницы (закомментировать его), чтобы выяснить, как страница отображается без этого фрагмента. Это может помочь при выявлении ошибок в HTML-коде.

Выявив ошибку, мы уберем теги комментария, в который заключили фрагмент (раскомментируем его).

Урок 5. Структура текста

Абзацы и заголовки.
Списки.
Семантическая разметка.
Блочные элементы.
Блочный контейнер.

Любой текст для удобства чтения структурируется — по меньшей мере, разбивается на абзацы и заголовки. Всевозможные перечни оформляют в виде списков, маркированных и нумерованных, а ссылки на другие тексты — в виде цитат. А если текст очень велик, не помешает разделить его на более крупные семантические блоки: шапку, поддон и основное содержимое. Последнее, в свою очередь, имеет смысл разбить на отдельные статьи, а статьи — на отдельные секции.

5.1. Базовые средства структурирования текста

5.1.1. Абзацы и заголовки

Для создания абзацев и заголовков применяются следующие парные теги (первые два нам уже хорошо знакомы):

- `<p>` — абзац;
- `<h1>` — заголовок первого уровня. Им предваряются отдельные материалы: статьи, книги и часто сами веб-страницы. Пример:

```
<h1>Суши-бар Йокогама</h1>
<p>У нас вы найдете: сашими, суши, роллы, соусы,
 горячие блюда.</p>
```

- <h2> — заголовок второго уровня. Предваряет крупные фрагменты материалов — например, части;
- <h3> — заголовок третьего уровня. Более мелкие фрагменты — главы;
- <h4> — заголовок четвертого уровня. Еще более мелкие фрагменты — разделы.

Пример:

```
<h1>HTML и CSS: 33 урока для начинающих</h1>
<h2>Часть II. HTML</h2>
<h3>Урок 5. Структура текста</h3>
<h4>5.1. Базовые средства структурирования текста</h4>
```

- <h5> — заголовок пятого уровня;
- <h6> — заголовок шестого уровня.

Этими заголовками предваряются части параграфов, врезки, цитаты, примечания:

```
<h6>Суши</h6>
<p>
    Блюдо традиционной японской кухни, приготовленное из риса с
    уксусной приправой и различных морепродуктов, а также других
    ингредиентов.
</p>
```

Заголовки первого, второго и третьего уровня веб-обозреватель по умолчанию выводит увеличенным шрифтом, заголовок четвертого уровня — таким же шрифтом, как абзац, а заголовки пятого и шестого уровня — уменьшенным шрифтом. В любом случае для заголовков используется шрифт полужирного начертания. Вот пример:

5.1.2. Списки

5.1.2.1. Маркированные и нумерованные списки

На практике наиболее часто применяются следующие списки:

- *маркированные* (или *неупорядоченные*) — отдельные пункты помечены маркерами;
- *нумерованные* (или *упорядоченные*) — с пронумерованными пунктами.

Для создания списков применяются следующие парные теги:

- `<ul>` — создает маркированный список;
- `<ol>` — создает нумерованный список.

Отдельный пункт любого из этих списков создается парным тегом `<li>`, в который помещается содержимое пункта. А сами теги `<li>` записываются в теге `<ul>` или `<ol>`.

Пример	Результат

`<ul>`
`  <li>Абзацы;</li>`
`  <li>заголовки;</li>`
`  <li>списки.</li>`
`</ul>`
	• Абзацы;
	• заголовки;
	• списки.

`<ol>`
`  <li>Абзацы;</li>`
`  <li>заголовки;</li>`
`  <li>списки.</li>`
`</ol>`
	1. Абзацы;
	2. заголовки;
	3. списки.

По умолчанию пункты списков выводятся с небольшим отступом слева. Маркеры в маркированном списке имеют вид черных кружков, нумерация в нумерованных списках выполняется арабскими цифрами.

Тег `<ol>` поддерживает два атрибута, которые могут быть полезны:

- `start` — номер, с которого начнется нумерация пунктов списка. Значение этого атрибута тега по умолчанию — 1;
- `reversed` — атрибут без значения, предписывает нумеровать пункты списка в обратном порядке.

Пример	Результат

`<ol start="4">`
`  <li>Абзацы;</li>`
`  <li>заголовки;</li>`
`  <li>списки.</li>`
`</ol>`
	4. Абзацы;
	5. заголовки;
	6. списки.

`<ol reversed>`
`  <li>Абзацы;</li>`
`  <li>заголовки;</li>`
`  <li>списки.</li>`
`</ol>`
	3. Абзацы;
	2. заголовки;
	1. списки.

Допускается создавать вложенные списки. Для этого тег, создающий вложенный список, следует поместить между содержимым и закрывающим тегом () того пункта «внешнего» списка, после которого должен располагаться вложенный список.

Пример	Результат
  - Абзацы; - заголовки; - списки: - маркированные; - нумерованные. 	• Абзацы; • заголовки; • списки: ◦ маркированные; ◦ нумерованные.

Можно вкладывать друг в друга списки разных типов: маркированные — в нумерованные и наоборот. Также можно создавать вложенные списки в других вложенных списках.

Пункты вложенных списков по умолчанию выводятся с увеличенным отступом слева. Маркеры у пунктов вложенных списков отличаются от маркеров «внешних» списков (так, в предыдущем примере маркеры у пунктов вложенного списка имеют вид окружностей).

5.1.2.2. Список описаний

> ***Список описаний***
>
> Перечень терминов, за каждым из которых следует разъяснение.

Список описаний создается парным тегом <dl>. Внутри этого тега помещают набор парных тегов: <dt>, создающих сами термины, и <dd>, каждый из которых задает разъяснение термина.

Пример:

```
<dl>
    <dt>HTML</dt>
    <dd>Язык написания веб-страниц</dd>
    <dt>CSS</dt>
    <dd>Язык написания стилей для веб-страниц</dd>
    <dt>Веб-сервер</dt>
```

```
    <dd>Программа, пересылающая веб-страницы веб-обозревателю
        по сети</dd>
</dl>
```

✓ Результат

HTML
    Язык написания веб-страниц
CSS
    Язык написания стилей для веб-страниц
Веб-сервер
    Программа, пересылающая веб-страницы веб-обозревателю по сети

Текст терминов и разъяснений по умолчанию выводится тем же шрифтом, что и текст абзацев. Разъяснения выводятся с отступом слева.

5.1.3. Адрес и блочная цитата

Еще пара полезных тегов:

- `<address>` — адрес. Применяется для вывода почтовых адресов, адресов электронной почты, а также имен авторов статей.

  Содержимое этого тега выводится *курсивом*.

Пример	Результат

`<address>`
`  Наш адрес: ул. Зеленая, 17.`
`</address>`
	Наш адрес: ул. Зеленая, 17.

- `<blockquote>` — *блочная цитата*. Применяется для вывода больших цитат, взятых из других источников.

  Содержимое блочной цитаты выводится с отступом слева.

  Пример:

```
<p>В нашем суши-баре вы можете приобрести
   замечательные суши...</p>
<blockquote>
   Суши - блюдо традиционной японской кухни, приготовленное из риса
   с уксусной приправой и различных морепродуктов, а также других
   ингредиентов.
</blockquote>
<p>...а также сашими...</p>
```

✓ Результат ›

В нашем суши-баре вы можете приобрести замечательные суши...

> Суши - блюдо традиционной японской кухни, приготовленное из риса с уксусной приправой и различных морепродуктов, а также других ингредиентов.

...а также сашими...

Текст блочной цитаты может быть структурирован с применением абзацев и заголовков. Пример:

```
<p>...а также сашими...</p>
<blockquote>
    <h6>Сашими</h6>
    <p>Блюдо национальной японской кухни.</p>
    <p>Готовится из филе разнообразных сортов рыб, других
       морепродуктов и мяса, нарезанного на небольшие кусочки.</p>
</blockquote>
<p>...и многое другое.</p>
```

✓ Результат ›

...а также сашими...

> **Сашими**
>
> Блюдо национальной японской кухни.
>
> Готовится из филе разнообразных сортов рыб, других морепродуктов и мяса, нарезанного на небольшие кусочки.

...и многое другое.

5.2. Средства семантической разметки

HTML предоставляет ряд тегов, с помощью которых можно разбить содержание страницы на семантические части: шапку, поддон, основное содержимое, отдельная статья, секция статьи и др. Все эти теги являются парными:

- `<header>` — шапка;
- `<footer>` — поддон.

  Обычно шапка и поддон делаются одинаковыми на всех страницах сайта;
- `<main>` — основное содержимое, уникальное для страницы.

Пример:

```
<header>
    <h1>Суши-бар Йокогама</h1>
</header>
<main>
    <p>У нас вы найдете:</p>
    . . .
    <p>Ждем вас!</p>
</main>
<footer>
    <address>Наш адрес: ул. Зеленая, 17.</address>
</footer>
```

- `<article>` — законченный фрагмент содержимого: отдельная статья, публикация в форуме или блоге, сведения об отдельном товаре и т. п.:

```
<main>
    <article>
        <h2>Чукка</h2>
        <p>Диетические суши, приготовляемые из одноименного салата,
           риса, кунжута и водорослей нори.</p>
    </article>
    <article>
        <h2>Магуро</h2>
        <p>Классические суши. Готовятся из тунца и риса.</p>
    </article>
    . . .
</main>
```

Если основное содержимое включает лишь один законченный фрагмент (например, одну статью), для его размещения можно использовать только тег `<article>`, а тег `<main>` — убрать:

```
<header>
    <h1>Суши-бар Йокогама</h1>
</header>
<article>
    <h2>Чукка</h2>
    <p>Диетические суши, приготовляемые из одноименного салата,
       риса, кунжута и водорослей нори.</p>
</article>
<footer>
    <address>Наш адрес: ул. Зеленая, 17.</address>
</footer>
```

Впрочем, законченный фрагмент можно поместить и в тег <main>:

```
<main>
    <h2>Чукка</h2>
    <p>Диетические суши, приготовляемые из одноименного салата,
       риса, кунжута и водорослей нори.</p>
</main>
```

- <section> — часть большого фрагмента содержимого, например, глава большой статьи:

```
<article>
    <section class="dish-title">
        <h2>Чукка</h2>
    </section>
    <section class="dish-description">
        <p>Диетические суши, приготовляемые из одноименного салата,
           риса, кунжута и водорослей нори.</p>
    </section>
</article>
```

- <aside> — врезка с примечанием, графической иллюстрацией, рекламным объявлением и др.:

```
<h1>Суши-бар Йокогама</h1>
<p><img src="images/sushi.jpg"></p>
<aside>
    <h6>Рекламная акция!</h6>
    <p>Купите три суши и получите четвертое в подарок!</p>
</aside>
```

Как правило, врезка, созданная этим тегом, средствами CSS сдвигается к левому и правому краю страницы (как это сделать, описано в *разд. 22.1*);

- <hgroup> — заголовок с подзаголовками:

```
<hgroup>
    <h1>HTML и CSS. 33 урока для начинающих</h1>
    <p>Автор: Владимир Дронов</p>
</hgroup>
<h2>Предисловие</h2>
. . .
```

Тег весьма специфический и применяется редко.

По умолчанию семантические элементы при выводе никак не выделяются. Однако к ним можно применить стиль с необходимым оформлением.

Еще пару тегов семантической разметки мы изучим позднее.

5.3. Специализированные средства разметки

HTML также поддерживает два тега, создающие специфические элементы страниц, применяемые в особых случаях.

5.3.1. Текст фиксированного формата

> ***Текст фиксированного формата***
>
> Выводится в том виде, в каком он набран в HTML-коде: с сохранением всех последовательностей пробелов, символов табуляции и разрывов строк.

Текст фиксированного формата создается парным тегом `<pre>`.

Его содержимое выводится моноширинным шрифтом.

Пример	Результат

`<pre>li {`
`  font-size: 14pt;`
`  list-style-type: square;`
`}</pre>`
	`li {`
	`  font-size: 14pt;`
	`  list-style-type: square;`
	`}`

Такой элемент страницы часто применяется для вывода HTML-, CSS-кода, исходных текстов программ и пр.

5.3.2. Разделитель

> ***Разделитель***
>
> Имеет вид горизонтальной линии и служит для отделения одного элемента страницы от другого.

Создается одинарным тегом `<hr>`.

Пример:

```
<p>Подаваемые напитки: чай, кофе, мате.</p>
<hr>
<p>Ждем вас!</p>
```

✓ Результат

Подаваемые напитки: чай, кофе, мате.

Ждем вас!

5.4. Блочные элементы

Элементы страниц, создаваемые рассмотренными нами к этому моменту тегами, являются *блочными*.

> ***Блочный элемент веб-страницы***
>
> Элемент страницы, ведущий себя как абзац текста.
>
> Блочные элементы располагаются по вертикали, друг под другом. Содержимое блочного элемента выводится с новой строки.

Блочными являются абзац, заголовки всех уровней, все списки, пункты списков, адрес, блочная цитата, элементы семантической разметки, текст фиксированного формата, а также разделитель (хоть он и не содержит текста).

Для блочных элементов можно задать выравнивание текста, размеры, местоположение, просветы, рамки и многие другие параметры. Средства CSS, применяемые для этого, мы рассмотрим на следующих уроках.

5.5. Блочный контейнер

HTML-теги, рассмотренные в *разд. 5.2*, применяются, если какой-либо фрагмент содержания страницы необходимо преобразовать в семантическую часть: шапку, поддон или врезку. Если же требуется просто объединить несколько элементов страниц с целью применить к ним стиль (например, задающий у всех этих элементов одинаковое выравнивание), следует применить блочный контейнер.

> ***Блочный контейнер (или блок)***
>
> Элемент страницы, предназначенный для объединения нескольких блочных элементов (обычно с целью применить к ним какой-либо стиль).

Блочный контейнер создается парным тегом `<div>`.

Пример:

```
<div>
    <h1>Суши-бар Йокогама</h1>
    <p><img src="images/sushi.jpg"></p>
</div>
```

По умолчанию блочный контейнер и его содержимое никак не выделяются при выводе на экран.

Блочный контейнер можно использовать и для вывода текста, так же, как и обычный абзац. Но нужно иметь в виду, что блок выводится без просветов сверху и снизу. В результате фрагменты текста, выведенные соседними блоками, будут располагаться вплотную друг к другу.

Пример:

```
<div>Добро пожаловать в наш суши-бар!</div>
<div>Участвуйте в нашей рекламной акции!</div>
<div>Акция действует только в выходные.</div>
```

✓ Результат

Добро пожаловать в наш суши-бар!
Участвуйте в нашей рекламной акции!
Акция действует только в выходные.

Поэтому в таких случаях к блокам применяется какой-либо стиль для улучшения читаемости текста.

5.6. Упражнение. Доработка веб-страницы

Сделаем следующее:

- разобьем содержание единственной страницы нашего сайта на семантические шапку и основное содержимое;
- добавим сведения о рекламной акции, представленные в виде нумерованного списка.

1. Найдем в папке 3\s3.4 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   В семантическую шапку поместим заголовок первого уровня и находящийся под ним абзац с изображением суши. Остальное содержание страницы оформим как семантическое основное содержимое.

2. Откроем файл index.html в текстовом редакторе и добавим необходимые теги семантической разметки:

```
<header>
    <h1>Суши-бар Йокогама</h1>
    <p class="picture"><img src="images/sushi.jpg"></p>
</header>
<main>
    <p>У нас вы найдете:</p>
    . . .
    <address>Наш адрес: ул. Зеленая, 17.</address>
</main>
```

   Как говорилось в *разд. 5.2*, теги семантической разметки никак не выделяются на странице визуально. Поэтому внешний вид нашей страницы не изменится.

   Сведения о рекламной акции поместим внизу страницы, между вторым списком и последним абзацем («Ждем вас!»). Они будут представлять собой вводный абзац и нумерованный список из трех пунктов, перечисляющий действия, которые необходимо выполнить для участия в акции.

3. Добавим необходимый код:

```
<ul>
    <li>чай;</li>
    <li>кофе;</li>
    <li>мате.</li>
</ul>
```

```
<p>Участвуйте в нашей рекламной акции!</p>
<ol>
    <li>Посетите наш суши-бар между 16:00 и 19:00.</li>
</ol>
<p class="greeting">Ждем вас!</p>
```

Стоп! Мы совсем забыли, что указанное время действительно только для будних дней. Нужно добавить соответствующее предупреждение.

Это предупреждение запишем в виде двух абзацев сразу под созданным ранее нумерованным списком из одного пункта.

4. Добавим требуемый код:

```
<ol>
    <li>Посетите наш суши-бар между 16:00 и 19:00.</li>
</ol>
<p>Время действительно только для будних дней.</p>
<p>В выходные &mdash; с 14:00 до 20:00.</p>
<p class="greeting">Ждем вас!</p>
```

Для вставки неразрывного пробела и длинного тире мы применили специальные символы.

Нужно добавить еще один нумерованный список — с двумя пунктами, содержащими два оставшихся действия. И указать, чтобы нумерация пунктов в этом списке начиналась с 2.

5. Запишем код, создающий этот список:

```
<p>В выходные &mdash; с 14:00 до 20:00.</p>
<ol start="2">
    <li>Купите три суши или ролла.</li>
    <li>Получите четвертый в подарок!</li>
</ol>
<p class="greeting">Ждем вас!</p>
```

✓ Что получилось? ⌄

Участвуйте в нашей рекламной акции!

1. Посетите наш суши-бар между 16:00 и 19:00.

Время действительно только для будних дней.

В выходные — с 14:00 до 20:00.

2. Купите три суши или ролла.
3. Получите четвертый в подарок!

Все хорошо, только картину портят два абзаца с примечаниями по поводу времени. Заключим их в блочный контейнер со стилевым классом `advertising` и напишем для этого блока стиль, который укажет просвет слева подходящей величины.

6. Вставим HTML-код, который заключит эти абзацы в блок:

```
<div class="advertising">
    <p>Время действительно только для будних дней.</p>
    <p>В выходные &mdash; с 14:00 до 20:00.</p>
</div>
```

7. Откроем таблицу стилей styles/2.1.css и добавим стиль, который укажет у блока просвет слева:

```
.advertising { margin-left :30pt; }
```

Для задания просвета слева применяется атрибут стиля `margin-left`. Значение просвета 30 пунктов подобрано экспериментально, чтобы текст абзацев выводился с тем же отступом, что и пункты списка.

✓ Результат ⌄

1. Посетите наш суши-бар между 16:00 и 19:00.

   Время действительно только для будних дней.

   В выходные — с 14:00 до 20:00.

2. Купите три суши или ролла.

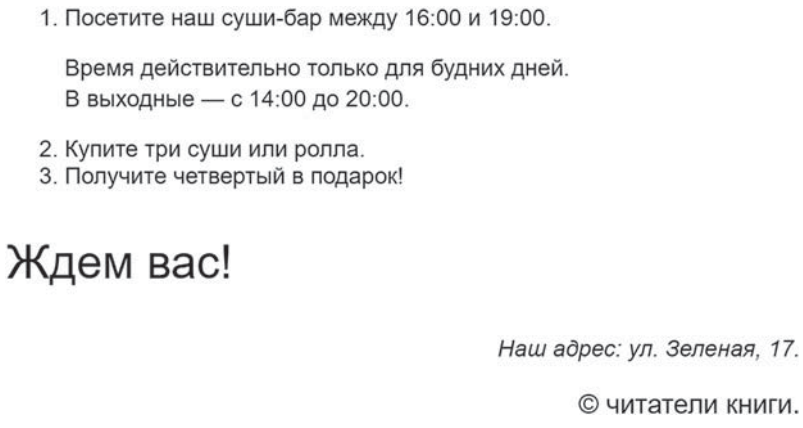
5.7. Самостоятельные упражнения

- Задайте у семантической шапки (тега `<header>`) выравнивание текста по центру.
- Удалите из ранее написанного стиля с селектором `h1` атрибут стиля `text-align`, который там более не нужен.
- Удалите стиль с селектором `.picture`, в котором присутствует только атрибут стиля `text-align` и который также теперь не нужен.
- Добавьте на страницу семантический поддон (тег `<footer>`), содержащий абзац с авторскими правами разработчиков. Также перенесите в этот поддон адрес суши-бара.
- Укажите у семантического поддона выравнивание текста по правому краю. Удалите не нужный более стиль с селектором `address`.

- Задайте у абзацев с примечаниями, касающимися времени проведения акции, кегль 12 пунктов и просветы сверху и внизу в 2 пункта. Для указания просветов сверху и снизу используйте атрибуты стиля `margin-top` и `margin-bottom` соответственно.

✓ У вас должно получиться ⌄

Показана нижняя часть страницы (остальное не изменится).



Урок 6. Текст

Выделение фрагментов текста.
Встроенные элементы.
Встроенный контейнер.
Переносы строк.

HTML предоставляет ряд тегов для выделения фрагментов текста с целью придать им особое значение (например, пометить текст как важный или как короткую цитату). Также имеются инструменты для указания переносов строк.

6.1. Выделение фрагментов текста

Для выделения фрагментов текста, помещенного в блочные элементы (абзацы, заголовки, списки и др.), с целью придать им особое значение язык HTML предусматривает следующие парные теги:

- `<strong>` — очень важный текст. Выводится **полужирным** шрифтом;
- `<em>` — важный текст. Выводится *курсивом*;
- `<mark>` — текст, достойный внимания, но не настолько важный, как взятый в теги `<strong>` и `<em>`. Выводится на желтом фоне:
- `<sup>` — верхний индекс:

Пример	Результат
`a<sup>2</sup> + b<sup>3</sup>`	$a^2 + b^3$

- `<sub>` — нижний индекс:

Пример	Результат
`<p>H<sub>2</sub>O</p>`	H_2O

- `<q>` — короткая цитата. Выводится «в кавычках»;
- `<cite>` — название чего-либо (книги, картины, музыкального произведения, фильма и т. п.). Выводится *курсивом*;
- `<dfn>` — первое упоминание в тексте какого-либо термина. Выводится *курсивом*.

  Как правило, в этом теге указывается атрибут `title` (описан в *разд. 4.5*), задающий определение термина:

```
<p>
    Файл с
    <dfn title="Описание оформления веб-страницы">
        таблицей стилей
    </dfn>
    имеет расширение css.
</p>
```

  Всплывающая подсказка с определением появляется при наведении на термин курсора мыши;

- `<abbr>` — сокращение, аббревиатура. Выводится обычным шрифтом.

  Обычно в этом теге указывается атрибут `title` (см. *разд. 4.5*), задающий расшифровку сокращения:

```
<p>Веб-страницы пишутся на языке
   <abbr title="HyperText Markup Language">HTML</abbr></p>
```

- `<code>` — короткий фрагмент HTML-, CSS-кода, кода компьютерной программы и др. Выводится моноширинным шрифтом;
- `<kbd>` — текст, набираемый пользователем на клавиатуре[1]. Выводится моноширинным шрифтом;
- `<samp>` — результат, выводимый компьютерной программой[2]. Выводится моноширинным шрифтом;
- `<var>` — имя переменной в коде компьютерной программы. Выводится *курсивом*;
- `<ins>` — текст, добавленный в содержание веб-страницы в процессе редактуры. Выводится подчеркнутым;
- `<del>` — текст, подлежащий удалению из окончательной редакции веб-страницы. Выводится ~~зачеркнутым~~;
- `<s>` — текст, помеченный в процессе редактуры как некорректный и требующий замены. Выводится ~~зачеркнутым~~.

---

[1] Компьютерщики называют его *клавиатурным вводом*, или просто *вводом*.

[2] Его также называют *экранным выводом*, или просто *выводом*.

6.1.1. Устаревшие средства для выделения текста

Существует ряд тегов для выделения текста, оставшихся в «наследство» от старых версий HTML, но поддерживаемых до сих пор и часто встречающихся в старом коде:

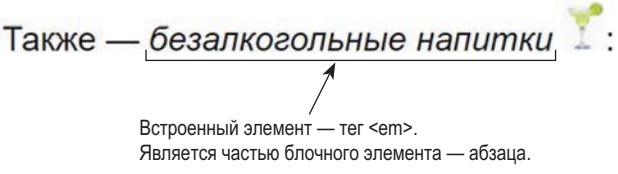
- `<b>` — **полужирный** текст;
- `<i>` — *курсивный* текст;
- `<small>` — текст с уменьшенным кеглем;
- `<u>` — подчеркнутый текст.

В настоящее время эти теги применяются редко и в крайне специфических случаях.

6.2. Встроенные элементы

Все элементы страницы, создаваемые рассмотренными в *разд. 5.1* тегами, относятся к *встроенным*.

> ***Встроенный элемент веб-страницы***
>
> Элемент страницы, ведущий себя как отдельное слово текста.
>
> Встроенные элементы выстраиваются по горизонтали, друг за другом. Если веб-обозревателю не хватает места для вывода всех встроенных элементов в одну строку, он выполняет перенос строк.
>
> Встроенные элементы всегда помещаются в блочные элементы.



Кроме того, встроенными элементами являются все неизвестные теги (см. врезку в *разд. 4.2.4*).

В отличие от блочных, для встроенных элементов невозможно задать рамку, просветы, размеры и местоположение[3]. Мы можем лишь указать у них шрифт и очень ограниченный набор других параметров.

[3] Указать-то можно, но веб-обозреватель их проигнорирует.

6.3. Встроенный контейнер

HTML-теги, рассмотренные в *разд. 6.1*, используются, если требуется дать какому-либо фрагменту текста особое значение. Если же надо просто выделить часть текста с целью применить к ней стиль (например, указывающий цвет шрифта), следует задействовать встроенный контейнер.

> ***Встроенный контейнер***
>
> Элемент страницы, предназначенный для выделения произвольного фрагмента текста (обычно с целью применить к нему какой-либо стиль).

Встроенный контейнер создается парным тегом `<span>`.

Пример:

```
<p>Участвуйте в нашей <span>рекламной акции</span>!</p>
```

По умолчанию встроенный контейнер и его содержимое никак не выделяются при выводе на экран.

6.4. Переносы строк

Два следующих одинарных тега управляют переносом строк:

- `<br>` — указывает веб-обозревателю всегда выполнять перенос (*разрыв*) строки в том месте, в котором находится.

  Пример:

  ```
  <p>Наш адрес:<br>ул. Зеленая, 17.</p>
  ```

  Результат (в месте, где поставлен тег `<br>`, веб-обозреватель всегда выполняет перенос строки):

  Наш адрес:
  ул. Зеленая, 17.

- `<wbr>` — помечает место, в котором веб-обозреватель при необходимости может выполнить перенос строки. Обычно ставится в очень длинных словах.

  Пример:

  ```
  <p>Суши нигири<wbr>дзуси</p>
  ```

  Результат:

  - если веб-обозревателю хватит места для вывода текста в одну строку, перенос выполнен не будет:

    Суши нигиридзуси

- если же места не хватит, будет выполнен перенос строки в том месте, где поставлен тег <wbr>:

  Суши нигири
  дзуси

  К сожалению, знак переноса — дефис — веб-обозреватель при этом не ставит.

Оба этих тега также относятся к встроенным элементам, хотя и не содержат никакого текста.

6.4.1. Специальный символ «перенос строки»

В качестве альтернативы тегу <wbr> можно использовать специальный символ «перенос строки» . Он ведет себя так же, как и тег <wbr>, только в месте переноса строки ставится дефис.

Пример:

```
<p>Суши нигири&shy;дзуси</p>
```

Результат:

Суши нигири-
дзуси

6.5. Упражнение. Доработка веб-страницы

Разделим заголовок нашей страницы на две строки: «Суши-бар» и «Йокогама». Дополнительно укажем у названия суши-бара кегль 60 пунктов и форму строчных букв, схожую с формой прописных букв. Так название станет более заметным.

1. Найдем в папке 5\s5.7 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Для разделения строки заголовка применим тег переноса <br>.

2. Откроем страницу index.html в текстовом редакторе и соответственно исправим код:

   ```
   <h1>Суши-бар Йокогама</h1>
   <h1>Суши-бар<br>Йокогама</h1>
   ```

   Чтобы применить стиль к названию суши-бара, присутствующему в строке заголовка, название необходимо заключить во встроенный контейнер (тег) и указать у последнего какой-либо стилевой класс — например, bar-title.

3. Добавим тег встроенного контейнера:

```
<h1>Суши-бар<br>Йокогама</h1>
<h1>Суши-бар<br><span class="bar-title">Йокогама</span></h1>
```

4. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и добавим стиль, задающий оформление у названия суши-бара:

```
.bar-title {
    font-variant-caps: small-caps;
    font-size: 60pt;
}
```

Для указания формы строчных букв служит атрибут стиля `font-variant-caps`. Значение `small-caps` задает у них форму, схожую с формой прописных букв.

✓ Результат ⌄

6.6. Самостоятельное упражнение

Укажите у всех четырех значений времени, присутствующих на странице, начертание цифр в «старом» стиле и увеличенный кегль. Для этого заключите каждое значение времени во встроенный контейнер с каким-либо стилевым классом и напишите соответствующий стиль. Для задания нужного начертания цифр используйте атрибут стиля `font-variant-numeric` со значением `oldstyle-nus`, а для задания увеличенного кегля — атрибут стиля `font-size` со значением `larger`.

✓ У вас должно получиться ⌄

Обратите внимание на то, как выглядят цифры в значениях времени.

1. Посетите наш суши-бар между 16:00 и 19:00.

Время действительно только для будних дней.
В выходные — с 14:00 до 20:00.

Урок 7. Графика и мультимедиа

Вставка изображений.
Вставка аудио и видео.
Форматы графики, аудио и видео.
Встроенно-блочные элементы.
Семантическая иллюстрация.
Вставка видео с YouTube.

7.1. Графика

7.1.1. Вставка графических изображений

Графическое изображение помещается на веб-страницу с помощью знакомого нам одинарного тега `<img>`.

Этот тег поддерживает следующие атрибуты:

- `src` — ссылка на файл, в котором хранится изображение. Обязателен к указанию;
- `alt` — *текст замены*.

> ***Текст замены***
>
> Отображается в том месте на странице, где должно находиться изображение, пока содержащий его файл загружается.

Пример:

```
<img src="images/logo.jpg" alt="Логотип">
```

Если текст замены не указан, ничего выводиться не будет.

Атрибут тега `alt` имеет смысл указывать только у больших изображений, хранящихся в объемных файлах, которые загружаются достаточно долго;

- `width` и `height` — соответственно, ширина и высота изображения при выводе на экран. Задаются положительными целыми числами и измеряются в пикселах. Пример:

```
<img src="images/logo.jpg" width="300" height="200">
```

Если атрибуты ширины и высоты не заданы, выведенное изображение будет иметь изначальные размеры.

Можно указать лишь один из размеров — ширину или высоту. В этом случае другой размер будет вычислен веб-обозревателем на основании соотношения сторон изображения. Пример:

```
<img src="images/logo.jpg" width="300">
```

Эти атрибуты тега применяются при вставке больших изображений, чтобы веб-обозреватель сразу же зарезервировал для них место на странице. В таком случае изображение после загрузки будет выведено на экран быстрее;

- `loading` — указывает, когда веб-обозревателю следует загрузить файл с изображением. Можно задать одно из следующих значений:
  - `eager` — загрузить файл немедленно, даже если позиция, в которой выводится изображение, в текущий момент не присутствует на экране (например, находится за пределами части страницы, показываемой посетителю);
  - `lazy` — загрузить файл только тогда, когда изображение должно быть выведено (когда посетитель, прокручивая страницу, приближается к той ее части, где расположено текущее изображение).

Значение по умолчанию: `eager`.

Если страница велика и содержит много больших изображений, у тех из них, которые посетитель сразу после загрузки страницы однозначно не увидит, можно указать атрибут тега `loading` со значением `lazy` — это значительно ускорит первоначальное отображение страницы.

Пример:

```
<img src="images/illustrations/1.jpg">
. . .
<img src="images/illustrations/2.jpg" loading="lazy">
. . .
<img src="images/illustrations/3.jpg" loading="lazy">
```

- `decoding` — указывает, когда веб-обозревателю следует считать изображение из загруженного файла и вывести на экран. Можно указать одно из следующих значений:
  - `sync` — считать и вывести изображение немедленно;
  - `async` — отложить считывание и вывод до того, как будет выведено все остальное содержание страницы;
  - `auto` — веб-обозреватель сам решит, когда вывести изображение.

  Значение по умолчанию: `auto`.

  Пример:

```
<img src="images/very-big-picture.jpg" decoding="async">
```

  Указывать значение `async` следует у изображений больших и не очень важных. Если изображение невелико и является важным, у него имеет смысл указать значение `sync`.

7.1.2. Форматы графики, применяемые в WWW

В WWW применяются семь форматов графических файлов. Шесть форматов рассчитаны на хранение растровых изображений, и один — векторных.

7.1.2.1. Растровые форматы

> ***Растровое изображение***
>
> Представляется в виде растра — набора пикселов.

В растровом виде можно сохранить практически любое изображение: сканированную фотографию, картину, схему, чертеж, декоративный элемент оформления и пр. Однако при выводе растровой графики с увеличенным масштабом теряется качество — изображение становится зернистым.

Все растровые форматы (табл. 7.1) хранят изображение в сжатом виде ради уменьшения объема файла.

Таблица 7.1

Описание формата	Преимущества	Недостатки
GIF, Graphics Interchange Format — формат для обмена графикой. В этом формате обычно сохраняется штриховая графика, декоративные элементы и короткие анимационные ролики. Расширение файлов: gif	• Реализует сжатие без потерь качества. • Хорошо подходит для хранения штриховой графики (чертежей, схем, диаграмм). • Изображение может включать прозрачные области. • Позволяет хранить анимацию.	• Изображение может содержать не более 256 цветов. • Плохо подходит для хранения сканированных изображений.

Таблица 7.1 *(окончание)*

Описание формата	Преимущества	Недостатки
JPEG, Joint Photographic Experts Group — объединенная группа экспертов по фотографии. Применяется для хранения сканированной графики (фотографий, картин и т. п.). Расширения файлов: jpeg, jpg, jpe, jfif (последние два применяются редко)	• Использует эффективные алгоритмы сжатия, в результате чего объем файла существенно уменьшается. • Отлично подходит для хранения высококачественной сканированной графики.	• При сильном сжатии часть информации об изображении теряется, что приводит к ухудшению его качества. • Изображение не может содержать прозрачные области. • Не позволяет хранить анимацию.
PNG, Portable network graphics — переносимая сетевая графика. Хорошо подходит для сохранения и штриховых, и сканированных изображений. Расширение файлов: png	• Совмещает в себе преимущества форматов GIF и JPEG, в частности, позволяет хранить сканированную графику, в том числе с прозрачными областями, и реализует эффективное сжатие без потерь качества.	• Не поддерживает хранение анимации.
WebP, Web Picture — веб-изображение. Подходит для хранения графики любого вида. Расширение файлов: webp	• Использует алгоритмы, сжимающие графику на 25–35% эффективнее, нежели JPEG и PNG. • Позволяет хранить анимацию.	• У штриховой графики в некоторых случаях может быть потеря четкости.
AVIF, AV1 Image File Format — формат файлов изображений, сжатых с помощью алгоритма AV1. Подходит для хранения любого вида графики. Расширение файлов: avif	• Использует алгоритм AV1, сжимающий графику на 50% эффективнее, чем WebP. • Позволяет хранить анимацию. • Позволяет хранить изображения с большим диапазоном цветов.	• Требует много системных ресурсов для считывания и вывода. • Ограниченная в настоящее время поддержка веб-обозревателями (в частности, Firefox не поддерживает анимацию этого формата).
APNG, Animated PNG — анимированный PNG. Позволяет сохранять графику любого типа и анимацию. Расширения файлов: png и apng	• Позволяет хранить анимацию. • Щадящие требования к системным ресурсам по сравнению с AVIF.	• Формат менее развитый, нежели WebP и AVIF. • Ограниченный выбор программ для создания изображений в этом формате.

7.1.2.2. Векторный формат

> ***Векторное изображение***
>
> Изображение, представленное в виде набора отдельных геометрических фигур — примитивов, в качестве которых обычно выступают прямые и кривые линии.

Векторная графика (табл. 7.2) прекрасно масштабируется до любых размеров без потерь качества. Простые векторные изображения имеют меньший объем, чем такие же растровые. Однако высококачественную сканированную графику (картины, фото) преобразовать в векторное представление без ухудшения качества практически невозможно.

Таблица 7.2

Описание формата	Преимущества	Недостаток
SVG, Scalable Vector Graphics — масштабируемая векторная графика. Удобен для хранения штриховых изображений и декоративных элементов. Расширения файлов: svg (несжатое изображение) и svgz (сжатое)	• Позволяет хранить графику с прозрачными областями. • Поддерживает анимацию. • Изображение может включать фрагменты растровой графики в указанных ранее форматах (см. табл. 7.1). • Описание изображения сохраняется в обычном текстовом файле, который можно править в любом текстовом редакторе	• Очень большой объем массива данных, представляющих изображение

7.2. Встроенно-блочные элементы

Графическое изображение, помещенное на страницу тегом `<img>`, относится к *встроенно-блочным элементам*.

> ***Встроенно-блочный элемент веб-страницы***
>
> Ведет себя как встроенный элемент (то есть как отдельное слово текста). Однако, как и у блочных элементов, мы можем задать у него средствами CSS размеры, просветы и рамку (но не местоположение).

7.3. Аудио и видео

7.3.1. Вставка аудио- и видеороликов

Для размещения на страницах аудио- и видеороликов применяются парные теги `<audio>` и `<video>` соответственно. Эти теги всегда записываются пустыми (не имеющими содержимого).

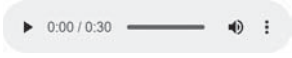
Для указания ссылки на мультимедийный файл и прочих параметров служат следующие атрибуты, поддерживаемые обоими этими тегами:

- `src` — ссылка на файл с аудио- или видеороликом:

  ```
  <audio src="media/very-cool-song.mp3"></audio>
  ```

- `controls` — атрибут без значения, предписывающий вывести набор элементов для управления воспроизведением ролика.

  Этот набор включает указатель позиции воспроизведения, кнопку для запуска и приостановки воспроизведения, регулятор громкости, кнопку приглушения звука и кнопку для переключения в полноэкранный режим (у видеороликов). Внешний вид элементов управления слегка различается у разных веб-обозревателей.

Пример	Результат
`<audio src="media/audio.mp3" controls></audio>`	
`<video src="media/video.mp4" controls></video>`	

  Если этот атрибут тега не указан, элементы управления выведены не будут. В таком случае элемент аудиоролика вообще не будет отображаться на странице, а у видеоролика будет видна лишь панель для воспроизведения видео;

- `muted` — атрибут без значения, предписывает изначально приглушить звук;
- `autoplay` — атрибут без значения, запускает воспроизведение ролика сразу после открытия страницы. Если он отсутствует в теге, воспроизведение запущено не будет, и пользователю придется запустить его вручную нажатием на кнопку пуска;

ВНИМАНИЕ!

Современные веб-обозреватели автоматически запускают на воспроизведение только видеоролики, не имеющие звукового сопровождения или с изначально приглушенным звуком (что достигается вставкой в тег `<video>` атрибута `muted`). Аудиоролики и видеоролики с неприглушенным звуком воспроизводиться автоматически не будут.

- `loop` — атрибут без значения, указывает зациклить воспроизведение ролика. Если не указан, ролик будет воспроизведен лишь один раз;

- `preload` — управляет предварительной загрузкой ролика. Можно указать одно из трех значений:
  - `none` — вообще не загружать ролик после загрузки страницы. Задав это значение, можно несколько ускорить открытие страницы, однако в этом случае воспроизведение ролика после запуска начнется с задержкой, а в элементе страницы изначально не будут выведены ни продолжительность ролика, ни первый кадр видео. Пример:

    ```
    <video src="media/video.mp4" preload="none"></video>
    ```

  - `metadata` — загрузить лишь начало ролика, где хранятся его продолжительность, размеры видео и его первый кадр, которые будут выведены в элементе;
  - `auto` или `""` (пустая строка) — загрузить весь ролик. Указывать это значение имеет смысл только у тех роликов, которые гарантированно будут воспроизведены посетителем.

  Значение по умолчанию: `metadata`.

  Если указан атрибут тега без значения `autoplay`, значение атрибута тега `preload` игнорируется.

Тег `<video>` дополнительно поддерживает атрибуты:

- `width` и `height` —ширина и высота видеоролика, выведенного на страницу. Значения записываются в виде обычных чисел и измеряются в пикселах. Пример:

  ```
  <video src="media/video.mp4" width="320" height="240"></video>
  ```

  Если эти атрибуты тега не указаны, ролик будет иметь изначальные размеры;

- `poster` — ссылка на файл с *постером* — графическим изображением, которое изначально будет выведено в панели воспроизведения видео. Пример:

  ```
  <video src="media/video.mp4" poster="images/pocter.png"></video>
  ```

  Если атрибут тега `poster` не указан, будет выведен первый кадр ролика или вообще ничего (в зависимости от значения атрибута тега `preload`, описанного ранее).

Аудио- и видеоролики, помещенные на страницу тегами `<audio>` и `<video>`, являются встроенно-блочными элементами (см. *разд. 7.2*).

7.3.2. Форматы аудио и видео, применяемые в WWW

Форматы аудио, поддерживаемые всеми веб-обозревателями, приведены в табл. 7.3.

Таблица 7.3

Формат файла	Формат кодирования звука
MP3	MP3
MP4	AAC или Opus
WebM	Vorbis или Opus
Ogg	

Форматы видео, поддерживаемые всеми веб-обозревателями, приведены в табл. 7.4.

Таблица 7.4

Формат файла	Формат кодирования звука	Формат кодирования видео
MP4	AAC или Opus	AVC, AV1 или VP9
WebM		AV1, VP8 или VP9
Ogg	Vorbis или Opus	VP8 или VP9

Отдельные веб-обозреватели поддерживают другие форматы аудио и видео. Так, Firefox поддерживает формат кодирования видео MP4V-ES, но остальные веб-обозреватели его поддержкой похвастаться не могут. Chrome, Edge, Firefox и Safari могут воспроизводить звук, закодированный в формате FLAC, а Opera — не может.

Понятно, что форматы, поддерживаемые не всеми веб-обозревателями, использовать не рекомендуется.

7.4. Упражнение. Видеокиоск

Видеокиоск — это обычная веб-страница, всю площадь которой занимает видеоролик, обычно запускаемый сразу после загрузки и воспроизводящийся бесконечно.

Сделаем видеокиоск, непрерывно воспроизводящий короткий фильм с видами леса.

1. Создадим где-либо на локальном диске папку kiosk, в котором будем хранить файлы киоска.
2. Создадим в ранее созданной папке kiosk папку media, в которую чуть позже поместим фильм, предназначенный для воспроизведения в киоске.
3. Найдем в папке 7\!sources сопровождающего книгу файлового архива (см. *приложение 4*) файл forest.mp4, содержащий фильм, и скопируем его в созданную ранее папку kiosk\media.

Теперь создадим единственную страницу нашего киоска.

В теге видеоролика обязательно укажем приглушение звука, иначе веб-обозреватель не запустит его воспроизведение после загрузки страницы. Зациклим ролик. А вот выводить элементы управления воспроизведением не нужно.

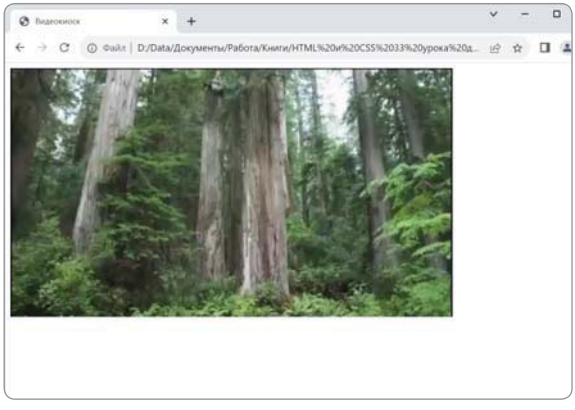
И сразу же привяжем к странице внешнюю таблицу стилей, в которой чуть позже запишем оформление киоска. Условимся, что таблица стилей будет храниться в файле styles\main.css.

4. Создадим в папке kiosk файл index.html и сохраним в нем следующий HTML-код:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="UTF-8">
        <link rel="stylesheet" href="styles/main.css">
        <title>Видеокиоск</title>
    </head>
    <body>
        <video src="media/forest.mp4" autoplay loop muted></video>
    </body>
</html>
```

У тега `<html>` мы указали обозначение языка, на котором написано содержание страницы, — в атрибуте тега `lang`.

✓ Что получилось? ›



В настоящий момент видеоролик занимает лишь часть страницы. А нам нужно, чтобы он занимал всю страницу. Для этого необходимо указать необходимые стили.

5. Создадим в папке kiosk папку styles.
6. Создадим в папке kiosk\styles таблицу стилей main.css и запишем в нее код, приведенный далее:

```
video {
    display: block;
```

```
    width: 100vw;
    height: 100vh;
}
```

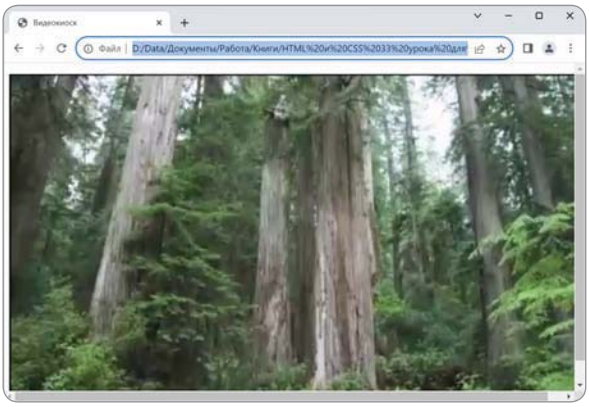
Здесь присутствует единственный стиль, применяемый к видеоролику (тегу `<video>`). Он делает следующее:

- превращает видеоролик, изначально являющийся встроенно-блочным элементом, в блочный — указанием значения `block` у атрибута стиля `display`.

  Если этого не сделать, веб-обозреватель при выводе преобразует пробельные символы между тегами `<body>` и `<video>` и между тегами `</video>` и `</body>` в пробелы, что создаст вертикальные просветы между границами окна и видеороликом (см. *разд. 4.2.3*), нам совершенно не нужные;

- указывает у видеоролика ширину, равную ширине окна веб-обозревателя, — с помощью знакомого нам атрибута стиля `width`. Единица измерения `vw` — это $^{1}/_{100}$ ширины окна веб-обозревателя;
- указывает у видеоролика высоту, равную высоте окна веб-обозревателя, — посредством атрибута стиля `height`. Единица измерения `vh` — это $^{1}/_{100}$ высоты окна веб-обозревателя.

✓ Что получилось? ⌄



Получилось так себе... Между левой и верхней границами окна веб-обозревателя и видеороликом все равно присутствуют небольшие просветы, из-за чего правая и нижняя части ролика вылезли за пределы окна. Такие же просветы имеются у правой и нижней границ окна — они станут видны, если прокрутить страницу.

Но даже если мы и уберем эти просветы, то убедимся, что панель для воспроизведения видео не всегда занимает все пространство элемента, — в ряде случаев по сторонам будут присутствовать свободные полосы, сквозь которые станет просвечивать белый фон страницы.

Нам нужно сделать две вещи. Во-первых, убрать просветы между границами окна и содержимым страницы — тогда ролик будет полностью

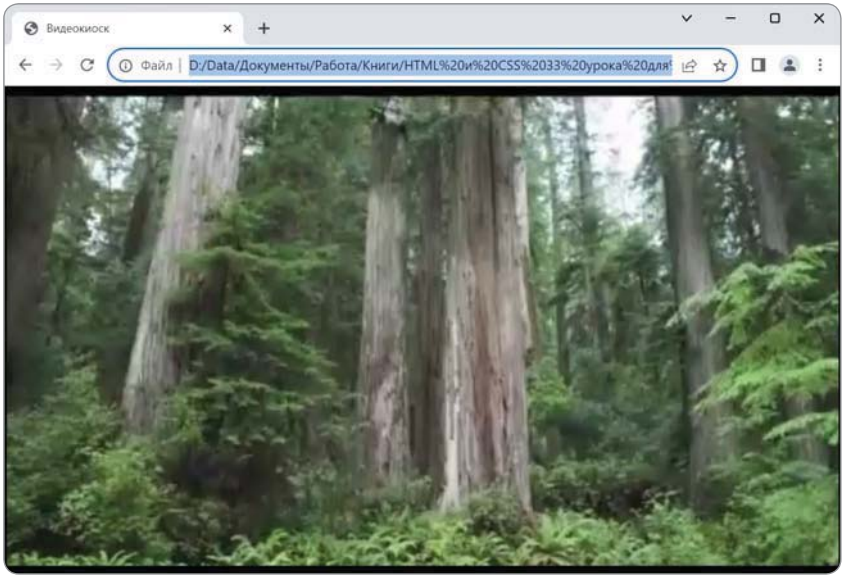
помещаться в окне. Во-вторых, указать у страницы черный цвет фона, чтобы избавиться от белых полос сверху и внизу.

7. Добавим в таблицу стилей styles\main.css еще один стиль:

```
body {
    margin: 0pt;
    background-color: black;
}
```

Этот стиль применится к секции тела страницы (тегу `<body>`). Атрибут стиля `margin` задаст просветы между всеми четырьмя границами окна и содержимым страницы равными 0 пунктов, т. е. уберет их совсем. А атрибут стиля `background-color` укажет черный (`black`) цвет фона страницы.

✓ Результат



Попробуйте изменить размеры окна веб-обозревателя — и убедитесь, что в любом случае элемент видеоролика занимает все окно целиком.

7.5. Упражнение. Размещение на веб-странице видео YouTube

Мы также можем поместить на страницу видео, опубликованное на видеохостинге YouTube.

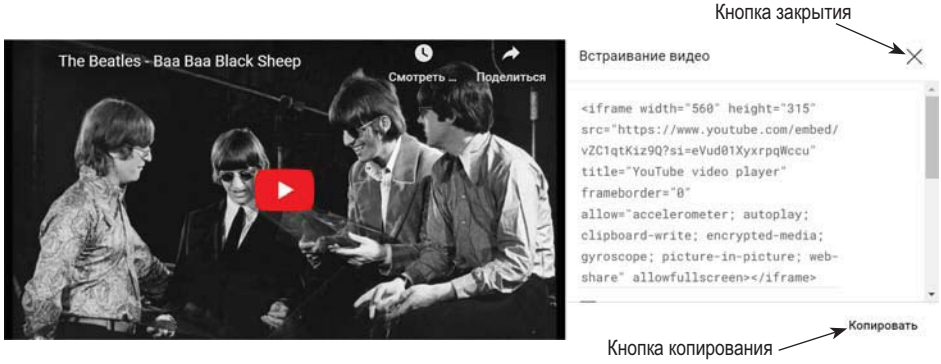
Создадим еще одну страничку и поместим на нее видео с концертным исполнением песни «Baa Baa Black Sheep» группы «The Beatles».

1. Создадим где-либо на локальном диске папку the-beatles, в которой будем хранить файлы новой страницы.
2. Создадим в папке the-beatles файл index.html и запишем в него тот же HTML-код, что записали в страницу из *упражнения 7.4*, только без тегов `<link>` и `<video>`.
3. Зайдем на страницу YouTube, на которой находится нужное нам видео[1].
4. Под панелью воспроизведения ролика найдем кнопку **Поделиться** и щелкнем на ней.

5. В появившемся на странице «окошке» нажмем кнопку **Встроить**.

[1] См. **https://www.youtube.com/watch?v=vZC1qtKiz9Q**.

6. Когда появится следующее «окошко», щелкнем на кнопке **Копировать**, находящейся в его правом нижнем углу.



   В результате HTML-код, реализующий вывод этого видео с YouTube, будет скопирован в буфер обмена.

7. Закроем «окошко», щелкнув на кнопке закрытия, которая находится в правом верхнем его углу и имеет вид крестика.

8. Вставим скопированный фрагмент HTML-кода в секцию тела страницы index.html. Этот фрагмент будет выглядеть так (выделен полужирным шрифтом):

```
<body>
    <iframe
        width="560" height="315"
        src="https://www.youtube.com/embed/vZC1qtKiz9Q?↳
        si=eVud01XyxrpqWccu"
        title="YouTube video player"
        frameborder="0"
        allow="accelerometer; autoplay; clipboard-write;↳
        encrypted-media; gyroscope; picture-in-picture; web-share"
        allowfullscreen>
    </iframe>
</body>
```

✓ Результат ›

Лицензия Creative Commons

Многие видеоролики, опубликованные на YouTube, защищены лицензией, которая запрещает размещение этих роликов на сторонних ресурсах. Если же такой ролик все же поместить на стороннем ресурсе, он воспроизводиться не будет.

Для размещения на сторонних ресурсах следует использовать ролики, опубликованные по лицензии Creative Commons.

7.6. Семантическая иллюстрация

Для размещения на странице изображения или видеоролика и подписи к нему можно использовать тег *семантической иллюстрации*.

> ***Семантическая иллюстрация***
>
> Элемент страницы, включающий изображение или видеоролик и текстовую подпись к нему.

Семантическая иллюстрация создается парным тегом `<figure>`, в котором помещается как изображение (видеоролик), так и подпись. Последняя заключается в парный тег `<figcaption>`.

Семантическая иллюстрация представляет собой блочный элемент. Помещенная в нее подпись (но не изображение или видеоролик!) также выводится как блочный элемент.

Пример:

```
<figure>
    <img src="images/the_beatles.jpg">
    <figcaption>The Beatles</figcaption>
</figure>
```

✓ Результат ›

The Beatles

Изображение (видеоролик) и подпись отображаются выровненными по левому краю с небольшим отступом слева. Для показа подписи применяется тот же шрифт, что и для вывода текста абзацев.

7.7. Вывод данных других форматов

На странице можно разместить данные других форматов, поддерживаемых веб-обозревателем, в частности, PDF-документов и изображений формата BMP. Для этого применяются два тега, рассматриваемых далее.

7.7.1. Тег <*embed*>

Одинарный тег `<embed>` выводит на страницу содержимое указанного файла. Этот тег поддерживает следующие атрибуты:

- `src` — ссылка на файл, в котором хранятся выводимые данные. Обязателен к указанию;
- `width` и `height` — соответственно, ширина и высота содержимого указанного файла при выводе на экран. Задаются положительными целыми числами и измеряются в пикселах. Если эти атрибуты не заданы, выведенное содержимое получит размеры 300 × 150 пикселов;
- `type` — MIME-тип содержимого файла.

> ***MIME-тип***
>
> MIME-тип (от Multipurpose Internet Mail Extensions, многоцелевые расширения интернет-почты) — обозначение типа содержимого файла.

Например, документ в формате PDF имеет MIME-тип `application/pdf`[2].

Если этот атрибут тега не указан, веб-обозреватель сделает попытку узнать тип содержимого по расширению файла.

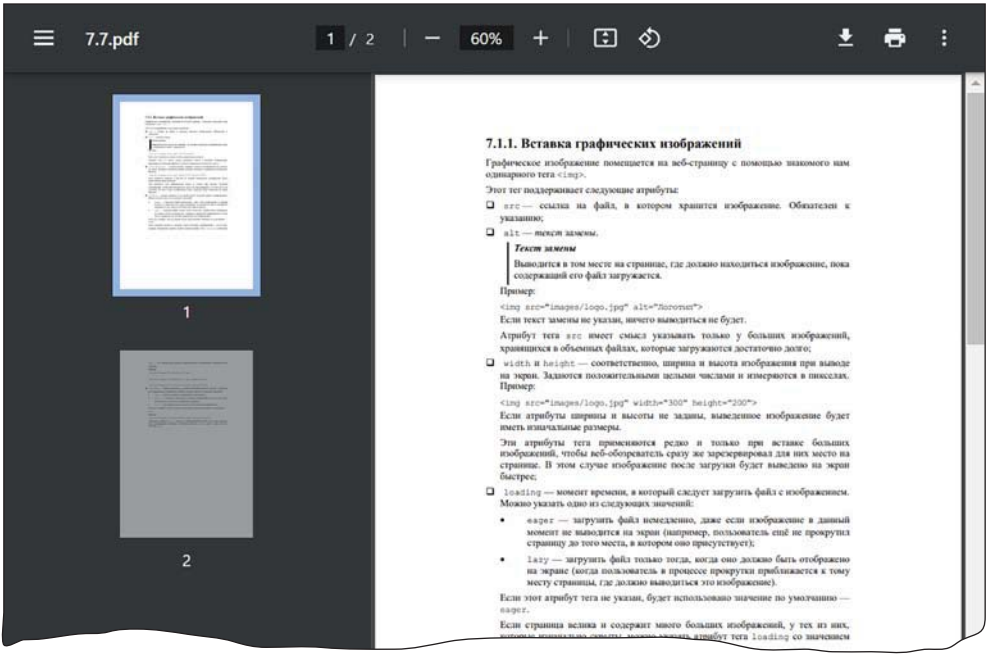
Атрибут тега `type` имеет смысл задавать, если расширение выводимого мультимедийного файла нехарактерно для его формата (например, файл с PDF-документом имеет расширение txt).

Пример вывода PDF-документа:

```
<embed src="docs/doc.pdf" width="800" height="600">
```

[2] Список наиболее востребованных MIME-типов приведен на странице **https://ru.wikipedia.org/wiki/Список_MIME-типов**.

✓ Результат



Как видим, документ можно даже печатать.

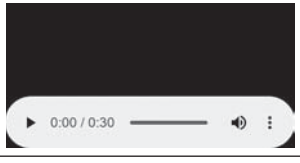
Пример вывода изображения BMP:

```
<embed src="images/sushi.bmp">
```

Изображение выводится в таком же виде, как и размещенное с помощью тега <img>.

Элемент, создаваемый тегом <embed>, является встроенно-блочным.

Посредством этого же тега также можно выводить графику и мультимедиа форматов, приведенных в *разд. 7.1.2* и *7.3.2*:

Пример	Результат
`<embed src="media/audio.mp3">`	
`<embed src="media/video.mp4">`	

Правда, аудио- и видеоролики, помещенные на страницу таким образом, выводятся на фоне черного прямоугольника, что выглядит некрасиво.

7.7.2. Тег <*object*>

Тег `<object>` полностью аналогичен тегу `<embed>` за двумя исключениями:

- является парным;
- для указания ссылки на выводимый файл в нем применяется атрибут тега `data`.

Содержимое у тега `<object>` не указывается.

Пример:

```
<object data="docs/doc.pdf" width="800" height="600"></object>
```

Наличие двух тегов с одинаковым назначением объясняется тем, что эти теги были предложены разными компаниями: тег <embed> — компанией Netscape, разработавшей веб-обозреватель Navigator (на базе которого впоследствии был создан Mozilla Firefox), а тег <object> — компанией Microsoft, выпустившей веб-обозреватель Internet Explorer. Эти веб-обозреватели занимали примерно одинаковые доли рынка, и W3C принял соломоново решение — включить в состав интернет-стандартов оба этих тега.

7.8. Внедренные элементы

> ***Внедренный элемент веб-страницы***
>
> Элемент, содержимое которого загружается из другого файла.

К внедренным элементам относятся изображения, аудио- и видеоролики, а также данные, выведенные тегами `<embed>` и `<object>`.

7.9. Самостоятельные упражнения

- Добавьте на страницу the-beatles\index.html аудиоролик с фрагментом песни «Let It Be» группы «The Beatles». Аудиофайл the_beatles_let_it_be.mp3 с этим фрагментом можно найти в папке 7\!sources сопровождающего книгу файлового архива (см. *приложение 4*).
- Добавьте на страницу the-beatles\index.html видеоклип с песней «Maxwell's Silver Hammer» той же группы, опубликованный на YouTube[3].

[3] Интернет-адрес страницы с этим клипом: **https://www.youtube.com/watch?v=bhLL_Ie8E2U**.

Урок 8. Таблицы

Создание таблиц.
Слияние ячеек.
Таблицы со сложным содержимым.
Секции таблиц.
Группы колонок и колонки.

8.1. Создание таблиц

Для создания таблицы используется набор из следующих парных тегов, вкладываемых друг в друга:

- `<table>` — создает саму таблицу. В этот тег вкладываются теги, создающие строки таблицы:
  - `<tr>` — создает отдельную строку таблицы. В этот тег вкладываются теги, создающие ячейки строки:
    - `<th>` — создает отдельную *ячейку шапки*. В такой ячейке выводится заголовок столбца, строки или подраздела в таблице;
    - `<td>` — создает отдельную ячейку общего назначения.

ВНИМАНИЕ!

Количество ячеек в каждой строке таблицы должно быть одинаковым (если не выполнялось слияние ячеек, о котором говорится в *разд. 8.2*).

Ячейки таблицы могут иметь произвольное, сколь угодно сложное содержимое: простой текст, абзацы, заголовки, списки, изображения, аудио- и видеоролики, даже другие таблицы.

Пример (код отформатирован для наглядности):

```
<table>
    <tr>
        <th>Число</th> <th>Квадрат числа</th> <th>Куб числа</th>
```

```
    </tr>
    <tr>
        <th>1</th>      <td>1</td>              <td>1</td>
    </tr>
    <tr>
        <th>2</th>      <td>4</td>              <td>8</td>
    </tr>
    <tr>
        <th>3</th>      <td>9</td>              <td>27</td>
    </tr>
</table>
```

✓ Результат

Число	Квадрат числа	Куб числа
1	1	1
2	4	8
3	9	27

По умолчанию таблица выводится без каких бы то ни было рамок между ячейками. Текст в ячейках шапки выводится полужирным шрифтом и выравнивается по центру. Текст в ячейках общего назначения выводится тем же шрифтом, что и текст абзацев, и выравнивается по левому краю. Во всех ячейках вертикальное выравнивание содержимого выполняется по центру.

Тег `<th>`, создающий ячейку шапки, поддерживает атрибут `scope`, указывающий, что озаглавливает создаваемая ячейка. Можно задать одно из пяти значений:

- `col` — ячейка шапки озаглавливает столбец, в котором находится;
- `row` — строку, в которой находится;
- `colgroup` — группу столбцов, в которой находится;
- `rowgroup` — группу строк, в которой находится;
- `""` (пустая строка) — веб-обозреватель сам решает, что озаглавливает ячейка.

Значение этого атрибута тега по умолчанию: `""` (пустая строка).

Значения `colgroup` и `rowgroup` используются, если ячейка шапки охватывает несколько столбцов или строк соответственно (что достигается слиянием ячеек, описанным в *разд. 8.2*).

Пример использования атрибута тега `scope`:

```
<table>
    <tr>
        <th scope="col">Число</th>
        <th scope="col">Квадрат числа</th>
        <th scope="col">Куб числа</th>
```

```
    </tr>
    <tr>
        <th scope="row">1</th>     <td>1</td>       <td>1</td>
    </tr>
    <tr>
        <th scope="row">2</th>     <td>4</td>       <td>8</td>
    </tr>
    <tr>
        <th scope="row">3</th>     <td>9</td>       <td>27</td>
    </tr>
</table>
```

Впрочем, атрибут тега `scope` задают редко, обычно для того, чтобы применить к ячейке какой-либо стиль.

8.2. Слияние ячеек таблицы

> ***Слияние ячеек***
>
> Объединение нескольких ячеек таблицы в одну. Если выполняется слияние по горизонтали, результирующая ячейка займет несколько соседних столбцов, при слиянии по вертикали — несколько соседних строк.

Для выполнения слияния ячеек следует выполнить следующие действия:

1. Найти тег `<th>` или `<td>`, создающий первую из ячеек, которые необходимо подвергнуть слиянию.
2. Записать в этот тег один из двух атрибутов:
   - `colspan` — для слияния ячеек по горизонтали;
   - `rowspan` — для слияния ячеек по вертикали.
3. Указать в этом атрибуте тега количество объединяемых ячеек, *включая текущую*.
4. Если ячейки объединяются:
   - по горизонтали — удалить из той же строки (тега `<tr>`) теги `<th>` или `<td>`, создающие следующие из объединяемых ячеек;
   - по вертикали — удалить из последующих строк теги, создающие следующие объединяемые ячейки.

   Эти ячейки более не нужны, поскольку текущая ячейка растянется по горизонтали или вертикали и займет их место.

Количество ячеек, подвергаемых слиянию, не ограничено.

Пример	Результат
Исходная таблица <table> <tr> <td>1</td> <td>2</td> <td>3</td> </tr> <tr> <td>4</td> <td>5</td> <td>6</td> </tr> <tr> <td>7</td> <td>8</td> <td>9</td> </tr> </table>	<table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></table>
Слияние двух ячеек по горизонтали <table> <tr> <td colspan="2">1 + 2</td> <td>3</td> </tr> <tr> <td>4</td> <td>5</td> <td>6</td> </tr> <tr> <td>7</td> <td>8</td> <td>9</td> </tr> </table>	<table><tr><td colspan="2">1 + 2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></table>
Слияние трех ячеек по вертикали <table> <tr> <td>1</td> <td rowspan="3">2 + 5 + 8</td> <td>3</td> </tr> <tr> <td>4</td> <td>6</td> </tr> <tr> <td>7</td> <td>9</td> </tr> </table>	<table><tr><td>1</td><td rowspan="3">2 + 5 + 8</td><td>3</td></tr><tr><td>4</td><td>6</td></tr><tr><td>7</td><td>9</td></tr></table>

8.3. Упражнение. Простая таблица

Добавим в состав разрабатываемого сайта суши-бара вторую страницу — с прайс-листом, который оформим в виде таблицы из двух столбцов: названия блюда и его цены. Таблицу разобьем на два раздела: «Суши» и «Сашими».

1. Найдем в папке 6\s6.6 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.
2. Создадим в папке site папку pages, в которой сохраним все остальные страницы сайта.
3. Создадим в папке pages страницу price-list.html и запишем в нее следующий HTML-код:

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <link rel="stylesheet" href="../styles/2.1.css">
        <title>Прайс-лист :: Йокогама: суши-бар</title>
    </head>
    <body>
        <header>
            <h1>Суши-бар<br>
                <span class="bar-title">Йокогама</span></h1>
        </header>
        <main>
            <h2>Прайс-лист</h2>
        </main>
        <footer>
            <address>Наш адрес: ул. Зеленая, 17.</address>
            <p>&copy; читатели книги.</p>
        </footer>
    </body>
</html>
```

Ради компактности мы сделали у этой страницы «сокращенную» шапку — без картинки суши. А в тег семантического основного содержимого (<main>) сразу добавили заголовок второго уровня с текстом «Прайс-лист».

Кроме того, поскольку эта страница хранится не в корневой, а во вложенной в нее папке, мы соответственно исправили относительный путь к внешней таблице стилей, записанный в теге <link>.

Напишем таблицу с прайс-листом. И сразу же зададим у нее «говорящий» стилевой класс `price-list`, чтобы потом применить стиль с оформлением.

4. Добавим в тег семантического основного содержимого код, создающий таблицу:

```
<table class="price-list">
    <tr><th>Товар</th><th>Цена, руб.</th></tr>
    <tr><th>Суши</th><th></th></tr>
    <tr><td>Чукка</td><td>40</td></tr>
    <tr><td>Магуро</td><td>80</td></tr>
    <tr><td>Тобико</td><td>80</td></tr>
    <tr><th>Сашими</th><th></th></tr>
    <tr><td>Унаги</td><td>340</td></tr>
    <tr><td>Хамачи</td><td>400</td></tr>
</table>
```

Названия разделов мы поместили в ячейки шапки (теги `<th>`), а остальной текст — в ячейки общего назначения (теги `<td>`).

✓ Что получилось? ⌄

Товар	Цена, руб.
Суши	
Чукка	40
Магуро	80
Тобико	80
Сашими	
Унаги	340
Хамачи	400

Веб-обозреватель — программа незатейливая, и предлагаемое им оформление по умолчанию тоже не отличается изяществом. Поэтому напишем стиль, который оформит таблицу.

5. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и добавим такой стиль:

```
table.price-list th, table.price-list td {
    border: grey thin solid;
    padding: 2pt;
}
```

Он содержит два селектора, приведенных через запятую, и, следовательно, будет применен к двум группам элементов страницы, на которые указывают эти селекторы. Селектор `table.price-list th` указывает на ячейки шапки, вложенные в таблицу (тег `<table>`) со стилевым классом `price-list`, а селектор `table.price-list td` — на ячейки общего назначения, вложенные в ту же таблицу.

Атрибут стиля `border` создает рамку, рисуемую по границе элемента страницы, в нашем случае — ячейки. Его значение `grey thin solid` предписывает создать серую (`grey`) тонкую (`thin`) рамку из сплошных (`solid`) линий.

Атрибут стиля `padding` создает просветы между рамкой и содержимым ячеек. Двух пунктов будет вполне достаточно.

✓ Что получилось?

Товар	Цена, руб.
Суши	
Чукка	40
Магуро	80
Тобико	80
Сашими	
Унаги	340
Хамачи	400

Некрасиво выглядят пустые ячейки в правой части строк с названиями разделов прайс-листа. Давайте уберем их, выполнив слияние по горизонтали каждой из этих ячеек с предыдущей, в которой выводится название раздела.

6. Исправим код, создающий строку таблицы с названием раздела «Суши»:

```
. . .
<tr><th>Суши</th><th></th></tr>
<tr><th colspan="2">Суши</th></tr>
. . .
```

Что получилось?

Товар	Цена, руб.
Суши	
Чукка	40

7. Внесем аналогичную правку в код, создающий строку таблицы с названием раздела «Сашими».

Что бы нам еще такое сделать?.. Ага! У суши «Магуро» и «Тобико» указана одинаковая цена — 80 руб. Выполним слияние ячеек с этой ценой по вертикали.

8. Исправим код, чтобы достичь заявленной цели:

```
. . .
<tr><td>Магуро</td><td>80</td></tr>
<tr><td>Магуро</td><td rowspan="2">80</td></tr>
<tr><td>Тобико</td><td>80</td></tr>
<tr><td>Тобико</td></tr>
. . .
```

✓ Результат

Магуро	80
Тобико	

Здесь хорошо заметно, что веб-обозреватель по умолчанию производит вертикальное выравнивание содержимого ячеек по вертикали. По крайней мере, нам самим не придется это делать...

8.4. Упражнение. Таблица со сложным содержимым

Как говорилось в *разд. 8.1*, ячейки таблицы могут иметь сколь угодно сложное содержимое. Мы можем помещать в ячейки изображения, аудио- и видеоролики, выделять фрагменты текста и структурировать текст с применением блочных элементов.

Добавим в состав сайта еще одну страницу, на которую поместим каталог всех разновидностей суши, предлагаемых баром «Йокогама», с иллюстрациями. Оформим этот каталог в виде таблицы.

1. Найдем в папке 8\ex8.3 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.
2. Найдем в папке 8\!sources сопровождающего книгу файлового архива (см. *приложение 4*) файлы sushi-chukka.jpg, sushi-maguro.jpg и sushi-tobiko.jpg, хранящие необходимые изображения, которые станут иллюстрациями к каталогу, и скопируем их в папку site\images.
3. Создадим в папке pages страницу sushi.html, взяв за образец HTML-код из *пункта 3 упражнения 8.3*. Только не забудем исправить текст названия страницы и заголовка второго уровня, заменив «Прайс-лист» на «Суши».

   У таблицы с каталогом сразу укажем стилевой класс `catalog`, чтобы позже написать стили с оформлением для нее.
4. Добавим в семантическое основное содержимое (тег `<main>`) страницы sushi.html код, создающий таблицу:

```
<table class="catalog">
    <tr>
        <td><img src="../images/sushi-chukka.jpg"></td>
        <td>
            <h3>Чукка</h3>
            <p>Диетические суши, приготовляемые из одноименного
               салата, риса, кунжута и водорослей нори.</p>
        </td>
    </tr>
    <tr>
        <td><img src="../images/sushi-maguro.jpg"></td>
```

```
            <td>
                <h3>Магуро</h3>
                <p>Классические суши.</p>
                <p>Готовятся из тунца и риса.</p>
            </td>
        </tr>
        <tr>
            <td><img src="../images/sushi-tobiko.jpg"></td>
            <td>
                <h3>Тобико</h3>
                <p>Икра летучей рыбы, рис, нори.</p>
            </td>
        </tr>
    </table>
```

Напишем два стиля, которые оформят таблицу.

5. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и добавим первый стиль:

```
table.catalog td {
    border-bottom: black thin dotted;
    padding: 3pt;
}
```

Он будет применен к ячейкам таблицы. Атрибут стиля `border-bottom` задает параметры нижней стороны рамки у элемента. Мы отделяем строки таблицы друг от друга черными (`black`) тонкими (`thin`) пунктирными (`dotted`) линиями. Также задаем просветы между линиями и содержимым ячеек в 3 пункта.

6. Запишем второй стиль:

```
table.catalog img { width: 200px; }
```

Он задаст у изображений, помещенных в таблицу, ширину в 200 пикселов (единица измерения `px` обозначает пикселы).

✓ Результат

Мы рассмотрели простой и наглядный пример того, как, пользуясь таблицами и средствами CSS, можно создавать красивые перечни из каких-либо позиций, содержащих графические иллюстрации.

8.5. Секции таблицы

В *разд. 5.2* описаны инструменты для разбиения содержания страницы на семантические части: шапку, основное содержимое, поддон и др. Нечто подобное можно делать и с таблицами, разбивая их содержимое на секции.

> ***Секция таблицы***
>
> Семантическая часть таблицы: шапка, основное содержимое или поддон.

Для создания секций таблицы применяются следующие парные теги:

- `<thead>` — создает секцию шапки;
- `<tbody>` — создает секцию основного содержимого;
- `<tfoot>` — создает секцию поддона.

Эти теги вкладываются в тег `<table>`, формирующий таблицу, а уже в них помещаются теги `<tr>`, создающие строки, которые должны быть помещены в соответствующие секции. Теги секций внутри тега `<table>` должны присутствовать в приведенном ранее порядке.

Пример (теги секций подчеркнуты):

```
<table>
    <thead>
        <tr>
            <th>Число</th><th>Квадрат числа</th><th>Куб числа</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <th>1</th><td>1</td><td>1</td>
        </tr>
        <tr>
            <th>2</th><td>4</td><td>8</td>
        </tr>
        <tr>
            <th>3</th><td>9</td><td>27</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td colspan="3">Всего 3 строки без заголовка</td>
        </tr>
    </tfoot>
</table>
```

По умолчанию содержимое секций таблицы никак не выделяется при выводе. Однако к секциям можно применить стили.

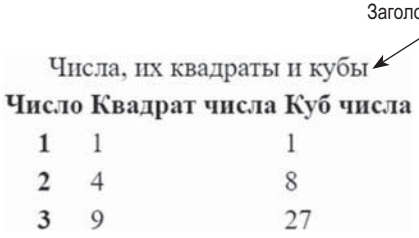
8.6. Заголовок таблицы

У таблицы можно указать *заголовок*. Его текст следует поместить в парный тег <caption>, который вставляется в самое начало содержимого тега <table>, перед всеми остальными тегами.

Пример (тег заголовка подчеркнут):

```
<table>
    <caption>Числа, их квадраты и кубы</caption>
    <tr>
        <th>Число</th><th>Квадрат числа</th><th>Куб числа</th>
    </tr>
    . . .
</table>
```

✓ Результат



По умолчанию заголовок таблицы выводится над таблицей, отображается обычным шрифтом и выравнивается по центру. Его можно оформить иначе (в частности, поместить под таблицей), написав стиль.

8.7. Группы колонок и колонки

Оформить каким-либо образом отдельные строки таблицы просто — достаточно применить нужный стиль к соответствующим тегам <tr>, например, указав у них стилевой класс:

```
<table>
    <tr>
        <th>Товар</th><th>Цена, руб.</th>
    </tr>
    <tr class="chukka">
        <td>Чукка</td><td>40</td>
    </tr>
    <tr>
        <td>Магуро</td><td rowspan="2">80</td>
    </tr>
</table>
. . .
```

```
.chukka {
    background-color: yellow;
}
```

Но что, если необходимо оформить отдельные столбцы таблицы? Задавать стилевой класс у всех тегов <td>, которые создают ячейки, входящие в состав оформляемых столбцов, трудоемко (особенно, если таблица большая).

Наилучшее решение — сформировать в таблице семантические группы колонок и, возможно, отдельные колонки и применить стили к ним.

8.7.1. Группы колонок

> ***Группа колонок***
>
> Объединяет произвольное количество столбцов таблицы, *расположенных друг за другом*. Используется с целью применить единый стиль к этим столбцам.

Группа колонок создается парным тегом <colgroup>. Атрибут тега span задает количество столбцов, входящих в группу колонок. Если он не указан, группа включит в себя лишь один столбец.

Тег <colgroup> помещается внутри тега <table>, в самом его начале, сразу после тега заголовка (<caption>), если он присутствует, и перед тегами секций или строк.

В таблице можно создать произвольное количество групп колонок. Первая группа объединит указанное количество столбцов, располагающихся в начале таблицы, вторая — указанное количество столбцов, находящихся далее, и т. д.

Пример таблицы с двумя группами колонок: первая содержит один столбец, вторая — два. Стиль, примененный к первой группе колонок, задает толстую рамку, стиль, примененный ко второй, — тонкую рамку. Теги групп колонок подчеркнуты:

```
<table>
    <colgroup class="col1"></colgroup>
    <colgroup class="col2-3" span="2"></colgroup>
    <tr>
        <th>Число</th><th>Квадрат числа</th><th>Куб числа</th>
    </tr>
    . . .
</table>
. . .
.col1 { border: black thick solid; }
.col2-3 { border: black thin solid; }
```

✓ Результат ›

Число	Квадрат числа	Куб числа
1	1	1
2	4	8
3	9	27

8.7.2. Колонки

Если необходимо задать дополнительное оформление у колонок, входящих в какую-либо группу, следует оформить столбцы, входящие в группу колонок, в виде семантических колонок и применить нужный стиль к соответствующей колонке.

> ***Колонка***
>
> Представляет собой один из столбцов, входящих в состав группы колонок. Используется с целью применить к этому столбцу стиль, задающий дополнительное оформление (помимо указанного у группы колонок).

Колонка создается одинарным тегом `<col>`.

Набор тегов `<col>`, описывающих отдельные колонки, помещается в тег `<colgroup>`, создающий группу колонок. Первая колонка будет создана на основе первого столбца из группы колонок, вторая колонка — на основе второго столбца и т. д.

Вот пример таблицы, аналогичной рассмотренной в *разд. 8.7.1*. Вторая группа колонок включает две колонки, создаваемые на основе, соответственно, первого и второго столбцов, входящих в группу. Стиль, примененный к первой колонке, задает светло-серый цвет фона, стиль, примененный ко второй колонке — ширину в 300 пикселов. Теги колонок подчеркнуты:

```
<table>
    <colgroup class="col1"></colgroup>
    <colgroup class="col2-3" span="2">
        <col class="col2">
        <col class="col3">
    </colgroup>
    <tr>
        <th>Число</th><th>Квадрат числа</th><th>Куб числа</th>
    </tr>
    . . .
</table>
. . .
.col2 { background-color: lightgrey; }
.col3 { width: 300px; }
```

✓ Результат ›

Число	Квадрат числа	Куб числа
1	1	1
2	4	8
3	9	27

Тег `<col>` поддерживает атрибут `span`, задающий количество столбцов таблицы, которые войдут в состав семантической колонки[1]. Если этот атрибут тега не указан, в колонку войдет один столбец.

8.8. Самостоятельные упражнения

- Добавьте в прайс-лист (страница pages\price-list.html) подраздел «Роллы» и три ролла: «Кунсэй батакон», «Аригато» и «Мексиканский». Задайте у них одинаковую цену в 360 руб. После этого выполните слияние всех трех ячеек с ценой по вертикали.
- Добавьте в прайс-лист еще одну строку с единственной ячейкой, занимающей два столбца и содержащей текст «Всего 8 наименований».
- Разделите таблицу с прайс-листом на семантические секции. В секцию шапки поместите шапку таблицы, в секцию поддона — последнюю строку с количеством наименований блюд, в секцию тела — остальные строки таблицы.
- Укажите у шапки таблицы белый цвет текста и серый цвет фона. Для задания цвета текста используйте атрибут стиля `color`. Белый цвет в CSS носит наименование `white`, серый — `grey`.
- Укажите у поддона таблицы курсивное начертание текста и выравнивание по правому краю.

✓ У вас должно получиться ›

Товар	Цена, руб.
Суши	
Чукка	40
Магуро	80
Тобико	
Сашими	
Унаги	340
Хамачи	400
Роллы	
Кунсэй батакон	360
Аригато	
Мексиканский	
Всего 8 наименований	

- Создайте в папке pages страницы sashimi.html и rolls.html с каталогами сашими и роллов. Текст для кратких описаний поищите в Интернете. Файлы изображений sashimi-unagi.jpg, sashimi-hamachi.jpg, roll-kunsei-batakon.jpg, roll-arigato.jpg и roll-mexican.jpg находятся в папке 8\!sources сопровождающего книгу файлового архива (см. *приложение 4*).

[1] На взгляд автора, эта функциональность избыточна.

✓ У вас должно получиться ⌄

Страница pages\sashimi.html

Страница pages\rolls.html

Урок 9. Гиперссылки

Гиперссылки.
Якоря.
Панель навигации.
Карта-изображение.

К настоящему времени мы изготовили пять страниц сайта о суши-баре «Йокогама». Но формально веб-сайтом они еще не являются, поскольку не связаны друг с другом.

> ***Веб-сайт***
>
> Набор веб-страниц, связанных между собой гиперссылками.

9.1. Собственно гиперссылки

> ***Гиперссылка***
>
> Ссылка на другой файл, опубликованный в Интернете, — как правило, на другую страницу. По щелчку на ней вызывает открытие этого файла в веб-обозревателе или, если формат файла не поддерживается веб-обозревателем, сохранение на локальном диске.
>
> ***Целевой файл***
>
> Файл, на который ссылается гиперссылка. Если гиперссылка указывает на страницу, говорят о *целевой веб-странице*.

Именно гиперссылки, связывающие отдельные страницы друг с другом, и превращают их в сайт.

Гиперссылка создается парным тегом `<a>`, внутрь которого помещается содержимое гиперссылки, — например, строка текста. Тег `<a>` поддерживает два атрибута:

- `href` — ссылка на целевой файл. Записывается в виде полного интернет-адреса или пути (подробности — в *разд. 4.4*). Обязателен к указанию;
- `target` — указание на то, где веб-обозреватель откроет целевой файл. Можно задать одно из двух значений:
  - `_self` — в текущей вкладке или окне;
  - `_blank` — в новой вкладке или окне.

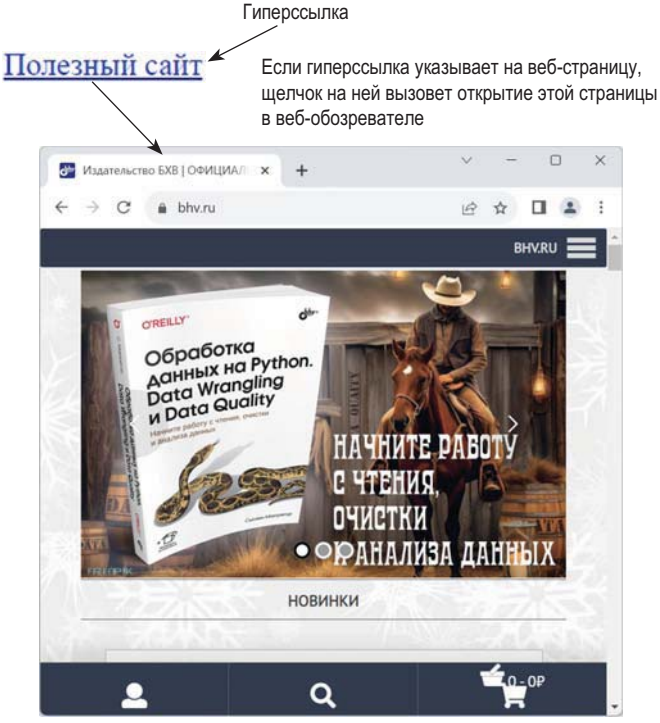
  Значение по умолчанию: `_self`.

Гиперссылка является встроенным элементом страницы.

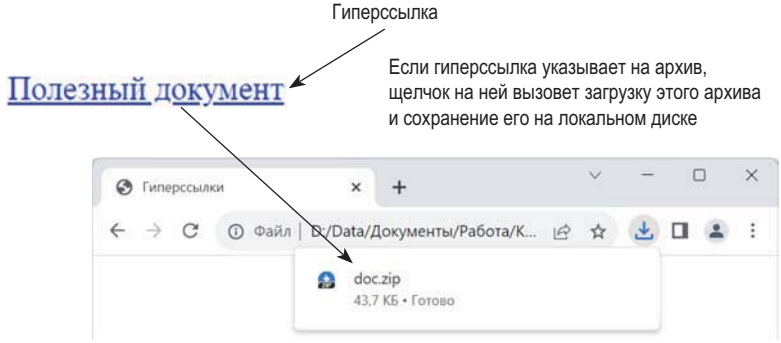
Пример:

```
<a href="https://bhv.ru/">Полезный сайт</a>
<a href="doc.zip">Полезный документ</a>
```

✓ Результат



То же самое произойдет, если гиперссылка указывает на любой файл, формат которого поддерживается веб-обозревателем: изображение, аудио- и/или видеоролик, документ PDF и др. Форматы графики, поддерживаемые веб-обозревателем, приведены в *разд. 7.1.2*, форматы аудио и видео — в *разд. 7.3.2*.



То же самое произойдет, если гиперссылка указывает на любой файл, формат которого *не* поддерживается веб-обозревателем: исполняемый файл, документ Word, фильм формата AVI и др.

По умолчанию текстовое содержимое гиперссылки выводится шрифтом синего цвета, с подчеркиванием. Текст *посещенной* гиперссылки выводится темно-красным цветом. При наведении на гиперссылку курсор мыши принимает вид «указующего перста».

> ***Посещенная гиперссылка***
>
> Гиперссылка, по которой уже был выполнен переход.

Обычная гиперссылка содержит текст и указывает на какой-либо файл, опубликованный в Интернете. Гиперссылки других разновидностей имеют другое содержимое и (или) указывают на другие интернет-ресурсы.

9.1.1. Графическая гиперссылка

> ***Графическая гиперссылка***
>
> Содержит графическое изображение, созданное тегом <img>.

Пример:

```
<a href="../index.html">
    <img src="../images/sushi.jpg">
</a>
```

По умолчанию графическая гиперссылка отображается так же, как и обычное изображение, и визуально никак не выделяется. Единственный признак, по которому ее можно отличить от обычного изображения, — курсор мыши при наведении на нее примет форму «указующего перста».

9.1.2. Почтовая гиперссылка

> ***Почтовая гиперссылка***
>
> Указывает на адрес электронной почты.

Почтовая гиперссылка создается тем же тегом `<a>`, в атрибуте `href` которого записывается целевой почтовый адрес, предваренный префиксом `mailto:` (двоеточие в конце обязательно). Пробелы между префиксом и адресом не допускаются.

Пример:

```
<a href="mailto:reception@yokogama.ru">Заказ столиков</a>
```

По щелчку на такой гиперссылке будет открыто окно программы, зарегистрированной в системе как почтовый клиент по умолчанию, подготовленное для написания нового письма с уже подставленным целевым адресом электронной почты.

Внешне почтовая гиперссылка ничем не отличается от обычной.

9.1.3. Загрузочная гиперссылка

> ***Загрузочная гиперссылка***
>
> Гиперссылка, по щелчку на которой *всегда* выполняется сохранение целевого файла на локальном диске, а не вывод его на экран (даже если формат целевого файла поддерживается веб-обозревателем).

ВНИМАНИЕ!

Щелчок на загрузочной гиперссылке вызовет сохранение файла лишь в том случае, если текущая страница была загружена с веб-сервера. Если страница была открыта непосредственно с локального диска, щелчок на загрузочной гиперссылке вызовет открытие файла в веб-обозревателе (если формат целевого файла им поддерживается).

Загрузочная гиперссылка создается тем же тегом `<a>`, только в нем записывается атрибут `download`. Его можно использовать двояко:

- как атрибут тега без значения — целевой файл будет сохранен под исходным именем:
  ```
  <a href="files/map.gif" download>Карта</a>
  ```
- как обычный атрибут тега, значение которого представляет имя файла, — целевой файл будет сохранен под именем, заданным в атрибуте тега:
  ```
  <a href="files/map.gif" download="our-location.gif">Карта</a>
  ```

Внешне загрузочная гиперссылка ничем не отличается от обычной.

9.1.4. Гиперссылка, ссылающаяся на фрагмент веб-страницы. Якоря

Если страница содержит большой текст, логично в ее начале создать оглавление из гиперссылок, указывающих на различные разделы этого текста. Заголовку каждого такого раздела назначается *якорь*, и ссылка на него записывается в соответствующей гиперссылке оглавления.

> ***Якорь***
>
> Уникальная пометка, своего рода закладка, поставленная в каком-либо теге с целью впоследствии сослаться на него.

Якорь записывается в теге с помощью атрибута `id`, описанного в *разд. 4.5*. Якорь должен содержать только буквы латиницы, цифры, дефисы и символы подчеркивания.

Пример записи якоря `par9-1-3` в заголовке раздела:

```
<h4 id="par9-1-3">9.1.3. Загрузочная гиперссылка</h4>
```

Ссылка на якорь, записываемая в теге гиперссылки, может быть указана в трех формах:

- если якорь находится на той же странице — в виде якоря, предваренного символом решетки:
  ```
  <a href="#par9-1-3">Раздел 9.1.3</a>
  ```
- если якорь находится на другой странице того же сайта — в виде комбинации пути к файлу целевой страницы, символа решетки и якоря, записанных вплотную, без пробелов:
  ```
  <a href="lesson9.html#par9-1-3">Урок 9, абзац 9.1.3</a>
  ```
- если якорь находится на странице другого сайта, — в виде комбинации полного интернет-адреса целевой страницы, символа решетки и якоря, также записанных вплотную:
  ```
  <a href="http://www.htmllessons.ru/pages/lesson9.html#par9-1-3">
      Урок 9, абзац 9.1.3
  </a>
  ```

Внешне гиперссылка, ссылающаяся на якорь, не отличается от обычной.

9.1.5. Пустая гиперссылка

> ***Пустая гиперссылка***
>
> Не содержит ссылки на какой-либо целевой файл.

Пустая гиперссылка может помочь в случае, если целевая страница, на которую указывает гиперссылка, еще не готова и будет сделана позже. Можно сказать, что в таком случае ставится временная «заглушка».

Создать пустую гиперссылку можно, указав в качестве значения атрибута `href` тега `<a>` символ решетки (`#`):

```
<a href="#">соусы</a>
```

Эксперименты автора показали, что с тем же самым успехом в атрибуте тега `href` можно указать и пустую строку:

```
<a href="">соусы</a>
```

По щелчку на пустой гиперссылке веб-обозреватель перезагружает открытую в настоящий момент страницу, в результате чего она прокручивается в самое начало. Изменить это поведение средствами HTML невозможно.

Внешне пустая гиперссылка ничем не отличается от обычной.

9.2. Семантические средства навигации

HTML позволяет создать две разновидности семантических элементов страниц, которые можно использовать для размещения гиперссылок, объединенных каким-либо общим признаком.

9.2.1. Панель навигации

Гиперссылки, ведущие на разные страницы сайта, помещают в семантическую *панель навигации*, которую делают как можно более заметной. Благодаря этому посетитель, в процессе путешествия по Интернету оказавшийся на одной из страниц сайта, сможет без проблем перейти на другие его страницы.

> ***Панель навигации***
>
> Элемент страницы, содержащий все гиперссылки, которые ведут на другие страницы сайта.

Гиперссылки в панели навигации могут располагаться как по вертикали, так и по горизонтали.

> ***Полоса навигации***
>
> Горизонтальная панель навигации.

Панель навигации создается парным тегом `<nav>` и представляет собой блочный элемент страницы.

Пример вертикальной панели навигации:

```
<nav>
    <div><a href="pages/lesson7.html">Урок 7. Графика</a></div>
```

```
    <div><a href="pages/lesson8.html">Урок 8. Таблицы</a></div>
    <div><a href="pages/lesson9.html">Урок 9. Гиперссылки</a></div>
</nav>
```

✓ Результат

Урок 7. Графика

Урок 8. Таблицы

Урок 9. Гиперссылки

Пример горизонтальной панели (полосы) навигации:

```
<nav>
    <a href="pages/lesson7.html">Урок 7. Графика</a> |
    <a href="pages/lesson8.html">Урок 8. Таблицы</a> |
    <a href="pages/lesson9.html">Урок 9. Гиперссылки</a>
</nav>
```

✓ Результат

Урок 7. Графика | Урок 8. Таблицы | Урок 9. Гиперссылки

По умолчанию содержимое панели навигации никак не выделается на экране при выводе. Поэтому к панели навигации практически всегда применяют стили с необходимым оформлением.

Панель навигации — ключевой элемент страницы, наряду с шапкой, основным содержимым и поддоном. В нее помещаются гиперссылки либо на все страницы сайта без исключения, либо, если сайт велик, на страницы, представляющие его разделы или иные крупные структурные единицы.

9.2.2. Меню

Гиперссылки, ведущие на страницы с дополнительными материалами к статье с текущей страницы, можно поместить в *меню*.

> ***Меню***
>
> Маркированный список, предназначенный для размещения гиперссылок.

Меню создается так же, как и обычный маркированный список (см. *разд. 5.1.2.1*), только вместо тега `<ul>` используется тег `<menu>`.

Пример:

```
<p>Дополнительные материалы к уроку 9:</p>
<menu>
    <li><a href="pages/lesson4.html">Урок 4. Основы HTML</a></li>
    <li><a href="pages/lesson5.html">Урок 5. Структура</a></li>
    <li><a href="pages/lesson7.html">Урок 7. Графика</a></li>
</menu>
```

По умолчанию меню, выведенное на экран, не отличается от обычного маркированного списка.

9.3. Упражнение. Панель навигации

Добавим на страницы нашего сайта панель навигации, посредством которой посетитель сможет перейти на каталоги суши, сашими, роллов и прайс-лист.

1. Найдем в папке 8\s8.8 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Панель навигации мы поместим в левой части страницы, а левую границу основного содержимого отодвинем вправо, чтобы эти ключевые элементы не налезали друг на друга.

2. Откроем главную страницу index.html в текстовом редакторе и между тегами <header> и <main> вставим HTML-код, формирующий панель навигации:

```
<header>
    . . .
</header>
<nav>
    <div><a href="index.html">Главная</a></div>
    <div><a href="pages/sashimi.html">Сашими</a></div>
    <div><a href="pages/sushi.html">Суши</a></div>
    <div><a href="pages/rolls.html">Роллы</a></div>
    <div><a href="pages/price-list.html">Прайс-лист</a></div>
</nav>
<main>
    . . .
</main>
```

Стоп! Правилами хорошего тона веб-верстки предписывается каким-либо образом уведомлять посетителя, на какой странице сайта он находится. Например, можно оформить позицию в панели навигации, соответствующую этой странице, не в виде гиперссылки, а в виде обычного текста.

Давайте и мы превратим первую гиперссылку, ведущую на текущую страницу, в обычный текст.

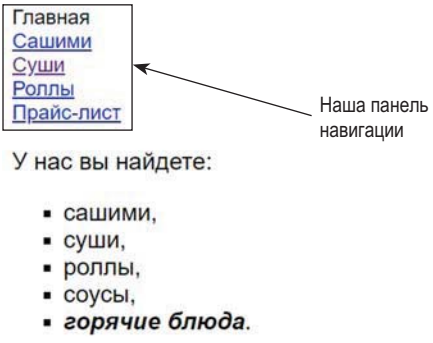
3. Внесем в код необходимое исправление:

```
<nav>
    <div><a href="index.html">Главная</a></div>
    <div>Главная</div>
    . . .
</nav>
```

Что получилось? ❯

Можете попробовать перейти по какой-либо гиперссылке (автор не удержался и посетил раздел суши).

Получившееся, прямо скажем, не впечатляет. Поэтому прибегнем к магии CSS.



Как мы условились ранее, сместим панель навигации к левому краю страницы и отодвинем от нее основное содержимое. Чтобы дополнительно выделить панель навигации, зададим для нее фон привлекательного цвета под названием `blanchedalmond` и стильные скругленные углы.

4. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и добавим стиль для панели навигации:

```
nav {
    background-color: blanchedalmond;
    float: left;
    width: 200px;
    margin: 10px 0px 10px 10px;
    padding: 20px;
    border-radius: 10px;
}
```

Рассмотрим лишь атрибуты стиля, с которыми плохо знакомы или не знакомы вообще:

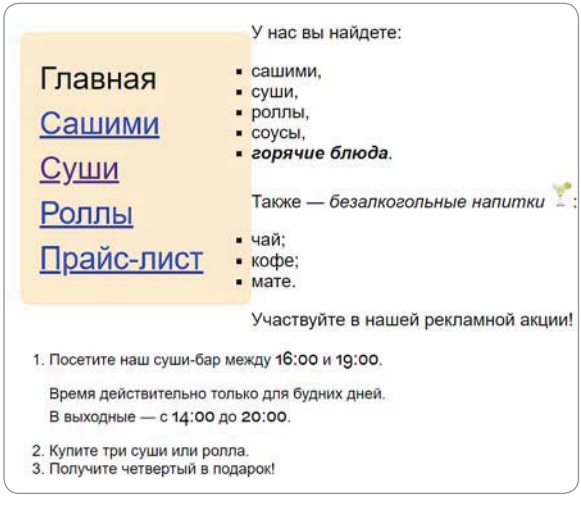
- атрибут стиля `float` со значением `left` сдвигает элемент страницы к левому краю его родителя (в нашем случае — страницы), в то время как содержимое других элементов будет обтекать его справа. Одновременно с этим понадобится задать для этого элемента ширину — значение в 200 пикселов было подобрано автором экспериментально;
- атрибут стиля `margin` задает величины «внешних» просветов между границей элемента, внутри которой рисуется фон, и соседними элементами. Его значение `10px 0px 10px 10px` определит просветы в 10 пикселов сверху, 0 (т. е. отсутствие просветов) — справа, 10 — снизу и столько же — слева;
- атрибут тега `padding` задает величины «внутренних» (между границей и содержимым элемента) просветов. Значение `20px` определяет просветы 20 пикселов со всех сторон;
- атрибут стиля `border-radius` со значением `10px` задаст радиус скругления углов границы элемента — 10 пикселов.

Займемся блоками с гиперссылками, находящимися в панели навигации. Увеличим кегль шрифта, которым они выводятся, и немного разнесем их друг от друга по вертикали.

5. Добавим еще один стиль — для блоков с гиперссылками, находящимися в панели навигации:

```
nav div {
    font-size: 24pt;
    margin: 10px 0px;
}
```

Значение `10px 0px` атрибута стиля `margin` задаст «внешние» просветы в 10 пикселов сверху и внизу и отсутствие таких просветов слева и справа.



✓ Что получилось? ›

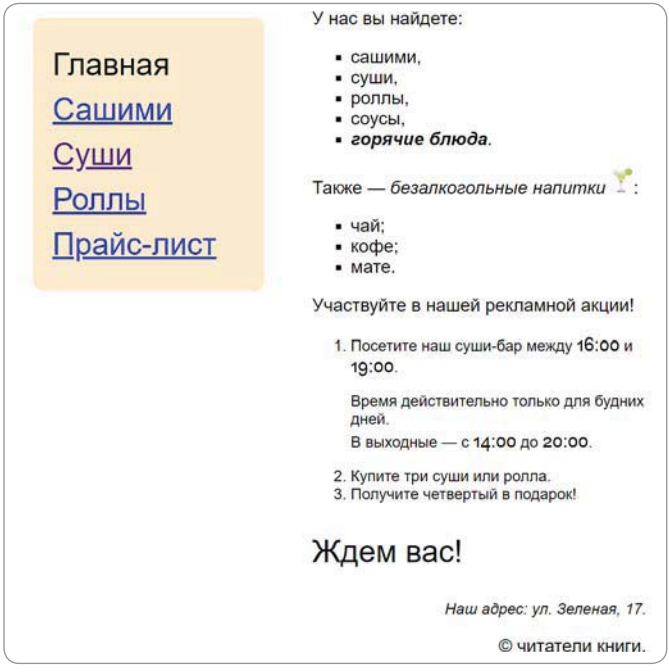
Панель навигации выглядит хорошо. Но остальное содержимое обтекает ее справа (таков эффект от применения атрибута стиля `float` со значением `left`) и, вдобавок, немного налезает на нее. Поэтому, как мы решили ранее, отодвинем левую границу основного содержимого вправо.

6. Добавим третий стиль — для основного содержимого:

```
main { margin-left: 300px; }
```

Атрибут стиля `margin-left` задает «внешний» просвет слева. 300 пикселов будет достаточно.

✓ Результат



9.4. Карта-изображение. Наименования тегов

> ***Карта-изображение***
>
> Изображение, включающее в себя произвольное количество областей, каждая из которых является отдельной гиперссылкой. Представляет собой обычное изображение, к которому применена *карта*.
>
> ***Карта***
>
> Описание набора областей, на основе которого будет создана карта-изображение. Должна быть применена к какому-либо изображению.

Карта создается парным тегом `<map>`. У создаваемой карты необходимо указать *наименование* (которое впоследствии будет записано в теге `<img>`, создающем изображение).

> ***Наименование***
>
> Уникальный идентификатор элемента страницы, который используется самим веб-обозревателем для выполнения различных действий с этим элементом.

Наименование записывается в теге посредством атрибута `name` (см. *разд. 4.5*). Наименование должно содержать только буквы латиницы, цифры, дефисы и символы подчеркивания.

Внутри тега `<map>` помещается набор одинарных тегов `<area>`, описывающих отдельные области. Тег `<area>` поддерживает атрибуты `shape` (задает форму области) и `coords` (размеры области), как показано в табл. 9.1.

Таблица 9.1

shape	coords	Создаваемая область
`rect`	`<x1>,<y1>,<x2>,<y2>`	**Прямоугольная** `x1,y1` — координаты левого верхнего угла, `x2,y2` — правого нижнего угла
`circle`	`<x>,<y>,<r>`	**Круглая** `x,y` — координаты центра, `r` — радиус
`poly`	`<x1>,<y1>,<x2>,<y2>,` `. . .,<xn>,<yn>`	**Полигональная** `xi,yi` — координаты *i*-й вершины полигона

Координаты отсчитываются от левого верхнего угла изображения и измеряются в пикселах. Ось абсцисс направлена вправо, ось ординат — вниз.

Для указания интернет-адреса целевого файла в теге `<map>` применяется знакомый нам атрибут тега `href` (см. *разд. 9.1*).

Если требуется создать неактивную область, не указывающую ни на какой целевой файл, атрибут тега `href` в этой области не указывается.

Тег `<map>` также поддерживает атрибуты `target` (см. *разд. 9.1*) и `download` (см. *разд. 9.1.3*).

Изображение, к которому должна быть применена созданная ранее карта, формируется обычным тегом `<img>`. В нем посредством атрибута тега `usemap` следует указать наименование созданной ранее карты, предваренное символом решетки.

Пример:

```
<map name="mainmap">
    <area shape="rect" coords="100,30,160,70" href="sashimi.html">
    <area shape="circle" coords="158,22,22" href="sushi.html">
```

```
    <area shape="poly" coords="10,0,10,10,24,20,10,46,0,46,0,0,10,0"
          href="rolls.html">
</map>
<img src="images/map.jpg" usemap="#mainmap">
```

Сама карта, созданная тегом <map>, на экран не выводится.

9.5. Самостоятельное упражнение

Поместите созданную при выполнении *упражнения 9.3* панель навигации на остальные страницы сайта. Не забудьте соответственно исправить пути к файлам страниц, записанные в ее гиперссылках. На каждой странице пункт панели навигации, ведущий на ту же страницу, превратите в обычный текст (как мы сделали это с пунктом «Главная» на главной странице).

✓ У вас должно получиться ⌄

Показана страница каталога суши.

Урок 10. Веб-формы, элементы управления, спойлер и специальные элементы

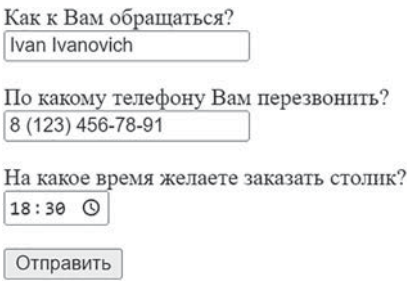
Обработка пользовательских данных.
Веб-формы.
Поля ввода и выбора.
Флажки и переключатели.
Списки.
Кнопки.
Надписи и группы.
Спойлер.

10.1. Обработка пользовательских данных. Веб-формы и веб-приложения

Многие веб-сайты принимают, обрабатывают и хранят данные, заданные посетителями (*пользовательские данные*): заказы, электронные письма и т. п. Для этого применяются *веб-формы* и *веб-приложения*.

> ***Веб-форма***
>
> Элемент страницы, предназначенный для занесения в него пользовательских данных (сведений о заказе, электронного письма и т. п.). Включает в себя элементы управления: поля ввода, списки, флажки, переключатели, кнопки и др.

Пример веб-формы:



После нажатия посетителем сайта на кнопку **Отправить** веб-обозреватель отправит веб-серверу клиентский запрос на запуск веб-приложения, включающий пользовательские данные из веб-формы.

> ***Веб-приложение***
>
> Программа, хранящаяся в файле в составе веб-сайта и обрабатывающая переданные ей пользовательские данные.
>
> Для написания веб-приложений применяются программные платформы PHP, Python, Ruby, Node.js и др. Их рассмотрение выходит за рамки книги.

В ответ на поступивший запрос веб-сервер запустит веб-приложение и передаст ему извлеченные из запроса пользовательские данные. Обработав их, веб-приложение сформирует результат их обработки — обычную веб-страницу, которую отошлет клиенту в составе серверного ответа. Клиент (веб-обозреватель) — выведет полученную страницу на экран.

Веб-обозреватель «не знает», что полученная им страница сформирована программно, и «полагает», что она хранится непосредственно в файле веб-приложения — даже выводит интернет-адрес этого файла в строке адреса.

Как вариант, веб-форма может отправлять данные по электронной почте. В таком случае для обработки данных не придется писать веб-приложение.

> ***Почтовая веб-форма***
>
> Веб-форма, отправляющая занесенные в нее данные на указанный адрес электронной почты.

После заполнения посетителем почтовой веб-формы и нажатия им на кнопку отправки веб-обозреватель обращается к программе почтового клиента, установленной на компьютере посетителя. Программа создаст новое письмо, подставит в него данные из веб-формы и адрес назначения, который записан в параметрах веб-формы. Посетитель при желании может перед отправкой дополнить содержание письма.

Почтовые веб-формы успешно работают только с программами почтовых клиентов — с онлайновыми службами они не взаимодействуют. К тому же, при формировании электронного письма символы кириллицы могут быть преобразованы в непонятные знаки (как правило, тому виной особенности настройки различных почтовых программ).

10.2. Веб-формы

10.2.1. Создание веб-форм

Веб-форма создается парным тегом `<form>`. Внутри него помещаются теги, создающие отдельные элементы управления.

Пример HTML-кода веб-формы, представленной на рисунке ранее:

```
<form>
    <p>Как к Вам обращаться?<br>
       <input name="client-name"></p>
    <p>По какому телефону Вам перезвонить?<br>
       <input name="client-phone"></p>
    <p>На какое время желаете заказать столик?<br>
       <input type="time" name="time-of-order"></p>
    <p><input type="submit"></p>
</form>
```

Бóльшая часть элементов управления создается одинарными тегами `<input>` (в приведенном коде подчеркнуты). В атрибуте `type` этого тега задается тип создаваемого элемента управления: обычное поле ввода, поле для указания времени, *кнопка отправки данных* и др.

> ***Кнопка отправки данных***
>
> При нажатии на нее запускает процедуру отправки данных, занесенных в веб-форму.

У каждого элемента управления, в который заносятся пользовательские данные, необходимо указать наименование — в атрибуте тега `name`. Если этого не сделать, данные, занесенные в этот элемент управления, не будут отправлены.

Во всех подробностях создание элементов управления будет рассмотрено в *разд. 10.3*.

10.2.2. Как работает веб-форма? Ключевые параметры веб-форм

При нажатии кнопки отправки данных веб-обозреватель извлекает из всех элементов управления, имеющихся в веб-форме, занесенные в них значения и формирует пары вида:

<наименование элемента управления>=<введенное в него значение>

Для приведенной ранее веб-формы таких пар будет три:

```
client-name=Ivan Ivanovich
client-phone=8 (123) 456-78-91
time-of-order=18:30
```

После формирования пар веб-обозреватель считывает ключевые параметры веб-формы, записанные в трех атрибутах тега `<form>`:

- `action` — ссылка на файл веб-приложения, которое будет обрабатывать данные на стороне сервера. Может быть указана в виде пути, абсолютного или относительного, или полного интернет-адреса. Пример:

  ```
  <form action="/apps/get-order.php" . . .>
  ```

  Веб-сервер не отправит этот файл клиенту, а запустит на выполнение хранящееся в нем приложение, передав ему полученные из веб-формы данные.

  У почтовой веб-формы здесь указывается адрес электронной почты, по которому будут отправлены данные. Его следует предварить префиксом `mailto:` (двоеточие в конце обязательно). Пример:

  ```
  <form action="mailto:order@yokogama.ru" . . .>
  ```

- `method` — *метод пересылки данных*. Можно указать одно из двух значений:

  - `get` — метод пересылки GET.

    Согласно этому методу, значения из элементов управления сначала кодируются: пробелы преобразуются в символы `+` (плюс), а буквы кириллицы и знаки препинания — в их числовые коды. Далее полученные пары добавляются в конец пути к файлу веб-приложения, записанного в атрибуте тега `action`, отделяются от пути вопросительным знаком (`?`), а друг от друга — амперсандами (`&`). Полученный в результате путь пересылается веб-серверу в составе клиентского запроса.

    Пример клиентского запроса, выполненного методом GET:

    Путь к файлу:
    /apps/get-order.php?client-name=Ivan+Ivanovich&
    client-phone=8+(123)+456-78-91&time-of-order=18:30

Веб-сервер извлекает из пути, пришедшего в составе запроса, полученные данные, декодирует их в изначальный вид и передает веб-приложению при запуске.

GET — метод пересылки данных по умолчанию; он используется, если атрибут `method` в теге `<form>` не указан.

Методом GET следует пересылать только данные небольшого объема. Веб-сервер, в зависимости от его настроек, может счесть путь к файлу веб-приложения слишком длинным и обрезать его.

Кроме того, этим методом нельзя пересылать конфиденциальные данные (например, пароли). Вспомним: результатом обработки пользовательских данных будет обычная страница, и при ее выводе веб-обозреватель покажет в панели адреса интернет-адрес файла веб-приложения, которое ее сгенерировало. И все переданные веб-приложению данные будут отображены в составе этого адреса.

- `post` — метод пересылки POST.

  Аналогичен методу GET, только данные пересылаются не в составе пути к файлу веб-приложения, а отдельно от него.

  Пример клиентского запроса, выполненного методом POST:

  Методом POST можно пересылать данные любого объема, в том числе и конфиденциальные. Они надежно «скрыты» в клиентском запросе, не выводятся на экран, и их никто не увидит.

ВНИМАНИЕ!

В почтовых веб-формах следует указывать метод пересылки данных POST. Если задать метод GET, веб-форма работать не будет.

- `enctype` — *метод кодирования данных,* принимается во внимание только при использовании метода пересылки данных POST. Поддерживаются три значения:
  - `text/plain` — данные вообще не кодируются и представляются в виде обычного текста. Указывается только у почтовых веб-форм;

- application/x-www-form-urlencoded — данные кодируются так же, как и при применении метода пересылки GET. Этот метод кодирования применяется по умолчанию, если атрибут тега enctype не указан;
- multipart/form-data — то же самое, что и предыдущий метод кодирования, но, помимо всего прочего, кодирует отправляемые из веб-формы файлы. Применяется в случае, если веб-форма содержит поля выбора файла, посредством которых и производится отправка файлов веб-приложению.

Пример веб-формы:

```
<form action="/apps/get-order.php" method="post"
      enctype="application/x-www-form-urlencoded">
    . . .
</form>
```

Прочитав все необходимые параметры веб-формы, веб-обозреватель выполняет отправку данных. Если веб-форма не является почтовой, он ожидает, пока веб-приложение не сгенерирует результирующую страницу.

10.2.3. Дополнительные параметры веб-форм

Тег <form> поддерживает ряд атрибутов, позволяющих задать дополнительные параметры веб-форм:

- autocomplete — управляет функцией автодополнения во всех полях ввода, присутствующих в веб-форме. Можно указать одно из двух значений:
  - off — отключить автодополнение;
  - on — включить автодополнение.

  Значение по умолчанию: on.

  Пример:

  ```
  <form autocomplete="off" . . . >
      . . .
  </form>
  ```

- autocapitalize — управляет автоматическим приведением букв, заносимых во все поля ввода, к верхнему регистру. Можно указать одно из следующих значений:
  - none или off — не приводить буквы к верхнему регистру при вводе;
  - sentences или on — приводить к верхнему регистру первые буквы в каждом предложении;
  - words — первые буквы в каждом слове;

- `characters` — все вводимые буквы без исключения.

Значение по умолчанию: `off` (у Firefox) или `on` (у остальных веб-обозревателей).

Пример:

```
<form autocapitalize="sentences" . . . >
    . . .
</form>
```

- `novalidate` — атрибут без значения, предписывает не проверять занесенные в элементы управления данные на корректность;
- `target` — см. *разд. 9.1.*

10.3. Элементы управления

10.3.1. Параметры, поддерживаемые всеми элементами управления

Существует ряд параметров и задающих их атрибутов тегов, которые поддерживаются всеми элементами управления:

- `required` — атрибут без значения, указывает, что в элемент управления обязательно должны быть занесены какие-то данные. Если не указан, посетитель может оставить этот элемент управления незаполненным.

  `Required` — один из атрибутов тегов, выполняющих валидацию.

> ***Валидация***
>
> Проверка данных, занесенных в элемент управления, на соответствие заданным правилам.

- `autofocus` — атрибут без значения, указывает, что текущий элемент управления должен получить фокус ввода сразу после загрузки страницы. Должен присутствовать лишь у одного элемента управления на странице;
- `readonly` — атрибут без значения, делает элемент управления доступным только для чтения;
- `disabled` — атрибут без значения, делает элемент управления недоступным для ввода;
- `tabindex` — номер в порядке обхода, выполняемого при последовательном нажатии клавиши <Tab>. Задается в виде целого числа. Пример:

```
<input type="submit" tabindex="4">
<input type="time" name="time-of-order" tabindex="3">
```

```
<input name="client-phone" tabindex="2">
<input name="client-name" tabindex="1">
```

По умолчанию обход элементов управления выполняется в том порядке, в котором они были созданы в HTML-коде.

К сожалению, делать элемент то недоступным, то, наоборот, доступным для ввода в зависимости от каких-то условий средствами HTML нельзя. Для этого необходимо писать *веб-сценарий* — программу, которая будет проверять необходимое условие и соответственно делать элемент доступным или недоступным. Такие веб-сценарии пишутся на языке программирования JavaScript. Рассказ о веб-программировании выходит за рамки этой книги.

10.3.2. Поля ввода и выбора

HTML позволяет создать несколько разновидностей полей для ввода или выбора значений разных типов. Все они создаются одинарным тегом `<input>`, а разновидность поля задается значением атрибута `type` этого тега. Вот доступные значения атрибута `type` тега `<input>` и соответствующие им виды полей ввода и выбора:

- `text` — обычное поле ввода, в которое можно занести значение любого вида: имя, фамилию, физический адрес и др.:

  ```
  Ваше имя:<br>
  <input type="text" name="client-name">
  ```

  Это значение атрибута `type` тега `<input>` по умолчанию. То есть, если атрибут `type` в теге `<input>` не указан, создается обычное поле ввода. Пример:

  ```
  <input name="client-name">
  ```

- `search` — поле ввода искомой строки для выполнения поиска. Ничем не отличается от обычного поля ввода;
- `password` — поле для ввода пароля. Скрывает вводимые символы, показывая на экране вместо них черные точки. Пример:

  ```
  Пароль:<br>
  <input type="password" name="user-password">
  ```

✓ Результат

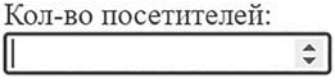
Пароль:
••••••••••

- `email` — поле для ввода адреса электронной почты. Выполняет валидацию занесенного в него значения, проверяя, является ли оно адресом;

- `tel` — поле ввода телефона. Ничем не отличается от обычного поля ввода;
- `url` — поле для ввода интернет-адреса. Выполняет валидацию, проверяя, является ли занесенное значение правильно сформированным интернет-адресом;
- `number` — поле для ввода целого числа. Выполняет валидацию, проверяя, является ли занесенное значение целым числом. Содержит в правой части кнопки для увеличения и уменьшения введенного числа на единицу. Пример:

```
Кол-во посетителей:
<br><input type="number" name="visitor-count">
```

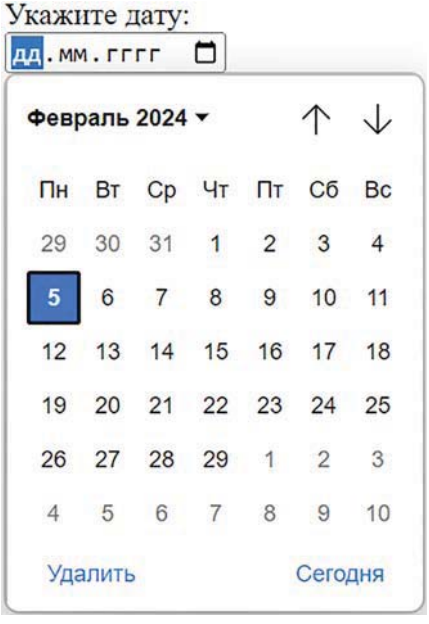
✓ Результат



- `date` — поле для ввода значения даты. По щелчку на расположенной в правой части поля кнопке выводится всплывающее окно календаря, в котором и выполняется выбор нужной даты. Дату также можно ввести вручную. Пример:

```
Укажите дату:<br>
<input type="date" name="date-of-visit">
```

✓ Результат



- `time` — поле для ввода значения времени. Время вводится вручную. В некоторых веб-обозревателях по щелчку на расположенной в правой части поля кнопке выводится всплывающее окно с прокручивающимся списком для выбора времени. Пример:

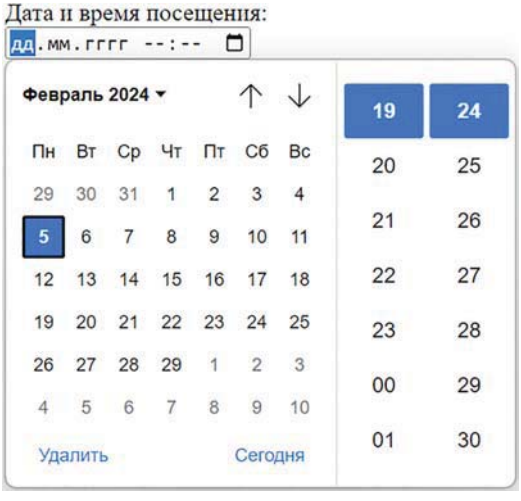
```
Укажите время посещения:<br>
<input type="time" name="time-of-visit">
```

✓ Результат ›

- `datetime-local` — поле для ввода значений даты и времени. По щелчку на расположенной в правой части поля кнопке выводится всплывающее окно с календарем и — в отдельных веб-обозревателях — прокручивающимся списком для выбора времени. Дату и время можно ввести и вручную. Пример:

```
Дата и время посещения:<br>
<input type="datetime-local" name="date-of-visit">
```

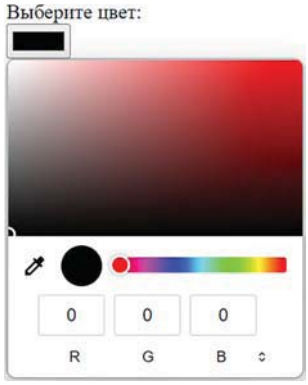
✓ Результат ⌄



- `color` — поле для выбора цвета. Отображается в виде кнопки, окрашенной в выбранный в настоящее время цвет. По нажатию этой кнопки появляется, в зависимости от веб-обозревателя, либо всплывающее окошко для указания цвета, либо стандартное диалоговое окно выбора цвета. Пример:

```
Выберите цвет:<br>
<input type="color" name="preferred-color">
```

✓ Результат



Для задания различных параметров полей ввода и выбора применяются следующие атрибуты тега `<input>` (табл. 10.1).

Таблица 10.1

Атрибут тега	Описание	Поддерживается полями

`value`	Значение, которое должно изначально присутствовать в поле ввода:
	`<input type="time" name="time-of-visit"`
	`value="18:00">`
		Всеми

`autocomplete`	Управляет функцией автодополнения в текущем поле ввода. Перекрывает значение одноименного атрибута тега `<form>`. Подробности — в *разд. 10.2.3*	

`autocapitalize`	Управляет автоматическим приведением вводимых букв к верхнему регистру в текущем поле ввода. Перекрывает значение одноименного атрибута тега `<form>`. Подробности — в *разд. 10.2.3*	`text`
		`search`
		`tel`
		`number`
		`date`
		`time`
		`datetime-local`

`size`	Ширина поля ввода в символах (по умолчанию: 20):
	`<input name="client-name" size="40">`
		`text`
		`search`
		`password`
		`email`
		`tel`
		`url`

`minlength`	Минимальная допустимая длина заносимой строки в символах (по умолчанию — не ограничена)	
`maxlength`	Максимальная допустимая длина заносимой строки в символах (по умолчанию — не ограничена)	

Таблица 10.1 *(окончание)*

Атрибут тега	Описание	Поддерживается полями

`placeholder`	Текст подсказки, выводимой непосредственно в поле ввода:
	`<input name="client-name" placeholder="Введите свое имя">`
	Результат:
	 
		`text`
		`search`
		`password`
		`email`
		`tel`
		`number`
		`url`

`min`	Минимально допустимое для ввода число, дата, время или дата со временем (по умолчанию — не ограничено)	`number`
		`date`
		`time`
		`datetime-local`

`max`	Максимально допустимое для ввода число, дата, время или дата со временем (по умолчанию — не ограничено)	
`step`	Интервал между заносимыми в поле целыми числами, датами (в днях), значениями времени и даты со временем (в секундах). По умолчанию: 1. Пример: `<input type="number" min="10" max="100" step="5">` В это поле можно ввести числа 10, 15, 20, 25, ... 95, 100	
`multiple`	Атрибут без значения, указывает, что в поле можно ввести несколько адресов электронной почты, разделив их запятыми	`email`

Все поля ввода и выбора (и вообще все элементы управления, создаваемые тегом `<input>`), являются встроенно-блочными элементами страницы.

Для размещения элементов управления в веб-форме в требуемом порядке их помещают в абзацы (как в примере из *разд. 10.2.1*), блоки или ячейки таблицы.

10.3.3. Флажок

Флажок создается одинарным тегом `<input>`, в атрибуте `type` которого задано значение `checkbox`. Поддерживаются два дополнительных атрибута тега:

- `value` — значение, которое будет пересылаться веб-приложению;
- `checked` — атрибут без значения, указывает, что флажок должен быть установлен изначально.

Если флажок установлен, в составе данных веб-формы будет присутствовать пара <наименование флажка>=<значение флажка>. Если же флажок сброшен, эта пара в данных веб-формы присутствовать не будет.

Пример:

```
<input type="checkbox" name="group" value="yes"> Групповое посещение
```

✓ Результат

10.3.4. Переключатель

Переключатель создается одинарным тегом <input>, в атрибуте type которого задано значение radio. Также поддерживаются два дополнительных атрибута тега:

- value — значение, которое будет пересылаться веб-приложению;
- checked — атрибут без значения; указывает, что переключатель должен быть установлен изначально. Делать установленным изначально следует лишь один переключатель в наборе.

Всем переключателям, входящим в один набор, дается одинаковое наименование (указывается в атрибуте name тега <input>).

Если какой-либо переключатель в наборе установлен, в составе данных веб-формы будет присутствовать пара:

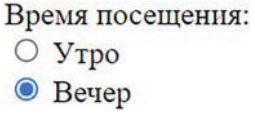
<наименование переключателя>=<значение переключателя>

Если ни один переключатель в наборе не установлен, эта пара в данных веб-формы присутствовать не будет.

Пример:

```
Время посещения:<br>
<input type="radio" name="time-of-visit" value="mourning"> Утро<br>
<input type="radio" name="time-of-visit" value="evening" checked> Вечер
```

✓ Результат



10.3.5. Область редактирования

Область редактирования создается парным тегом <textarea>. В нем записывается текст, который должен присутствовать в области редактирования изначально, сразу после загрузки страницы.

Тег <textarea> поддерживает следующие атрибуты:

- cols — ширина области редактирования в символах текста (по умолчанию: 20);
- rows — высота области редактирования, выраженная в строках текста (по умолчанию: 2);
- wrap — управляет переносом строк, которые по ширине не помещаются в области редактирования. Можно задать одно из трех значений:
  - soft — переносить строки;
  - hard — переносить строки и в местах переноса вставлять в текст символы разрыва строк. При указании этого значения необходимо задать ширину строк в атрибуте cols тега <textarea>;
  - off — не переносить строки. В этом случае при появлении слишком длинной строки в области редактирования отобразится горизонтальная полоса прокрутки.

  Значение по умолчанию: soft;
- spellcheck — управляет проверкой орфографии. Доступны для указания три значения:
  - true — задействует проверку орфографии;
  - false — отключает проверку орфографии;
  - default — задействовать или не задействовать проверку орфографии, решает веб-обозреватель.

  Значение по умолчанию: default.

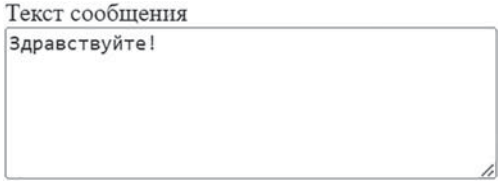
Также поддерживаются следующие атрибуты тега <textarea>: autocomplete, autocapitalize, minlength, maxlength и placeholder, описанные в *разд. 10.3.2*.

Область редактирования является встроенно-блочным элементом страницы.

Пример:

```
Текст сообщения:<br>
<textarea cols="40" rows="8">Здравствуйте!</textarea>
```

✓ Результат



Текст в области редактирования по умолчанию выводится моноширинным шрифтом. Если он не помещается в области редактирования полностью, появится вертикальная и (или) горизонтальная полосы прокрутки.

В правом нижнем углу области редактирования присутствует треугольный захват, который можно буксировать мышью, тем самым изменяя размеры области.

10.3.6. Списки

HTML позволяет создавать как обычные, так и раскрывающиеся списки.

Список создается парным тегом `<select>`. В нем записываются теги, формирующие отдельные пункты списка (будут рассмотрены чуть позже).

Тег `<select>` поддерживает следующие атрибуты:

- `size` — размер списка в пунктах, или, другими словами, количество пунктов списка, отображаемых на экране. Остальные пункты будут скрыты, и, чтобы увидеть их, потребуется прокрутить список.

  Если указано значение 0 или 1, будет создан раскрывающийся список, если же значение больше 1 — обычный список.

  Значение по умолчанию: 0;

- `multiple` — атрибут без значения, задает возможность выбора в списке произвольного количества пунктов. Принимается во внимание, только если у списка указан размер более 1 пункта.

  Если этот атрибут без значения отсутствует в теге, в списке можно выбрать лишь один пункт.

Список представляет собой встроенно-блочный элемент страницы.

Отдельный пункт списка создается парным тегом `<option>`. В него помещается текст создаваемого пункта.

Вот атрибуты, поддерживаемые тегом `<option>`:

- `value` — значение, которое будет пересылаться веб-приложению.

  Если этот атрибут тега не указан, веб-приложению будет отправлен текст пункта, записанный в теге `<option>`;

- `selected` — атрибут без значения, предписывает сделать текущий пункт списка выбранным сразу после загрузки страницы.

  Если текущий список позволяет выбрать лишь один пункт, атрибут `selected` тега `<option>` должен присутствовать только в одном пункте списка;

- `disabled` — атрибут без значения, делает текущий пункт недоступным для выбора;
- `label` — текст пункта. Если этот атрибут тега `<option>` не указан, в качестве текста пункта будет использовано содержимое текущего тега `<option>`[1].

Все теги `<option>`, создающие пункты списка, должны находиться внутри тега `<select>`, который создает сам список.

Веб-приложению в составе данных веб-формы будет отправлена пара вида `<наименование списка>=<значение выбранного в нем пункта>`. Если в списке выбрано несколько пунктов, такая пара будет сформирована для представления каждого из выбранных пунктов.

Пример	Результат
`Заказываемое блюдо: ` `<select name="dish">` `  <option value="chukka">Чукка</option>` `  <option value="maguro">Магуро</option>` `  <option value="tobiko">Тобико</option>` `</select>`	Заказываемое блюдо: Чукка
`Заказываемое блюдо: ` `<select name="dish" size="2">` `  <option value="chukka">Чукка</option>` `  <option value="maguro">Магуро</option>` `  <option value="tobiko">Тобико</option>` `</select>`	Заказываемое блюдо: Чукка Магуро
`Заказываемое блюдо: ` `<select name="dish" size="5">` `  <option value="chukka">Чукка</option>` `  <option value="maguro">Магуро</option>` `  <option value="tobiko">Тобико</option>` `</select>`	Заказываемое блюдо: Чукка Магуро Тобико
`Заказываемое блюдо: ` `<select name="dish" size="5" multiple>` `  <option value="chukka">Чукка</option>` `  <option value="maguro">Магуро</option>` `  <option value="tobiko">Тобико</option>` `</select>`	Заказываемое блюдо: Чукка Магуро Тобико

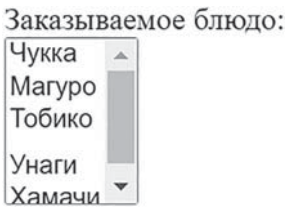
Отдельные наборы пунктов списка можно отделять друг от друга разделителями, имеющими, в зависимости от веб-обозревателя, вид либо довольно

[1] Непонятно, зачем нужен этот атрибут тега…

заметного вертикального просвета между пунктами, либо горизонтальной линии. Такой разделитель создается одинарным тегом `<hr>` (знакомым нам по *разд. 5.3.2*).

Пример	Результат

```
Заказываемое блюдо:<br>
<select name="dish" size="5">
    <option value="chukka">Чукка</option>
    <option value="maguro">Магуро</option>
    <option value="tobiko">Тобико</option>
    <hr>
    <option value="unagi">Унаги</option>
    <option value="hamachi">Хамачи</option>
</select>
```



Также можно объединять наборы из произвольного количества пунктов в группы, имеющие заголовки.

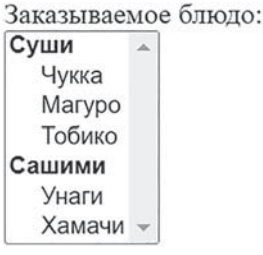
Группа пунктов создается парным тегом `<optgroup>`. Внутри этого тега помещаются теги `<option>`, создающий пункты, входящие в группу.

Тег `<optgroup>` поддерживает два полезных атрибута:

- `label` — заголовок группы. Если его не указать, группа не будет иметь заголовка[2];
- `disabled` — атрибут без значения, делает все пункты из текущей группы недоступными для выбора.

Пример	Результат

```
Заказываемое блюдо:<br>
<select name="dish" size="7">
    <optgroup label="Суши">
        <option value="chukka">Чукка</option>
        <option value="maguro">Магуро</option>
        <option value="tobiko">Тобико</option>
    </optgroup>
    <optgroup label="Сашими">
        <option value="unagi">Унаги</option>
        <option value="hamachi">Хамачи</option>
    </optgroup>
</select>
```



По умолчанию заголовки групп выводятся полужирным шрифтом, а пункты, входящие в группу, — с небольшим отступом слева.

[2] На взгляд автора, группа без заголовка выглядит некрасиво.

10.3.7. Кнопки

Можно создавать как простые кнопки, содержащие лишь текстовую надпись или изображение, так и кнопки со сложным содержимым, которое может быть произвольным.

10.3.7.1. Простые кнопки

Простая кнопка, содержащая обычную текстовую надпись или изображение, создается одинарным тегом `<input>`, в атрибуте `type` которого задано одно из следующих значений:

- `submit` — кнопка отправки данных, нажатие на которую вызывает отправку данных из веб-формы веб-приложению;
- `reset` — *кнопка сброса*. По нажатию очищает веб-форму, занося во все ее элементы управления изначальные значения, записанные в атрибутах `value`, `checked` и `selected` тегов `<input>` и `<option>`;
- `button` — простая кнопка, которая при нажатии на нее ничего не делает. Простые кнопки применяются для запуска веб-сценариев, выполняющих какие-либо действия над введенными в веб-форму данными;
- `image` — графическое изображение, работающее как кнопка отправки данных (*графическая кнопка*). Выглядит и ведет себя как графическая гиперссылка — при наведении на нее курсора мыши последний принимает вид «указующего перста».

Поддерживаются следующие атрибуты тега (табл. 10.2).

Таблица 10.2

Атрибут тега	Описание	Поддерживается кнопками

`value`	Надпись на кнопке. Если не указана, кнопки отправки данных и сброса будут выведены с надписями по умолчанию (зависят от веб-обозревателя, обычно — вида «Отправить» и «Сбросить»), а обычная кнопка будет выведена без надписи	`submit`
		`reset`
		`button`

`src`	Ссылка на файл с выводимым изображением	`image`
`width`	Ширина выводимого изображения в пикселах	
`height`	Высота выводимого изображения в пикселах	
`formaction`	Аналогичен атрибуту `action` тега `<form>` (см. *разд. 10.2.2*) и перекрывает его (то есть, если атрибут тега `formaction` в кнопке указан, веб-обозреватель использует именно его, а атрибут `action` тега `<form>` в веб-форме игнорирует)	`submit` `image`

Таблица 10.2 *(окончание)*

Атрибут тега	Описание	Поддерживается кнопками
`formmethod`	Аналогичен атрибуту `method` тега `<form>` (см. *разд. 10.2.2*) и перекрывает его	`submit` `image`
`formenctype`	Аналогичен атрибуту `enctype` тега `<form>` (см. *разд. 10.2.2*) и перекрывает его	
`formnovalidate`	Аналогичен атрибуту `novalidate` тега `<form>` (см. *разд. 10.2.2*) и перекрывает его	
`formtarget`	Аналогичен атрибуту `target` тега `<form>` (см. *разд. 10.2.2*) и перекрывает его	

Пример	Результат
`<input type="submit">`	Отправить
`<input type="submit" value="Заказать">`	Заказать
`<input type="reset">`	Сбросить
`<input type="button" value="Оставить отзыв">`	Оставить отзыв
`<input type="image" src="submit.svg" width="80">`	

10.3.7.2. Кнопки со сложным содержимым

Кнопка со сложным содержимым может содержать абзац, блок, изображение, видеоролик, таблицу или сочетание всего этого.

Такая кнопка создается парным тегом `<button>`. В него помещается содержание кнопки.

У кнопки необходимо указать тип, воспользовавшись для этого одним из атрибутов тега `type` (см. *разд. 10.3.7.1*). Поддерживаются значения: `submit`, `reset` и `button`.

Также поддерживаются все атрибуты тега, характерные для кнопок, — см. *разд. 10.3.7.1*.

Кнопка со сложным содержимым является встроенно-блочным элементом страницы.

Пример	Результат
`<button type="submit">Отправить</button>`	Отправить
`<button type="submit">` `  <strong>Отправить данные</strong>` `</button>`	Отправить данные
`<button type="submit">` `  Отправить данные ` `  <img src="submit.svg" width="80">` `</button>`	Отправить данные submit

10.3.8. Перечень для быстрого выбора

> ***Быстрый выбор***
>
> Предоставление посетителю заранее подготовленного перечня позиций, из которых он может выбрать нужную для занесения в поле ввода, связанное с перечнем.
>
> Перечень для быстрого выбора отображается во всплывающем окошке после двойного щелчка на связанном поле ввода. Ряд веб-обозревателей выводят в правой части поля ввода стрелку, направленную вниз, при нажатии на которую выполняется отображение перечня.

Перечень позиций для быстрого выбора создается парным тегом `<datalist>`. В этом теге записываются отдельные позиции создаваемого перечня, которые формируются тегами `<option>` (см. *разд. 10.3.6*). Текст позиции записывается в самом теге `<option>`, а значение, заносимое в связанное поле ввода, — в атрибуте `value` этого тега.

У созданного перечня следует задать какой-либо якорь (в атрибуте `id` тега `<datalist>`). Он понадобится для того чтобы связать перечень с полем ввода.

Чтобы связать перечень с полем ввода, в теге `<input>`, создающем поле ввода, следует указать атрибут `list`, значением которого должен являться якорь подготовленного ранее перечня. Атрибут `list` тега `<input>` поддерживается полями ввода всех типов, за исключением `password` (подробности о полях ввода и выбора — в *разд. 10.3.2*).

Пример:

```
<datalist id="dishes">
    <option>Чукка</option>
    <option>Магуро</option>
```

```
    <option>Тобико</option>
    <option>Унаги</option>
    <option>Хамачи</option>
</datalist>
. . .
Заказываемое блюдо:<br>
<input name="dish" list="dishes">
```

✓ Результат ›

10.3.9. Регулятор

Регулятор применяется для указания целых чисел. Он представляет собой шкалу с указателем, который можно перемещать, задавая нужное число.

Регулятор создается одинарным тегом `<input>`, атрибуту `type` которого дано значение `range`.

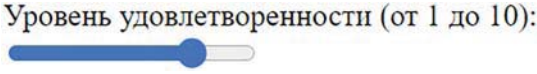
Поддерживаются следующие атрибуты тега `<input>`:

- `value` — значение, которое должно быть установлено на регуляторе изначально (по умолчанию: 50);
- `min` — минимальное из задаваемых значений (по умолчанию: 0);
- `max` — максимальное из задаваемых значений (по умолчанию: 100);
- `step` — интервал между задаваемыми значениями (по умолчанию: 1).

Пример:

```
Уровень удовлетворенности (от 1 до 10)<br>
<input type="range" name="level" min="1" max="10" value="8">
```

✓ Результат ›



Можно связать регулятор с перечнем для быстрого выбора. Тогда в атрибутах `value` тегов `<option>`, создающих позиции перечня, следует указать целочисленные эквиваленты этих позиций.

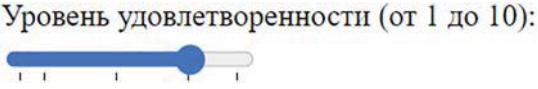
На шкале регулятора, связанного с перечнем быстрого выбора, будут отображаться отметки, соответствующие отдельным позициям перечня.

Пример:

```
<datalist id="levels">
    <option value="1">Хуже некуда</option>
    <option value="2">Бывало и хуже (но реже)</option>
```

```
    <option value="5">Недурно</option>
    <option value="8">Хорошо</option>
    <option value="10">Выше всех похвал</option>
</datalist>
. . .
Уровень удовлетворенности (от 1 до 10)<br>
<input type="range" name="level" min="1" max="10" value="8"
       list="levels">
```

✓ Результат ›



10.3.10. Поле выбора файла

> ***Поле выбора файла***
>
> Служит для выбора файла (или файлов), который будет отправлен веб-приложению.

Поле выбора файла создается тегом `<input>`, атрибуту `type` которого дано значение `file`.

Чтобы файлы были успешно отправлены, в атрибуте `enctype` тега `<form>` веб-формы следует указать метод кодирования данных: `multipart/form-data`, а в атрибуте `method` — метод отправки данных `post`. Если этого не сделать, файлы отправлены не будут.

Поле выбора файла поддерживает следующие атрибуты тега `<input>`:

- `accept` — типы файлов, доступные для выбора. Можно указать следующие значения:
  - расширение файла, обязательно с начальной точкой;
  - перечень расширений файлов с начальными точками, разделенных запятыми или комбинациями запятой и пробела;
  - `image/*` — любой графический файл формата, поддерживаемого веб-обозревателем;
  - `audio/*` — любой аудиофайл, поддерживаемый веб-обозревателем;
  - `video/*` — любой видеофайл, поддерживаемый веб-обозревателем.

  Если этот атрибут тега не указан, в поле можно выбрать файл любого формата;
- `multiple` — атрибут без значения, разрешает выбирать произвольное количество файлов. Если не указан, в поле можно выбрать только один файл.

Пример	Результат
`Выберите документ Microsoft Word: ` `<input type="file" accept=".doc,.docx">`	Выберите документ Microsoft Word: Выберите файл Веб-формы.doc

Поле выбора файла отображается в виде кнопки с надписью **Выберите файл** или **Обзор** (у разных веб-обозревателей надписи различаются), по нажатию на которую появляется стандартное диалоговое окно выбора файла. Правее кнопки показывается имя выбранного файла (файлов) или надпись **Файл не выбран**.

10.3.11. Скрытое поле

> ***Скрытое поле***
>
> Элемент управления, не отображающийся на экране и хранящий какое-либо неизменяемое значение.

Скрытое поле создается тегом `<input>` с атрибутом `type`, имеющим значение `hidden`. Само значение, которое должно содержаться в скрытом поле, записывается в атрибуте `value` тега `<input>`.

Пример:

```
<input type="hidden" name="operation-type" value="order">
```

10.3.12. Декоративные элементы

Декоративные элементы не предназначены для ввода данных, а служат лишь для оформления веб-форм.

10.3.12.1. Надпись

Надписью для элемента управления может послужить обычный текст (в примере подчеркнут):

```
Как к Вам обращаться?<br>
<input name="client-name">
```

Но для этой цели удобнее использовать специальный элемент страницы.

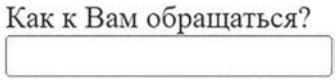
Надпись создается парным тегом `<label>`, в который помещается сам текст надписи. В атрибуте `for` этого тега указывается якорь элемента управления, с которым требуется связать надпись.

Надпись представляет собой встроенный элемент управления.

Пример:

```
<label for="client-name">Как к Вам обращаться?</label><br>
<input name="client-name" id="client-name">
```

✓ Результат ›



Надпись, выведенная на экран, ничем не отличается от обычного текста. Однако при наведении на нее курсор мыши принимает форму обычной стрелки, а при щелчке на ней связанный элемент управления активизируется: поле, область редактирования или список получает фокус ввода, флажок изменяет состояние, а переключатель устанавливается.

10.3.12.2. Группа элементов управления

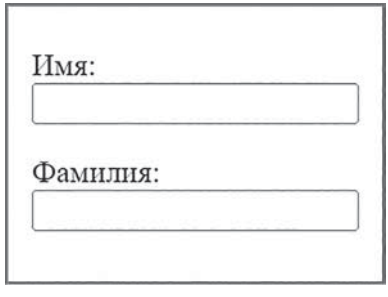
> ***Группа элементов управления***
>
> Служит для визуального объединения элементов управления, предназначенных для занесения данных, которые входят в состав одного блока.

Группа элементов управления создается парным тегом `<fieldset>`, в который помещаются теги элементов, входящих в эту группу.

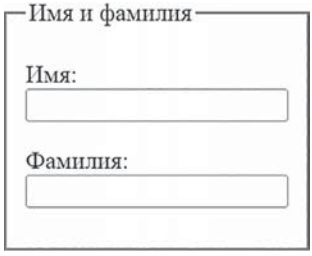
Тег `<fieldset>` поддерживает атрибут без значения `disabled`, который, будучи указанным, делает все элементы управления из текущей группы недоступными для ввода.

Группа — это блочный элемент страницы.

В теге группы можно указать парный тег `<legend>`, задающий заголовок группы. Текст заголовка записывается внутри тега. Тег `<legend>` может присутствовать в любом месте содержимого тега `<fieldset>`, но обычно его ставят в самом начале.

Пример	Результат
`<fieldset>` `  <p>Имя: ` `  <input name="name1"></p>` `  <p>Фамилия: ` `  <input name="name2"></p>` `</fieldset>`	

Пример	Результат

<fieldset>
  <legend>Имя и фамилия</legend>
  Имя:
 	
  <input name="name1">
  Фамилия:
 	
  <input name="name2">
</fieldset>
	

По умолчанию группа выделяется темно-серой рамкой, охватывающей все элементы управления, которые входят в группу. Заголовок, если он задан, выводится на верхней стороне рамки и выравнивается по левому краю.

10.4. Упражнение. Веб-форма заказа

Добавим в сайт суши-бара «Йокогама» страницу с веб-формой заказа еды на вынос. В этой веб-форме будут присутствовать:

- поле для ввода имени заказчика;
- обычный список для выбора заказываемых блюд, позволяющий выбрать произвольное количество пунктов;
- поле для занесения телефона заказчика;
- набор из трех переключателей для указания упаковки: стандартная, праздничная или экологически чистая. Первый переключатель в наборе должен быть установлен изначально;
- простая кнопка для отправки данных.

Оба поля ввода и список должны быть помечены как обязательные для заполнения.

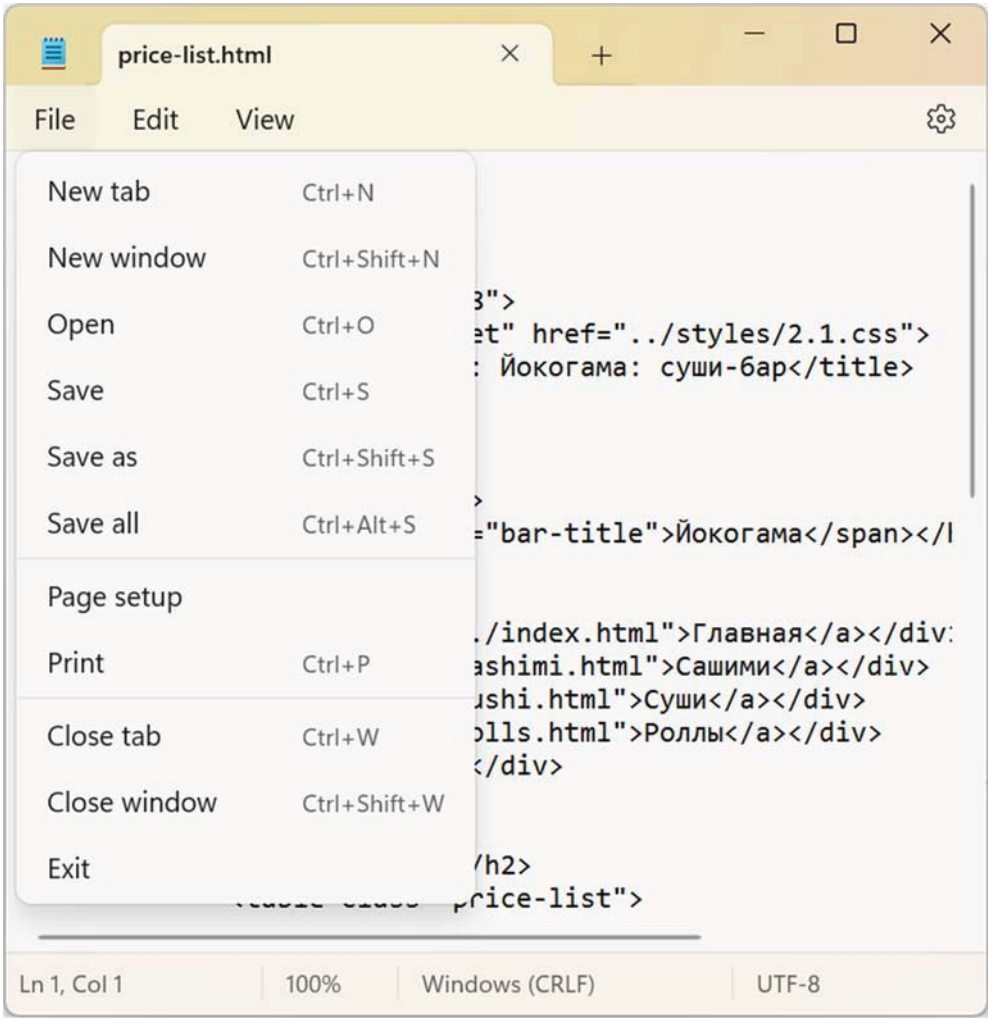
Надписи для элементов управления оформим тегами <label>.

1. Найдем в папке 9\s9.5 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.
2. Откроем страницу pages\price-list.html в текстовом редакторе и пересохраним ее в той же папке под именем order.html.

   Для пересохранения открытого файла, т. е. создания его копии с другим именем и (или) в другой папке, достаточно:

   - в английской версии Блокнота — выбрать в меню **File** пункт **Save as** или нажать комбинацию клавиш <Ctrl>+<Shift>+<S>;

- в русской — выбрать в меню **Файл** пункт **Сохранить как**.



А когда появится стандартное окно сохранения файла, занести в него сведения о создаваемой копии файла и выполнить сохранение (см. *упражнение 1.1*).

3. Откроем только что созданную страницу pages\order.html, удалим все основное содержимое (содержимое тега `<main>`), кроме заголовка второго уровня, и заменим во всем HTML-коде, кроме панели навигации, слова «Прайс-лист» на «Заказ блюд».

   Наша веб-форма будет почтовой и станет отправлять сведения о заказах по адресу **order@yokogama.ru**.

4. Поместим в теге <main> под заголовком второго уровня HTML-код, создающий саму веб-форму:

```
<h2>Заказ блюд</h2>
<form action="mailto:order@yokogama.ru" method="post"
      enctype="text/plain">
</form>
```

5. Добавим в тег <form> код, создающий обычное поле ввода, в которое будет заноситься имя заказчика:

```
<form . . .>
    <p>
        <label for="name">Как к Вам обращаться?</label><br>
        <input name="name" id="name" required>
    </p>
</form>
```

Пункты списка блюд, доступных для заказа, разделим на тематические группы.

6. Добавим в веб-форму список блюд в составе суши и сашими (роллы добавим потом):

```
<form . . .>
    . . .
    <p>
        <label for="dishes">Что желаете заказать?</label><br>
        <select name="dishes" id="dishes" size="6" multiple
required>
            <optgroup label="Суши">
                <option value="chukka">Чукка</option>
                <option value="maguro">Магуро</option>
                <option value="tobiko">Тобико</option>
            </optgroup>
            <optgroup label="Сашими">
                <option value="unagi">Унаги</option>
                <option value="hamachi">Хамачи</option>
            </optgroup>
        </select>
    </p>
</form>
```

7. Допишем код, создающий поле для ввода телефона заказчика:

```
<form . . .>
    . . .
    <p>
```

```
            <label for="phone">Ваш телефон:</label><br>
            <input type="tel" name="phone" id="phone" required>
        </p>
    </form>
```

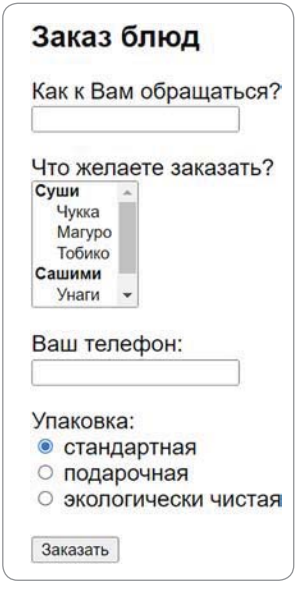
8. Добавим в веб-форму набор переключателей:

```
    <form . . .>
        . . .
        <p>
            Упаковка:<br>
            <input type="radio" name="package" id="package1"
                   value="standard" checked>
            <label for="package1"> стандартная</label><br>
            <input type="radio" name="package" id="package2"
    value="gift">
            <label for="package2"> подарочная</label><br>
            <input type="radio" name="package" id="package3" value="eco">
            <label for="package3"> экологически чистая</label>
        </p>
    </form>
```

9. Добавим код, создающий кнопку отправки данных:

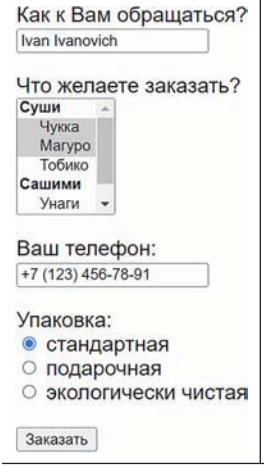
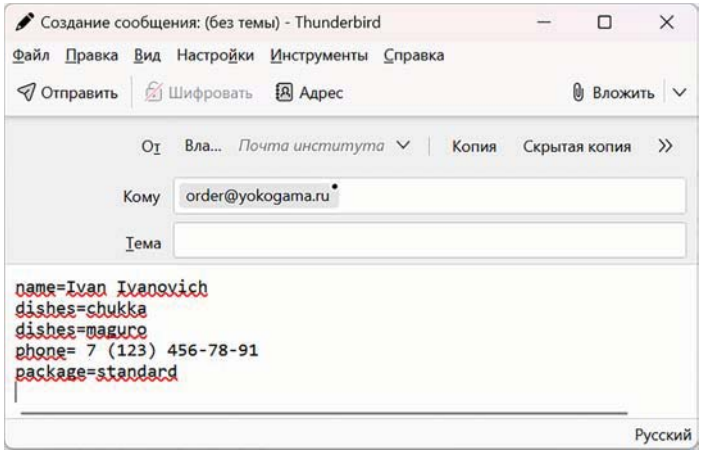
```
    <form . . .>
        . . .
        <p>
            <input type="submit" value="Заказать">
        </p>
    </form>
```

✓ Результат ›



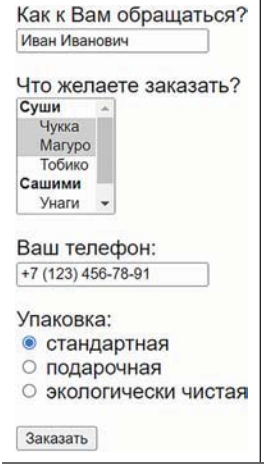
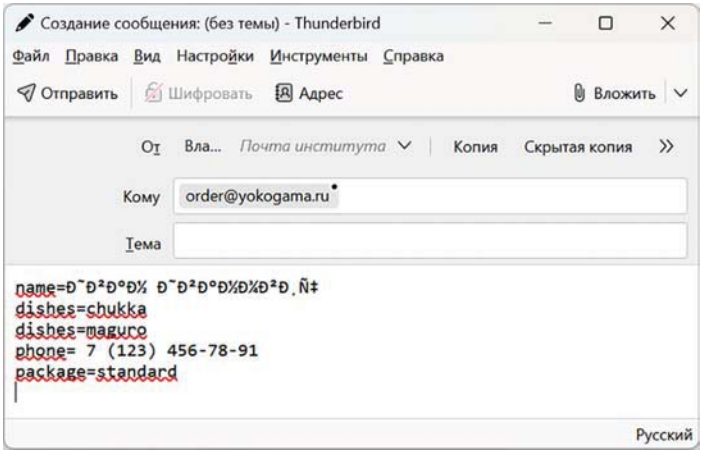
10.4.1. Эксперименты с веб-формой

1. Занесем в веб-форму какие-либо данные, указав имя заказчика латиницей, нажмем кнопку **Заказать** и подождем, пока не откроется окно почтового клиента, содержащее введенные данные.

Веб-форма с данными	Окно почтового клиента[1]
	

Отметим, что для представления каждого пункта, выбранного в списке блюд, создается отдельная пара вида *<наименование списка>=<значение выбранного пункта>*. В нашем случае таких пар две.

2. Сделаем то же самое, только теперь наберем имя заказчика буквами кириллицы.

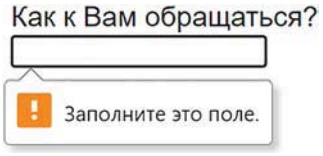
Веб-форма с данными	Окно почтового клиента
	

[1] Автор использовал Mozilla Thunderbird.

К сожалению, буквы кириллицы при формировании письма преобразуются в непонятные значки.

3. Обновим открытую в веб-обозревателе страницу, чтобы удалить из веб-формы все введенные данные.
4. Не вводя ничего, нажмем кнопку **Заказать**.

   Веб-обозреватель выведет предупреждение, требующее занести в первое поле ввода какое-либо значение.



5. Введем что-либо в первое поле ввода и нажмем кнопку **Заказать**.

   Веб-обозреватель выведет предупреждение, требующее выбрать в списке хотя бы один пункт.

   Подобного рода сообщения уведомляют посетителя о том, что занесенные им данные не прошли валидацию на соответствие правилам, заданным в HTML-коде.

10.5. Спойлер

> ***Спойлер***
>
> Панель, разворачивающаяся и сворачивающаяся по щелчку на заголовке.

Спойлер создается парным тегом `<details>`. Внутри этого тега помещается содержимое спойлера, которое может быть произвольным.

По умолчанию заголовок спойлера, в зависимости от веб-обозревателя, содержит надпись **Сведения** или **Подробности**. Сам спойлер по умолчанию выводится свернутым.

Тег `<details>` поддерживает атрибут без значения `open`, который делает спойлер изначально развернутым.

Спойлер — это блочный элемент страницы.

Пример:

```
<details>
    <h4>HTML</h4>
    <p>Язык, применяемый для написания веб-страниц.</p>
</details>
```

✓ Результаты

В свернутом состоянии	В развернутом состоянии
▶ Сведения	▼ Сведения **HTML** Язык, применяемый для написания веб-страниц.

По умолчанию спойлер никак не выделяется при выводе. Слева от заголовка присутствует маркер в виде черной стрелки, направленной вправо (в свернутом состоянии) или вниз (в развернутом состоянии).

У спойлера можно указать собственный заголовок, который может иметь произвольное содержимое. Содержимое заголовка помещается в парный тег `<summary>`, а сам этот тег — внутрь тега `<details>`.

Пример:

```
<details>
    <summary>HTML</summary>
    <p>Язык, применяемый для написания веб-страниц.</p>
</details>
```

✓ Результаты

В свернутом состоянии	В развернутом состоянии
▶ HTML	▼ HTML Язык, применяемый для написания веб-страниц.

Тег `<summary>` может присутствовать в любом месте тега `<details>` — на внешнем виде спойлера это никак не отразится:

```
<details>
    <p>Язык, применяемый для написания веб-страниц.</p>
    <summary>HTML</summary>
</details>
```

10.6. Специальные элементы

Специальные элементы применяются для вывода специфических данных.

10.6.1. Индикатор процесса

Индикатор процесса имеет вид шкалы с неподвижным указателем, показывающим процент выполнения какого-либо действия.

Индикатор процесса создается парным тегом `<progress>`. Содержимое у этого тега никогда не указывается.

Тег `<progress>` поддерживает следующие атрибуты:

- `max` — число, обозначающее, что задача выполнена полностью (по умолчанию: 1).

  В качестве значения допускается указывать как целые, так и вещественные числа.

  > ***Вещественное число (или число с плавающей точкой)***
  >
  > Десятичная дробь.
  >
  > В качестве разделителя целой и дробной части используется точка (а не запятая). Если целая часть числа равна 0, ее можно опустить.

- `value` — значение, выводимое индикатором процесса. Должно представлять собой целое или вещественное число от 0 до значения, заданного в атрибуте `max` тега `<progress>` (или 1, если этот атрибут тега не указан).

  Если атрибут `value` тега `<progress>` не указан, предполагается, что процент выполнения действия неизвестен. Тогда индикатор процесса будет находиться в неопределенном состоянии, при котором его указатель перемещается влево-вправо.

Индикатор процесса является встроенно-блочным элементом страницы.

Пример	Результат
`Заказ выполнен на 75% ` `<progress value="0.75"></progress>` `Заказ выполнен на 75% ` `<progress max="100" value="75"></progress>`	Заказ выполнен на 75%
`Заказ будет выполнен неизвестно когда ` `<progress></progress>`	Заказ будет выполнен неизвестно когда

10.6.2. Метр

> ***Метр***
>
> Шкала с неподвижным указателем, выводящим какое-либо значение. Разделена на три диапазона: нижний, средний и верхний — один из которых помечен как приоритетный. В зависимости от того, в какой диапазон попадает выводимое значение, указатель отображается разными цветами.

Метр создается парным тегом `<meter>`. Содержимое у этого тега никогда не указывается.

Тег `<meter>` поддерживает следующие атрибуты:

- `min` — минимальное из выводимых значений (по умолчанию: 0);
- `max` — максимальное из выводимых значений (по умолчанию: 100);
- `low` — верхняя граница нижнего диапазона (и нижняя граница среднего). Если не указан, его значение принимается равным значению атрибута тега `min` (т. е. фактически нижнего диапазона не будет);
- `high` — нижняя граница верхнего диапазона (и верхняя граница среднего). Если не указан, его значение принимается равным значению атрибута тега `max` (т. е. фактически верхнего диапазона не будет).

  Если не указать оба атрибута: `min` и `max`, метр будет иметь лишь один диапазон — средний. При этом он станет аналогом обычного индикатора процесса (см. *разд. 10.6.1*);
- `optimum` — оптимальное значение. Указывает, какой именно диапазон из имеющихся трех является предпочтительным. Если оптимальное значение располагается в пределах между значениями атрибутов тега `min` и `low`, предпочтительным является нижний диапазон; если оно располагается между значениями `low` и `high` — средний, а если между `high` и `max` — верхний диапазон. По умолчанию: 50;
- `value` — значение, выводимое метром (по умолчанию: 0).

В качестве значений приведенных атрибутов тега `<meter>` допускаются как целые, так и вещественные числа.

Метр является встроенно-блочным элементом страницы.

Пример	Результат
`Время выполнения заказа 10 мин.: ` `<meter min="0" max="60" low="20"` `high="40" optimum="15"` `value="10"></meter>`	Время выполнения заказа 10 мин.:

Пример	Результат
`Время выполнения заказа 25 мин.: ` `<meter min="0" max="60" low="20"` `high="40" optimum="15"` `value="25"></meter>`	Время выполнения заказа 25 мин.: 
`Время выполнения заказа 45 мин.: ` `<meter min="0" max="60" low="20"` `high="40" optimum="15"` `value="45"></meter>`	Время выполнения заказа 45 мин.: 

Если значение, выводимое метром, находится в предпочтительном диапазоне, шкала метра закрашивается зеленым цветом (см. первый пример), если в соседнем диапазоне — желтым (см. второй пример), а если в диапазоне, находящемся на противоположном конце шкалы, — красным (см. третий пример).

10.6.3. Прочие специальные элементы

Прочие специальные элементы создаются следующими парными тегами:

- `<output>` — область для вывода результатов каких-либо вычислений. Такие вычисления производятся веб-сценарием, написанным на языке JavaScript.

  В атрибуте `for` тега `<output>` можно указать якоря элементов управления, в которые заносятся исходные данные для выполнения вычислений, записанные через пробел.

  Область для вывода относится к встроенным элементам страницы.

  Пример:

  ```
  Первый множитель:<br>
  <input type="number" name="mult1" id="mult1">
  Второй множитель:<br>
  <input type="number" name="mult2" id="mult2">
  . . .
  <output for="mult1 mult2"></output>
  ```

- `<search>` — область для вывода результатов поиска. Является блочным элементом страницы.

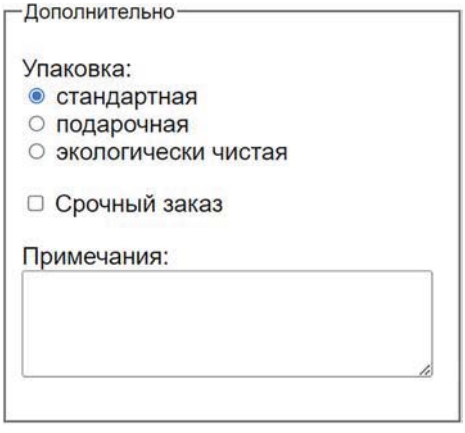
10.7. Самостоятельные упражнения

На странице pages\order.html:

- дополните список доступных для заказа блюд роллами. Создайте для них в списке отдельную группу пунктов **Роллы**;
- замените обычную кнопку отправки данных на следующую: ›

  Файл sushi-button.png с изображением суши для кнопки находится в папке 10\!sources сопровождающего книгу файлового архива (см. *приложение 4*);
- добавьте к веб-форме флажок **Срочный заказ** (наименование `urgent`) и область редактирования **Примечания**, не обязательную для заполнения (наименование `addenda`).
- заключите набор переключателей **Упаковка**, флажок **Срочный заказ** и область редактирования **Примечания** в группу с заголовком **Дополнительно**.

✓ У вас должно получиться ⌄



- Добавьте в панели навигации, располагающиеся на всех страницах сайта, гиперссылку **Заказ**, ведущую на страницу pages\order.html.

ЧАСТЬ III

CSS

- Настройки текста.
- Создание просветов и рамок.
- Указание размеров элементов.
- Параметры изображений, мультимедиа, таблиц, списков и элементов управления.
- Сплошные, градиентные и графические фоны.
- Позиционирование элементов.
- Задание формы элементов.
- Фильтры.
- Создание анимации.
- Верстка веб-страниц.

Урок 11. Основные понятия CSS (начало)

Правила набора CSS-кода.
Внешние и внутренние таблицы стилей.
Встроенные стили.
Правило каскадности.
Комментарии CSS.

CSS — это язык, предназначенный для описания оформления веб-страниц и их отдельных элементов. Мы познакомились с ним на *уроке 2*. Пришло время более обстоятельного разговора об этом весьма мощном языке.

На этом и следующем уроке практических упражнений не будет. Практика начнется на *уроке 13*.

11.1. Как записывается оформление веб-страниц?

Оформление страниц можно записать в виде таблиц стилей двух разновидностей, а также в виде встроенных стилей.

11.1.1. Таблицы стилей: внешние и внутренние

> ***Таблица стилей***
>
> Описание оформления, вынесенное за пределы HTML-кода, формирующего содержание страницы. Представляет собой набор отдельных стилей, определяющих оформление для одной из групп элементов страниц.

Каждый стиль состоит из *селектора*, описывающего элементы страницы, к которым будет применено оформление, и собственно описания

оформления. Последнее представляет собой набор *атрибутов стилей*, соответствующих отдельным параметрам оформления, и их значений.

Преимущества и недостаток таблиц стилей:

Преимущество	Недостаток
Все стили находятся в одном месте, что делает сопровождение сайтов более удобным.	Каждому стилю необходимо дать селектор, чтобы указать веб-обозревателю, к каким элементам страницы следует применить этот стиль.

11.1.1.1. Внешние таблицы стилей

> ***Внешняя таблица стилей***
>
> Хранится в отдельном файле и записывается в виде обычного неформатированного текста в текстовой кодировке UTF-8.

Файлы с внешними таблицами стилей должны иметь расширение css. Именовать эти файлы следует по тем же правилам, что и файлы со страницами (см. *разд. 4.1*).

Чтобы внешняя таблица стилей стала действовать на страницу, следует выполнить ее *привязку* к странице.

Привязка к странице внешних таблиц стилей выполняется с помощью одинарного тега `<link>`. Для указания ссылки на файл с привязываемой таблицей стилей и прочих параметров в нем предусмотрены следующие атрибуты тега:

- `href` — ссылка на файл с внешней таблицей стилей. Обязателен к указанию;
- `rel` со значением `stylesheet` — указывает, что привязываемый файл является внешней таблицей стилей.

Как правило, теги `<link>` помещаются в секцию заголовка страницы (в тег `<head>`), чтобы веб-обозреватель загрузил таблицы стилей *перед* загрузкой секции тела страницы.

Пример:

```
<head>
    . . .
    <link href="styles/main.css" rel="stylesheet">
</head>
```

Одна и та же внешняя таблица стилей может быть привязана к произвольному количеству страниц.

Преимущество и недостатки внешних таблиц стилей:

Преимущество	Недостатки
Могут быть привязаны к произвольному количеству страниц, что делает удобным указание единого оформления у всех страниц сайта.	• Дополнительные файлы в составе сайта. • Необходимость явной привязки к страницам.

Внешние таблицы стилей применяются практически всегда, за исключением крайне специфических случаев, описываемых далее.

11.1.1.2. Внутренние таблицы стилей

> ***Внутренняя таблица стилей***
>
> Помещается непосредственно в HTML-коде страницы.

Для создания внутренней таблицы стилей применяется парный тег `<style>`. В него записывается CSS-код таблицы стилей.

Как правило, тег `<style>` помещается в секции заголовка страницы.

Пример:

```
<head>
    . . .
    <style>
        body { font-family: Arial; }
        h1 {
            font-size: 32pt;
            text-align: center;
            font-family: Verdana;
        }
    </style>
</head>
```

В странице может присутствовать произвольное количество внутренних таблиц стилей. На практике обычно создают лишь одну.

Преимущества и недостаток внешних таблиц стилей:

Преимущества	Недостаток
• Отсутствие дополнительных файлов в составе сайта. • Нет необходимости явной привязки к странице	Действуют лишь на текущую страницу

Внутренние таблицы стилей используются редко, обычно при экспериментах с веб-технологиями. Также их применяют при разработке сайтов, состоящих из одной страницы, — *одностраничных сайтов*.

11.1.1.3. Несколько таблиц стилей в одной веб-странице

В одной странице можно указать произвольное количество таблиц стилей — как внешних, так и внутренних:

```
<head>
    . . .
    <!-- Основное оформление -->
    <link href="styles/basic.css" rel="stylesheet">

    <!-- Макет страницы -->
    <link href="styles/layout.css" rel="stylesheet">

    <!-- Оформление изображений -->
    <link href="styles/pictures.css" rel="stylesheet">

    <!-- Оформление веб-форм -->
    <link href="styles/forms.css" rel="stylesheet">

    <!-- Специфическое оформление текущей страницы -->
    <style>
        . . .
    </style>
</head>
```

В этом случае к странице будут применены все стили, присутствующие во всех таблицах стилей, которые записаны в HTML-коде.

Порядок применения параметров оформления, записанных в разных стилях, устанавливается веб-обозревателем согласно правилу каскадности (см. *разд. 2.5* и в подробностях — *разд. 11.4*).

11.1.2. Встроенные стили

> ***Встроенный стиль***
>
> Помещается непосредственно в теге, к которому должен быть применен.

Встроенный стиль записывается в атрибуте `style`, поддерживаемом всеми тегами. Он должен представлять собой последовательность атрибутов стилей со значениями, набранных в одну строку через символы точки с запятой. Селекторы и фигурные скобки во встроенном стиле не записываются[1].

[1] Поскольку они там не нужны — веб-обозреватель и так «знает», к какому тегу должен быть применен встроенный стиль.

Пример:

```
<img src="images/drink.jpg" style="width: 400px;">
```

Преимущества и недостаток встроенных стилей:

Преимущества	Недостаток
• За счет отсутствия селекторов более компактны, чем полноценные стили. • Имеют наивысший приоритет среди всех стилей.	Действуют лишь на те элементы страницы, в которых записаны.

Встроенные стили применяются крайне редко, лишь при разработке наиболее простых одностраничных сайтов и всевозможных экспериментах с оформлением.

Если вы, уважаемые читатели, встретите в HTML-коде какого-либо существующего сайта встроенные стили, то, скорее всего, этот сайт делали неаккуратно и на скорую руку.

11.2. Правила набора CSS-кода

11.2.1. Правила набора таблиц стилей

Таблица стилей, внешняя или внутренняя, представляет собой последовательность стилей, набранных один за другим.

Отдельный стиль набирается согласно следующему формату:

```
<селектор 1>, <селектор 2>, . . . <селектор N> {
    <атрибут стиля 1>: <значение атрибута стиля 1>;
    <атрибут стиля 2>: <значение атрибута стиля 2>;
    . . .
    <атрибут стиля N>: <значение атрибута стиля N>;
}
```

Пример стиля, применяемого к абзацам:

```
p {
    font-family: Arial;
    font-size: 14pt;
}
```

Если в стиле через запятую указано несколько *селекторов*, стиль будет применен к нескольким группам элементов страницы, описываемым этими селекторами. Пример стиля, применяемого к заголовкам всех шести уровней и адресам:

```
h1, h2, h3, h4, h5, h6, address {
    color: red;
    text-align: center;
}
```

Селекторы, атрибуты стиля и их значения, за исключением редких специфических случаев (о которых мы позже поговорим особо), допускается набирать буквами произвольного регистра:

```
P {
    FONT-FAMILY: aRIAL;
    fOnT-SiZe: 14PT;
}
```

Но практически всегда их набирают буквами нижнего регистра.

Между селекторами, запятыми, фигурными скобками, атрибутами стиля, их значениями, символами двоеточия и точки с запятой допускается произвольное количество пробелов. Также эти составные части стиля можно набирать вплотную, без пробелов. Так, следующие три стиля полностью равноценны:

```
p{font-family:Arial;font-size:14pt;}
p  {  font-family  :  Arial  ;  font-size  :  14pt  ;  }
p {
    font-family
                  :
                    Arial;
    font-size
                  :
                    14pt;
  }
```

После значения последнего атрибута стиля, присутствующего в стиле, допускается не ставить точку с запятой:

```
p {
    font-family: Arial;
    font-size: 14pt
}
```

Существуют особые программы, называемые *минификаторами*, которые уменьшают объем таблиц стилей за счет удаления из CSS-кода пробельных символов.

11.2.2. Правила набора встроенных стилей

Встроенный стиль, записываемый в атрибуте тега `style`, набирается в одну строку в следующем формате:

```
<атрибут стиля 1>: <значение атрибута стиля 1>; ↵
<атрибут стиля 2>: <значение атрибута стиля 2>; ↵
. . . ↵
<атрибут стиля N>: <значение атрибута стиля N>;
```

Ни селектор, ни фигурные скобки в нем не указываются.

В остальном встроенные стили схожи с полноценными, имеющими селекторы (см. *разд. 11.2.1*). В частности, в них допускается произвольное количество пробельных символов, равно как и их отсутствие. Например, следующие два встроенных стиля полностью аналогичны:

```
<img . . . style="width: 400px; height: 300px;">
<img . . . style="width:400px;height:300px;">
```

Во встроенные стили стараются не добавлять лишние пробелы, чтобы сделать HTML-код компактнее.

После *значения* последнего *атрибута стиля* допускается не ставить точку с запятой:

```
<img . . . style="width: 400px; height: 300px">
```

11.2.3. Форматирование CSS-кода при наборе

- Селектор с открывающей фигурной скобкой, каждый из атрибутов стиля вместе со значением и закрывающая фигурная скобка ставятся в отдельных строках. Атрибуты стиля со значениями набираются с отступом слева, традиционно равным 4 пробелам (иногда для компактности применяют отступ в 2 пробела). Между селектором и открывающей фигурной скобкой, а также двоеточием и значением атрибута стиля, ставятся пробелы. Пример:

```
p {
    font-family: Arial;
    font-size: 14pt;
}
```

- Если стиль содержит несколько селекторов, между каждой запятой и следующим селектором ставится пробел:

```
h1, h2, h3, h4, h5, h6, address {
    color: red;
    text-align: center;
}
```

- Слишком длинное значение атрибута стиля можно разбить на отдельные строки, но только в промежутках между отдельными величинами, входящими в состав этого значения. Вторая и каждая последующая строки обычно начинаются с колонки, на которой находится первый символ начала значения, записанного в первой строке. Пример:

```
table.catalog td {
    border-top: black thin
                dotted;
}
```

- Небольшой стиль можно набрать в одну строку (ради компактности):

```
.advertising { margin-left :30pt; }
```

- Иногда селекторы и открывающую фигурную скобку вводят в разных строках[2]:

```
p
{
    font-family: Arial;
    font-size: 14pt;
}
```

- Иногда набор атрибутов стиля и их значений оформляются в виде своего рода таблицы из двух столбцов вида:

```
p {
    font-family:      Arial;
    font-size:        14pt;
}
```

  Оформленный так код лучше читается, но имеет больший объем (за счет пробелов, добавленных для создания такой «таблицы»).

Как и в случае с веб-страницами, следует воздерживаться от вставки в код лишних пробельных символов, которые не всегда добавляют наглядности, но гарантированно увеличивают объем кода и, соответственно, замедляют его передачу по каналам связи (особенно медленным).

[2] На взгляд автора, это делает CSS-код более громоздким.

Неизвестные атрибуты стиля и их значения

Допускается использовать атрибуты стиля и их значения, не поддерживаемые веб-обозревателем (*неизвестные атрибуты стиля* и *значения*). При чтении CSS-кода веб-обозреватель их проигнорирует.

Так, в следующем примере неизвестный атрибут стиля `pretty-formatting` и неизвестное значение `fancy` атрибута стиля `border` будут проигнорированы, а атрибут стиля `border` получит значение по умолчанию:

```
.advertising {
    margin-left :30pt;
    pretty-formatting: very-pretty;
    border: fancy;
}
```

Однако неизвестные атрибуты стиля могут поддерживаться другими интернет-программами, например, службами онлайн-перевода (как и неизвестные HTML-теги).

11.3. Комментарии CSS

> ***Комментарий CSS***
>
> Фрагмент CSS-кода, не обрабатываемый веб-обозревателем. Применяется для внесения в код всевозможных заметок и примечаний.

Формат написания комментария:

```
/* <текст комментария> */
```

Текст комментария может быть разбит на строки и иметь произвольный размер. Между *текстом комментария* и ограничителями `/*` и `*/` допускается присутствие произвольного количества пробельных символов.

Короткие комментарии набираются в одной строке:

```
/* Не забыть задать у заголовка шрифт Verdana */
h1 { font-size: 32pt; }
```

Если комментарий велик, ограничители `/*` и `*/` располагают на отдельных строках, а текст комментария указывают между ними с отступом слева:

```
/*
    Задать у заголовка шрифт Verdana, выровнять текст по центру и указать
    форму букв нижнего регистра такую же, что и у букв верхнего регистра
*/
h1 { font-size: 32pt; }
```

11.4. Что, если на один элемент веб-страницы действуют несколько стилей? Правило каскадности

Поскольку в странице можно указать произвольное количество таблиц стилей, обычной является ситуация, когда на один элемент страницы действуют несколько стилей, в том числе задающие разные настройки оформления. Как в таком случае поступает веб-обозреватель?

Он объединяет настройки оформления из всех стилей, действующих на элемент и присутствующих во *всех* таблицах стилей, указанных в коде страницы, в один результирующий стиль, который и применяет к элементу. Если несколько разных стилей содержат одну и ту же настройку оформления с разными значениями, вступает в силу *правило каскадности*.

> ***Правило каскадности***
>
> Значение настройки берется из стиля, содержащего селектор с *наивысшим приоритетом*.

- Приоритет селектора стилевого класса выше, чем селектора тега, а приоритет комбинированного селектора еще выше. Вычисление приоритета селектора рассмотрено в *разд. 13.10*.

Пример*	Результат

`p, div { font-size: 9pt; }`
`.special { font-size: 16pt; font-weight: bold; }`
`p.special { font-weight: normal; }`
	`<div>Блок</div>`
	`<p>Абзац</p>`
	`<div class="special">`**Блок**`</div>`
	`<p class="special">`Абзац`</p>`

* Значение `normal` атрибута стиля `font-weight` задает обычную насыщенность шрифта.

- Последовательность, в которой в таблицах стиля записаны стили, роли не играет — значение имеет лишь приоритет их селекторов.

Пример	Результат

`p.special { font-weight: normal; }`
`.special { font-size: 16pt; font-weight: bold; }`
`p, div { font-size: 9pt; }`
	`<div>Блок</div>`
	`<p>Абзац</p>`
	`<div class="special">`**Блок**`</div>`
	`<p class="special">`Абзац`</p>`

- Если стили имеют одинаковый приоритет — настройка берется из стиля, записанного в таблицах стилей позже.

Пример	Результат

`p { font-size: 16pt; }`
`p { font-weight: bold; }`
`p { font-style: italic; }`
	`<p>`***Абзац***`</p>`

- Встроенные стили имеют наивысший приоритет.

Пример	Результат
`p { font-size: 16pt; }`	`<p>`Абзац`</p>` `<p style="font-size: 9pt;">`Абзац`</p>`

11.5. Важные атрибуты стиля

Любой атрибут стиля, записанный в каком-либо стиле, можно пометить как *важный*.

> ***Важный атрибут стиля***
>
> Атрибут стиля, значение которого имеет наивысший приоритет среди всех остальных стилей (даже наиболее высокоприоритетных — встроенных).

Чтобы превратить атрибут стиля в важный, следует вставить между его значением и завершающей точкой с запятой слово `!important` (восклицательный знак в начале ставится вплотную, без пробелов).

Пример:

```
p { font-size: 16pt!important; }
```

Для наглядности слово `!important` можно отделить от значения атрибута стиля пробельными символами:

```
p { font-size: 16pt !important; }
```

✓ Результат ⌄

```
<p style="font-size: 9pt;">Абзац</p>
```

Урок 12. Основные понятия CSS (завершение)

Функции CSS.
Единицы измерения CSS.
Задание цвета.
Переменные CSS.
Директивы CSS.

12.1. Способы задания значений у атрибутов стиля

Значение у любого атрибута стиля можно указать в виде константы, функции или специальной величины.

12.1.1. Константы

> ***Константа***
>
> Значение, записываемое непосредственно в CSS-коде и всегда остающееся неизменным.

Примеры атрибутов стиля со значениями, являющимися константами:

Атрибут стиля	Описание значения-константы
`font-size: 16pt;`	16 типографских пунктов
`color: red;`	Красный цвет
`text-align: right;`	Обозначение выравнивания текста по правому краю
`font-family: Verdana;`	Название шрифта Verdana

12.1.2. Функции CSS

Если значение какого-либо атрибута стиля должно вычисляться на основе заданных величин, следует применить одну из *функций*, поддерживаемых языком CSS.

> ***Функция***
>
> Языковая конструкция, выполняющая заданные действия над переданными ей параметрами и выдающая результат этих действий.
>
> ***Параметр***
>
> Значение, передаваемое функции для выполнения над ним вычислений.
>
> ***Возврат результата***
>
> Выдача функцией результата сделанных ею вычислений.

В качестве примера можно привести функцию `rgb()`, применяемую для указания цвета. Она принимает три параметра — доли, соответственно, красной, зеленой и синей составляющей в требуемом цвете, выраженные в виде чисел от 0 до 255. Результатом, возвращаемым этой функцией, станет созданный ею цвет.

Пример (в результате получится голубой цвет):

```
color: rgb(63, 255, 190);
```

Если требуется упомянуть о какой-либо функции в обычном тексте (например, руководству по языку CSS), имя этой функции записывается с пустыми круглыми скобками — например: `rgb()`. Это делается для того чтобы читатель сразу понял, что речь идет о функции (а не, скажем, об атрибуте стиля).

12.1.3. Специальные величины

Специальные величины предписывают атрибуту стиля либо использовать значение по умолчанию, либо взять значение у родителя. Специальные величины поддерживаются всеми атрибутами стиля:

- `initial` — задает у текущего атрибута стиля значение по умолчанию.

  Может оказаться полезным, поскольку разные атрибуты стиля имеют разные значения по умолчанию (`none`, `normal`, `baseline`, `auto`, `0` и др.), и запомнить их трудно, а у некоторых атрибутов стиля (скажем, `font-family`) значение по умолчанию определить вообще невозможно.

Пример	Результат
`p { font-size: 16pt; }` `p.special { font-size: initial; }`	`<p>Абзац 1</p>` `<p class="special">Абзац 2</p>`

- `inherit` — предписывает унаследовать значение текущего атрибута стиля у родителя. Поддерживается даже ненаследуемыми атрибутами стиля.

Пример*	Результат

`div {`
`  border: 1px solid black;`
`}`
`p.special {`
`  border: inherit;`
`}`
	`<div>`
	`  <p>Абзац 1</p>`
	`  <p class="special">Абзац 2</p>`
	`</div>`

*Атрибут стиля `border` — ненаследуемый, однако специальное значение `inherit` предписывает ему все же унаследовать значение от родителя.

- `unset` — задает у текущего атрибута стиля значение по умолчанию, если этот атрибут стиля ненаследуемый, или предписывает ему унаследовать значение у родителя, если он наследуемый.

Пример*	Результат

`div { font-size: 16pt; }`
`p {`
`  font-size: 9pt;`
`  border: 1px solid black;`
`}`
`p.spec {`
`  font-size: unset;`
`  border: unset;`
`}`
	`<div>`
	`  <p>Абзац 1</p>`
	`  <p class="spec">Абзац 2</p>`
	`</div>`

*Атрибут стиля `font-size` — наследуемый, поэтому, получив специальное значение `unset`, унаследует значение у родителя.

12.2. Типы значений атрибутов стилей

Каждый конкретный атрибут стиля принимает значение, относящееся к одному из шести следующих типов (независимо от способа указания этого значения — см. *разд. 12.1*):

- числовые — заданные в виде чисел, как правило, с какой-либо единицей измерения (кегль шрифта, размеры элементов, величины просветов, угол поворота, продолжительность анимации и др.);
- цветовые — обозначающие цвета (текста, фона, рамки и др.);
- ссылки на сторонние файлы (в качестве примера можно привести графические изображения, используемые в качестве фона);
- строковые — последовательности из произвольных символов;
- предопределенные (например, значение `right` атрибута стиля `text-align`, задающее выравнивание по правому краю);
- прочие — записываемые согласно правилам конкретного атрибута стиля (к ним относится, скажем, название шрифта).

Предопределенные и произвольные значения мы рассмотрим далее, изучая конкретные атрибуты стиля.

А о числовых, цветовых, ссылках на файлы и строковых значениях следует поговорить прямо сейчас.

12.2.1. Числовые значения

12.2.1.1. Форматы указания чисел в CSS

Целые числа в CSS записываются по правилам школьной математики. Примеры:

```
10, 16, 50, 1234567
```

Вещественные (дробные) числа записываются так же, за тем исключением, что в качестве десятичного разделителя используется *точка* (не запятая!):

```
1.5, 4.3, 1234.567
```

Если у вещественного числа целая часть равна `0`, ее можно не указывать:

`.5` (0,5), `.962` (0,962)

У отрицательных чисел впереди записывается символ - (минус, он же дефис):

`-10`, `-50`, `-123`, `-765.4321`, `-.5` (–0,5)

12.2.1.2. Единицы измерения CSS

Многие числовые значения могут быть выражены в различных единицах измерения. Так, ширина элемента может быть выражена в пикселах, типографских пунктах и множестве других единиц измерения размеров, поддерживаемых CSS.

Формат указания числового значения с единицей измерения таков:

<числовое значение><обозначение единицы измерения>

Значение и *обозначение единицы измерения* должны быть записаны вплотную. Какие бы то ни было пробельные символы между ними недопустимы.

Пример:

```
width: 500px;
```

Если задаваемое значение равно 0, единицу измерения указывать необязательно[1]:

```
margin: 0;
```

Единицы измерения, поддерживаемые CSS, приведены далее.

Некоторые числовые значения в CSS записываются без указания единицы измерения (например, множители или номера позиций в каком-либо порядке).

12.2.1.2.1. Единицы измерения геометрических величин

В этих единицах измерения выражаются размеры (включая кегль), значения просветов, толщины, расстояний и др. (табл. 12.1).

Таблица 12.1

Обозначение	Описание
Абсолютные	
px	Пикселы
pt	Типографские пункты
pc	Типографские пики
mm	Миллиметры
cm	Сантиметры
Q	$^1/_4$ миллиметра
in	Дюймы
Процентные	
%	Проценты от соответствующей настройки оформления родителя

[1] Что логично: в каких бы единицах ни был выражен ноль — он всегда останется нулем.

Таблица 12.1 *(продолжение)*

Обозначение	Описание
Относительные, основанные на кегле шрифта	
`em`	Кегль шрифта, указанного у текущего элемента
`rem`	Кегль шрифта, указанного у тега `<html>`
`ex`	Высота строчных букв у текущего элемента
`cap`	Высота прописных букв у текущего элемента
`ch`	Ширина цифры `0` у текущего элемента
`lh`	Высота строки у текущего элемента
`rlh`	Высота строки у тега `<html>`
Относительные, основанные на размерах *области просмотра* (внутренней области окна веб-обозревателя, в которой выводится страница)	
`svw`	$^1/_{100}$ ширины области просмотра в минимизированном виде — с выведенными боковыми панелями*
`svh`	$^1/_{100}$ высоты области просмотра в минимизированном виде — с выведенными верхними и нижними панелями*
`svmin`	Наименьшее из значений: `svw` или `svh`
`svmax`	Наибольшее из значений: `svw` или `svh`
`lvw`	$^1/_{100}$ ширины области просмотра в максимизированном виде — со скрытыми боковыми панелями*
`lvh`	$^1/_{100}$ высоты области просмотра в максимизированном виде — со скрытыми верхними и нижними панелями*
`lvmin`	Наименьшее из значений: `lvw` или `lvh`
`lvmax`	Наибольшее из значений: `lvw` или `lvh`
`dvw`	$^1/_{100}$ ширины области просмотра в текущем, минимизированном или максимизированном, виде**
`dvh`	$^1/_{100}$ высоты области просмотра в текущем, минимизированном или максимизированном, виде**
`dvmin`	Наименьшее из значений: `dvw` или `dvh`
`dvmax`	Наибольшее из значений: `dvw` или `dvh`
`vw`	$^1/_{100}$ ширины области просмотра в виде по умолчанию (каком именно, решает веб-обозреватель)***

Таблица 12.1 *(окончание)*

Обозначение	Описание
vh	$^{1}/_{100}$ высоты области просмотра в виде по умолчанию (каком именно, решает веб-обозреватель) ***
vmin	Наименьшее из значений: vw или vh
vmax	Наибольшее из значений: vw или vh

* Значения размеров, заданные в этих единицах измерения, остаются постоянными при изменении размеров области просмотра в результате вывода или скрытия верхних, нижних и боковых панелей.

** А значения размеров, заданные в этих единицах измерения, напротив, адаптируются к изменениям размеров области просмотра.

*** Эти единицы измерения появились раньше всех остальных единиц, основанных на размерах области просмотра. Они имеют существенный недостаток: невозможность точного указания вида области просмотра[2]. Вследствие чего сейчас применяются довольно редко (по большей части, в старых сайтах).

Примеры:

```
/* Кегль 24 пункта */
font-size: 24pt;

/* Ширина 20 см. */
width: 20cm;

/* Ширина 50% от ширины родителя */
width: 50%;

/*
    Кегль, равный 1/10 от высоты области просмотра
    в максимизированном виде
*/
font-size: 10lvh;
```

12.2.1.2.2. Единицы измерения углов

В этих единицах измерения задаются углы поворотов элементов при создании преобразований (см. *урок 29*).

Обозначение	Описание	Обозначение	Описание
deg	Градусы	rad	Радианы
grad	Грады	turn	Полные обороты вокруг оси

[2] Увы, даже в информационных технологиях первый блин часто выходит комом…

Пример:

```
/* Поворот элемента на 35° по часовой стрелке */
rotate: 35grad;

/* Поворот элемента на 1/2 оборота против часовой стрелки */
rotate: -.5turn;
```

12.2.1.2.3. Единицы измерения времени

В этих единицах измерения записывается продолжительность анимации (см. *урок 30*).

Обозначение	Описание	Обозначение	Описание
s	Секунды	ms	Миллисекунды

Пример:

```
/* Продолжительность анимации 5 сек. */
transition-duration: 5s;
```

12.2.1.3. Вычисление числовых значений. Функция *calc()*

Функция `calc()` вычисляет заданное *арифметическое выражение* и возвращает результат. Формат ее записи:

`calc(`*<арифметическое выражение>*`)`

Арифметическое выражение записывается согласно школьным правилам элементарной арифметики с применением *операторов*.

> ***Оператор***
>
> Обозначение арифметической операции.

Поддерживаются операторы: `+` (сложение), `-` (вычитание), `*` (умножение) и `/` (деление). Формат их записи:

<операнд 1> <оператор> <операнд 2>

Операнды могут быть выражены в любых поддерживаемых единицах измерения, в том числе и разных (например, *операнд 1* — в пунктах, а *операнд 2* — в миллиметрах).

Операторы `+` и `-` должны отделяться от *операндов* пробельными символами, операторы `*` и `/` отделять от *операндов* необязательно. У операторов `*` и `/` в качестве *операндов* допускается использовать обычные числа, без указания единиц измерения.

Примеры:

```
/* Ширина, равная сумме 24 пунктов и 40 пикселов  */
width: calc(24pt + 40px);

/*
    Ширина, равная половине ширины родителя за вычетом
    результата умножения 5 мм на 2
*/
width: calc(50% - 5mm * 2);

/*
    Внешний просвет слева, равный сумме 1 см, 4 пунктов и 14 пикселов
    за вычетом 1% ширины родителя
*/
margin-left: calc(1cm + 4pt + 12px - 1%);
```

Нулевые числовые значения, использованные в качестве операндов, необходимо записывать с обозначением единиц измерения:

```
width: calc(24pt + 0px);
```

Операторы `*` и `/` имеют более высокий приоритет, нежели операторы `+` и `-`, поэтому выполняются раньше. Повысить приоритет операторов `+` и `-` можно, заключив их, вместе с операндами, в круглые скобки.

Пример	Порядок выполнения	Результат
`calc(20px + 5px * 2)`	Сначала выполняется умножение 5 пикселов на 2, потом 20 пикселов складываются с полученным произведением	30 пикселов
`calc((20px + 5px) * 2)`	Сначала выполняется сложение 20 пикселов и 5 пикселов, потом получившаяся сумма умножается на 2	50 пикселов

Круглые скобки можно вкладывать друг в друга:

```
margin-left: calc((2cm * (5pt - 2px)) / 3);
```

Вместо круглых скобок допускается применение вложенных функций `calc()`:

```
margin-left: calc(calc(2cm * calc(5pt - 2px)) / 3);
```

Однако такая конструкция выглядит чересчур громоздко.

12.2.2. Цветовые значения

Любое цветовое значение может быть указано в виде:

- имени стандартного цвета[3]. Примеры:

```
background-color: black;                    /* Черный */
color: white;                               /* Белый */
color: blue;                                /* Синий */
background-color: green;                    /* Зеленый */
background-color: turquoise;                /* Бирюзовый */
border-color: red;                          /* Красный */
color: gold;                                /* Золотой */
```

- предопределенного значения `currentcolor` — цвет текста текущего элемента:

```
/* Цвет рамки такой же, как и цвет текста */
border: 1px solid currentcolor;
```

- значения формата `#<R><G><B>[<A>]` — где:
  - *R*, *G*, *B* — доли, соответственно, красной, зеленой и синей составляющих в требуемом цвете, представленные в виде двузначных шестнадцатеричных чисел от `00` до `ff`;
  - *A* — уровень непрозрачности цвета, также представленный в виде двузначного шестнадцатеричного числа от `00` (цвет полностью прозрачен) до `ff` (полностью непрозрачен). Если эта составляющая не указана, результирующий цвет станет полностью непрозрачным.

  Шестнадцатеричные числа можно набирать в любом регистре.

  Примеры:

```
color: #f8f12e;                     /* Ярко-желтый */
background-color: #2200ff;          /* Темно-синий */
background-color: #2200FF11;  /* Почти прозрачный темно-синий */
```

  Допускается указывать все четыре составляющие в виде однозначных шестнадцатеричных цифр от `0` до `f`. Тогда при формировании результирующего цвета каждая из этих цифр удвоится. Например, последние два цвета можно записать так:

```
background-color: #20f;
background-color: #20f1;
```

[3] Имена стандартных цветов, поддерживаемых CSS, приведены на странице: **https://developer.mozilla.org/en-US/docs/Web/CSS/named-color**.

- имени системного цвета[4]. Примеры:

```
/* Цвет текста такой же, как у надписей на кнопках */
color: ButtonText;
/* Цвет рамки элемента — как у рамок вокруг кнопок */
color: ButtonBorder;
```

- предопределенного значения `transparent` — прозрачный цвет:

```
/* Прозрачный фон */
background-color: transparent;
```

Для задания цветов также можно использовать следующие функции CSS:

- `rgb()` — возвращает цвет, составленный из заданных красной (*R*), зеленой (*G*), синей (*B*) составляющих и необязательной степени непрозрачности (*A*). *R*, *G* и *B* задаются в виде целых чисел от 0 до 255. Поддерживаются два формата записи:

```
rgb(<R>, <G>, <B>[, <A (в виде вещественного числа от 0.0 до 1.0)>])
rgb(<R> <G> <B>[ / <A (в процентах от 0 до 100)>])
```

  Примеры:

```
background-color: rgb(34, 0, 255);
background-color: rgb(34 0 255 / 5%);
```

- `rgba()` — то же самое, что и `rgb()`;
- `hsl()` — возвращает цвет, составленный из заданных оттенка (*H*), насыщенности (*S*), яркости (*L*) и необязательной степени непрозрачности (*A*). Поддерживаются два формата записи:

```
hsl(<H>, <S>, <L>[, <A (в виде вещественного числа от 0.0 до 1.0)>])
hsl(<H> <S> <L>[ / <A (в процентах от 0 до 100)>])
```

  H задается либо в виде целого числа без единицы измерения (и тогда измеряется в градусах), либо в виде числового значения с указанием одной из единиц измерения угла (см. *разд. 12.2.1.2.2*). *S* и *L* задаются в процентах.

  Вместо нулевого значения любого из четырех параметров можно указать слово `none`.

  Примеры:

```
background-color: hsl(34, 0%, 80%);              /* Светло-серый */
background-color: hsl(34deg none 80% / none);    /* То же самое */
```

- `hsla()` — то же самое, что и `hsl()`;

---

[4] Имена системных цветов, поддерживаемых CSS, приведены на странице: **https://developer.mozilla.org/en-US/docs/Web/CSS/system-color**.

- `hwb(<H> <S> <L>[ / <A>])` — возвращает цвет, составленный из заданных оттенка (*H*), доли белого (*W*), доли черного цвета (*B*) и необязательной степени непрозрачности (*A*).

  H задается так же, как и у функции `hsl()`, *W*, *B* и *A* — в процентах. Вместо нулевого значения любого из четырех параметров можно указать слово `none`. Пример:

  ```
  color: hwb(34deg 10% 70%);                    /* Темно-коричневый */
  ```

- `lch(<L> <C> <H>[ / <A>])` — возвращает цвет, составленный из заданных яркости (*L*), цветности (*C*), оттенка (*H*) и необязательной степени непрозрачности (*A*).

  H и `A` задаются так же, как и у функции `hsl()`, `W` и `B` — в процентах. Вместо нулевого значения любого из четырех параметров можно указать слово `none`. Пример:

  ```
  color: lch(10% 40% 34deg);                    /* Темно-бордовый */
  ```

12.2.3. Ссылки на сторонние файлы

Для указания *ссылки на сторонний файл* применяется функция `url()`, записываемая в формате:

`url(<`*ссылка на сторонний файл*`>)`

В качестве *ссылки* можно указать абсолютный, относительный путь к файлу или его полный интернет-адрес.

ВНИМАНИЕ!

Относительный путь к файлам отсчитывается:

- если записан во внешней таблице стилей — относительно папки, в которой хранится эта внешняя таблица стилей;
- если записан во внутренней таблице стилей или встроенном стиле — относительно папки, в которой хранится текущая страница.

Пример (атрибут стиля `background-image` задает графическое изображение в качестве фона):

```
background-image: url(../images/background.jpg);
```

Ссылку можно дополнительно заключить в одинарные или двойные кавычки:

```
background-image: url('../images/background.jpg');
background-image: url("../images/background.jpg");
```

12.2.4. Строковые значения

Строковое значение (*строка*) — это последовательность из произвольных символов, заключенная в одинарные или двойные кавычки[5]. В строках можно использовать любые символы, поддерживаемые кодировкой UTF-8. Примеры строк:

```
'*', 'Важное примечание!', "Need attention", "Ważna uwaga"
```

Чтобы вставить в строку символ с указанным *кодом*, следует использовать специальный символ `\<код>`. Это позволяет применять в строках символы, которые нельзя ввести с клавиатуры. Примеры:

`'\2713'` («✓»), `'БХВ\2122'` («БХВ™»), `"\221A 4 = 2"` («√ 4 = 2»)

12.3. Переменные CSS

> ***Переменная***
>
> Ячейка памяти с уникальным именем, способная хранить какое-либо значение. Это значение впоследствии можно передать атрибуту стиля.

Имя переменной должно содержать лишь буквы латиницы, цифры, символы дефиса, подчеркивания и обязательно должно начинаться с двойного дефиса (`--`). Примеры имен переменных:

```
--color, --nav-width, --header-font-size
```

Регистр букв, которым набрано имя переменной, имеет значение. Так, `--nav-width`, `--Nav-Width` и `--NAV-WIDTH` — это разные переменные.

> ***Объявление переменной***
>
> Создание переменной с заданным именем и занесение в нее значения.

Формат объявления переменной:

```
<имя переменной>: <значение переменной>;
```

Объявление переменных выполняется в каком-либо стиле, обычно низкоприоритетном — с селектором тега `<html>` или `<body>`.

> ***Обращение к переменной***
>
> Извлечение значения из переменной с целью передать его атрибуту стиля.

Обращение к переменной с указанным *именем* выполняется посредством функции `var()`, которая записывается в формате:

```
var(<имя переменной>[, <значение по умолчанию>])
```

[5] Автор предпочитает применять одинарные кавычки, поскольку их проще набирать.

Если переменная с заданным *именем* не была объявлена или хранит значение, не подходящее для этого атрибута стиля (например, переменная `--angle` хранит значение угла, а обращение к ней выполняется в атрибуте стиля `text-align`), функция возвращает указанное *значение по умолчанию* или, если таковое не задано — пустое значение.

Пример	Результат

`html { --para-font-size: 16pt; }`
`p {`
`  font-size: var(--para-font-size);`
`}`
	`<p>Абзац</p>`

Уже объявленную переменную можно объявить повторно, в стиле с более приоритетным селектором, указав у нее новое значение. Это значение будет извлекаться при обращении к этой переменной в еще более приоритетных стилях.

Пример	Результат

`html { --para-font-size: 16pt; }`
`p {`
`  font-size: var(--para-font-size);`
`}`
`div p { --para-font-size: 9pt; }`
`div p.special {`
`  font-size: var(--para-font-size);`
`}`
	`<p>Абзац</p>`
	`<div>`
	`  <p>Абзац</p>`
	`  <p class="special">`
	`  Абзац в блоке`
	`  </p>`
	`</div>`

12.4. Прочие инструменты CSS

12.4.1. Математические функции

CSS предоставляет ряд функций, выполняющих разнообразные математические действия:

- `min()` — возвращает наименьшее из указанных *значений*:
  `min(<значение 1>, <значение 2>, . . ., <значение N>)`

  В качестве любого из *значений* можно указать:

  - константу;
  - арифметическое выражение, подобно указываемому в функции `calc()` (см. *разд. 12.2.1.3*);

- другую функцию `min()` или `max()` (описана далее);
- одно из следующих слов: `pi` (число π), `e` (число e — основание натурального логарифма), `infinity` (∞ — математическая бесконечность) или `-infinity` (−∞).

Примеры:

```
width: min(50%, 200px);
width: min(50%, 200px - 2 * 2pt);
margin-left: calc(10pt + 2 * min(1em, 10px));
```

- `max()` — возвращает наибольшее из указанных значений. В остальном аналогична функции `min()`;
- `clamp()` — возвращает:
  - *предпочтительное значение* — если оно находится между *минимальным* и *максимальным*;
  - *минимальное значение* — если *предпочтительное* значение меньше *минимального*;
  - *максимальное значение* — если *предпочтительное значение* больше *максимального*.

  Формат записи:

  ```
  clamp(<минимальное значение>, <предпочтительное значение>,
                                <максимальное значение>)
  ```

  В качестве любого из *значений* можно использовать как константы, так и другие функции.

  Пример:

  ```
  width: clamp(100px, 10vw, 400px);
  ```

- `sin(<угол>)` — возвращает синус заданного *угла*. *Угол* может быть указан в виде константы с применением любой из поддерживаемых единиц измерений углов (см. *разд. 12.2.1.2.2*) или целого числа без единицы измерения — тогда угол будет измеряться в радианах. Пример:

  ```
  width: calc(sin(30deg) * 200px);
  ```

- `cos(<угол>)` — возвращает косинус заданного *угла*;
- `tan(<угол>)` — возвращает тангенс заданного *угла*;
- `asin(<значение>)` — возвращает арксинус указанного *значения*;
- `acos(<значение>)` — возвращает арккосинус указанного *значения*;
- `atan(<значение>)` — возвращает арктангенс указанного *значения*.

12.4.2. Получение сведений о платформе

Иногда бывает необходимо при оформлении страницы учесть какие-либо характеристики текущей компьютерной платформы. Например, если страница будет просматриваться на мобильном устройстве, экран которого имеет скругленные углы, следует выяснить координаты прямоугольной области, в которой можно располагать элементы, не опасаясь, что один из углов элемента окажется за границами экрана.

Получить подобного рода сведения, имеющие указанный *идентификатор*, позволяет функция `env()`:

`env(<идентификатор сведений о платформе>)`

Поддерживаются следующие *идентификаторы*:

- `safe-area-inset-left` — горизонтальный просвет между левой границей экрана и левой границей «безопасной» прямоугольной области, описанной ранее, в пикселах;
- `safe-area-inset-top` — вертикальный просвет между верхней границей экрана и верхней границей «безопасной» прямоугольной области в пикселах;
- `safe-area-inset-right` — горизонтальный просвет между правой границей экрана и правой границей «безопасной» прямоугольной области в пикселах;
- `safe-area-inset-bottom` — вертикальный просвет между нижней границей экрана и нижней границей «безопасной» прямоугольной области в пикселах.

Пример:

```
margin-left: env(safe-area-inset-left);
margin-top: calc(env(safe-area-inset-top) + 2px);
```

12.4.3. Импорт таблиц стилей. Директивы CSS

> ***Импорт таблицы стилей***
>
> Загрузка содержимого указанной внешней таблицы стилей и включение ее в состав текущей таблицы стилей.
>
> ***Директива***
>
> Команда, указывающая веб-обозревателю, как обрабатывать таблицу стилей или ее фрагмент.
>
> Имена директив в CSS всегда начинаются с символа `@`.

Для импорта внешней таблицы стилей из файла, хранящегося по заданной *ссылке*, применяется директива `@import`:

```
@import <ссылка на файл с импортируемой таблицей стилей>;
```

Ссылка на файл записывается с применением функции `url()` (см. *разд. 12.2.3*).

Пример:

```
@import url(vendor/photogallery/styles.css);
```

12.4.4. Атрибут стиля *all*

Атрибут стиля `all` обозначает все атрибуты стиля, поддерживаемые CSS. Он используется для указания у всех атрибутов стиля одинаковых значений, как правило, специальных величин: `initial`, `inherit` или `unset` (см. *разд. 12.1.3*).

Пример	Результат
`p { font-size: 16pt; }` `.special { all: initial; }`	`<p>Абзац 1</p>` `<p class="special">Абзац 2</p>`

Урок 13. Селекторы

Основные селекторы.
Комбинированный селектор.
Селекторы атрибутов тегов.
Псевдоклассы.
Псевдоэлементы.
Разделители.
Приоритет селектора и его вычисление.

Селектор записывается в составе стиля и указывает на группу элементов, к которой будет применен этот стиль.

13.1. Основные селекторы

> ***Основной селектор***
>
> Наиболее простой селектор. Может применяться как сам по себе, так и в составе комбинированного селектора.

Служит основой для построения комбинированного селектора.

Вот основные селекторы, поддерживаемые CSS (два из них нам уже знакомы по *уроку 2*):

- *селектор тега* — указывает на тег и записывается в виде имени этого тега без угловых скобок.

Пример	Результат
`p { font-weight: bold; }`	`<p>`**Абзац**`</p>` `<div>Блок</div>`

- *селектор стилевого класса* — указывает на стилевой класс и записывается в виде стилевого класса, предваренного точкой.

Пример	Результат

`.bolded { font-weight: bold; }`	`<p class="bolded">Абзац 1</p>`
	`<div class="bolded">Блок</div>`
	`<p>Абзац 2</p>`

- *селектор якоря* — указывает на якорь и записывается в виде якоря, предваренного символом решетки (#).

Пример	Результат
`#bolded { font-weight: bold; }`	`<p id="bolded">Абзац 1</p>` `<p>Абзац 2</p>`

- *универсальный селектор* — указывает на любой элемент страницы и записывается в виде символа звездочки (*).

Пример	Результат

`* { font-style: italic; }`	`<h3>Заголовок</h3>`
	`<p>Абзац</p>`
	`<div>Блок</div>`

13.2. Комбинированный селектор

Комбинированный селектор также знаком нам по *уроку 2*. На всякий случай — вот его определение.

> ***Комбинированный селектор***
>
> Комбинация более простых селекторов разных типов. Указывает на элементы, удовлетворяющие одновременно всем записанным в нем простым селекторам.

Формат написания комбинированного селектора:

<основные селекторы><селекторы по атрибутам тега><псевдоклассы><псевдоэлементы>

Селекторы по атрибутам тега, *псевдоклассы* и *псевдоэлементы* будут рассмотрены далее на этом уроке.

Отдельные простые селекторы записываются вплотную, без каких бы то ни было пробельных символов между ними.

Основные селекторы записываются в формате:

[*<селектор тега>*][*<селектор якоря>*][*<селектор стилевого класса>*]

Пример	Результат

p#emp.ital {
  font-weight: bold;
  font-style: *italic*;
}
	Абзац 1
	 Абзац 2
	 Абзац 3
	 Абзац 4

Указывать основные селекторы всех трех видов необязательно — достаточно одного или двух.

Пример	Результат
#emp { font-weight: **bold**; } *p.ital* { font-style: *italic*; }	**Абзац 1** *Абзац 2* Абзац 3

Допускается записывать несколько селекторов стилевых классов, опять-таки вплотную, без пробелов между ними. Такой селектор укажет на элементы, у которых заданы все перечисленные стилевые классы.

Пример	Результат

p.bolded.ital {
  font-weight: bold;
  font-style: *normal*;
}
	Абзац 1
	 Абзац 2
	 Абзац 3
	 Абзац 4

Не забываем, что стилевые классы в атрибуте тега `class` записываются через пробел.

13.3. Селекторы атрибутов тегов

> ***Селектор атрибута тега***
>
> Указывает на тег с атрибутом, имеющим заданное значение. Применяется только в комбинированных селекторах.

Вот список всех селекторов атрибутов, поддерживаемых CSS:

- [*<атрибут тега>*] — указывает на тег с заданным *атрибутом*. Значение *атрибута тега* может быть произвольным.

Пример	Результат

a[target] {
  font-weight: **bold;**
}
	
	  Гиперссылка 1
	
	
	  Гиперссылка 2
	
	
	  Гиперссылка 3
	

- [*<атрибут тега>=<значение>*] — указывает на тег с *атрибутом*, имеющим заданное *значение*.

ВНИМАНИЕ!

Здесь и далее, если заданное значение не содержит букв кириллицы, сравнение производится без учета регистра символов, в противном случае — с учетом регистра символов.

Пример	Результат

a[target=_blank] {
  font-weight: **bold;**
}
	
	  Гиперссылка 1
	
	
	  Гиперссылка 2
	  (регистр букв латиницы
	  не учитывается)
	
	
	  Гиперссылка 3
	
	
	  Гиперссылка 4
	

- [*<атрибут тега>*=<подстрока>*] — указывает на тег с *атрибутом*, чье значение содержит заданную *подстроку* (фрагмент строки).

Пример	Результат

div[title*=Альбом] {
  font-weight: **bold;**
}
	
	  Блок 1
	
	<div
	  title="Yesterday и Let It Be. Альбомы">
	  Блок 2
	
	  Блок 3
	  (регистр букв кириллицы учитывается)
	
	  Блок 4

- [*<атрибут тега>*~=*<слово>*] — указывает на тег с *атрибутом*, чье значение содержит заданное *слово* (словом является подстрока, ограниченная пробелами или знаками препинания).

Пример	Результат

div[title~=Альбом] {
  font-weight: **bold;**
}
	
	  Блок 1
	
	  Блок 2
	
	<div
	  title="Yesterday и Let It Be. Альбомы">
	  Блок 3

- [*<атрибут тега>*^=*<подстрока>*] — указывает на тег с *атрибутом*, чье значение начинается с заданной *подстроки*.

Пример	Результат

div[title^=Альбом] {
  font-weight: **bold;**
}
	
	  Блок 1
	
	<div
	  title="Альбомы Yesterday и Let It Be">
	  Блок 2
	
	  Блок 3

- [*<атрибут тега>*|=*<слово>*] — указывает на тег с *атрибутом*, чье значение начинается с заданного *слова*.

Пример	Результат

div[title|=Альбом] {
  font-weight: **bold;**
}
	
	  Блок 1
	
	<div
	  title="Альбомы Yesterday и Let It Be">
	  Блок 2
	
	  Блок 3

- [*<атрибут тега>*$=*<подстрока>*] — указывает на тег с *атрибутом*, чье значение заканчивается заданной *подстрокой*.

Пример	Результат

div[title$=Альбом] {
  font-weight: **bold**;
}
	
	  Блок 1
	
	<div
	  title="Альбомы Yesterday и Let It Be">
	  Блок 2

13.4. Псевдоклассы

> ***Псевдокласс***
>
> Указывает на элемент страницы, имеющий определенный признак (например, являющийся первым потомком своего родителя) или определенное состояние (скажем, находящийся под курсором мыши). Применяется только в комбинированных селекторах (за исключением нескольких псевдоклассов, рассмотренных отдельно).

Отличительной особенностью любого псевдокласса является наличие в его начале символа двоеточия (:).

CSS поддерживает довольно много псевдоклассов:

- :first-child — указывает на элемент, относящийся к заданному основным селектором типу и являющийся первым потомком своего родителя.

Пример	Результат

p:first-child {
  font-weight: **bold**;
}
	<article>
	  Абзац — первый потомок
	  Абзац — не первый потомок
	  Абзац — не первый потомок
	</article>
	<article>
	  Не абзац
	  Абзац — не первый потомок
	  Абзац — не первый потомок
	</article>

- `:first-of-type` — указывает на первый элемент, относящийся к заданному основным селектором типу.

Пример	Результат

`p:first-of-type {`
`  font-weight: bold;`
`}`
	`<article>`
	`  <p>Первый абзац</p>`
	`  <p>Не первый абзац</p>`
	`  <p>Не первый абзац</p>`
	`</article>`
	`<article>`
	`  <div>Не абзац</div>`
	`  <p>Первый абзац</p>`
	`  <p>Не первый абзац</p>`
	`</article>`

- `:last-child` — указывает на элемент, относящийся к заданному основным селектором типу и являющийся последним потомком своего родителя.

Пример	Результат

`p:last-child {`
`  font-weight: bold;`
`}`
	`<article>`
	`  <p>Абзац — не последний потомок</p>`
	`  <p>Абзац — не последний потомок</p>`
	`  <p>Абзац — последний потомок</p>`
	`</article>`
	`<article>`
	`  <div>Не абзац</div>`
	`  <p>Абзац — не последний потомок</p>`
	`  <address>Не абзац</address>`
	`</article>`

- `:last-of-type` — указывает на последний элемент, относящийся к заданному основным селектором типу.

Пример	Результат

`p:last-of-type {`
`  font-weight: bold;`
`}`
	`<article>`
	`  <div>Не абзац</div>`
	`  <p>Не последний абзац</p>`
	`  <p>Последний абзац</p>`
	`</article>`
	`<article>`
	`  <div>Не абзац</div>`
	`  <p>Последний абзац</p>`
	`  <address>Не абзац</address>`
	`</article>`

- `:nth-child(<N>)` — указывает на *N*-й элемент, относящийся к заданному основным селектором типу и являющийся *N*-м потомком своего родителя. В качестве *N* можно задать:
  - `<n>` (целое число) — укажет на *n*-й по счету элемент.

Пример	Результат

`p:nth-child(3) {`
`  font-weight: bold;`
`}`
	`<article>`
	`  <div>Не абзац</div>`
	`  <p>Абзац — 2-й потомок</p>`
	`  <p>`**`Абзац — 3-й потомок`**`</p>`
	`  <p>Абзац — 4-й потомок</p>`
	`  <p>Абзац — 5-й потомок</p>`
	`  <p>Абзац — 6-й потомок</p>`
	`</article>`
	`<article>`
	`  <div>Не абзац</div>`
	`  <p>Абзац — 2-й потомок</p>`
	`  <div>Не абзац</div>`
	`  <p>Абзац — 4-й потомок</p>`
	`  <p>Абзац — 5-й потомок</p>`
	`  <p>Абзац — 6-й потомок</p>`
	`</article>`

  - `odd`, `even` — укажет на каждый, соответственно, нечетный и четный элемент.

Пример	Результат

`p:nth-child(even) {`
`  font-weight: bold;`
`}`
	`<article>`
	`  <p>Абзац — нечетный потомок</p>`
	`  <p>`**`Абзац — четный потомок`**`</p>`
	`  <p>Абзац — нечетный потомок</p>`
	`  <h4>Не абзац</h4>`
	`  <p>Абзац — нечетный потомок</p>`
	`  <p>`**`Абзац — четный потомок`**`</p>`
	`  <p>Абзац — нечетный потомок</p>`
	`</article>`

  - формулу вида `<a>n + <b>`, где *a* и *b* — целые числа, а вместо `n` будут подставляться последовательно увеличивающиеся целые числа, начиная с `0`, — она укажет на элемент с номером, полученным в результате вычисления формулы.

Пример	Результат

`p:nth-child(2n + 1) {`
`  font-weight: bold;`
`}`
	`<article>`
	`  <p>Абз. — 1-й потомок (2×0+1=1)</p>`
	`  <p>Абзац — 2-й потомок</p>`
	`  <p>Абз. — 3-й потомок (2×1+1=3)</p>`
	`  <p>Абзац — 4-й потомок</p>`
	`  <div>Не абзац</div>`
	`  <p>Абзац — 6-й потомок</p>`
	`  <p>Абз. — 7-й потомок (2×3+1=7)</p>`
	`  <p>Абзац — 8-й потомок</p>`
	`</article>`

- `:nth-of-type(<N>)` — указывает на *N*-й элемент, относящийся к заданному основным селектором типу.

Пример	Результат

`p:nth-of-type(2n + 1) {`
`  font-weight: bold;`
`}`
	`<article>`
	`  <p>1-й абзац (2×0 + 1 = 1)</p>`
	`  <p>2-й абзац</p>`
	`  <p>3-й абзац (2×1 + 1 = 3)</p>`
	`  <p>4-й абзац</p>`
	`  <div>Не абзац</div>`
	`  <p>5-й абзац (2×2 + 1 = 5)</p>`
	`  <p>6-й абзац</p>`
	`  <p>7-й абзац (2×3 + 1 = 7)</p>`
	`</article>`

- `:nth-last-child(<N>)` — то же самое, что и `:nth-child()`, только отсчет элементов ведется с конца родителя;
- `:nth-last-of-type(<N>)` — то же самое, что и `:nth-of-type()`, только отсчет элементов ведется с конца родителя;
- `:only-child` — указывает на элемент, относящийся к заданному основным селектором типу и являющийся единственным потомком своего родителя.

Пример	Результат

`p:only-child {`
`  font-weight: bold;`
`}`
	`<footer>`
	`  <p>Абзац — единственный потомок</p>`
	`</footer>`
	`<footer>`
	`  <div>Не абзац</div>`
	`</footer>`
	`<footer>`
	`  <div>Не абзац</div>`
	`  <p>Абзац — не единственный потомок</p>`
	`</footer>`

- `:only-of-type` — указывает на единственный элемент, относящийся к заданному основным селектором типу.

Пример	Результат
`p:only-of-type { font-weight: bold; }`	`<footer><p>Единственный абзац</p></footer><footer><div>Не абзац</div></footer><footer><div>Не абзац</div><p>Единственный абзац</p></footer>`

- `:empty` — указывает на любой пустой элемент, не имеющий содержимого, даже текстового.

Пример	Результат
`/* Скрываем все пустые элементы */ *:empty { display: none; }`	`<!-- Эти элементы не будут выведены на экран --> <p></p> <div></div> <h1></h1> <p>А этот элемент — будет выведен!</p>`

- `:hover` — указывает на элемент, на который наведен курсор мыши:

```
div.interactive:hover { background-color: yellow; }
. . .
<div class="interactive">
    При наведении курсора мыши на этот блочный контейнер
    цвет его фона станет желтым
</div>
```

- `:lang(<обозначение языка>)` — указывает на все элементы с заданным *обозначением языка*. Обозначение языка задается в атрибуте тега `lang` (см. *разд. 4.5*).

Пример	Результат
`h1:lang(en) { font-weight: bold; }`	`<div lang="ru"><h1>Суши-бар Йокогама</h1><h1 lang="en">Yokogama Sushi Bar</h1></div>`

13.4.1. Псевдоклассы, используемые сами по себе

Как упоминалось ранее, некоторые псевдоклассы могут использоваться сами по себе. Вот они:

- `:not(<селектор>)` — указывает на все элементы, *не* соответствующие указанному в скобках *селектору*.

  В следующем примере стиль с селектором `:not()` применяется ко всем элементам, не являющимся абзацами.

Пример	Результат

`*:not(p) { font-weight: bold; }`	`<p>Абзац</p>`
	`<div>`**`Не абзац`**`</div>`
	`<address>`**`Не абзац`**`</address>`

- `:is()` — указывает на все элементы, которые соответствуют хотя бы одному из приведенных *селекторов*:
  `:is(<селектор 1>, <селектор 2>, . . ., <селектор N>)`

  Если в каком-либо стиле содержатся несколько селекторов, записанных через запятую, возможно, имеет смысл заменить их единственным селектором, содержащим псевдокласс `:is()`. В результате стиль станет компактнее и нагляднее.

  Это иллюстрируется приведенным далее примером. Он показывает два стиля, задающих одинаковое оформление: первый — с несколькими селекторами, второй — лишь с одним, содержащим псевдокласс `:is()`. Второй стиль несколько компактнее.

Пример	Результат

`div p, div div, div address { font-weight: bold; }`	`<div class="cls1">`
	`<p>`**`Абзац`**`</p>`
	`<div>`**`Блок`**`</div>`
	`<address>`**`Адрес`**`</address>`
	`<blockquote>`
	`Цитата`
	`</blockquote>`
	`</div>`

`div :is(p, div, address) { font-weight: bold; }`	

  При вычислении приоритета стиля, селектор которого включает псевдокласс `:is()`, веб-обозреватель учитывает приоритеты приведенных в нем *селекторов*.

  Это показывает следующий пример. В нем приоритет первого стиля, имеющий селектор, который содержит псевдокласс `:is()` с селектором

`p.bolded` в скобках, оказывается выше, чем у второго стиля, с селектором тега.

Пример	Результат
`:is(p.bolded, div.bolded) { font-weight: bold; } p { font-weight: normal; }`	`<div class="cls1"> <p>Абзац</p> <div>Блок</div> </div>`

- `:where()` — то же самое, что и `:is()`, только приоритет селекторов, приведенных в круглых скобках, *не* учитывается.

  В следующем примере приоритет первого стиля, с селектором `:where()`, оказывается ниже, чем у второго стиля.

Пример	Результат
`:where(p.bolded, div.bolded) { font-weight: bold; } p { font-weight: normal; }`	`<div class="cls1"> <p>Абзац</p> <div>Блок</div> </div>`

- `:root` — *корневой селектор*, указывает на корневой элемент страницы — тег `<html>`. Применяется для указания стилей, которые должны действовать на все содержимое страницы. Пример:

```
:root {
    color: darkgrey;
    background-color: white;
}
```

  Этот селектор *всегда* применяется сам по себе.

Еще несколько псевдоклассов мы изучим позднее.

13.5. Упражнение. Доработка каталогов и прайс-листа, часть 1

На страницах каталогов сайта о суши-баре описания блюд отделяются друг от друга пунктирными линиями. Вот только под последними позициями также присутствуют эти линии. Их нужно убрать.

Кроме того, на странице прайс-листа не помешает выровнять заголовки подразделов «Суши», «Сашими» и «Роллы» по центру.

1. Найдем в папке 10\s10.7 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Разделяющие линии мы создавали, указав у каждой из ячеек таблицы, которая формирует каталог, нижнюю сторону рамки в виде тонкой черной пунктирной линии. Тогда, чтобы убрать эту линию у последней позиции каталога, мы просто применим стиль к *ячейкам последней строки таблицы-каталога* и в этом стиле укажем отсутствие нижней стороны рамки.

2. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и добавим стиль, убирающий ненужную завершающую линию:

```
table.catalog tr:last-child td {
    border-bottom: none;
}
```

   В этом стиле мы применили составной селектор, состоящий из трех простых селекторов. Первый из простых селекторов укажет на таблицу со стилевым классом `catalog` (таблицу-каталог), второй — на последнюю строку этой таблицы (для чего применяется псевдокласс `:last-child`, недавно изученный нами), а третий — на ячейки этой строки. Значение `none` атрибута стиля `border-bottom` уберет нижнюю сторону рамки у элемента.

✓ Результат ⌄

Показана последняя позиция каталога суши и адрес.

Тобико

Икра летучей рыбы, рис, нори.

Наш адрес: ул. Зеленая, 17.

Заголовки подразделов в таблице прайс-листа находятся в ячейках шапки (тегах `<th>`), подвергнутых слиянию по горизонтали. Чтобы применить к этим ячейкам стиль, выравнивающий содержимое по центру, можно использовать селектор атрибута `colspan` (который и выполняет слияние по горизонтали).

3. Добавим стиль, выравнивающий заголовки подразделов по центру:

```
table.price-list th[colspan=2] {
    text-align: center;
}
```

Мы указали селектор, указывающий на ячейки заголовка, в которых присутствует атрибут тега `colspan` со значением 2.

✓ Результат ›

Товар	Цена, руб.
Суши	
Чукка	40
Магуро	80
Тобико	
Сашими	
Унаги	340
Хамачи	400
Роллы	
Кунсэй батакон	360
Аригато	
Мексиканский	
Всего 8 наименований	

13.6. Псевдоэлементы

> ***Псевдоэлемент***
>
> Указывает на часть содержимого какого-либо элемента (например, на его первый символ). Применяется только в комбинированных селекторах.

В начале любого псевдоэлемента присутствует двойной символ двоеточия (`::`).

Вот псевдоэлементы, поддерживаемые CSS:

- `::first-letter` — указывает на первый символ содержимого элемента.

Пример	Результат
`p::first-letter { font-size: 16pt; }`	`<p>`А`бзац</p>`

- `::first-line` — указывает на первую строку содержимого элемента.

Пример	Результат
`p::first-line { font-size: 16pt; }`	`<p>` Содержимое абзаца `</p>`

- `::marker` — указывает на маркер или нумерацию пункта списка.

Пример	Результат

`li::marker {`
`  font-size: 16pt;`
`}`
	`<ol>`
	`  1. <li>Суши.</li>`
	`  2. <li>Сашими.</li>`
	`  3. <li>Роллы.</li>`
	`</ol>`

- `::selection` — указывает на текст, выделенный посетителем.

Пример:

Пример	Результат

`p::selection {`
`  color: black;`
`  background-color: lightgrey;`
`}`
	Чукка
	Диетические суши, приготовляемые из одноименного салата, риса, кунжута и водорослей нори.

Еще несколько псевдоэлементов мы изучим позже.

13.7. Разделители

> ***Разделитель***
>
> Отделяет друг от друга простые селекторы, входящие в состав составного селектора.

Приведем список всех поддерживаемых CSS разделителей:

- пробел («пустой» разделитель) — *разделитель потомка*, давно нам знакомый. Указывает, что правый элемент должен быть либо потомком левого элемента, либо потомком его потомка, либо потомком потомка потомка и т. д.

Пример	Результат

`article p {`
`  font-weight: bold;`
`}`
	`<article>`
	`  <p>Абзац — потомок статьи</p>`
	`  <div>`
	`  <p>Абзац — потомок потомка статьи</p>`
	`  </div>`
	`  <div>Не абзац</div>`
	`</article>`

- > (знак «больше») — *разделитель непосредственного потомка*: правый элемент должен быть непосредственным потомком левого (но не потомком потомка, не потомком потомка потомка и т. д.):

Пример	Результат

`article > p { font-weight: bold; }`	`<article>`
	`  <p>Абзац — непосредств. потомок статьи</p>`
	`  <div>`
	`  <p>Абзац — не непосредств. потомок статьи</p>`
	`  </div>`
	`  <div>Не абзац</div>`
	`</article>`

- ~ (тильда) — *разделитель следующего соседа*: правый элемент должен быть первым, вторым, третьим и т. д. следующим соседом левого элемента.

Пример	Результат

`h1 ~ p { font-weight: bold; }`	`<h1>Заголовок</h1>`
	`<p>Абзац — следующий сосед заголовка</p>`
	`<p>Абзац — следующий сосед заголовка</p>`
	`<div>Не абзац</div>`
	`<p>Абзац — следующий сосед заголовка</p>`

- + (знак «плюс») — *разделитель первого следующего соседа*: правый элемент должен быть первым следующим соседом левого элемента (но не вторым, третьим, четвертым и т. д.).

Пример	Результат

`h1 + p { font-weight: bold; }`	`<h1>Заголовок</h1>`
	`<p>Абзац — первый следующий сосед заголовка</p>`
	`<p>Абзац — не первый след. сосед заголовка</p>`
	`<p>Абзац — не первый след. сосед заголовка</p>`

13.8. Псевдокласс *:has()*

Псевдокласс `:has(<селектор>)` указывает на элемент 1, внутри или возле которого присутствует элемент 2. Указываемый в нем *селектор* должен представлять собой комбинацию:

- из разделителя — обозначающего местоположение элемента 2 относительно элемента 1;
- из произвольного простого селектора — обозначающего элемент 2.

Пример	Результат
`article:has(> p) { border: black thin solid; }`	`<article>   <h3>Внутри статьи есть абзац!</h3>   <p>Абзац</p> </article>` `<article>   <h3>Внутри статьи нет абзаца</h3>   <div>Только блок</div> </article>`

Если в указанном *селекторе* используется разделитель потомка («пустой»), этот разделитель можно не указывать. Так, два следующих стиля полностью идентичны:

```
article:has( p) { border: black thin solid; }
article:has(p) { border: black thin solid; }
```

Также допускаются следующие форматы записи псевдокласса `:has()`:

- `:has(<селектор 1>, <селектор 2>, . . ., <селектор N>)` — указывает на элемент, содержащий в качестве потомка хотя бы один элемент, обозначаемый каким-либо из приведенных *селекторов*.

  Пример:

  ```
  article:has(p, div) { border: black thin solid; }
  ```

  Результаты:

  ```
  <article>
      <h3>Есть абзац и блок</h3>
      <p>Абзац</p>
      <div>Блок</div>
  </article>
  ```

  ```
  <article>
      <h3>Есть абзац</h3>
      <p>Абзац</p>
      <address>Адрес</address>
  </article>
  ```

  ```
  <article>
      <h3>Есть блок</h3>
      <div>Блок</div>
      <address>Адрес</address>
  </article>
  ```

  ```
  <article>
    <h3>Ни абзаца, ни блока</h3>
    <hr>
    <address>Адрес</address>
  </article>
  ```

- `:has(<селектор 1>):has(<селектор 2>). . .:has(<селектор N>)` — указывает на элемент, содержащий в качестве потомков все элементы, обозначаемые всеми приведенными *селекторами*.

Пример:

```
article:has(p):has(div) { border: black thin solid; }
```

Результаты:

```
<article>
    <h3>Есть абзац и блок</h3>
    <p>Абзац</p>
    <div>Блок</div>
</article>
```

```
<article>
    <h3>Только абзац</h3>
    <p>Абзац</p>
    <address>Адрес</address>
</article>
```

```
<article>
    <h3>Только блок</h3>
    <div>Блок</div>
    <address>Адрес</address>
</article>
```

```
<article>
  <h3>Ни абзаца, ни блока</h3>
  <hr>
  <address>Адрес</address>
</article>
```

13.9. Упражнение. Доработка каталогов и прайс-листа, часть 2

Немного оживим сайт суши-бара. На страницах каталогов у первых абзацев каждой позиции сделаем *буквицы*, а на главной странице маркеры пунктов обоих маркированных списков окрасим в более спокойный цвет и немного увеличим в размерах.

> ***Буквица***
>
> Увеличенная первая буква раздела, главы или всей страницы.

1. Найдем в папке 13\ex13.5 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Буквицу можно создать, просто увеличив кегль первой буквы первого абзаца, следующего за заголовком третьего уровня в каждой позиции каталога, например, в 1,5 раза. Чтобы сделать буквицы красивее, дополнительно закрасим их фон цветом `blanchedalmond` (которым при выполнении *упражнения 9.3* закрасили панель навигации).

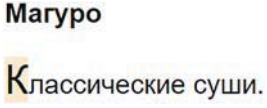
2. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и запишем стиль буквиц:

```
table.catalog h3 + p::first-letter {
    font-size: 150%;
    background-color: blanchedalmond;
}
```

Селектор этого стиля указывает на первый символ содержимого абзаца — первого соседа заголовка третьего уровня в таблице со стилевым классом `catalog`. Да-да, постепенно привыкаем писать длинные селекторы!..

Кегль шрифта, как и все остальные геометрические размеры, можно указывать в процентах от аналогичного параметра оформления родителя.

✓ Результат ›



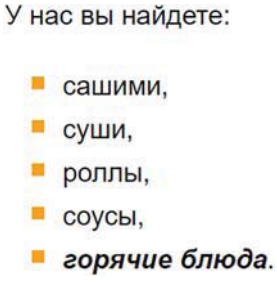
Неплохо. Но можно сделать и лучше.
И мы обязательно сделаем — на последующих уроках.

А пока займемся маркерами двух маркированных списков на главной странице. Окрасим их в оранжевый (`orange`) цвет и увеличим их кегль также в 1,5 раза.

3. Добавим стиль для маркеров пунктов в маркированных списках:

```
ul li::marker {
    font-size: 150%;
    color: orange;
}
```

✓ Результат ›



Маркеры тоже вышли неплохо.
Позже мы вернемся к ним...

13.10. Приоритет селектора и его вычисление

Стили, задающие одинаковое оформление элемента страницы, применяются согласно приоритету их селекторов — так гласит правило каскадности (см. *разд. 2.5*). Приоритет селектора вычисляется согласно следующим правилам:

- каждый селектор тега и псевдоэлемент, присутствующий в селекторе, считается за 1;
- каждый селектор стилевого класса, атрибута тега и псевдокласс — за 10;

- каждый селектор якоря — за 100;
- каждый универсальный селектор и разделитель — за 0;
- селектор, присутствующий в псевдоклассе `:is()`, — учитывается;
- селектор, присутствующий в псевдоклассе `:where()`, — не учитывается (считается за 0);
- полученные числа складываются — и их сумма покажет искомый приоритет.

Пример:

```
p#main_para { font-size: 12pt; }
* { font-size: 10pt; }
div p.special:only-child { font-size: 16pt; }
p { font-size: 14pt; }
. . .
<div>
    <p id="main_para" class="special">
        Шрифтом какого кегля будет выведен этот абзац?
    </p>
</div>
```

Рассчитаем приоритеты селекторов стилей, имеющихся в таблице стилей, в порядке, в котором они приведены:

- `p#main_para` (селектор тега и селектор якоря): 1 + 100 = 101;
- `*` (универсальный селектор): 0;
- `div p.special:only-child` (два селектора тега, селектор стилевого класса и псевдокласс): 2 × 1 + 10 + 10 = 22;
- `p` (селектор тега): 1.

В результате к абзацу будет применен первый стиль — с селектором `p#main_para`, — поскольку он имеет наивысший приоритет 101.

13.11. Самостоятельные упражнения

- На странице прайс-листа уберите у ячеек с ценами рамку и задайте у них светло-серый цвет фона (атрибут стиля `background-color` со значением `lightgrey`). В селекторе стиля для указания на эти ячейки используйте псевдокласс `:last-child` (поскольку это последние ячейки в строках).

✓ У вас должно получиться ›

Товар	Цена, руб.
Суши	
Чукка	40
Магуро	80
Тобико	
Сашими	
Унаги	340
Хамачи	400
Роллы	
Кунсэй батакон	360
Аригато	
Мексиканский	
Всего 8 наименований	

На приведенном рисунке видно, что некоторые ячейки с названиями блюд также оказались выделенными серым фоном. Эти ячейки являются единственными в своих строках, и веб-обозреватель также посчитал их последними.

- Исправьте указанный недочет, заменив псевдокласс `:last-child` комбинацией псевдоклассов `:last-child:not(td:only-child)` — то есть последний, но не единственный элемент в родителе.
- Увеличьте кегль шрифта, которым набраны цены в ячейках, подвергшихся слиянию по вертикали. Используйте для этого селектор по атрибуту тега `rowspan` и значение `larger` атрибута стиля `font-size`.

✓ У вас должно получиться ⌄

Товар	Цена, руб.
Суши	
Чукка	40
Магуро	80
Тобико	
Сашими	
Унаги	340
Хамачи	400
Роллы	
Кунсэй батакон	360
Аригато	
Мексиканский	
Всего 8 наименований	

Урок 14. Шрифт и текст

Гарнитура, кегль, насыщенность и начертание.
Цвет текста.
Загружаемые шрифты.
Выравнивание.
Тень у текста.
Управление переносом слов.
Генерируемое содержание.
Нумерация элементов.

14.1. Параметры шрифта

Все рассмотренные в этом разделе атрибуты стилей могут указываться у элементов любой разновидности: блочных, встроенных, встроенно-блочных и др.

14.1.1. Гарнитура, кегль, насыщенность и начертание шрифта

- `font-family` — наследуемый, задает гарнитуру шрифта. В качестве значения можно указать:
  - название шрифта:
    ```
    font-family: Arial;
    ```
    Если название шрифта включает пробелы, оно берется в одинарные или двойные кавычки:
    ```
    font-family: 'Times New Roman';
    ```
  - название одного из поддерживаемых *семейств шрифтов*: `serif` (шрифт с засечками), `sans-serif` (без засечек), `monospace` (моноширинный),

cursive, fantasy (две разновидности декоративных шрифтов), system-ui (основной шрифт, используемый в интерфейсе операционной системы), ui-sans-serif (интерфейсный шрифт без засечек), ui-serif (интерфейсный шрифт с засечками), ui-monospace (интерфейсный моноширинный) или ui-rounded (интерфейсный декоративный). Веб-обозреватель выберет первый из установленных на компьютере шрифтов, относящихся к заданному семейству. Пример:

```
font-family: sans-serif;
```

- несколько названий шрифтов или семейств шрифтов через запятую. Сначала веб-обозреватель попытается найти первый из указанных шрифтов, в случае неудачи попробует отыскать второй и т. д. Пример:

```
font-family: Tahoma, Arial, sans-serif;
```

По умолчанию веб-обозреватель выбирает гарнитуру шрифта самостоятельно;

- font-size — наследуемый, задает кегль шрифта. Возможные значения:
  - числовая величина кегля с использованием какой-либо единицы измерения, обычно пунктов:

```
font-size: 14pt;
```

  - одно из предопределенных значений: xx-small (самый маленький кегль), x-small, small, medium, large, x-large, xx-large или xxx-large (самый большой кегль). Конкретная величина кегля, задаваемая этими значениями, различается у разных веб-обозревателей. Пример:

```
font-size: small;
```

  - предопределенные значения: smaller или larger, задающие кегль, соответственно, на размер меньший или больший унаследованного от родителя:

```
div { font-size: x-large; }
address { font-size: smaller; }
. . .
<div>
   <p>Шрифт кегля x-large</p>
   <address>Шрифт кегля large</address>
</div>
```

Значение по умолчанию: medium;

- `font-weight` — наследуемый, задает насыщенность шрифта. Поддерживаемые значения:
  - `normal` — обычная насыщенность:

    ```
    font-weight: normal;
    ```

  - `bold` — **полужирный** шрифт:

    ```
    font-weight: bold;
    ```

  - одно из предопределенных значений: 100, 200, 300, 400 (шрифт обычной насыщенности), 500, 600, 700 (полужирный шрифт), 800 или 900:

    ```
    font-weight: 200;
    ```

    Однако нужно иметь в виду, что значения, отличные от 400 и 700, поддерживаются не всеми шрифтами;

  - `lighter` или `bolder` — насыщенность, соответственно, на ступень меньше или больше унаследованной от родителя:

    ```
    div { font-weight: 400; }
    address { font-weight: bolder; }
    . . .
    <div>
        <p>Шрифт насыщенности 400</p>
        <address>Шрифт насыщенности 500</address>
    </div>
    ```

  Значение по умолчанию: `normal` (у большинства тегов) или `bold` (у заголовков, очень важного шрифта, ячеек шапки и некоторых других тегов);

- `font-style` — наследуемый, начертание шрифта. Доступные значения:
  - `normal` — шрифт обычного начертания;
  - `italic` — *курсивный* шрифт;
  - `oblique` — шрифт декоративного начертания с углом наклона по умолчанию (поддерживается не всеми шрифтами);
  - `oblique` <*угол наклона*> — шрифт декоративного начертания с заданным *углом наклона* (поддерживается не всеми шрифтами):

    ```
    font-style: obligue 45deg;
    ```

  Значение по умолчанию: `normal` (у почти всех тегов) или `italic` (у важного текста и некоторых других тегов).

14.1.2. Цвет текста

Для задания цвета текста используется наследуемый атрибут стиля `color`. В нем можно указать значение цвета в любом из поддерживаемых CSS форматах (см. *разд. 12.2.2*). Пример:

```
color: red;
```

Значение по умолчанию: `CanvasText` (системный цвет текста).

14.1.3. Преобразование букв текста

Для преобразования букв текста в другой регистр применяется наследуемый атрибут стиля `text-transform`. Вот поддерживаемые им значения:

- `none` — отсутствие какого-либо преобразования;
- `capitalize` — Первая Буква Каждого Слова Приводится К Верхнему Регистру;
- `uppercase` — ВСЕ БУКВЫ ПРИВОДЯТСЯ К ВЕРХНЕМУ РЕГИСТРУ;
- `lowercase` — все буквы приводятся к нижнему регистру.

Значение по умолчанию: `none`.

14.1.4. Плотность шрифта

Для указания плотности шрифта следует использовать наследуемый атрибут стиля `font-stretch`. В качестве значения можно указать:

- непосредственно величину плотности в процентах от 50 до 200:

```
address { font-stretch: 150%; }
blockquote { font-stretch: 75%; }
```

- одно из предопределенных значений: `ultra-condensed` (плотность 50%, наиболее плотный шрифт), `extra-condensed` (62,5%), `condensed` (75%), `semi-condensed` (87,5%), `normal` (100%, шрифт обычной плотности), `semi-expanded` (112,5%), `expanded` (125%), `extra-expanded` (150%) или `ultra-expanded` (200%, наиболее разреженный шрифт):

```
address { font-stretch: extra-expanded; }
blockquote { font-stretch: condensed; }
```

Значение по умолчанию: `normal`.

Следует отметить, что не все шрифты поддерживают разные значения плотности.

14.1.5. Форма символов

Есть возможность указать у текста другую форму букв или цифр, воспользовавшись следующими атрибутами стиля:

- `font-variant-caps` — наследуемый, указывает форму строчных букв. Доступные значения:
  - `normal` — обычная форма;
  - `small-caps` — строчные буквы выводятся в виде *капителей* (уменьшенных прописных букв). Прописные буквы имеют увеличенный кегль;
  - `all-small-caps` — то же самое, что и `small-caps`, только прописные буквы имеют тот же кегль, что и строчные.

  Значение по умолчанию: `normal`;
- `font-variant-numeric` — наследуемый, указывает форму цифр.

Значение	Описание	Пример
`normal`	Обычная форма цифр	1234567890, 5.00078
`slashed-zero`	Нуль либо с точкой внутри, либо перечеркнутый[1]	1234567890, 5.00078
`oldstyle-nums`	Цифры в «старом» стиле	1234567890, 5.00078

Значение по умолчанию: `normal`.

Для одновременного указания формы и строчных букв, и цифр можно использовать наследуемый атрибут стиля `font-variant`. В качестве значения у него можно указать одно из двух:

- `normal` — обычная форма букв и цифр;
- [<*форма строчных букв* (`font-variant-caps`)>]
  [<*форма цифр* (`font-variant-numeric`)>]

Пример:

```
font-variant: small-caps oldstyle-nums;
```

14.1.6. Декоративная линия

> ***Декоративная линия***
>
> Рисуется под текстом (подчеркивание), над текстом (надчеркивание) или поперек текста (зачеркивание).

[1] Чтобы его сразу можно было отличить от буквы «О».

- `text-decoration-line` — ненаследуемый, задает вид декоративной линии у текста. Доступные значения:
  - `none` — отсутствие какой-либо декоративной линии;
  - `underline` — подчеркивание;
  - `overline` — надчеркивание;
  - `line-through` — ~~зачеркивание~~.

  Значение по умолчанию: `none` (у всех элементов, кроме гиперссылок) или `underline` (у гиперссылок);
- `text-decoration-style` — ненаследуемый, форма декоративной линии в виде одного из значений:
  - `solid` — одинарная сплошная линия;
  - `double` — двойная сплошная;
  - `dotted` — пунктирная;
  - `dashed` — штриховая;
  - `wavy` — волнистая.

  Значение по умолчанию: `solid`;
- `text-decoration-color` — ненаследуемый, цвет декоративной линии в виде любого цветового значения, поддерживаемого CSS (см. *разд. 12.2.2*). По умолчанию: `currentcolor` (то есть текущий цвет текста);
- `text-decoration-thickness` — ненаследуемый, толщина декоративной линии в виде:
  - числового значения в любой из поддерживаемых единиц измерения (см. *разд. 12.2.1.2.1*);
  - `auto` — веб-обозреватель сам выбирает толщину линии;
  - `from-font` — если текущий шрифт содержит сведения о толщине линии, будут использованы они, в противном случае веб-обозреватель сам выберет толщину линии.

  Примеры:

```
text-decoration-thickness: 1pt;
text-decoration-thickness: 6pt;
```

- `text-decoration-skip-ink` — наследуемый, указывает, будет ли декоративная линия рисоваться поверх свисаний и выносных элементов букв.

Значение	Описание	Пример
auto	Будет рисоваться под свисаниями и выносными элементами	Пример
none	Будет рисоваться поверх свисаний и выносных элементов	Пример

Значение по умолчанию: auto.

Принимается во внимание только в случае подчеркивания и надчеркивания (если атрибуту стиля text-decoration-line дано значение underline или overline);

- text-underline-offset — наследуемый, величина просвета между линией подчеркивания и базовой линией текста.

> ***Базовая линия***
>
> Воображаемая прямая линия, проходящая по нижнему краю символов без учета свисаний и нижних выносных элементов.

Значение можно указать в виде:

- числового значения в любой из поддерживаемых единиц измерения (см. *разд. 12.2.1.2.1*). Можно указать нулевое или отрицательное значение;
- auto — величину просвета установит веб-обозреватель.

Значение по умолчанию: auto.

Принимается во внимание только в случае подчеркивания (если атрибуту стиля text-decoration-line дано значение underline).

Пример	Результат
text-underline-offset: auto;	Пример
text-underline-offset: 0;	Пример
text-underline-offset: 10pt;	Пример
text-underline-offset: -7pt;	П-р-и-м-е-р

- `text-underline-position` — наследуемый, местоположение линии подчеркивания:
  - `auto` — местоположение задаст веб-обозреватель;
  - `from-font` — если текущий шрифт содержит сведения о местоположении линии подчеркивания, будут использованы они, в противном случае веб-обозреватель сам установит местоположение линии;
  - `under` — строго ниже всех свисаний и нижних элементов букв.

  Значение по умолчанию: `auto`.

  Принимается во внимание только в случае подчеркивания (если атрибуту стиля `text-decoration-line` дано значение `underline`).

Пример	Результат
`text-underline-position: auto;`	Пример
`text-underline-position: under;`	Пример

Также имеется возможность указать сразу несколько параметров декоративной линии, воспользовавшись ненаследуемым атрибутом стиля `text-decoration`. Его значение записывается в формате:

<*вид линии* (text-decolation-line)> [<*цвет линии* (text-decoration-color)>]
[<*форма линии* (text-decoration-style)>]
[<*толщина линии* (text-decoration-thickness)>]

Пример	Результат
`text-decoration: underline;`	Пример
`text-decoration: overline green 3pt;`	Пример
`text-decoration: overline green wavy 3pt;`	Пример

14.2. Загружаемые шрифты

> ***Загружаемый шрифт***
>
> Шрифт, который хранится в файле в составе сайта и загружается веб-обозревателем согласно указаниям, записанным в таблице стилей.

Форматы загружаемых шрифтов, поддерживаемые веб-обозревателями (табл. 14.1).

Таблица 14.1

Название	Описание	Расширение файлов	Рекомендован к применению?
TrueType Format (TTF)	Наиболее старый формат	ttf	Нет
OpenType Format (OTF)	Усовершенствованный TTF	otf	
Web Open Font Format 1.0 (WOFF 1.0)	OTF с поддержкой сжатия	woff	Да
Web Open Font Format 2.0 (WOFF 2.0)	Улучшенное сжатие по сравнению с WOFF 1.0		

Чтобы веб-обозреватель смог использовать загружаемый шрифт для вывода текста, следует записать в таблице стилей указание загрузить его. Для этого применяется директива `@font-face`, записываемая в формате:

```
@font-face {
    <атрибуты стиля, описывающие загружаемый шрифт>
}
```

Атрибуты стиля, указываемые в этой директиве, записываются так же, как и обычные атрибуты стиля.

Атрибут стиля `src` задает ссылку на файл, в котором хранится шрифт. Запись ссылок на файлы рассмотрена в *разд. 12.2.3*.

Атрибут стиля `font-family` в директиве `@font-face` указывает название загружаемого шрифта, под которым его можно будет в дальнейшем использовать для вывода текста. Это название можно выбрать произвольно. Если название содержит пробелы, его следует взять в одинарные или двойные кавычки.

Обычно в одном файле хранится лишь один вариант шрифта — какого-либо одного начертания (обычного, курсивного или декоративного), имеющий одно значение насыщенности (обычный или полужирный) и плотности (например, `normal` или `expanded`). Иногда в файле хранятся несколько вариантов шрифтов — с разными значениями насыщенности и плотности, укладывающимися в определенный диапазон, но, опять же, одного начертания.

Веб-обозреватель самостоятельно не может «узнать» параметры шрифта, загруженного из файла директивой `@font-face`: начертание, значение или

диапазон значений насыщенности и плотности. Эти параметры необходимо записать в упомянутой директиве явно, использовав следующие атрибуты стиля:

- `font-style` — начертание шрифта: `normal`, `italic` или `oblique` (подробности — в *разд. 14.1.1*).

  У шрифта декоративного (`oblique`) начертания можно указать доступное значение угла наклона:

  ```
  font-style: oblique 45deg;
  ```

  Также можно задать диапазон значений углов, записав минимальное и максимальное значения через пробел:

  ```
  font-style: oblique 15deg 45deg;
  ```

- `font-weight` — насыщенность шрифта (подробности — в *разд. 14.1.1*). Можно указать как одно значение, так и диапазон, записав минимальную и максимальную насыщенность через пробел;
- `font-stretch` — плотность шрифта (см. *разд. 14.1.4*). Можно указать как одно значение, так и диапазон, записав минимальную и максимальную плотность через пробел.

Пример загрузки из двух разных файлов двух вариантов шрифта SomeFont — обычной насыщенности и полужирного:

```
@font-face {
    src: url(../fonts/SomeFont-Regular.woff);
    font-family: SomeFont;
    font-style: normal;
    font-weight: normal;
}
@font-face {
    src: url(../fonts/SomeFont-Bold.woff);
    font-family: SomeFont;
    font-style: normal;
    font-weight: bold;
}
```

Загруженный шрифт можно использовать так же, как и обычный:

```
p { font-family: SomeFont; }
.bolded { font-weight: bold; }
. . .
<p>Шрифт SomeFont обычного начертания</p>
<p class="bolded">Шрифт SomeFont полужирного начертания</p>
```

Пример загрузки шрифта UniversalFont, имеющего курсивное начертание и поддерживающего диапазоны значений насыщенности и плотности:

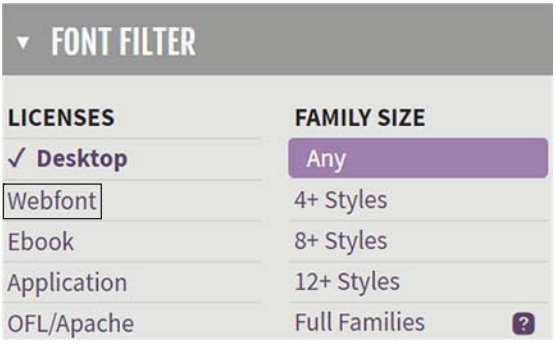
```
@font-face {
    src: url(../fonts/UniversalFont.woff);
    font-family: UniversalFont;
    font-style: italic;
    font-weight: 200 700;
    font-stretch: 50% 150%
}
```

14.3. Упражнение. Использование загружаемых шрифтов

Все-таки наш сайт суши-бара скучноват... Давайте оформим названия блюд, приведенных на страницах каталогов, красивым загружаемым шрифтом Caviar Dreams.

1. Найдем в папке 13\s13.11 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

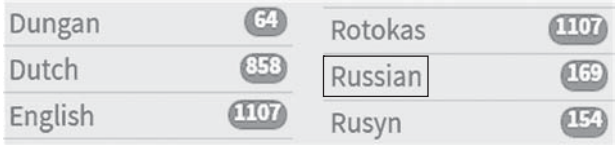
   Сначала загрузим файлы, в которых хранятся разные варианты нужного шрифта. Для чего посетим один чудесный сайт.
2. Перейдем на сайт Font Squirrel[2] — крупное хранилище загружаемых шрифтов, в том числе и бесплатных.
3. Прокрутим вниз открывшуюся главную страницу, пока в ее правой колонке не покажется графа **Font Filter** (**Фильтр шрифтов**), и щелкнем на расположенной в ней гиперссылке **Webfont** (**Веб-шрифт**), чтобы вывести только шрифты интернет-форматов.



[2] См. **https://www.fontsquirrel.com/**.

4. Отыщем расположенную ниже графу **Languages** (**Языки**), организованную в виде спойлера, и, если она свернута, щелкнем на ней, чтобы развернуть.

5. В открывшемся под заголовком графы весьма обширном списке найдем гиперссылку **Russian** (**Русский**) и щелкнем на ней, чтобы вывести только русскоязычные шрифты.



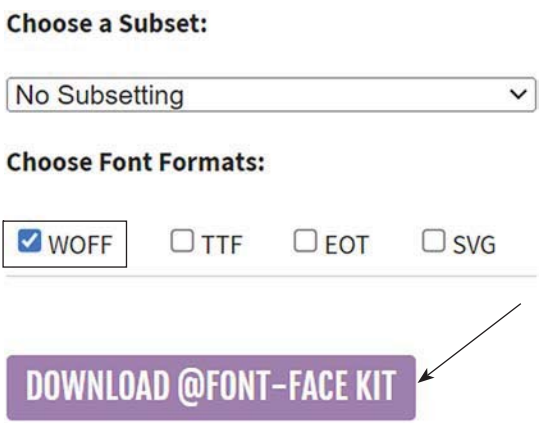
6. Прокрутим главную страницу вверх, найдем поле ввода **Search Fonts & Foundries** (**Искать шрифты и подборки шрифтов**), занесем в него название искомого шрифта и щелкнем на кнопке поиска, расположенной правее поля ввода и содержащей изображение лупы.

В левой колонке появится шрифт Caviar Dreams.

Видно, что этот шрифт доступен в четырех вариантах: обычном, полужирном, курсивном и полужирном курсивном.

7. Переключимся на вкладку **Webfont Kit** (**Набор веб-шрифтов**), где можно загрузить шрифт в формате WOFF.

На появившейся вкладке будет находиться веб-форма для указания параметров загружаемого шрифта.



8. В раскрывающемся списке **Choose a Subset** (**Выберите подмножество**) выберем пункт **No Subsetting** (здесь: **Не делить шрифт на подмножества**).

   Если этого не сделать, сайт выдаст нам шрифт без букв кириллицы, а это нам совсем не нужно.

9. В наборе **Choose Font Formats** (**Выберите форматы шрифтов**) — установим флажок **WOFF** и сбросим остальные флажки.
10. Нажмем кнопку **Download @font-face Kit** (**Загрузить набор для директивы `@font-face`**), чтобы загрузить шрифт.
11. Распакуем полученный ZIP-архив в какую-либо папку.

    В результате получим следующую структуру папок и файлов (вложенность показана отступами слева, ненужные файлы не указаны):

```
web fonts
    caviardreams_bold
        Caviar_Dreams_Bold-webfont.woff
    caviardreams_bolditalic
        CaviarDreams_BoldItalic-webfont.woff
    caviardreams_italic
        CaviarDreams_Italic-webfont.woff
    caviardreams_regular
        CaviarDreams-webfont.woff
```

В папку web fonts вложены четыре папки, хранящие WOFF-файлы с полужирным (bold), полужирным курсивным (bolditalic), курсивным (italic) и обычным (regular) вариантами шрифта.

12. Создадим в папке site папку fonts и переместим в нее из упомянутых ранее папок файлы шрифтов Caviar_Dreams_Bold-webfonts.woff, CaviarDreams_BoldItalic-webfonts.woff, CaviarDreams_Italic-webfonts.woff и CaviarDreams-webfonts.woff.

    Остальные файлы и папки из полученного архива можно удалить, поскольку они не нужны.

13. Сократим имена упомянутых ранее файлов с разными вариантами шрифта, убрав из них суффикс -webfonts.

> ***Пояснение***
>
> Чем короче имя файла или иной сущности, тем быстрее оно набирается на клавиатуре.

14. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и вставим в самое начало CSS-код, загружающий все четыре варианта шрифта Caviar Dreams:

```
@font-face {
    font-family: 'Caviar Dreams';
    src: url(../fonts/CaviarDreams.woff);
}
@font-face {
    font-family: 'Caviar Dreams';
    src: url(../fonts/Caviar_Dreams_Bold.woff);
    font-weight: bold;
}
@font-face {
    font-family: 'Caviar Dreams';
    src: url(../fonts/CaviarDreams_Italic.woff);
    font-style: italic;
}
@font-face {
    font-family: 'Caviar Dreams';
    src: url(../fonts/CaviarDreams_BoldItalic.woff);
    font-weight: bold;
    font-style: italic;
}
```

Теперь мы можем использовать загруженный шрифт для вывода названий блюд в каталогах.

Чтобы добавить немножко стильности, укажем у названий блюд вывод строчных букв в виде капителей и одинаковый кегль у прописных и строчных букв. Также увеличим кегль шрифта названий до 32 пунктов — шрифт Caviar Dreams мелковат...

15. Добавим необходимый стиль:

```
table.catalog h3 {
    font-family: 'Caviar Dreams';
    font-size: 32pt;
    font-variant-caps: all-small-caps;
}
```

✓ Результат ›

Ну не красота ли?

Бесплатные и платные шрифты

Часть шрифтов, опубликованных на сайте Font Squirrel, являются платными. Если щелкнуть на синей гиперссылке, появится страница покупки такого шрифта.

14.4. Параметры строк текста

14.4.1. Интервалы, отступ и высота строки

Два следующих атрибута стиля, управляющие интервалами, также могут быть указаны у элементов любой разновидности:

- `letter-spacing` — наследуемый, указывает дополнительный интервал между символами текста. В качестве значения можно задать:
  - `normal` — дополнительный интервал между символами не добавляется.

Пример	Результат
`letter-spacing: normal;`	Текст обычной плотности

- числовую величину в какой-либо единице измерения.

Пример	Результат
`letter-spacing: 4pt;`	Р а з р е ж е н н ы й т е к с т

Можно указать отрицательное значение — тогда символы будут располагаться ближе друг к другу:

Пример	Результат
`letter-spacing: -1pt;`	Уплотненный текст

Значение по умолчанию: `normal`;

- `word-spacing` — наследуемый, указывает дополнительный интервал между словами. Поддерживаются значения:
  - `normal` — дополнительный интервал между словами не добавляется;
  - числовая величина в какой-либо единице измерения:

Пример	Результат
`word-spacing: 5mm;`	Большой интервал между словами

Значение по умолчанию: `normal`.

Следующие два атрибута стиля применимы лишь к блочным элементам:

- `text-indent` — наследуемый, задает отступ красной строки. Значение указывается в виде числовой величины в какой-либо единице измерения.

Пример	Результат
`text-indent: 1cm;`	Отступ красной строки равный 1 см.

Значение по умолчанию: `0`;

- `line-height` — наследуемый, указывает высоту строки, опосредованно задавая интервал между строками. Поддерживаются следующие значения:
  - `normal` — обычная высота строки (разная у различных веб-обозревателей);
  - числовая величина в какой-либо единице измерения.

Примеры	Результаты
`font-size: 12pt;` `line-height: 24pt;`	У этого абзаца задана высота строки, равная 24 пунктам
`font-size: 12pt;` `line-height: 12pt;`	У этого абзаца задана высота строки, равная 12 пунктам

- числовой множитель без указания единицы измерения. Высота строки будет получена умножением этого множителя на кегль шрифта.

Пример	Результат
`font-size: 12pt;` `line-height: 1.5;`	У этого абзаца задана высота строки, равная $12 \times 1{,}5 = 18$ пунктам

Значение по умолчанию: `normal`.

14.4.2. Выравнивание по горизонтали и вертикали

Выравнивание содержимого по горизонтали можно указать лишь у блочных элементов. Оно задается следующими наследуемыми атрибутами стиля:

- `text-align` — выравнивание всех строк текста в виде одного из следующих значений:
  - `left` или `start` — по левому краю;
  - `center` — по центру;
  - `right` или `end` — по правому краю;
  - `justify` — по ширине.

  Значение по умолчанию: `left` (у всех элементов, кроме ячеек шапки) или `center` (у ячеек шапки);
- `text-align-last` — выравнивание последней строки текста:
  - `auto` — такое же, как и у остальных строк текста (задается атрибутом стиля `text-align`);
  - `left`, `start`, `center`, `right`, `end` или `justify` — см. ранее.

  Значение по умолчанию: `auto`.

Пример	Результат
`text-align: justify;`	Выравнивание содержимого по горизонтали можно указать лишь у блочных элементов
`text-align: justify;` `text-align-last: right;`	Выравнивание содержимого по горизонтали можно указать лишь у блочных элементов
`text-align: justify;` `text-align-last: justify;`	Выравнивание содержимого по горизонтали можно указать лишь у блочных элементов

Напротив, вертикальное выравнивание можно указать у встроенных и встроенно-блочных элементов. Управляя вертикальным выравниванием, мы можем выровнять буквицы и большие изображения, вставленные в текст, относительно верха или низа строк этого текста.

Вертикальное выравнивание задается ненаследуемым атрибутом стиля `vertical-align`, который поддерживает следующие значения (табл. 14.2).

Таблица 14.2

Значение	Пример	Результат
`baseline` — выравнивание по базовой линии	`p::first-letter { vertical-align: baseline; }`	Диетические
`top` — выравнивание по самому высокому символу в строке	`p::first-letter { vertical-align: top; }`	Диетические
`text-top` — по самому высокому символу, что имеется в шрифте	`p::first-letter { vertical-align: text-top; }`	Диетические
`middle` — по середине	`p::first-letter { vertical-align: middle; }`	Диетические
`bottom` — по самому низкому символу в строке	`p::first-letter { vertical-align: bottom; }`	Диетические
`text-bottom` — по самому низкому символу, что имеется в шрифте	`p::first-letter { vertical-align: text-bottom; }`	Диетические
`super` — верхний индекс	`.digit-index { vertical-align: super; }`	$a^2 + b^2$
`sub` — нижний индекс	`.digit-index { vertical-align: sub; }`	H_2O
Числовая величина смещения вверх в какой-либо единице измерения	`p::first-letter { vertical-align: 15pt; }`	Диетические
Отрицательная величина вызывает смещение вниз	`p::first-letter { vertical-align: -15pt; }`	Диетические

Значение по умолчанию: `baseline`.

14.4.3. Тень у текста

Один из способов придать тексту, выводящемуся в каком-либо элементе страницы, эффектный вид — создать у него тень. Единственное условие: текст должен быть набран достаточно крупным шрифтом, иначе он будет плохо читаться.

Для указания параметров тени применяется наследуемый атрибут стиля `text-shadow`. Его можно указывать у любых элементов страницы. Поддерживаются следующие значения:

- *<X> <Y> [<R>] [<цвет>]* — создает у текста тень с заданными параметрами. Здесь:
  - *X* — смещение тени относительно текста по горизонтали. Положительные значения вызывают смещение вправо, отрицательные — влево;
  - *Y* — смещение тени относительно текста по вертикали. Положительные значения вызывают смещение вниз, отрицательные — вверх;
  - *R* — радиус размытия тени. Если не указан, принимается равным `0`. Эти три параметра задаются в виде числовых значений в какой-либо единице измерения;
  - *цвет* тени. Если не указан, тень получит тот же цвет, которым выводится текст;
- `none` — тень отсутствует.

Значение по умолчанию: `none`.

Пример	Результат
`text-shadow: 4pt 4pt 3pt grey;`	Пример
`text-shadow: -4pt -4pt lightgrey;`	Пример
`text-shadow: 0pt 0pt 10pt;`	Пример

Как показывает практика, в значении атрибута стиля `text-shadow` обозначение цвета можно поставить перед величинами смещений:

```
text-shadow: grey 4pt 4pt 3pt;
```

14.4.4. Перенос слов

Мы можем дать веб-обозревателю предписание выполнять перенос слов согласно правилам указанного языка.

Чтобы перенос слов успешно выполнялся, необходимо указать язык текста — в атрибуте тега `lang` (см. *разд. 4.5*). Обычно язык текста всей (или, по крайней мере, большей части) страницы задается в теге `<html>`, например:

```
<html lang="ru">
    . . .
</html>
```

Для управления переносом слов служит наследуемый атрибут стиля `hyphens`, который можно указать у элемента любой разновидности. Он поддерживает значения, приведенные в табл. 14.3.

Тестовый HTML-код:

```
<div>Полу&shy;автоматический</div>
```

Таблица 14.3

Значение	Описание	Пример
`manual`	Перенос выполняется лишь в тех местах слов, в которых присутствуют специальные символы `&shy;` («перенос слова», см. *разд. 6.4.1*)	Полу- автоматический
`auto`	Перенос выполняется полностью автоматически согласно правилам языка	Полу- автомати- ческий
`none`	Перенос слов вообще не выполняется	Полуавтоматический

Значение по умолчанию: `manual`.

14.4.5. Обработка пробельных символов и разрывов строк

В *разд. 4.2.3* говорилось, что веб-обозреватель при выводе «сжимает» последовательность любых пробельных символов (включая разрывы строк) в единичный пробел и самостоятельно производит перенос строк. Это поведение по умолчанию, которое можно изменить.

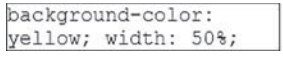
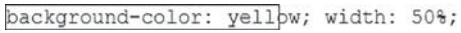
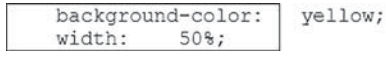
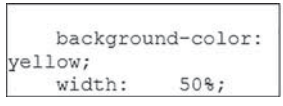
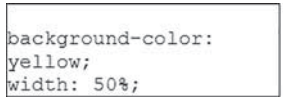
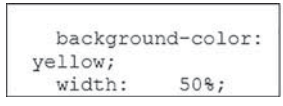
Для задания режима обработки пробельных символов и переноса строк служит атрибут стиля `white-space`. Его можно задавать у любых элементов.

Поддерживаемые им значения приведены в табл. 14.4.

Тестовый HTML-код:

```
<div>
    background-color:   yellow;
    width:    50%;
</div>
```

Таблица 14.4

Значение	Описание	Результат
`normal`	Все пробельные символы «сжимаются», перенос строк выполняется автоматически	
`nowrap`	То же самое, что и `normal`, только перенос строк не выполняется	
`pre`	Все пробелы и символы табуляции выводятся как есть, перенос строк выполняется лишь в местах их разрывов и по тегам ` `	
`pre-wrap`	То же самое, что и `pre`, только перенос строк при необходимости выполняется автоматически, а пробелы и символы табуляции в местах переноса «сжимаются»	
`pre-line`	То же самое, что и `pre-wrap`, только все пробелы «сжимаются»	
`break-spaces`	То же самое, что и `pre-wrap`, только пробелы и символы табуляции в местах переноса строк выводятся как есть	

Значение по умолчанию: `pre` (у текста фиксированного формата `<pre>`) или `normal` (у остальных тегов).

При указании у атрибута стиля `white-space` значения `pre-wrap`, `pre-line` или `break-spaces` в начале содержимого элемента может появиться ненужная пустая строка. Убрать ее можно, набрав открывающий тег и первую строку содержимого элемента в одну строку.

Тестовый CSS-код:

```
div { white-space: pre-wrap; }
```

Пример	Результат

`<div>`
`  background-color: yellow;`
`  width: 50%;`
`</div>`
	background-color: yellow;
	width: 50%;

`<div>  background-color: yellow;`
`  width: 50%;`
`</div>`
	background-color: yellow;
	width: 50%;

Может оказаться полезным атрибут стиля `tab-size`, задающий количество пробелов, в которые будет преобразован знак табуляции при выводе на экран (знак табуляции обозначается специальным символом `&#0009;`). Он указывается у блочных элементов. Значение по умолчанию: 8. Пример:

```
pre { tab-size: 4; }
```

14.5. Указание сразу нескольких параметров шрифта и строк текста

Наследуемый атрибут стиля `font` позволяет указать сразу несколько параметров шрифта: гарнитуру, кегль, насыщенность, плотность, начертание, форму строчных букв и высоту строки. Его значение записывается в формате:

```
[<начертание (font-style)>] [<форма строчных букв (font-variant)>]
[<насыщенность (font-weight)>] [<плотность (font-stretch)>]
[<кегль (font-size)>][/ <высота строки (line-height)>]
[<гарнитура (font-family)>];
```

Все параметры шрифта разделяются пробелами, за исключением параметров *кегль* и *высота строки*, которые отделяются друг от друга слешем. Должен быть указан, по крайней мере, один из параметров: *кегль* или *гарнитура*. Остальные параметры являются необязательными — если какой-то из них пропущен, у него устанавливается значение по умолчанию.

Примеры:

```
p { font: 14pt Arial; }
address { font: bold 12pt/14pt 'Times New Roman'; }
q { font: italic small-caps bold 10pt; }
```

Атрибут стиля `font` дополнительно позволяет указывать следующие *гарнитуры* системных шрифтов:

- `caption` — шрифт, которым выводятся надписи у элементов управления;
- `icon` — шрифт надписей у значков;
- `menu` — шрифт пунктов меню;
- `message-box` — шрифт надписей, которые выводятся в диалоговых окнах;
- `small-caption` — шрифт надписей у небольших элементов управления;
- `status-bar` — шрифт надписей, выводимых в строке состояния.

Пример:

```
p { font: 10pt menu; }
```

Следует отметить, что атрибут стиля `font-family` (см. *разд. 14.1.1*) эти значения не поддерживает.

Параметры *начертание*, *форма строчных букв* и *насыщенность* в атрибуте стиля `font` могут располагаться в произвольном порядке. Так, следующие три стиля дадут одинаковый результат:

```
q { font: italic small-caps bold 10pt; }
q { font: italic bold small-caps 10pt; }
q { font: small-caps bold italic 10pt; }
```

14.6. Генерируемое содержание

> ***Генерируемое содержание***
>
> Фрагменты содержания страниц, которые не записываются в HTML-коде, а формируются самим веб-обозревателем по заданным правилам.

Генерируемое содержание может быть выведено перед или после содержимого элемента, записанного в HTML-коде, и представлять собой произвольную строку, графическое изображение, значение указанного атрибута тега и пр.

Описание генерируемого содержания, которое требуется вывести у заданных элементов страницы, помещается в стиль, указывающий на эти элементы. Например, если генерируемое содержание следует вывести у блочных цитат, его следует описать в теге с селектором тега `<blockquote>`.

В селекторе того же стиля необходимо вставить указание на место, в котором должно быть отображено генерируемое содержание, в виде одного из следующих псевдоэлементов:

- `::before` — перед содержимым элемента, записанным в HTML-коде:
- `::after` — после содержимого элемента.

Для указания самого генерируемого содержания применяется ненаследуемый атрибут стиля `content`. Он поддерживает следующие варианты значений:

- `normal` — генерируемое содержание по умолчанию (зависит от типа элемента — у большинства тегов отсутствует, но, например, у тега `<q>` представляет собой кавычки);
- `none` — генерируемое содержание не выводится в любом случае;
- строка — выводится как есть:

```
li.new::before { content: 'Новинка! '; }
. . .
<ul>
    <li>чай,</li>
    <li>кофе,</li>
    <li class="new">мате. (Будет выведен с префиксом «Новинка!»)</li>
</ul>
```

- `open-quote` и `close-quote` — соответственно, открывающая и закрывающая кавычка:

```
cite::before { content: open-quote; }
cite::after { content: close-quote; }
. . .
<p>Название книги <cite>HTML и CSS</cite>
   будет выведено в кавычках.</p>
```

- ссылка на графический файл — изображение из этого файла:

```
a::after { content: url(../images/go-to.gif); }
```

- функция `attr(<`*`имя атрибута тега`*`>)` — значение атрибута тега с указанным *именем*;
- комбинация приведенных ранее значений, за исключением функции `url()`, разделенных пробелами:

```
cite::after { content: close-quote ' (Цитата)'; }
```

14.6.1. Счетчики

> **Счетчик**
>
> Переменная CSS, хранящая порядковый номер очередного присутствующего на странице элемента указанного типа. Используется для создания нумерации элементов.

Чтобы создать счетчик, следует написать стиль, применяемый к родителю всех элементов страницы, которые необходимо пронумеровать, или к первому из нумеруемых элементов. В этом стиле указывается ненаследуемый атрибут стиля `counter-reset`, который создаст счетчик с указанным *именем* и занесет в него заданное *изначальное значение*. Значение у этого атрибута стиля задается в формате:

<имя создаваемого счетчика> [*<изначальное значение>*]

Если *изначальное значение* не задано, в счетчик будет занесено значение `0`.

Пример:

```
h1:first-of-type { counter-reset: ch1 0; }
```

Можно создать произвольное количество счетчиков, записав в атрибуте стиля `counter-reset` нужное количество значений приведенного ранее формата, разделив их пробелами:

```
h1:first-of-type { counter-reset: ch1 0 ch2; }
```

Значение `none` атрибута стиля `counter-reset` отменяет создание счетчиков. Это, кстати, его значение по умолчанию.

Создав счетчик, следует указать веб-обозревателю, чтобы он с его помощью вел подсчет всех элементов требуемого типа, присутствующих на странице. Подсчет производится увеличением значения счетчика на заданную величину (обычно — на `1`).

Для этого следует написать стиль, применяемый к нумеруемым элементам, и указать в нем ненаследуемый атрибут стиля `counter-increment`, который будет изменять значение счетчика с заданным *именем* на указанную *величину*. Вот формат значения этого атрибута стиля:

<имя счетчика> [*<величина, на которую изменится значение счетчика>*]

Если *величина* не задана, счетчик будет увеличен на `1`. Можно указать отрицательную *величину* — тогда значение счетчика будет соответственно уменьшаться.

Значение `none` атрибута стиля `counter-increment` отменяет изменение значения какого бы то ни было счетчика и является его значением по умолчанию.

Пример:

```
h1 { counter-increment: ch1; }
```

Осталось лишь вывести текущее значение счетчика перед или после очередного из нумеруемых элементов страницы. Значение счетчика выводится в составе генерируемого содержания, посредством атрибута стиля `content` (см. *разд. 14.6*).

Функция CSS `counter(<имя счетчика> [<стиль нумерации>])` выводит в составе генерируемого содержания текущее значение счетчика с заданным *именем* в указанном *стиле*.

В качестве *стиля* можно записать одно из значений: `decimal` (арабские цифры), `decimal-leading-zero` (арабские цифры с начальным нулем), `lower-roman` (римские цифры в нижнем регистре), `upper-roman` (римские цифры в верхнем регистре), `lower-alpha` (буквы латиницы в нижнем регистре), `lower-latin` (то же самое, что и `lower-alpha`), `upper-alpha` (буквы латиницы в верхнем регистре) или `upper-latin` (то же самое, что и `upper-alpha`). Если *стиль* не указан, используется значение `decimal`.

Пример:

```
h1::before { content: counter(ch1); }
```

Рассмотрим пару более развернутых примеров.

Тестовый HTML-код:

```
<h1>Параметры шрифта</h1>
<h1>Загружаемые шрифты</h1>
<h1>Параметры строк текста</h1>
<h1>Генерируемое содержание</h1>
```

Пример	Результат

`h1:first-of-type {`
`  counter-reset: ch1;`
`}`
`h1 { counter-increment: ch1; }`
`h1::before {`
`  content: counter(ch1) '. ';`
`}`
	1. Параметры шрифта
	2. Загружаемые шрифты
	3. Параметры строк текста
	4. Генерируемое содержание

`h1:first-of-type {`
`  counter-reset: ch1 5;`
`}`
`h1 { counter-increment: ch1 -1; }`
`h1::before {`
`  content: counter(ch1) '. ';`
`}`
	4. Параметры шрифта
	3. Загружаемые шрифты
	2. Параметры строк текста
	1. Генерируемое содержание

Изменять значение счетчика и выводить его на экран можно и в одном стиле — выполняющем вывод значения счетчика:

```
h1::before {
    counter-increment: ch1;
    content: counter(ch1) '. ';
}
```

С помощью счетчиков можно создавать иерархическую нумерацию, при которой, например, главы и разделы, входящие в состав глав, нумеруются по отдельности. Для этого достаточно создать два отдельных счетчика — для глав и разделов, и при достижении очередной главы обнулять счетчик разделов. Счетчики удобнее создавать в элементе, являющемся родителем всех нумеруемых элементов.

Тестовый HTML-код (иерархия заголовков показана отступами слева):

```
<h1>Параметры шрифта</h1>
    <h2>Гарнитура</h2>
    <h2>Кегль</h2>
    <h2>Насыщенность</h2>
<h1>Загружаемые шрифты</h1>
<h1>Параметры строк текста</h1>
    <h2>Отступы и интервалы</h2>
    <h2>Выравнивание</h2>
<h1>Генерируемое содержание</h1>
```

Пример	Результат

`body {`
`  counter-reset: ch1 ch2;`
`}`
`h1 {`
`  counter-increment: ch1;`
`  counter-reset: ch2;`
`}`
`h2 {`
`  counter-increment: ch2;`
`}`
`h1::before {`
`  content: counter(ch1) '. ';`
`}`
`h2::before {`
`  content: counter(ch1) '.'`
`  counter(ch2, lower-latin)`
`  '. ';`
`}`
	1. Параметры шрифта
	1.a. Гарнитура
	1.b. Кегль
	1.c. Насыщенность
	2. Загружаемые шрифты
	3. Параметры строк текста
	3.a. Отступы и интервалы
	3.b. Выравнивание
	4. Генерируемое содержание

14.7. Упражнение. HTML-значки

Теория

> ***HTML-значок***
>
> Какой-либо пустой HTML-тег, посредством которого на страницу помещается оформленное заданным образом изображение, хранящееся в определенном файле.

Обычно HTML-значок создается пустым тегом `<i>`, `<b>` или `<span>` с определенным стилевым классом, который задает выводимое изображение и его оформление. Само изображение выводится в составе генерируемого содержания этого тега.

Применение HTML-значков вместо тегов `<img>` несколько упрощает верстку страниц.

Практика

Превратим небольшое изображение фужера, присутствующее на главной странице сайта суши-бара, в HTML-значок.

1. Найдем в папке 14\ex14.3 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Изображение фужера, на основе которого мы создадим значок, хранится в файле images\drink.jpg. Запомним это.

   Наш HTML-значок будет представлять собой пустой тег `<i>` со стилевым классом `icon-drink`.

   Как говорилось ранее, требуемое изображение выводится в HTML-значке в составе генерируемого содержания. Выведем его в начале содержимого значка. Кроме того, укажем высоту значка равной кеглю текущего шрифта. Высоту задает атрибут стиля `height`.

2. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и добавим стиль для HTML-значка:

```
i.icon-drink {
    content: url(../images/drink.jpg);
    height: 1em;
}
```

Для вывода изображения в качестве генерируемого содержания применяем функцию `url()` (см. *разд. 14.6*). Единица измерения `em` — это кегль текущего шрифта.

3. Откроем страницу index.html в текстовом редакторе и вместо тега `<img>`, размещающего изображение фужера, вставим тег `<i>` со стилевым классом `icon-drink`, создающий HTML-значок:

```
<p>Также &mdash; <em>безалкогольные напитки</em>
<img src="images/drink.jpg">:</p>
<i class="icon-drink"></i>:</p>
```

Визуально страница никак не изменится. Лишь изображение фужера станет чуть меньше, подстроившись под кегль шрифта.

14.8. Уровень непрозрачности

Уровень непрозрачности у элемента любого вида можно указать в ненаследуемом атрибуте стиля `opacity` в виде вещественного числа от `0.0` (полная прозрачность) до `1.0` (полная непрозрачность):

```
p.secret-notes { opacity: 0.1; }
```

Значение по умолчанию: `1.0`.

Следует помнить, что атрибут стиля `opacity` задает степень непрозрачности всех частей элемента: его текста, фона, рамки (если она указана) и элементов-потомков.

14.9. Самостоятельные упражнения

На страницах каталогов блюд сайта суши-бара укажите:

- у заголовков страниц — кегль 28 пунктов и преобразование букв к верхнему регистру;
- у названий блюд:
  - интервал между символами в 10 пунктов;
  - вывод символа двоеточия в конце (примените генерируемое содержание);
- у буквиц — вертикальное выравнивание по самому высокому символу в тексте.

✓ У вас должно получиться ⌄

СУШИ

ЧУККА:

Диетические суши, приготовляемые из одноименного салата, риса, кунжута и водорослей нори.

Урок 15. Блоки

Рамки вокруг элементов.
Графические рамки.
Просветы внутренние и внешние.
Указание размеров элементов.
Соотношение сторон.
Тени у блочных элементов.
Переполнение. Поведение при переполнении.

На этом уроке мы рассмотрим параметры оформления, которые можно задать лишь у блочных и встроенно-блочных элементов. Будучи заданными у любого встроенного элемента, они будут проигнорированы.

ВНИМАНИЕ!

Все атрибуты стиля, описываемые здесь, являются ненаследуемыми.

15.1. Рамки

Рамка рисуется по воображаемой границе элемента страницы. Можно вывести рамку как со всех сторон элемента, так и лишь с некоторых, — например, с левой или с нижней.

ВНИМАНИЕ!

Рамка, выведенная у элемента страницы, находится в составе этого элемента и, таким образом, увеличивает его совокупные размеры (подробности о размерах элементов — в *разд. 15.4*).

Чтобы создать у элемента рамку, необходимо указать, по крайней мере, стиль ее линий. Также можно задать ее толщину и цвет.

Стиль линии у одной из сторон рамки задается атрибутами стилей: `border-top-style` (верхняя сторона рамки), `border-right-style` (правая),

border-bottom-style (нижняя) и border-left-style (левая). Они поддерживают весьма много значений:

- none — полное отсутствие линии;
- hidden — невидимая линия. Присутствует в составе элемента, увеличивая его совокупные размеры, но не выводится на экране;
- solid — сплошная линия;
- dashed — пунктирная линия;
- dotted — штриховая линия;
- double — двойная сплошная линия;
- ridge — трехмерный «гребень»;
- groove — трехмерный «желоб»;
- inset — «углубление»;
- outset — «возвышенность».

Значение по умолчанию: none.

Толщина линии у одной из сторон рамки указывается с применением атрибутов стилей: border-top-width (верхняя сторона рамки), border-right-width (правая), border-bottom-width (нижняя) и border-left-width (левая). Вот поддерживаемые ими значения:

- числовая величина толщины в какой-либо поддерживаемой единице измерения (см. *разд. 12.2.1.2.1*);
- одно из предопределенных значений: thin (малая толщина), medium (средняя) или thick (большая). Конкретные величины толщины линий в этих случаях у разных веб-обозревателей различаются.

Значение по умолчанию: medium.

Для указания цвета рамки применяются атрибуты стилей: border-top-color (верхняя сторона рамки), border-right-color (правая), border-bottom-color (нижняя) и border-left-color (левая). В качестве значения можно указать цвет в любом из поддерживаемых форматов (см. *разд. 12.2.2*). Значение по умолчанию: currentcolor (цвет текста).

Все эти атрибуты стиля позволяют задать разные параметры у разных сторон рамки.

Пример	Результат

`border-left-width: medium;`
`border-left-style: double;`
`border-left-color: black;`
`border-right-width: medium;`
`border-right-style: double;`
`border-right-color: black;`
	‖ Слева и справа ‖

Код, написанный с применением описанных здесь атрибутов стиля, получается очень громоздким. Поэтому CSS предлагает четыре атрибута стиля, задающие сразу все параметры одной из сторон рамки: `border-top` (верхняя сторона рамки), `border-right` (правая), `border-bottom` (нижняя) и `border-left` (левая). Их значения записываются в формате:

`[`*<толщина>*`] <стиль> [`*<цвет>*`]`

Отдельные величины разделяются пробелами. Обязательной к указанию величиной является лишь *стиль* линии.

Пример	Результат
`border-left: medium double black;` `border-right: medium double black;`	‖ Слева и справа ‖

Еще три атрибута стиля задают параметры сразу у всех сторон рамки: `border-width` (толщина), `border-style` (стиль) и `border-color` (цвет). В их значениях можно указать четыре, три, две или одну величину, разделив их пробелами:

- *<сверху> <справа> <снизу> <слева>*:
  `border-width: thin 2pt 4pt 2pt;`
- *<сверху> <справа и слева> <снизу>*:
  `border-width: thin 2pt 4pt;`
- *<сверху и снизу> <справа и слева>*:
  `border-style: double dotted;`
- *<сверху, справа, снизу и слева>*:
  `border-style: double;`

Пример	Результат

`border-width: 2px 10px;`
`border-style: none solid dotted solid;`
`border-color: green grey black;`
	Мудреная рамка

Если же требуется создать рамку, все стороны которой выглядят одинаково, на помощь придет атрибут стиля `border`. Его значение записывается в формате:

`[<толщина>] <стиль> [<цвет>]`

Отдельные величины разделяются пробелами. Единственной обязательной к указанию величиной является, опять же, *стиль* линии.

Пример	Результат
`border: 2px dashed grey;`	Рамка со всех сторон

Какие атрибуты стиля выбрать для создания рамки — дело вкуса и привычек веб-верстальщика.

Как показывает практика, в значениях атрибутов стиля «семейства» `border`, задающих сразу все параметры рамки, отдельные величины можно указать в произвольном порядке. Так, следующие три атрибута стиля создадут одинаковую рамку:

```
border: 2px dashed grey;
border: dashed 2px grey;
border: grey dashed 2px;
```

15.1.1. Графические рамки

В графической рамке линии рисуются указанным графическим изображением.

ВНИМАНИЕ!

Графическая рамка, в отличие от обычной, рисуется поверх содержимого элемента страницы. Поэтому у элемента следует задать достаточные размеры, чтобы графическая рамка не налезла на содержимое элемента.

При этом надо учесть, что графическая рамка не увеличивает совокупные размеры элемента страницы, у которого выведена.

Для задания ссылки на файл с изображением, которым будут рисоваться линии графической рамки, служит атрибут стиля `border-image-source`. В качестве значения можно указать:

- ссылку на графический файл — с помощью функции `url()`;
- `none` — будет нарисована обычная рамка.

Значение по умолчанию: `none`.

Пример:

```
border-image-source: url(../images/border.png);
```

Изображение, указанное в этом атрибуте стиля, делится веб-обозревателем на 9 частей, и каждая получившаяся часть используется для рисования одного из углов или одной из сторон рамки:

1 — левого верхнего угла рамки;

2 — верхней стороны;

3 — правого верхнего угла;

4 — левой стороны;

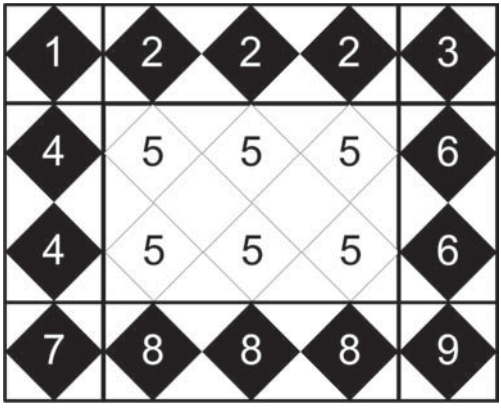
5 — заливки (если указана);

6 — правой стороны;

7 — нижнего левого угла;

8 — нижней стороны;

9 — правого нижнего угла.



Атрибут стиля `border-image-slice` служит для задания размера каждой из этих частей: высоты частей 2 и 8, ширины частей 4 и 6, ширины и высоты «углов» 1, 3, 7 и 9. В значении этого атрибута стиля можно указать четыре, три, две или одну величину, разделив их пробелами:

- *<выс. ч. 1-3> <шир. ч. 3, 6 и 9> <выс. ч. 7-9> <шир ч. 1, 4 и 7>*;
- *<выс. ч. 1-3> <шир. ч. 1, 3, 4, 6, 7 и 9> <выс. ч. 7-9>*;
- *<высота ч. 1-3, 7-9> <ширина ч. 1, 3, 4, 6, 7 и 9>*;
- *<высота и ширина всех частей>*.

Отдельная величина размера измеряется в пикселах и задается в виде целого неотрицательного числа.

В любом месте значения атрибута стиля `border-image-slice` можно указать слово `fill`, которое предпишет веб-обозревателю заполнить остальное пространство элемента частью 5 указанного изображения. Подобного рода заполнение будет выведено поверх фона элемента (если таковой был указан — см. *урок 19*).

Атрибут стиля `border-image-width` задает толщину графической рамки. В его значении можно указать четыре, три, две или одну величину, разделив их пробелами:

- *<сверху> <справа> <снизу> <слева>*;
- *<сверху> <справа и слева> <снизу>*;

- *<сверху и снизу> <справа и слева>*;
- *<сверху, справа, снизу и слева>*.

Доступны для указания величины в следующих форматах:

- числовая величина в любой поддерживаемой единице измерения;
- `auto` — изначальный размер изображения.

ВНИМАНИЕ!

У атрибутов стиля `border-image-slice` и `border-image-width` обязательно следует указать значения. В противном случае графическая рамка не будет нарисована.

Пример	Результат
`border-image-width: auto;` `border-image-slice: 30;`	
`border-image-width: auto;` `border-image-slice: 15;`	
`border-image-width: auto;` `border-image-slice: 10 5 25 5;`	
`border-image-width: auto;` `border-image-slice: 30 fill;`	
`border-image-width: 10px;` `border-image-slice: 30;`	

Атрибут стиля `border-image-repeat` указывает, как будут формироваться стороны рамки. В его значении можно указать две или одну величину, разделив их пробелами:

- *<сверху и снизу> <справа и слева>*;
- *<сверху, справа, снизу и слева>*.

В качестве отдельной величины можно задать одно из приведенных в табл. 15.1 предопределенных значений.

Таблица 15.1

Значение	Описание	Пример
`stretch`	Изображение растягивается на всю ширину или высоту элемента	
`repeat`	Изображение повторяется. Отдельные копии изображения выводятся вплотную друг к другу и без изменения размеров. Вследствие этого в составе какой-либо стороны рамки может быть выведена неполная копия изображения	
`round`	Изображение повторяется. Размеры его копий подгоняются таким образом, чтобы избежать вывода неполных копий изображения	
`space`	Изображение повторяется. Между отдельными копиями выводятся просветы с целью избежать вывода неполных копий изображения	

Значение по умолчанию: `stretch`.

Пример	Результат

`border-image-width: 10px;`
`border-image-slice: 30;`
`border-image-repeat: stretch round;`
	

Атрибут стиля `border-image-outset` задает величину просвета между рисуемой графической рамкой и границей элемента страницы (по которой рисуется обычная рамка). Его значение может содержать четыре, три, две или одну величины, разделенные пробелами:

- *<сверху> <справа> <снизу> <слева>*;
- *<сверху> <справа и слева> <снизу>*;
- *<сверху и снизу> <справа и слева>*;
- *<сверху, справа, снизу и слева>*.

Отдельная величина задается в виде неотрицательного числового значения в любой из поддерживаемых единиц измерения.

Значение по умолчанию: `0`.

Пример	Результат
`border-image-width: auto;` `border-image-slice: 30;`	

`border-image-width: auto;`
`border-image-slice: 30;`
`border-image-outset: 15px;`
	

`border-image-width: auto;`
`border-image-slice: 30;`
`border-image-outset: 30px;`
	

Атрибут стиля `border-image` позволяет указать сразу все параметры графической рамки. Его значение задается в формате:

```
<ссылка на файл с изображением (border-image-source)>
/ <размер части (border-image-slice)>
/ <толщина рамки (border-image-width)>
[/ <смещение рамки (border-image-outset)>]
[<режим формирования сторон рамки (border-image-repeat)>]
```

Пример:

```
border-image: url(../images/border.png) / 30 / 10 space;
```

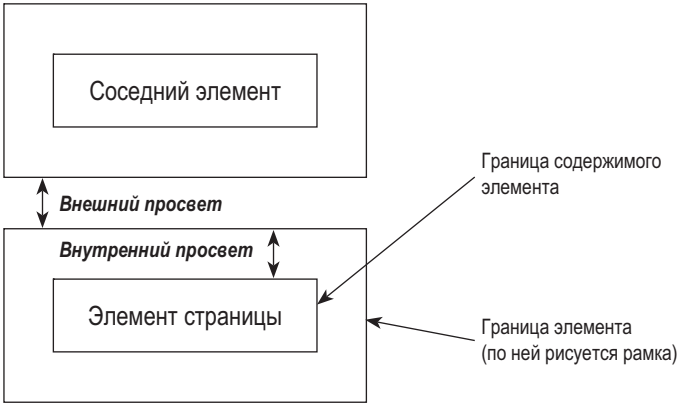
15.2. Просветы: внешние и внутренние

Внешний просвет

Просвет между воображаемой границей элемента страницы, внутри которой выполняется заливка фоном и по которой рисуется рамка, и границами соседних элементов и родителя.

Внутренний просвет

Просвет между границами содержимого элемента страницы и воображаемой границей самого элемента.



Для указания внешних просветов применяются атрибуты стилей: `margin-top` (внешний просвет сверху), `margin-right` (справа), `margin-bottom` (снизу) и `margin-left` (слева). В качестве значения у них можно указать:

- числовую величину просвета в какой-либо единице измерения;
- `auto` — величина просвета станет равной величине неиспользуемого свободного пространства родителя, соответственно, выше, правее, ниже или левее элемента.

Значение по умолчанию: `0`. Однако в реальности некоторые элементы, в частности, заголовки, выводятся с довольно значительными просветами сверху и снизу.

Внутренние просветы задаются атрибутами стилей: `padding-top` (внутренний просвет сверху), `padding-right` (справа), `padding-bottom` (снизу) и `padding-left` (слева). Значение указывается в виде числовой величины в какой-либо из поддерживаемых единиц измерения. Значение по умолчанию: `0`.

Для задания величин внешних и внутренних просветов сразу со всех сторон элемента применяются атрибуты стиля `margin` и `padding` соответственно.

В их значениях можно записать четыре, три, две или одну величины просветов, разделив их пробелами:

- *<сверху> <справа> <снизу> <слева>*;
- *<сверху> <справа и слева> <снизу>*;
- *<сверху и снизу> <справа и слева>*;
- *<сверху, справа, снизу и слева>*.

Пример	Результат
`margin: 0;` `padding: 0;`	
`margin: 4pt 12pt 4pt 2pt;` `padding: 5pt 2pt 5pt 8pt;`	
`margin: 4pt;` `padding: 2pt 8pt;`	

Как убрать просветы между границами области просмотра и содержимым веб-страницы?

Для этого достаточно указать у секции тела страницы (тега `<body>`) нулевые внешние просветы:

```
body { margin: 0; }
```

15.3. Упражнение. Стильный заголовок

Сделаем у сайта суши-бара стильный привлекательный заголовок. Он станет двустрочным: название суши-бара займет первую строку, будет выведено крупным шрифтом Caviar Dreams на красивом фоне и подчеркнуто, а вторую строку мы отведем под слово «Суши-бар», выведенное меньшим кеглем. Содержимое заголовка выровняем по правому краю.

Изображение тарелки с суши, присутствовавшее в заголовке ранее, пока уберем. Однако сам файл с картинкой оставим — позже, на *уроке 19*, мы вновь выведем его в заголовке вместе с еще одной симпатичной картинкой.

1. Найдем в папке 14\s14.9 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Сначала переделаем заголовок на главной странице сайта. А потом, закончив работу, перенесем его на остальные страницы.

   Содержимое заголовка у нас находится в теге семантической шапки `<header>`. Перед началом работы его следует удалить наряду со стилями, задававшими оформление старого заголовка.

2. Откроем главную страницу index.html в текстовом редакторе и удалим старое содержимое тега `<header>`:

```
<header>
    <h1>Суши-бар<br><span class="bar-title">Йокогама</span></h1>
    <p class="picture"><img src="images/sushi.jpg"></p>
</header>
```

3. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и удалим стили с селекторами: `h1`, `.picture img`, `header` и `.bar-title`.

   Сразу же укажем у семантической шапки `<header>` выравнивание по правому краю.

4. Добавим в таблицу стилей стиль для семантической шапки:

```
header { text-align: right; }
```

   Для создания двустрочного заголовка применим два заголовка первого уровня. Чтобы впоследствии применить к этим заголовкам стили, укажем у них стилевые классы `header1` и `header2`.

5. Переключимся на главную страницу и вставим в семантическую шапку HTML-код, создающий двустрочный заголовок:

```
<header>
    <h1 class="header1">Йокогама</h1>
    <h1 class="header2">Суши-бар</h1>
</header>
```

   У первой строки заголовка, где выводится название суши-бара, укажем:

   - шрифт — Caviar Dreams, кегль 48 пунктов, полужирный;
   - преобразование букв — к верхнему регистру;
   - расстояние между символами — 18 пунктов.

   Так мы сделаем заголовок более крупным и заметным;

- цвет фона — blanchedalmond;
- внешние просветы — нулевые.

  По умолчанию заголовки выводятся с довольно значительными просветами сверху и снизу. Если их не убрать, над нашим заголовком появится много пустого пространства, что выглядит некрасиво;
- внутренние просветы — 12 пунктов сверху, 8 — справа, нулевые — снизу и слева;
- рамка — только снизу, толщиной 12 пунктов, сплошная, цвета salmon.

6. Добавим в таблицу стилей стиль, который задаст описанное оформление:

```
header h1.header1 {
    font: bold 48pt 'Caviar Dreams';
    text-transform: uppercase;
    letter-spacing: 18pt;
    background-color: blanchedalmond;
    margin: 0;
    padding: 12pt 8pt 0 0;
    border-bottom: 12pt solid salmon;
}
```

✓ Что получилось? ⌄

Границы области просмотра обозначены рамкой.

Займемся нижней строкой заголовка. Зададим у нее:

- шрифт — Verdana, кегль 36 пунктов, обычная насыщенность;
- расстояние между символами — 8 пунктов;
- внешние просветы — нулевые;
- внутренние просветы — 36 пунктов справа, нулевые — с остальных сторон.

В результате вторая строка с текстом «Суши-бар» будет заметна, но не станет превалировать над первой строкой.

7. Добавим необходимый стиль:

```
header h1.header2 {
    font: 36pt Verdana;
    letter-spacing: 8pt;
    margin: 0;
    padding: 0 36pt 0 0;
}
```

✓ Что получилось? ⌄

Однако все равно остаются небольшие, но некрасивые просветы между границами области просмотра и новым заголовком. Но мы уже знаем, как их убрать (см. *разд. 15.2*).

8. Исправим стиль, задающий оформление для секции тела страницы:

```
~~body { font-family: Arial; }~~
body {
    font-family: Arial;
    margin: 0;
}
```

✓ Результат ⌄

9. Переделайте заголовки на остальных страницах сайта самостоятельно.

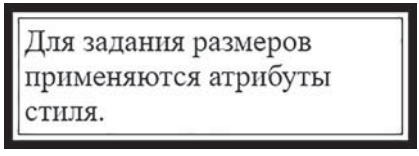
15.4. Указание размеров элементов

15.4.1. Размеры элементов

Для задания геометрических размеров применяются два следующих атрибута стиля:

- `width` — ширина элемента в виде:
  - числового значения в любой из поддерживаемых CSS единиц измерения (см. *разд. 12.2.1.2.1*);
  - одного из предопределенных значений, приведенных в табл. 15.2.

Таблица 15.2

Значение	Описание	Пример
`fit-content`	Ширина элемента равна ширине доступного свободного пространства в родителе	
`max-content`	Ширина элемента равна ширине его содержимого, выведенного в одну строку. Элемент может выйти за пределы родителя	
`min-content`	Минимально необходимая ширина для вывода содержимого	

  - значения `auto` — если содержимое элемента имеет строго установленную ширину (например, элемент является изображением), аналогично `max-content`, в противном случае — `fit-content`.

  Значение по умолчанию: `auto`;

- `height` — высота элемента. Можно указать числовую величину в любой из поддерживаемых CSS единиц измерения, предопределенное значение `fit-content`, `max-content` или `min-content`.

  Значение `auto`, если содержимое элемента имеет строго установленную высоту, аналогично `max-content`, в противном случае — `min-content`.

15.4.2. Предельные размеры элементов

Если размеры элемента указаны в процентах, в относительных единицах, основанных на размерах области просмотра или в виде значений `fit-content` или `auto`, то, возможно, элемент будет менять размеры при изменении размеров окна веб-обозревателя. В таком случае может оказаться полезным задать предельные размеры элемента.

Для этого применяются следующие атрибуты стиля:

- `min-width` — минимальная ширина элемента;
- `max-width` — максимальная ширина элемента;
- `min-height` — минимальная высота элемента;
- `max-height` — максимальная высота элемента.

Они принимают те же значения, что и атрибуты стиля `width` и `height`.

После указания предельных размеров элемент не уменьшится ниже заданного минимума и не увеличится сверх максимума. Однако, если у элемента задан слишком большой минимальный размер, а размер окна веб-обозревателя слишком мал, элемент перестанет помещаться в окне, и в нем возникнут полосы прокрутки.

15.4.3. Режим установки размеров

Атрибут стиля `box-sizing` задает режим установки размеров. Вот поддерживаемые им значения:

- `content-box` — атрибуты стиля, рассмотренные в *разд. 15.4.1* и *15.4.2*, задают размеры *содержимого* элемента, без учета указанных у него внутренних просветов и рамки. В результате фактический размер элемента окажется больше указанного нами. Пример:

```
div.sized {
    box-sizing: content-box;
    width: 200px;
    padding-left: 20px;
    padding-right: 30px;
    border: 5px solid blue;
}
```

  Результирующая ширина блока со стилевым классом `sized` будет равна сумме:

  - ширины содержимого, заданного атрибутом стиля `width` (200 пикселов);

- величин внутренних просветов слева и справа (20 и 30 пикселов);
- величин толщины левой и правой сторон рамки (по 5 пикселов).

Итого: 200 + 20 + 30 + 2 x 5 = 260 пикселов;

- ♦ `border-box` — атрибуты стиля, рассмотренные в *разд. 15.4.1* и *15.4.2*, задают полные размеры элемента.

Значение по умолчанию: `content-box`.

При создании сложной верстки обычно у всех элементов включают режим установки полных размеров, чтобы упростить подгонку элементов друг к другу. Для этого в таблицу стилей добавляют следующий стиль, действующий на все элементы (и содержащий универсальный селектор `*`):

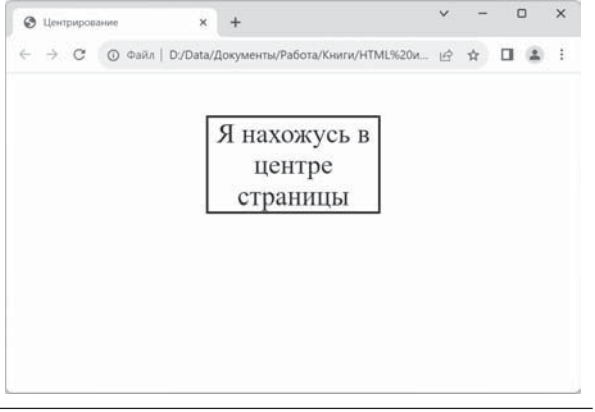
```
* { box-sizing: border-box; }
```

Как выровнять элемент страницы строго по центру родителя?

Для этого достаточно указать у выравниваемого по центру элемента:

- одно из двух:
  - ширину — атрибутом стиля `width`;
  - минимальную и максимальную ширину — атрибутами стиля `min-width` и `max-width`;
- внешние просветы слева и справа, равные `auto`. В результате каждый из внешних просветов станет равным половине ширины свободного пространства в родителе, не занятого элементом.

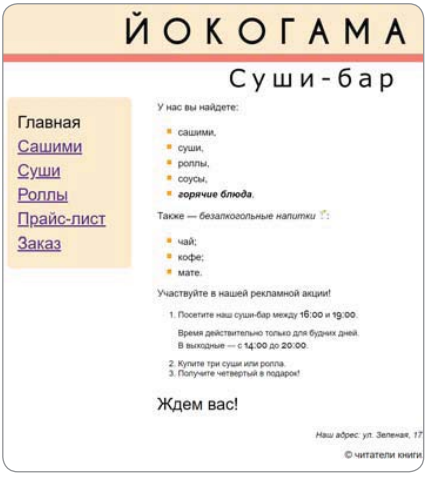
Пример	Результат

`div.centered {`
`  width: 1000px;`
`  margin-left: auto;`
`  margin-right: auto;`
`}`
	

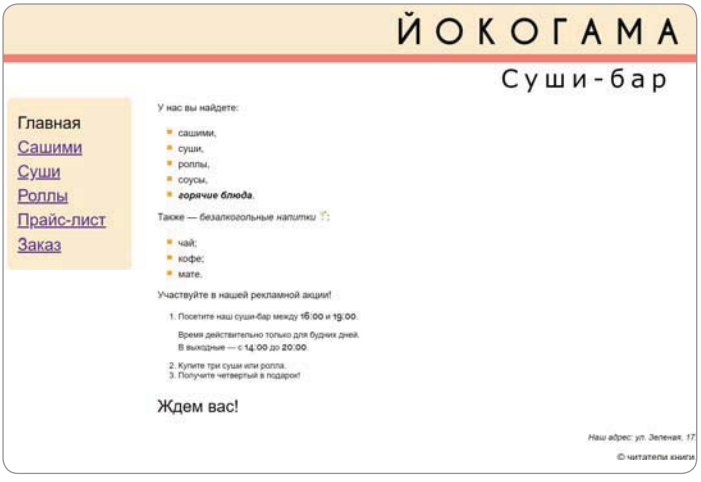
15.5. Упражнение. Ограничение ширины содержимого веб-страниц

Страницы сайта суши-бара имеют серьезный недостаток. Если значительно увеличить ширину окна веб-обозревателя, их содержимое чересчур растянется, а если сделать окно слишком узким — чересчур съежится. В обоих случаях его станет неудобно читать.

В качестве выхода из ситуации можно указать у всего содержания страницы минимальную и максимальную ширину. Так и сделаем.



Стандартная ширина окна



Увеличенная ширина окна

Уменьшенная ширина окна

1. Найдем в папке 15\ex15.3 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Содержание страниц нашего сайта находится в семантических шапке, панели навигации, основном содержимом и поддоне (тегах `<header>`, `<nav>`, `<main>` и `<footer>`). Чтобы указать у них минимальную и макси-

мальную ширину, мы заключим их в блок. А чтобы позже применить к этому блоку стиль, запишем в нем стилевой класс `container`.

2. Откроем главную страницу index.html и заключим теги `<header>`, `<nav>`, `<main>` и `<footer>` в блок со стилевым классом `container`:

```
<div class="container">
    <header>
        . . .
    </header>
    <nav>
        . . .
    </nav>
    <main>
        . . .
    </main>
    <footer>
        . . .
    </footer>
</div>
```

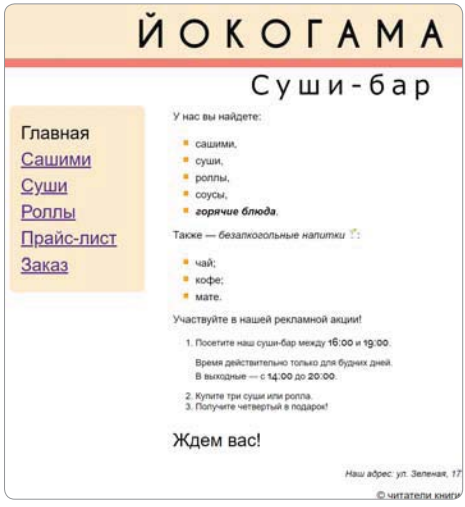
У этого блока укажем минимальную ширину в 768 пикселов, а максимальную — в 1024 пиксела (значения подобраны экспериментально). Заодно выровняем этот блок по центру родителя (см. *разд. 15.4.3*).

3. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и добавим необходимый стиль:

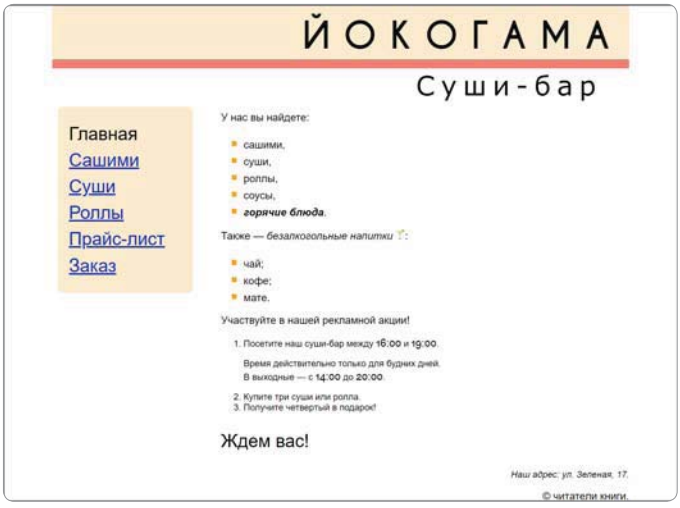
```
div.container {
    min-width: 768px;
    max-width: 1024px;
    margin-left: auto;
    margin-right: auto;
}
```

✓ Результат ›

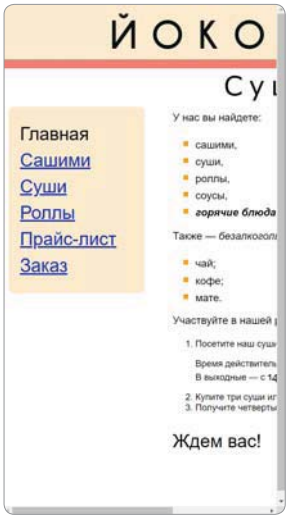
При растягивании окна веб-обозревателя по достижении указанной нами максимальной ширины содержание страницы перестанет растягиваться (границы области просмотра обозначены рамкой). А при сужении окна по достиже-



Стандартная ширина окна



Максимальная ширина окна



Минимальная ширина окна

нии заданной минимальной ширины содержание перестанет сужаться (и в окне появляется горизонтальная полоса прокрутки).

4. Внесем аналогичные изменения в код остальных страниц самостоятельно.

15.6. Соотношение сторон

Если у элемента страницы один из размеров указан, а другой — нет, можно сделать так, чтобы второй размер устанавливался соответственно заданному соотношению сторон.

Для указания соотношения сторон у элемента служит атрибут стиля `aspect-ratio`. В качестве его значения можно задать:

- отношение ширины к высоте элемента в формате:

  <ширина> `[/` *<высота>*`]`

  Если *высота* пропущена, она принимается равной 1;

- `auto` — если содержимое элемента имеет строго установленное соотношение сторон (например, элемент является изображением), используется оно, в противном случае элемент не будет иметь строго установленного соотношения сторон.

Значение по умолчанию: `auto`.

Пример	Результат

`width: 100px;`
`height: auto;`
`aspect-ratio: 1 / 1;`
	

`width: 100px;`
`height: auto;`
`aspect-ratio: 5 / 2;`
	

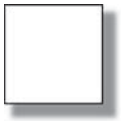
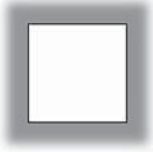
`width: 100px;`
`height: auto;`
`aspect-ratio: 0.5;`
	

15.7. Тень у блока

Для создания тени у блочного элемента применяется атрибут стиля `box-shadow`. Им поддерживаются следующие значения:

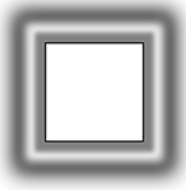
- *<X> <Y>* `[`*<R>* `[`*<S>*`]] [`*<цвет>*`] [inset]` — создает у элемента тень с заданными параметрами. Здесь:
  - *X* — числовая величина смещения тени по горизонтали относительно самого элемента. Положительные значения задают смещение вправо, отрицательные — влево;
  - *Y* — числовая величина смещения тени по вертикали относительно самого элемента. Положительные значения задают смещение вниз, отрицательные — вверх;
  - *R* — числовая величина радиуса размытия тени. Если не задан или равен `0`, тень не будет размыта;
  - *S* — числовая величина радиуса распространения тени — увеличения ее размеров относительно размеров элемента. Если не указан или равен `0`, тень будет иметь такие же размеры, как и элемент;
  - *цвет* тени. Если не указан, тень выводится тем же цветом, что задан у текста элемента;
  - `inset` — если этот параметр указан, тень выводится «внутри» элемента, если не указан — «снаружи»;
- `none` — тень отсутствует.

Значение по умолчанию: `none`.

Пример	Результат
`box-shadow: 4pt 4pt 2pt 1pt grey;`	
`box-shadow: -4pt -4pt 2pt 1pt grey;`	
`box-shadow: 0 0 4pt 8pt grey;`	
`box-shadow: 4pt 4pt 2pt 1pt grey inset;`	

Можно создать у элемента сразу несколько теней, задав их параметры в атрибуте стиля `box-shadow` через запятую. При этом тени станут перекрываться в порядке, обратном порядку, в котором они заданы: сначала будет выведена последняя тень, поверх нее — предпоследняя и т. д., а первая тень окажется выведена самой верхней.

Пример	Результат

`box-shadow: 0 0 2pt 4pt grey,`
`0 0 4pt 8pt white,`
`0 0 8pt 12pt black;`
	

На практике значение `inset` может присутствовать в любом месте значения атрибута стиля `box-shadow`, например:

```
box-shadow: inset 4pt 4pt 2pt 1pt grey;
```

15.8. Поведение при переполнении

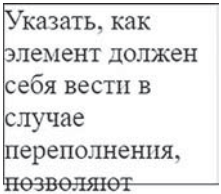
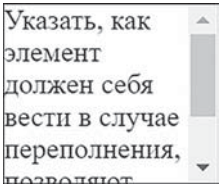
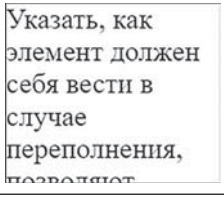
> ***Переполнение***
>
> Выход содержимого элемента страницы за его границы. Возникает, если элемент имеет жестко заданные размеры.

Указать, как элемент должен себя вести в случае переполнения, позволяют следующие два атрибута стиля:

- `overflow-x` — поведение при переполнении по горизонтали;
- `overflow-y` — поведение при переполнении по вертикали.

Эти атрибуты стиля поддерживают одинаковый набор значений (табл. 15.3).

Таблица 15.3

Значение	Описание	Пример
`visible`	Излишнее содержимое все равно выводится на экран, выходя за границы элемента	
`auto`	В случае переполнения в элементе появляются полосы прокрутки	
`scroll`	Полосы прокрутки в элементе присутствуют всегда, даже если переполнения нет	
`hidden`	Излишнее содержимое обрезается	

Значение по умолчанию: `visible`.

Также поддерживается атрибут стиля `overflow`, в котором можно указать поведение при переполнении и по горизонтали, и по вертикали. Его значение может представлять собой две или одну величину, разделенные пробелами:

- *<по горизонтали> <по вертикали>*;
- *<по горизонтали и вертикали>*.

Пример:

```
overflow: hidden auto;
```

Описанные атрибуты стиля применяются при создании *прокручиваемых элементов*.

> ***Прокручиваемый элемент веб-страницы***
>
> Элемент страницы, содержимое которого при переполнении может быть прокручено посредством полос прокрутки.

Если содержимое, выходящее за границы элемента, обрезается (атрибуту стиля «семейства» `overflow` дано значение `hidden`) и при этом отключен перенос строк (атрибут стиля `white-space` имеет значение `nowrap`), можно указать, чем должна завершаться выводящаяся часть содержимого. Для этого следует использовать атрибут стиля `text-overflow`, который поддерживает значения, приведенные в табл. 15.4.

Таблица 15.4

Значение	Описание	Пример
`clip`	Выводящаяся часть содержимого ничем не завершается	Указать, как элемент долж
`ellipsis`	Выводящаяся часть содержимого завершается символом многоточия	Указать, как элемент до…

Значение по умолчанию: `clip`.

15.9. Контур

> ***Контур***
>
> Аналогичен рамке (см. *разд. 15.1*), только рисуется вокруг границ элемента и не находится в его составе.

Для указания параметров контура применяются следующие атрибуты стиля:

- `outline-style` — стиль линий контура. Поддерживает те же значения, что и атрибуты стиля «семейства» `border-...-style`, кроме `hidden` (см. *разд. 15.1*);
- `outline-width` — толщина линий контура. Поддерживает те же значения, что и атрибуты стиля «семейства» `border-...-width`;
- `outline-color` — цвет линий контура. Поддерживает те же значения, что и атрибуты стиля «семейства» `border-...-color`;

- `outline-offset` — просвет между линиями контура и границами содержимого элемента. Задается в виде числового значения в любой поддерживаемой единице измерения. По умолчанию: `0`.

Пример	Результат

`border: thin solid grey;`
`outline-style: double;`
`outline-width: thick;`
`outline-color: black;`
`outline-offset: 4pt;`
	Для указания параметров контура применяются атрибуты стиля.

Атрибут стиля `outline` позволяет указать сразу стиль, толщину и цвет линий контура. Его значение записывается в формате:

`[<`*цвет (*`outline-color`*)*`>] <`*стиль (*`outline-style`*)*`>`
`[<`*толщина (*`outline-width`*)*`>]`

Пример:

```
outline: black double thick;
outline-offset: 4pt;
```

Как показывает практика, величины в значении атрибута стиля `outline` могут быть указаны в любом порядке.

15.10. Самостоятельное упражнение

Сделайте на страницах сайта суши-бара поддон, подобный представленному далее (границы области просмотра показаны рамкой).

ПОДСКАЗКИ

У верхней строки поддона (с адресом) используйте полужирный курсивный шрифт Caviar Dream кеглем 20 пунктов, цвет `blanchedalmond` для фона и `salmon` — для линии внизу. У нижней строки (с правами авторов) укажите достаточно большой *внутренний* просвет внизу, чтобы отделить абзац от низа области просмотра (задание внешнего просвета не даст такого результата).

Урок 16. Списки и таблицы

Параметры списков.
Параметры таблиц.
Параметры строк таблиц.
Параметры ячеек таблиц.
Параметры заголовка таблицы.
Параметры групп колонок и отдельных колонок.

16.1. Параметры списков

16.1.1. Параметры маркеров и нумерации

CSS позволяет задать стиль маркеров или нумерации пунктов списка (в зависимости от того, маркированный это список или нумерованный), использовать в качестве маркера произвольное изображение и указать местоположение маркера (нумерации).

Наследуемый атрибут стиля `list-style-type` указывает стиль маркеров (нумерации). Он поддерживает довольно много значений:

- `disc` — черный кружок;
- `circle` — светлый кружок;
- `square` — черный квадратик;
- `decimal` — арабские цифры;
- `decimal-leading-zero` — арабские цифры с начальным нулем;

- `upper-roman` — римские цифры, набранные прописными буквами;
- `lower-roman` — римские цифры, набранные строчными буквами;
- `upper-alpha` и `upper-latin` — прописные латинские буквы;
- `lower-alpha` и `lower-latin` — строчные латинские буквы;
- произвольная строка — будет использована в качестве маркера.
- `none` — маркер или нумерация не выводится.

Значение по умолчанию: `disk` (у маркированных списков) или `decimal` (у нумерованных списков).

Пример	Результат

`ol {`
`  list-style-type: upper-alpha;`
`}`
	A. Суши;
	B. сашими;
	C. роллы.

`ul {`
`  list-style-type: circle;`
`}`
	○ Суши;
	○ сашими;
	○ роллы.

`ul {`
`  list-style-type: '\2713';`
`}`
	✓Суши;
	✓сашими;
	✓роллы.

Наследуемый атрибут стиля `list-style-image` задает ссылку на изображение, которое будет использовано в качестве маркера. Доступные значения:

- собственно ссылка на файл с изображением — задается функцией `url()`. Атрибут стиля `list-style-type` в этом случае игнорируется. Пример:

  ```
  ol { list-style-image: url(../images/dot.gif); }
  ```

- `none` — используется стиль маркера или нумерации, заданный атрибутом стиля `list-style-type`.

Значение по умолчанию: `none`.

Если файл с изображением загрузить не удастся, будет использован стиль маркера (нумерации), указанный атрибутом стиля `list-style-type`.

Наследуемый атрибут стиля `list-style-position` указывает местоположение маркера (нумерации) относительно текста пункта. Вот его значения:

- `outside` — маркер (нумерация) выводится вне текста пункта.

Пример	Результат

`ol {`
`  list-style-position: outside;`
`}`
	I. Посетите наш суши-бар между 16:00 и 19:00.
	II. Купите три суши или ролла.
	III. Получите четвертый в подарок!

- `inside` — маркер (нумерация) выводится непосредственно в тексте пункта.

Пример	Результат

`ol {`
`  list-style-position: inside;`
`}`
	I. Посетите наш суши-бар между 16:00 и 19:00.
	II. Купите три суши или ролла.
	III. Получите четвертый в подарок!

Значение по умолчанию: `outside`.

Наследуемый атрибут стиля `list-style` задает сразу все параметры маркера (нумерации), которые указываются через пробел в следующей последовательности:

`[<`*изображение-маркер* `(list-style-image)>] [<`*стиль* `(list-style-type)>]`
`[<`*местоположение* `(list-style-position)>]`

Пример:

```
ol { list-style: square inside; }
```

Эти же параметры можно задать и для отдельного пункта списка. Вот пример указания для последнего пункта нумерованного списка нумерации в виде арабских цифр:

Пример	Результат

`ol {`
`  list-style-type: upper-roman;`
`}`
`ol li:last-child {`
`  list-style-type: decimal;`
`}`
	I. Посетите наш суши-бар между 16:00 и 19:00.
	II. Купите три суши или ролла.
	3. Получите четвертый в подарок!

Псевдоэлемент `::marker`, описанный в *разд. 13.6*, указывает на маркер или нумерацию пунктов списка. С его помощью можно писать стили, задающие оформление маркеров (нумерации). В таких стилях можно использовать атрибуты стиля, указывающие параметры шрифта, включая цвет (см. *разд. 14.1*), управляющие обработкой пробельных символов (см. *разд. 14.4.5*) и задающие генерируемое содержание (см. *разд. 14.6*).

Пример	Результат

`ol li::marker {`
`  font-size: larger;`
`  font-weight: bold;`
`}`
	1. Суши;
	2. сашими;
	3. роллы.

Отдельные величины в значении атрибута стиля `list-style` можно указать в произвольном порядке.

16.1.2. Параметры просветов

По умолчанию списки выводятся со значительными вертикальными внешними просветами сверху и внизу. Точные значения этих просветов различаются у разных веб-обозревателей. Задать свои значения просветов можно, воспользовавшись атрибутами стиля `margin-top` и `margin-bottom`.

Величину горизонтального просвета между левой границей списка и левыми границами содержимого его пунктов можно регулировать, указывая у списка значение внутреннего просвета слева посредством атрибута стиля `padding-left`.

Пример	Результат

`ul {`
`  padding-left: 10pt;`
`}`
	Блюда:
	• Суши;
	• сашими;
	• роллы.

`ul {`
`  padding-left: 30pt;`
`}`
	Блюда:
	• Суши;
	• сашими;
	• роллы.

К сожалению, регулировать просветы между маркерами (нумерацией), левой границей списка и левыми границами текстового содержимого пунктов нельзя.

16.2. Параметры таблиц и их составляющих

Таблицы не относятся ни к блочным, ни к встроенным, ни к встроенно-блочным элементам страницы. Таблицы «ходят сами по себе» и поддерживают весьма ограниченный набор атрибутов стиля. То же самое относится к отдельным составляющим таблиц: строкам, ячейкам, заголовку и колонкам.

16.2.1. Параметры самих таблиц

У таблиц мы можем указать следующие параметры:

- параметры текста их ячеек, описанные на *уроке 14*, поскольку задающие их атрибуты стиля являются наследуемыми.

Пример	Результат

`.table1 {`
`  font-style: italic;`
`  text-align: center;`
`}`
	Товар Цена, руб.
	Чукка 40
	Магуро 80

- размеры (в том числе минимальные и максимальные) — атрибутами стиля «семейств» `width` и `height`:

```
table.catalog {
    width: 50%;
    min-width: 10cm;
}
```

  Если размеры не указаны явно, таблица получит размеры, необходимые, чтобы вместить все ее содержимое;

- внешние просветы — атрибутами стиля «семейства» `margin`;
- фон.

Пример	Результат

`.table2 {`
`  color: white;`
`  background-color: black;`
`}`
	Товар Цена, руб.
	Чукка 40
	Магуро 80

- рамку вокруг самой таблицы — атрибутами стиля «семейства» `border`.

Пример	Результат

`.table3 {`
`  border: 2px solid black;`
`}`
	Товар Цена, руб.
	Чукка 40
	Магуро 80

Отметим, что рамка рисуется только у таблицы, но не у ее ячеек. Для ячеек нам придется задавать рамки отдельно;

- просвет между рамками отдельных ячеек — наследуемым атрибутом стиля `border-spacing`. В качестве его значения можно указать:
  - числовую величину в какой-либо единице измерения — задаст величину просвета и по горизонтали, и по вертикали.

Пример	Результат

`.table4 {`
`  border: 2px solid black;`
`  border-spacing: 8px;`
`}`
`.table4 th, .table4 td {`
`  border: 1px dashed black;`
`}`
	Товар Цена, руб.
	Чукка 40
	Магуро 80

  - две числовых величины через пробел — первая укажет просвет по горизонтали, вторая — по вертикали.

Пример	Результат

`.table5 {`
`  border: 2px solid black;`
`  border-spacing: 24px 8px;`
`}`
`.table5 th, .table5 td {`
`  border: 1px dashed black;`
`}`
	Товар Цена, руб.
	Чукка 40
	Магуро 80

Значение по умолчанию: `0`;

- режим рисования рамок между ячейками — с применением наследуемого атрибута стиля `border-collapse`. Вот поддерживаемые им значения:
  - `separate` — отдельная рамка рисуется вокруг каждой ячейки.

Пример	Результат

`.table6 {`
`  border: 2px solid black;`
`  border-collapse: separate;`
`}`
`.table6 th, .table6 td {`
`  border: 1px dashed black;`
`}`
	Товар Цена, руб.
	Чукка 40
	Магуро 80

- `collapse` — между ячейками проводятся разделяющие линии.

Пример	Результат

`table7 {`
`  border: 2px solid black;`
`  border-collapse: collapse;`
`}`
`.table7 th, .table7 td {`
`  border: 1px dashed black;`
`}`
	

При этом атрибут стиля `border-spacing` игнорируется.

Значение по умолчанию: `separate`;

- режим обработки пустых ячеек, не имеющих содержимого, — с помощью наследуемого атрибута стиля `empty-cells`, который поддерживает значения:
  - `show` — выводить пустые ячейки, рисуя вокруг них рамки.

Пример	Результат

`.table8 {`
`  empty-cells: show;`
`}`
`.table8 th, .table8 td {`
`  border: 1px solid black;`
`}`
	

  - `hide` — вообще не выводить пустые ячейки.

Пример	Результат

`.table9 {`
`  empty-cells: hide;`
`}`
`.table9 th, .table9 td {`
`  border: 1px solid black;`
`}`
	

Принимается во внимание, если атрибут стиля `border-collapse` имеет значение `separate`.

Значение по умолчанию: `show`;

- режим указания размеров — посредством ненаследуемого атрибута стиля `table-layout`, который поддерживает значения:
  - `auto` — размеры таблицы устанавливаются такими, чтобы полностью вместить ее содержимое. Если размеры, заданные у таблицы в атрибутах стиля `width` и `height`, слишком малы, они будут соответственно увеличены;

- `fixed` — таблица в любом случае будет иметь размеры, заданные в атрибутах стиля `width` и `height`. Однако при этом часть содержимого может выйти за пределы таблицы или быть обрезана (в зависимости от указанного поведения при переполнении, подробности — в *разд. 15.8*).

Значение по умолчанию: `auto`.

16.2.2. Параметры строк таблицы

Строки таблиц — элементы весьма специфические. Для них мы можем задать следующие параметры:

- параметры текста, рассмотренные на *уроке 14*;
- фон — этим фоном будут заполнены ячейки, содержащиеся в этой строке.

Пример	Результат

`.table10 th, .table10 td {`
`  border: 1px solid black;`
`}`
`.table10 tr:first-child {`
`  background-color: lightgrey;`
`}`
	Товар | Цена, руб.
	Чукка |
	Магуро | 80
	Тобико |

- высоту.

Пример	Результат

`.table11 th, .table11 td {`
`  border: 1px solid black;`
`}`
`.table11 tr:last-child {`
`  height: 4em;`
`}`
	Товар | Цена, руб.
	Чукка |
	Магуро | 80
	Тобико |

16.2.3. Параметры ячеек таблиц

Ячейки таблиц — также весьма специфические элементы. Задать у них можно:

- параметры текста, рассмотренные на *уроке 14*;
- размеры:

```
.table12 td.wide { width: 200px; }
```

Минимальные и максимальные размеры у ячеек таблиц задать нельзя — соответствующие атрибуты стиля будут проигнорированы;

- фон — отличный от фона самой таблицы.

Пример	Результат

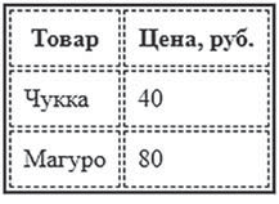
`.table13 {`
`  color: white;`
`  background-color: black;`
`}`
`.table13 tr td:last-child {`
`  color: black;`
`  background-color: lightgrey;`
`}`
	

- рамки. Примеры из *разд. 16.2.1* показывают, что у ячеек можно задавать рамки с параметрами, отличными от параметров рамки самой таблицы;
- вертикальное выравнивание содержимого ячеек — атрибутом стиля `vertical-align` (см. *разд. 14.4.2*). Поддерживаются значения: `top` (по верхнему краю ячейки), `middle` (по центру) и `bottom` (по нижнему краю). Значение по умолчанию: `middle`. Пример:

```
.table14 td { vertical-align: top; }
```

- просветы между границами ячеек (по которым проводятся рамки) и их содержимым — посредством атрибутов стиля «семейства» `padding`:

Пример	Результат

`.table15 {`
`  border: 2px solid black;`
`}`
`.table15 th, .table15 td {`
`  border: 1px dashed black;`
`  padding: 8px;`
`}`
	

16.2.4. Параметры заголовка таблицы

У заголовка таблицы, создаваемого тегом `<caption>` (см. *разд. 8.6*), можно указать параметры текста, выравнивания, фона и внутренние просветы.

Следует лишь отметить, что фон, указанный для самой таблицы, к заголовку автоматически не применяется, и его придется задавать явно.

Пример	Результат

`.table16 {`
`  color: white;`
`  background-color: black;`
`}`
`.table16 caption {`
`  font-style: italic;`
`  text-align: right;`
`  background-color: grey;`
`  padding: 2pt;`
`}`
	Прайс-лист
	Товар Цена, руб.
	Чукка 40
	Магуро 80

Дополнительно можно задать местоположение заголовка, воспользовавшись наследуемым атрибутом стиля `caption-side`. Вот поддерживаемые им значения:

- `top` — заголовок помещается над таблицей;
- `bottom` — заголовок помещается под таблицей.

Значение по умолчанию: `top`.

Пример	Результат

`.table17 caption {`
`  caption-side: bottom;`
`}`
	Товар Цена, руб.
	Чукка 40
	Магуро 80
	Прайс-лист

16.2.5. Параметры групп колонок и отдельных колонок

У групп колонок и отдельных колонок, создаваемых тегами `<colgroup>` и `<col>` (см. *разд. 8.7*), можно указать:

- ширину;
- фон;
- рамку — которая будет нарисована вокруг всей колонки или группы колонок.

16.3. Самостоятельные упражнения

- Укажите у пунктов нумерованного списка, находящегося на главной странице сайта суши-бара и перечисляющего шаги, которые необходимо выполнить для участия в рекламной акции, нумерацию в виде римских цифр, а у самой нумерации, — цвет текста `orange`.

✓ У вас должно получиться ⌄

I. Посетите наш суши-бар между 16:00 и 19:00.

Время действительно только для будних дней.
В выходные — с 14:00 до 20:00.

II. Купите три суши или ролла.
III. Получите четвертый в подарок!

- Задайте у первых ячеек таблиц-каталогов блюд (в которых выводятся иллюстрации) ширину, равную 33% от ширины родителя.

Урок 17. Гиперссылки, изображения и мультимедиа

Параметры гиперссылок.
Параметры графических изображений.
Параметры аудио и видео.

17.1. Параметры гиперссылок

Гиперссылки не имеют каких-либо специфических параметров оформления. Для оформления их текстового содержимого используются атрибуты стиля, рассмотренные на *уроке 14*.

Однако имеется возможность применить какой-либо стиль к гиперссылке еще не посещенной или, наоборот, посещенной. Для этого достаточно использовать в селекторах соответствующих стилей следующие псевдоклассы:

- `:link` — указывает на *непосещенные* гиперссылки (по которым еще ни разу не выполнялся переход):

```
/* Делаем непосещенные гиперссылки зелеными */
a:link { color: green; }
```

- `:visited` — указывает на посещенные гиперссылки;
- `:any-link` — указывает на непосещенные и посещенные гиперссылки;
- `:active` — указывает на *активную* гиперссылку (на которой выполняется щелчок мышью);

- `:target` — указывает на элемент, на который был выполнен переход по щелчку на гиперссылке с якорем (о якорях рассказывалось в *разд. 9.1.4*):

```
/*
    Выделяем элемент, на который был выполнен переход по щелчку
    на гиперссылке с якорем, тонкой красной рамкой
*/
*:target { border: 1px solid red; };
```

17.2. Упражнение. Оформление гиперссылок в панели навигации

Гиперссылки, находящиеся в панели навигации, принято оформлять особым образом — не так, как находящиеся в обычном тексте. Как правило, у таких гиперссылок убирают подчеркивание, задают цвет текста, отличный от стандартного синего, непосещенные и посещенные гиперссылки оформляют одинаково, а активную гиперссылку и гиперссылку, находящуюся под курсором мыши, делают как можно более заметными.

Давайте и мы сделаем подобного рода оформление у гиперссылок, входящих в состав панели навигации сайта суши-бара.

1. Найдем в папке 16\s16.3 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Сначала необходимо убрать у всех гиперссылок из панели навигации подчеркивание и закрасить каким-либо цветом — с одной стороны, заметным, а с другой, не используемым в остальном содержании страницы. Выберем для этого красивый и контрастный цвет `maroon`.

2. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и добавим стиль, который укажет оформление у всех гиперссылок:

```
nav a {
    color: maroon;
    text-decoration: none;
}
```

   Теперь необходимо выделить активную гиперссылку и гиперссылку под курсором мыши, чтобы они сразу бросались в глаза. Для этого часто применяется своего рода «негативное» представление гиперссылок, при котором фон закрашивается цветом, использованным для вывода текста, а текст — цветом фона. Сделаем так и мы.

3. Добавим стиль для активной гиперссылки и гиперссылки под курсором мыши:

```
nav a:active, nav a:hover {
    color: blanchedalmond;
    background-color: maroon;
}
```

✓ Результат ›

Курсор мыши наведен на гиперссылку «Сашими».

17.3. Параметры графических изображений

Изображения относятся к встроенно-блочным элементам, причем весьма специфическим (поскольку они не содержат текста). У них мы можем указать довольно ограниченный набор параметров:

- размеры, в том числе минимальные и максимальные, — с помощью атрибутов стиля «семейств» `width` и `height`:

```
.img1 {
    width: 25%;
    max-width: 300px;
    max-height: 200px;
}
```

- внешние, внутренние просветы и рамку:

Пример	Результат

`.img2 {`
`  border: 3px solid black;`
`  margin: 0px 20px;`
`  padding: 5px;`
`}`
	Безалкогольные напитки.
	 
	В их числе:

- вертикальное выравнивание — атрибутом стиля `vertical-align`:

Пример	Результат

`.img3 {`
`  vertical-align: text-top;`
`}`
	Безалкогольные напитки  :

Если размеры изображения, выводимого в теге <img>, отличаются от размеров, указанных у самого этого тега (в атрибутах width и height тега <img> или атрибутах стиля одноименных «семейств»), можно задать режимы масштабирования и позиционирования изображения в стиле, действующем на тег <img>.

Масштабирование изображения в теге <img> устанавливается ненаследуемым атрибутом стиля object-fit. Поддерживаемые им значения приведены в табл. 17.1.

Таблица 17.1

Значение	Описание	Пример
none	Изображение не масштабируется	
contain	Изображение масштабируется так, чтобы заполнить тег без нарушения своих пропорций. При этом какие-то части тега могут оказаться не заполненными изображением	
cover	Изображение масштабируется так, чтобы заполнить тег без нарушения своих пропорций. При этом какие-то части изображения могут выйти за границы тега и, таким образом, не будут видимы	
fill	Изображение масштабируется так, чтобы заполнить тег. Однако пропорции изображения могут быть нарушены	
scale-down	Если изображение меньше тега, оно не масштабируется (как в случае указания значения none), если больше — масштабируется с сохранением пропорций (аналогично contain)	

Значение по умолчанию: fill.

Позиционировать изображение в теге <img> имеет смысл лишь в том случае, если размеры изображения меньше размеров тега. Позиционирование выполняется наследуемым атрибутом стиля object-position. Его значение можно указать в виде:

- *<местоположение по горизонтали>* — в виде одного из предопределенных значений: left (у левого края тега), center (в его центре) или right (у правого края тега). По вертикали изображение всегда центрируется.

Пример	Результат

`.img1 {`
`  object-fit: none;`
`  object-position: right;`
`}`
	

- *<местоположение по горизонтали> <местоположение по вертикали>* — разделенные пробелом. *Местоположение по вертикали* задается как `top` (у верхнего края тега), `center` (в его центре) или `bottom` (у нижнего края).

Пример	Результат

`.img2 {`
`  object-fit: none;`
`  object-position: right bottom;`
`}`
	

- числовой величины в какой-либо единице измерения — укажет горизонтальную координату изображения. Координата отсчитывается вправо от левого верхнего угла тега. По вертикали изображение центрируется.

Пример	Результат

`.img3 {`
`  object-fit: none;`
`  object-position: 20%;`
`}`
	

- двух числовых величин через пробел — укажут горизонтальную и вертикальную координаты изображения. Горизонтальная координата отсчитывается вправо от левого верхнего угла тега, вертикальная — вниз.

Пример	Результат

`.img4 {`
`  object-fit: none;`
`  object-position: 20% 75px;`
`}`
	

Как видно из примера, при определенных значениях местоположения часть изображения (и даже все изображение целиком) может оказаться за границами тега `<img>`;

- комбинации из числового и предопределенного значений через пробел. Первое значение задаст горизонтальную координату, второе — вертикальную.

Пример	Результат

`.img5 {`
`  object-fit: none;`
`  object-position: 20% top;`
`}`
	

`.img6 {`
`  object-fit: none;`
`  object-position: right 75px;`
`}`
	

- *<вертикальная сторона> <местоположение по горизонтали>*

  <горизонтальная сторона> <местоположение по вертикали>

  Местоположение по горизонтали задается в виде просвета между указанной *вертикальной стороной* тега и соответствующей стороной изображения. В качестве *вертикальной стороны* можно указать `left` (левая) или `right` (правая).

  Местоположение по вертикали задается в виде просвета между указанной *горизонтальной стороной* тега и соответствующей стороной изображения. В качестве *горизонтальной стороны* можно указать `top` (верхняя) или `bottom` (нижняя).

Пример	Результат

`.img7 {`
`  object-fit: none;`
`  object-position: left 25px bottom 15px;`
`}`
	

`.img8 {`
`  object-fit: none;`
`  object-position: right 40% top 30%;`
`}`
	

Значение по умолчанию: `50% 50%`.

17.4. Параметры аудио и видео

Аудио и видео — теги `<audio>` и `<video>` — также относятся к встроенно-блочным элементам. У них можно указать только величины внешних и внутренних просветов и рамку.

У видеоролика можно установить режимы масштабирования и позиционирования видео, воспользовавшись атрибутами стиля `object-fit` и `object-position` (см. *разд. 17.3*).

Отдельные части аудио- или видеоролика, такие, как кнопки управления и панель для просмотра видео, оформить средствами CSS, к сожалению, невозможно.

17.5. Самостоятельные упражнения

- Укажите у тегов `<img>`, в которых выводятся иллюстрации блюд, размеры 100 × 100% и масштабирование изображений с сохранением пропорций и без выхода за пределы тега.
- Укажите у видеокиоска, сделанного при выполнении *упражнения 7.4* (папка 7\ex7.4\kiosk), масштабирование видео с сохранением пропорций и возможным выходом за пределы тега. Благодаря этому, удастся убрать черные полосы по краям видео, правда, за счет того, что часть видео выйдет за края проигрывателя.

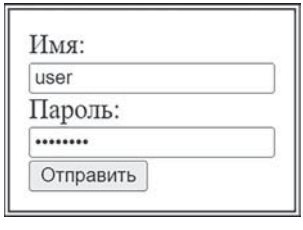
Урок 18. Оформление веб-форм и элементов управления

Параметры веб-форм.
Параметры элементов управления.
Параметры декоративных элементов.
Параметры спойлеров.
Параметры специальных элементов.

18.1. Параметры веб-форм

Веб-формы являются блочными элементами страницы. Следовательно, мы можем указать для них любые параметры оформления, о которых шел разговор на *уроках 14* и *15*: параметры шрифта, строк, размеры, просветы, рамку и др.

Пример	Результат

`form {`
`  font-size: 14pt;`
`  padding: 10pt;`
`  border: thick double black;`
`  width: min-content;`
`}`
	

К сожалению, параметры шрифта, указанные у веб-формы, не наследуются содержащимися в ней элементами управления. У элементов управления эти параметры придется задавать отдельно.

18.2. Параметры элементов управления

Элементы управления являются встроенно-блочными элементами страницы. Мы можем указать у них все параметры оформления, применимые к элементам такого рода: настройки шрифта, строк, размеры, просветы, рамку и др.

Пример	Результат

`input {`
`  font: 14pt monospace;`
`  text-align: center;`
`  border: solid black;`
`  background-color: lightgrey;`
`}`
`input[type=text], input[type=password] {`
`  width: 100pt;`
`  padding: 2pt;`
`}`
`input[type=submit] {`
`  font-weight: bold;`
`}`
	

CSS поддерживает ряд псевдоклассов, которые применимы лишь к элементам управления:

- `:focus` — указывает на элемент управления, который в настоящий момент имеет фокус ввода. Вот пример выделения такого элемента управления красной рамкой:

  ```
  input:focus { border: solid red; }
  ```

- `:focus-visible` — указывает на элемент управления, который в настоящий момент имеет фокус ввода и который, по мнению веб-обозревателя, должен быть визуально помечен как имеющий фокус ввода.

  Например, с точки зрения веб-обозревателя, поле ввода должно быть визуально помечено как имеющее фокус ввода, — независимо от того, как оно получило его: посредством щелчка мышью, нажатия клавиши <Tab> или комбинации клавиш <Shift>+<Tab>. Напротив, кнопка и список должны быть помечены таким образом лишь в случае, если они получили фокус ввода при нажатии <Tab> или <Shift>+<Tab>, а при щелчке мышью помечать их излишне;

- `:required` — указывает на поля ввода, области редактирования, обязательные для заполнения, и списки, в которых необходимо выбрать хотя бы один пункт (у которых в тегах `<input>`, `<textarea>` и `<select>` поставлен атрибут `required`);
- `:optional` — указывает на поля ввода, области редактирования, необязательные для заполнения, и списки, в которых нет необходимости делать выбор (у которых в тегах `<input>`, `<textarea>` и `<select>` отсутствует атрибут `required`);
- `:read-write` — указывает на элементы управления, доступные и для чтения, и для записи;
- `:read-only` — указывает на элементы управления, доступные только для чтения:

```
input:read-only { background-color: lightgrey; }
```

- `:enabled` — указывает на элементы управления, доступные для ввода;
- `:disabled` — указывает на элементы управления, недоступные для ввода;
- `:valid` — указывает на элементы управления, в которые занесены корректные значения, а также помеченные как обязательные к заполнению и заполненные значениями.

  Нужно помнить, что этот псевдокласс будет указывать на все элементы управления с корректными изначальными значениями непосредственно после открытия страницы, — еще до того, как пользователь начнет с ними взаимодействовать;
- `:user-valid` — то же самое, что и `:valid`, только указывает лишь на те элементы управления, с которыми пользователь уже взаимодействовал;
- `:invalid` — указывает на элементы управления, в которые занесены некорректные значения (например, на поле ввода числа, в которое было введено что-либо, отличное от числа), а также на помеченные как обязательные к заполнению и оставшиеся незаполненными.

  Имейте в виду, что этот псевдокласс будет указывать на все элементы управления с некорректными изначальными значениями сразу после загрузки страницы, — еще до того, как пользователь начнет с ними взаимодействовать;
- `:user-invalid` — то же самое, что и `:invalid`, только указывает лишь на те элементы управления, с которыми пользователь уже взаимодействовал:

```
*:user-invalid { background-color: red; }
```

- `:in-range` — указывает на поля ввода числа и регуляторы, в которых указаны значения, укладывающиеся в заданные границы (они задаются атрибутами `min` и `max` тега `<input>`);
- `:out-of-range` — указывает на поля ввода числа и регуляторы, в которых указаны значения, не укладывающееся в заданные границы;
- `:checked` — указывает на установленные флажки, переключатели и выбранные пункты списков:

```
input:checked + label {
    font-weight: bold;
}
. . .
<input type="checkbox" id="urgent">
<label for="urgent">Позвоните мне побыстрее</label>
```

Результат: флажок сброшен	Результат: флажок установлен
☐ Позвоните мне побыстрее	☑ **Позвоните мне побыстрее**

- `:autofill` — указывает на поле ввода, значение в которое было подставлено путем выбора из списка автодополнения;
- `:default` — указывает на установленные флажки, переключатели, выбранные пункты списков и *первую* кнопку отправки данных в веб-форме;
- `:placeholder-shown` — указывает на пустые поля ввода и области редактирования, в которых в настоящий момент выведен текст подсказки.

Кроме того, поддерживаются два полезных псевдоэлемента:

- `::placeholder` — указывает на текст подсказки, которая выводится непосредственно в поле ввода или области редактирования.

Пример	Результат

`input::placeholder {`
`  font-style: italic;`
`  text-align: center;`
`}`
`input:placeholder-shown {`
`  border-style: dotted;`
`}`
	Ваш телефон

- `::file-selector-button` — указывает на кнопку **Выберите файл**, присутствующую в составе поля выбора файлов.

Наследуемый атрибут стиля `accent-color` задает цвет:

- заполнения — в установленных флажках и переключателях;
- закраски левой («заполненной») части шкалы — у регуляторов и индикаторов процесса.

В качестве его значения можно указать:

- нужный цвет — в любом из форматов, поддерживаемых CSS (см. *разд. 12.2.2*);
- `auto` — цвет будет выбран веб-обозревателем.

Значение по умолчанию: `auto`.

Пример	Результат
`input[type=checkbox] { accent-color: auto; }`	
`input[type=checkbox] { accent-color: lightgrey; }`	

Наследуемый атрибут стиля `caret-color` задает цвет текстового курсора в поле ввода или области редактирования. В качестве значения можно указать цвет в любом поддерживаемом формате (см. *разд. 12.2.2*) или `auto` (цвет будет выбран веб-обозревателем). Значение по умолчанию: `auto`. Пример:

```
textarea { caret-color: blue; }
```

Пункты списка (теги `<option>`) оформить средствами CSS нельзя. Однако эти элементы унаследуют все наследуемые атрибуты стиля, указанные у родителя — списка.

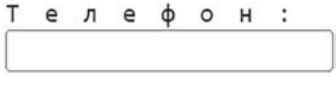
Пример	Результат

`select {`
`  font-weight: bold;`
`  font-style: italic;`
`  font-variant: small-caps;`
`}`
	

18.2.1. Параметры декоративных элементов

Надпись у элемента управления (тег `<label>`, см. *разд. 10.3.12.1*) — это обычный встроенный элемент страницы, и его можно оформлять так же, как и любой другой встроенный элемент.

Пример	Результат

`label {`
`  font-family: monospace;`
`  letter-spacing: 1em;`
`}`
	

Группа элементов управления (тег <fieldset>, см. *разд. 10.3.12.2*) — это обычный блочный элемент управления и оформляется так же, как и любой другой подобный элемент.

Пример	Результат

fieldset {
  border: none;
  background-color: lightgrey;
  width: min-content;
}
fieldset legend {
  color: white;
  background-color: black;
}
	

18.2.2. Полное изменение внешнего вида элементов управления

В ряде случаев возникает необходимость сделать у элементов управления полностью оригинальное оформление.

Предварительно следует убрать у элементов управления оформление по умолчанию. Для этого достаточно в стиле, применяемом к элементам, указать атрибут стиля appearance со значением none.

Пример стилей, задающих оригинальное оформление для флажков:

```
.super-checkbox {
    appearance: none;
    width: 1.5em;
    height: 1.5em;
    background-color: white;
    outline: solid black;
    outline-offset: 2pt;
    margin-right: 8pt;
    vertical-align: middle;
}
.super-checkbox:checked {
    background-color: black;
}
. . .
<input type="checkbox" id="super" class="super-checkbox">
<label for="super">Супер-флажок</label>
```

Результат: флажок сброшен	Результат: флажок установлен
 Супер-флажок	 Супер-флажок

Значение атрибута стиля `appearance` по умолчанию: `auto` (выводить элемент управления с оформлением по умолчанию).

18.3. Упражнение. Оформление веб-формы

Наш сайт суши-бара уже смотрится хорошо. Вот только веб-форма заказа блюд выглядит слишком уж непрезентабельно... Давайте оформим ее!

1. Найдем в папке 17\s17.4 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   У самой веб-формы укажем минимально необходимую ширину, внутренние просветы слева и справа в 10 пунктов, тонкую штриховую рамку цвета `orange` с радиусом скругления углов в 5 пунктов.

   Внутренние просветы сверху и внизу указывать не будем, так как для размещения элементов управления в форме ранее применили абзацы. А абзацы по умолчанию выводятся со значительными внешними просветами сверху и снизу.

   Кроме того, зададим у веб-формы внешний просвет снизу, равный 10 пунктам, чтобы нижняя сторона рамки не слилась с адресом суши-бара, находящемся в поддоне.

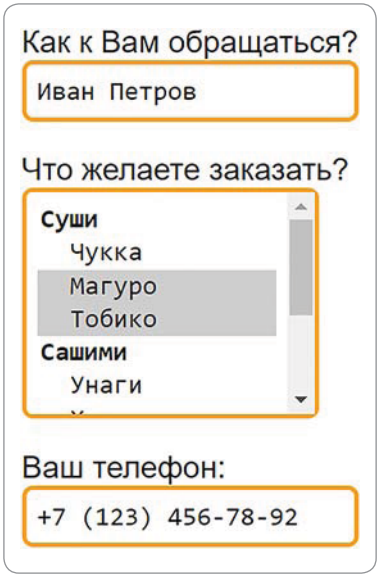
2. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и добавим стиль, который сделает веб-форму красивее:

```
form {
    width: min-content;
    padding: 0 10pt;
    margin-bottom: 10pt;
    border: thin dotted orange;
    border-radius: 5pt;
}
```

   У элементов управления также укажем рамку цвета `orange` с радиусом скругления углов в 5 пунктов, но на этот раз уже сплошную и средней толщины. А еще зададим моноширинный шрифт с кеглем 12 пунктов и внутренний просвет со всех сторон в 5 пунктов.

3. Добавим необходимый стиль:

```
input, select, textarea {
    font: 12pt monospace;
    border: medium solid orange;
    border-radius: 5pt;
    padding: 5pt;
}
```



✓ Что получилось? ›

Приведены лишь поле ввода имени заказчика, список блюд и поле для занесения телефона.

Элементы управления, обязательные для заполнения, в настоящее время обычно помечают символом звездочки. Давайте и мы так сделаем.

Поместим после каждого элемента управления, обязательного к заполнению, пустой встроенный контейнер (тег `<span>`). В этом контейнере и будем выводить звездочки, которые оформим как генерируемое содержание.

4. Откроем страницу заказа pages\order.html в текстовом редакторе и добавим пустые встроенные контейнеры после каждого «обязательного» элемента управления:

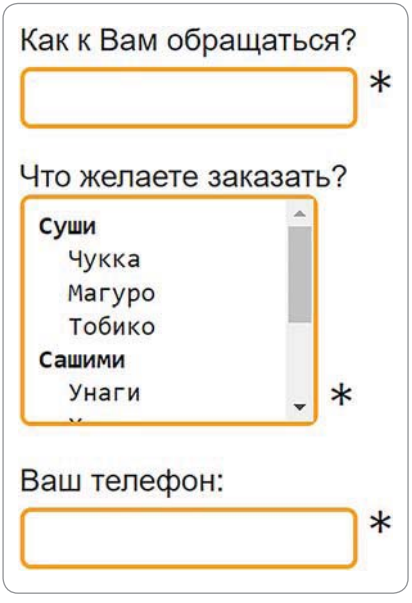
```
<p>
    <label for="name">Как к Вам обращаться?</label><br>
    <input name="name" id="name" required>
    <span></span>
</p>
<p>
    <label for="dishes">Что желаете заказать?</label><br>
    <select name="dishes" id="dishes" size="6" multiple required>
        . . .
    </select>
    <span></span>
</p>
<p>
    <label for="phone">Ваш телефон:</label><br>
    <input type="tel" name="phone" id="phone" required>
    <span></span>
</p>
```

У выводимых звездочек укажем моноширинный шрифт, как и у элементов управления, только покрупнее — 24 пункта. Так мы сделаем их заметнее.

5. Переключимся на таблицу стилей и допишем в нее стиль, помечающий элементы управления, которые должны быть заполнены:

```
input:required + span::after, select:required + span::after {
    font: 24pt monospace;
    content: '*';
}
```

✓ Что получилось? ›

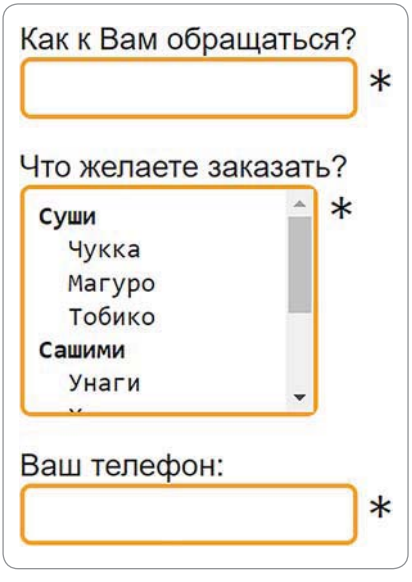


Все хорошо, только у списка звездочка сиротливо маячит где-то внизу... Поднимем ее, указав вертикальное выравнивание по верху (атрибут стиля `vertical-align` со значением `top`).

6. Дополним ранее написанный стиль:

```
input:required + span::after, ↳
select:required + span::after {
    font: 24pt monospace;
    content: '*';
    vertical-align: top;
}
```

✓ Результат ›



Напоследок поможем будущим посетителям сайта вводить данные в веб-форму. Пометим элементы управления с корректными данными зеленой рамкой, а содержащие некорректные данные — красной. Поскольку мы хотим, чтобы элементы управления становились помеченными лишь после того, как посетитель введет в них данные, в селекторах соответствующих стилей применим псевдоклассы `:user-valid` и `:user-invalid`.

7. Напишем стили, задающие необходимое оформление:

```
input:user-valid, select:user-valid {
    border-color: green;
}
input:user-invalid, select:user-invalid {
    border-color: red;
}
```

Занесем что-либо в поле ввода **Как к Вам обращаться**, выберем какие-либо пункты в списке **Что желаете заказать**, оставим пустым поле **Ваш телефон** и нажмем кнопку **Заказать**. Первое поле ввода и список будут помечены зелеными рамками, а второе поле ввода — красной рамкой.

18.4. Параметры спойлера

Спойлер, создаваемый тегом `<details>` (см. *разд. 10.5*), — это блочный элемент страницы, и мы можем указать для него любые настройки оформления, применяемые к блочным элементам.

Заголовок спойлера, который создается тегом `<summary>`, — это фактически пункт списка. У него можно изменить маркер, выводимый левее текста заголовка и отмечающий состояние спойлера, воспользовавшись атрибутом стиля `list-style-type` (см. *разд. 16.1.1*).

Чтобы применить стиль к развернутому спойлеру, достаточно использовать в соответствующем стиле селектор по атрибуту тега `open`.

Пример:

```
details {
    padding: 4pt;
    border: thin solid black;
    width: fit-content;
}
details summary {
    margin: 2pt;
    padding: 4pt;
    color: white;
    background-color: black;
}
details[open] {
    border-width: thick;
}
```

✓ Результат

Спойлер свернут	Спойлер развернут
▶ HTML	▼ HTML Язык, применяемый для написания веб-страниц.

18.5. Параметры специальных элементов

У индикатора процесса (тег `<progress>`, см. *разд. 10.6.1*) и метра (тег `<meter>`, см. *разд. 10.6.2*) можно указать лишь ширину и внешние просветы. Также можно управлять их высотой, указывая величины внутренних просветов сверху и снизу.

Пример	Результат

`.progress1 {`
`  width: 150pt;`
`}`
	

`.progress2 {`
`  padding: 20pt;`
`  width: 150pt;`
`}`
	

Область для вывода результатов вычислений (тег `<output>`) является встроенным элементом, а область для вывода результатов поиска (тег `<search>`) — блочным (см. *разд. 10.6.3*). У них можно указать настройки оформления, применимые к элементам соответствующей разновидности.

18.6. Самостоятельное упражнение

- Сделайте у веб-формы заказа на сайте суши-бара флажок, аналогичный представленному в *разд. 18.2.2*, только цвета `orange`, поменьше (размерами `1ex×1ex`) и без скругленных углов (они задаются стилем, написанным при выполнении *упражнения 18.3*).

ПОДСКАЗКА

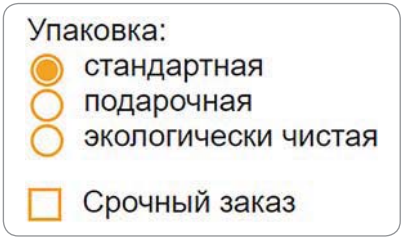
Чтобы убрать скругленные углы, используйте атрибут стиля `border-radius` со значением `0`.

- Сделайте там же переключатели, аналогичные этому флажку, только круглые.

ПОДСКАЗКА

Чтобы создать круглый элемент страницы, поместите в действующий на него стиль атрибут стиля `border-radius` со значением `50%`.

✓ У вас должно получиться ⌄



Урок 19. Фоны

Сплошной фон.
Градиентный фон.
Линейные, радиальные и угловые градиенты.
Графический фон.
Смешивание.
Множественные фоны.

ВНИМАНИЕ!

Все атрибуты стиля, рассматриваемые на этом уроке, применимы к элементам любых разновидностей и являются ненаследуемыми.

19.1. Сплошной фон

Проще всего сделать у элемента страницы *сплошной* цветной фон.

> ***Сплошной фон***
>
> Однотонная заливка указанным цветом.

Сплошной фон создается с применением атрибута стиля `background-color`, который давно нам знаком. В качестве его значения можно указать любой цвет из поддерживаемых CSS (см. *разд. 12.2.2*). Значение по умолчанию: `transparent` (прозрачный цвет).

Пример	Результат

`border: 20px dotted black;`
`padding: 20px;`
`background-color: lightgrey;`
	Элемент со сплошным фоном

Атрибут стиля `background-clip` указывает область элемента страницы, заполняемую фоном:

- `border-box` — пространство под рамкой, внутренние просветы и содержимое элемента.

Пример	Результат

`border: 20px dotted black;`
`padding: 20px;`
`background-color: lightgrey;`
`background-clip: border-box;`
	

- `padding-box` — только внутренние просветы и содержимое элемента.

Пример	Результат

`border: 20px dotted black;`
`padding: 20px;`
`background-color: lightgrey;`
`background-clip: padding-box;`
	

- `content-box` — только содержимое элемента.

Пример	Результат

`border: 20px dotted black;`
`padding: 20px;`
`background-color: lightgrey;`
`background-clip: content-box;`
	

Значение по умолчанию: `border-box`.

Атрибут стиля `background` можно использовать для указания обоих параметров сплошного фона. Эти параметры записываются через пробел в последовательности:

`<`*цвет* (`background-color`)`>` `[<`*заполняемая область* (`background-clip`)`>]`

Пример:

```
.sbg4 { background: lightgrey content-box; }
```

19.2. Градиентные фоны

Градиентные фоны выглядят эффектно и привлекают внимание к странице.

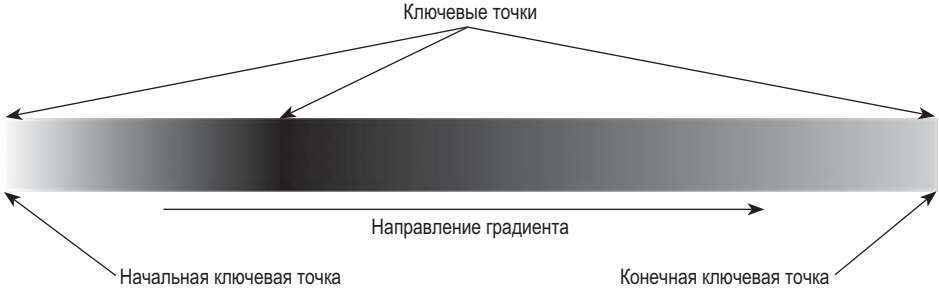
> ***Градиент***
>
> Вид фона, в котором один цвет постепенно переходит в другой, потом в третий и т. д.

Для указания градиентного фона предназначается атрибут стиля `background-image`. Также можно использовать атрибут стиля `background`.

19.2.1. Линейный градиент

> ***Линейный градиент***
>
> Градиент, в котором переходящие друг в друга цвета располагаются на прямой линии.
>
> ***Ключевая точка***
>
> Точка градиента, в которой присутствует указанный «чистый» цвет.

Пример линейного градиента, в котором белый цвет переходит в черный, а потом — в светло-серый (градиент с тремя *ключевыми точками*):



Линейный градиент описывается двумя параметрами: направлением (снизу вверх, слева направо и т. п.) и набором ключевых точек. Каждая ключевая точка в наборе, в свою очередь, описывается ее местоположением относительно начала градиента и значением «чистого» цвета, который должен в ней присутствовать.

Линейный градиент создается функцией `linear-gradient()`, записываемой в формате:

```
linear-gradient([<направление>, ]
                <описания ключевых точек, разделенные запятыми>)
```

Эта функция используется в качестве значения атрибута стиля `background-image` или `background`.

Параметры создаваемого линейного градиента, записываемые в функции `linear-gradient()`:

- *направление* градиента может быть задано в виде:
  - предопределенных значений: `to top` (снизу вверх), `to right` (слева направо), `to bottom` (сверху вниз) или `to left` (справа налево).

Пример	Результат

`background-image:`
`  linear-gradient(to right,`
`  white 0%, grey 30%,`
`  lightgrey 100%);`
	

  - предопределенных значений: `to top right` (из левого нижнего угла в правый верхний), `to bottom right` (из левого верхнего угла в правый нижний), `to bottom left` (из правого верхнего угла в левый нижний) или `to top left` (из правого нижнего угла в левый верхний).

Пример	Результат

`background-image:`
`  linear-gradient(to bottom right,`
`  white 0%, grey 30%,`
`  lightgrey 100%);`
	

  - значения угла, отсчитываемого от горизонтали, в любой поддерживаемой единице измерения (см. *разд. 12.2.1.2.2*). Положительные значения отсчитываются по часовой стрелке, отрицательные — против часовой стрелки.

Пример	Результат

`background-image:`
`  linear-gradient(20deg,`
`  white 0%, grey 30%,`
`  lightgrey 100%);`
	

`background-image:`
`  linear-gradient(-20deg,`
`  white 0%, grey 30%,`
`  lightgrey 100%);`
	

Если *направление* не указано, используется значение `to bottom` (сверху вниз).

Пример	Результат

`background-image:`
`  linear-gradient(white 0%, grey 30%,`
`  lightgrey 100%);`
	

- отдельное *описание ключевой точки* записывается в формате:

  `[<положение относительно начала градиента>] <«чистый» цвет>`

  Положение указывается в любой из поддерживаемых единиц измерения, обычно в процентах относительно размеров элемента.

Пример	Результат

`background-image:`
`  linear-gradient(to right,`
`  white 0%, grey 30%,`
`  lightgrey 100%);`
	

Ключевых точек в градиенте может быть сколько угодно — например, пять.

Пример	Результат

`background-image:`
`  linear-gradient(to right,`
`  white 0%, grey 30%,`
`  black 50%, white 70%,`
`  lightgrey 100%);`
	

Или всего две: начальная и конечная.

Пример	Результат

`background-image:`
`  linear-gradient(to right,`
`  white 0%, black 100%);`
	

Начальная ключевая точка совсем необязательно должна находиться в начале градиента, а конечная — в его конце. В этом случае пространство от начала градиента до начальной ключевой точки будет закрашено цветом,

указанным в начальной ключевой точке, а пространство от конечной ключевой точки до конца градиента — цветом, указанным в конечной ключевой точке.

Пример	Результат

`background-image:`
`  linear-gradient(to right,`
`  white 10%, grey 30%,`
`  lightgrey 70%);`
	

Положение ключевой точки можно не указывать. В таком случае:

- начальная ключевая точка будет находиться в начале градиента;
- конечная ключевая точка будет находиться в конце градиента.

Пример	Результат

`background-image:`
`  linear-gradient(to right,`
`  white, grey 30%,`
`  lightgrey);`
	

- промежуточная ключевая точка будет поставлена точно между предыдущей и следующей ключевыми точками.

Пример	Результат

`background-image:`
`  linear-gradient(to right,`
`  white, grey,`
`  lightgrey);`
	

Если положение не указано у нескольких идущих подряд ключевых точек, они будут равномерно распределены между предыдущей и следующей ключевыми точками с явно указанным местоположением.

Пример	Результат

`background-image:`
`  linear-gradient(to right,`
`  white, black, white,`
`  black, white);`
	

19.2.2. Радиальный градиент

> ***Радиальный градиент***
>
> Градиент, в котором переходящие друг в друга цвета располагаются на концентрических эллипсах, имеющих единый центр.

Пример радиального градиента, в котором белый цвет переходит в черный, потом — в светло-серый, а центр находится в середине элемента:

Радиальный градиент описывается следующими параметрами:

- формой (эллипс или круг);
- местоположением центра, т. е. начала градиента (например, центр или левый верхний угол элемента);
- местоположением окончания градиента (например, ближняя к центру сторона элемента);
- набором ключевых точек с указанием положения и «чистого» цвета у каждой из них.

Для создания радиального градиента служит функция `radial-gradient()`:

```
radial-gradient([[<форма>] [<положение окончания>]
                  [at <положение начала>], ]
                <описания ключевых точек, разделенные запятыми>)
```

Эта функция указывается в качестве значения атрибута стиля `background-image` или `background`.

`Описания ключевых точек` форматируются так же, как и в случае линейного градиента (см. *разд. 19.2.1*).

Остальные параметры создаваемого градиента:

- *форма*:
  - `ellipse` — эллиптический градиент.

Пример	Результат

`background-image:`
`  radial-gradient(ellipse farthest-side`
`  at 50% 0%,`
`  white, black 40%,`
`  lightgrey);`
	

  - `circle` — круглый градиент.

Пример	Результат

`background-image:`
`  radial-gradient(circle farthest-side`
`  at 50% 0%,`
`  white, black 40%,`
`  lightgrey);`
	

  Если *форма* не указана, рисуется эллиптический градиент;

- *положение окончания* градиента:
  - `closest-corner` — угол элемента, ближайший к началу градиента.

Пример	Результат

`background-image:`
`  radial-gradient(closest-corner at 40% 30%,`
`  white, black 40%,`
`  lightgrey);`
	

  - `closest-side` — сторона элемента, ближайшая к началу градиента.

Пример	Результат

`background-image:`
`  radial-gradient(closest-side at 40% 30%,`
`  white, black 40%,`
`  lightgrey);`
	

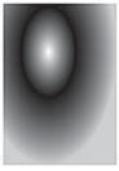
- `farthest-corner` — угол элемента, самый дальний от начала градиента.

Пример	Результат

`background-image:`
`  radial-gradient(farthest-corner at 40% 30%,`
`  white, black 40%,`
`  lightgrey);`
	

- `farthest-side` — сторона элемента, самая дальняя от начала градиента.

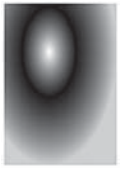
Пример	Результат

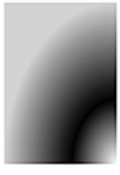
`background-image:`
`  radial-gradient(farthest-side at 40% 30%,`
`  white, black 40%,`
`  lightgrey);`
	

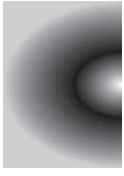
Если *положение окончания* не указано, используется значение по умолчанию `farthest-corner`;

♦ *положение начала* градиента. Указывается в том же формате, что и значение атрибута стиля `object-position` (см. *разд. 17.3*).

Примеры	Результат

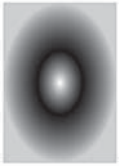
`background-image:`
`  radial-gradient(farthest-side at 40% 30%,`
`  white, black 40%, lightgrey);`
	

`background-image:`
`  radial-gradient(farthest-side at right bottom,`
`  white, black 40%, lightgrey);`
	

`background-image:`
`  radial-gradient(farthest-side at right,`
`  white, black 40%, lightgrey);`
	

Положение начала, вместе со словом `at`, можно не указывать — тогда начало градиента будет находиться в центре элемента (как если бы было задано положение `50% 50%`).

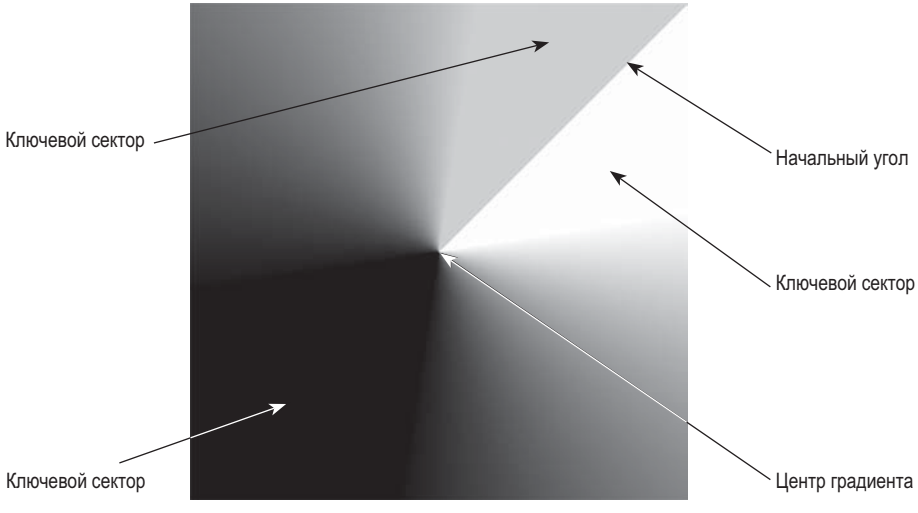
Пример	Результат

`background-image:`
`  radial-gradient(farthest-side,`
`  white, black 40%, lightgrey);`
	

19.2.3. Угловой градиент

> ***Угловой градиент (или конусный градиент)***
>
> Градиент, в котором переходящие друг в друга цвета располагаются по кругу в направлении хода часовой стрелки. Начало такого градиента находится в начальном углу.
>
> ***Начальный угол***
>
> Угол, на котором начинается угловой градиент. Отсчитывается от вертикальной оси.
>
> ***Ключевой сектор***
>
> Сектор углового градиента, в котором присутствует указанный «чистый» цвет. В «вырожденном» случае может представлять собой линию.

Пример углового градиента, в котором белый цвет переходит в черный, потом — в светло-серый, а начальный угол равен 45°:



Угловой градиент описывается следующими параметрами:

- значением начального угла;
- местоположением центра воображаемого круга, по которому будет располагаться градиент;
- набором ключевых секторов с указанием начального, конечного углов и «чистого» цвета у каждого из них.

Угловой градиент создается функцией `conic-gradient()`:

```
conic-gradient([[from <начальный угол градиента>]
                [at <положение центра>], ]
               <описания ключевых секторов, разделенные запятыми>)
```

Она указывается в качестве значения атрибута стиля `background-image` или `background`.

Параметры углового градиента:

- *`начальный угол градиента`* — указывается в любой из поддерживаемых единиц измерения. Положительные значения отсчитываются по часовой стрелке, отрицательные — против часовой стрелки.

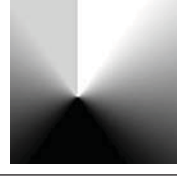
Пример	Результат

`background-image:`
`  conic-gradient(from 45deg at 40% 60%,`
`  white 0 10%, black 40% 60%,`
`  lightgrey 90% 100%);`
	

`background-image:`
`  conic-gradient(from -45deg at 40% 60%,`
`  white 0 10%, black 40% 60%,`
`  lightgrey 90% 100%);`
	

Если *`начальный угол градиента`*, вместе со словом `from`, не указан, он принимается равным `0`.

Пример	Результат

`background-image:`
`  conic-gradient(at 40% 60%,`
`  white 0 10%, black 40% 60%,`
`  lightgrey 90% 100%);`
	

- *положение центра* градиента. Указывается в том же формате, что и значение атрибута стиля `object-position` (см. *разд. 17.3*).

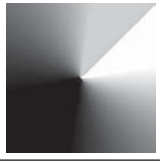
Пример	Результат

`background-image:`
`  conic-gradient(from 45deg at 40% 60%,`
`  white 0 10%, black 40% 60%,`
`  lightgrey 90% 100%);`
	

`background-image:`
`  conic-gradient(from 45deg at left top,`
`  white 0 10%, black 40% 60%,`
`  lightgrey 90% 100%);`
	

Положение центра, вместе со словом `at`, можно не указывать — тогда центр градиента будет находиться в центре элемента (как если бы было задано положение `50% 50%`).

Пример	Результат

`background-image:`
`  conic-gradient(from 45deg,`
`  white 0 10%, black 40% 60%,`
`  lightgrey 90% 100%);`
	

- отдельное *описание ключевого сектора* записывается в формате:
  <«чистый» цвет> [*<начальный угол сектора>* [*<конечный угол сектора>*]]
  Начальный и *конечный углы сектора* могут быть указаны как в любой из поддерживаемых единиц измерения углов, отсчитываемых от заданного ранее *начального угла градиента*, так и в процентах (которые будут рассчитываться от полного оборота).

Пример	Результат

`background-image:`
`  conic-gradient(from 45deg,`
`  white 0 10deg,`
`  black 100deg 260deg,`
`  white 350deg 360deg);`
	

`background-image:`
`  conic-gradient(white 0 2%, black 30% 35%,`
`  white 48% 52%, black 64% 69%,`
`  white 98% 100%);`
	

Если *конечный угол сектора* не задан, он принимается равным *начальному углу сектора*. В результате ключевой сектор примет «вырожденный» вид линии.

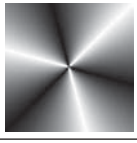
Пример	Результат

`background-image:`
`  conic-gradient(from 45deg,`
`  white 0, black 180deg,`
`  white 360deg);`
	

Также можно не указывать оба *угла сектора*. В таком случае первый ключевой сектор будет находиться в начале градиента, конечный — в конце, а промежуточные будут равномерно распределены между предыдущим и следующим ключевыми секторами с явно указанными углами. Все секторы без явно указанных углов примут «вырожденный» вид.

Пример	Результат

`background-image:`
`  conic-gradient(from 45deg,`
`  white, black, white);`
	

`background-image:`
`  conic-gradient(from 45deg,`
`  white, black, white, black,`
`  white, black, white);`
	

Сайт **https://angrytools.com/gradient/** предоставляет удобные визуальные инструменты для создания линейных, радиальных и угловых градиентов. Результат выводится на экране, а также представляется в виде готового к использованию CSS-кода.

19.2.4. Повторяющиеся градиенты

Если конечная ключевая точка любого градиента не совпадает с его концом, оставшаяся часть градиента будет заполнена тем цветом, что задан в конечной ключевой точке.

Пример	Результат

`background-image:`
`  linear-gradient(to right,`
`  white, black 20%, lightgrey 50%);`
	

Мы можем создать *повторяющийся градиент*, который будет повторяться раз за разом, чтобы покрыть весь элемент. Для этого следует использовать функции:

- `repeating-linear-gradient()` — для создания повторяющегося линейного градиента.

Пример	Результат

`background-image:`
`  repeating-linear-gradient(to right,`
`  white, black 20%,`
`  lightgrey 50%);`
	

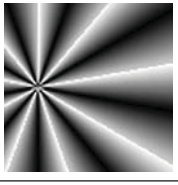
- `repeating-radial-gradient()` — для создания повторяющегося радиального градиента.

Пример	Результат

`background-image:`
`  repeating-radial-gradient(to right,`
`  white, black 20%,`
`  lightgrey 50%);`
	

- `repeating-conic-gradient()` — для создания повторяющегося углового градиента.

Пример	Результат

`background-image:`
`  repeating-conic-gradient(at 20% 50%,`
`  white, black 20deg,`
`  lightgrey 40deg);`
	

19.3. Упражнение. Красивые градиентные фоны

Сделаем на страницах сайта суши-бара красивые градиентные фоны, которые заметно оживят сайт.

- Секцию тела страницы заполним «сетчатым» фоном в виде тонких полосок цвета `blanchedalmond`, распространяющихся наискосок вправо и вверх под углом. Реализуем их в виде повторяющегося линейного градиента. В первой ключевой точке укажем белый цвет, во второй, отстоящей от 1 мм от начала, — `maroon`.

- У блока со стилевым классом `container`, вмещающего все содержание страницы, сделаем белый полупрозрачный сплошной фон (сквозь который будет просвечивать сделанный ранее «сетчатый» фон).
- У первой строки заголовка в семантической шапке сделаем радиальный круговой фон с центром на правой стороне, заканчивающийся на стороне, наиболее отдаленной от центра (т. е. левой). В начальной точке этот градиент будет иметь цвет `blanchedalmond`, во второй, расположенной на двух третях расстояния от начала заголовка, — такой же, а в конечной, находящейся на 90% расстояния от начала, — белый.

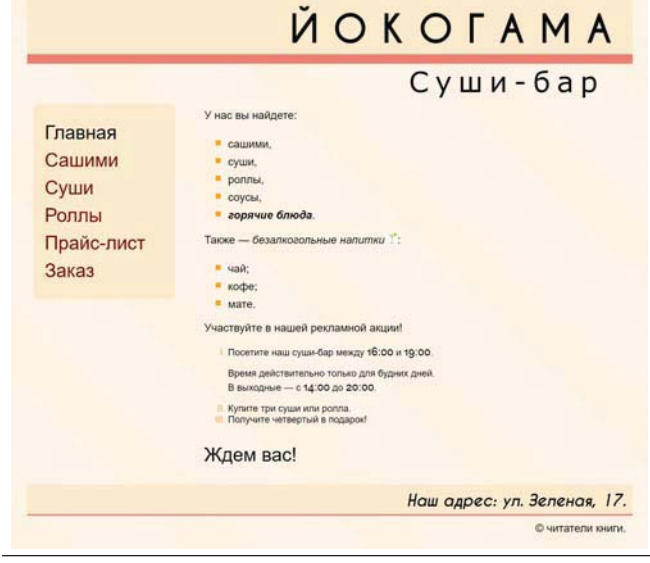
1. Найдем в папке 18\s18.6 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Сначала создадим «полосатый» фон у секции тела страницы (тега `<body>`).

2. Откроем таблицу стилей styles\2.1.css и дополним стиль, применяемый к секции тела, следующим образом:

```
body {
    . . .
    background-image:
        repeating-linear-gradient(45deg, white, blanchedalmond 1mm);
}
```

✓ Что у нас получилось? ⌄

Общий вид главной страницы	Часть фона (увеличено)
	

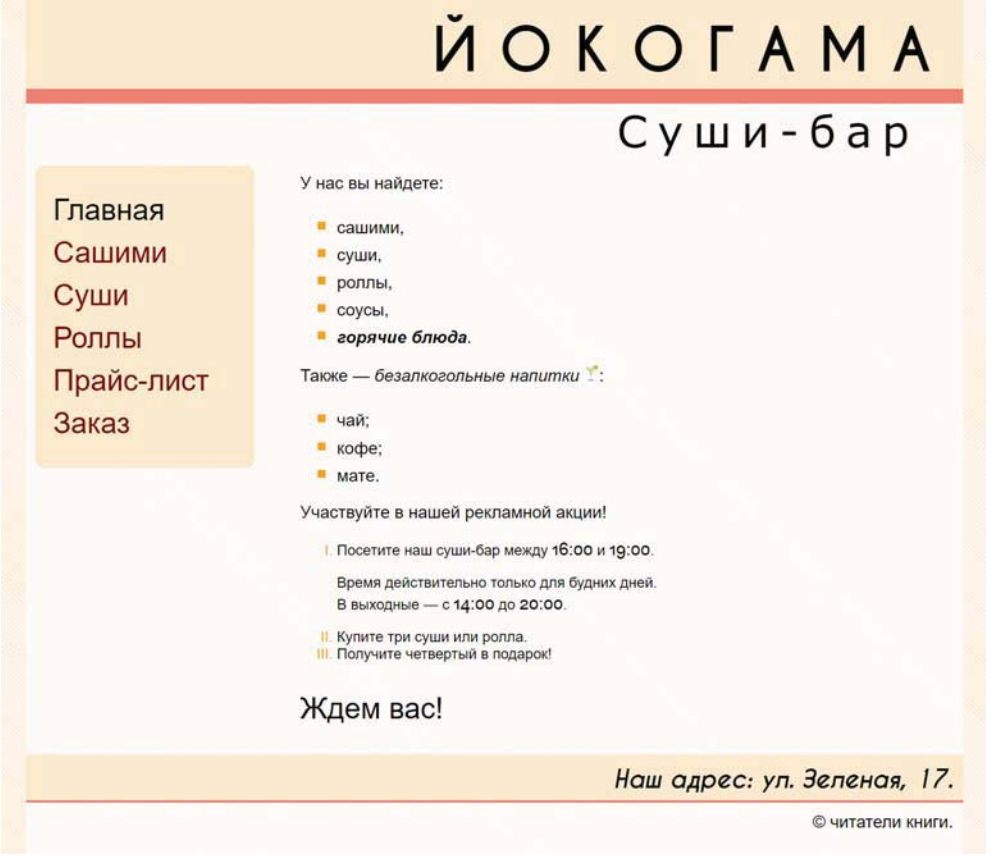
Теперь закрасим блок со стилевым классом `container`, в котором находится содержание страниц, полупрозрачным белым сплошным фоном.

3. Дополним стиль, применяемый к блоку со стилевым классом `container`, так:

```
div.container {
    . . .
    background-color: rgba(255, 255, 255, 0.8);
}
```

Для указания полупрозрачного цвета мы применили функцию `rgba()` (см. *разд. 12.2.2*). Степень непрозрачности фона (`0.8`) была подобрана автором экспериментально.

✓ Получилось? Разумеется! ⌄



И, наконец, займемся самым сложным фоном, которым будет закрашена первая строка заголовка, находящегося в семантической шапке.

4. Исправим стиль, который применяется к заголовку первого уровня со стилевым классом `header1`, находящемуся в семантической шапке, следующим образом:

```
header h1.header1 {
    . . .
    background-color: blanchedalmond;
    background-image:
        radial-gradient(circle farthest-side at right,
                        blanchedalmond, blanchedalmond 67%,
                        white 90%);
    . . .
}
```

✓ Что у нас получилось? ⌄

Показан только заголовок.

19.4. Графический фон

Во многих случаях придать лоск странице или ее отдельному элементу позволяет *графический фон*.

> ***Графический фон***
>
> Графическое изображение, используемое в качестве фона.

Для указания графического фона применяется уже знакомый нам атрибут стиля `background-image`. В качестве его значения можно указать:

- ссылку на файл с нужным графическим изображением — с помощью функции `url()`;
- `none` — будет выведен сплошной фон, заданный в атрибуте стиля `background-color`, или фон по умолчанию.

Значение по умолчанию: `none`.

Атрибут стиля `background-attachment` указывает, где должен выводиться графический фон. Его значения:

- `scroll` — графический фон выводится в самом элементе страницы и прокручивается вместе с содержимым элемента;

- `local` — графический фон выводится в самом элементе страницы и *не* прокручивается вместе с содержимым элемента;
- `fixed` — графический фон выводится в секции тела страницы, а в элементе отображается только часть фона, попадающая в этот элемент. Фон также не прокручивается.

Значение по умолчанию: `scroll`.

Примеры:

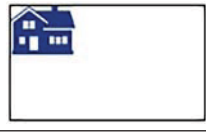
```
body {
    background-image: url(../images/body-bg.jpg);
    background-attachment: fixed;
}
main {
    background-image: url(../images/main-bg.jpg);
    background-attachment: local;
}
```

Атрибут стиля `background-size` устанавливает размер графического фона. Поддерживаемые им значения:

- `auto` или `auto auto` — фон выводится с изначальными размерами.

Пример	Результат
`background-size: auto;`	

- числовая величина в какой-либо единице измерения укажет ширину фона. При этом высота будет подобрана таким образом, чтобы пропорции фона не нарушались.

Пример	Результат
`background-size: 40px;`	

- две числовые величины, разделенные пробелом, — первая укажет ширину фона, вторая — высоту.

Пример	Результат
`background-size: 40px 60px;`	

Вместо любой из величин можно указать слово `auto` — тогда соответствующий размер будет подобран автоматически, чтобы пропорции фона не нарушались.

Пример	Результат
`background-size: auto 60px;`	

- `cover` — фон примет такие размеры, чтобы полностью покрыть элемент, без нарушения своих пропорций. При этом какие-то части фона могут оказаться за границами элемента и, таким образом, не будут видимы.

Пример	Результат
`background-size: cover;`	

- `contain` — фон примет такие размеры, чтобы полностью поместиться в элементе, без нарушения своих пропорций. При этом часть элемента может оказаться не покрытой фоном.

Пример	Результат
`background-size: contain;`	

Значение по умолчанию: `auto auto`.

Если графический фон по размерам меньше элемента, по умолчанию он будет повторяться таким образом, чтобы покрыть весь элемент. Это поведение можно изменить.

Атрибут стиля `background-repeat` управляет режимом повторения графического фона. В нем можно указать:

- одно значение — задаст режим повторения фона и по горизонтали, и по вертикали:
  - `repeat` — фон повторяется по горизонтали и вертикали. При этом какие-то копии графического фона могут оказаться обрезанными.

Пример	Результат
`background-repeat: repeat;`	

- `space` — то же самое, что и `repeat`, только отдельные копии фона выводятся с просветами между ними. В результате этого элемент окажется покрытым целым числом копий фона.

Пример	Результат
`background-repeat: space;`	

- `round` — то же самое, что и `space`, только для достижения того же эффекта будут варьироваться размеры копий фона.

Пример	Результат
`background-repeat: round;`	

- `repeat-x` — фон повторяется только по горизонтали.

Пример	Результат
`background-repeat: repeat-x;`	

- `repeat-y` — фон повторяется только по вертикали.

Пример	Результат
`background-repeat: repeat-y;`	

- `no-repeat` — фон вообще не повторяется (выводится только единожды).

Пример	Результат
`background-repeat: no-repeat;`	

- два приведенных ранее значения через пробел: первое укажет режим повторения по горизонтали, второе — по вертикали. Допускается указывать значения `repeat`, `space`, `round` и `no-repeat`.

Пример	Результат
`background-repeat: space no-repeat;`	

Значение по умолчанию: `repeat`.

Если фон по размерам меньше, чем покрываемый им элемент, и не повторяется, мы можем задать местоположение фона в элементе. Для этого применяются следующие атрибуты стиля:

- `background-position-x` задает местоположение по горизонтали;
- `background-position-y` — местоположение по вертикали.

В качестве значения у этих атрибутов стиля можно задать:

- числовую величину в какой-либо единице измерения — укажет горизонтальную или вертикальную координату левого верхнего угла фона. Координаты отсчитываются от левого верхнего угла элемента: горизонтальная — вправо, вертикальная — вниз;
- одно из предопределенных значений:
  - по горизонтали — `left` (у левого края элемента), `center` (в его центре) или `right` (у правого края);
  - по вертикали — `top` (у верхнего края элемента), `center` (в его центре) или `bottom` (у нижнего края).

Значения по умолчанию у обоих атрибутов стиля: `0%`.

Пример	Результат
`background-position-x: 80%;` `background-position-y: 30px;`	

Пример	Результат
`background-position-x: left;` `background-position-y: bottom;`	
`background-position-y: bottom;`	

Атрибут стиля `background-position` позволяет указать сразу обе координаты фона. Его значение записывается в тех же форматах, которые поддерживаются атрибутом стиля `object-position` (см. *разд. 17.3*). Значение по умолчанию: `0% 0%`.

Пример	Результат
`background-position: 80% 30px;`	
`background-position: left bottom;`	
`background-position: left 25px bottom 15px;`	

Атрибут стиля `background-origin` задает точку элемента, относительно которой позиционируется графический фон. Он поддерживает три значения:

- `padding-box` — фон позиционируется относительно левого верхнего угла пространства, образованного внутренними просветами.

Пример	Результат

`padding: 20px;`
`border: 20px dotted grey;`
`background-size: contain;`
`background-repeat: no-repeat;`
`background-origin: padding-box;`
	

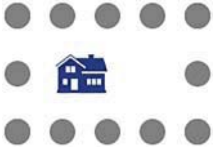
- `border-box` — фон позиционируется относительно левого верхнего угла границы элемента, по которой рисуется рамка.

Пример	Результат

`padding: 20px;`
`border: 20px dotted grey;`
`background-size: contain;`
`background-repeat: no-repeat;`
`background-origin: border-box;`
	

- `content-box` — фон позиционируется относительно левого верхнего угла содержимого элемента.

Пример	Результат

`padding: 20px;`
`border: 20px dotted grey;`
`background-size: contain;`
`background-repeat: no-repeat;`
`background-origin: content-box;`
	

Значение по умолчанию: `padding-box`.

Атрибут стиля `background-clip`, рассмотренный в *разд. 19.1*, поддерживается и применительно к графическому фону. С его помощью можно указать область покрытия.

19.5. Указание сразу всех параметров фона

Атрибут стиля `background` может быть использован для указания сразу всех параметров фона. Отдельные параметры задаются через пробелы в таком порядке:

```
[<режим прокрутки (background-attachment)>]
[<область покрытия (background-clip)>]
[<точка позиционирования (background-origin)>]
<цвет сплошного фона (background-color)>|↳
<градиентный или графический фон (background-image)>
[<местоположение (background-position)>][ / <размер (background-size)>]
[<режим повторения (background-repeat)>]
```

Обязательными к указанию являются лишь один из параметров: *цвет сплошного фона* или *градиентный или графический фон*. Остальные параметры можно не задавать — они получат значения по умолчанию.

Пример	Результат

`width: 100px;`
`height: 70px;`
`padding: 5px;`
`border: 5px double grey;`
`background: local content-box content-box`
`url(../images/bg.png)`
`center / contain no-repeat;`
	

Отдельные величины, находящиеся в составе значения атрибута стиля `background`, могут располагаться в произвольном порядке за двумя исключениями:

- параметры *область покрытия и точка позиционирования* должны находиться строго друг за другом;
- параметр *размер* должен находиться после *местоположения* и отделяться от него слешем.

Так, следующий атрибут стиля создаст фон, аналогичный представленному ранее:

```
background: no-repeat center / contain local content-box content-box
                                                    url(../images/bg.png);
```

19.6. Смешивание содержимого элемента и фона его родителя

По умолчанию содержимое элемента полностью перекрывает фон его родителя.

Пример	Результат

`.parent {`
`color: blue;`
`background-image:`
`linear-gradient(to right,`
`green, yellow, green);`
`}`
`. . .`
`<div class="parent">`
`<div class="child">Смешивание</div>`
`</div>`
	Смешивание

Однако можно предписать веб-обозревателю выполнять *смешивание* содержимого элемента и фона его родителя согласно заданному режиму.

> ***Смешивание***
>
> Вычисление результирующего цвета, которым будет закрашено содержимое элемента, на основе цвета, указанного у содержимого, и цвета фона родителя.

Для указания режима смешивания содержимого элемента с фоном его родителя служит атрибут стиля `mix-blend-mode`. Вот поддерживаемые им значения:

- `normal` — содержимое элемента полностью перекрывает фон его родителя без какого бы то ни было смешивания.

Пример	Результат

`.child {`
`  mix-blend-mode: normal;`
`}`
	

- `multiply` — цвета содержимого и фона родителя перемножаются. В результате темные участки содержимого становятся темнее, светлые не изменяются.

Пример	Результат

`.child {`
`  mix-blend-mode: multiply;`
`}`
	

- `screen` — цвета содержимого и фона родителя инвертируются, перемножаются, полученное произведение также инвертируется. В результате темные участки содержимого не изменяются, светлые становятся светлее.

Пример	Результат

`.child {`
`  mix-blend-mode: screen;`
`}`
	

- `overlay` — если цвет фона темнее цвета содержимого, то аналогично `multiply`, в противном случае — `screen`.

Пример	Результат

`.child {`
`  mix-blend-mode: overlay;`
`}`
	

- `darken` — берется наиболее темный цвет.

Пример	Результат

`.child {`
`  mix-blend-mode: darken;`
`}`
	

- `lighten` — берется наиболее светлый цвет.

Пример	Результат

`.child {`
`  mix-blend-mode: lighten;`
`}`
	

- `color-dodge` — цвет фона делится на инвертированный цвет содержимого. Режим подобен `screen`, но светлые цвета осветляются в меньшей степени.

Пример	Результат

`.child {`
`  mix-blend-mode: color-dodge;`
`}`
	

- `color-burn` — цвет фона инвертируется, делится на цвет содержимого, полученное частное также инвертируется. Режим подобен `mutliply`, но темные цвета затемняются в меньшей степени.

Пример	Результат

`.child {`
`  mix-blend-mode: color-burn;`
`}`
	

- `hard-light` — если цвет содержимого темнее цвета фона, то аналогично `multiply`, в противном случае — `screen`.

Пример	Результат

`.child {`
`  mix-blend-mode: hard-light;`
`}`
	

- `soft-light` — аналогичен `hard-light`, только с более мягкими цветами.

Пример	Результат

`.child {`
`  mix-blend-mode: soft-light;`
`}`
	

- `difference` — из наиболее светлого цвета вычитается наиболее темный.

Пример	Результат

`.child {`
`  mix-blend-mode: difference;`
`}`
	

- `exclusion` — аналогичен `difference`, только с менее контрастными цветами.

Пример	Результат

`.child {`
`  mix-blend-mode: exclusion;`
`}`
	

- `hue` — результирующий цвет составляется из оттенка цвета содержимого, насыщенности и яркости цвета фона.

Пример	Результат

`.child {`
`  mix-blend-mode: hue;`
`}`
	

- `saturation` — результирующий цвет составляется из насыщенности цвета содержимого, оттенка и яркости цвета фона.

Пример	Результат

`.child {`
`  mix-blend-mode: saturation;`
`}`
	

- `color` — результирующий цвет составляется из оттенка и насыщенности цвета содержимого и яркости цвета фона.

Пример	Результат

`.child {`
`  mix-blend-mode: color;`
`}`
	

- `luminosity` — результирующий цвет составляется из яркости цвета содержимого, оттенка и насыщенности цвета фона.

Пример	Результат

`.child {`
`  mix-blend-mode: luminosity;`
`}`
	

Значение по умолчанию: `normal`.

Не забываем, что атрибут стиля `mix-blend-mode` указывается у элемента, содержимое которого следует смешивать с фоном родителя.

19.7. Множественные фоны

CSS позволяет заполнить элемент несколькими фонами, возможно, накладывающимися друг на друга. Для этого достаточно указать параметры этих фонов в атрибуте стиля `background` через запятую:

`background: <фон 1>, <фон 2>, <фон 3>, . . ., <фон N>`

Фон 1 будет выведен самым верхним, *фон 2* — под *фоном 1*, *фон 3* — под *фоном 2* и т. д.

Если в составе накладываемых *фонов* должен присутствовать сплошной фон, его следует указать самым последним. В результате все графические фоны будут выводиться поверх сплошного.

Пример	Результат

`background:`
`  no-repeat url(../images/bg.png) 0% 0% / 40%,`
`  no-repeat url(../images/bg.png) 60% 60% / 30%,`
`  no-repeat url(../images/bg.png) 95% 95% / 20%,`
`  lightgrey;`
	

Для указания параметров накладываемых фонов можно применять и другие атрибуты стиля «семейства» `background`. Параметры отдельных фонов также приводятся через запятую. Цвет сплошного фона, если таковой также необходимо наложить, записывается в атрибуте стиля `background-color`.

Пример	Результат

`background-image: url(../images/bg.png),`
`url(../images/bg.png),`
`url(../images/bg.png);`
`background-repeat: no-repeat, no-repeat, no-repeat;`
`background-position: 0% 0%, 60% 60%, 95% 95%;`
`background-size: 40%, 30%, 20%;`
`background-color: lightgrey;`
	

Если значение какого-либо параметра одинаково у всех накладываемых фонов (например, значение атрибута стиля `background-repeat` в приведенном примере), его можно указать лишь единожды. Это позволит существенно сократить код.

Пример	Результат

`background-image: . . .;`
~~`background-repeat: no-repeat, no-repeat, no-repeat;`~~
`background-repeat: no-repeat;`
`. . .`
	

Если два фона или более, неважно, графических, градиентных или сплошных, накладываются друг на друга, у них можно задать смешивание.

Смешиванием фонов управляет ненаследуемый атрибут стиля `background-blend-mode`, аналогичный атрибуту стиля `mix-blend-mode` (см. *разд. 19.6*).

Пример	Результат

`background:`
`no-repeat url(../images/bg.png) center / 80%,`
`no-repeat url(../images/bg2.png) bottom / contain;`
	

`background:`
`no-repeat url(../images/bg.png) center / 80%,`
`no-repeat url(../images/bg2.png) bottom / contain;`
`background-blend-mode: darken;`
	

19.8. Упражнение. Использование множественных фонов для создания коллажей

В составе сайта суши-бара присутствует «бесхозный» файл images\sushi.jpg — с картинкой суши, которая выводилась в старом заголовке сайта. Давайте выведем эту картинку в новом заголовке, поместив ее слева, на белом поле градиентного фона. А правее поместим другую картинку — с роллом, тем самым создав нечто подобное коллажу.

Коллаж этот мы сделаем на основе множественного фона, который укажем у первой строки заголовка (с текстом «Йокогама»).

1. Найдем в папке 19\ex19.3 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

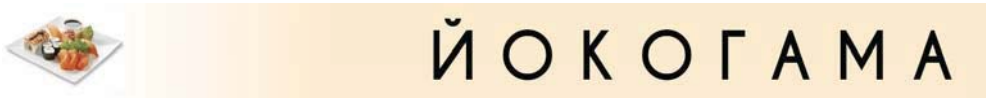
   При выполнении *упражнения 19.3* мы указали у первой строки заголовка градиентный фон. И сейчас превратим его в множественный, поместив в его начале изображение суши из файла images\sushi.jpg, чтобы оно выводилось поверх градиента.

   Картинку с суши разместим у левого края заголовка и отцентрируем по вертикали. Высоту изображения сделаем равной 80% от высоты элемента, чтобы оно не слишком бросалось в глаза. В качестве областей покрытия и позиционирования укажем область пространства, образованную внутренними просветами (эти параметры автор подобрал экспериментально). И отключим повторение изображения.
2. Откроем таблицу стилей styles\2.1.css в текстовом редакторе, найдем стиль, который применяется к заголовку первого уровня со стилевым классом `header1`, находящемуся в семантической шапке, и исправим в нем фрагмент, задающий фон:

```
header h1.header1 {
    . . .
    background-image:
    background:
        padding-box padding-box url(../images/sushi.jpg)
                                left / auto 80% no-repeat,
        radial-gradient(circle farthest-side at right,
                        blanchedalmond, blanchedalmond 67%,
                        white 90%);
    . . .
}
```

✓ Что у нас получилось? ⌄

Показана только первая строка заголовка.



Теперь поместим правее изображение ролла.

3. Найдем в папке 19\!sources сопровождающего книгу файлового архива (см. *приложение 4*) файл rolls.png с изображением роллов. Скопируем его в папку images.

   Картинку с роллами вставим в состав созданного ранее множественного фона, сразу после картинки с суши (чтобы роллы также выводились поверх градиента). Укажем у роллов те же параметры, что и у суши. За исключением горизонтального местоположения, которое зададим равным 160 пикселов (это значение подобрано автором экспериментально).

4. Исправим стиль, применяемый к первой строке заголовка сайта, так:

```
header h1.header1 {
    . . .
    background:
        padding-box padding-box url(../images/sushi.jpg)
                                left / auto 80% no-repeat,
        padding-box padding-box url(../images/rolls.png)
                                160px / auto 80% no-repeat,
        radial-gradient(circle farthest-side at right,
                        blanchedalmond, blanchedalmond 67%,
                        white 90%);
    . . .
}
```

✓ Результат ⌄

19.9. Самостоятельное упражнение

- Разместите под панелью навигации изображение чашки мате. Файл с этим изображением называется mate.png и находится в папке 19\!sources сопровождающего книгу файлового архива (см. *приложение 4*).

ПОДСКАЗКА

Превратите сплошной фон, ранее указанный у блока со стилевым классом `container`, во множественный и поместите изображение мате в его начале, чтобы оно выводилось поверх сплошного фона. Местоположение и размеры картинки с мате подберите самостоятельно.

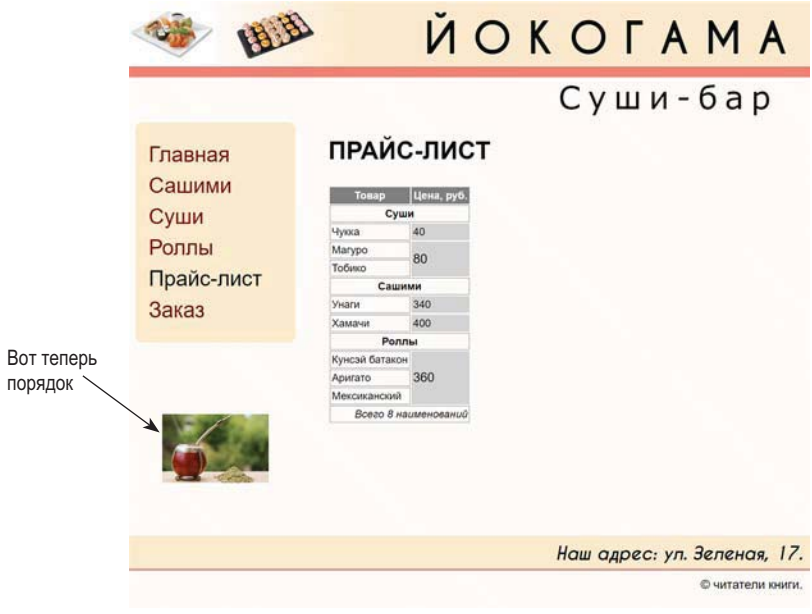
✓ У вас должно получиться ⌄

Проблема: на странице с прайс-листом первая строка поддона (с адресом суши-бара) налезает на изображение мате.

Это происходит оттого, что содержание этой страницы имеет слишком маленькую высоту. Устранить проблему можно, увеличив ее.

- Устраните проблему, задав у семантического основного содержания достаточно большую минимальную высоту (в атрибуте стиля `min-height`), подобрав ее экспериментально.

✓ У вас должно получиться ⌄



Урок 20. Колонки

Верстка текста в несколько колонок.
Просвет между колонками и разделительная линия.

CSS предоставляет инструменты для организации *многоколоночной верстки*.

> ***Многоколоночная верстка***
>
> Вывод текста в несколько колонок.

ВНИМАНИЕ!

Все атрибуты стиля, рассматриваемые на этом уроке, применимы только к блочным элементам и являются ненаследуемыми.

20.1. Основные параметры колонок

Сверстать текст в несколько колонок можно двумя способами:

- указать желаемую ширину отдельной колонки — в атрибуте стиля `column-width` в виде:
  - числовой величины в какой-либо единице измерения — при этом количество колонок подберет сам веб-обозреватель.

Пример	Результат

`.col1 {`
`  column-width: 150px;`
`}`
	Это фан-сайт великой рок-группы.
	
	The Beatles возникли в далеком 1959 году. Поначалу они подражали популярным в те времена американским рок-н-ролльным командам 50-х годов, но позднее пришли к собственному стилю и звучанию, которые и принесли им всемирную популярность.
	
	«Ливерпульская четверка» — именно так их называли и продолжают называть поныне — стали первой британской рок-группой, добившейся мировой популярности. И первой рок-группой, породившей глобальное явление — битломанию.

- `auto` — параметры многоколоночной верстки будут взяты из атрибута стиля `column-count` (см. далее), или, если он не указан, текст будет сверстан в одну колонку.

Значение по умолчанию: `auto`;

- ♦ указать желаемое количество колонок — в атрибуте стиля `column-count` в виде:
  - целого числа без единицы измерения — при этом ширина колонок будет подобрана веб-обозревателем.

Пример	Результат

`.col2 {`
`  column-count: 3;`
`}`
	Это фан-сайт великой рок-группы.
	
	The Beatles возникли в далеком 1959 году. Поначалу они подражали популярным в те времена американским рок-н-ролльным командам 50-х годов, но позднее пришли к собственному стилю и звучанию, которые и принесли им всемирную популярность.
	
	«Ливерпульская четверка» — именно так их называли и продолжают называть поныне — стали первой британской рок-группой, добившейся мировой популярности. И первой рок-группой, породившей глобальное явление — битломанию.

  - `auto` — параметры многоколоночной верстки будут взяты из атрибута стиля `column-width`, или, если он не указан, текст будет сверстан в одну колонку.

Значение по умолчанию: `auto`.

20.2. Параметры просвета между колонками и разделительной линией

Для указания этих параметров применяются следующие атрибуты стиля:

- ♦ `column-gap` — величина просвета между колонками в виде:
  - числовой величины в какой-либо единице измерения.

Пример	Результат

`.col3 {`
`  column-count: 2;`
`  column-gap: 60px;`
`}`
	Это фан-сайт великой рок-группы.
	
	The Beatles возникли в далеком 1959 году. Поначалу они подражали популярным в те времена американским рок-н-ролльным командам 50-х годов, но позднее пришли к собственному стилю и звучанию, которые и принесли им всемирную популярность.
	
	«Ливерпульская четверка» — именно так их называли и продолжают называть поныне — стали первой британской рок-группой, добившейся мировой популярности. И первой рок-группой, породившей глобальное явление — битломанию.

- `normal` — просвет устанавливается веб-обозревателем (обычно равен `1em`).

Значение по умолчанию: `normal`;

- `column-rule-style` — стиль разделительной линии между колонками. Аналогичен атрибутам стиля, задающим стиль рамки вокруг элемента (см. *разд. 15.1*);
- `column-rule-width` — толщина разделительной линии. Аналогичен атрибутам стиля, задающим толщину рамки вокруг элемента;
- `column-rule-color` — цвет разделительной линии. Аналогичен атрибутам стиля, задающим цвет рамки вокруг элемента.

Пример	Результат

`.col4 {`
`  column-width: 150px;`
`  column-rule-width: thick;`
`  column-rule-style: double;`
`  column-rule-color: grey;`
`}`
	Это фан-сайт великой рок-группы.
	
	The Beatles возникли в далеком 1959 году. Поначалу они подражали популярным в те времена американским рок-н-ролльным командам 50-х годов, но позднее пришли к собственному стилю и звучанию, которые и принесли им всемирную популярность.
	
	«Ливерпульская четверка» — именно так их называли и продолжают называть поныне — стали первой британской рок-группой, добившейся мировой популярности. И первой рок-группой, породившей глобальное явление — битломанию.

- `column-rule` — сразу все параметры разделительной линии. Аналогичен атрибутам стиля, задающим параметры рамки вокруг элемента (см. *разд. 15.1*).

Пример	Результат

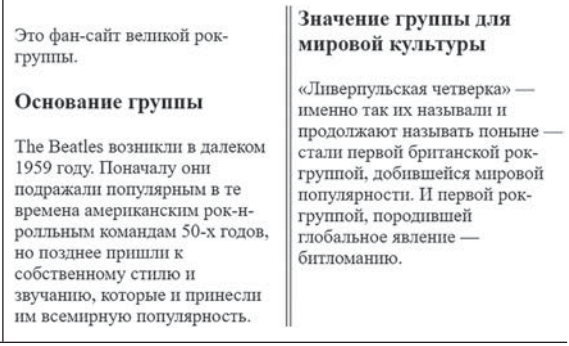
`.col5 {`
`  column-width: 150px;`
`  column-rule: thick double grey;`
`}`
	Это фан-сайт великой рок-группы.
	
	The Beatles возникли в далеком 1959 году. Поначалу они подражали популярным в те времена американским рок-н-ролльным командам 50-х годов, но позднее пришли к собственному стилю и звучанию, которые и принесли им всемирную популярность.
	
	«Ливерпульская четверка» — именно так их называли и продолжают называть поныне — стали первой британской рок-группой, добившейся мировой популярности. И первой рок-группой, породившей глобальное явление — битломанию.

20.3. Дополнительные параметры колонок

Атрибут стиля `column-span` позволяет указать, должен ли какой-либо элемент страницы выводиться в одной колонке или занять их все. Поддерживаемые им значения:

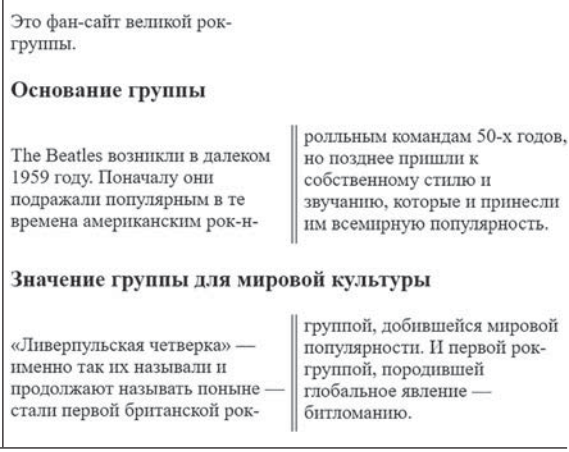
- `none` — элемент должен выводиться в одной колонке.

Пример	Результат

`.col6 {`
`  column-width: 150px;`
`  column-rule: thick double grey;`
`}`
`.col6 h3 {`
`  column-span: none;`
`}`
	

- `all` — элемент должен занимать все колонки по ширине.

Пример	Результат

`.col7 {`
`  column-width: 150px;`
`  column-rule: thick double grey;`
`}`
`.col7 h3 {`
`  column-span: all;`
`}`
	

Значение по умолчанию: `none`.

Атрибут стиля `column-fill` включает или отключает балансировку колонок.

> ***Балансировка колонок***
>
> Выравнивание колонок по высоте.

Поддерживаемые значения:

- `auto` — балансировка колонок *не* выполняется, в результате чего последняя колонка может иметь значительно меньшую высоту, чем остальные.

  Если балансировка колонок не выполняется, у элемента следует явно задать высоту, иначе текст, содержащийся в нем, будет сверстан в одну колонку.

Пример	Результат

`.col8 {`
`  height: 250pt;`
`  column-width: 150px;`
`  column-fill: auto;`
`}`
	Это фан-сайт великой рок-группы.
	
	The Beatles возникли в далеком 1959 году. Поначалу они подражали популярным в те времена американским рок-н-ролльным командам 50-х годов, но позднее пришли к собственному стилю и звучанию, которые и принесли им всемирную популярность.
	
	«Ливерпульская четверка» — именно так их называли и продолжают называть поныне — стали первой британской рок-группой, добившейся мировой популярности. И первой рок-группой, породившей глобальное явление — битломанию.

- `balance` — балансировка колонок выполняется.

Пример	Результат

`.col9 {`
`  column-width: 150px;`
`  column-fill: balance;`
`}`
	Это фан-сайт великой рок-группы.
	
	The Beatles возникли в далеком 1959 году. Поначалу они подражали популярным в те времена американским рок-н-ролльным командам 50-х годов, но позднее пришли к собственному стилю и звучанию, которые и принесли им всемирную популярность.
	
	«Ливерпульская четверка» — именно так их называли и продолжают называть поныне — стали первой британской рок-группой, добившейся мировой популярности. И первой рок-группой, породившей глобальное явление — битломанию.

Значение по умолчанию: `balance`.

Урок 21. Представление и видимость элементов

Представление элементов.
Превращение встроенных элементов в блочные.
«Легкая» панель навигации.
Видимость элементов.

Атрибуты стиля, описанные здесь, могут быть указаны у элемента любой разновидности.

21.1. Представление элемента

Представление определяет разновидность элемента (будет он блочным, встроенно-блочным или встроенным) и некоторые его особенности.

Для указания представления элемента применяется ненаследуемый атрибут стиля `display`, который поддерживает довольно много значений:

- `block` — блочный элемент;
- `inline` — встроенный.

Пример	Результат

`p {`
`  display: inline;`
`}`
`strong {`
`  display: block;`
`}`
	Абзац 1Абзац 2Абзац 3
	Важный текст 1
	Важный текст 2
	Важный текст 3

- `inline-block` — встроенно-блочный;
- `list-item` — пункт списка. По умолчанию выводится с маркером в виде черного кружка. Задать другой маркер или сменить его на нумерацию можно средствами, описанными в *разд. 16.1.1*. Кроме того, необходимо указать достаточно большой внешний просвет слева — для вывода маркера или нумерации.

Пример (HTML)	Пример (CSS)	Результат

`<div>`
`  <p>Суши;</p>`
`  <p>сашими;</p>`
`  <p>роллы.</p>`
`</div>`
	`div p {`
	`  display: list-item;`
	`  list-style-type: decimal;`
	`  margin-left: 20pt;`
	`}`
		1. Суши;
		2. сашими;
		3. роллы.

- `table` — таблица;
- `table-row` — строка таблицы;
- `table-cell` — ячейка таблицы;
- `table-header-group` — секция шапки (тег `<thead>`, см. *разд. 8.5*);
- `table-row-group` — секция основного содержимого (тег `<tbody>`);
- `table-footer-group` — секция поддона (тег `<tfoot>`);
- `table-caption` — заголовок таблицы (тег `<caption>`, см. *разд. 8.6*);
- `table-column-group` — группа колонок (тег `<colgroup>`, см. *разд. 8.7.1*);
- `table-column` — колонка (тег `<col>`, см. *разд. 8.7.2*).

  Указав приведенные значения у атрибута стиля `display`, можно превратить в таблицу, строку, ячейку таблицы и др. любой элемент страницы;

- `inline-table` — таблица, ведущая себя как встроенный элемент.

  Пример:

```
.inlinetable { display: inline-table; }
.inlinetable td { border: thin dotted grey; }
. . .
У нас вы найдете:
<table>
    <tr>
        <td>суши</td><td>сашими</td><td>роллы</td>
    </tr>
</table>
&mdash; и напитки.
```

Результат:

- `none` — элемент вообще не будет выводиться на экран, как будто он и не записан в HTML-коде страницы.

Пример (HTML)	Пример (CSS)	Результат

`<p>Видимый абзац</p>`
`<p class="hidden">Скрытый абзац</p>`
`<p>Видимый абзац</p>`
	`.hidden {`
	`  display: none;`
	`}`
		Видимый абзац
		Видимый абзац

Значение по умолчанию зависит от разновидности элемента: `block` — у блочного элемента, `inline` — у встроенного, `table` — у таблицы и т. д.

21.2. Упражнение. Упрощение панели навигации

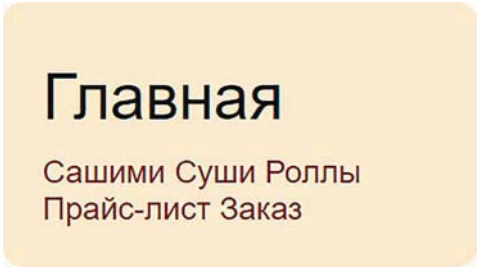
На страницах сайта суши-бара присутствует семантическая панель навигации, созданная тегом `<nav>`, в который вложены блоки (теги `<div>`) с гиперссылками (теги `<a>`). Упростим ее код, удалив все блоки и превратив гиперссылки в блочные элементы путем смены их представления.

1. Найдем в папке 19\s19.9 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.
2. Откроем главную страницу index.html в текстовом редакторе, найдем фрагмент HTML-кода, создающий панель навигации, и удалим все теги `<div>`, в которых содержатся гиперссылки, оставив сами гиперссылки:

```
<nav>
    <div>Главная</div>
    <div><a href="pages/sashimi.html">Сашими</a></div>
    <a href="pages/sashimi.html">Сашими</a>
    <div><a href="pages/sushi.html">Суши</a></div>
    <a href="pages/sushi.html">Суши</a>
    <div><a href="pages/rolls.html">Роллы</a></div>
    <a href="pages/rolls.html">Роллы</a>
    <div><a href="pages/price-list.html">Прайс-лист</a></div>
    <a href="pages/price-list.html">Прайс-лист</a>
    <div><a href="pages/order.html">Заказ</a></div>
    <a href="pages/order.html">Заказ</a>
</nav>
```

Отметим еще раз, что удалить надо лишь те блоки, в которых находятся гиперссылки. Первый блок, с текстом «Главная», удалять не нужно.

✓ Что у нас получилось? ›



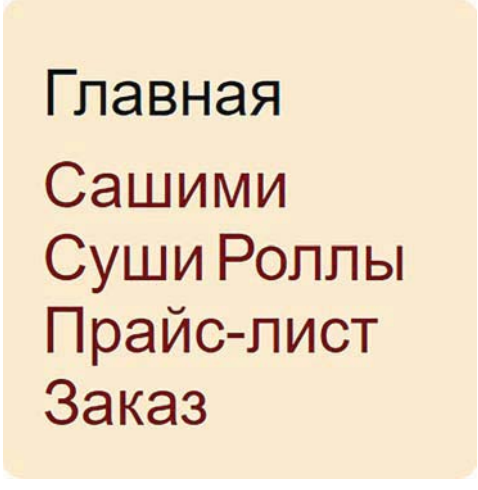
Гиперссылки, будучи от природы встроенными элементами, и выводятся как встроенные элементы — в одну строку. К тому же на них перестал действовать стиль, задающий параметры шрифта.

Поэтому необходимо исправить стиль, который действует на блоки, находящиеся в панели навигации, так, чтобы он действовал и на располагающиеся там же гиперссылки.

3. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и исправим этот стиль:

```
~~nav div {~~
nav div, nav a {
    font-size: 24pt;
    margin: 10px 0px;
}
```

✓ Что у нас получилось? ›



Гиперссылки все еще ведут себя как встроенные элементы. Превратим их в блочные.

4. Исправим стиль гиперссылок, находящихся в панели навигации:

```
nav a {
    color: maroon;
    text-decoration: none;
    display: block;
}
```

После этого панель навигации станет выглядеть так же, как и до выполнения этого упражнения. Однако сейчас она включает в себя меньше элементов и, таким образом, стала немного «легче».

В процессе обработки HTML-кода страницы веб-обозреватель для представления каждого из элементов, имеющихся в ее составе, создает в оперативной памяти особую структуру данных, называемую *объектом*. Каждый объект отнимает определенный объем памяти. Поэтому, сокращая количество элементов на странице, мы экономим оперативную память.

21.4. Видимость элемента

Еще можно сделать какой-либо элемент страницы скрытым, оставив на странице «пустое место».

Видимостью элемента управляет наследуемый атрибут стиля `visibility`. Вот поддерживаемые им значения:

- `visible` — элемент выводится на экран;
- `hidden` — элемент скрыт, однако на странице в том месте, где он должен находиться, присутствует пустое пространство.

Пример 1:

```
.invisible { visibility: hidden; }
. . .
<p>Видимый абзац</p>
<p class="invisible">Невидимый абзац</p>
<p>Видимый абзац</p>
```

✓ Результат 1 ⌄

Видимый абзац

Видимый абзац

Пример 2:

```
.invisible { visibility: hidden; }
. . .
<table>
    <tr><th>Товар</th><th>Цена, руб.</th></tr>
    <tr class="invisible"><td>Чукка</td><td>40</td></tr>
    <tr><td>Магуро</td><td>80</td></tr>
</table>
```

✓ Результат 2 ›

Товар	Цена, руб.
Магуро	80

- `collapse` — в случае применения к строке, группе колонок или колонке таблицы убирает пустое пространство в том месте, где должен выводиться элемент. В случае применения к элементам остальных типов ведет себя аналогично `hidden`.

  Пример:

```
.invisible { visibility: collapse; }
. . .
<table>
    <tr><th>Товар</th><th>Цена, руб.</th></tr>
    <tr class="invisible"><td>Чукка</td><td>40</td></tr>
    <tr><td>Магуро</td><td>80</td></tr>
</table>
```

✓ Результат

Товар	Цена, руб.
Магуро	80

Значение по умолчанию: `visible`.

Урок 22. Местоположение элементов

Плавающие элементы.
Позиционируемые элементы.
Свободно позиционируемые элементы.
Приклеивающиеся элементы.

ВНИМАНИЕ!

Все описываемые здесь атрибуты стиля применимы лишь к блочным элементам и являются ненаследуемыми.

22.1. Плавающие элементы

По умолчанию все блочные элементы располагаются по вертикали, строго друг под другом.

> ***Плавающий элемент страницы***
>
> Блочный элемент, сдвинутый к левой или правой границе родителя, при этом остальное содержимое родителя обтекает его с противоположной стороны.

Чтобы превратить элемент в плавающий, достаточно воспользоваться атрибутом стиля `float`. Поддерживаемые им значения:

- `none` — элемент не является плавающим.

Пример	Результат

`.illustration {`
`  float: none;`
`}`
	
	Йерба мате, или просто мате, — напиток из высушенных и измельченных листьев падуба парагвайского или так называемого падуба тернистого, или кустистого.
	
	Мате оказывает тонизирующее действие на центральную нервную систему за счет содержания кофеина.

- `left` — элемент становится плавающим и сдвигается к левому краю родителя, а остальное содержимое родителя обтекает его справа.

Пример	Результат

`.illustration {`
`  float: left;`
`}`
	
	Йерба мате, или просто мате, — напиток из высушенных и измельченных листьев падуба парагвайского или так называемого падуба тернистого, или кустистого.
	
	Мате оказывает тонизирующее действие на центральную нервную систему за счет содержания кофеина.

- `right` — элемент становится плавающим и сдвигается к правому краю родителя, а остальное содержимое родителя обтекает его слева.

Пример	Результат

`.illustration {`
`  float: right;`
`}`
	
	Йерба мате, или просто мате, — напиток из высушенных и измельченных листьев падуба парагвайского или так называемого падуба тернистого, или кустистого.
	
	Мате оказывает тонизирующее действие на центральную нервную систему за счет содержания кофеина.

Значение по умолчанию: `none`.

В виде плавающих элементов часто реализуют боковые врезки с графическими иллюстрациями, текстовыми пояснениями, дополнениями и т. п. Такие врезки обычно создают с помощью парного тега `<aside>` (см. *разд. 5.2*). Пример:

```
<h4>Мате</h4>
<aside class="illustration">
    <div><img src="teacup.png"></div>
    <p>Мате всегда имеется в наличии в суши-баре
       &laquo;Йокогама&raquo;.</p>
</aside>
<p>Йерба мате, или просто мате, &mdash; напиток из . . .</p>
```

Если содержимое плавающего элемента не имеет строго установленной ширины (например, является текстом), то у него необходимо указать ширину (см. *разд. 15.4*). Если этого не сделать, плавающий элемент растянется почти на всю ширину родителя, и остальное содержимое обтекать его, скорее всего, не будет.

Пример	Результат

`div.parent {`
`  width: 200px;`
`}`
`aside.illustration {`
`  float: left;`
`  border: medium double black;`
`}`
	

`div.parent {`
`  width: 200px;`
`}`
`aside.illustration {`
`  float: left;`
`  border: medium double black;`
`  width: 150px;`
`}`
	

У плавающего элемента можно указать внешние просветы, чтобы отодвинуть от него обтекающее его содержимое:

```
.illustration {
    float: left;
    margin: 8pt 16pt 8pt 0;
}
```

Ненаследуемый атрибут стиля `clear` управляет выводом элемента страницы, у которого он указан, относительно плавающих элементов — будет ли он выводиться под ними или же обтекать их. Вот поддерживаемые значения:

- `none` — элемент обтекает любые плавающие элементы.

Пример	Результат

`.illustration {`
`  float: left;`
`}`
`h3 {`
`  clear: none;`
`}`
	

- `left` — элемент выводится под плавающими элементами, сдвинутыми влево (и при этом обтекает плавающие элементы, сдвинутые вправо).

Пример	Результат

`.illustration {`
`  float: left;`
`}`
`h3 {`
`  clear: left;`
`}`
	

- `right` — элемент выводится под плавающими элементами, сдвинутыми вправо (и при этом обтекает плавающие элементы, сдвинутые влево);
- `both` — элемент выводится под любыми плавающими элементами.

Значение по умолчанию: `none`.

Ранее плавающие элементы часто применялись для создания дизайна страниц, при котором панель навигации находится слева, а основное содержание — справа. Панель навигации превращалась в плавающий элемент, сдвинутый к левому краю страницы, а у основного содержимого указывался достаточно большой внешний просвет слева, чтобы оно полностью находилось правее панели навигации, не обтекая ее. Именно такой дизайн мы сделали у страниц нашего сайта суши-бара.

Переполнение и плавающие элементы

Если какой-либо из плавающих элементов имеет высоту бóльшую, чем родитель, он выйдет за границы родителя. Возникнет переполнение.

Пример	Результат

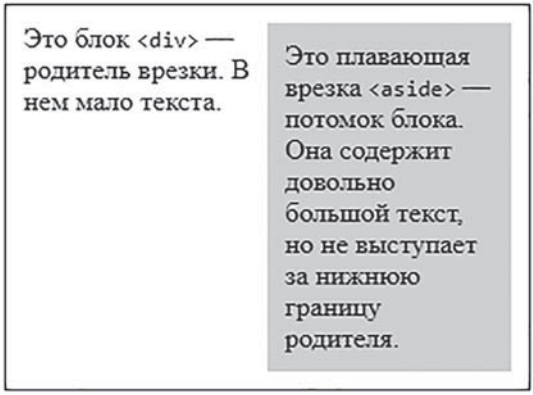
`div {`
`  border: 1px solid black;`
`}`
`aside {`
`  float: right;`
`  background: lightgrey;`
`}`
	Это блок `<div>` — родитель врезки. В нем мало текста.
	Это плавающая врезка `<aside>` — потомок блока. Она содержит довольно большой текст и выступает за нижнюю границу родителя.

Решить эту проблему можно двумя способами:

- указав в стиле, действующем на родитель, атрибут стиля `overflow` (или `overflow-y`) со значением `hidden`;
- указав в том же стиле атрибут стиля `display` со значением `flow-root`.

После этого родитель растянется в высоту, чтобы полностью охватить всех своих плавающих потомков.

Пример	Результат

`div {`
`  border: 1px solid black;`
`  overflow: hidden;`
`}`
`aside {`
`  float: right;`
`  background: lightgrey;`
`}`
	

`div {`
`  border: 1px solid black;`
`  display: flow-root;`
`}`
`aside {`
`  float: right;`
`  background: lightgrey;`
`}`	

22.2. Позиционируемые элементы

По умолчанию местоположением блочных элементов страницы управляет веб-обозреватель, выводя их по вертикали, строго друг под другом. Поэтому такие элементы называются *непозиционируемыми*.

> ***Позиционируемый элемент веб-страницы***
>
> Элемент, чье местоположение может быть задано произвольно, путем указания геометрических координат его сторон.

22.2.1. Создание позиционируемых элементов

CSS позволяет создавать позиционируемые элементы разных видов. Для указания вида позиционируемого элемента применяется атрибут стиля `position`, поддерживающий значения:

- `static` — непозиционируемый элемент;
- `absolute` — *свободно позиционируемый элемент*.

> ***Свободно позиционируемый (свободный) элемент веб-страницы***
>
> Позиционируемый элемент, координаты которого указываются относительно сторон его родителя.

Свободно позиционируемые элементы позиционируются относительно первого позиционируемого родителя или, если такового нет, относительно секции тела страницы.

Свободно позиционируемые элементы прокручиваются вместе с остальным содержимым родителя;

- `relative` — *относительно позиционируемый элемент*.

> ***Относительно позиционируемый (относительный) элемент веб-страницы***
>
> Позиционируемый элемент, координаты которого указываются относительно точки, в которой он находился бы, если бы был непозиционируемым.

Относительно позиционируемые элементы прокручиваются вместе с остальным содержимым родителя;

- `fixed` — *фиксированный элемент*.

> ***Фиксированный элемент веб-страницы***
>
> Позиционируется относительно границ области просмотра. Не прокручивается вместе с остальным содержимым родителя.

- `sticky` — *приклеивающийся элемент*.

> ***Приклеивающийся элемент веб-страницы***
>
> В процессе прокрутки содержимого родителя, выйдя за пределы области с указанными координатами, «приклеивается» к соответствующей стороне этой области.

Значение по умолчанию: `static`.

Для указания координат позиционируемых элементов применяются следующие атрибуты стиля:

- `left` — горизонтальная координата левой стороны элемента;
- `right` — горизонтальная координата правой стороны элемента;
- `top` — вертикальная координата верхней стороны элемента;
- `bottom` — вертикальная координата нижней стороны элемента.

В качестве значения у этих атрибутов стиля можно указать:

- числовую величину координаты в любой из поддерживаемых единиц измерения;
- `auto` — тогда веб-обозреватель сам задаст соответствующую координату.

Значение по умолчанию у всех этих атрибутов стиля: `auto`.

Пример:

```
.positioning {
    position: absolute;
    left: 30px;
    top: 100px;
    right: 65px;
    bottom: 9pt;
}
```

Вместо приведенных атрибутов стиля также можно использовать атрибут стиля `inset`, задающий сразу все координаты. В его значении можно записать четыре, три, две или одну величины координат, разделив их пробелами:

- *<верхняя> <правая> <нижняя> <левая>*;
- *<верхняя> <правая и левая> <нижняя>*;
- *<верхняя и нижняя> <правая и левая>*;
- *<верхняя, правая, нижняя и левая>*.

Пример:

```
.positioning {
    position: absolute;
    inset: 100px 65px 9pt 30px;
}
```

Атрибуты стиля `left`, `right`, `top`, `bottom` и `inset` принимаются во внимание только в том случае, если они указаны у позиционируемого элемента. Будучи указанными у непозиционируемого элемента, они игнорируются.

22.2.1.1. Создание свободно позиционируемых элементов

Для создания свободно позиционируемого элемента необходимо атрибуту стиля `position` дать значение `absolute` и указать координаты и (или) размеры элемента в атрибутах стиля `left`, `right`, `top`, `bottom`, `width` и `height` соответственно.

Координаты свободно позиционируемого элемента указываются относительно сторон его первого позиционируемого родителя: левая координата — относительно левой стороны, нижняя — относительно нижней и т. д. Родитель может быть как свободно, так и относительно позиционируемым. Если позиционируемого родителя нет, координаты указываются относительно сторон секции тела страницы.

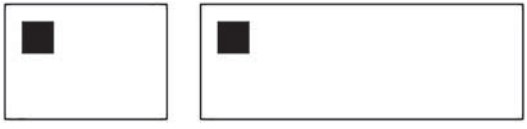
Если в качестве какой-либо координаты указано значение `auto`, эта координата будет рассчитываться веб-обозревателем на основе координаты

противоположной стороны элемента и его размера в соответствующем направлении. Например, если координата левой стороны равна `auto`, она будет рассчитана на основе координаты правой стороны и ширины элемента. Если же и координата противоположной стороны равна `auto`, обе координаты получат такие значения, как если бы элемент был непозиционируемым.

Разные способы позиционирования свободных элементов:

- задание координат двух сторон, образующих угол, и обоих размеров (фиксация угла), — тогда элемент будет находиться в заданном месте при любом изменении размеров родителя.

Пример	Результат

`.el1 {`
`  position:`
`  absolute;`
`  left: 10px;`
`  top: 10px;`
`  width: 20px;`
`  height: 20px;`
`}`
	

- задание координат трех сторон, образующих два угла, и размера перпендикулярной стороны (фиксация двух углов), — тогда элемент будет растягиваться по линии, расположенной между двумя этими углами, и станет *адаптивным*.

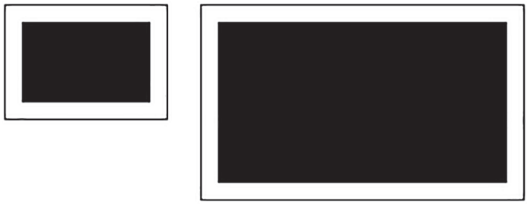
> ***Адаптивный элемент веб-страницы***
>
> Элемент, адаптирующийся к изменениям размеров родителя.

Пример	Результат

`.el2 {`
`  position:`
`  absolute;`
`  left: 10px;`
`  top: 10px;`
`  right: 10px;`
`  height: 20px;`
`}`
	

- задание координат всех четырех сторон (фиксация четырех углов), — тогда элемент, также ставший адаптивным, станет растягиваться в обоих направлениях.

Пример	Результат

.el3 {
position:
absolute;
left: 10px;
top: 10px;
right: 10px;
bottom: 10px;
}
	

- задание только координат двух сторон, образующих угол, — тогда размеры элемента буду подобраны веб-обозревателем на основе размеров содержимого элемента.

Пример	Результат

.el4 {
position: absolute;
left: 10px;
top: 10px;
}
	

- задание координаты лишь одной стороны — тогда координата прилегающей к ней стороны будет установлена такой, как будто элемент является непозиционируемым, а размеры, опять же, будут рассчитаны на основе размеров содержимого элемента.

Пример	Результат

.el5 {
position: absolute;
left: 10px;
}
	

Если не указана ни одна координата, свободно позиционируемый элемент разместится по координатам [0, 0] — в левом верхнем углу первого позиционируемого родителя.

Как говорилось ранее, если у свободного элемента нет ни одного позиционируемого родителя, он будет позиционироваться относительно секции тела страницы. Но часто возникает необходимость позиционировать такой элемент относительно какого-либо из его родителей. В этом случае родитель

можно сделать относительно позиционируемым, *не указывая у него координаты*, — тогда его местоположение на странице не изменится. Пример:

```
/* Превращаем родитель в относительно позиционируемый */
.parent { position: relative; }
/* И без проблем позиционируем свободный потомок относительно него */
.child {
    position: absolute;
    . . .
}
. . .
<div class="parent">
    <div class="child">
        Я — свободно позиционируемый элемент.
        Я позиционирован относительно родителя.
    </div>
</div>
```

22.2.1.2. Создание относительно позиционируемых элементов

Относительно позиционируемый элемент создается заданием у атрибута стиля `position` значения `relative`. Координаты элемента, записываемые в атрибутах стиля `left`, `right`, `top` и `bottom`, отсчитываются от той точки, в которой находился бы элемент, если бы был обычным, непозиционируемым.

Если в качестве какой-либо координаты указано значение `auto`, веб-обозреватель ориентируется на координату противоположной стороны. Так, если координата левой стороны равна `auto`, будет использована координата правой стороны. Если же и координата противоположной стороны равна `auto`, элемент не будет смещен в соответствующем направлении.

Позиционируются относительные элементы двумя способами:

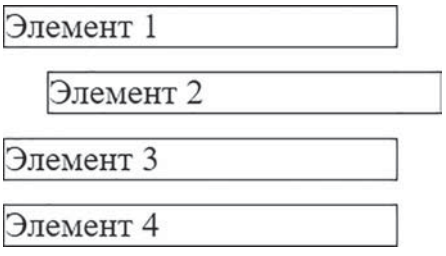
- заданием координат двух сторон, образующих угол, — тогда элемент сместится в двух направлениях.

Пример	Результат

`.element2-1 {`
`  position: relative;`
`  left: 20px;`
`  top: -10px;`
`}`
	Элемент 1
	Элемент 2
	Элемент 3
	Элемент 4

- заданием координаты лишь одной стороны — тогда элемент сместится в одном направлении.

Пример	Результат

`.element2-2 {`
`  position: relative;`
`  left: 20px;`
`}`
	

Если не задать ни одной координаты, относительный элемент останется на месте.

22.2.1.3. Создание фиксированных элементов

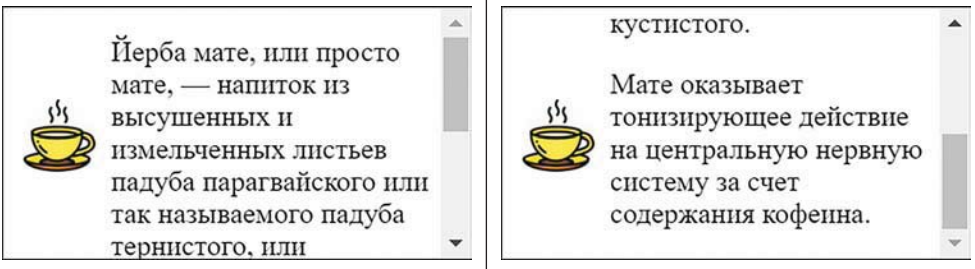
Для создания фиксированного элемента атрибуту стиля `position` следует дать значение `fixed`. В остальном он схож со свободным элементом (см. *разд. 22.2.1.1*), за тем исключением, что *не* прокручивается с остальным содержимым родителя.

Пример:

HTML	CSS

`<div class="parent">`
`  <div class="illustration">`
`  <img src="teacup.png">`
`  </div>`
`  <p>`
`  Йерба мате, или просто мате, &mdash;`
`  напиток из`
`  . . .`
`  </p>`
`  <p>`
`  Мате оказывает тонизирующее`
`  действие на центральную`
`  . . .`
`  </p>`
`</div>`
	`.parent {`
	`  width: 200px;`
	`  height: 140px;`
	`  overflow-x: auto;`
	`  padding-left: 60px;`
	`  border: thin solid black;`
	`}`
	`.illustration {`
	`  position: fixed;`
	`  left: 20px;`
	`  top: 60px;`
	`  width: 40px;`
	`  height: 40px;`
	`}`

Результат:



22.2.1.4. Создание приклеивающихся элементов

Приклеивающийся элемент создается указанием у атрибута стиля `position` значения `sticky`.

Атрибуты стиля `left`, `right`, `top`, `bottom` и `inset` задают координаты сторон области, при выходе из которой в процессе прокрутки приклеивающийся элемент должен, собственно, «приклеиться» к границе этой области. Эти координаты указываются относительно сторон первого прокручиваемого родителя или, если такового не существует, секции тела страницы (о прокручиваемых элементах рассказано в *разд. 15.8*).

Пример:

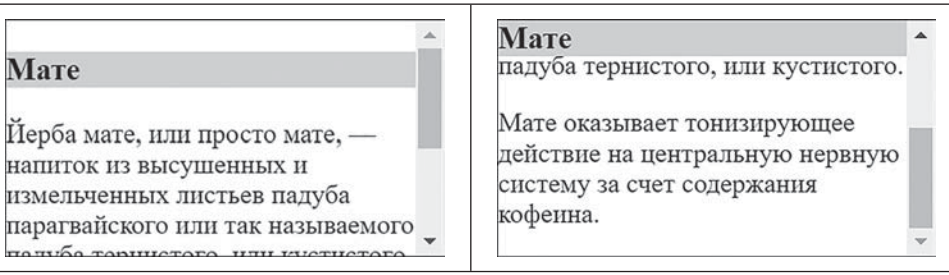
HTML

```
<div class="parent">
    <h3 class="sticky">Мате</h3>
    <p>
        Йерба мате, или просто
        мате, &mdash; напиток из
        . . .
    </p>
    <p>
        Мате оказывает тонизирующее
        действие на центральную
        . . .
    </p>
</div>
```

CSS

```
.cont {
    width: 260px;
    height: 140px;
    overflow-x: auto;
    border: thin solid black;
}
.cont > h3 {
    position: sticky;
    left: 0;
    top: 0;
    right: 0;
    bottom: 0;
    background-color: lightgrey;
}
```

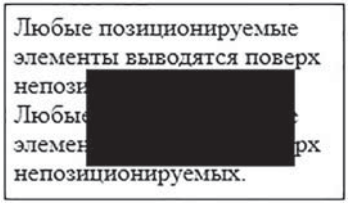
Результат (видно, что при прокрутке содержимого элемента вверх заголовок «приклеился» к верхней границе заданной области):



22.2.2. Наложение позиционируемых элементов

Любые позиционируемые элементы выводятся поверх непозиционируемых, перекрывая их.

Пример	Результат

`.el6 {`
`  position: absolute;`
`  background-color: black;`
`  left: 50px;`
`  top: 40px;`
`  right: 30px;`
`  bottom: 20px;`
`}`
	

Если какие-либо позиционируемые элементы накладываются друг на друга, они по умолчанию перекрывают друг друга в том порядке, в котором они присутствуют в HTML-коде (элемент, записанный в HTML-коде вторым, перекрывает первый элемент, и т. п.).

Пример	Результат

`.el7, .el8, .el9 {`
`  position: absolute;`
`}`
`.el7 {`
`  color: white;`
`  background-color: black;`
`}`
`.el8 {`
`  color: white;`
`  background-color: grey;`
`}`
`.el9 {`
`  color: black;`
`  background-color: lightgrey;`
`}`
	

Атрибут стиля `z-index` задает порядок, в котором позиционируемые элементы будут перекрывать друг друга. В качестве значения можно указать:

- `auto` — элементы, определенные в HTML-коде позже, перекрывают элементы, определенные раньше;
- положительное целое число без единицы измерения — укажет номер в порядке перекрытия. Элементы с бóльшим номером будут перекрывать элементы с меньшим номером. Элементы с указанным номером в порядке перекрытия станут выводиться поверх элементов, у которых порядок не указан.

Пример	Результат
`.el7 { z-index: 2; }` `.el8 { z-index: 1; }`	

- отрицательное целое число без единицы измерения — элементы с указанным номером в порядке перекрытия будут выводиться *под* элементами, у которых порядок не указан.

Пример	Результат
`.el7 { z-index: -10; }` `.el9 { z-index: -5; }`	

Значение по умолчанию: `auto`.

22.3. Упражнение. Доработка видеокиоска

Добавим в видеокиоск, сделанный при выполнении *упражнения 7.4*, шапку и поясняющую подпись, выводящиеся поверх видеоролика. Шапка будет состоять из двух строк («Редвуд» и «Redwood») и располагаться в верхней части страницы, ближе к правому краю, на градиентном фоне, цвета которого меняются от прозрачного зеленого к почти непрозрачному зеленому. Подпись с текстом: «Совокупность из четырех охраняемых территорий,

находящаяся в штате Калифорния, США», — поместим в левом нижнем углу также на почти прозрачном зеленом фоне.

И шапку, и поясняющую надпись оформим в виде свободно позиционируемых элементов — это позволит нам разместить их в нужных местах без проблем.

1. Найдем в папке 17\s17.4 сопровождающего книгу файлового архива (см. *приложение 4*) папку kiosk и скопируем ее куда-либо на локальный диск.

   Сначала напишем HTML-код, создающий шапку. Реализуем ее в виде обычного блока со стилевым классом `heading` (который позволит потом применить к шапке стиль). Две строки, выводящиеся в шапке, оформим в виде заголовков, соответственно, первого и второго уровня.

2. Откроем главную страницу index.html в текстовом редакторе и добавим в секцию тела следующий код:

```
<video src="media/forest.mp4" autoplay loop muted></video>
<div class="heading">
    <h1>Редвуд</h1>
    <h2 lang="en">Redwood</h2>
</div>
```

   Если сейчас открыть главную страницу в веб-обозревателе, шапка будет выведена под видеороликом, за границами области просмотра и, вдобавок, черным текстом на черном же фоне. Так что смотреть там пока не на что.

   Чтобы вывести шапку поверх видеоролика, превратим ее в свободно позиционируемый элемент, укажем координату верхней стороны в 20 пунктов, такую же координату правой стороны и ширину в 400 пунктов (все значения подобраны опытным путем).

3. Откроем таблицу стилей styles\main.css в текстовом редакторе и добавим стиль для шапки:

```
.heading {
    position: absolute;
    top: 20pt;
    right: 20pt;
    width: 400pt;
}
```

✓ Что у нас получилось? ›

Оформим оба заголовка, входящие в состав шапки. Укажем у обоих ширину в 400 пунктов, нулевые внешние просветы, внутренний просвет справа в 10 пунктов, белый цвет текста и выравнивание по правому краю. А еще сделаем у них одинаковый градиент из красивого темно-зеленого цвета `rgba(58, 153, 61)`, распространяющийся в направлении справа налево, имеющий 80% непрозрачности в начале, 50% непрозрачности — на расстоянии 300 пунктов от начала и полную прозрачность — на расстоянии 350 пунктов.

У верхнего заголовка укажем полужирный шрифт `sans-serif` кеглем 64 пункта. А еще зададим внешний просвет снизу, равный 6 пунктов, чтобы отделить заголовки друг от друга пустым пространством.

У нижнего заголовка укажем тот же шрифт, что и у верхнего, только кеглем в 48 пунктов.

Все приведенные числовые значения также подобраны опытным путем.

4. Добавим три стиля, которые зададут необходимое оформление:

```
.heading h1, .heading h2 {
    background-image:
        linear-gradient(to left, rgba(58, 153, 61, 0.8) 0,
                                 rgba(58, 153, 61, 0.5) 300pt,
                                 rgba(58, 153, 61, 0.0) 350pt);
    color: white;
    margin: 0;
    padding-right: 10pt;
    text-align: right;
}
.heading h1 {
    font: bold 64pt sans-serif;
    margin-bottom: 6pt;
}
.heading h2 {
    font: bold 48pt sans-serif;
}
```

✓ А неплохо получилось, не правда ли? ⌄

Увеличим размеры окна веб-обозревателя, чтобы вновь созданная шапка поместилась в окне.

Поясняющую подпись оформим в виде блока со стилевым классом `footing`, а содержащийся в ней текст — в виде обычного абзаца.

5. Переключимся на страницу index.html и добавим в секцию тела код, создающий подпись:

```
<video src="media/forest.mp4" autoplay loop muted></video>
<div class="heading">
    . . .
</div>
<div class="footing">
    <p>Совокупность из четырех охраняемых территорий,
       находящаяся в штате Калифорния, США</p>
</div>
```

У самой подписи укажем абсолютное позиционирование, координату нижней стороны в 20 пунктов, такую же координату левой стороны и ширину в 200 пунктов.

Абзацу с текстом подписи дадим белый цвет текста, полупрозрачный цвет `rgba(58, 153, 61)` у фона, шрифт `sans-serif` кеглем 14 пунктов, нулевые внешние просветы и внутренние просветы в 10 пунктов.

6. Переключимся на таблицу стилей styles\main.css и добавим необходимые стили:

```
.footing {
    position: absolute;
    left: 20pt;
    bottom: 20pt;
    width: 200pt;
}
.footing p {
    color: white;
    background-color: rgba(58, 153, 61, 0.5);
    font: 14pt sans-serif;
    margin: 0;
    padding: 10pt;
}
```

✓ Результат

22.4. Самостоятельное упражнение

Добавьте в видеокиоск еще одну поясняющую подпись — находящуюся в правом нижнем углу и содержащую строчки «Секвойя» и «Sequoia sempervirens».

✓ У вас должно получиться ⌄

Урок 23. Флекс-разметка

Флекс-разметка.
Выравнивание элементов.
Растягивание и сжатие элементов.
Просветы между элементами.
Кирпичная кладка.

> ***Флекс-разметка***
>
> Разметка, выстраивающая элементы-потомки по горизонтали или вертикали, с возможным переносом по строкам или столбцам, с указанным выравниванием или распределением свободного места между ними.

ВНИМАНИЕ!

Все атрибута стиля рассматриваемые здесь, применимы лишь к блочным элементам и являются ненаследуемыми.

23.1. Создание флекс-разметки

Чтобы создать в каком-либо элементе страницы флекс-разметку, достаточно записать в стиле, применяемом к этому элементу, атрибут стиля `display` со значением `flex`:

```
.flex-element { display: flex; }
```

По умолчанию потомки этого элемента будут выстраиваться по горизонтали без переноса по строкам.

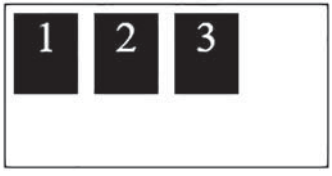
Атрибут стиля `display` также поддерживает значение `inline-flex`, которое, помимо создания флекс-разметки, заставляет элемент вести себя как встроенный. Оно применяется редко и в специфических случаях.

23.2. Направление выстраивания и перенос потомков по строкам или столбцам

Для задания направления выстраивания элементов-потомков в элементе-родителе с флекс-разметкой применяется атрибут стиля `flex-direction`. Его значения:

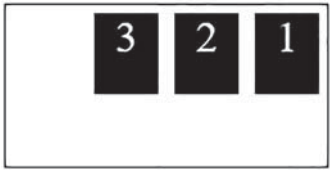
- `row` — потомки выстраиваются по горизонтали, слева направо.

Пример	Результат

`.flex1 {`
`  display: flex;`
`  flex-direction: row;`
`}`
	

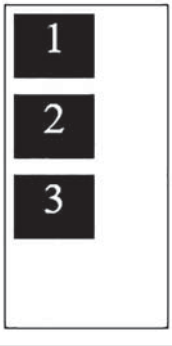
- `row-reverse` — потомки выстраиваются по горизонтали, справа налево.

Пример	Результат

`.flex2 {`
`  display: flex;`
`  flex-direction: row-reverse;`
`}`
	

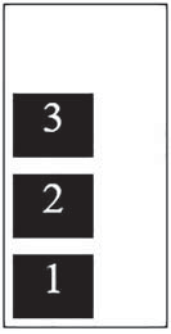
- `column` — потомки выстраиваются по вертикали, сверху вниз.

Пример	Результат

`.flex3 {`
`  display: flex;`
`  flex-direction: column;`
`}`
	

- `column-reverse` — потомки выстраиваются по вертикали, снизу вверх,

Пример	Результат

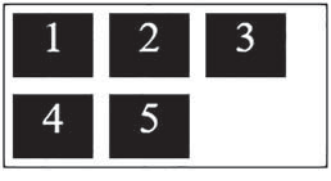
`.flex4 {`
`  display: flex;`
`  flex-direction: column-reverse;`
`}`
	

Значение по умолчанию: `row`.

Атрибут стиля `flex-wrap` управляет переносом элементов-потомков в элементе с флекс-разметкой по строкам или столбцам (в зависимости от направления выстраивания). Вот поддерживаемые им значения:

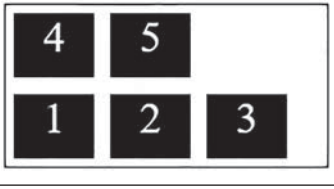
- `nowrap` — перенос не выполняется;
- `wrap` — перенос выполняется, новые строки располагаются ниже, а новые столбцы — правее.

Пример	Результат

`.flex5 {`
`  display: flex;`
`  flex-wrap: wrap;`
`}`
	

- `wrap-reverse` — перенос выполняется, новые строки располагаются выше, а новые столбцы — левее.

Пример	Результат

`.flex6 {`
`  display: flex;`
`  flex-wrap: wrap-reverse;`
`}`
	

Значение по умолчанию: `nowrap`.

Атрибут стиля flex-flow позволяет указать оба параметра одновременно. Его значение задается в формате:

```
[<направление выстраивания (flex-direction)>]
[<управление переносом (flex-wrap)>]
```

Оба параметра указывать необязательно — достаточно задать лишь один из них. Примеры:

```
.flex55 {
    display: flex;
    flex-flow: column wrap;
}
```

```
.flex6 {
    display: flex;
    flex-flow: wrap-reverse;
}
```

23.3. Выравнивание потомков

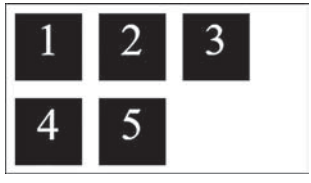
23.3.1. Выравнивание всей совокупности потомков в родителе

Выравнивание всей совокупности элементов-потомков в элементе-родителе с флекс-разметкой при условии, что в родителе есть свободное место, задают два атрибута стиля, указываемые у родителя и рассматриваемые далее.

Атрибут стиля justify-content задает выравнивание в направлении выстраивания потомков: по горизонтали, если задано горизонтальное расположение потомков, и по вертикали, если задано вертикальное расположение. Этот атрибут стиля поддерживает следующие значения:

- normal, start или flex-start[1] — выравнивание по началу направления выстраивания. Например, если потомки выстраиваются по горизонтали слева направо, выполняется выравнивание по левому краю.

Пример	Результат

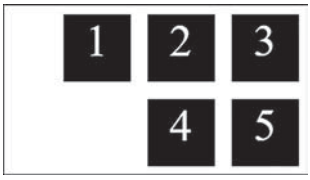
.flex7 {
display: flex;
flex-wrap: wrap;
justify-content: normal;
}
	

.flex7 {
display: flex;
flex-wrap: wrap;
justify-content: start;
}	

[1] Здесь и далее значение flex-start считается устаревшим (хотя и поддерживается до сих пор). Значение normal было введено, вероятно, во имя наглядности CSS-кода.

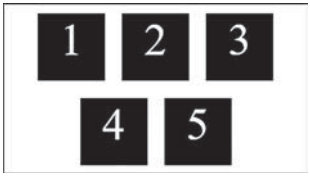
- `end` или `flex-end`[2] — по концу направления выстраивания.

Пример	Результат

`.flex8 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  justify-content: end;`
`}`
	

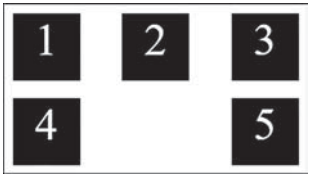
- `center` — по центру.

Пример	Результат

`.flex9 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  justify-content: center;`
`}`
	

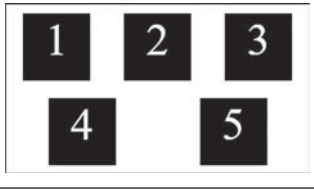
- `space-between` — потомки равномерно распределяются в пространстве родителя, и между ними вставляются просветы.

Пример	Результат

`.flex10 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  justify-content: space-between;`
`}`
	

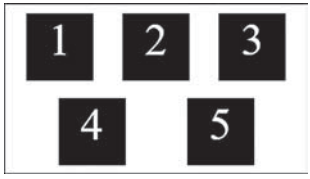
- `space-around` — то же самое, что и `space-between`, только просветы меньшей величины также помещаются между потомками и границами родителя.

Пример	Результат

`.flex11 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  justify-content: space-around;`
`}`
	

[2] Здесь и далее значение `flex-end` считается устаревшим.

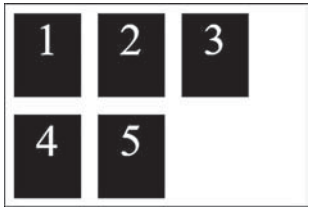
- `space-evenly` — то же самое, что и `space-around`, только все просветы имеют одинаковую величину.

Пример	Результат
`.flex12 { display: flex; flex-wrap: wrap; justify-content: space-evenly; }`	

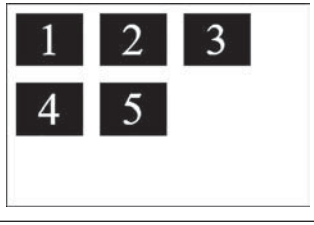
Значение по умолчанию: `normal`.

Атрибут стиля `align-content` задает выравнивание совокупности потомков внутри родителя в направлении, *перпендикулярном направлению выстраивания потомков*: по вертикали, если задано горизонтальное расположение, и по горизонтали, если задано вертикальное расположение. Этот атрибут стиля поддерживает такие значения:

- `normal` или `stretch` — растягивание потомков на все пространство родителя.

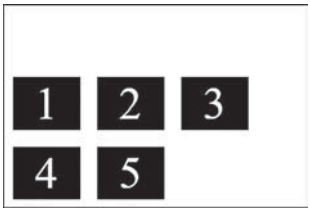
Пример	Результат
`.flex13 { display: flex; flex-wrap: wrap; align-content: normal; }`	
`.flex13 { display: flex; flex-wrap: wrap; align-content: stretch; }`	

- `start` или `flex-start` — выравнивание по началу направления.

Пример	Результат
`.flex14 { display: flex; flex-wrap: wrap; align-content: start; }`	

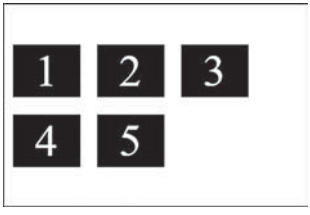
- `end` или `flex-end` — выравнивание по концу направления.

Пример	Результат

`.flex15 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  align-content: end;`
`}`
	

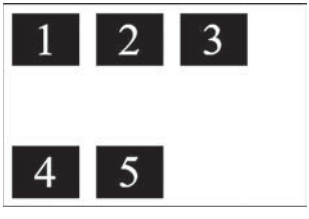
- `center` — выравнивание по центру родителя.

Пример	Результат

`.flex16 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  align-content: center;`
`}`
	

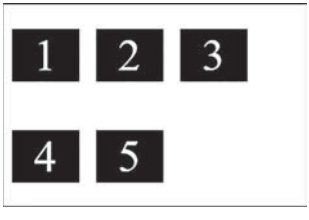
- `space-between` — потомки равномерно распределяются в пространстве родителя и между ними вставляются просветы.

Пример	Результат

`.flex17 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  align-content: space-between;`
`}`
	

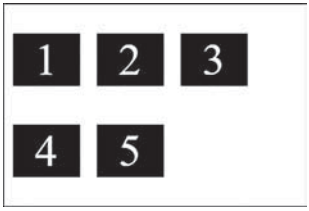
- `space-around` — то же самое, что и `space-between`, но просветы меньшей величины также вставляются между потомками и границами родителя.

Пример	Результат

`.flex18 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  align-content: space-around;`
`}`
	

- `space-evenly` — то же самое, что и `space-around`, только все просветы имеют одинаковую величину.

Пример	Результат

`.flex19 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  align-content: space-evenly;`
`}`
	

Значение по умолчанию: `normal`.

Указать выравнивание сразу в обоих направлениях можно, использовав атрибут стиля `place-content`, значение которого записывается в формате:

```
<выравнивание в направлении, противоположном направлению выстраивания
 (align-content)>
<выравнивание в направлении выстраивания (justify-content)>
```

Оба параметра обязательны к указанию.

Пример:

```
flex66 {
    display: flex;
    flex-wrap: wrap;
    place-content: flex-start center;
}
```

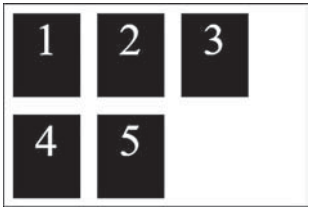
23.3.2. Выравнивание всех потомков внутри совокупности

Можно задать выравнивание всех потомков внутри совокупности, относительно отдельных участков строк или столбцов, в которых они должны выводиться, в направлении, перпендикулярном направлению выстраивания.

Для этого применяется атрибут стиля `align-items`, указываемый у элемента-родителя. Он поддерживает такие значения:

- `normal` или `stretch` — потомки растягиваются на все участки в строке или столбце, внутри которых они должны выводиться.

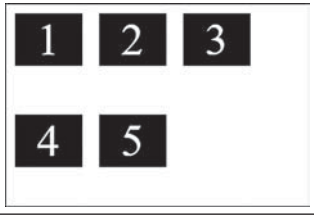
Пример	Результат

`.flex20 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  align-items: normal;`
`}`
	

`.flex20 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  align-items: stretch;`
`}`	

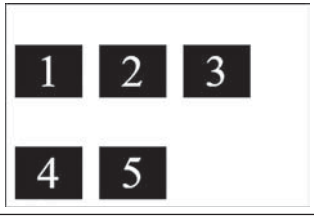
- `start`, `flex-start` или `baseline`[3] — выравнивание по началу участков.

Пример	Результат

`.flex21 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  align-items: start;`
`}`
	

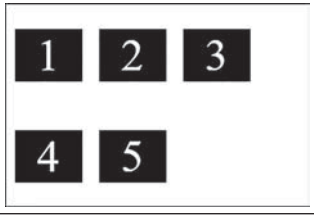
- `end` или `flex-end` — выравнивание по концу участков.

Пример	Результат

`.flex22 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  align-items: end;`
`}`
	

- `center` — выравнивание по центру участков.

Пример	Результат

`.flex23 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  align-items: center;`
`}`
	

Значение по умолчанию: `normal`.

[3] Значение `baseline`, вероятно, введено в качестве задела на будущее.

Полное центрирование элемента страницы

Можно отцентрировать элемент страницы в его родителе и по горизонтали, и по вертикали. Для этого следует:

- убедиться, что центрируемый элемент является единственным потомком родителя;
- указать у родителя флекс-разметку;
- указать у родителя центрирование потомков по горизонтали;
- указать у родителя центрирование потомков по вертикали.

Пример	Результат

`.parent {`
`  display: flex;`
`  justify-content: center;`
`  align-items: center;`
`}`
`. . .`
`<div class="parent">`
`  <div>Я нахожусь в центре`
`  родителя</div>`
`</div>`
	

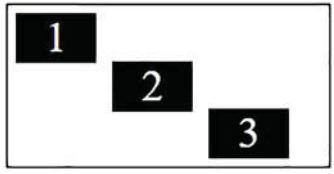
23.3.3. Выравнивание отдельных потомков внутри совокупности

Еще можно указать у отдельного потомка другое выравнивание относительно участка строки или столбца, в котором он должен выводиться, в направлении, перпендикулярном направлению выстраивания.

Для этого служит атрибут стиля `align-self`, *указываемый у потомка, который требуется выровнять иным образом*. Поддерживаемые им значения:

- `auto` — используется значение выравнивания, заданное у родителя атрибутом стиля `align-items` (см. *разд. 23.3.2*);
- одно из значений выравнивания, поддерживаемых атрибутом стиля `align-items`.

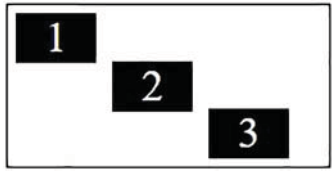
Пример	Результат

`.flex24 {`
`  display: flex;`
`  align-items: center;`
`}`
`.flex24 *:first-child {`
`  align-self: start;`
`}`
`.flex24 *:last-child {`
`  align-self: end;`
`}`
	

Значение по умолчанию: `auto`.

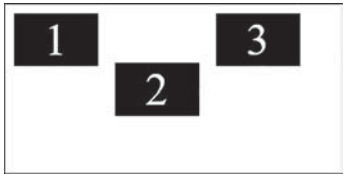
Другой способ выровнять отдельный потомок внутри совокупности — использовать атрибуты стиля «семейства» `margin` (см. *разд. 15.2*). Достаточно указать у внешнего просвета со стороны, противоположной той, к которой следует сдвинуть элемент, значение `auto`.

Пример	Результат

`.flex25 {`
`  display: flex;`
`  align-items: center;`
`}`
`.flex25 *:first-child {`
`  margin-bottom: auto;`
`}`
`.flex25 *:last-child {`
`  margin-top: auto;`
`}`
	

Указав значение `auto` у внешних просветов с противоположных сторон, можно центрировать элемент.

Пример	Результат

`.flex26 {`
`  display: flex;`
`  align-items: center;`
`}`
`.flex26 *:nth-child(2) {`
`  margin-top: auto;`
`  margin-bottom: auto;`
`}`
	

23.4. Управление растягиванием и сжатием потомков

Если в элементе-родителе не хватает места для размещения потомков, а перенос по строкам или столбцам отключен, веб-обозреватель уменьшит размеры потомков (сожмет их), чтобы вместить их в родитель. Есть возможность задать у разных потомков разную степень сжатия (например, сделать так, чтобы один потомок сжимался вдвое сильнее, чем другой).

Если же в родителе есть свободное пространство, можно предписать веб-обозревателю увеличить размеры потомков (расширить их) таким образом, чтобы они заполнили весь родитель. Поддерживается указание у разных потомков разной степени расширения.

Сначала нужно задать у потомков базовый размер: ширину — при выстраивании по горизонтали или высоту — при выстраивании по вертикали. Это производит атрибут стиля `flex-basis`. В качестве его значения можно записать:

- величину размера — в одном из форматов, поддерживаемых атрибутами стиля `width` и `height` (см. *разд. 15.4.1*);
- `content` — веб-обозреватель сам подберет размер, основываясь на размерах содержимого элемента;
- `0` — минимально возможный размер;
- `auto` — будет использован размер, указанный в атрибуте стиля `width` или `height` (в зависимости от направления выстраивания), или, если он не указан, значение `content`.

Значение по умолчанию: `auto`.

Степень сжатия элемента задается атрибутом стиля `flex-shrink`. Доступные к указанию значения:

- число без единицы измерения — задаст степень сжатия. Может быть как целочисленным, так и вещественным. Отрицательные числа не допускаются;
- `0` — элемент вообще не будет сжиматься.

Значение по умолчанию: `1`.

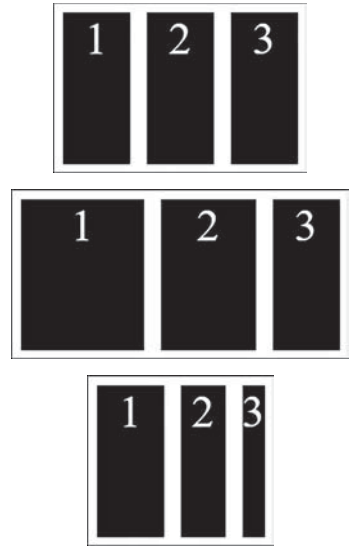
Степень расширения элемента задается атрибутом стиля `flex-grow`. В качестве значения можно указать:

- число без единицы измерения — задаст степень расширения. Может быть как целочисленным, так и вещественным. Отрицательные числа не допускаются;
- `0` — элемент вообще не будет расширяться.

Значение по умолчанию: `0`.

В представленном примере все три элемента-потомка имеют одинаковую базовую ширину. Первый потомок при расширении родителя будет расширяться вдвое сильнее второго потомка, а при сжатии родителя не будет сжиматься. Третий потомок будет сжиматься вдвое сильнее второго и не будет расширяться.

Пример	Результат

`.flex-parent {`
`  display: flex;`
`}`
`.flex-child-1 {`
`  flex-basis: 40px;`
`  flex-grow: 2;`
`  flex-shrink: 0;`
`}`
`.flex-child-2 {`
`  flex-basis: 40px;`
`  flex-grow: 1;`
`}`
`.flex-child-3 {`
`  flex-basis: 40px;`
`  flex-shrink: 2;`
`}`
	

Указать сразу все параметры расширения и сжатия поможет атрибут стиля `flex`. В качестве его значения можно использовать:

- одну числовую величину:
  - без указания единицы измерения — задаст степень расширения (атрибут стиля `flex-grow`). Степень сжатия (`flex-shrink`) будет установлена равной 1, а базовый размер (`flex-basis`) — `0`;
  - с указанием единицы измерения — задаст базовый размер. Степени сжатия и расширения будут установлены равными `1`;
- две числовые величины через пробел — первая укажет степень расширения, а вторая, будучи заданной:
  - без указания единицы измерения — задаст степень сжатия. Базовый размер будет установлен равным `0`;
  - с указанием единицы измерения — задаст базовый размер. Степень сжатия будет установлена равной `1`;
- три числовые величины:
  <степень расширения (`flex-grow`*)> <степень сжатия (*`flex-shrink`*)> <базовый размер (*`flex-basis`*)>*

Пример:

```
.flex-child-1 { flex: 2 0 40px; }
.flex-child-2 { flex: 1 40px; }
.flex-child-3 { flex: 0 2 40px; }
```

23.5. Просветы между потомками

Просветы между потомками родителя с флекс-разметкой можно создать уже знакомыми средствами — атрибутами стиля «семейства» `margin` (см. *разд. 15.2*):

```
.flex-parent { display: flex; }
.flex-child { margin: 4pt; }
```

Однако для этого удобнее использовать следующие атрибуты стиля:

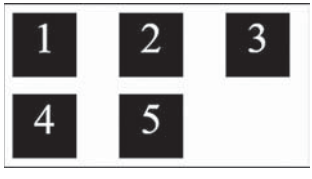
- `column-gap` — горизонтальные просветы;
- `row-gap` — вертикальные просветы.

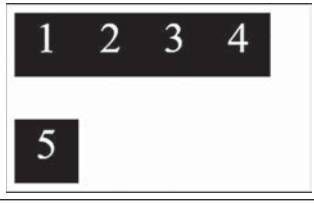
В качестве значения у этих атрибутов стиля можно указать:

- числовую величину просвета в какой-либо из поддерживаемых единиц измерения:
- `normal` — отсутствие просвета (то же самое, что и `0`).

Значение по умолчанию: `normal`.

Пример	Результат

`.flex27 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  column-gap: 20pt;`
`  row-gap: 8pt;`
`}`
	

`.flex28 {`
`  display: flex;`
`  flex-wrap: wrap;`
`  row-gap: 20pt;`
`}`
	

Указать одновременно и горизонтальные, и вертикальные просветы можно в атрибуте стиля `gap`. Его значение может представлять собой:

- одну числовую величину с указанием единицы измерения — задаст одинаковые просветы и по горизонтали, и по вертикали:

  ```
  gap: 4pt;
  ```

- две числовые величины с единицами измерения, разделенные пробелом, — первая задаст просвет по вертикали, вторая — по горизонтали:

```
gap: 8pt 20pt;
```

23.6. Порядок следования элементов

По умолчанию потомки выстраиваются в родителе с флекс-разметкой в том порядке, в котором они записаны в HTML-коде. Однако есть возможность задать другой порядок их следования средствами CSS.

Атрибут стиля `order` указывается у элемента-потомка и задает его номер в порядке следования потомков друг за другом. Этот номер записывается в виде числа без единицы измерения. Допускаются отрицательные числа. Значение по умолчанию: `0`.

Если несколько элементов имеют одинаковые номера в порядке следования, они выстроятся друг за другом в том порядке, в котором они записаны в HTML-коде.

Пример	Результат

`.flex-child-1 { order: 5; }`
`.flex-child-2 { order: 1; }`
`.flex-child-3 { order: 2; }`
`.flex-child-4 { order: 1; }`
`.flex-child-5 { order: -1; }`
	

23.7. Упражнение. Кирпичная кладка, часть 1

> ***Кирпичная кладка***[4]
>
> Набор элементов со сложным содержимым, которые выстроены по горизонтали с переносом по строкам.

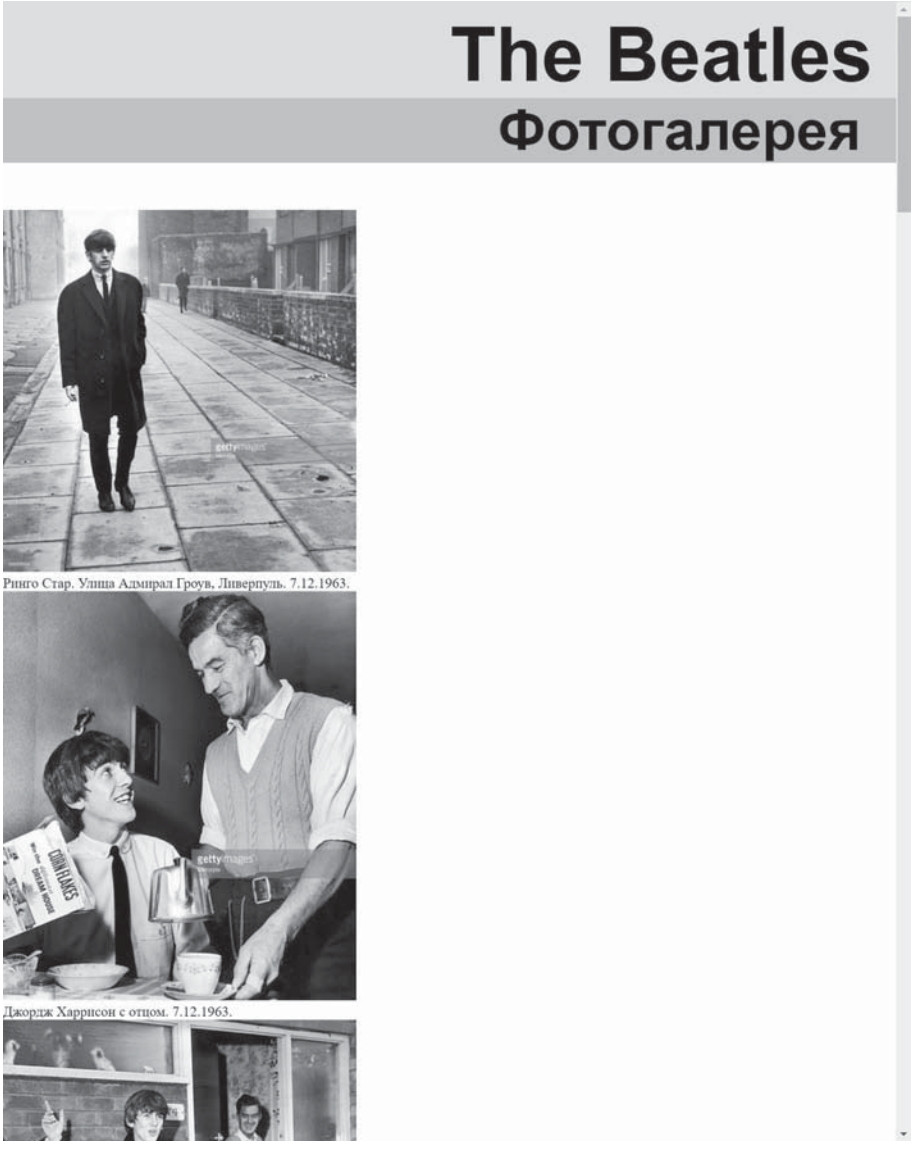
Создадим небольшой сайт из одной страницы, содержащей подборку фотографий группы «The Beatles» и подписей к ним, на основе представленной заготовки. Оформим эту страницу в стиле кирпичной кладки со строками, содержащими по четыре фотографии.

[4] По-английски — masonry. Названа так из-за отдаленного сходства с кирпичной стеной.

1. Найдем в папке 23\!sources сопровождающего книгу файлового архива (см. *приложение 4*) папку the-beatles-photos и скопируем ее куда-либо на локальный диск.

   Эта папка содержит заготовку для создания сайта, включающую главную страницу index.html, папку styles с таблицей стилей main.css, содержащей начальное оформление, и папку images с фотографиями.

   Главная страница сейчас выглядит так:



Шапка и поддон (на рисунке не виден) уже оформлены надлежащим образом. Нам останется лишь оформить элемент, в котором выводятся все фотографии, и элемент-отдельное фото.

2. Откроем главную страницу index.html в текстовом редакторе и найдем HTML-код, создающий элемент с фотографиями и сами фотографии:

```
<main>
    <div class="photo">
        <img src="images/photo1.jpg">
        <div>Ринго Стар. Улица Адмирал Гроув, Ливерпуль.
             7.12.1963.</div>
    </div>
    <div class="photo">
        <img src="images/photo2.jpg">
        <div>Джордж Харрисон с отцом. 7.12.1963.</div>
    </div>
    . . .
</main>
```

Все фотографии подборки находятся в семантическом основном содержимом (в теге <main>). Отдельная фотография оформлена в виде блока (тега) со стилевым классом photo. Внутри такого блока располагается изображение (тег <img>) с фотографией и блок с подписью к ней.

Сначала необходимо указать у семантического основного содержимого флекс-разметку и перенос потомков по строкам. Также следует задать внутренние просветы слева и справа, например, в 20 пунктов, чтобы фотографии не примыкали к границам элемента вплотную.

3. Откроем таблицу стилей styles\main.css в текстовом редакторе и добавим стиль, оформляющий семантическое основное содержимое:

```
main {
    display: flex;
    flex-wrap: wrap;
    padding: 0 20pt;
}
```

Интернет ведет свою родословную с 1969 года

Именно тогда в США была создана компьютерная сеть ARPANET, предшественник Интернета. Она использовалась, по большей части, научными учреждениями.

А коммутация сетевых пакетов, на которой основаны все современные сетевые стандарты, была разработана в 1967 году в Великобритании.

Результат пока не впечатляет…

Показаны только фотографии.

Отдельные элементы-фотографии в родителе располагаются наперекосяк, поскольку имеют разную ширину. Необходимо задать у них одинаковое значение ширины, равное $^1/_4$ от ширины родителя (чтобы в каждой строке помещалось по четыре фотографии).

4. Добавим стиль для элементов со стилевыми классами `photo`, вложенных в семантическое основное содержимое, который задаст нужное оформление:

```
main .photo { width: 25%; }
```

Лучше не стало…

Изображения из отдельных элементов-фотографий налезают друг на друга. Вдобавок изображения из крайних правых элементов вылезли за пределы окна веб-обозревателя.

Исправить это можно, указав у тегов `<img>` ширину, равную ширине родителя, предварительно превратив их в блочные элементы.

5. Добавим необходимый стиль для тегов `<img>`, вложенных в элементы со стилевыми классами `photo`:

```
main .photo img {
    display: block;
    width: 100%;
}
```

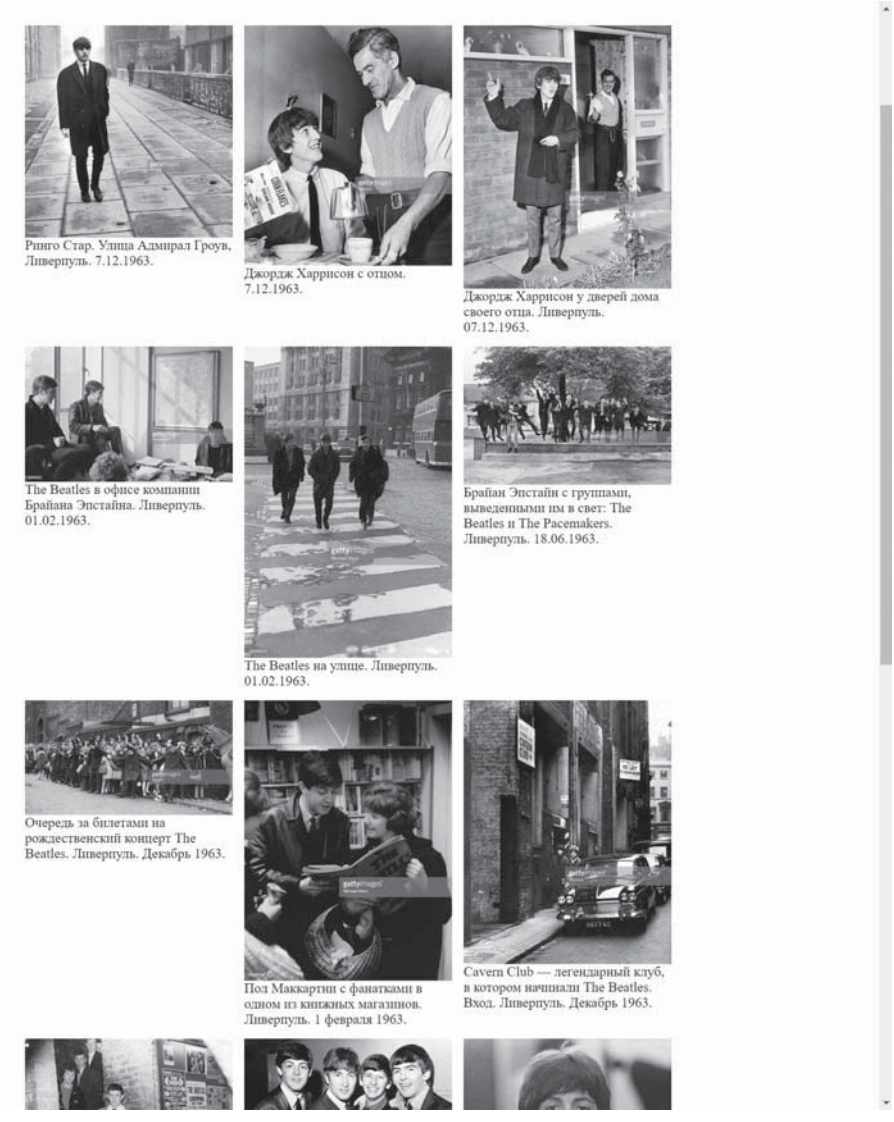
✓ Что у нас получилось? ⌄

Теперь следует немного, пунктов на 10, отделить фото друг от друга, задав у них соответствующие просветы по горизонтали и вертикали.

6. Дополним стиль, применяемый к семантическому основному содержимому:

```
main {
    . . .
    gap: 10pt;
}
```

Непорядок! ⌄



Указав у элементов-фотографий просветы по горизонтали, мы тем самым увеличили их ширину. В результате веб-обозревателю не хватит места,

чтобы вывести в одной строке четыре фотографии, и он сможет вывести лишь три. Что мы и наблюдаем на рисунке...

Нужно скорректировать ширину отдельного элемента-фото таким образом, чтобы учесть размеры горизонтальных просветов между фотографиями. Новое значение ширины мы получим, вычтя из всей ширины родителя произведение 10 пунктов (размер просвета) и 3 (количество горизонтальных просветов), после чего разделив полученную разность на 4 (требуемое количество фотографий в строке).

7. Исправим стиль, применяемый к элементам со стилевыми классами `photo`:

```
~~main .photo { width: 25%; }~~
main .photo { width: calc((100% - 10pt * 3) / 4); }
```

✓ Вот теперь порядок! ⌄

23.8. Упражнение. Кирпичная кладка, часть 2

Всем хороша подборка «битловских» фото, сделанная при выполнении *упражнения 23.7*. Вот только у многих фото само изображение занимает лишь часть отведенного под него места, а остальное пространство остается незанятым... Исправим это!

1. Найдем в папке 23\ex23.7 сопровождающего книгу файлового архива (см. *приложение 4*) папку the-beatles-photos и скопируем ее куда-либо на локальный диск.

   Элементы-фотографии (блоки со стилевыми классами `photo`) уже растянуты в направлении, перпендикулярном направлению выстраивания, — это поведение веб-обозревателя по умолчанию (см. *разд. 23.3.2*).

   Нам останется сделать так, чтобы тег `<img>` занимал все пространство элемента-фотографии, не занятое подписью (обычным блоком). Достичь этого можно, указав у элемента-фотографии флекс-разметку с выстраиванием по вертикали, предписав тегу `<img>` растягиваться в направлении выстраивания, чтобы занять все свободное пространство в родителе, и запретив подписи сжиматься.

   Сначала укажем у элемента-фотографии флекс-разметку и выстраивание по вертикали.

2. Откроем таблицу стилей styles\main.css в текстовом редакторе и исправим стиль, применяемый к элементам со стилевыми классами `photo`:

   ```
   main .photo { width: calc((100% - 10pt * 3) / 4); }
   main .photo {
       width: calc((100% - 10pt * 3) / 4);
       display: flex;
       flex-direction: column;
   }
   ```

   Далее укажем у тегов `<img>` степень растягивания по направлению выстраивания, равную 1, чтобы эти теги растягивались. Базовый размер указывать не будем — тогда веб-обозреватель примет его равным изначальной высоте изображения, записанной в графическом файле.

3. Дополним стиль, применяемый к изображениям, находящимся в элементах со стилевыми классами `photo`:

   ```
   main .photo img {
       . . .
       flex-grow: 1;
   }
   ```

✓ Что у нас получилось? ⌄

Показана лишь одна строка с изображениями.

Сами изображения, выводимые в тегах `<img>`, оказались растянутыми. Устранить этот недочет можно, указав у этих тегов такой режим масштабирования выводимого изображения, при котором оно полностью покрывает тег.

4. Дополним тот же стиль:

```
main .photo img {
    . . .
    object-fit: cover;
}
```

✓ Результат ⌄

Замечательно!

Осталось указать у подписей степень сжатия, равную 0, чтобы запретить им сжиматься в направлении выстраивания. Так сказать, на всякий случай.

5. Добавим стиль, применяемый к блокам, находящимся внутри блоков со стилевыми классами `photo`:

```
main .photo div { flex-shrink: 0; }
```

Визуально страница после этого не изменится, а мы устраним возможные проблемы.

23.9. Самостоятельные упражнения

- Переместите фото членов группы «The Beatles» в начало подборки.

ПОДСКАЗКА

Используйте инструменты, описанные в *разд. 23.6*. Сами стили, задающие порядок следования, можете оформить в виде встроенных.

- Выведите подпись к фотографиям белым шрифтом с засечками кеглем 14 пунктов на черном фоне.

✓ У вас должно получиться ⌄

- Создайте другую редакцию того же сайта — у которой элементы-фотографии имеют фиксированную ширину в 400 пикселов. Сохраните ее в папке the-beatles-photos-fixed.

Урок 24. Сеточная разметка

Фиксированная сетка.
Автоматическая сетка.
Позиционирование элементов в ячейках сетки.
Выравнивание элементов.
Просветы между элементами.

Сеточная разметка

Разметка, при которой пространство внутри элемента-родителя организуется в виде сетки, а элементы-потомки помещаются в ячейки этой сетки.

Сетка

Структура, в виде которой организуется пространство родителя при сеточной разметке. Имеет вид таблицы, состоящей из заданного количества строк и столбцов указанных размеров.

ВНИМАНИЕ!

Все описанные здесь атрибуты стилей являются ненаследуемыми.

24.1. Создание сеточной разметки

Для создания в нужном элементе страницы сеточной разметки следует добавить в стиль этого элемента атрибут стиля `display` со значением `grid`:

```
.grid-element { display: grid; }
```

По умолчанию будет создана автоматическая сетка (см. *разд. 24.3*) из одного столбца. В ячейках этой сетки и будут размещаться элементы-потомки.

Атрибут стиля `display` также поддерживает значение `inline-grid`, которое, помимо создания сеточной разметки, заставляет элемент вести себя как встроенный.

24.2. Фиксированная сетка

> ***Фиксированная сетка***
>
> Сетка, все строки и столбцы которой определены явно.

В ячейках фиксированной сетки можно разместить ограниченное количество элементов-потомков. Если потомков окажется больше, чем ячеек в фиксированной сетке, для размещения оставшихся потомков будет создана автоматическая сетка (см. *разд. 24.3*).

24.2.1. Описание фиксированной сетки

Для описания фиксированной сетки применяются два следующих атрибута стиля:

- `grid-template-rows` — описание набора строк, формирующих сетку;
- `grid-template-columns` — описание набора столбцов.

Значения обоих атрибутов стиля записываются в формате:

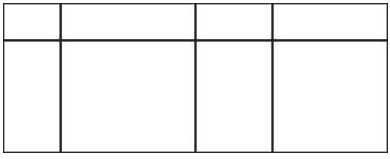
```
<размер строки или столбца 1> <размер строки или столбца 2>
                              . . .
                              <размер строки или столбца N>
```

Сколько указано величин *размеров*, столько строк или столбцов в сетке и будет создано.

В качестве отдельного *размера* можно указать:

- величину в какой-либо единице измерения;
- *<количество частей>*`fr` — свободное пространство родителя будет разделено на равные части, и строка (столбец) получит размер, равный заданному *количеству* таких частей.

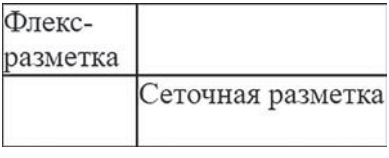
Пример	Результат

`.grid1 {`
`  display: grid;`
`  width: 200px;`
`  grid-template-rows: 1fr 3fr;`
`  grid-template-columns: 30px 70px 40px 60px;`
`}`
	

При использовании единицы измерения `fr` следует удостовериться, что в родителе есть свободное пространство. В противном случае все строки (столбцы), размеры которой заданы с применением этой единицы измерения, получат нулевой размер;

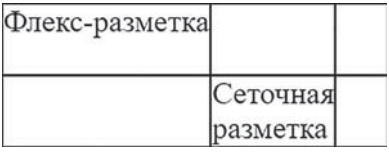
- `minmax(<`*минимум*`>, <`*максимум*`>)` — размер будет подобран веб-обозревателем, основываясь на размерах родителя и содержимого потомков, в диапазоне между заданными *минимумом* и *максимумом*.

Пример	Результат

`.grid2 {`
`  width: 200px;`
`  display: grid;`
`  grid-template-rows: 1fr 1fr;`
`  grid-template-columns:`
`  minmax(50px, 90px)`
`  minmax(110px, 150px);`
`}`
	

- `min-content` — минимальный необходимый размер для вывода содержимого всех потомков, расположенных в этой строке (этом столбце);
- `max-content` — размер, необходимый для вывода в одну строку содержимого всех потомков, расположенных в этой строке (этом столбце).

Пример	Результат

`.grid3 {`
`  width: 200px;`
`  display: grid;`
`  grid-template-rows: 1fr 1fr;`
`  grid-template-columns:`
`  max-content`
`  min-content 1fr;`
`}`
	

- `auto` — размер будет подобран веб-обозревателем в диапазоне между значениями `min-content` и `max-content` (фактически аналог `minmax(min-content, max-content)`).

Пример

```
.grid4 {
    width: 200px;
    display: grid;
    grid-template-rows: 1fr 1fr;
    grid-template-columns: auto auto;
}
```

Результат

Флекс-разметка	
	Сеточная разметка

- `fit-content(`*значение*`)` — размер будет подобран веб-обозревателем в диапазоне между:
  - наименьшим из следующих значений: `min-content` и заданное *значение*;
  - значением `max-content`.

Пример

```
.grid5 {
    width: 200px;
    display: grid;
    grid-template-rows: 1fr 1fr;
    grid-template-columns:
                fit-content(40%)
                fit-content(60%);
}
```

Результат

Флекс-разметка	
	Сеточная разметка

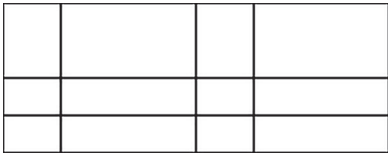
- `repeat()` — создает заданное *количество* единичных строк (столбцов) с указанным размером или наборов таковых. Функция записывается в формате:

  `repeat(<`*количество*`>, <`*описание строки (столбца) или их набора*`>)`

  Описание строки (столбца) или их набора записывается в любом из приведенных ранее форматов.

  В качестве *количества* можно указать:
  - целое число без единицы измерения;
  - `auto-fill` — строка (столбец) или их набор, *описание* которого было задано, будут повторены столько раз, сколько необходимо, чтобы заполнить родитель.

Пример	Результат

`.grid6 {`
`  width: 200px;`
`  display: grid;`
`  grid-template-rows:`
`  2fr repeat(2, 1fr);`
`  grid-template-columns:`
`  repeat(auto-fill, 30px 70px);`
	

- `none` — строки (столбцы) в фиксированной сетке определены не будут. Для размещения потомков будет создана автоматическая сетка (см. *разд. 24.3*).

Значение по умолчанию: `none`.

24.2.2. Позиционирование потомков в ячейках сетки

По умолчанию элементы-потомки родителя, в котором создана сеточная разметка, располагаются в ячейках созданной сетки построчно: первый потомок — в первой ячейке первой строки, второй — во второй ячейке первой строки, третий — в первой ячейке второй строки и т. д.

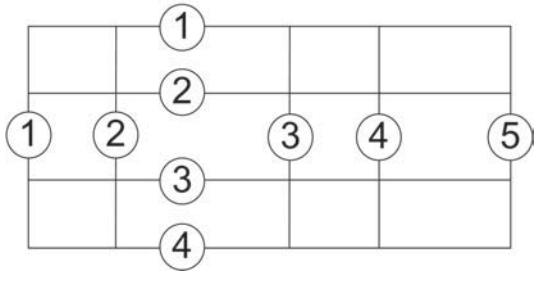
Разумеется, мы можем располагать потомки в произвольных ячейках. Допускается также при необходимости не заполнять все ячейки сетки, оставив часть ячеек пустыми.

Позиционировать элементы-потомки в ячейках сетки, описанной в элементе-родителе, можно тремя способами.

24.2.2.1. Позиционирование по номерам линий сетки

Первый способ — позиционирование потомков по порядковым номерам разделительных линий сетки.

Линии сетки, описанной в родителе, нумеруются, начиная с 1. Горизонтальные линии отсчитываются сверху вниз, вертикальные — слева направо. Пример:



Пример CSS-кода, создающего такую сетку:

```
.grid1, .grid2, grid3, grid4 {
    grid-template-rows: repeat(3, 40px);
    grid-template-columns: 30px 70px 40px 60px;
}
```

Для указания номеров линий, по которым должен позиционироваться потомок, применяются следующие атрибуты стиля:

- `grid-column-start` — номер начальной вертикальной линии (на ней будет находиться левая граница потомка);
- `grid-column-end` — номер конечной вертикальной линии (по которой будет располагаться правая граница потомка);
- `grid-row-start` — номер начальной горизонтальной линии (верхняя граница потомка);
- `grid-row-end` — номер конечной горизонтальной линии (нижняя граница потомка).

Эти атрибуты стиля записываются в стиле, применяемом к потомку, который нужно разместить в сетке.

В качестве значения у них можно указать:

- целое число без единицы измерения — задаст порядковый номер линии, горизонтальной или вертикальной. Если эта числовая величина:
  - положительная — отсчет линий ведется сверху вниз и слева направо;
  - отрицательная — отсчет линий ведется снизу вверх и справа налево.

  Величина `0` является недопустимой.

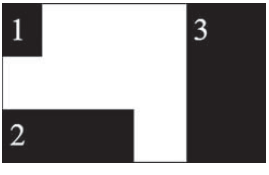
Пример

```
.grid1 > .child1 {
    grid-row-start: 1;
    grid-row-end: 2;
    grid-column-start: 1;
    grid-column-end: 2;
}
.grid1 > .child2 {
    grid-row-start: 3;
    grid-row-end: 4;
    grid-column-start: 1;
    grid-column-end: 3;
}
```

Пример (продолжение)

```
.grid1 > .child3 {
    grid-row-start: 1;
    grid-row-end: -1;
    grid-column-start: 4;
    grid-column-end: -1;
}
```

Результат



- `auto` — построчное позиционирование потомков.

Значение по умолчанию у всех этих атрибутов стиля: `auto`.

У атрибутов стиля `grid-column-end` и `grid-row-end` значение также можно указать в формате:

`span [<`*количество ячеек*`>]`

Оно задаст *количество ячеек* в соответствующем направлении, которое должен занять потомок. Если *количество ячеек* не задано, оно получит значение `1`.

Пример

```
.grid2 > .child1 {
    grid-row-start: 1;
    grid-row-end: span;
    grid-column-start: 1;
    grid-column-end: span;
}
.grid2 > .child2 {
    grid-row-start: 3;
    grid-row-end: span;
    grid-column-start: 1;
    grid-column-end: span 2;
}
```

Пример (продолжение)

```
.grid2 > .child3 {
    grid-row-start: 1;
    grid-row-end: span 3;
    grid-column-start: 4;
    grid-column-end: span;
}
```

Результат

CSS-код, написанный с применением приведенных ранее атрибутов стиля, получается чрезвычайно громоздким. Существенно сократить его можно, применив такие атрибуты стиля:

- `grid-column` — задает начальную и конечную вертикальные линии сетки;
- `grid-row` — задает начальную и конечную горизонтальные линии сетки.

Значения у обоих атрибутов стиля можно записать следующие:

- *<номер начальной линии>* `[/ <`*номер конечной линии*`>]`

  Если *номер конечной линии* не указан, он получит значение `span`.

Пример

```
.grid3 > .child1 {
    grid-row: 1;
    grid-column: 1;
}
.grid3 > .child2 {
    grid-row: 3;
    grid-column: 1 / span 2;
}
.grid3 > .child3 {
    grid-row: 1 / span 3;
    grid-column: 4;
}
```

Результат

- `auto` — построчное позиционирование потомков.

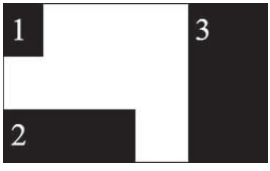
Значение по умолчанию: `auto`.

Еще значительнее сократить CSS-код можно, применив атрибут стиля `grid-area`, устанавливающий сразу все параметры позиционирования у потомка. Его значение можно записать в виде:

- *<номер нач. гориз. линии>* / *<номер нач. верт. линии>* [/ *<номер конеч. гориз. линии>* [/ *<номер конеч. верт. линии>*]]

  Если параметры *номер конеч. верт. линии* и (или) *номер конеч. гориз. линии* пропущены, они получат значение `span`.

Пример	Результат

`.grid4 > .child1 {`
`  grid-area: 1 / 1 / 2 / 2;`
`}`
`.grid4 > .child2 {`
`  grid-area: 3 / 1 / 4 / 3;`
`}`
`.grid4 > .child3 {`
`  grid-area: 1 / 4 / -1 / -1;`
`}`
	

- `auto` — построчное позиционирование потомков.

Значение по умолчанию: `auto`.

24.2.2.2. Позиционирование по именам линий сетки

Некоторым (или всем) линиям сетки можно дать удобные для запоминания имена с целью упростить позиционирование потомков по ним.

Имена линий должны содержать только буквы, цифры, символы дефиса и подчеркивания. Буквы можно применять любые, хотя традиционно используются лишь символы основной латиницы[1].

Чтобы дать линиям сетки имена, их следует поместить в состав значений атрибутов стиля `grid-template-rows` и `grid-template-columns` (см. *разд. 24.2.1*). Имена записываются в квадратных скобках в промежутках между отдельными значениями размера строки или столбца и отделяются от них пробелами. Если требуется дать имя первой линии сетки, это имя надо поместить перед первым значением размера; если нужно дать имя последней линии сетки, имя помещается после последнего значения размера.

[1] Поскольку их проще набирать на клавиатуре.

Пример сетки, у которой все горизонтальные линии имеют имена вида r<*номер*>, а все вертикальные — имена вида c<*номер*>:

```
.grid5 {
    grid-template-rows: [r1] 40px [r2] 40px [r3] 40px [r4];
    grid-template-columns: [c1] 30px [c2] 70px [c3] 40px [c4] 60px [c5];
}
```

Для указания на определенную линию сетки ее имя записывается в атрибутах стиля `grid-column-start`, `grid-row-start`, `grid-column-end`, `grid-row-end`, `grid-column` и `grid-row` вместо числовой величины.

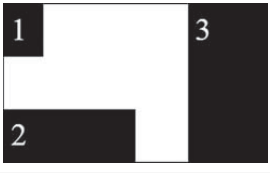
Пример

```
.grid5 > .child1 {
    grid-row-start: r1;
    grid-row-end: r2;
    grid-column-start: c1;
    grid-column-end: c2;
}
.grid5 > .child2 {
    grid-row-start: r3;
    grid-row-end: r4;
    grid-column-start: c1;
    grid-column-end: c3;
}
```

Пример (продолжение)

```
.grid5 > .child3 {
    grid-row-start: r1;
    grid-row-end: r4;
    grid-column-start: c4;
    grid-column-end: c5;
}
```

Результат



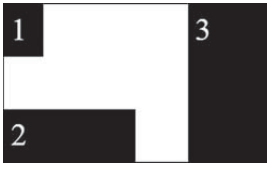
Давать имена всем линиям сетки совершенно необязательно:

```
.grid6 {
    grid-template-rows: 40px 40px [r3] 40px;
    grid-template-columns: 30px 70px [c3] 40px [c4] 60px;
}
```

Пример

```
.grid6 > .child1 {
    grid-row: 1;
    grid-column: 1;
}
.grid6 > .child2 {
    grid-row: r3;
    grid-column: 1 / c3;
}
.grid6 > .child3 {
    grid-row: 1 / -1;
    grid-column: c4;
}
```

Результат



Какой-либо линии можно дать сразу несколько имен, записав их в квадратных скобках через пробел:

```
.grid7 {
    grid-template-rows: [r1 r-first] 40px [r2] 40px [r3] 40px
                                                    [r4 r-last];
    grid-template-columns: [c1] 30px [c2] 70px [c3] 40px [c4] 60px
                                                    [c5 c-last];
}
```

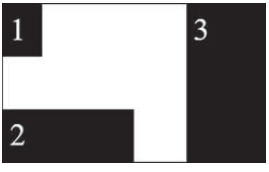
Пример

```
.grid7 > .child1 {
    grid-row-start: r1;
    grid-row-end: r2;
    grid-column-start: c1;
    grid-column-end: c2;
}
.grid7 > .child2 {
    grid-row-start: r3;
    grid-row-end: r4;
    grid-column-start: c1;
    grid-column-end: c3;
}
```

Пример (продолжение)

```
.grid7 > .child3 {
    grid-row-start: r-first;
    grid-row-end: r-last;
    grid-column-start: c4;
    grid-column-end: c-last;
}
```

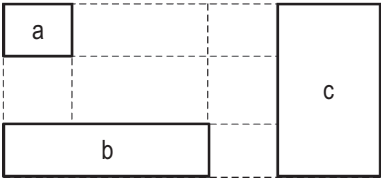
Результат



24.2.2.3. Позиционирование по именованным областям сетки

Существует возможность создать в сетке готовые области прямоугольной формы и дать им уникальные имена. Впоследствии, чтобы поместить тот или иной потомок в такую область, достаточно лишь указать имя этой области.

Так, в следующем примере в сетке созданы три области с именами `a`, `b` и `c`:



Создать можно лишь строго прямоугольные именованные области. Области другой формы (например, Г-образные) создать нельзя.

Имена областей должны удовлетворять тем же требованиям, что предъявляются к именам линий сетки (см. *разд. 24.2.2.2*). Как правило, имена областей делают однобуквенными (как в приведенном ранее примере) — для удобства их ввода.

Набор областей в готовой сетке записывается в атрибуте стиля `grid-template-areas`. Его значением может быть:

- набор строковых значений, разделенных пробелами, — каждое строковое значение представляет одну строку сетки.

  Каждое из этих строковых значений должно содержать набор из определенных символьных последовательностей, заключенных в одинарные или двойные кавычки и разделенных пробелами. Каждая такая символьная последовательность представляет очередную ячейку, входящую в состав соответствующей строки сетки. В качестве символьной последовательности можно записать:

  - имя одной из создаваемых областей — тогда соответствующая ячейка строки войдет в состав области с указанным именем:
  - точки (`.`) — тогда соответствующая ячейка не войдет в состав ни одной из создаваемых областей.

    Чтобы поместить элемент-потомок в такую ячейку, следует воспользоваться инструментами, описанными в *разд. 24.2.2.1* и *24.2.2.2*;

- `none` — в сетке не будут созданы именованные области.

Значение по умолчанию: `none`.

Пример CSS-кода, создающего сетку с областями из приведенного ранее примера:

```
.grid8 {
    grid-template-rows: repeat(3, 40px);
    grid-template-columns: 30px 70px 40px 60px;
    grid-template-areas: 'a . . c'
                         '. . . c'
                         'b b . c';
}
```

Для указания, в какой области следует поместить элемент-потомок, следует использовать уже знакомый нам атрибут стиля `grid-area`. В качестве его значения можно указать:

- имя области сетки;
- `auto` — потомок позиционируется на основе сведений, записанных в других атрибутах стиля (см. *разд. 24.2.2.1* и *24.2.2.2*).

Значение по умолчанию: `auto`.

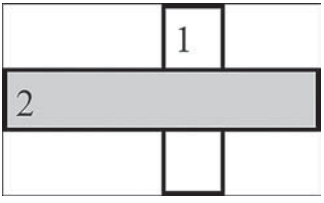
Пример	Результат

`.grid8 > .child1 {`
`  grid-area: a;`
`}`
`.grid8 > .child2 {`
`  grid-area: b;`
`}`
`.grid8 > .child3 {`
`  grid-area: c;`
`}`
	

24.2.2.4. Наложение потомков

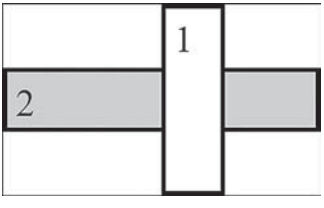
Одну и ту же ячейку сетки могут занимать несколько потомков, накладываясь друг на друга. Потомки, записанные в HTML-коды позже, будут перекрывать потомков, записанных раньше.

Пример	Результат

`.grid9 > .child1 {`
`  grid-row: 1 / 4;`
`  grid-column: 3 / 4;`
`  background-color: white;`
`}`
`.grid9 > .child2 {`
`  grid-row: 2 / 3;`
`  grid-column: 1 / 5;`
`  background-color: lightgrey;`
`}`
	

Изменить порядок перекрытия потомками друг друга можно, применив атрибут стиля `z-index` (см. *разд. 22.2.2*).

Пример	Результат

`.grid10 > .child1 {`
`  grid-row: 1 / 4;`
`  grid-column: 3 / 4;`
`  background-color: white;`
`  z-index: 1;`
`}`
`.grid10 > .child2 {`
`  grid-row: 2 / 3;`
`  grid-column: 1 / 5;`
`  background-color: lightgrey;`
`}`
	

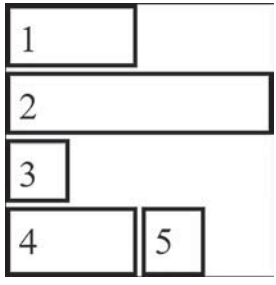
24.2.3. Параметры автоматического позиционирования потомков

Если местоположение того или иного потомка не указано явно, веб-обозреватель позиционирует его самостоятельно, согласно заданным правилам.

Для указания правил автоматического позиционирования потомков служит атрибут стиля `grid-auto-flow`. Вот поддерживаемые им значения:

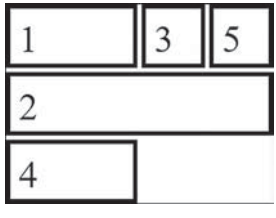
- `row` — потомки выстраиваются по строкам, строго в том порядке, в котором они записаны в HTML-коде. В некоторых строках сетки могут остаться незаполненные ячейки.

Пример	Результат

`.grid11 {`
`  grid-auto-flow: row;`
`}`
`.grid11 > .child1 {`
`  grid-column: 1 / 3;`
`}`
`.grid11 > .child2 {`
`  grid-column: 1 / 5;`
`}`
`.grid11 > .child4 {`
`  grid-column: 1 / 3;`
`}`
	

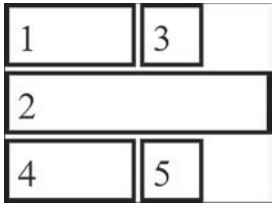
- `row dense` — то же самое, что и `row`, только веб-обозреватель, по возможности, будет заполнять все ячейки в строках.

Пример	Результат

`.grid12 {`
`  grid-auto-flow: row dense;`
`}`
`.grid12 > .child1 {`
`  grid-column: 1 / 3;`
`}`
`.grid12 > .child2 {`
`  grid-column: 1 / 5;`
`}`
`.grid12 > .child4 {`
`  grid-column: 1 / 3;`
`}`
	

- `column` — потомки выстраиваются по столбцам, строго в том порядке, в котором они записаны в HTML-коде. В некоторых столбцах сетки могут остаться незаполненные ячейки.

Пример	Результат

`.grid13 {`
`  grid-auto-flow: column;`
`}`
`.grid13 > .child1 {`
`  grid-column: 1 / 3;`
`}`
`.grid13 > .child2 {`
`  grid-column: 1 / 5;`
`}`
`.grid13 > .child4 {`
`  grid-column: 1 / 3;`
`}`
	

- `column dense` — то же самое, что и `column`, только веб-обозреватель, по возможности, будет заполнять все ячейки в столбцах.

Значение по умолчанию: `row`.

24.2.4. Указание сразу всех параметров фиксированной сетки

Задать сразу все параметры фиксированной сетки — ее описание и набор областей — можно в атрибуте стиля `grid-template`. Его значение записывается в одном из двух форматов:

- *<описание строк сетки (*`grid-template-rows`*)>* /
  <описание столбцов сетки (`grid-template-columns`*)>*

  При использовании этого формата сетка не будет содержать именованных областей.

  Пример:

```
.grid1 {
    grid-template: repeat(3, 40px) / 30px 70px 40px 60px;
}
```

- *<описание областей в строке 1> <размер строки 1>*
  <описание областей в строке 2> <размер строки 2>
  . . .
  <описание областей в строке N> <размер строки N> /
  <описание столбцов сетки (`grid-template-columns`*)>*

Описания именованных *областей в строках* записываются в виде строковых значений в том же формате, что применяется в атрибуте стиля `grid-template-areas` (см. *разд. 24.2.2.3*). *Размеры строк* задаются в том же формате, который применяется в атрибуте стиля `grid-template-rows` (см. *разд. 24.2.2.1*).

Пример:

```
.grid8 {
    grid-template: 'a . . c' 40px
                   '. . . c' 40px
                   'b b . c' 40px /
                   30px 70px 40px 60px;
}
```

24.3. Автоматическая сетка

Если у элемента-родителя, в котором указана сеточная разметка, потомков больше, чем ячеек в созданной фиксированной сетке, веб-обозреватель для размещения «лишних» потомков создаст автоматическую сетку.

> ***Автоматическая сетка***
>
> Дополнение к фиксированной сетке, создаваемое для размещения потомков, которые не поместились в ячейках фиксированной сетки.

Для указания параметров автоматической сетки применяются следующие атрибуты стиля:

- `grid-auto-rows` — описывает строки автоматической сетки. Поддерживает те же значения, что и атрибут стиля `grid-template-rows` (см. *разд. 24.2.1*). Значение по умолчанию: `auto`;
- `grid-auto-columns` — описывает столбцы автоматической сетки. Поддерживает те же значения, что и атрибут стиля `grid-template-columns`. Значение по умолчанию: `auto`.

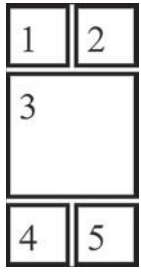
Веб-обозреватель может создать в автоматической сетке произвольное количество строк и столбцов, основываясь на описаниях, которые записаны в приведенных атрибутах стиля. Рассмотрим пример:

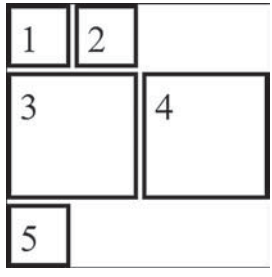
```
grid-auto-rows: 20pt 50pt;
```

Первая строка автоматической сетки будет иметь высоту в 20 пунктов, вторая — 50 пунктов, третья — снова 20 пунктов, четвертая — 50 пунктов и т. д.

В остальном автоматическая сетка ничем не отличается от фиксированной. Мы можем как положиться на инструменты автоматического позиционирования потомков (см. *разд. 24.2.3*), так и позиционировать потомки самостоятельно, пользуясь инструментами, описанными в *разд. 24.2.2*.

Пример	Результат

`.grid14 {`
`  grid-template-rows: 40px;`
`  grid-template-columns: repeat(2, 40px);`
`  grid-auto-rows: 80px 40px;`
`}`
`.grid14 > .child3 { grid-column: 1 / 3; }`
	

`.grid15 {`
`  grid-template-rows: 40px;`
`  grid-template-columns: repeat(2, 40px);`
`  grid-auto-rows: 80px 40px;`
`  grid-auto-columns: 80px;`
`}`
`.grid15 > .child3 { grid-column: 1 / 3; }`
`.grid15 > .child4 {`
`  grid-row: 2 / 3;`
`  grid-column: 3 / 4;`
`}`
	

Создавать и фиксированную, и автоматическую сетку совершенно необязательно. Можно ограничиться лишь фиксированной сеткой, а можно сформировать только автоматическую.

24.4. Указание сразу всех параметров фиксированной и автоматической сетки

Наиболее часто создаются две разновидности сетки:

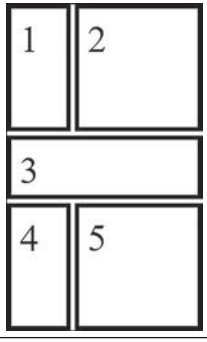
- с фиксированным набором столбцов, автоматически создаваемыми строками и выстраиванием по строкам, — тогда сетка будет «расти» вниз;
- с фиксированным набором строк, автоматически создаваемыми столбцами и выстраиванием по столбцам, — тогда сетка будет «расти» вправо.

В таких случаях удобно использовать атрибут стиля `grid`, позволяющий описать сразу все необходимые параметры создаваемой сетки. Значение этого атрибута стиля может быть записано в одном из указанных далее форматов.

- `auto-flow [dense] <описание автоматических строк> / <описание фиксированных столбцов>`

  Создается сетка с фиксированным набором столбцов, автоматически создаваемыми строками и выстраиванием по строкам, — «растущая» вниз. Если указано слово `dense`, будет включен режим «уплотнения» (как при указании атрибута стиля `grid-auto-flow` со значением `row dense`).

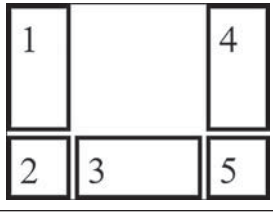
Пример	Результат

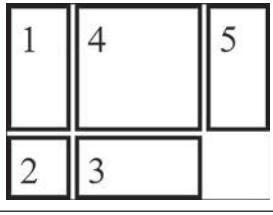
`.grid16 {`
`  grid: auto-flow`
`  80px 40px / 40px 80px;`
`}`
`.grid16 > .child3 {`
`  grid-column: 1 / 3;`
`}`
	

- `<описание фиксированных строк> / auto-flow [dense] <описание автоматических столбцов>`

  Создается сетка с фиксированным набором строк, автоматически создаваемыми столбцами и выстраиванием по столбцам, — «растущая» вправо. Если указано слово `dense`, будет включен режим «уплотнения» (как при указании атрибута стиля `grid-auto-flow` со значением `column dense`).

Пример	Результат

`.grid17 {`
`  grid: 80px 40px /`
`  auto-flow 40px 80px;`
`}`
`.grid17 > .child3 {`
`  grid-row: 2 / 3;`
`}`
	

`.grid18 {`
`  grid: 80px 40px /`
`  auto-flow dense 40px 80px;`
`}`
`.grid18 > .child3 {`
`  grid-row: 2 / 3;`
`}`
	

- в формате, поддерживаемом атрибутом стиля `grid-template` (см. *разд. 24.2.4*) — будет создана полностью фиксированная сетка.

24.5. Выравнивание элементов-потомков

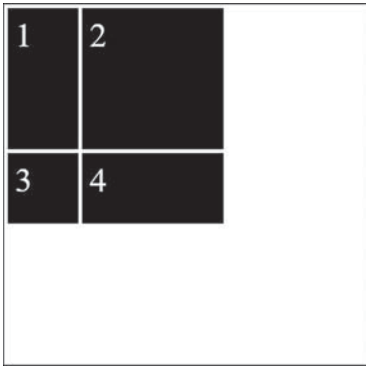
24.5.1. Выравнивание всей совокупности потомков внутри родителя

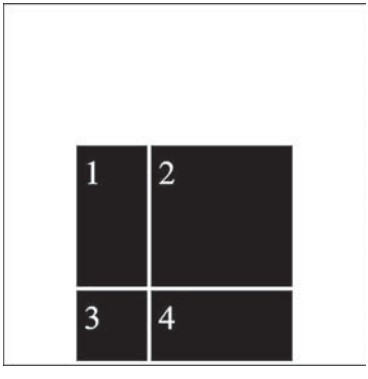
Если совокупные размеры строк и столбцов в сетке меньше размеров элемента-родителя (т. е. в родителе есть свободное место), можно указать выравнивание совокупности потомков в родителе.

Для этого применяются следующие атрибуты стиля, знакомые нам по *разд. 23.3.1*:

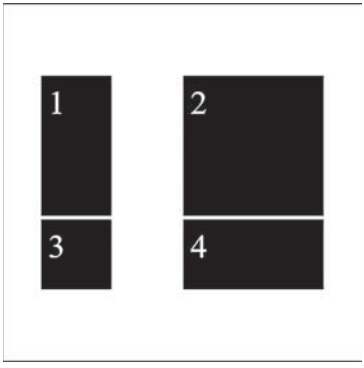
- `justify-content` — выравнивание по горизонтали;
- `align-content` — выравнивание по вертикали.

Пример	Результат

`.grid19 {`
`  width: 250px;`
`  height: 250px;`
`  grid-template: 100px 50px / 50px 100px;`
`  justify-content: normal;`
`  align-content: normal;`
`}`
	

`.grid20 {`
`  width: 250px;`
`  height: 250px;`
`  grid-template: 100px 50px / 50px 100px;`
`  justify-content: center;`
`  align-content: end;`
`}`
	

Пример	Результат

`.grid21 {`
`  width: 250px;`
`  height: 250px;`
`  grid-template: 100px 50px / 50px 100px;`
`  justify-content: space-around;`
`  align-content: center;`
`}`
	

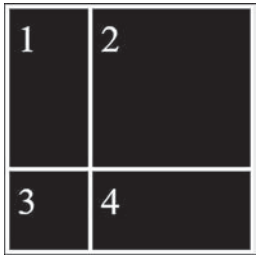
Также можно использовать атрибут стиля `place-content`, который задаст выравнивание сразу в обоих направлениях (см. *разд. 23.3.1*).

24.5.2. Выравнивание всех потомков внутри ячеек сетки

Для выравнивания всех потомков внутри ячеек сетки применяются два атрибута стиля, описываемые далее.

Атрибут стиля `align-items`, описанный в *разд. 23.3.2*, задает выравнивание по вертикали.

Пример	Результат

`.grid22 {`
`  grid-template: 100px 50px / 50px 100px;`
`  align-items: normal;`
`}`
	

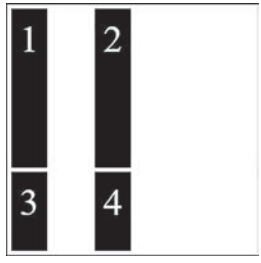
`.grid23 {`
`  grid-template: 100px 50px / 50px 100px;`
`  align-items: end;`
`}`
	

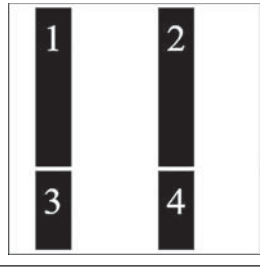
Для задания выравнивания потомков в ячейках по горизонтали следует использовать атрибут стиля `justify-items`. Вот поддерживаемые им значения:

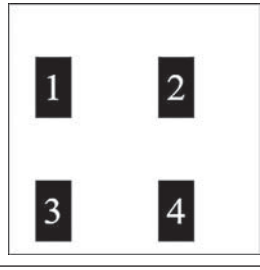
- `normal`, `legacy` или `stretch` — потомки растягиваются на всю ширину ячеек;
- `start`, `flex-start` или `baseline` — выравнивание по левым краям ячеек;
- `end` или `flex-end` — по правым краям;
- `center` — выравнивание по центрам.

Значение по умолчанию: `legacy`.

Пример	Результат

`.grid24 {`
`  grid-template: 100px 50px / 50px 100px;`
`  justify-items: start;`
`}`
	

`.grid25 {`
`  grid-template: 100px 50px / 50px 100px;`
`  justify-items: center;`
`}`
	

`.grid26 {`
`  grid-template: 100px 50px / 50px 100px;`
`  justify-items: center;`
`  align-items: center;`
`}`
	

Также доступен атрибут стиля `place-items`, позволяющий указать выравнивание в обоих направлениях. В качестве его значения можно задать:

- *<выравнивание по горизонтали и вертикали>*

  Пример:

  ```
  place-items: center;
  ```

- *<выравнивание по вертикали> <выравнивание по горизонтали>*

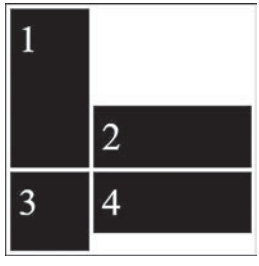
  Пример:

  ```
  place-items: start center;
  ```

24.5.3. Выравнивание отдельных потомков внутри ячеек сетки

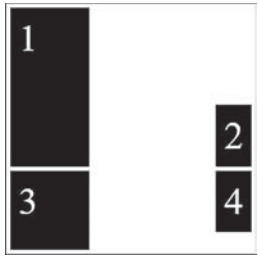
Для задания выравнивания отдельного потомка в ячейке сетки по вертикали служит атрибут стиля `align-self` (см. *разд. 23.3.3*).

Пример	Результат

`.grid27 {`
`  grid-template: 100px 50px / 50px 100px;`
`}`
`.grid27 > .child2 {`
`  align-self: end;`
`}`
`.grid27 > .child4 {`
`  align-self: start;`
`}`
	

Выравнивание отдельного элемента в ячейке по горизонтали можно задать в атрибуте стиля `justify-self`. Он поддерживает те же значения, что и атрибут стиля `justify-items` (см. *разд. 24.5.2*).

Пример	Результат

`.grid28 {`
`  grid-template: 100px 50px / 50px 100px;`
`}`
`.grid28 > .child2 {`
`  justify-self: end;`
`  align-self: end;`
`}`
`.grid28 > .child4 {`
`  justify-self: end;`
`  align-self: start;`
`}`
	

Атрибут стиля `place-self` позволяет указать выравнивание в обоих направлениях. В качестве его значения можно задать:

- *<выравнивание по горизонтали и вертикали>*

  Пример:

  ```
  place-self: end;
  ```

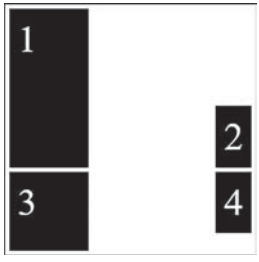
- *<выравнивание по вертикали> <выравнивание по горизонтали>*

  Пример:

  ```
  place-self: start end;
  ```

Кроме того, можно воспользоваться атрибутами стиля «семейства» `margin` (см. *разд. 23.3.3*).

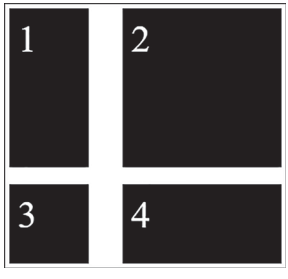
Пример	Результат

`.grid29 {`
`  grid-template: 100px 50px / 50px 100px;`
`}`
`.grid29 > .child2 {`
`  margin-left: auto;`
`  margin-top: auto;`
`}`
`.grid29 > .child4 {`
`  margin-left: auto;`
`  margin-bottom: auto;`
`}`
	

24.6. Просветы между потомками

Просветы между отдельными потомками могут быть созданы атрибутами стиля `column-gap`, `row-gap` и `gap` (см. *разд. 23.5*).

Пример	Результат

`.grid30 {`
`  grid-template: 100px 50px / 50px 100px;`
`  column-gap: 16pt;`
`  row-gap: 8pt;`
`}`
	

24.7. Упражнение. Современный рекламный проспект

Пока мы изучали сеточную разметку, знакомый владелец фирмы по торговле товарами для детского творчества прослышал о наших достижениях. И обратился к нам с просьбой сделать современный рекламный сайт-проспект для своей фирмы.

Наподобие такого:

Знакомый предоставил нам изображения для этого сайта и красивый загружаемый шрифт Exo2, которым следует вывести весь текст.

1. Найдем в папке 24\!sources сопровождающего книгу файлового архива (см. *приложение 4*) папку kid-creativity и скопируем ее куда-либо на локальный диск.

   В этой папке находятся папки:

   - fonts — с файлами Exo2-Regular.woff, Exo2-Bold.woff, Exo2-Italic.woff и Exo2-BoldItalic.woff, содержащими разные начертания загружаемого шрифта Exo2;
   - images — с файлами kids.jpg, funny-lines.jpg и color-splashes.jpg, хранящими изображения для сайта.

   Вся работа будет проводиться в папке kid-creativity.
2. Создадим главную страницу index.html и запишем в нее следующий код:

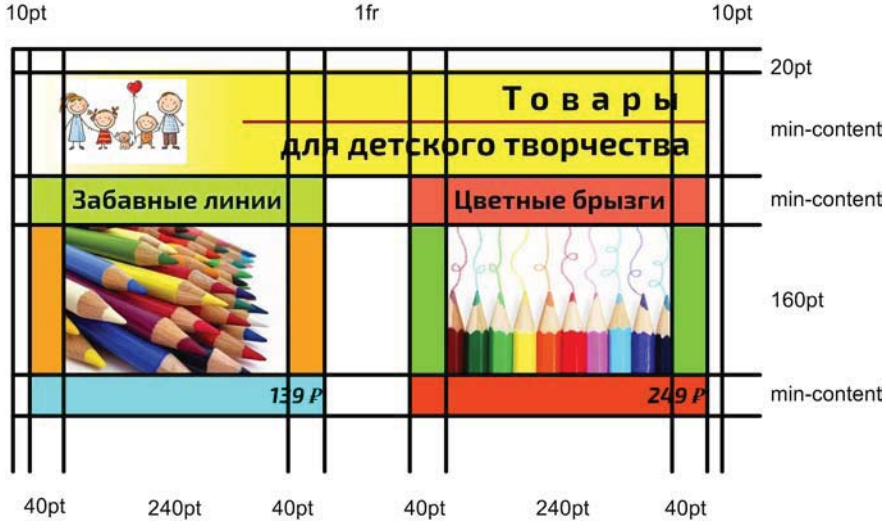
```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="UTF-8">
        <title>Товары для детского творчества</title>
        <link rel="stylesheet" href="styles/main.css">
    </head>
    <body>
    </body>
</html>
```

3. Создадим папку styles, а в ней — таблицу стилей main.css.

   Сначала необходимо записать в таблице стилей описание шрифта Exo2, чтобы веб-обозреватель смог его загрузить и использовать для вывода текста.

4. Добавим в таблицу стилей styles\main.css код, описывающий шрифт Exo2, самостоятельно, пользуясь указаниями из *разд. 14.2* и *упражнения 14.3*.

   Посмотрим на изображение страницы, приведенное ранее, и подумаем, как разбить его на отдельные строки и столбцы, сформировав тем самым сетку. Заодно прикинем, какие указать размеры у этих строк и столбцов.



   Необходимые размеры были подобраны автором экспериментально.

   Самая верхняя строка (высотой 40 пунктов), крайние левый и правый столбцы (по 10 пунктов) создадут просветы между краями области просмотра и содержанием страницы. Средний столбец (`1fr`) создаст просвет между левой и правой колонками с описаниями товаров.

   Строки 2 и 3 будут иметь высоту, равную минимальной высоте, необходимой для вывода их содержимого (`min-content`).

   Можно начинать работу.

   Все содержание страницы мы поместим в блок, расположенный по центру страницы (как сделали при выполнении *упражнения 15.5*), — его

ширина будет меняться в диапазоне 800–1000 пикселов. Укажем у этого блока стилевой класс `container`, чтобы позже применить к нему стиль. Также уберем просветы между границами области просмотра и содержанием страницы.

5. Вставим в секцию тела страницы index.html код, который создаст «всеобъемлющий» блок:

```
<body>
    <div class="container">
    </div>
</body>
```

6. Добавим в таблицу стилей styles\main.css стили, задающие оформление для секции тела и ранее созданного блока:

```
body { margin: 0; }
.container {
    min-width: 800px;
    max-width: 1000px;
    margin: 0 auto;
}
```

У блока со стилевым классом `container` укажем сеточную разметку. Сформируем в ней фиксированную сетку, содержащую весь набор столбцов и начальные две строки: первая из них создаст просвет, а вторая — вместит заголовок сайта. Остальные строки оформим в виде автоматической сетки — это упростит нам дальнейшую работу по добавлению на страницу описаний других товаров.

7. Добавим в стиль, применяемый к блоку со стилевым классом `container`, CSS-код, создающий требуемую сетку:

```
.container {
    . . .
    display: grid;
    grid-template-rows: 20pt min-content;
    grid-template-columns: 10pt 40pt 240pt 40pt 1fr 40pt 240pt 40pt
                                                              10pt;
    grid-auto-rows: min-content 160pt min-content;
}
```

Заголовок сайта с текстом «Товары для детского творчества» оформим в виде двух заголовков первого уровня, помещенных в семантическую шапку. Дадим заголовкам первого уровня стилевые классы `heading1` и `heading2`, чтобы применить к ним необходимые стили.

8. Допишем в код страницы index.html фрагмент, который создаст заголовок сайта:

```
<div class="container">
    <header>
        <h1 class="heading1">Товары</h1>
        <h1 class="heading2">для детского творчества</h1>
    </header>
</div>
```

Семантическая шапка, создающая заголовок сайта, займет вторую сверху строку сетки и все столбцы со второго по девятый. Укажем у шапки шрифт Exo2 и выравнивание по правому краю.

А еще создадим у шапки красивый множественный фон, включающий детский рисунок из файла images\kids.jpg, который выводится слева, и бело-желтый градиент, заполняющий шапку целиком. Этот фон похож на тот, что мы в свое время сделали у сайта суши-бара (см. *упражнение 19.8*).

9. Добавим в таблицу стилей styles\main.css стиль для семантической шапки:

```
header {
    grid-row: 2 / 3;
    grid-column: 2 / 9;
    font-family: Exo2;
    text-align: right;
    background: no-repeat url(../images/kids.jpg)
                          30pt center / 140pt auto,
                linear-gradient(to left, yellow 0, yellow 500pt,
                                                  white 100%);
}
```

У обоих заголовков первого уровня зададим нулевые внешние просветы, кегль шрифта 36 пунктов и полужирность.

10. Добавим стиль для элементов со стилевыми классами `heading1` и `heading2`:

```
.heading1, .heading2 {
    margin: 0;
    font-size: 36pt;
    font-weight: bold;
}
```

Верхний заголовок первого уровня получит внутренние просветы: 10 пунктов сверху, 20 пунктов справа и нулевые с остальных сторон. Так мы сделаем заголовок более крупным и заметным. Для этой же цели укажем у него дополнительное расстояние между буквами в 12 пунктов. С нижней стороны заголовка проведем коричневую толстую сплошную линию. А чтобы эта линия не растянулась на всю длину заголовка (что будет выглядеть некрасиво), укажем у заголовка внешний просвет слева, равный длине родителя за вычетом 500 пунктов (значение подобрано опытным путем).

11. Запишем стиль для элемента со стилевым классом `heading1`:

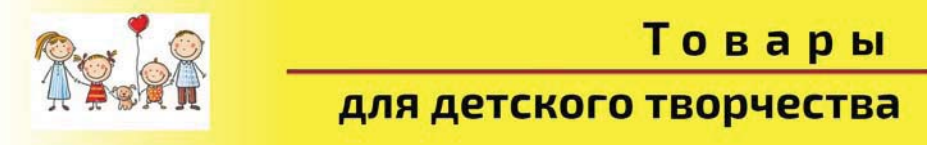
```
.heading1 {
    letter-spacing: 12pt;
    margin-left: calc(100% - 500pt);
    padding: 10pt 20pt 0 0;
    border-bottom: thick solid brown;
}
```

У нижнего заголовка первого уровня укажем только внутренние просветы: 20 пунктов справа, 10 пунктов снизу и нулевые с остальных сторон. Это нужно, чтобы сделать заголовок крупнее.

12. Добавим стиль для элемента со стилевым классом `heading2`:

```
.heading2 { padding: 0 20pt 10pt 0; }
```

✓ Что у нас получилось? ⌄



Весьма недурной заголовок, не находите?

Приступим к работе над левой колонкой, со сведениями о первом товаре — карандашах «Забавные линии» за 139 руб.

Левая колонка будет состоять из пяти блоков (тегов `<div>`):

- верхнего — со стилевыми классами `heading` и `good1-heading` — содержит заголовок второго уровня с названием товара («Забавные линии»).

  Для этого блока мы запишем два стиля. Стиль с селектором `.heading h2` задаст оформление текста заголовка (шрифт, выравнивание

и просветы) и будет также использован для оформления аналогичного блока во второй колонке. Стиль с селектором `.good1-heading` укажет местоположение и цвет фона для этого конкретного блока;

- левого — со стилевым классом `good1-left` — пустого;
- среднего — со стилевым классом `good1-image` — содержит изображение (тег `<img>` со стилевым классом `image` выводит картинку из файла images\funny-lines.jpg);
- правого — со стилевым классом `good1-right` — пустого;
- нижнего — со стилевыми классами `footing` и `good1-footing` — содержит цену («139 ₽»).

  Для этого блока мы также напишем два стиля. Стиль с селектором `.footing` задаст оформление текста цены (шрифт, выравнивание и просветы) и будет также использован для оформления аналогичного блока во второй колонке. Стиль с селектором `.good1-footing` укажет местоположение и цвет фона для этого конкретного блока.

13. Добавим в HTML-код страницы index.html фрагмент, создающий все эти блоки:

```
<div class="container">
    <header>
        . . .
    </header>
    <div class="heading good1-heading">
        <h2>Забавные линии</h2>
    </div>
    <div class="good1-left"></div>
    <div class="good1-center">
        <img class="image" src="images/funny-lines.jpg">
    </div>
    <div class="good1-right"></div>
    <div class="footing good1-footing">
        139 &#8381;
    </div>
</div>
```

Специальный символ `&#8381;` выводит знак рубля (₽).

Заголовок второго уровня, содержащий название товара, будет выводиться полужирным шрифтом Exo2 кеглем 28 пунктов, с выравнива-

нием по центру, без внешних просветов, с внутренними просветами в 10 пунктов со всех сторон.

14. Запишем в таблицу стилей styles\main.css стиль для заголовка второго уровня, находящегося в элементе со стилевым классом `heading`:

```
.heading h2 {
    font: bold 28pt Exo2;
    text-align: center;
    margin: 0;
    padding: 10pt;
}
```

Верхний блок (с названием товара) займет третью строку и столбцы 2–4 и получит желто-зеленый (`greenyellow`) цвет фона.

15. Напишем стиль для элемента со стилевым классом `.good1-heading`:

```
.good1-heading {
    grid-row: 3 / 4;
    grid-column: 2 / 5;
    background-color: greenyellow;
}
```

Левый блок займет четвертую строку и второй столбец и получит оранжевый (`orange`) сплошной фон.

16. Добавим стиль для элемента со стилевым классом `.good1-left`:

```
.good1-left {
    grid-row: 4 / 5;
    grid-column: 2 / 3;
    background-color: orange;
}
```

Изображения, выводящиеся в средних блоках, предварительно превратим в блочные элементы, чтобы без проблем задать у них размеры. Растянем их на всю ширину и высоту родителя и укажем заполнение тега без искажения пропорций и выхода за его пределы.

17. Наберем стиль для элементов со стилевыми классами `image`:

```
.image {
    display: block;
    width: 100%;
    height: 100%;
    object-fit: cover;
}
```

Средний блок поместим в четвертую строку и третий столбец сетки.

18. Введем стиль для элемента со стилевым классом `good1-center`;

```
.good1-center {
    grid-row: 4 / 5;
    grid-column: 3 / 4;
}
```

Правый блок разместим в четвертой строке и четвертом столбце и зададим у него оранжевый фон.

19. Добавим стиль для элемента со стилевым классом `good1-right`:

```
.good1-right {
    grid-row: 4 / 5;
    grid-column: 4 / 5;
    background-color: orange;
}
```

Текст цены будет выводиться полужирным курсивным шрифтом Exo2 кеглем 24 пункта, с выравниванием по правому краю, без внешних просветов, с внутренними просветами в 6 пунктов со всех сторон.

20. Допишем стиль для элементов со стилевыми классами `footing`:

```
.footing {
    font: bold italic 24pt Exo2;
    text-align: right;
    margin: 0;
    padding: 6pt;
}
```

Нижний блок разместится в пятой строке, займет столбцы 2–4 и получит фон цвета циан (`cyan`).

21. Добавим стиль для элемента со стилевым классом `good1-footing`:

```
.good1-footing {
    grid-row: 5 / 6;
    grid-column: 2 / 5;
    background-color: cyan;
}
```

✓ Что у нас получилось? ⌄

Правую колонку с описанием второго товара — карандашей «Цветные брызги» за 249 руб. — вы можете сделать сами. Она будет аналогична левой колонке. Ее отдельные блоки займут строки 3–5 и столбцы 6–8 сетки. Цвета для фонов можете подобрать на собственное усмотрение.

22. Создайте правую колонку самостоятельно.

24.8. Самостоятельные упражнения

- Добавьте на сайт «Товары для детского творчества» еще две колонки ниже имеющихся — с описаниями еще двух товаров: карандашей «Цветной мир» за 229 руб. (изображение хранится в файле images\color-world.jpg) и «Блокнот» за 89 руб. (images\sketchbook.jpg). Фоновые цвета подберите самостоятельно.
- Вставьте вертикальные просветы между заголовком и колонками с описаниями товаров, а также между отдельными колонками.

ПОДСКАЗКА

Добавьте в описание автоматической сетки пустую строку, высоту которой подберите самостоятельно.

✓ У вас должно получиться ⌄

Урок 25. Макеты веб-страниц

Классический макет из двух колонок.
Макет на основе плавающих элементов.
Табличный макет.
Рамочный макет.

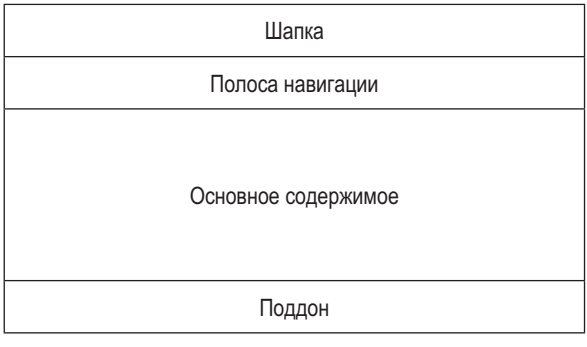
Макет

Шаблон, на основе которого создаются веб-страницы, принадлежащие одному сайту. Включает повторяющиеся на всех страницах элементы (шапку, поддон и панель навигации) и элемент для размещения основного содержимого.

25.1. Классические макеты

В современном веб-дизайне применяются два макета, которые можно назвать классическими:

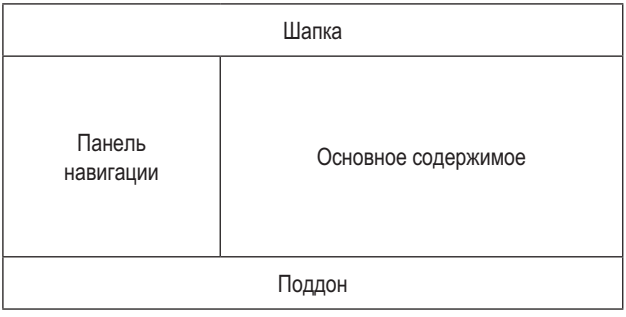
- *простой*, или *одноколоночный*, макет:



Изготовление страниц на основе такого макета не представляет никаких сложностей: нужно лишь расположить блочные элементы с необходимым содержимым — и они сами выстроятся по вертикали друг за другом. В качестве таковых практически всегда применяются элементы семантической разметки (см. *разд. 5.2*).

На основе такого макета (правда, без полосы навигации) мы создали сайт фотогалереи «The Beatles» (см. *упражнения 23.7* и *23.8*);

- *двухколоночный* макет:



Этот макет реализовать сложнее, поскольку придется размещать два блочных элемента по горизонтали. Сделать это можно несколькими способами, описываемыми на этом уроке.

На основе двухколоночного макета мы сделали сайт суши-бара «Йокогама».

Существуют также и другие макеты, в частности, с тремя колонками. Однако они применяются довольно редко.

25.2. Двухколоночный макет на основе плавающего элемента

Вероятно, проще всего создать двухколоночный макет на основе плавающего элемента (см. *разд. 22.1*). Для этого нужно:

- превратить панель навигации в плавающий элемент, сдвинутый к левому краю страницы;
- указать у основного содержимого достаточный внешний просвет слева, чтобы оно не обтекало панель навигации, а полностью располагалось правее нее;

- если существуют элементы-соседи, которые должны выводиться строго под панелью навигации (например, поддон), — указать у них соответствующую настройку стиля (см. *разд. 22.1*).

Пример:

```
nav {
    float: left;
    width: 200px;
}
main { margin-left: 300px; }
footer { clear: left; }
```

На основе такого рода двухколоночного макета построены страницы сайта суши-бара. Кстати, приведенный пример почти полностью взят из CSS-кода этого сайта.

Преимущества и недостатки подобного макета:

Преимущество	Недостатки
Простота создания	• Возможные побочные эффекты (например, незапланированное обтекание плавающего элемента его соседями). • Плавающий элемент занимает лишь часть отведенного ему пространства родителя (и, таким образом, не является настоящей колонкой).

В частности, когда мы указали у панели навигации сайта суши-бара фон, этим фоном оказался закрашен лишь прямоугольный фрагмент пространства родителя. В нашем сайте такой эффект оказался вполне к месту, но это не значит, что он окажется к месту на других сайтах.

Двухколоночный макет с плавающим элементом вследствие своей простоты применяется до сих пор, но только при разработке сайтов с простым дизайном.

25.3. Упражнение. Табличный макет

Теория

> ***Табличный макет***
>
> Макет, сделанный на основе таблицы.

Таблица, формирующая такой макет, может быть создана как с помощью специализированных тегов: `<table>`, `<tr>` и `<td>`, так и на основе

элементов с измененным представлением (см. *разд. 21.1*). Второй вариант более приемлем, если нужно переделать под новый макет уже существующие страницы.

Преимущества и недостатки табличного макета:

Преимущество	Недостатки
Все элементы являются настоящими колонками (поэтому их можно без проблем, например, закрасить фонами).	• Для формирования таблицы требуется создавать дополнительные элементы (по крайней мере строки). • Нет возможности выполнить слияние ячеек (см. *разд. 8.2*).

Табличный макет был довольно популярен ранее, до появления более удобных инструментов (в частности, фиксированных элементов и сеточной разметки). В настоящее время его популярность несколько снизилась, однако он до сих пор применяется, поскольку довольно прост в реализации.

Практика

Создадим новую редакцию сайта суши-бара, страницы которого сделаем на основе табличного макета.

1. Найдем в папке 21\ex21.2 сопровождающего книгу файлового архива (см. *приложение 4*) папку site, скопируем ее куда-либо на локальный диск и дадим копии папки имя site-table.

   В таблицу мы превратим не все содержимое страницы, а лишь фрагмент, содержащий семантические панель навигации и основное содержимое. Так мы немного упростим себе работу.

   Оба этих элемента заключим в два вложенных друг в друга блока. У внешнего блока укажем стилевой класс `table`, а у внутреннего — `row`. Внешний блок чуть позже превратим в таблицу, а внутренний — в строку таблицы.
2. Откроем главную страницу index.html в текстовом редакторе и внесем в код следующие правки:

```
<div class="table">
    <div class="row">
        <nav>
            . . .
        </nav>
        <main>
            . . .
```

```
        </main>
    </div>
</div>
```

Помимо превращения внешнего блока в таблицу, необходимо указать, чтобы рамки проводились между ячейками этой таблицы (чтобы устранить нежелательные просветы между ними), и растянуть «таблицу» на всю ширину родителя.

3. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и напишем необходимые стили для элементов со стилевыми классами `table` и `row`:

```
.table {
    display: table;
    border-collapse: collapse;
    width: 100%;
}
.row { display: table-row; }
```

Теперь необходимо превратить семантические панель навигации и основное содержимое в «ячейки» только что созданной «таблицы».

4. Дополним стили, применяемые к семантическим панели навигации и основному содержимому:

```
nav {
    . . .
    display: table-cell;
}
main {
    . . .
    display: table-cell;
}
```

В стиле панели навигации присутствует код, превращающий ее в плавающий элемент, задающий внешние просветы и скругленные углы. А в стиле основного содержимого имеется атрибут стиля, задающий внешний просвет слева. Этот код более не нужен, и его следует удалить.

5. Удалим из упомянутых стилей ненужный код:

```
nav {
    background-color: blanchedalmond;
    float: left;
    width: 200px;
```

```
    margin: 10px 0px 10px 10px;
    padding: 20px;
    border-radius: 10px;
    display: table-cell;
}
main {
    margin-left: 300px;
    . . .
}
```

✓ Что у нас получилось? ⌄

Всё, как и мы планировали. Вот только текст семантического основного содержимого слева вплотную примыкает к границе. Нужно его отодвинуть, задав подходящее значение внутреннего просвета слева.

6. Добавим нужный код в стиль, применяемый к семантическому основному содержимому:

```
main {
    min-height: 600px;
    display: table-cell;
    padding-left: 20pt;
}
```

✓ Результат

7. Переделаем остальные страницы сайта самостоятельно.

25.4. Упражнение. Рамочный макет

Теория

> ***Рамочный макет***
>
> Макет, который основан на наборе элементов, располагаемых на странице произвольно. Некоторые из этих элементов могут быть превращены в прокручиваемые (см. *разд. 15.8*).

Рамочный макет реализуется на основе абсолютно позиционируемых (см. *разд. 22.2.1.1*), фиксированных элементов (см. *разд. 22.2.1.3*) или на сеточной разметке (см. *урок 24*).

Преимущества и недостатки рамочного макета:

Преимущества	Недостатки
• Элементы можно расположить на странице произвольно. • Как правило, не требуется создавать дополнительные элементы страниц.	• Требуется довольно сложный CSS-код. • Необходима точная подгонка элементов друг к другу.

Рамочный макет в настоящее время имеет довольно широкое распространение, поскольку относительно прост в разработке и позволяет создать практически любой дизайн страниц.

Практика

Сделаем еще одну редакцию сайта суши-бара — теперь с применением рамочного макета.

1. Найдем в папке 21\ex21.2 сопровождающего книгу файлового архива (см. *приложение 4*) папку site, скопируем ее куда-либо на локальный диск и дадим копии папки имя site-frames.

   Семантические шапку, панель навигации, основное содержимое и поддон мы позже превратим в абсолютно позиционируемые элементы и будем позиционировать их относительно ближайшего общего родителя — блока со стилевым классом `container`. Необходимо превратить этот блок в позиционируемый элемент (в противном случае все приведенные ранее элементы станут позиционироваться относительно секции тела, и результат окажется не таким, на какой мы рассчитывали).

   Мы превратим блок в относительно позиционируемый элемент, не указывая его координаты. В результате блок станет позиционируемым

элементом и не сдвинется со своего места, что нам на руку. Сделаем это прямо сейчас.

2. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и добавим нужный код в стиль, применяемый к блоку со стилевым классом `container`:

```
div.container {
    . . .
    position: relative;
}
```

Теперь превратим остальные элементы в абсолютно позиционируемые.

3. Добавим стиль для семантических шапки, панели навигации, основного содержимого и поддона:

```
header, nav, main, footer { position: absolute; }
```

Здесь мы столкнемся с проблемой. Если какой-либо элемент содержит только позиционируемые потомки, его высота станет равной нулю. В результате мы не сможем позиционировать потомки по нижней границе родителя и лишимся указанного у него фона.

У нас элементов, включающих только позиционируемые потомки, два:

- блок со стилевым классом `container` — охватывает семантические шапку, панель навигации, основное содержимое и поддон — позиционируемые элементы;
- секция тела страницы — содержит блок со стилевым классом `container` — также позиционируемый.

Необходимо растянуть секцию тела на всю область просмотра, а блок со стилевым классом `container` — на всю высоту секции тела (по ширине его растягивать необязательно, так как по умолчанию его ширина равна ширине родителя).

4. Внесем необходимые правки в стили секции тела и блока со стилевым классом `container`:

```
body {
    . . .
    width: 100vw;
    height: 100vh;
}
div.container {
    . . .
    height: 100%;
}
```

Для задания размеров секции тела мы применили единицы измерения vw и vh (см. *разд. 12.2.1.2.1*), а не проценты. Размеры в процентах рассчитываются от соответствующих размеров видимого родителя, какового у секции тела нет.

Решив проблему, зададим местоположение элементов. Значения координат подберем опытным путем.

5. Исправим стиль семантической шапки:

```
header { text-align: right; }
header {
    text-align: right;
    left: 0;
    top: 0;
    right: 0;
}
```

В стилях панели навигации и основного содержимого присутствует лишний CSS-код. Заодно удалим его.

6. Переделаем стиль панели навигации:

```
nav {
    background-color: blanchedalmond;
    float: left;
    width: 200px;
    margin: 10px 0px 10px 10px;
    padding: 20px;
    border-radius: 10px;
    left: 10pt;
    top: 150pt;
    bottom: 100pt;
}
```

Основное содержимое превратим в прокручиваемый элемент.

7. Изменим стиль основного содержимого:

```
main {
    margin-left: 300px;
    min-height: 600px;
    left: 240pt;
    top: 150pt;
    right: 0;
    bottom: 100pt;
    overflow-y: auto;
}
```

Атрибут стиля, задающий минимальную высоту, также лучше убрать, поскольку при уменьшении высоты окна веб-обозревателя он вызовет появление в окне полос прокрутки, что исказит разметку.

8. Исправим стиль поддона:

```
footer { text-align: right; }
footer {
    text-align: right;
    left: 0pt;
    right: 0pt;
    bottom: 0pt;
}
```

✓ Результат

Одно из преимуществ рамочного макета состоит в том, что для его реализации практически никогда не требуется создавать новые элементы страниц. Так, нам не пришлось переделывать ни одну страницу сайта.

25.5. Самостоятельные упражнения

- Создайте третью редакцию сайта суши-бара, основанную на рамочном макете, реализованном с применением сеточной разметки. Подберите параметры сетки самостоятельно. Сохраните ее в папке site-grid.

  На страницах первой редакции сайта суши-бара — ниже панели навигации — выводилось изображение чашки мате. Оно выводится до сих пор, просто не показывается на экране — перекрывается сплошным непрозрачным фоном панели навигации.

- Отобразите изображение чашки мате на страницах всех трех редакций сайта суши-бара. Поместите ее в составе панели навигации, ниже гиперссылок.

✓ У вас должно получиться ⌄

Урок 26. Форма элементов

Элементы со скругленными углами.
Элементы произвольной формы.
Лоскутное одеяло.
Форма обтекания.

CSS позволяет создавать элементы страниц формы, отличной от строго прямоугольной.

ВНИМАНИЕ!
Все описанные здесь атрибуты стилей являются ненаследуемыми.

26.1. Элементы со скругленными углами

Проще всего создать прямоугольный элемент со скругленными углами.

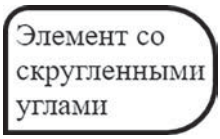
Для указания радиусов кривизны у углов элемента применяются следующие атрибуты стиля:

- `border-top-right-radius` — радиус скругления для правого верхнего угла;
- `border-bottom-right-radius` — для правого нижнего угла;
- `border-bottom-left-radius` — для левого нижнего угла;
- `border-top-left-radius` — для левого верхнего угла.

В качестве значения у этих атрибутов стиля можно указать:

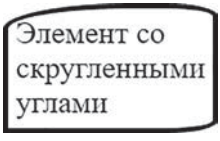
- числовую величину в какой-либо единице измерения — задаст радиус скругления и по горизонтали, и по вертикали.

Пример	Результат

`.rounded1 {`
`  border-top-right-radius: 20pt;`
`  border-bottom-right-radius: 20pt;`
`  border-bottom-left-radius: 0pt;`
`  border-top-left-radius: 10pt;`
`}`
	

- две числовые величины в какой-либо единице измерения, разделенные пробелом: первая задаст радиус скругления по горизонтали, вторая — по вертикали.

Пример	Результат

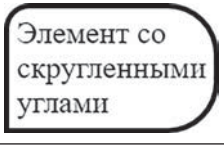
`.rounded2 {`
`  border-top-right-radius: 20pt 5pt;`
`  border-bottom-right-radius:`
`  20pt 5pt;`
`  border-bottom-left-radius: 0pt;`
`  border-top-left-radius: 50pt 10pt;`
`}`
	

Значение по умолчанию: `0`.

Если у рамки нужно скруглить все углы, применять рассмотренные атрибуты стиля неудобно, поскольку CSS-код получается слишком громоздким. Лучшим выбором будет атрибут стиля `border-radius`, задающий радиус скругления сразу у всех углов рамки. В качестве его значений можно указать четыре, три, две или одну величину, разделив их пробелами:

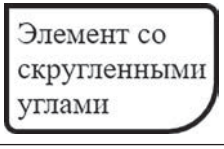
- *<левый верхний угол> <правый верхний угол> <правый нижний угол> <левый нижний угол>*.

Пример	Результат

`.rounded3 {`
`  border-radius: 10pt 20pt 20pt 0pt;`
`}`
	

- *<левый верхний угол> <правый верхний и левый нижний угол> <правый нижний угол>*.

Пример	Результат

`.rounded4 {`
`  border-radius: 0pt 5pt 20pt;`
`}`
	

- *<левый верхний и правый нижний угол>* *<правый верхний и левый нижний угол>*;
- *<все углы>*.

Пример	Результат

`.rounded5 {`
`  border: medium solid black;`
`  border-radius: 5pt;`
`}`
	Элемент со скругленными углами

26.2. Элементы произвольной формы

Одно из последних нововведений в CSS — набор инструментов для создания элементов произвольной, в том числе и далекой от прямоугольной, формы.

ВНИМАНИЕ!

Описываемые здесь инструменты не изменяют форму элемента страницы, а лишь определяют часть элемента, которая должна выводиться на экране. Соответственно, содержимое элемента, не попавшее в эту часть, будет скрыто.

Для указания формы элемента применяется атрибут стиля `clip-path`. В качестве его значения можно указать:

- `[`*<точка позиционирования фигуры>*`]` *<фигура, описывающая форму>*
  Задает форму элемента, описываемую заданной *фигурой*;
- `none` — элемент будет иметь прямоугольную форму.

Значение по умолчанию: `none`.

26.2.1. Описание фигур, задающих форму

Для описания фигур, задающих форму элементов, применяется ряд функций CSS, рассматриваемых далее.

26.2.1.1. Описание простых фигур

Для создания простых фигур применяются следующие функции:

- `inset()` — прямоугольник. Записывается в формате:
  `inset(`*<местоположения сторон>* `[round` *<радиусы скругления углов>*`])`
  Радиусы скругления углов записываются в том же формате, что применяется в атрибуте стиля `border-radius` (см. *разд. 26.1*).

Местоположения сторон записываются в виде расстояний между сторонами прямоугольника и соответствующими сторонами элемента (местоположение верхней границы прямоугольника отсчитывается от верхней границы элемента, местоположение нижней границы прямоугольника — от нижней границы элемента), выраженных в любой из поддерживаемых единиц измерения. Можно указать четыре, три, два значения или одно значение:

- *<сверху> <справа> <снизу> <слева>*;
- *<сверху> <справа и слева> <снизу>*;
- *<сверху и снизу> <справа и слева>*;
- *<сверху, справа, снизу и слева>*.

Пример	Результат[1]

`.clip1 {`
`  padding: 30px;`
`  clip-path: inset(50px 40px 20px 30px);`
`}`
	

`.clip2 {`
`  padding: 30px;`
`  clip-path: inset(10px 50px round 0 10px 10px 0);`
`}`
	

♦ `rect()` — то же, что и `inset()`, только местоположения всех сторон прямоугольника отсчитываются от левой и верхней границ элемента. Если в качестве какого-либо из местоположений указать слово `auto`, соответствующая граница пройдет прямо по границе элемента. Допускается указание только четырех значений.

Пример	Результат

`.clip3 {`
`  padding: 30px;`
`  clip-path: rect(50px 140px 100px 30px);`
`}`
	

[1] Здесь и далее черным цветом закрашена часть элемента, которая будет видима после указания соответствующей формы.

Пример	Результат

`.clip4 {`
`  padding: 30px;`
`  clip-path: rect(10px 130px 110px 50px`
`  round 0 10px 10px 0);`
`}`
	

`.clip5 {`
`  padding: 30px;`
`  clip-path: rect(30px auto 90px auto);`
`}`
	

- `xywh()` — прямоугольник:

`xywh(`*<координата левой стороны>* *<координата верхней стороны>* *<ширина>* *<высота>* `[round` *<радиусы скругления углов>*`])`

Координата левой стороны прямоугольника отсчитывается от левой стороны элемента, *координата верхней стороны* прямоугольника — от верхней стороны элемента. Все значения задаются в виде числовых величин в какой-либо из поддерживаемых единиц измерения.

Пример	Результат

`.clip6 > div:last-child {`
`  padding: 30px;`
`  clip-path: xywh(30px 50px 110px 50px);`
`}`
	

`.clip7 > div:last-child {`
`  padding: 30px;`
`  clip-path: xywh(50px 10px 80px 100px`
`  round 0 10px 10px 0);`
`}`
	

- `circle(<радиус> [ at <местоположение центра>])` — круг.

  Радиус круга можно задать в виде:

  - числовой величины в любой единице измерения;
  - значений `closest-side` или `farthest-side`, поддерживаемых функцией `radial-gradient()` (см. *разд. 19.2.2*).

  Местоположение центра круга можно задать в виде:

  - пары числовых величин в любой единице измерения, разделенных пробелом, — зададут, соответственно, горизонтальную и вертикальную координату. Отсчет ведется от левого верхнего угла элемента, соответственно, вправо и вниз;
  - пары предопределенных значений, поддерживаемых атрибутом стиля `object-position` (см. *разд. 17.3*).

  Если местоположение не указано, оно принимается равным `50% 50%` (центр элемента).

Пример	Результат

`.clip8 {`
`  padding: 30px;`
`  clip-path: circle(80px at 70px 90px);`
`}`
	

`.clip9 {`
`  padding: 30px;`
`  clip-path: circle(160px at right center);`
`}`
	

`.clip10 {`
`  padding: 30px;`
`  clip-path: circle(50px);`
`}`
	

- `ellipse()` — эллипс:

  `ellipse(<радиус по горизонтали> <радиус по вертикали>`
  `[at <местоположение центра>])`

  Оба *радиуса* и *местоположение центра* задаются так же, как и у функции `circle()`.

Пример	Результат

`.clip11 {`
`  padding: 30px;`
`  clip-path: ellipse(60px 160px at 70px 90px);`
`}`
	

`.clip12 {`
`  padding: 30px;`
`  clip-path: ellipse(160px 60px at right center);`
`}`
	

`.clip13 {`
`  padding: 30px;`
`  clip-path: ellipse(90px 40px);`
`}`
	

- `polygon()` — многоугольник:

```
polygon([<правило определения местоположения точек>,]
        <гор. коорд. точки 1> <верт. коорд. точки 1>,
        <гор. коорд. точки 2> <верт. коорд. точки 2>,
        . . .,
        <гор. коорд. точки N> <верт. коорд. точки N>)
```

Координаты точек задаются в виде числовых величин в любой поддерживаемой единице измерения. Отсчет ведется от левого верхнего угла элемента вправо и вниз.

Начальная и конечная точки многоугольника должны либо совпадать друг с другом, либо находиться на какой-либо из границ элемента.

Пример	Результат

`.clip14 {`
`  padding: 30px;`
`  clip-path: polygon(0 50%, 30% 0, 100% 50%, 30% 100%, 0 50%);`
`}`
	

`.clip15 {`
`  padding: 30px;`
`  clip-path: polygon(0 0, 60px 10px, 160px 60px, 60px 110px, 0 120px);`
`}`
	

Правило определения местоположения точки указывает веб-обозревателю, какие точки многоугольника следует считать принадлежащими ему, а какие — находящимися вне его. *Правило* указывается в виде одного из следующих слов:

- `nonzero` — из точки, местоположение которой следует выяснить, проводится бесконечная прямая в произвольном направлении, и ведется подсчет отрезков многоугольника, пересекающих эту прямую. Подсчет начинается с `0`. Если очередной отрезок, пересекший линию, проведен в направлении слева направо, к количеству отрезков добавляется 1, если отрезок проведен справа налево — вычитается 1. Если результирующее количество отрезков получилось отличным от `0`, точка принадлежит фигуре, если количество нулевое — не принадлежит;
- `evenodd` — то же самое, что и `nonzero`, только точка считается принадлежащей фигуре, если получившееся количество отрезков является нечетным, и находящейся вне фигуры — если количество четное.

Если *правило* не указано, оно принимается равным `nonzero`.

Пример	Результат

`.figure1 {`
`  clip-path: polygon(50% 0, 70% 100%, 0 30%,`
`  100% 30%, 30% 100%,`
`  50% 0);`
`}`
	

`.figure2 {`
`  clip-path: polygon(evenodd, 50% 0, 70% 100%,`
`  0 30%, 100% 30%,`
`  30% 100%, 50% 0);`
`}`
	

26.2.1.2. Описание сложных фигур. Пути

Элементу можно придать форму, описываемую более сложной фигурой. Такая фигура представляется в виде *пути*.

> ***Путь***
>
> Контур сложной фигуры, составляемый из множества отрезков различной формы (*примитивов*).

Для описания пути применяется функция `path()`:

`path([<правило определения местоположения точек>,] <описание пути>)`

Правило определения местоположения точек указывается в том же формате, что и у функции `polygon()` (см. *разд. 26.2.1.1*).

Описание пути задается в виде строкового значения (см. *разд. 12.2.4*). Оно представляет собой последовательность из описаний отдельных примитивов, составляющих путь, которые разделяются пробелами.

Каждое из описаний примитива начинается с буквы латиницы, играющей роль команды на начало рисования. За командой идут числовые параметры рисуемого примитива (в частности, координаты его точек), которые записываются *без единиц измерения* и *всегда выражаются в пикселах*.

Чтобы точно указать координаты точек примитивов создаваемого пути, желательно знать размеры элемента. Поэтому имеет смысл указывать размеры элемента явно.

Путь, примитив за примитивом, рисуется виртуальным пером. Перо также можно перемещать с места на место, чтобы нарисовать несвязанные фрагменты пути. Рисование следующего примитива начинается в той точке, в которой закончилось рисование предыдущего.

Для описания перемещений пера применяются специальные команды, записываемые в составе *описания пути*:

- `M <X> <Y>` — перемещает перо в точку с координатами [*X*, *Y*]. Координаты отсчитываются от левого верхнего угла элемента. Положительные значения координат отсчитываются в направлениях вправо и вниз, отрицательные — влево и вверх;
- `m <dX> <dY>` — смещает перо на расстояния *dX* по горизонтали и *dY* по вертикали. Положительные величины вызывают смещение вправо и вниз, отрицательные — влево и вверх.

Рисование различных примитивов обеспечивают следующие команды, записываемые в составе *описания пути*:

- `L <X> <Y>` — прямая линия от текущего положения пера до точки с координатами [*X*, *Y*];
- `l <dX> <dY>` — прямая линия от текущего положения пера до точки, отстоящей от текущего положения на величины [*dX*, *dY*];
- `H <X>` — горизонтальная прямая линия от текущего положения пера до точки с горизонтальной координатой *X* (вертикальная координата останется той же);
- `h <dX>` — горизонтальная прямая линия от текущего положения пера до точки, отстоящей по горизонтали на величину *dX*;

- `V <Y>` — вертикальная прямая линия от текущего положения пера до точки с вертикальной координатой *Y* (горизонтальная координата останется той же);
- `v <dY>` — вертикальная прямая линия от текущего положения пера до точки, отстоящей по вертикали на величину *dY*.

Пример	Результат

`.clip16 {`
`  padding: 30px;`
`  clip-path: path('M 20 20 H 100 L 170 100 ↳`
`  H 40 L 20 70 V 20');`
`}`
	

`.clip17 {`
`  padding: 30px;`
`  clip-path: path('M 20 20 H 80 V 100 h -60 ↳`
`  v -80 M 100 20 H 160 ↳`
`  V 100 h -60 v -80');`
`}`
	

Последний пример показывает, что путь может содержать не связанные между собой фрагменты;

- `A` — эллиптическая дуга от текущего положения пера до точки с координатами [*X*, *Y*], с радиусами по горизонтали *Rx* и вертикали *Ry* и углом поворота *A*:

  `A <Rx> <Ry> <A> <большая дуга?> <направление> <X> <Y>`

  Угол *A* указывается в градусах и отсчитывается от горизонтальной оси. Положительные значения угла вызывают поворот по часовой стрелке, отрицательные — против часовой стрелки.

Пример	Результат

`.clip18 {`
`  padding: 30px;`
`  clip-path: path('M 30 90 A 90 50 0 1 1 ↳`
`  150 90 L 30 90');`
`}`
	

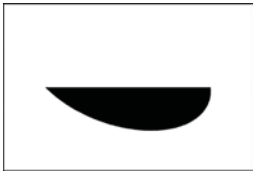
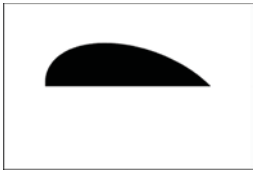
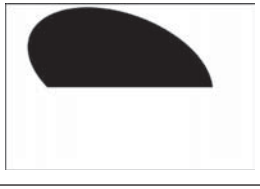
`.clip19 {`
`  padding: 30px;`
`  clip-path: path('M 30 90 A 90 50 30 1 1 ↳`
`  150 90 L 30 90');`
`}`
	

Параметр *большая дуга?* указывает величину проводимой дуги:

- 0 — будет проведена маленькая дуга;
- 1 — будет проведена большая дуга.

Параметр *направление* задает направление, в котором будет проводиться дуга:

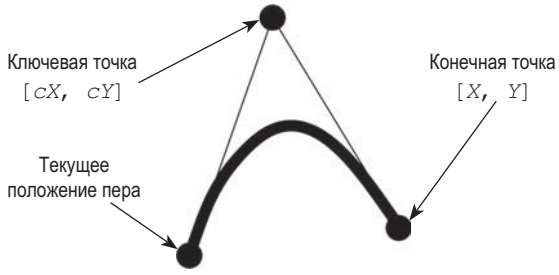
- 0 — против часовой стрелки;
- 1 — по часовой стрелке.

Пример	Результат
`.figure3 { clip-path: path('M 30 60 A 70 40 20 0 0 150 60 L 30 60'); }`	
`.figure4 { clip-path: path('M 30 60 A 70 40 20 1 0 150 60 L 30 60'); }`	
`.figure5 { clip-path: path('M 30 60 A 70 40 20 0 1 150 60 L 30 60'); }`	
`.figure6 { clip-path: path('M 30 60 A 70 40 20 1 1 150 60 L 30 60'); }`	

- a — то же самое, что и A, только дуга проводится до точки, отстоящей от текущего положения пера на величины [*dX*, *dY*]:
  a <*Rx*> <*Ry*> <*A*> <*большая дуга?*> <*направление*> <*dX*> <*dY*>
- Q — *квадратическая кривая Безье*.

> ***Кривая Безье***
>
> Кривая, описываемая начальной, конечной и одной (*квадратическая кривая Безье*) или двумя ключевыми (*кубическая кривая Безье*) точками.

Квадратическая кривая Безье (с одной ключевой точкой):



Формат записи команды:

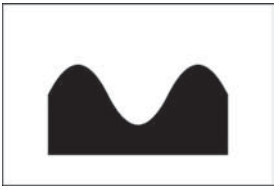
`Q <cX> <cY> <X> <Y>`

Пример	Результат

`.clip20 {`
`  padding: 30px;`
`  clip-path: path('M 20 20 Q 280 60 20 100 ↳`
`  L 20 20');`
`}`
	

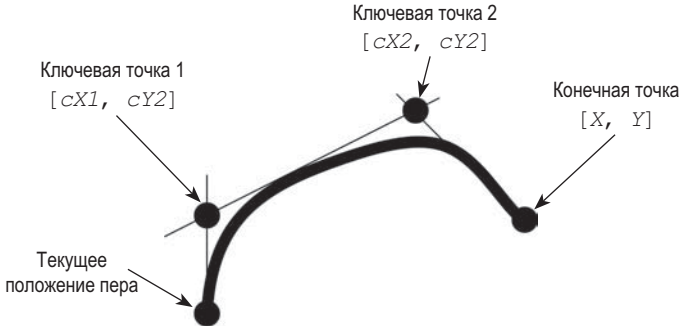
- `q <dcX> <dcY> <dX> <dY>` — то же самое, что и `Q`, только ключевая точка кривой располагается со смещением на величины [*dcX*, *dcY*] от текущего положения пера, а конечная точка — со смещением на величины [*dX*, *dY*] от него;
- `T <X> <Y>` — квадратичная кривая Безье с конечной точкой [*X*, *Y*]. Ключевая точка этой кривой является зеркальным отражением ключевой точки квадратичной кривой, проведенной ранее, относительно текущего положения пера. Если ранее проведенный отрезок не является квадратичной кривой Безье, ключевая точка совпадает с текущим положением пера.

Пример	Результат

`.figure7 {`
`  clip-path: path('M 30 60 Q 50 20 70 60 ↳`
`  T 110 60 T 150 60 V 100 ↳`
`  H 30 V 60');`
`}`
	

- `t <dX> <dY>` — то же самое, что и `T`, только конечная точка кривой отстоит на величины [*dX*, *dY*] от текущего положения пера;

- `C` — кубическая кривая Безье:



Формат записи команды:

`C <cX1> <cY1> <cX2> <cY2> <X> <Y>`

Пример	Результат

`.clip21 {`
`  padding: 30px;`
`  clip-path: path('M 20 20 C 180 0 180 120 ↪`
`  20 100 L 20 20');`
`}`
	

- `c <dcX1> <dcY1> <dcX2> <dcY2> <dX> <dY>` — то же самое, что и `C`, только ключевые точки кривой располагаются со смещением на величины, соответственно, [*dcX1*, *dcY1*] и [*dcX2*, *dcY2*] от текущего положения пера, а конечная точка — со смещением на величины [*dX*, *dY*] от него;
- `S <cX2> <cY2> <X> <Y>` — кубическая кривая Безье со второй ключевой точкой [*cX2*, *cY2*] и конечной точкой [*X*, *Y*]. Первая ключевая точка этой кривой является зеркальным отражением второй ключевой точки кубической кривой, проведенной ранее, относительно текущего положения пера. Если ранее проведенный отрезок не является кубической кривой Безье, первая ключевая точка совпадает с текущим положением пера;
- `s <dcX2> <dcY2> <dX> <dY>` — то же самое, что и `S`, только вторая ключевая и конечная точки кривой отстоят на величины, соответственно, [*dcX2*, *dcY2*] и [*dX*, *dY*] от текущего положения пера.

Если требуется нарисовать подряд несколько примитивов одинакового типа (например, прямых линий), то в командах рисования второго и последующих примитивов буквенные обозначения можно не указывать — это немного сократит код.

Пример	Результат

`.clip22 {`
`  padding: 30px;`
`  clip-path: path('M 30 20 L 160 40 L 150 90 ↳`
`  L 10 100 L 30 20');`
`}`
	

`.clip22 {`
`  padding: 30px;`
`  clip-path: path('M 30 20 160 40 150 90 10 100 ↳`
`  30 20');`
`}`	

Команда `Z` или `z` замыкает создаваемый путь посредством проведения прямой линии от конечной точки последнего примитива в начальную точку первого примитива.

Пример	Результат

`.clip22 {`
`  padding: 30px;`
`  clip-path: path('M 30 20 160 40 150 90 10 100 Z');`
`}`
	

26.2.2. Задание точки позиционирования фигуры

Точка позиционирования фигуры, задаваемая в атрибуте стиля `clip-path`, указывает, относительно какой точки элемента будет позиционироваться фигура, определяющая форму элемента. В качестве этой *точки* можно указать:

- `margin-box` — фигура позиционируется относительно левого верхнего угла воображаемого прямоугольника, образованного внешними просветами.

Пример	Результат

`.clip23 {`
`  margin: 20px;`
`  padding: 20px;`
`  border: 10px dotted black;`
`  clip-path: margin-box xywh(0 0 110px 70px);`
`}`
	

- `border-box` — относительно левого верхнего угла воображаемого прямоугольника, образованного рамкой.

Пример	Результат

`.clip24 {`
`  margin: 20px;`
`  padding: 20px;`
`  border: 10px dotted black;`
`  clip-path: border-box xywh(0 0 110px 70px);`
`}`
	

- `padding-box` — относительно левого верхнего угла воображаемого прямоугольника, образованного внутренними просветами.

Пример	Результат

`.clip25 {`
`  margin: 20px;`
`  padding: 20px;`
`  border: 10px dotted black;`
`  clip-path: padding-box xywh(0 0 110px 70px);`
`}`
	

- `content-box` — относительно левого верхнего угла содержимого элемента.

Пример	Результат

`.clip26 {`
`  margin: 20px;`
`  padding: 20px;`
`  border: 10px dotted black;`
`  clip-path: content-box xywh(0 0 110px 70px);`
`}`
	

Если точка позиционирования фигуры не указана, она принимается равной `border-box`.

26.3. Упражнение. Лоскутное одеяло

> ***Лоскутное одеяло***
>
> Дизайн веб-страниц, при котором для размещения фрагментов содержания и оформления применяются элементы непрямоугольной формы, подогнанные друг к другу.

Слава о нас как о знаменитых веб-верстальщиках постепенно распространяется по свету... Вот и еще один знакомый прослышал о наших успехах в области веб-верстки и заказал разработку простого рекламного сайта цветной бумаги для аппликаций, состоящего из одной страницы. И попросил выполнить этот сайт в виде лоскутного одеяла.

Конечный результат должен быть таким:

1. Создадим где-либо папку patch-blanket, в которой будут храниться файлы создаваемого сайта.

   Дальнейшая работа будет протекать в этой папке.

2. Создадим главную страницу index.html и запишем в него начальный HTML-код:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="UTF-8">
        <title>Цветные лоскутки: Бумага для аппликаций</title>
        <link rel="stylesheet" href="styles/main.css">
    </head>
```

```
    <body>
    </body>
</html>
```

3. Создадим папку styles, а в ней — таблицу стилей main.css.

   На приведенном рисунке видно, что все содержание страницы помещено в элемент, который находится в центре страницы. Этот элемент имеет размеры 1000 × 600 пикселов и тонкую черную рамку.

   Сама страница заполнена сплошным фоном цвета `beige`.

   Сначала напишем стиль для секции тела страницы. Он уберет просветы между границами области просмотра и содержанием страницы, задаст флекс-разметку, определит выравнивание единственного потомка по центру, растянет секцию тела на всю область просмотра (иначе потомок не будет выровнен по центру) и установит цвет фона `beige`.

4. Запишем в таблицу стилей styles\main.css стиль для секции тела:

```
body {
    margin: 0;
    width: 100vw;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: beige;
}
```

   «Всеобъемлющий» элемент мы создадим из семантического основного содержимого (тега `<main>`) со стилевым классом `container`. В него поместим четыре пустых блока (теги `<div>`), которые сформируют четыре «лоскутка», находящихся на заднем плане. Укажем у этих блоков стилевые классы `patch1`, `patch2`, `patch3` и `patch4`. Шапку с заголовками создадим потом.

5. Добавим в секцию тега главной страницы `index.html` необходимый код:

```
<body>
    <main class="container">
        <div class="patch1"></div>
        <div class="patch2"></div>
        <div class="patch3"></div>
        <div class="patch4"></div>
    </main>
</body>
```

У семантического основного содержимого зададим размеры 1000 × 600 пикселов и тонкую черную рамку.

Блоки-«лоскутки» мы будем позиционировать абсолютно — это наиболее простой способ точно подогнать их друг к другу. Чтобы они без проблем позиционировались относительно семантического основного содержимого, превратим последнее в относительно позиционируемый элемент.

6. Добавим в таблицу стилей styles\main.css стиль для элемента со стилевым классом `container`:

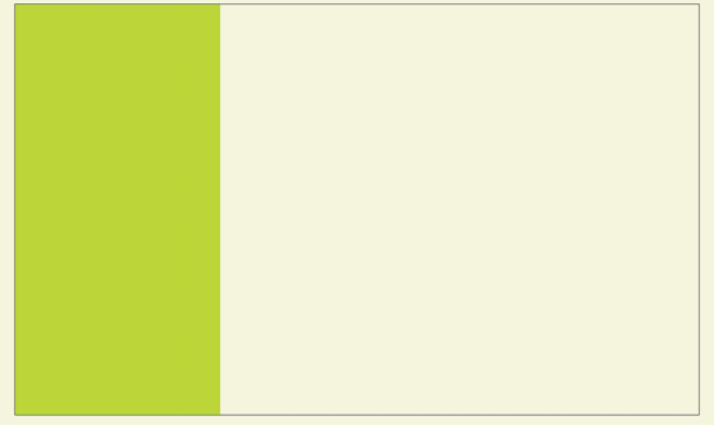
```
.container {
    width: 1000px;
    height: 600px;
    border: thin solid black;
    position: relative;
}
```

Первый блок-«лоскуток» со стилевым классом `patch1` поместим в левой части «всеобъемлющего» элемента, укажем у него ширину в 300 пикселов и закрасим его симпатичным цветом `greenyellow`.

7. Введем стиль для элемента со стилевым классом `patch1`:

```
.patch1 {
    position: absolute;
    left: 0;
    top: 0;
    bottom: 0;
    width: 300px;
    background-color: greenyellow;
}
```

✓ Что у нас получилось? ⌄



Теперь зададим его форму, основываясь на приведенном в начале упражнения рисунке (параметры фигур, описывающих форму этого и последующих элементов, подобраны автором экспериментально).

8. Дополним ранее написанный стиль, применяемый к элементу со стилевым классом `patch1`:

```
.patch1 {
    . . .
    clip-path: polygon(0 0, 100% 0, 0 100%, 0 0);
}
```

✓ Что у нас получилось? ⌄

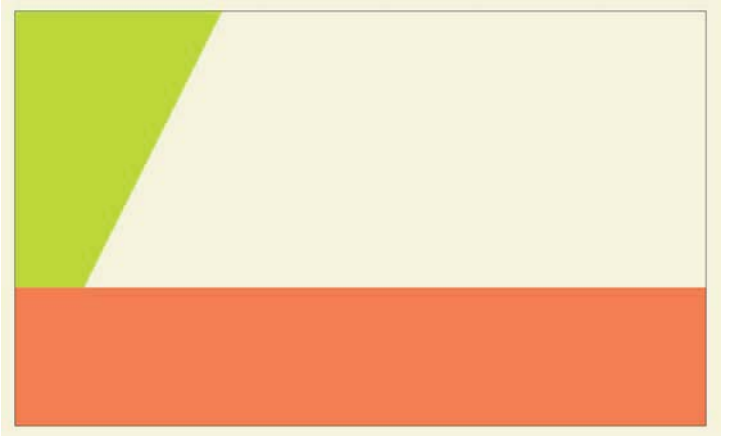
Мы использовали инструменты для создания простых фигур, описанные в *разд. 26.2.1.1*, поскольку сейчас требуется создать простую фигуру. Да и работать с этими инструментами проще.

Создадим следующий «лоскут». Поместим его внизу, укажем высоту в 200 пикселов и красивый цвет `coral`.

9. Запишем стиль для элемента со стилевым классом `patch2`:

```
.patch2 {
    position: absolute;
    left: 0;
    right: 0;
    bottom: 0;
    height: 200px;
    background-color: coral;
}
```

✓ Что у нас получилось? ⌄

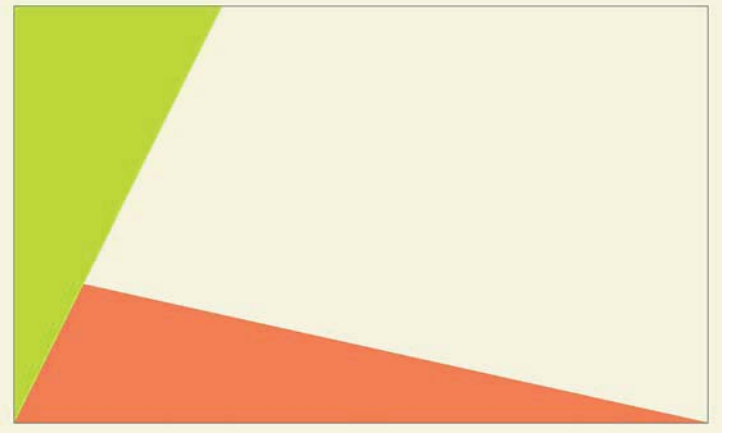


«Люблю резать!» — говорил доктор Уолтер Бишоп, герой сериала «За гранью[2]». Мы тоже любим это дело. Так что сразу же соответственно обрежем этот элемент, превратив его во второй «лоскут» и подогнав к уже имеющемуся.

10. Добавим необходимый код к стилю элемента со стилевым классом `patch2`:

```
.patch2 {
    . . .
    clip-path: polygon(0 100%, 100px 0, 100% 100%, 0 100%);
}
```

Лоскутное одеяло постепенно «шьется»…



[2] Оригинальное название — «Fringe». Также известен как «Грань».

Третий «лоскуток» — самый сложный. Его нужно подогнать к первому и второму. Расположим его вплотную к верхнему и нижнему краям родителя на расстоянии 100 пикселов от левого края и укажем ширину 600 пикселов. Закрасим его цветом `gold`.

11. Занесем стиль для элемента со стилевым классом `patch3`:

```
.patch3 {
    position: absolute;
    left: 100px;
    top: 0;
    bottom: 0;
    width: 600px;
    background-color: gold;
}
```

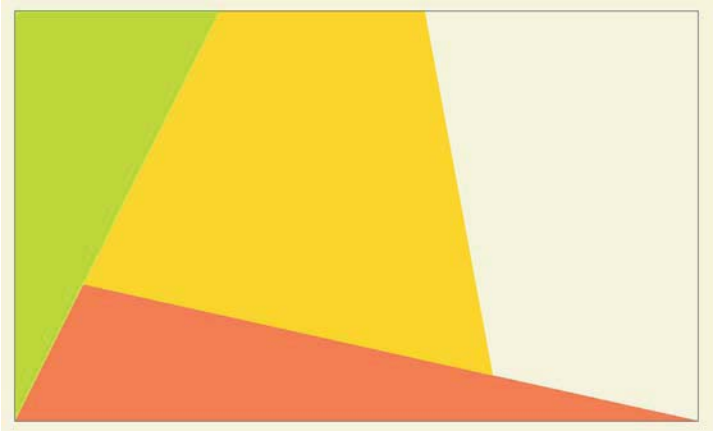
✓ Что у нас получилось? ⌄

Чтобы придать этому элементу нужную форму и подогнать его к уже имеющимся «лоскуткам», придется постараться.

12. Исправим стиль элемента со стилевым классом `patch3`:

```
.patch3 {
    . . .
    clip-path: polygon(200px 0, 0 400px, 100% 534px, 500px 0,
                                                    200px 0);
}
```

✓ Что у нас получилось? ⌄

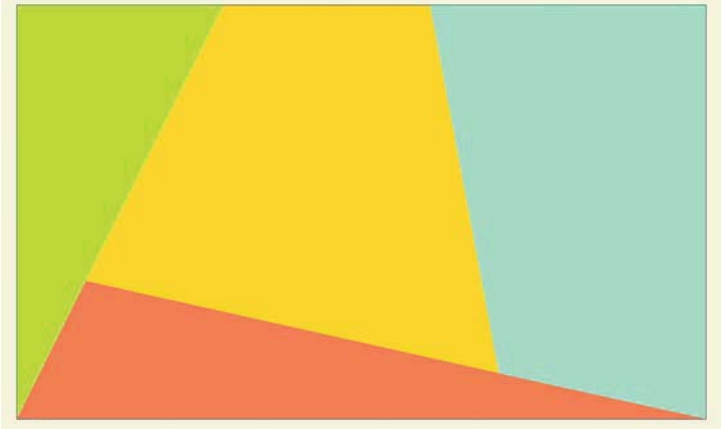


Осталось поместить четвертый «лоскуток» в оставшееся свободным место. Закрасим его цветом `aquamarine`.

13. Добавим стиль для элемента со стилевым классом `patch4`:

```
.patch4 {
    position: absolute;
    top: 0;
    bottom: 0;
    right: 0;
    width: 400px;
    background-color: aquamarine;
    clip-path: polygon(0 0, 100% 0, 100% 100%, 100px 534px, 0 0);
}
```

✓ Что у нас получилось? ⌄



Настала пора заняться шапкой сайта. Оформим ее в виде семантической шапки (тег `<header>`) с двумя заголовками: первого уровня — с текстом «Цветные лоскутки», и второго уровня — с текстом «Бумага для аппликаций».

14. Добавим в секцию тела главной страницы index.html код, создающий шапку:

```
<body>
    <main class="container">
        . . .
        <header>
            <h1>Цветные лоскутки</h1>
            <h2>Бумага для аппликаций</h2>
        </header>
    </main>
</body>
```

Шапку позиционируем в правом верхнем углу, укажем у нее достаточные внутренние просветы и закрасим слегка прозрачным белым фоном. Поскольку форма у шапки будет сложной, чтобы ее задать, мы используем инструменты для создания сложных фигур (см. *разд. 26.2.1.2*). Применять такие инструменты удобнее, если у элемента явно указаны размеры, поэтому мы зададим у шапки ширину и высоту. Величину просветов, координаты и размеры шапки, равно как и параметры фигуры, задающей ее форму, подберем экспериментально.

При описании фигуры, задающей форму элемента, следует учесть, что по умолчанию атрибуты стиля «семейств» `width` и `height` задают ширину и высоту содержимого элемента. Фактические размеры нашей шапки будут больше за счет указанных у них внутренних просветов.

15. Запишем необходимый стиль для семантической шапки:

```
header {
    position: absolute;
    top: 60px;
    right: 60px;
    width: 620px;
    height: 200px;
    background-color: rgba(255, 255, 255, 0.8);
    padding: 30pt;
    clip-path: path('M 30 0 L 100 30 200 0 400 40 C 800 0 800 260 ↳
                    400 240 L 300 220 100 230 30 230 50 150 Z');
}
```

Сразу же оформим оба заголовка, присутствующие в семантической шапке. Текст первого заголовка выведем полужирным шрифтом без засечек с кеглем 48 пунктов, — текст второго заголовка — полужирным курсивным шрифтом без засечек с кеглем 36 пунктов. У обоих заголовков уберем внешние просветы и установим внутренние просветы равными 10 пунктам.

16. Добавим стили для заголовков первого и второго уровня, вложенных в семантическую шапку:

```
header h1 {
    font: bold 48pt sans-serif;
    margin: 0;
    padding: 10pt;
}
header h2 {
    font: bold italic 36pt sans-serif;
    margin: 0;
    padding: 10pt;
}
```

После этого сайт станет выглядеть так, как показано на рисунке в начале этого упражнения.

26.4. Форма обтекания элемента

Любой блочный элемент страницы можно сделать плавающим, в результате чего он сдвинется к указанному краю родителя, а остальные элементы-соседи будут обтекать его с противоположной стороны (см. *разд. 22.1*).

По умолчанию *форма обтекания* элемента описывается воображаемым прямоугольником, образованным внешними просветами.

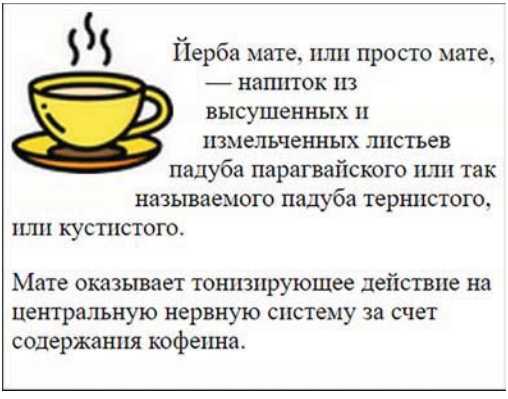
> ***Форма обтекания***
>
> Форма, по которой элементы-соседи обтекают плавающий элемент.

Однако можно определить другую форму обтекания, необязательно прямоугольную.

Форма обтекания задается атрибутом стиля `shape-outside`. В качестве его значения можно указать:

- простую фигуру, описанную функцией `inset()`, `circle()`, `ellipse()` или `polygon()` (см. *разд. 26.2.1.1*). Перед этой функцией можно указать точку позиционирования фигуры.

Пример	Результат

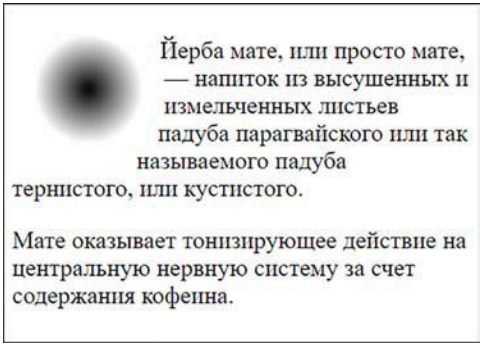
`aside.shos1 {`
`  float: left;`
`  width: 100px;`
`  height: 100px;`
`  margin: 0 20px 20px 0;`
`  shape-outside:`
`  polygon(50% 0, 130px 50%,`
`  50% 100%);`
`}`
	

`aside.shos2 {`
`  float: left;`
`  width: 100px;`
`  height: 100px;`
`  padding: 0 20px 20px 0;`
`  shape-outside:`
`  padding-box`
`  polygon(50% 0, 130px 50%,`
`  50% 100%);`
`}`	

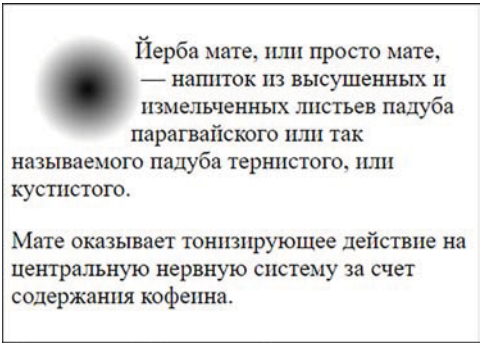
- градиент любого типа (см. *разд. 19.2*) — соседи будут обтекать элемент по границе области заданного градиента, которая имеет указанный уровень непрозрачности.

  Для задания уровня непрозрачности области градиента, по границе которой будет выполняться обтекание, предназначается атрибут стиля `shape-image-threshold`. Уровень непрозрачности указывается в виде вещественного числа от `0.0` (полностью прозрачная область) до `1.0` (полностью непрозрачная область). Значение по умолчанию: `0.0`.

Пример	Результат

`aside.shos3 {`
`  float: left;`
`  width: 100px;`
`  height: 100px;`
`  background-image:`
`  radial-gradient(black,`
`  transparent 50%,`
`  transparent);`
`  shape-outside:`
`  radial-gradient(black,`
`  transparent 70%,`
`  transparent);`
`}`
	

Пример	Результат

`aside.shos4 {`
`  float: left;`
`  width: 100px;`
`  height: 100px;`
`  background-image:`
`  radial-gradient(black,`
`  transparent 50%,`
`  transparent);`
`  shape-outside:`
`  radial-gradient(black,`
`  transparent 70%,`
`  transparent);`
`  shape-image-threshold: 0.3;`
`}`
	

- ссылку на графическое изображение, заданную посредством функции `url()`, — соседи будут обтекать элемент по границе области этого изображения, которая имеет указанный уровень непрозрачности. Уровень непрозрачности области изображения задается в атрибуте стиля `shape-image-threshold`.

ВНИМАНИЕ!

Чтобы графическое изображение было успешно использовано в качестве формы обтекания у элемента, страница должна быть загружена с веб-сервера. Если же страницу открыть с локального диска, веб-обозреватель заблокирует загрузку изображения (что делается ради безопасности), и элемент получит форму обтекания по умолчанию.

Пример	Результат

`aside.shos5 {`
`  float: left;7`
`  shape-outside: url(teacup.png);`
`}`
	

- одно из предопределенных значений: `margin-box`, `border-box`, `padding-box` или `content-box` (см. *разд. 26.2.2*).

Пример	Результат

`aside.shos6 {`
`  float: left;`
`  width: 100px;`
`  height: 100px;`
`  margin: 0 30px 30px 0;`
`  padding: 0 20px 20px 0;`
`  shape-outside: margin-box;`
`}`
	 Йерба мате, или просто мате, — напиток из высушенных и измельченных листьев падуба парагвайского или так называемого падуба тернистого, или кустистого.
	
	Мате оказывает тонизирующее действие на центральную нервную систему за счет содержания кофеина.

`aside.shos7 {`
`  float: left;`
`  width: 100px;`
`  height: 100px;`
`  margin: 0 30px 30px 0;`
`  padding: 0 20px 20px 0;`
`  shape-outside: padding-box;`
`}`
	 Йерба мате, или просто мате, — напиток из высушенных и измельченных листьев падуба парагвайского или так называемого падуба тернистого, или кустистого.
	
	Мате оказывает тонизирующее действие на центральную нервную систему за счет содержания кофеина.

- `none` — то же самое, что и `margin-box`.

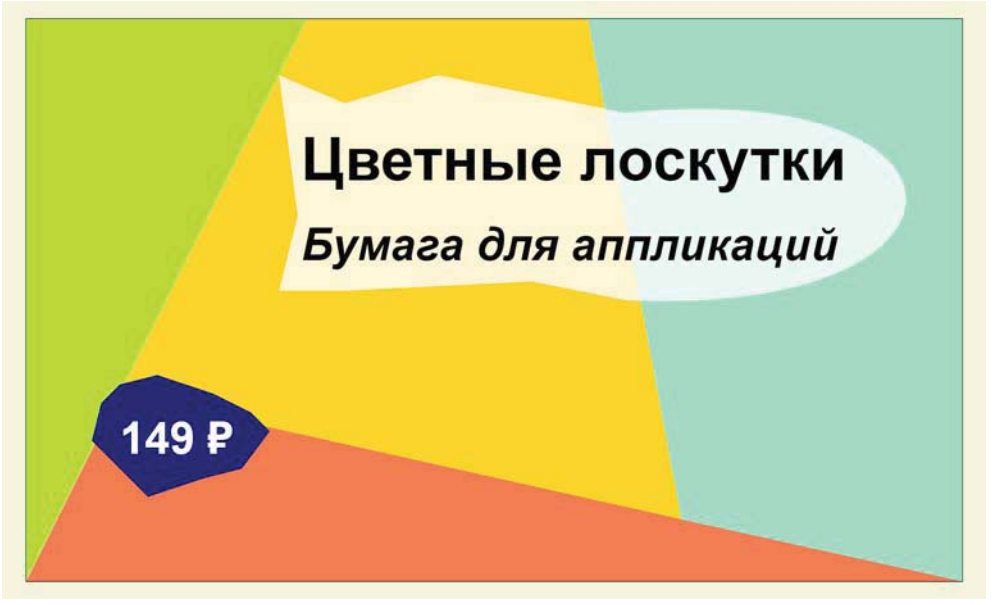
Значение по умолчанию: `none`.

26.5. Самостоятельное упражнение

Знакомый, продающий бумагу для аппликаций и просивший сделать рекламный сайт (см. *упражнение 26.5*), забыл сообщить нам цену этой бумаги и вспомнил об этом лишь тогда, когда сайт был закончен. Поместите на страницу сайта еще один «лоскут» — с ценой этой бумаги (149 руб.).

Закрасьте этот «лоскут» цветом `navy`, надпись выведите белым цветом, полужирным шрифтом без засечек кеглем 36 пунктов. Местоположение, размеры и параметры фигуры подберите опытным путем.

✓ У вас должно получиться ⌄



Урок 27. Фильтры

Фильтры CSS.
Наложение фильтров на элементы.
Наложение фильтров на области под элементами.

> ***Фильтр***
>
> Специальный эффект, накладываемый на элемент или область, расположенную за элементом: размытие, повышение яркости, преобразование в черно-белую гамму и др.

27.1. Наложение фильтров на элемент

Для наложения фильтра на сам элемент следует использовать ненаследуемый атрибут стиля `filter`. В качестве его значения указывается:

- `none` — никакие фильтры к элементу применены не будут.

Пример	Результат

`.control {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: none;`
`}`
	

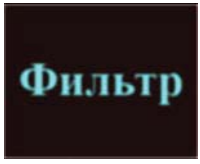
- описание фильтра, задаваемое с помощью одной из функций, которые представлены в *разд. 27.1.1*.

Пример	Результат

`.f1 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: blur(1px);`
`}`
	

- несколько описаний разных фильтров, разделенных пробелами, — тогда к элементу будут применены все эти фильтры — в том порядке, в котором приведены их описания.

Пример	Результат

`.f2 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: blur(1px) invert(100%);`
`}`
	

Значение по умолчанию: `none`.

27.1.1. Описание фильтров

Для описания фильтров применяются следующие функции CSS:

- `blur(<радиус размытия>)` — гауссово размытие с заданным *радиусом*. *Радиус размытия* задается в виде числового значения в любой единице измерения. Чем больше *радиус размытия*, тем более размытым будет элемент.

Пример	Результат

`.f1 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: blur(1px);`
`}`
	

`.f3 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: blur(5px);`
`}`
	

- `brightness(<степень>)` — изменение яркости элемента на указанную *степень*. *Степень* задается в виде числового значения без единицы измерения или в процентах. Значения *степени* меньшие 1 (или 100%) уменьшают яркость, значения большие 1 (100%) — увеличивают. Указание нулевой *степени* превращает элемент в черный силуэт.

Пример	Результат

`.f4 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: brightness(60%);`
`}`
	

`.f5 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: brightness(2);`
`}`
	

`.f6 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: brightness(0);`
`}`
	

- `contrast(<`*степень*`>)` — изменение контрастности элемента на указанную *степень*. *Степень* задается в виде числового значения без единицы измерения или в процентах. Значения *степени* меньшие 1 (или 100%) уменьшают контрастность, значения большие 1 (100%) — увеличивают. Указание нулевой *степени* превращает элемент в серый силуэт.

Пример	Результат

`.f7 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: contrast(40%);`
`}`
	

`.f8 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: contrast(2);`
`}`
	

`.f9 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: contrast(0);`
`}`
	

- `drop-shadow()` — тень у элемента, имеющая заданные *смещения*, *цвет* и *степень размытия*:

```
drop-shadow([<цвет тени>] <смещение по горизонтали>
            <смещение по вертикали> [<радиус размытия>])
```

Смещения тени задаются относительно самого элемента в виде числовых значений с использованием любых единиц измерения. Положительные значения вызывают смещение тени вправо и вниз, отрицательные — влево и вверх.

Если *цвет тени* не указан, он принимается равным цвету текста, заданному у родителя.

Радиус размытия задается в виде числа с использованием любой единицы измерения. Чем он больше, тем более размытой станет тень. Если *радиус размытия* не задан, он принимается равным `0` (размытие у тени отсутствует).

Пример	Результат

`.f10 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: drop-shadow(10pt 10pt 6pt);`
`}`
	

`.f11 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: drop-shadow(black -10pt -10pt);`
`}`
	

`.f12 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: drop-shadow(black 0 0 10pt);`
`}`
	

- `grayscale([<степень>])` — преобразование элемента в черно-белую гамму согласно заданной *степени*. *Степень* задается в виде вещественного числа без единицы измерения от `0.0` до `1.0` или от 0% до 100%. Значение `0.0` или 0% не изменяет элемент, значение `1.0` или 100% делает его черно-белым полностью. Если *степень* не указана, они принимается равной `1.0`.

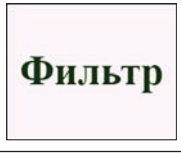
Пример	Результат

`.f13 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: grayscale(50%);`
`}`
	

`.f14 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: grayscale();`
`}`
	

- `hue-rotate(<угол>)` — поворот значений оттенка у всех цветов элемента на указанный *угол*. *Угол* задается в виде числового значения в любой поддерживаемой единице измерения (см. *разд. 12.2.1.2.2*). Допускаются как положительные, так и отрицательные значения *угла*. *Угол*, равный `0`, не изменяет элемент.

Пример	Результат

`.f15 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: hue-rotate(90deg);`
`}`
	

`.f16 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: hue-rotate(-120deg);`
`}`
	

- `invert(<степень>)` — инвертирование цветов элемента согласно указанной *степени*. *Степень* задается в виде вещественного числа без единицы измерения от `0.0` до `1.0` или от 0% до 100%. Значение `0.0` или 0% не изменяет элемент, значение `1.0` или 100% инвертирует его цвета полностью.

Пример	Результат

`.f17 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: invert(30%);`
`}`
	

`.f18 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: invert(1.0);`
`}`
	

- `opacity(<степень>)` — делает элемент полупрозрачным соответственно заданной *степени*. *Степень* указывается в виде вещественного числа без единицы измерения от `0.0` до `1.0` или от 0% до 100%. Значение `0.0` или 0% делает элемент полностью прозрачным, значение `1.0` или 100% — полностью непрозрачным.

Пример	Результат

`.f19 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: opacity(0.3);`
`}`
	

- `saturate(<степень>)` — изменение насыщенности цветов элемента на указанную *степень*. *Степень* задается в виде числового значения без единицы измерения или от 0% до 100%. Значения *степени* меньшие 1 (или 100%) уменьшают насыщенность, значения большие 1 (100%) — увеличивают. Указание нулевой *степени* делает элемент черно-белым.

Пример	Результат

`.f20 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: saturate(0.5);`
`}`
	

`.f21 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: saturate(3);`
`}`
	

`.f22 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: saturate(0);`
`}`
	

- `sepia(<степень>)` — преобразование элемента в сепию согласно заданной *степени*. *Степень* задается в виде вещественного числа без единицы измерения от `0.0` до `1.0` или от 0 до 100%. Значение `0.0` или 0% не изменяет элемент, значение `1.0` или 100% преобразует его в сепию полностью.

Пример	Результат

`.f23 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: sepia(30%);`
`}`
	

`.f24 {`
`  color: darkred;`
`  background-color: lightcyan;`
`  filter: sepia(100%);`
`}`
	

Как показывает практика, в функции `drop-shadow()` значение цвета можно указать как перед числовыми значениями, так и после них:

```
filter: drop-shadow(0 0 10pt black);
```

27.2. Наложение фильтров на область под элементом

Также можно наложить фильтр не на сам элемент, а на область, расположенную непосредственно под ним. Для этого применяется ненаследуемый атрибут стиля `backdrop-filter`, поддерживающий те же значения, что и атрибут стиля `filter` (см. *разд. 27.1*).

Пример	Результат

`.control {`
`  backdrop-filter: none;`
`}`
	

`.bdf1 {`
`  backdrop-filter: brightness(160%);`
`}`
	

`.bdf2 {`
`  backdrop-filter: brightness(160%) blur(10px);`
`}`
	

27.3. Самостоятельное упражнение

На сайте видеокиоска (см. *упражнение 22.4*) — укажите у шапки и обеих подписей два фильтра, накладываемые на области под этими элементами: первый — уменьшающий контрастность вдвое, второй — задающий размытие с радиусом 4 пункта. Посмотрите, какой эффект будет достигнут в результате.

Урок 28. Двухмерные преобразования

Смещение.
Масштабирование.
Поворот.
Скос.

Преобразование

Изменение местоположения, формы и (или) размеров элемента страницы.

Двухмерное преобразование

Преобразование, выполняемое в двухмерной плоскости.

Преобразование можно применить к элементу либо раз и навсегда, либо на время, обычно при наведении курсора мыши. В обоих случаях применение преобразований позволяет оживить страницу.

Для применения к элементам двухмерных преобразований CSS предоставляет два набора инструментов: новый, более удобный в использовании, но ограниченный в плане функциональных возможностей, и старый, менее удобный, однако намного более развитый.

ВНИМАНИЕ!

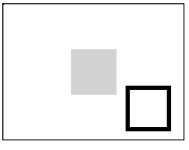
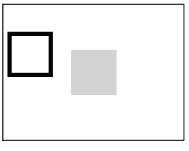
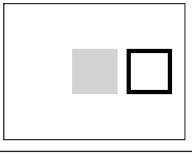
Все описанные здесь атрибуты стилей являются ненаследуемыми.

28.1. Создание двухмерных преобразований: новый инструментарий

Новый инструментарий для создания двухмерных преобразований представляет собой набор следующих атрибутов стиля:

- `translate` — смещение элемента относительно его изначального местоположения. Можно задать значение:
  - *<смещение по горизонтали>* [*<смещение по вертикали>*]

    Смещения указываются в виде числовых значений в любых поддерживаемых единицах измерения. Положительные значения вызывают смещение вправо и вниз, отрицательные — влево и вверх. Если *смещение по вертикали* не задано, оно принимается равным `0` (то есть по вертикали элемент не сместится);
  - `none` — элемент никуда не сместится.

Пример	Результат[1]
`.trans1 { translate: 60px 40px; }`	
`.trans2 { translate: -70px -20px; }`	
`.trans3 { translate: 60px; }`	

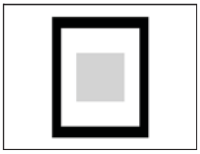
- `scale` — масштабирование элемента. Допустимые значения:
  - *<масштаб по горизонтали>* [*<масштаб по вертикали>*]

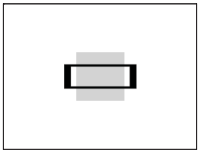
    Масштабы задаются в виде вещественных чисел без единиц измерения. Значения меньше `1.0` вызывают уменьшение элемента в соответствующем направлении, значение `1.0` не приводит к изменению масштаба, а значения больше `1.0` приводят к увеличению элемента. Если масштаб по вертикали не задан, он принимается равным заданному *масштабу по горизонтали*;

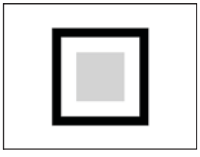
[1] Здесь и далее элемент в изначальном состоянии закрашен серым фоном, а тот же элемент после применения к нему преобразования обозначен толстой черной рамкой.

- `none` — элемент не подвергнется масштабированию.

Пример	Результат

`.trans4 {`
`  scale: 2.0 2.5;`
`}`
	

`.trans5 {`
`  scale: 1.5 0.5;`
`}`
	

`.trans6 {`
`  scale: 2.0;`
`}`
	

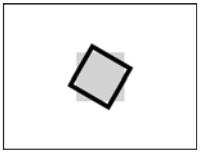
♦ `rotate` — поворот элемента вокруг оси Z. Доступные для указания значения:

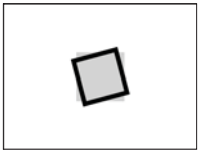
- `[z] <угол>`

  Угол задается в виде числового значения с применением любой поддерживаемой единицы измерения (см. *разд. 12.2.1.2.2*). Положительные значения *угла* вызывают поворот по часовой стрелке, отрицательные — против часовой стрелки. Букву `z` перед значением *угла* можно не указывать — она служит лишь для наглядности;

- `none` — элемент не будет повернут.

Пример	Результат

`.trans7 {`
`  rotate: 30deg;`
`}`
	

`.trans8 {`
`  rotate: -15deg;`
`}`
	

Атрибут стиля `transform-origin` задает точку, относительно которой элемент будет масштабироваться или поворачиваться. Его значение записывается в формате:

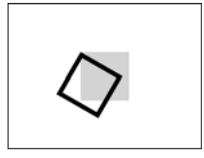
<горизонтальная координата> [*<вертикальная координата>*]

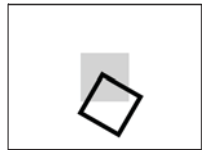
- *горизонтальная координата* может быть указана в виде числовой величины в какой-либо единице измерения, а также в виде значений `left` (левый край элемента), `center` (центр) или `right` (правый край);
- *вертикальная координата* может быть задана в виде числовой величины в какой-либо единице измерения, а также в виде значений `top` (верхний край элемента), `center` (центр) или `bottom` (нижний край). Если вертикальная координата не указана, она получит значение `50%`.

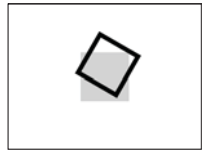
Координаты отсчитываются относительно левого верхнего угла элемента. Положительные значения отсчитываются в направлениях вправо и вниз, отрицательные — влево и вверх.

Значение по умолчанию: `50% 50%`.

Пример	Результат

`.trans9 {`
`  transform-origin: left top;`
`  rotate: 30deg;`
`}`
	

`.trans10 {`
`  transform-origin: -25px 50px;`
`  rotate: 30deg;`
`}`
	

`.trans11 {`
`  transform-origin: 50px;`
`  rotate: 30deg;`
`}`
	

ВНИМАНИЕ!

К сожалению, применяя новый инструментарий, можно задать у элемента только какое-либо одно двухмерное преобразование. Если попытаться указать два преобразования или более, работать будет лишь заданное самым последним.

28.2. Упражнение. Оживление панели навигации

На страницах сайта о суши-баре при наведении курсора мыши на какую-либо гиперссылку в панели навигации эта гиперссылка принимает «негативный» вид — светлый текст на темном фоне. Давайте дополнительно оживим панель навигации, сделав так, чтобы гиперссылка при наведении слегка смещалась вправо.

1. Найдем в папке 21\s21.2 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Смещать гиперссылки будем только по горизонтали и ненамного — на 10 пикселов, чтобы они не вышли за пределы панели навигации.

2. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и дополним стиль активной гиперссылки и гиперссылки под курсором мыши, находящихся в семантической панели навигации:

```
nav a:active, nav a:hover {
    . . .
    translate: 10px;
}
```

✓ Результат ›

Курсор мыши наведен на гиперссылку **Прайс-лист**.

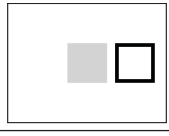
28.3. Создание двухмерных преобразований: старый инструментарий

Этот инструментарий хоть и стар, но отнюдь не устарел. Поскольку предоставляет значительно больше функциональных возможностей, нежели новый инструментарий (см. *разд. 28.1*). В частности, позволяет указывать у элемента произвольное количество преобразований.

Преобразования задаются с помощью атрибута стиля `transform`. Его значением может быть:

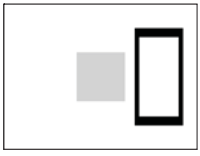
- описание какого-либо преобразования, задаваемое с помощью одной из функций, которые описаны в *разд. 28.3.1*.

Пример	Результат

`.trans12 {`
`  transform: translateX(60px);`
`}`
	

- несколько описаний разных преобразований, разделенных пробелами, — тогда к элементу будут применены все эти преобразования — в том порядке, в котором приведены их описания.

Пример	Результат

`.trans13 {`
`  transform: translateX(60px) scale(1.0, 2.0);`
`}`
	

- `none` — никакие преобразования к элементу применены не будут.

Значение по умолчанию: `none`.

28.3.1. Описание двухмерных преобразований

Для описания двухмерных преобразований применяются следующие функции CSS (большая часть которых аналогична атрибутам стиля, рассмотренным в *разд. 28.1*):

- `translate()` — смещение элемента относительно его изначального местоположения. Формат записи функции:

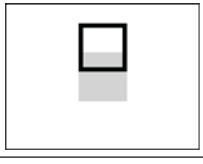
  `translate(`*<смещение по горизонтали>*`[, <`*смещение по вертикали*`>])`

  Смещения указываются в виде числовых значений в любых поддерживаемых единицах измерения. Положительные значения вызывают смещение вправо и вниз, отрицательные — влево и вверх. Если *смещение по вертикали* не задано, оно принимается равным `0`;

- `translateX(`*<смещение по горизонтали>*`)` — смещение элемента только по горизонтали;

- `translateY(`*<смещение по вертикали>*`)` — смещение элемента только по вертикали.

Пример	Результат

`.trans14 {`
`  transform: translateY(-30px);`
`}`
	

- `scale()` — масштабирование элемента:

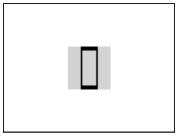
  `scale(`*<масштаб по горизонтали>*`[, <`*масштаб по вертикали*`>])`

  Масштабы задаются в виде вещественных чисел без единиц измерения. Значения меньше `1.0` вызывают уменьшение элемента в соответствую-

щем направлении, значение `1.0` не приводит к изменению масштаба, а значения больше `1.0` приводят к увеличению элемента. Если *масштаб по вертикали* не задан, он принимается равным *масштабу по горизонтали*;

- `scaleX(<масштаб по горизонтали>)` — масштабирование элемента только по горизонтали.

Пример	Результат

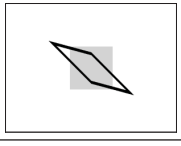
`.trans15 {`
`  transform: scaleX(.4);`
`}`
	

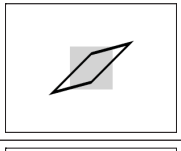
- `scaleY(<масштаб по вертикали>)` — масштабирование элемента только по вертикали;
- `rotate(<угол>)` — поворот элемента вокруг оси Z. *Угол* задается в виде числового значения с применением любой поддерживаемой единицы измерения (см. *разд. 12.2.1.2.2*). Положительные значения *угла* вызывают поворот по часовой стрелке, отрицательные — против часовой стрелки;
- `skew()` — скашивание элемента по горизонтали и вертикали. Формат записи функции:

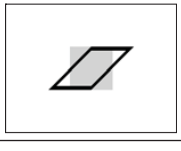
  `skew(<угол скоса по горизонтали>[, <угол скоса по вертикали>])`

  Углы задаются в виде числовых значений в любых единицах измерения (см. *разд. 12.2.1.2.2*). Положительные значения *угла* вызывают скос против часовой стрелки, отрицательные — по часовой стрелке. Если *угол скоса по вертикали* не задан, он принимается равным `0` (т. е. элемент не будет скошен по вертикали).

Пример	Результат

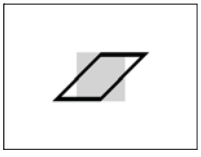
`.trans16 {`
`  transform: skew(45deg, 15deg);`
`}`
	

`.trans17 {`
`  transform: skew(-45deg, -15deg);`
`}`
	

`.trans18 {`
`  transform: skew(-45deg);`
`}`
	

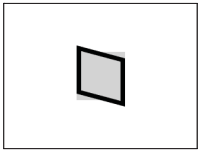
- `skewX(<угол скоса по горизонтали>)` — скашивание элемента только по горизонтали.

Пример	Результат

`.trans19 {`
`  transform: skewX(-45deg);`
`}`
	

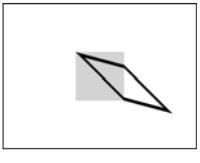
- `skewY(<угол скоса по вертикали>)` — скашивание элемента только по вертикали.

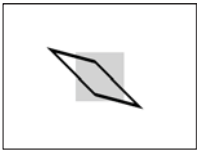
Пример	Результат

`.trans20 {`
`  transform: skewY(15deg);`
`}`
	

Для указания точки, относительно которой элемент будет масштабироваться, поворачиваться или скашиваться, можно использовать атрибут стиля `transform-origin` (см. *разд. 28.1*).

Пример	Результат

`.trans21 {`
`  transform-origin: left top;`
`  transform: skew(45deg, 15deg);`
`}`
	

`.trans22 {`
`  transform-origin: 20px 30px;`
`  transform: skew(45deg, 15deg);`
`}`
	

28.4. Упражнение. Оживление современного рекламного проспекта

Немного оживим современный рекламный проспект, сделанный при выполнении *упражнений 24.7* и *24.8*, а именно:

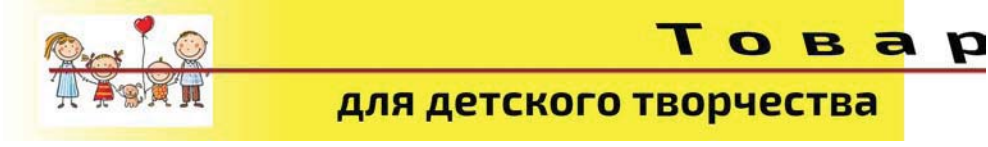
- первый заголовок из состава шапки скосим на 15° против часовой стрелки и растянем на 80% по горизонтали;
- второй заголовок из состава шапки скосим на 15° по часовой стрелке.

1. Найдем в папке 24\s24.8 сопровождающего книгу файлового архива (см. *приложение 4*) папку kid-creativity и скопируем ее куда-либо на локальный диск.

   К первому заголовку, находящемуся в шапке, необходимо применить сразу два преобразования. Это можно сделать, лишь применив старый (но, автор повторяет, не устаревший!) инструментарий.
2. Откроем таблицу стилей styles\main.css в текстовом редакторе и дополним стиль элемента со стилевым классом `heading1`:

```
.heading1 {
    . . .
    transform: skewX(15deg) scaleX(1.8);
}
```

✓ Что у нас получилось? ⌄



Получилось не здорово... Поскольку любой элемент по умолчанию масштабируется относительно центра и «растет» влево и вправо, правая часть заголовка «уехала» за край области просмотра. Чтобы исправить этот недочет, следует указать новую точку, от которой заголовок будет масштабироваться, — расположенную на его правом краю, чтобы заголовок «рос» только влево.

3. Дополним еще раз тот же стиль:

```
.heading1 {
    . . .
    transform: skewX(15deg) scaleX(1.8);
    transform-origin: right;
}
```

✓ Что у нас получилось? ⌄

Заголовок вернулся. Но подчеркивающая его нижняя сторона рамки немного выступает с правой стороны (вероятно, это результат скоса), а с левой стороны устремилась куда-то в бесконечность. Исправить это можно, задав небольшой внешний просвет справа и увеличив значение внешнего просвета слева. Подбирать точные значения просветов придется опытным путем.

4. Исправим тот же стиль:

```
.heading1 {
    . . .
    margin-left: calc(100% - 500pt);
    margin-left: calc(100% - 290pt);
    margin-right: 8pt;
    . . .
}
```

✓ Что у нас получилось? ⌄

Вот теперь все в порядке. Займемся нижним заголовком.

5. Добавим в стиль, применяемый к элементу со стилевым классом `heading2`, необходимый код:

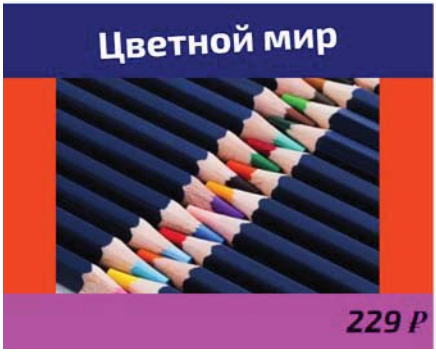
```
.heading2 { padding: 0 20pt 10pt 0; }
.heading2 {
    padding: 0 20pt 10pt 0;
    transform: skewX(-15deg);
}
```

✓ Результат ⌄

28.5. Самостоятельное упражнение

На странице современного рекламного проспекта поверните каждый нечетный заголовок с названием товара на 3° против часовой стрелки, а каждый четный — на 3° по часовой стрелке.

✓ У вас должно получиться ⌄



Урок 29. Трехмерные преобразования

Глубина перспективы.
Положение наблюдателя.
Трехмерные смещение, масштабирование, поворот и скос.
Дополнительные настройки.

> ***Трехмерное преобразование***
>
> Преобразование, выполняемое в трехмерном пространстве.

Для применения к элементам трехмерных преобразований также можно использовать один из двух наборов инструментов: новый, более простой в применении, но ограниченный, и старый, менее удобный, однако более развитый.

ВНИМАНИЕ!

Все описанные здесь атрибуты стилей являются ненаследуемыми.

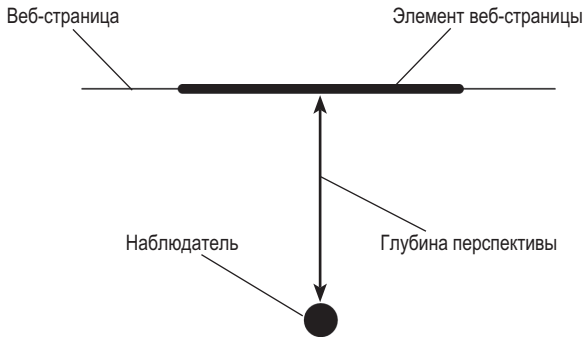
29.1. Указание глубины перспективы и положения наблюдателя

Перед указанием у какого-либо элемента трехмерных преобразований у *родителя* этого элемента необходимо задать *глубину перспективы* и при необходимости — *положение наблюдателя*.

> ***Глубина перспективы***
>
> Расстояние от элемента страницы до точки вне плоскости страницы, в которой находится воображаемый наблюдатель, рассматривающий этот элемент. Указывается у родителя элемента, к которому применяются трехмерные преобразования.

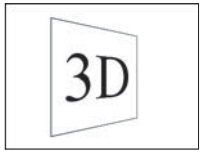
Положение наблюдателя

Положение воображаемой точки, в которой находится наблюдатель, относительно родителя элемента, к которому применяются трехмерные преобразования.



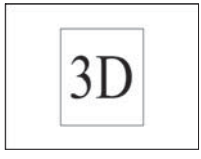
Глубина перспективы задается атрибутом стиля `perspective`. Доступные для указания значения:

- числовая величина в любой поддерживаемой единице измерения задаст глубину перспективы, после чего трехмерные преобразования, примененные к элементу, будут иметь видимый эффект.

Пример	Результат
`.parent1 { perspective: 100pt; }` `.trans3d1 { rotate: y 30deg; }`	

Если указать глубину перспективы равную `0`, трехмерные преобразования не будут иметь видимого эффекта;

- `none` — перспектива отсутствует (то же самое, что и `0`).

Пример	Результат
`.parent2 { perspective: none; }` `.trans3d2 { rotate: y 30deg; }`	

Значение по умолчанию: `none`.

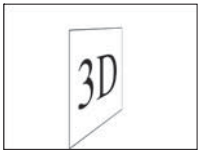
Чем меньше глубина перспективы, тем сильнее будет искажаться форма элемента после применения к нему трехмерных трансформаций.

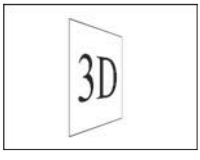
Для задания местоположения наблюдателя относительно родительского элемента служит атрибут стиля `perspective-origin`. Его значение может быть одним из следующих:

- *<местоположение по горизонтали>* [*<местоположение по вертикали>*]

  Местоположения задаются в виде числовых величин в любой из поддерживаемых единиц измерения. Они отсчитываются от левого верхнего угла родителя: положительные значения — вправо и вниз, отрицательные — влево и вверх. Если *местоположение по вертикали* не указано, оно принимается равным `50%`.

Пример	Результат

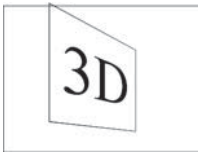
`.parent3 {`
`  perspective: 100pt;`
`  perspective-origin: 10px 20px;`
`}`
`.trans3d3 { rotate: y 30deg; }`
	

`.parent4 {`
`  perspective: 100pt;`
`  perspective-origin: 10px;`
`}`
`.trans3d4 { rotate: y 30deg; }`
	

- [*<местоположение по горизонтали>*] [*<местополож. по вертикали>*]

  Местоположение по горизонтали указывается в виде предопределенного значения `left` (левый край родителя), `center` (центр) или `right` (правый край), *местоположение по вертикали* — в виде значения `top` (верхний край), `center` (центр) или `bottom` (нижний край). Если какое-либо из местоположений не задано, оно принимается равным `center`.

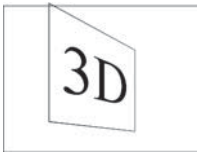
Пример	Результат

`.parent5 {`
`  perspective: 100pt;`
`  perspective-origin: center bottom;`
`}`
`.trans3d5 { rotate: y 30deg; }`
	

`.parent6 {`
`  perspective: 100pt;`
`  perspective-origin: bottom;`
`}`
`.trans3d6 { rotate: y 30deg; }`	

Можно одновременно использовать числовые и предопределенные значения.

Пример	Результат

`.parent7 {`
`  perspective: 100pt;`
`  perspective-origin: 50% bottom;`
`}`
`.trans3d7 { rotate: y 30deg; }`
	

Значение по умолчанию: `50% 50%`.

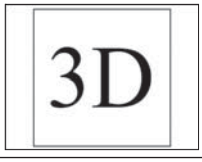
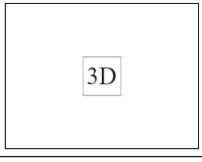
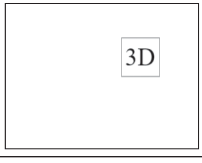
29.2. Создание трехмерных преобразований: новый инструментарий

Новый инструментарий — это набор атрибутов стиля, описанных в *разд. 28.1*, только с расширенным синтаксисом:

- `translate` — смещение элемента. Расширенный формат значения выглядит так:

  <смещение по горизонтали> [*<смещение по вертикали>* [*<смещение по оси Z>*]]

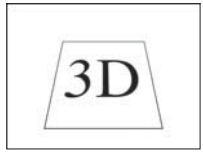
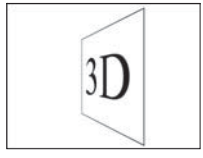
  Смещение по оси Z задается так же, как и два других *смещения*. Оно отсчитывается от плоскости родителя: положительные значения — в направлении к наблюдателю, отрицательные — от наблюдателя. Если *смещение по оси Z* не указано, оно принимается равным `0`.

Пример	Результат
`.parent8 { perspective: 100pt; }` `.trans3d8 { translate: 0 0 40px; }`	
`.parent9 { perspective: 100pt; }` `.trans3d9 { translate: 0 0 -200px; }`	
`.parent10 { perspective: 100pt; }` `.trans3d10 { translate: 100px -50px -200px; }`	

- `rotate` — поворот элемента вокруг оси X или Y. Расширенный формат значения:

  `[x|y|z] <угол>`

  Указание буквы *x* приводит к повороту вокруг оси X, указание буквы *y* — вокруг оси Y. Если задать букву `z` или не задавать вообще никакой буквы, элемент будет повернут вокруг оси Z.

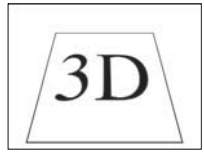
Пример	Результат
`.parent11 { perspective: 100pt; }` `.trans3d11 { rotate: x 30deg; }`	
`.parent12 { perspective: 100pt; }` `.trans3d12 { rotate: y -0.15turn; }`	

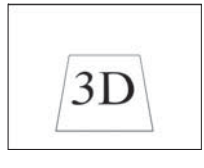
Для задания точки, относительно которой элемент будет поворачиваться, поддерживается расширенный формат атрибута стиля `transform-origin`:

<горизонтальная координата> [*<вертикальная координата>* [*<координата по оси Z>*]]

Координата по оси Z задается так же, как и две других *координаты*. Она отсчитывается от плоскости родителя: положительные значения — в направлении к наблюдателю, отрицательные — от наблюдателя. Если *координата по оси Z* не указана, она принимается равной `0`.

Пример	Результат

`.parent13 { perspective: 100pt; }`
`.trans3d13 {`
`  transform-origin: 0 0 10px;`
`  rotate: x 30deg;`
`}`
	

`.parent14 { perspective: 100pt; }`
`.trans3d14 {`
`  transform-origin: left bottom 10px;`
`  rotate: x 30deg;`
`}`
	

29.3. Создание трехмерных преобразований: старый инструментарий

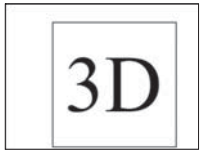
Старый инструментарий — это уже знакомый нам атрибут стиля `transform` (см. *разд. 28.3*). Для описания трехмерных преобразований в нем применяются следующие функции:

- `translate3d()` — смещение элемента по всем трем координатным осям:

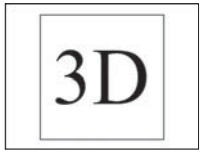
  `translate3d(<смещение по горизонтали>, <смещение по вертикали>, <смещение по оси Z>)`

  Смещения задаются так же, как в функции `translate()` (см. *разд. 28.3.1*) и атрибуте стиля `translate` (см. *разд. 29.2*). Следует отметить, что все три *смещения* обязательны для указания.

Пример	Результат

`.parent15 { perspective: 100pt; }`
`.trans3d15 {`
`  transform: translate3d(10px, 6px, 30px);`
`}`
	

- `translateZ(<смещение по оси Z>)` — смещение элемента только по оси Z.

Пример	Результат
`.parent16 { perspective: 100pt; }` `.trans3d16 { transform: translateZ(30px); }`	

- `scale3d()` — масштабирование элемента по всем трем координатным осям:

  `scale3d(<масштаб по горизонтали>, <масштаб по вертикали>, <масштаб по оси Z>)`

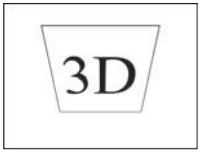
  Масштабы задаются так же, как и в функции `scale()` (см. *разд. 28.3.1*). Все три *масштаба* обязательны для указания.

  Чтобы увидеть эффект от масштабирования по оси Z, после масштабирования следует сместить элемент по оси Z.

Пример	Результат

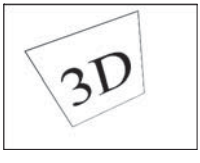
`.parent17 { perspective: 100pt; }`
`.trans3d17 {`
`  transform: scaleZ(1.4) translateZ(30px);`
`}`
	

- `scaleZ(<`*`масштаб по оси Z`*`>)` — масштабирование элемента только по оси Z;
- `rotateX(<`*`угол`*`>)` — поворот элемента вокруг горизонтальной оси на заданный *`угол`*. *`Угол`* задается так же, как и в функции `rotate()` (см. *разд. 28.3.1*).

Пример	Результат
`.parent18 { perspective: 100pt; }` `.trans3d18 { transform: rotateX(-30deg); }`	

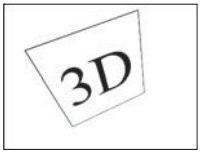
- `rotateY(<`*`угол`*`>)` — поворот элемента вокруг вертикальной оси на заданный *`угол`*;
- `rotateZ(<`*`угол`*`>)` — то же самое, что и `rotate()`.

Пример	Результат

`.parent19 { perspective: 100pt; }`
`.trans3d19 {`
`  transform: rotateX(-30deg) rotateY(20deg)`
`  rotateZ(-15deg);`
`}`
	

- `perspective(<`*`глубина перспективы`*`>)` — задает *`глубину перспективы`* у элемента. Может быть использована для задания у элемента глубины перспективы, отличной от указанной у его родителя (задается атрибутом стиля `perspective` — см. *разд. 29.1*).

Пример	Результат

`.parent20 { perspective: none; }`
`.trans3d20 {`
`  transform: perspective(100pt)`
`  rotateX(-30deg) rotateY(20deg)`
`  rotateZ(-15deg);`
`}`
	

29.4. Дополнительные настройки трехмерных преобразований

Если повернуть элемент страницы на угол более 90° вокруг горизонтальной или вертикальной оси, элемент перевернется, показав свою обратную сторону. Атрибут стиля `backface-visibility` указывает, следует ли в этом случае выводить содержимое элемента. Вот доступные значения:

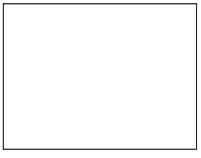
- `visible` — содержимое элемента будет выводиться на экране перевернутым, «просвечивая» сквозь элемент.

Пример	Результат

`.trans3d21 {`
`  transform: rotateY(0.4turn);`
`  backface-visibility: visible;`
`}`
	

- `hidden` — содержимое элемента не будет выводиться на страницу. Собственно, сам элемент целиком станет невидимым.

Пример	Результат

`.trans3d21 {`
`  transform: rotateY(0.4turn);`
`  backface-visibility: hidden;`
`}`
	

Значение по умолчанию: `visible`.

Может возникнуть необходимость применить трехмерные преобразования к потомкам элемента, к которому уже применены трехмерные преобразования. Атрибут стиля `transform-style` указывает, как потомки, подвергаемые трехмерным преобразованиям, должны размещаться внутри родителя.

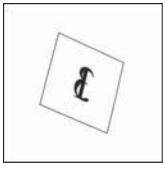
Тестовый код:

```
<div class="cont">
    <div class="cube">
        <div class="part part1">1</div>
        <div class="part part2">2</div>
        <div class="part part3">3</div>
        <div class="part part4">4</div>
        <div class="part part5">5</div>
        <div class="part part6">6</div>
    </div>
```

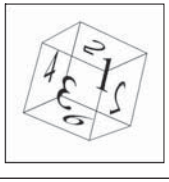
```
</div>
. . .
.cont { perspective: 1000pt; }
.cube {
    position: relative;
    transform: rotateX(15deg) rotateY(30deg) rotateZ(15deg);
}
.part {
    position: absolute;
    left: 0;
    top: 0;
    right: 0;
    bottom: 0;
    border: thin solid black;
}
.part1 { transform: translateZ(50px); }
.part2 { transform: rotateY(90deg) translateZ(50px); }
.part3 { transform: rotateY(0.5turn) translateZ(50px); }
.part4 { transform: rotateY(-90deg) translateZ(50px); }
.part5 { transform: rotateX(90deg) translateZ(50px); }
.part6 { transform: rotateX(-90deg) translateZ(50px); }
```

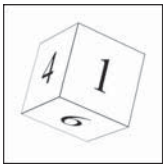
Значения, поддерживаемые атрибутом стиля `transform-style`:

- `flat` — потомки размещаются в плоскости элемента-родителя. Соответственно, все трехмерные преобразования, примененные к потомкам, не будут иметь видимого эффекта.

Пример	Результат
`.cube { transform-style: flat; }`	

- `preserve-3d` — потомки позиционируются в трехмерном пространстве.

Пример	Результат
`.cube { transform-style: preserve-3d; }`	

Пример	Результат
`.cube { transform-style: preserve-3d; }` `.part { backface-visibility: hidden; }`	

Значение по умолчанию: `flat`.

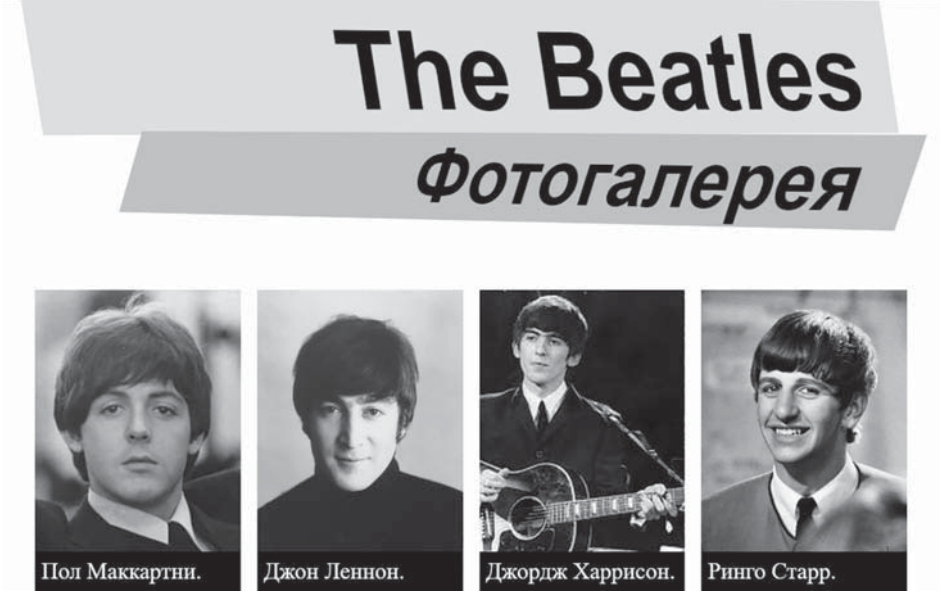
29.5. Самостоятельное упражнение

На странице сайта с подборкой «битловских» фото, в шапке:

- верхний заголовок — поверните по вертикальной оси на 10° по часовой стрелке, сожмите по горизонтали на 10% (чтобы заголовок не вылез за границы окна), скосите по горизонтали на 10° против часовой стрелки и сместите по оси Z «от себя» на 40 пикселов;
- нижний заголовок — поверните по вертикальной оси на 20° против часовой стрелки, сожмите по горизонтали на 10%, скосите по горизонтали на 20° по часовой стрелке и сместите по оси Z «на себя» на 40 пикселов.

Глубину перспективы укажите равной 1000 пунктов. Задайте ее у семантической шапки — родителя обоих заголовков.

✓ Получится весьма стильная шапка ⌄



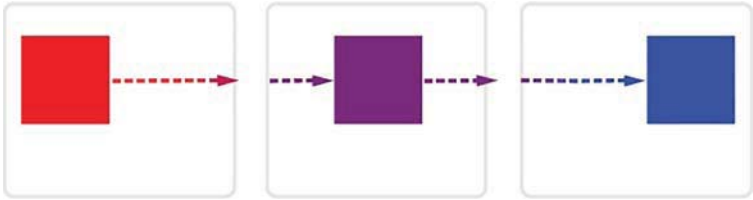
Урок 30. Анимация

Анимация с двумя состояниями.
Анимация с несколькими состояниями.
Анимация по фигуре.

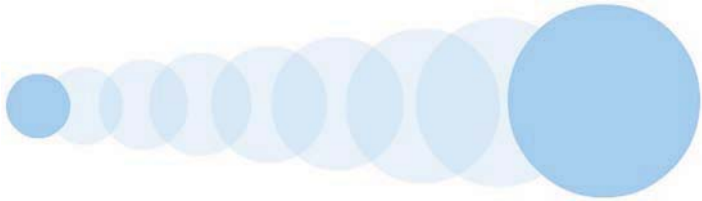
Анимация

Плавное изменение значений заданных атрибутов стиля, действующих на элемент страницы, в течение указанного времени. Благодаря этому, возникает впечатление, будто элемент страницы перемещается по ней, изменяет размеры, цвет текста или фона и т. п.

Анимировать можно местоположение элемента...



...его размеры...



...или сразу несколько параметров (например, местоположение, размеры и цвет фона) одновременно.

Состояние анимации

Определяет граничное положение анимируемого элемента: в начале анимации, при очередном изменении его параметров (например, в точке изменения траектории движения) и конце анимации.

Фаза анимации

Положение анимируемого элемента, сгенерированное веб-обозревателем в процессе выполнения анимации на основе заданных состояний анимации.

CSS позволяет создавать анимацию трех разновидностей.

ВНИМАНИЕ!

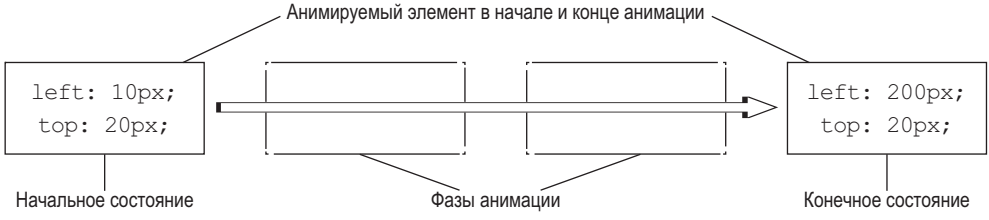
Все описанные здесь атрибуты стиля являются ненаследуемыми.

30.1. Анимация с двумя состояниями

Анимация с двумя состояниями

Анимация, включающая всего два состояния: начальное и конечное.

Вот пример анимации с двумя состояниями, при которой элемент меняет горизонтальную координату от 10 до 200 пикселов:



Начальное и конечное состояния анимации записываются в обычных CSS-стилях с применением рассмотренных ранее атрибутов стилей. Стиль, описывающий начальное состояние анимируемого элемента, применяется к анимируемому элементу изначально. Стиль, в котором описано конечное состояние, применяется к элементу в результате какого-либо действия пользователя (например, наведения курсора мыши на элемент). Применение этого стиля и запускает анимацию.

Анимироваться будут атрибуты стиля, присутствующие в обоих этих стилях и отвечающие следующим условиям:

- их значения в обоих стилях должны различаться.

  В этом случае значения атрибутов стиля будут изменяться веб-обозревателем от величин, заданных в начальном состоянии, до указанных в конечном состоянии. В результате чего и возникнет анимация;

- их значения должны представлять собой числовые значения размеров, выраженные в любых единицах измерения (кроме процентов), углов или значения цветов любого формата (в том числе заданные в виде имен цветов).

  Атрибуты стиля со значениями в виде строк, ссылок на файлы и с предопределенными значениями не будут анимироваться;

- они должны быть помечены как подвергаемые анимации (как это сделать, описано в *разд. 30.1.1*).

Атрибуты стиля, задающие параметры анимации, помещаются в стиль, описывающий *конечное состояние анимации* (если анимация не является обратной — см. *разд. 30.1.3*).

Преимущество и недостатки анимации с двумя состояниями:

Преимущество	Недостатки
Простота создания	• Позволяет реализовать лишь наиболее простые анимационные эффекты. • Может быть запущена лишь в ответ на действие пользователя.

30.1.1. Создание анимации с двумя состояниями

Важнейшая настройка анимации с двумя состояниями — ее продолжительность. Она задается атрибутом стиля `transition-duration`. В качестве его значения указывается числовая величина времени в любой из поддерживаемых единиц измерения (см. *разд. 12.2.1.2.3*). Значение по умолчанию: `0` (т. е. анимация отсутствует).

Пример CSS-кода, который создает анимацию, показанную на рисунке ранее, длящуюся 3 с и запускаемую при наведении на элемент курсора мыши:

```
.animated1 {
    position: absolute;
    left: 10px;
    top: 20px;
}
.animated1:hover {
    left: 200px;
    transition-duration: 3s;
}
```

Пример анимации, при которой элемент страницы не только перемещается по странице, но и меняет цвет с серого на желтый:

```
.animated2 {
    position: absolute;
```

```
    left: 10px;
    top: 20px;
    background-color: grey;
}
.animated2:hover {
    left: 200px;
    background-color: yellow;
    transition-duration: 3s;
}
```

Если необходимо подвергнуть анимации не все атрибуты стиля с изменяющимися значениями, а лишь один, на помощь придет атрибут стиля `transition-property`. В качестве его значения задается:

- имя атрибута стиля, который требуется анимировать;
- `all` — все атрибуты стиля с изменившимися значениями;
- `none` — ни один из атрибутов стиля.

Значение по умолчанию: `all`.

Пример, аналогичный приведенному ранее, но анимирующий лишь изменение местоположения элемента:

```
.animated3 {
    position: absolute;
    left: 10px;
    top: 20px;
    background-color: grey;
}
.animated3:hover {
    left: 200px;
    background-color: yellow;
    transition-duration: 3s;
    transition-property: left;
}
```

30.1.2. Дополнительные параметры анимации с двумя состояниями

Атрибут стиля `transition-delay` определяет задержку перед началом анимации. Его значение записывается в том же формате, что и у атрибута стиля `transition-duration`. Значение по умолчанию: `0` (анимация начнет воспроизводиться без задержки).

Атрибут стиля `transition-timing-function` задает закон протекания анимации. Вот его значения:

- `ease` — в начале скорость небольшая, далее она резко увеличивается, а незадолго до окончания анимации резко уменьшается;
- `linear` — скорость неизменна;
- `ease-in` — в начале скорость небольшая, потом она увеличивается и остается неизменной до конца;
- `ease-out` — почти все время скорость остается неизменной и ближе к концу уменьшается;
- `ease-in-out` — скорость медленно нарастает, а потом так же медленно убывает;
- `step-start` — атрибут стиля скачком меняет значение в начале анимации;
- `step-end` — атрибут стиля скачком меняет значение в конце анимации;
- `step(`*`<количество скачков>`*`[, `*`<правило>`*`])` — атрибут стиля меняет свое значение скачкообразно, за заданное *`количество скачков`*, согласно указанному правилу. В качестве правила можно задать:
  - `start` или `jump-start` — изменение значение происходит в начале каждого скачка;
  - `end` или `jump-end` — изменение значения происходит в конце каждого скачка;
  - `jump-none` — изменение значения происходит в середине каждого скачка;
  - `jump-both` — то же самое, что и `jump-none`, только на первом и последнем скачке изменения значения не происходит.

  Если правило не указано, оно принимается равным `end`.

Значение атрибута стиля `transition-timing-function` по умолчанию: `ease`.

Пример:

```
.animated4:hover {
    left: 200px;
    transition-duration: 3s;
    transition-delay: 0.5s;
    transition-timing-function: linear;
}
```

30.1.3. Обратная анимация

> ***Обратная анимация***
>
> Воспроизводится в обратном порядке — от конечного состояния к начальному (в отличие от обычной, прямой анимации).

Обратная анимация воспроизводится по окончании действия, которое производит над анимируемым элементом пользователь (например, после увода с элемента курсора мыши).

Атрибуты стиля, задающие параметры обратной анимации, помещаются, наоборот, в стиль, описывающий *начальное состояние анимации*.

Пример CSS-стиля, задающего обратную анимацию, которая воспроизводится в течении 5 с с задержкой в 1 с:

```
.animated5 {
    position: absolute;
    left: 10px;
    top: 20px;
    transition-duration: 5s;
    transition-delay: 1s;
}
.animated5:hover {
    left: 200px;
    transition-duration: 3s;
}
```

30.1.4. Сложная анимация с двумя состояниями

Сложная анимация с двумя состояниями состоит из произвольного количества простых анимаций такого рода. Каждая из простых анимаций включает отдельный анимируемый атрибут стиля, имеет свои значения продолжительности анимации, задержки перед ее началом и свой закон протекания.

Для создания сложной анимации достаточно записать параметры простых анимаций, входящих в ее состав, в атрибутах стиля, описанных в *разд. 30.1.1* и *30.1.2*, через запятую:

- в атрибуте стиля `transition-property` — сначала имя анимируемого атрибута стиля из первой анимации, потом, через запятую, имя анимируемого атрибута стиля из второй простой анимации и т. д.;
- в атрибуте стиля `transition-duration` — сначала продолжительность первой простой анимации, потом, через запятую, продолжительность второй простой анимации и т. д.;
- и т. д.

Пример сложной анимации, в которой анимируется горизонтальная координата в течение 3 с, потом цвет фона — в течение 4 с и, наконец, вертикальная координата — в течение 2 с:

```
.animated6 {
    position: absolute;
    left: 10px;
    top: 20px;
    background-color: grey;
}
.animated6:hover {
    left: 200px;
    top: 40px;
    background-color: yellow;
    transition-property: left, background-color, top;
    transition-duration: 3s, 4s, 2s;
}
```

Если в атрибуте стиля `transition-duration`, `transition-delay` или `transition-timing-function` указано параметров меньше, чем имен атрибутов стиля в атрибуте стиля `transition-property`, набор параметров в соответствующем атрибуте стиля будет повторен нужное количество раз.

Так, в следующем примере анимация горизонтальной координаты будет продолжаться 3 с, анимация фонового цвета — 4 с, анимация вертикальной координаты — снова 3 с, а анимация цвета текста — снова 4 с:

```
.animated7:hover {
    . . .
    transition-property: left, background-color, top, color;
    transition-duration: 3s, 4s;
}
```

30.1.5. Указание сразу всех параметров анимации с двумя состояниями

Атрибут стиля `transition` задает сразу все параметры анимации с двумя состояниями. Его значение записывается в следующем формате:

```
[<имя анимируемого атрибута стиля (transition-property)>]
<продолжительность (transition-duration)>
[<закон изменения скорости (transition-timing-function)>]
[<задержка (transition-delay)>]
```

Единственным обязательным к указанию параметром является *продолжительность* анимации.

Примеры:

```
.animated4:hover {
    . . .
    transition: left 3s linear 0.5s;
}
.animated1:hover {
    . . .
    transition: 3s;
}
```

В этом же атрибуте стиля можно записывать параметры сложной анимации — разделив параметры отдельных простых анимаций, входящих в ее состав, запятыми:

```
.animated6:hover {
    . . .
    transition: left 3s, background-color 4s, top 2s;
}
```

Отдельные параметры анимации, записанные в атрибуте стиля `transition`, можно располагать в произвольном порядке. Единственное ограничение: значение задержки перед началом анимации должно следовать *строго* за значением продолжительности анимации. Пример:

```
.animated4:hover {
    . . .
    transition: linear 3s 0.5s left;
}
```

30.2. Упражнение. Анимирование гиперссылок в панели навигации

Для практики анимируем гиперссылки в панели навигации на страницах сайта суши-бара. Сделаем так, чтобы при наведении курсора мыши они меняли цвет и сдвигались вправо за 1 с, при уводе курсора принимали изначальный цвет за 1,5 с с задержкой в 0,5 с и возвращались на исходное место за 2 с.

1. Найдем в папке 28\ex28.2 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Сначала создадим прямую анимацию. Начальное состояние у нас уже записано в стиле, применяемом к гиперссылкам из панели навигации,

а обратное — в стиле тех же гиперссылок, только находящихся под курсором мыши. Нам останется лишь добавить в стиль конечного состояния необходимые параметры анимации.

2. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и дополним стиль, который применяется к гиперссылкам панели навигации, находящимся под курсором мыши:

```
nav a:active, nav a:hover {
    . . .
    transition-property: color, background-color, translate;
    transition-duration: 1s;
}
```

Да-да, атрибуты стиля, задающие преобразования элементов (см. *уроки 28* и *29*), также можно анимировать.

Мы использовали особенность атрибутов стиля, задающих параметры анимации, которая описана в *разд. 30.1.4*, чтобы указать у всех анимируемых атрибутов одно значение продолжительности анимации. Таким образом мы немного упростили себе работу.

✓ Что у нас получилось? ⌄

Приведены фазы анимации гиперссылки **Прайс-лист**, на которую наводится курсор мыши.

Обратную анимацию запишем в стиле гиперссылок панели навигации, пребывающих в обычном состоянии.

3. Дополним стиль гиперссылок панели навигации в обычном состоянии:

```
nav a {
    . . .
    transition-property: color, background-color, translate;
    transition-duration: 1.5s, 1.5s, 2s;
    transition-delay: .5s, .5s, 0;
}
```

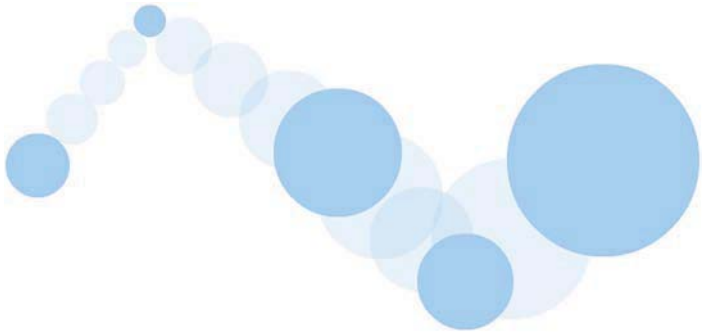
✓ Что у нас получилось? ⌄

Приведены фазы анимации гиперссылки **Прайс-лист**, с которой уводится курсор мыши.

Анимация — весьма впечатляющий инструмент, позволяющий заметно оживить веб-страницу. И мы только что убедились в этом на собственном опыте.

30.3. Анимация с несколькими состояниями

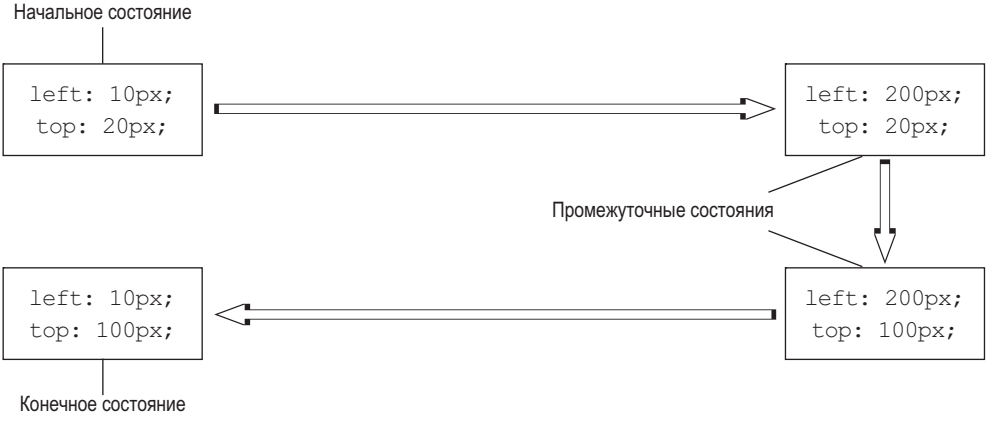
Реализовав *анимацию с несколькими состояниями*, мы можем заставить элемент двигаться по более сложной траектории...



...и (или) в указанные моменты времени изменять цвет фона с одного на другой, а потом — на третий.

> ***Анимация с несколькими состояниями***
>
> Анимация, включающая, помимо начального и конечного, произвольное количество промежуточных состояний.
>
> ***Промежуточное состояние анимации***
>
> Задает параметры анимируемого элемента на указанной отметке времени анимации между начальным и конечным состоянием. Количество промежуточных состояний не ограничено.

Пример анимации с четырьмя состояниями, при которой элемент перемещается между точками с координатами: [10, 20], [200, 20], [200, 100] и [10, 100]:



Все состояния такой анимации: начальное, промежуточные и конечное — описываются в виде *набора состояний* отдельно от каких бы то ни было стилей. Набору состояний дается уникальное имя.

Каждое из состояний анимации, содержащееся в наборе, должно включать атрибуты стиля, которые описывают это состояние. В состав состояния включают лишь те атрибуты стиля, значения которых изменяются.

Все параметры анимации с несколькими состояниями, включая имя описывающего ее набора состояний, продолжительность и пр., записываются в стиле, применяемом к анимируемому элементу. Для задания параметров анимации применяются специальные атрибуты стиля, описываемые далее.

Преимущества и недостаток анимации с двумя состояниями:

Преимущества	Недостаток
• Возможность реализации достаточно сложных анимационных эффектов. • Может быть запущена как в ответ на действие пользователя, так и автоматически, сразу после загрузки страницы.	• Создается несколько сложнее, чем анимация с двумя состояниями.

30.3.1. Описание набора состояний анимации

Набор состояний для анимации описывается посредством директивы `@keyframes` в следующем формате:

```
@keyframes <имя набора состояний> {
    <состояния>
}
```

Имя набора состояний должно быть уникальным, содержать лишь буквы, цифры, символы дефиса и подчеркивания. В качестве *имени* нельзя использовать слова `none` (предопределенное значение, рассмотренное в *разд. 30.3.2*), `initial`, `inherit` и `unset` (специальные величины, описанные в *разд. 12.1.3*).

Отдельное *состояние* анимации записывается в формате:

```
<отметка времени> {
    <атрибуты стиля, описывающие состояние>
}
```

В качестве *отметки времени* можно задать:

- числовую величину в % от общей продолжительности анимации;
- from — начало анимации (то же самое, что и 0%);
- to — конец анимации (то же самое, что и 100%).

Атрибуты стиля, описывающие состояние, записываются по тем же правилам, что и в случае обычного стиля.

Описание набора состояний должно быть расположено в таблице стилей перед первым стилем, который ссылается на этот набор. Обычно это описание ставят в начале таблицы стилей.

Пример CSS-кода, который описывает анимацию, показанную на рисунке ранее:

```
@keyframes anim8 {
    from {
        left: 10px;
        top: 20px;
    }
    33% {
        left: 200px;
        top: 20px;
    }
    67% {
        left: 200px;
        top: 100px;
    }
    to {
        left: 10px;
        top: 100px;
    }
}
```

30.3.2. Создание анимации с несколькими состояниями

У анимации с несколькими состояниями имеются две важных настройки: продолжительность и имя набора состояний, описывающего ее.

Для задания имени набора состояний анимации применяется атрибут стиля animation-name. Его значением может быть:

- имя набора состояний анимации;
- none — тогда элемент не будет анимирован[1].

[1] Именно поэтому в качестве имени набора состояний нельзя использовать слово none — веб-обозреватель примет его за предопределенное значение, отменяющее анимацию.

Значение по умолчанию: `none`.

Для указания остальных параметров анимации с несколькими состояниями применяется ряд атрибутов стиля, аналогичных используемым для создания анимации с двумя состояниями (см. *разд. 30.1.1* и *30.1.2*):

- `animation-duration` — продолжительность анимации, аналогичен атрибуту стиля `transition-duration` и поддерживает такие же значения;
- `animation-delay` — задержка перед началом анимации, аналогичен `transition-delay`;
- `animation-timing-function` — закон изменения скорости анимации, аналогичен `transition-timing-function`.

Пример:

```
.animated8 {
    position: absolute;
    animation-name: anim8;
    animation-duration: 4s;
}
```

Атрибут стиля `animation-direction` задает направление воспроизведения анимации. Вот поддерживаемые им значения:

- `normal` — анимация воспроизводится в прямом направлении: от начального состояния к конечному;
- `reverse` — в обратном направлении: от конечного состояния к начальному.

Значение по умолчанию: `normal`.

Пример анимации, воспроизводящейся в обратном направлении:

```
.animated9 {
    position: absolute;
    animation-name: anim8;
    animation-duration: 4s;
    animation-direction: reverse;
}
```

Еще два значения атрибута стиля `animation-direction`, применяемые при создании повторяющейся анимации, мы рассмотрим в *разд. 30.3.3*.

30.3.3. Создание повторяющейся анимации

По умолчанию анимация с несколькими состояниями выполняется лишь один раз. Однако ее можно выполнить многократно.

Атрибут стиля `animation-iteration-count` задает количество повторов анимации. Значения:

- неотрицательное целое или вещественное число без единицы измерения — непосредственно количество повторов.

  Если указано вещественное число, анимация во время последнего повтора будет воспроизведена не полностью. Так, если указать число `2.7`, то анимация два раза выполнится целиком и в третий раз — на 70%;

- `infinite` — бесконечное воспроизведение.

Значение по умолчанию: 1.

При создании повторяющейся анимации могут пригодиться еще два значения, поддерживаемые атрибутом стиля `animation-direction`, указывающим направление воспроизведения анимации:

- `alternate` — анимация воспроизводится сначала в прямом направлении, потом в обратном;
- `alternate-reverse` — сначала в обратном направлении, потом в прямом.

Пример бесконечной анимации, воспроизводящейся то в прямом, то в обратном направлении:

```
.animated10 {
    position: absolute;
    animation-name: anim8;
    animation-duration: 4s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
}
```

30.3.4. Дополнительные параметры анимации с несколькими состояниями

Атрибут стиля `animation-fill-mode` указывает, будут ли какие-либо состояния анимации применяться к элементу перед началом анимации (во время задержки, заданной в атрибуте стиля `animation-delay`) и после ее окончания. Его значения:

- `none` — никакие состояния анимации к элементу не применяются. Оформление элемента перед началом или после окончания анимации задается обычными стилями, действующими на этот элемент.

Рассмотрим пример анимации, которая воспроизводится с задержкой в 2 с:

```
.animated11 {
    position: absolute;
    animation-name: anim8;
    animation-duration: 4s;
    animation-delay: 2s;
}
```

В этом случае до начала анимации на элемент будет действовать лишь стиль со стилевым классом `animated11`, который задает абсолютное позиционирование, но не указывает местоположение элемента, в результате чего элемент будет помещен в левый верхний угол родителя. А когда заданная задержка истечет, и анимация начнет воспроизводиться, элемент «перепрыгнет» в точку, указанную в начальном состоянии, что будет выглядеть очень некрасиво.

Избежать этого можно, задав нужное положение элемента в стиле со стилевым классом `animated11`:

```
.animated11 {
    . . .
    left: 10px;
    top: 20px;
}
```

- `backwards` — перед началом анимации на элемент будет действовать начальное состояние анимации. Соответственно, уже во время задержки элемент окажется, что называется, на низком старте.

  Пример анимации, полностью аналогичной приведенной ранее, только реализованной по-другому:

```
.animated12 {
    position: absolute;
    animation-name: anim8;
    animation-duration: 4s;
    animation-delay: 2s;
    animation-fill-mode: backwards;
}
```

- `forwards` — после окончания анимации на элемент будет действовать конечное состояние. В результате элемент останется в том состоянии, какое было на момент завершения анимации;
- `both` — комбинация `backwards` и `forwards`.

Значение по умолчанию: `none`.

Атрибут стиля `animation-play-state` управляет текущим статусом анимации — воспроизводится ли она или приостановлена. Вот значения, поддерживаемые атрибутом:

- `running` — анимация воспроизводится;
- `paused` — анимация приостановлена.

Значение по умолчанию: `running`.

Пример анимации, приостанавливающейся при наведении курсора мыши на анимируемый элемент:

```
.animated13 {
    position: absolute;
    animation-name: anim8;
    animation-duration: 4s;
}
.animated13:hover { animation-play-state: paused; }
```

Если в составе анимируемых атрибутов стиля присутствуют `filter` (см. *разд. 27.1*), `backdrop-filter` (см. *разд. 27.2*) или `transform` (см. *уроки 28* и *29*), полезным может оказаться атрибут стиля `animation-composition`. Он определяет закон, согласно которому будут объединяться значения этих атрибутов стиля, заданные в состояниях анимации, со значениями тех же атрибутов стиля, записанных в обычном стиле.

Тестовый пример:

```
@keyframes anim14 {
    from { transform: translate(50px); }
    to { transform: translate(150px); }
}
.animated14 {
    transform: translate(100px) rotate(15deg);
    animation-name: anim14;
    animation-duration: 1s;
}
```

Значения атрибута стиля `animation-composition`:

- `replace` — значения атрибута стиля, заданные в состояниях анимации, заменяют собой значения, представленные в обычном стиле:

```
.animated14 { animation-composition: replace; }
/*
    Результирующие значения атрибута стиля transform в начале
    и конце анимации:
        * в начале — translate(50px);
        * в конце — translate(150px).
*/
```

- `add` — значения атрибута стиля, заданные в состояниях анимации, просто добавляются к значениям из обычного стиля:

```
.animated14 { animation-composition: add; }
/*
    Результирующие значения атрибута стиля transform в начале
    и конце анимации:
        * в начале — translate(100px) rotate(15deg) translate(50px);
        * в конце — translate(100px) rotate(15deg) translate(150px).
*/
```

- `accumulate` — значения параметров отдельных фильтров или преобразований складываются:

```
.animated14 { animation-composition: accumulate; }
/*
    Результирующие значения атрибута стиля transform в начале
    и конце анимации:
        * в начале — translate(150px) rotate(15deg);
        * в конце — translate(250px) rotate(15deg).
*/
```

Значение по умолчанию: `replace`.

30.3.5. Сложная анимация с несколькими состояниями

Сложная анимация с несколькими состояниями включает в себя несколько простых анимаций такого рода. Каждая из простых анимаций может описываться своим набором состояний, иметь собственные значения продолжительности, задержки перед началом, направления воспроизведения и пр.

Сложная анимация с несколькими состояниями создается по тем же принципам, что и сложная анимация с двумя состояниями (см. *разд. 30.1.4*) — указанием параметров простых анимаций, входящих в ее состав, в соответствующих атрибутах стиля через запятую.

Пример сложной анимации, включающей две простые анимации, из которых первая изменяет ширину элемента и имеет продолжительность 10 с, а вторая меняет цвет фона элемента и длится 4 с:

```
@keyframes anim15-1 {
    from { width: 100px; }
    to { width: 400px; }
}
@keyframes anim15-2 {
    from { background-color: red; }
    33% { background-color: green; }
```

```
    67% { background-color: blue; }
    to { background-color: yellow; }
}
.animated15 {
    animation-name: anim15-1, anim15-2;
    animation-duration: 10s, 4s;
    animation-timing-function: linear;
    animation-iteration-count: infinite;
    animation-direction: alternate;
}
```

Любопытно, что простые анимации, имеющие разную продолжительность, будут воспроизводиться независимо друг от друга. Так, в нашем примере анимированный элемент, увеличившись до конечной ширины (заданной в конечном состоянии из первого набора), успеет сменить цвет с изначального красного на зеленый, синий, желтый, снова на синий, зеленый, красный, зеленый и еще раз на синий.

30.3.6. Указание сразу всех параметров анимации с несколькими состояниями

Записать сразу все параметры анимации с несколькими состояниями можно в атрибуте стиля `animation` — в следующем формате:

```
<продолжительность (animation-duration)>
[<закон изменения скорости (animation-timing-function)>]
[<задержка (animation-delay)>]
[<количество повторов (animation-iteration-count)>]
[<направление (animation-direction)>]
[<применяемые состояния (animation-fill-mode)>]
[<статус (animation-play-state)>] <имя набора состояния (animation-name)>
```

Обязательны к указанию лишь *продолжительность* анимации и *имя набора состояний*.

Примеры:

```
.animated10 {
    . . .
    animation: 4s infinite alternate anim8;
}
.animated12 {
    . . .
    animation: 4s 2s backwards anim8;
}
```

```
.animated8 {
    . . .
    animation: 4s anim8;
}
```

Этот атрибут стиля можно применять и для описания сложной анимации — разделив параметры отдельных простых анимаций, входящих в ее состав, запятыми:

```
.animated15 {
    animation: 10s linear infinite alternate animated15-1,
               4s linear infinite alternate animated15-2;
}
```

Отдельные параметры анимации в атрибуте стиля `animation` можно располагать в произвольном порядке. Единственное ограничение: значение задержки перед началом анимации должно следовать *строго* за значением продолжительности анимации. Пример:

```
.animated10 {
    . . .
    animation: anim8 infinite alternate 4s;
}
```

30.4. Упражнение. Анимирование шапки и подписей у видеокиоска

При выполнении *упражнения 22.3* мы добавили в видеокиоск подписи с пояснениями. Однако все равно киоск выглядит как-то блекло... Давайте анимируем шапку и подписи, сделав так, чтобы они как бы всплывали из глубины, при этом одновременно проявляясь.

1. Найдем в папке 27\s27.3 сопровождающего книгу файлового архива (см. *приложение 4*) папку kiosk и скопируем ее куда-либо на локальный диск.

   «Всплытие» шапки и подписей реализуем с помощью трехмерных преобразований (см. *урок 29*).

   Для этого у общего родителя шапки и обоих подписей — секции тела страницы — укажем глубину перспективы. Сделаем ее достаточно большой — 1000 пунктов, чтобы элементы при применении к ним трехмерных преобразований не слишком искажались.

2. Откроем таблицу стилей styles\main.css в текстовом редакторе и дополним стиль секции тела страницы необходимым кодом:

```
body {
    . . .
    perspective: 1000pt;
}
```

Начнем с анимирования шапки.

Первым делом необходимо записать набор состояний для требуемой анимации. В начале анимации, на отметке 0%, шапка будет смещена на 100 пикселов «вглубь» (значение подобрано опытным путем) и будет полностью прозрачной. На отметке 5% она «всплывет» на нулевую «глубину» и станет полностью непрозрачной. В этом положении шапка останется до достижения отметки 45%, после чего начнет «опускаться» на «глубину» 100 пикселов и достигнет ее на отметке 50%. После чего останется пребывать там до конца анимации — до отметки 100%.

Анимация будет выглядеть более впечатляющей, если шапка станет «всплывать» из правого верхнего угла области просмотра. Чтобы достигнуть такого эффекта, мы в начале дополнительно сместим шапку по горизонтали на 50 пикселов вправо и на 40 пикселов вверх (значения подобраны экспериментально).

3. Вставим в начало таблицы стилей код, создающий набор состояний для анимации, которая будет применена к шапке:

```
@keyframes heading-anim {
    from {
        transform: translate3d(50px, -40px, -100px);
        opacity: 0.0;
    }
    5% {
        transform: none;
        opacity: 1.0;
    }
    45% {
        transform: none;
        opacity: 1.0;
    }
    50% {
        transform: translate3d(50px, -40px, -100px);
        opacity: 0.0;
    }
```

```
    to {
        transform: translate3d(50px, -40px, -100px);
        opacity: 0.0;
    }
}
```

Для указания смещений по всем трем координатным осям применяем функцию `translate3d()` (см. *разд. 29.3*), а для задания уровня непрозрачности — атрибут стиля `opacity` (см. *разд. 14.8*). Это позволит сделать код более компактным.

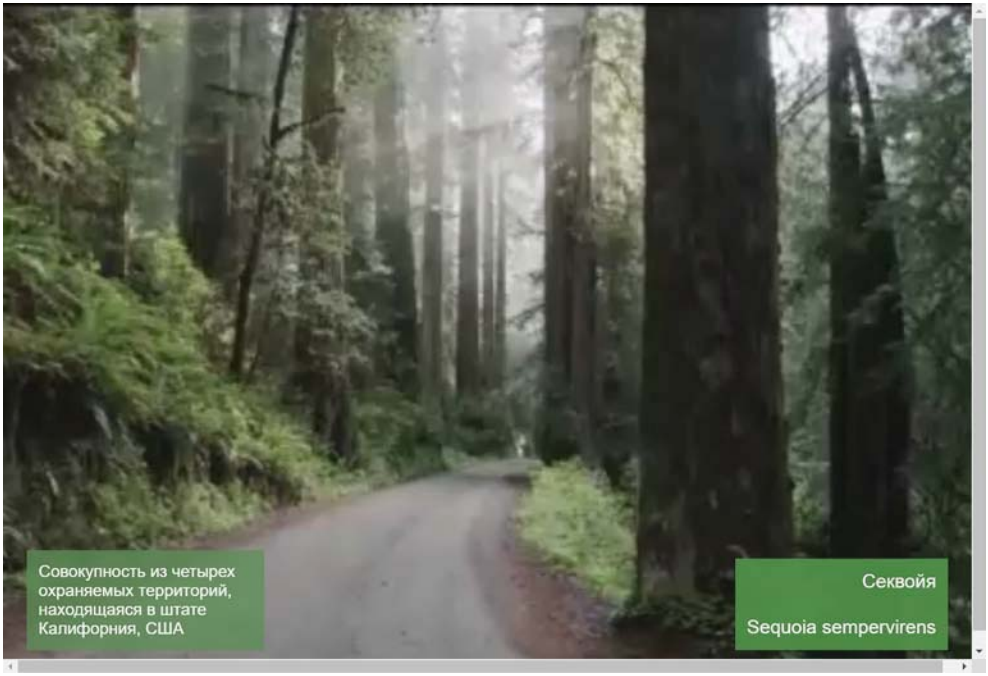
Осталось применить анимацию, описанную этим набором состояний, к шапке. Видео, воспроизводящееся в киоске, длится 29 с — такой и укажем продолжительность анимации. И сделаем ее бесконечной.

4. Вставим в стиль шапки необходимый код:

```
.heading {
    . . .
    animation: 29s infinite heading-anim;
}
```

Для уменьшения объема набираемого CSS-кода используем «всеобъемлющий» атрибут стиля `animation` (см. *разд. 30.3.6*).

✓ Что у нас получилось? ⌄



Шапка красиво «всплывает» и «погружается» в заданные нами отметки времени. Вот только справа и снизу появились полосы прокрутки, портящие всю картину... Это произошло из-за того, что шапка при смещении по горизонтальной и вертикальной осям выходит за границы области просмотра, вследствие чего возникает переполнение (см. *разд. 15.8*).

Убрать полосы прокрутки можно, предписав секции тела скрывать содержимое, выходящее за ее пределы. Сделаем это.

5. Дополним стиль секции тела страницы:

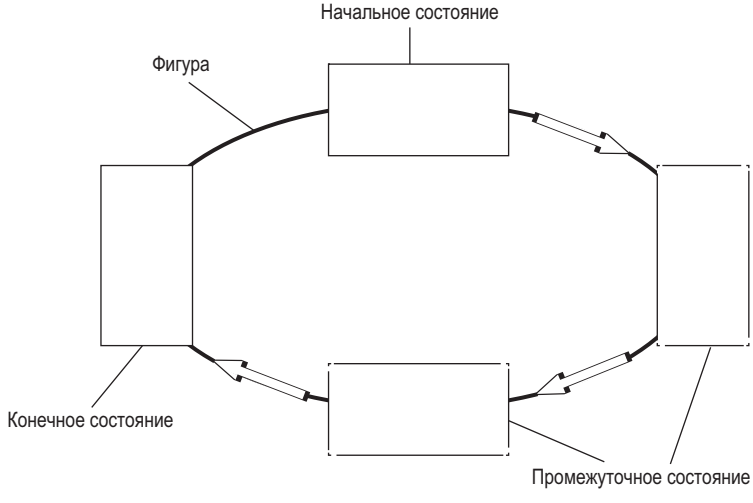
```
body {
    . . .
    overflow: hidden;
}
```

Вот теперь все замечательно! А обе подписи вы, уважаемые читатели, потом анимируете сами — аналогичным образом.

30.5. Анимация по фигуре

> ***Анимация по фигуре***
>
> Разновидность анимации с несколькими состояниями, при которой элемент перемещается по заданной фигуре: прямоугольнику, кругу, эллипсу, многоугольнику или пути.

Пример анимации по фигуре, представляющей собой эллипс:



Фигура, задающая маршрут для перемещения анимируемого элемента, создается средствами, описанными в *разд. 26.2.1.*

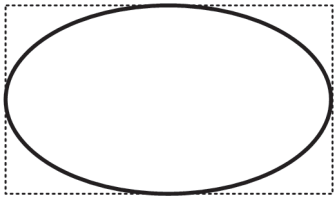
Анимация по фигуре, как и обычная анимация с несколькими состояниями, описывается набором состояний. Состояния, входящие в этот набор, должны содержать местоположение анимируемого элемента на заданной фигуре в заданные отметки времени (указание местоположение элемента на фигуре описано в *разд. 30.5.2*).

30.5.1. Описание фигуры, задающей маршрут для анимации

Описание фигуры, которая задаст маршрут для анимируемого элемента, указывается в атрибуте стиля `offset-path`. Он записывается в стиле, применяемом к анимируемому элементу. Значением этого атрибута стиля может быть:

- описание фигуры — заданное:
  - одной из функций: `inset()`, `rect()`, `xywh()`, `circle()`, `ellipse()`, `polygon()` (см. *разд. 26.2.1.1*) или `path()` (см. *разд. 26.2.1.2*). Координаты фигур и примитивов указываются относительно левого верхнего угла родителя анимируемого элемента.

Пример	Результат[1]

`.animated16 {`
`  offset-path: ellipse(100px 75px);`
`}`
	

  - функцией `ray()` — прямая линия, проведенная под указанным *углом* из точки с заданными *координатами* и имеющая указанную *длину*. Формат записи функции:

```
ray(<угол> [<длина>] [contain]
           [at <гор. коорд. начала> <верт. коорд. начала>])
```

    Угол задается в виде числового значения в любой из единиц измерения. Он откладывается от вертикальной оси: положительные значения — по часовой стрелке, отрицательные — против часовой стрелки.

    Длина проводимой линии задается в виде одного из следующих слов:

    - `closest-side` — расстояние от начала линии до ближайшей стороны родителя;
    - `closest-corner` — до ближайшего угла родителя;

[2] Здесь и далее пунктирной рамкой показаны границы родителя анимируемого элемента.

- `farthest-side` — до самой дальней стороны родителя;
- `farthest-corner` — до самого дальнего угла родителя;
- `sides` — до точки, где проводимая линия пересекается со стороной родителя.

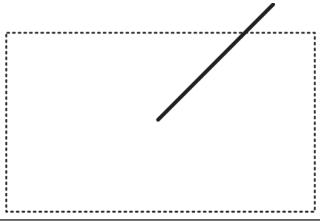
Если *направление* не указано, оно получит значение `closest-side`.

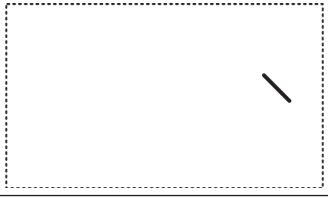
Координаты начала проводимой линии задаются в том же формате, что и значение атрибута стиля `object-position` (см. *разд. 17.3*). Если они не указаны, будут взяты координаты, заданные в атрибуте стиля `offset-position` (описан далее), а если и он не указан, начало будет находиться в центре родителя.

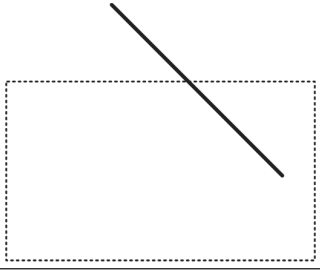
Слово `contain`, будучи указанным, предписывает веб-обозревателю уменьшить длину проведенной прямой таким образом, чтобы анимируемый элемент не вышел за границы родителя.

Пример	Результат

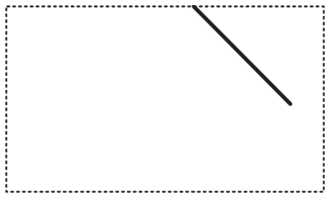
`.animated17 {`
`  offset-path: ray(45deg);`
`}`
	

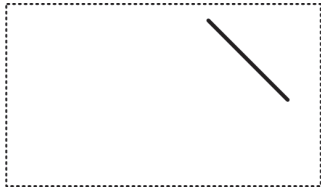
`.animated18 {`
`  offset-path: ray(45deg closest-corner);`
`}`
	

`.animated19 {`
`  offset-path: ray(-45deg closest-side`
`  at 180px 75px);`
`}`
	

`.animated20 {`
`  offset-path: ray(-45deg farthest-side`
`  at 180px 75px);`
`}`
	

Пример	Результат

`.animated21 {`
`  offset-path: ray(-45deg sides`
`  at 180px 75px);`
`}`
	

`.animated22 {`
`  offset-path: ray(-45deg sides contain`
`  at 180px 75px);`
`}`
	

- `none` — анимация по фигуре не будет создана.

Значение по умолчанию: `none`.

Если для создания фигуры используется функция `ray()` без указания начала проводимой прямой, то ее начало можно задать в атрибуте стиля `offset-position`. Поддерживаются следующие значения:

- местоположение начала прямой в формате, применяемом в атрибуте стиля `object-position` (см. *разд. 17.3*).

Пример	Результат

`.animated23 {`
`  offset-path: ray(-45deg closest-side);`
`  offset-position: 180px 75px;`
`}`
	

- `auto` — начало находится в левом верхнем углу родителя (то же самое, что и `0 0`);
- `normal` — начало находится в центре родителя (то же самое, что и `50% 50%`).

Значение по умолчанию: `normal`.

30.5.2. Описание набора состояний для анимации по фигуре

Набор состояний для анимации по фигуре пишется так же, как и для обычной анимации с несколькими состояниями (см. *разд. 30.3.1*).

Для указания положения анимируемого элемента на заданной фигуре следует применять атрибут стиля `offset-distance`. В качестве значения указывается расстояние от начала фигуры в виде числовой величины в любой поддерживаемой единице измерения (наиболее часто используются проценты). Значение по умолчанию: `0`.

Пример:

```
@keyframes anim16 {
    from { offset-distance: 0; }
    60% { offset-distance: 50%; }
    to { offset-distance: 100%; }
}
```

30.5.3. Создание анимации по фигуре

Анимация по фигуре создается так же, как и обычная анимация с несколькими состояниями, — с помощью атрибутов стиля «семейства» `animation` (см. *разд. 30.3*).

Не забываем указать в стиле, применяемом к анимируемому элементу и содержащем параметры анимации, фигуру, по которой будет двигаться элемент, — с помощью атрибута стиля `offset-path` и, при необходимости, `offset-position` (см. *разд. 30.5.1*).

Пример:

```
.animated16 {
    offset-path: ellipse(100px 75px);
    animation: 4s anim16;
}
```

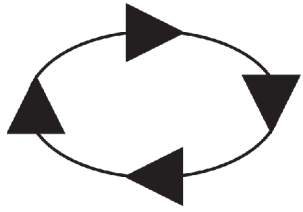
30.5.4. Дополнительные параметры анимации по фигуре

По умолчанию анимируемый элемент ориентируется веб-обозревателем по контуру фигуры и закрепляется на ней своим центром. Однако это поведение можно изменить.

Атрибут стиля `offset-rotate` задает угол, на который анимируемый элемент будет поворачиваться при движении по заданной фигуре. В качестве значения можно указать:

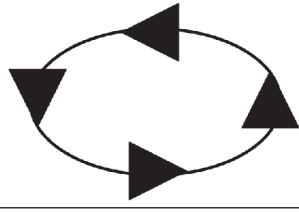
- `auto` — анимируемый элемент будет ориентирован по контуру фигуры. Угол, на который станет поворачиваться элемент, вычисляется веб-обозревателем самостоятельно.

Пример	Результат

`.animated24 {`
`  offset-path: ellipse(100px 75px);`
`  animation: 4s linear infinite anim16;`
`  offset-rotate: auto;`
`}`
	

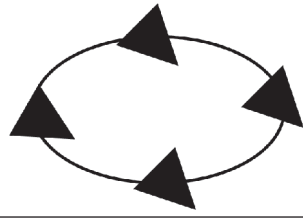
- `reverse` — то же самое, что и `auto`, только элемент будет развернут в противоположном направлении.

Пример	Результат

`.animated25 {`
`  offset-path: ellipse(100px 75px);`
`  animation: 4s linear infinite anim16;`
`  offset-rotate: reverse;`
`}`
	

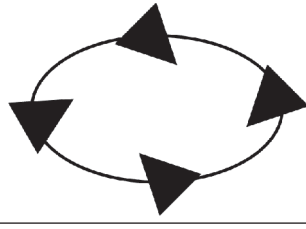
- *<угол>* — элемент всегда будет повернут на заданный *угол*. *Угол* отсчитывается от горизонтальной оси, положительные значения — по часовой стрелке, отрицательные — против часовой стрелки.

Пример	Результат

`.animated26 {`
`  offset-path: ellipse(100px 75px);`
`  animation: 4s linear infinite anim16;`
`  offset-rotate: 45deg;`
`}`
	

- `auto` *<угол>* — то же самое, что и `auto`, только элемент будет дополнительно повернут на заданный *угол*.

Пример	Результат

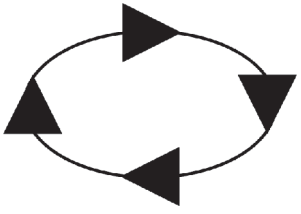
`.animated27 {`
`  offset-path: ellipse(100px 75px);`
`  animation: 4s linear infinite anim16;`
`  offset-rotate: auto 45deg;`
`}`
	

Значение по умолчанию: `auto`.

Атрибут стиля `offset-anchor` задает местоположение точки на анимируемом элементе, которой он закрепляется на фигуре. В качестве значения можно указать:

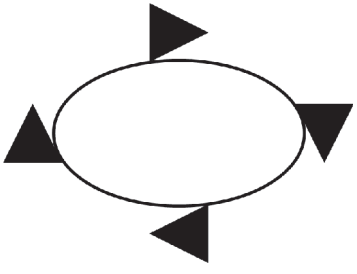
- `auto` — элемент закрепляется на фигуре точкой, местоположение которой задано в атрибуте стиля `transform-origin` (см. *разд. 28.1*), или если этот атрибут стиля не указан, своим центром.

Пример	Результат

`.animated28 {`
`  offset-path: ellipse(100px 75px);`
`  animation: 4s linear infinite anim16;`
`  offset-anchor: auto;`
`}`
	

- местоположение точки, заданное в формате, который применяется в атрибуте стиля `object-position` (см. *разд. 17.3*).

Пример	Результат

`.animated29 {`
`  offset-path: ellipse(100px 75px);`
`  animation: 4s linear infinite anim16;`
`  offset-anchor: left bottom;`
`}`
	

Значение по умолчанию: `auto`.

30.6. Самостоятельное упражнение

На странице видеокиоска анимируйте подписи аналогично шапке. Левая подпись пусть «всплывает» из левого нижнего угла страницы, правая подпись — из правого нижнего угла. Анимация левой подписи пусть начинается с задержкой в 0,5 с после начала анимации шапки, анимация правой подписи — с такой же задержкой после начала анимации левой подписи.

ЧАСТЬ IV

ДОПОЛНИТЕЛЬНЫЕ ИНСТРУМЕНТЫ HTML И CSS

- Адаптация веб-страниц под мобильные устройства.
- Фреймы.
- Всякая полезная всячина.

Урок 31. Создание адаптивных веб-страниц и их элементов

Настройка области просмотра.
Медиазапросы.
Адаптивные изображения.
Создание печатных редакций веб-страниц.

> ***Адаптивная веб-страница***
>
> Веб-страница, адаптирующаяся к характеристикам устройства, на котором она отображается: компьютера, смартфона, планшета и др.

Адаптивный элемент страницы — это элемент, адаптирующийся к размерам родителя (см. *разд. 22.2.1.1*).

31.1. Настройка области просмотра

Мобильные устройства с небольшим экраном (в основном, смартфоны) имеют странную и неприятную особенность. Подготавливая загруженную страницу к выводу, они почему-то формируют ее в таком виде, как будто собираются выводить на большом экране (аналогичном экрану обычного компьютера). А чтобы отобразить подготовленную таким образом страницу на своем экране, весьма небольших размеров, они уменьшают ее масштаб. В результате выведенная страница выглядит мелкой и совершенно нечитабельной, и пользователю для просмотра приходится ее увеличивать вручную.

Чтобы мобильное устройство этого не делало, в HTML-коде страницы необходимо указать соответствующие настройки области просмотра, а именно, задать ее ширину равной ширине экрана устройства (чтобы страница формировалась строго под экран устройства) и установить масштаб выведенной страницы равным изначальному (чтобы устройство при выводе ее не уменьшило).

Кроме того, указанные настройки области просмотра требуются также, чтобы успешно работали медиазапросы (описаны в *разд. 31.2*).

Для этого в секцию заголовка страницы (тег `<head>`) следует вставить одинарный тег `<meta>`, содержащий атрибут `name` со значением `viewport` и атрибут `content` с собственно настройками области просмотра. Последние записываются в виде набора пар формата:

<название настройки>=<значение настройки>

разделенных запятыми или комбинациями из запятой и пробела.

Пример тега `<meta>`, задающего ширину области просмотра равной ширине экрана устройства и масштаб выведенной страницы равный изначальному:

```
<head>
    . . .
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
```

Поддерживаются следующие настройки области просмотра:

- `width` — ширина области просмотра в виде:
  - неотрицательного числа от 1 до 10000 — в пикселах;
  - `device-width` — равная ширине экрана устройства;
- `height` — высота области просмотра в виде:
  - неотрицательного числа от 1 до 10000 — в пикселах;
  - `device-height` — равная высоте экрана устройства;
- `initial-scale` — масштаб страницы сразу после ее вывода в виде числа от `0.1` (10%) до `10` (1000%);
- `minimum-scale` — минимальный масштаб, до которого пользователь может уменьшить страницу, в виде числа от `0.1` до `10`. По умолчанию: `0.1`;
- `maximum-scale` — максимальный масштаб, до которого пользователь может увеличить страницу, в виде числа от `0.1` до `10`. По умолчанию: `10`;
- `user-scalable` — разрешает или запрещает пользователю масштабировать страницы. Значением может быть:
  - `0` или `no` — не может;
  - `1` или `yes` — может.

По умолчанию: `1`.

Запрет на масштабирование страницы может помешать пользователю прочитать ее, поэтому указывать его следует лишь в крайне специфических случаях;

- `interactive-widget` — указывает, что будет происходить при отображении экранной клавиатуры устройства:
  - `resizes-visual` — размеры области просмотра будут уменьшены, чтобы освободить место под вывод клавиатуры, но масштаб страницы не изменится;
  - `resizes-content` — масштаб страницы уменьшится, чтобы освободить место под вывод клавиатуры;
  - `overlays-content` — клавиатура будет выведена поверх страницы.

  По умолчанию: `resizes-visual`.

Пример, запрещающий масштабировать страницу менее 50% и более 200% и предписывающий выводить экранную клавиатуру поверх ее содержимого:

```
<head>
    . . .
    <meta name="viewport"
          content="width=device-width, initial-scale=1, ↳
                   minimum-scale=0.5, maximum-scale=2, ↳
                   interactive-widget=overlays-content">
</head>
```

31.2. Адаптация веб-страниц под параметры устройства. Медиазапросы

Есть возможность применить какие-либо стили или целые таблицы стилей только в том случае, если устройство, на котором выводится страница, удовлетворяет заданным характеристикам. Это позволяет, например, при выводе на настольном компьютере применять одно оформление, а при отображении на смартфоне — другое.

> ***Медиазапрос***
>
> Правило, применяющее заданные CSS-стили или целую таблицу стилей лишь в том случае, если устройство, на котором открывается веб-страница, удовлетворяет заданным характеристикам.

31.2.1. Две разновидности медиазапросов

Веб-стандарты предусматривают две разновидности медиазапросов, имеющих свои достоинства и недостатки.

31.2.1.1. HTML-медиазапросы — с таблицами стилей

> ***HTML-медиазапрос***
>
> При совпадении текущего устройства с заданными характеристиками применяет указанную таблицу стилей. Записывается в HTML-коде страницы.

HTML-медиазапрос представляет собой обычный тег `<link>`, посредством которого выполняется привязка внешней таблицы стилей, только содержащий атрибут `media`. В этом атрибуте тега и записывается набор характеристик, которому должно удовлетворять устройство (правила их записи рассматриваются в *разд. 31.2.2*).

Вот пример загрузки и применения таблицы стилей styles\narrow.css лишь в случае, если страница открывается на устройстве с экраном шириной не более 400 пикселов:

```
<link href="styles/narrow.css" rel="stylesheet"
      media="screen and (max-width: 400px)" />
```

Преимущество и недостатки HTML-медиазапросов:

Преимущество	Недостатки
Выполняется загрузка лишь стилей, непосредственно нужных для вывода страницы на текущем устройстве, что ускоряет вывод.	• Набор медиазапросов должен указываться в коде каждой из страниц сайта, что усложняет разработку и сопровождение. • Необходимо создавать отдельную таблицу стилей для каждой группы устройств, которые должны поддерживаться страницей, что также усложняет сопровождение.

31.2.1.2. CSS-медиазапросы — с отдельными стилями

> ***CSS-медиазапрос***
>
> При совпадении текущего устройства с заданными характеристиками применяет указанные стили. Записывается в CSS-коде таблицы стилей.

CSS-медиазапрос записывается с помощью директивы `@media`:

```
@media <набор характеристик устройства> {
    <стили, применяемые при совпадении текущего устройства ↳
     с заданным набором характеристик>
}
```

Пример указания у заголовка первого уровня кегля шрифта в 24 пункта, если страница открывается на устройстве с экраном шириной не более 400 пикселов:

```
@media screen and (max-width: 400px) {
    h1 { font-size: 24pt; }
}
```

Преимущества и недостаток CSS-медиазапросов:

Преимущества	Недостаток
• Набор медиазапросов достаточно записать лишь в одном месте — в таблице стилей. • Нет необходимости создавать отдельную таблицу стилей для каждой группы устройств, которые должны поддерживаться страницей, что упрощает сопровождение.	Для применения медиазапроса содержащая его таблица стилей должна быть загружена целиком, что может замедлить вывод страницы.

31.2.2. Запись наборов характеристик устройства

Каждый набор характеристик устройства, записываемый в медиазапросе любого типа, состоит из отдельных характеристик, разделенных особыми операторами, которые аналогичны операторам, применяемым в функции `calc()` (см. *разд. 12.2.1.3*).

31.2.2.1. Характеристики устройства

Характеристики устройства, указываемые в составе наборов характеристик, бывают простыми и сложными.

Простые характеристики задают разновидность устройства:

- `screen` — устройство, выводящее страницу на экране (компьютер, смартфон, планшет, устройство для чтения электронных книг, «умный» телевизор и т. п.):

```
/* Уменьшим шрифт, если ширина экрана не превышает 400 пикселов */
@media screen and (max-width: 400px) {
    h1 { font-size: smaller; }
}
```

- `print` — печатающее устройство:

```
/* Увеличим шрифт, если ширина страницы превышает 21 см */
@media print and (min-width: 21cm) {
    h1 { font-size: larger; }
}
```

- `all` — устройство любой разновидности.

Простая характеристика традиционно записывается самой первой, в начале набора. Если она не указана, медиазапрос будет применяться как при экранном выводе, так и при печати (как если бы была задана характеристика `all`).

Сложная характеристика записывается в формате:

```
(<название характеристики>[: <значение характеристики>])
```

Она обязательно должна быть заключена в круглые скобки. У некоторых характеристик *значение* не задается.

Вот наиболее востребованные из сложных характеристик:

- `width: <ширина>` — устройство должно иметь экран строго заданной *ширины*, которая может быть выражена в любой единице измерения CSS:

  ```
  @media screen and (width: 600px) {
      . . .
  }
  ```

  Для принтера эта характеристика указывает ширину бумаги:

  ```
  @media print and (width: 21cm) {
      . . .
  }
  ```

- `min-width: <ширина>` — устройство должно иметь экран (или печатать на бумаге) с шириной не менее указанной;
- `max-width: <ширина>` — устройство должно иметь экран (или печатать на бумаге) с шириной не более указанной;
- `height: <высота>` — устройство должно иметь экран строго заданной *высоты*, которая может быть выражена в любой единице измерения CSS:

  ```
  @media screen and (height: 800px) {
      . . .
  }
  ```

  Для принтера эта характеристика указывает высоту бумаги;

- `min-height: <высота>` — устройство должно иметь экран (или печатать на бумаге) с высотой не менее указанной;
- `max-height: <высота>` — устройство должно иметь экран (или печатать на бумаге) с высотой не более указанной;

- `resolution: <разрешение>` — устройство должно иметь экран со строго заданным *разрешением*. *Разрешение* задается в виде числа с применением единицы измерения `dpi` (точки на дюйм);
- `min-resolution: <разрешение>` — устройство должно иметь экран с разрешением не менее указанного:

```
@media screen and (min-resolution: 250dpi) {
    body { background-image: url(../images/bg-high-res.jpg); }
}
```

- `max-resolution: <разрешение>` — устройство должно иметь экран с разрешением не более указанного;
- `aspect-ratio: <соотношение сторон>` — устройство должно иметь экран со строго заданным *соотношением сторон*, записанным в формате:

  `<ширина>/<высота>`

  Ширина и *высота* задаются в относительных долях, представленных целыми числами без единиц измерения. Пример:

```
@media (aspect-ratio: 16/9) {
    . . .
}
```

- `min-aspect-ratio: <соотношение сторон>` — устройство должно иметь экран с соотношением сторон не менее заданного;
- `max-aspect-ratio: <соотношение сторон>` — устройство должно иметь экран с соотношением сторон не более заданного;
- `orientation: portrait|landscape` — устройство должно находиться (или печатать на бумаге) в указанной ориентации:
  - `portrait` — портретной (вертикальной, высота экрана больше ширины);
  - `landscape` — ландшафтной (горизонтальной, ширина экрана больше высоты).

  Пример:

```
.flex { display: flex; }
@media (orientation: portrait) {
    .flex { flex-direction: column; }
}
@media (orientation: landscape) {
    .flex { flex-direction: row; }
}
```

- `hover: hover|none` — если `hover`, основной инструмент ввода устройства должен предоставлять средства для наведения курсора на элементы страницы (например, мышь у обычного компьютера), если `none` — он не должен иметь такой функциональности:

```
/*
    Выделяем гиперссылки при наведении курсора мыши, лишь если
    устройство позволяет навести курсор на элементы
*/
@media (hover: hover) {
    nav a:active, nav a:hover {
        color: blanchedalmond;
        background-color: maroon;
    }
}
```

- `any-hover: hover|none` — если `hover`, устройство должно иметь хоть какие-то средства для наведения курсора на элементы страницы (например, мышь, подключенную к планшету), если `none` — оно не должно иметь такой функциональности;
- `pointer: fine|coarse|none` — указывает, должен ли основной инструмент ввода устройства реализовывать указующий курсор:
  - `fine` — да, точно позиционируемый (например, посредством мыши);
  - `coarse` — да, приблизительно позиционируемый (скажем, с помощью сенсорного экрана);
  - `none` — не должно (к устройствам такого рода относятся, в частности, «умные» телевизоры);
- `any-pointer: fine|coarse|none` — указывает, должно ли устройство иметь хоть какой-то инструмент для реализации указующего курсора;
- `prefers-color-scheme: light|dark` — на устройстве должна быть включена цветовая схема:
  - `light` — светлая (темный текст на светлом фоне);
  - `dark` — темная (светлый текст на темном фоне).

  Пример:

```
@media screen and (prefers-color-scheme: light) {
    body {
        color: black;
        background-color: white;
    }
}
```

```
@media screen and (prefers-color-scheme: dark) {
    body {
        color: white;
        background-color: black;
    }
}
```

- `prefers-contrast: <режим контрастности>` — на устройстве должен быть включен один из следующих режимов *контрастности*:
  - `no-preference` — обычный режим;
  - `more` — режим повышенной контрастности;
  - `less` — режим пониженной контрастности;
  - `custom` — режим контрастности, настраиваемой пользователем;
- `prefers-reduced-motion: no-preference|reduce` — на устройстве режим подавления ненужной анимации должен быть:
  - `no-preference` — отключен;
  - `reduce` — включен. В этом случае веб-обозреватель не воспроизводит анимацию, запускаемую сразу после загрузки страницы (однако анимация, выполняющаяся в ответ на действия пользователя, все же будет воспроизводиться).

  Пример запуска анимации элемента только в случае, если не задействован режим подавления ненужной анимации:

```
@media screen and (prefers-reduced-motion) {
    .animated { animation: 4s anim; }
}
```

- `update: <скорость обновления>` — устройство должно поддерживать одну из следующих *скоростей обновления* экрана:
  - `none` — вообще не должно иметь возможности обновлять выведенную страницу (например, представлять собой принтер);
  - `slow` — низкая скорость обновления (устройство чтения электронных книг или иное устройство с «медленным» экраном);
  - `fast` — высокая скорость обновления.

  Пример:

```
@media screen and (update: slow) {
    .animated { animation-duration: 30s; }
}
@media screen and (update: fast) {
    .animated { animation-duration: 3s; }
}
```

- `monochrome` — устройство должно иметь черно-белый экран или позволять печатать только в градациях серого:

```
@media (monochrome) {
    body {
        color: black;
        background-color: white;
    }
}
```

Также существуют сложные характеристики `device-width` (аналогична `width`), `min-device-width (min-width)`, `max-device-width (max-width)`, `device-height (height)`, `min-device-height (min-height)`, `max-device-height (max-height)`, `device-aspect-ratio (aspect-ratio)`, `min-device-aspect-ratio (min-aspect-ratio)` и `max-device-aspect-ratio (max-aspect-ratio)`. Они объявлены устаревшими в новых редакциях интернет-стандартов, однако все еще поддерживаются веб-обозревателями и встречаются в старом CSS-коде.

31.2.2.2. Операторы, применяемые в наборах характеристик

В наборах характеристик могут применяться следующие операторы:

- *<характеристика 1>* and *<характеристика 2>* — устройство должно совпадать *и* с *характеристикой 1*, *и* с *характеристикой 2* (оператор *логического И*):

```
/*
    Устройство должно выводить страницу на экран и иметь экран
    шириной не менее 1024 пиксела
*/
@media screen and (min-width: 1024px) {
    . . .
}
```

Произвольное количество операторов логического И можно записать друг за другом:

```
/*
    Устройство должно выводить страницу на экран и иметь экран
    шириной не менее 1024 пиксела и высотой не менее 768 пикселов
*/
@media screen and (min-width: 1024px) and (min-height: 768px) {
    . . .
}
```

- *<характеристика 1>*, *<характеристика 2>* — устройство должно совпадать *или* с *характеристикой 1*, *или* с *характеристикой 2*, *или* сразу с обеими (оператор *логического ИЛИ*):

```
/*
    Устройство должно иметь экран с низкой скоростью обновления или
    активный режим подавления ненужной анимации
*/
@media (update: slow), (prefers-reduced-motion: reduce) {
    . . .
}
```

Можно записать произвольное количество операторов логического ИЛИ:

```
/*
    Устройство должно иметь экран с низкой скоростью обновления, или
    активный режим подавления ненужной анимации, или черно-белый экран
*/
@media (update: slow), (prefers-reduced-motion: reduce), (monochrome)
{
    . . .
}
```

Оператор логического ИЛИ, записанный в виде запятой, нельзя использовать в круглых скобках (о них — далее). В этом случае следует применять альтернативную форму записи этого оператора;

- *<характеристика 1>* or *<характеристика 2>* — альтернативная форма записи оператора логического ИЛИ:

```
@media (update: slow) or (prefers-reduced-motion: reduce) {
    . . .
}
```

- not *<характеристика>* — устройство, напротив, не должно совпадать с заданной *характеристикой* (оператор *логического НЕ*, или *инверсии*). Может быть использован:
  - в самом начале набора характеристик — инвертирует *весь* набор характеристик. Следует отметить, что в начале набора необходимо поставить какую-либо из простых характеристик, приведенных в начале *разд. 31.2.2.1*. Пример:

```
/*
    Устройство не должно быть печатающим с шириной страницы 42 см
*/
@media not print and (width: 42cm) {
    . . .
}
```

- в середине набора характеристик — инвертирует лишь совокупность характеристик, стоящих *после* этого оператора:

```
/*
    Устройство должно быть печатающим, но не с шириной страницы
    42 см
*/
@media print and not (width: 42cm) {
    . . .
}
/*
    Устройство должно быть печатающим, но не с шириной страницы
    42 см и не черно-белым
*/
@media print and not (width: 42cm) and (monochrome) {
    . . .
}
```

- в начале одной из характеристик, присутствующих в составе оператора логического ИЛИ, — инвертирует лишь характеристику, в начале которой присутствует:

```
/*
    Устройство должно быть или печатающим, или с шириной экрана,
    не равной 30 000 пикселов, или черно-белым
*/
@media print, not (width: 42cm), (monochrome) {
    . . .
}
```

Наивысшим приоритетом обладает оператор логического И (`and`). Приоритет оператора логического НЕ (`not`) несколько ниже, а приоритет оператора логического ИЛИ (запятая или `or`) еще ниже.

Можно повысить приоритет какого-либо оператора, заключив его, вместе с операндами, в круглые скобки. Пример повышения приоритета оператора логического НЕ таким образом, чтобы он выполнился перед всеми операторами И:

```
/*
    Устройство должно быть печатающим, с шириной страницы, не равной
    42 см, и черно-белым
*/
@media print and (not (width: 42cm)) and (monochrome) {
    . . .
}
```

Пример повышения приоритета оператора логического ИЛИ:

```
/*
    Устройство должно иметь экран, имеющий или ширину не менее
    1024 пикселов, или высоту не менее 768 пикселов, и основной
    инструмент ввода, позволяющий наводить курсор на элементы
*/
@media ((min-width: 1024px) or (min-height: 768px)) and (hover: hover) {
    .mq6:before { content: 'mq6'; }
}
```

Следует отметить, что в круглые скобки допускается заключать только оператор ИЛИ, записанный в альтернативной форме (`or`). Оператор ИЛИ в основной форме (запятая) в круглых скобках недопустим.

31.3. Упражнение. Адаптация веб-сайта суши-бара

Адаптируем сайт суши-бара под мобильные устройства. Сделаем так, чтобы при отображении сайта на экране шириной менее 400 пикселов:

- панель навигации и основное содержимое выводились друг под другом;
- у «всеобъемлющего» блока (со стилевым классом `container`):
  - ширина совпадала с шириной области просмотра;
  - фоновое изображение с чашкой мате не выводилось (поскольку оно наверняка будет перекрыто текстом, в результате чего пользователь и толком не разглядит это изображение, и не сможет прочитать текст);
- в шапке сайта:
  - текст заголовков выводился уменьшенным кеглем;
  - фоновые изображения в левой части не выводились (все равно текст шапки налезет на них);
- на страницах каталогов — изображения и описания блюд выводились друг под другом.

1. Найдем в папке 30\ex30.2 сопровождающего книгу файлового архива (см. *приложение 4*) папку site и скопируем ее куда-либо на локальный диск.

   Для адаптации страниц мы используем медиазапросы. Чтобы они успешно работали, в HTML-коде необходимо указать настройки

области просмотра: ширину, равную ширине экрана устройства, и масштаб выведенной страницы, равный изначальному. Как это сделать, было описано в *разд. 31.1*.

2. Откроем главную страницу index.html в текстовом редакторе и вставим в ее код тег с настройками области просмотра:

```
<head>
    . . .
    <meta name="viewport"
          content="width=device-width, initial-scale=1">
    <title>Йокогама: суши-бар</title>
</head>
```

Итак, почву мы подготовили. Можно приступать к адаптации сайта.

Начнем с панели навигации и основного содержимого. Панель инструментов превратим в обычный, не плавающий блочный элемент, укажем у него ширину, внешние просветы и радиус скругления углов по умолчанию. У основного содержимого укажем значения по умолчанию у внешнего просвета слева и минимальной высоты.

3. Откроем таблицу стилей styles\2.1.css в текстовом редакторе и добавим CSS-код, создающий необходимый медиазапрос[1]:

```
@media screen and (max-width: 400px) {
    nav {
        float: none;
        width: initial;
        margin: initial;
        border-radius: initial;
    }
    main {
        margin-left: initial;
        min-height: initial;
    }
}
```

4. Откроем главную страницу в веб-обозревателе, чтобы посмотреть на результат...

...и увидим, что страница не изменилась.

[1] Опытные верстальщики при вводе медиазапроса обычно пишут вручную лишь саму директиву `@media`, после чего копируют в нее написанные ранее стили, задающие изначальное оформление, и соответственно исправляют их.

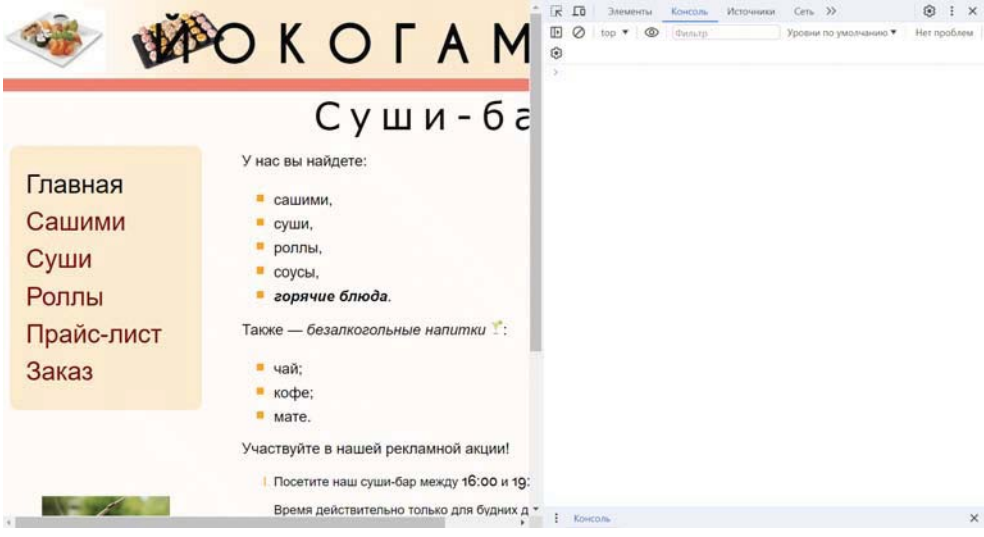
Увидеть изменения мы сможем на экране шириной 400 пикселов или менее. Уменьшить ширину окна веб-обозревателя до этого размера, к сожалению, не получится — программа просто не позволит этого сделать.

Однако веб-обозреватель можно переключить в режим эмуляции мобильного устройства с небольшим экраном. Что мы сейчас и сделаем.

Сначала нужно вывести панель инструментов разработчика, входящих в состав веб-обозревателя.

5. В окне веб-обозревателя нажмем клавишу <F12>.

Окно веб-обозревателя с панелью инструментов разработчика ⌄

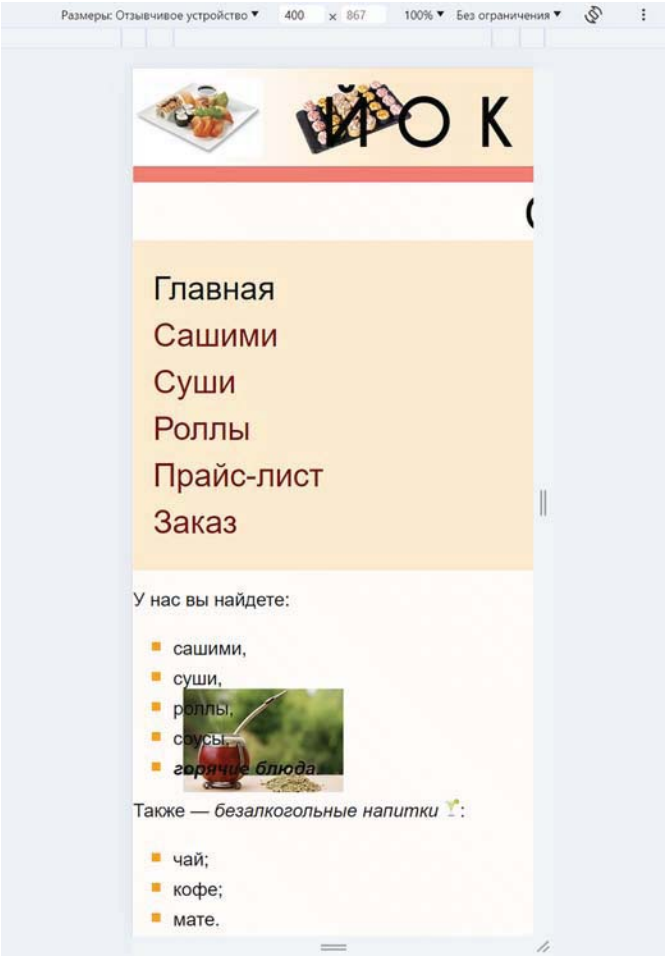


Панель инструментов разработчика изначально выводится в правой части окна веб-обозревателя (в левой все так же выводится страница). В панели находится несколько вкладок, из которых активной может быть любая (так, у автора активной оказалась вкладка **Консоль**).

Теперь можно переключить веб-обозреватель в режим эмуляции мобильного устройства.

6. Щелкнем мышью где-либо на пустом пространстве в панели инструментов разработчика, чтобы активизировать ее, и нажмем комбинацию клавиш <Ctrl>+<Shift>+<M>.

Левая часть окна, в которой выводится веб-страница в режиме эмуляции мобильного устройства ⌄



Если навести на страницу курсор мыши, он будет отображаться в виде довольно крупной серой точки. Этим курсором можно щелкать на гиперссылках, а также прокручивать страницу, буксируя ее содержимое вверх или вниз.

Потом мы еще «поиграем» с веб-обозревателем, эмулирующим мобильное устройство. А пока что посмотрим на результат наших трудов. Видно, что результат положительный — панели навигации и основное содержимое выводятся друг над другом.

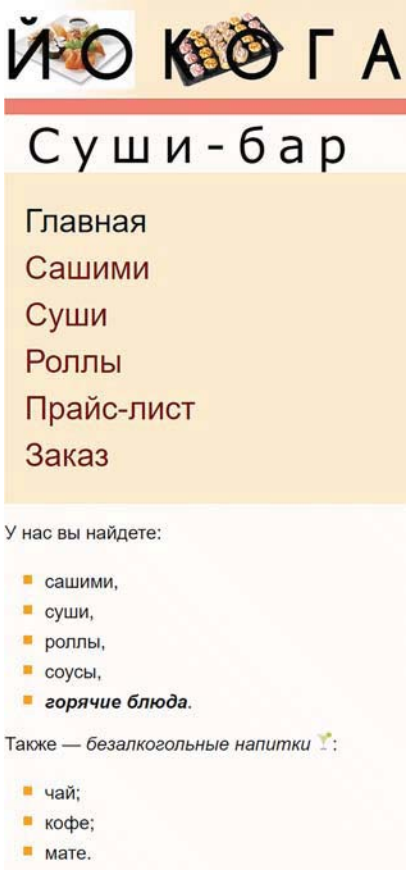
Займемся «всеобъемлющим» блоком, имеющим стилевой класс `container`. Для него укажем значения по умолчанию у минимальной

и максимальной ширины, ширину, равную ширине области просмотра, и наличие лишь полупрозрачного белого фона.

7. Добавим в написанный ранее медиазапрос стиль для блока со стилевым классом `container`:

```
@media screen and (max-width: 400px) {
    . . .
    div.container {
        min-width: initial;
        max-width: initial;
        width: 100vw;
        background: rgba(255, 255, 255, 0.8);
    }
}
```

✓ Что у нас получилось? ›



Показана только страница.

Так как мы задали ширину «всеобъемлющего» блока равной ширине области просмотра, в шапке вновь появились заголовки, «уехавшие» ранее вправо и скрывшиеся за границей видимости. Также пропала картинка с чашкой мате, мешающая читать текст.

Приступим к работе над шапкой, точнее, над находящимися в ней заголовками первого уровня. У верхнего заголовка, со стилевым классом `header1`, уменьшим кегль шрифта до 42 пунктов, зададим значение по умолчанию у интервала между буквами и только градиентный фон. У нижнего заголовка, со стилевым классом `header2`, убавим кегль до 32 пунктов и также сделаем интервал между символами по умолчанию.

8. Добавим в медиазапрос стили для заголовков первого уровня со стилевыми классами `header1` и `header2`, находящихся в семантической шапке:

```
@media screen and (max-width: 400px) {
    header h1.header1 {
        font: bold 42pt 'Caviar Dreams';
```

```
        letter-spacing: initial;
        background:
            radial-gradient(circle farthest-side at right,
                            blanchedalmond, blanchedalmond 67%,
                            white 90%);
    }
    header h1.header2 {
        font: 32pt Verdana;
        letter-spacing: initial;
    }
}
```

✓ Что у нас получилось? ›

Показана только шапка страницы.

Осталось разобраться со страницами каталогов. Начнем с каталога сашими. Как мы помним, собственно каталог представляет собой таблицу.

9. Откроем страницу каталога сашими pages\sashimi.html в текстовом редакторе и вставим в ее код тег с настройками области просмотра, аналогичный написанному на *шаге 2*.

   Чтобы предписать таблице, ее строкам и ячейкам выводиться строго друг под другом, достаточно превратить их в блочные элементы. Как это сделать, описано в *разд. 21.1*.

   Первая ячейка каждой строки таблицы-каталога, содержащая фотографию очередного блюда, имеет ширину, равную 33% от ширины таблицы. При адаптации имеет смысл задать у этих ячеек ширину по умолчанию, поскольку картинки, занимающие $^1/_3$ ширины экрана и выровненные по левому краю, выглядят некрасиво.

10. Переключимся на таблицу стилей и добавим в медиазапрос необходимые стили для таблицы со стилевым классом `catalog`, ее строк, ячеек и для первых ячеек строк этой же таблицы:

```
@media screen and (max-width: 400px) {
    . . .
    table.catalog, table.catalog tr, table.catalog tr td {
        display: block;
    }
    table.catalog tr td:first-of-type {
        width: initial;
    }
}
```

✓ Что у нас получилось? ›

Показан только каталог сашими в уменьшенном масштабе.

11. Запишите тег настроек области просмотра в код остальных страниц сайта самостоятельно.

    Закончив работу и проверив, правильно ли отображаются страницы адаптированного сайта, следует отключить в веб-обозревателе режим эмуляции мобильного устройства и закрыть панель инструментов разработчика.

12. Щелкнем мышью где-либо на пустом пространстве в панели инструментов разработчика и нажмем комбинацию клавиш <Ctrl>+<Shift>+<M>.

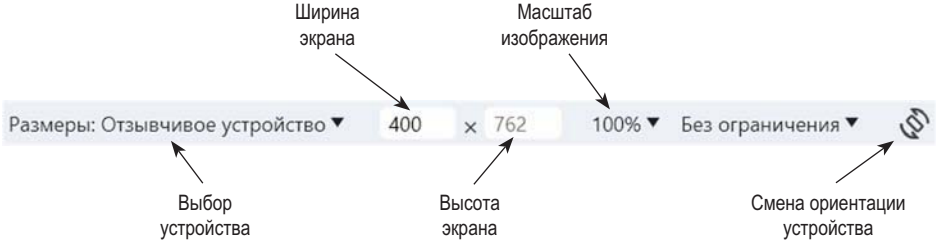
    Режим эмуляции мобильного устройства отключится.

13. Нажмем клавишу <F12>.

    Панель инструментов разработчика веб-обозревателя будет скрыта.

- Если нажать клавишу <F12>, не отключая режим эмуляции мобильного устройства явно, этот режим отключится автоматически после скрытия панели инструментов разработчика. Однако при повторном выводе этой панели эмуляция будет задействована снова.

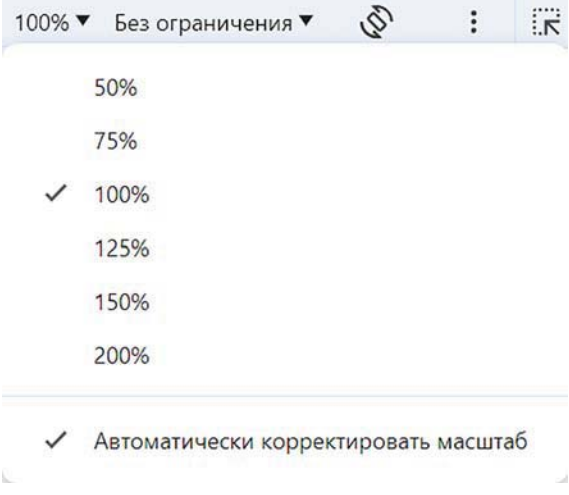
- Когда активен режим эмуляции мобильного устройства, в верхней части левой половины окна веб-обозревателя (где отображается страница) находится панель инструментов с несколькими полезными элементами управления.



- Меню выбора устройства позволяет выбрать устройство, которое нужно эмулировать. Пункт **Отзывчивое устройство** этого меню обозначает, судя по всему, некий усредненный смартфон.
- Поля ввода ширины и высоты экрана позволяют ввести параметры требуемого экрана вручную.

  Сразу после выбора какого-либо устройства в приведенном ранее меню устройства в этих полях ввода появятся параметры экрана выбранного устройства. При необходимости их можно изменить.

- Меню масштаба выводимой страницы и кнопка смены ориентации устройства с вертикальной на горизонтальную и наоборот.



Главная страница сайта суши-бара в горизонтальной ориентации ❯

31.4. Адаптация элементов веб-страниц под размеры их родителей

Адаптировать оформление элементов страницы можно не только под характеристики устройства, но и под размеры их общего родителя. Что может потребоваться, если размеры родителя зависят от размеров области просмотра (поскольку указаны в процентах или относительных единицах измерения, основанных на размерах области просмотра).

Для этого используются медиазапросы, которые проверяют не характеристики устройства, на котором выводится страница, а размеры общего родителя нужных элементов. Такие медиазапросы записываются только в CSS-коде.

Предварительно следует записать в стиле элемента-родителя указание на то, что его потомки должны адаптироваться под его размеры. Для этого

применяется ненаследуемый атрибут стиля `container-type`. В качестве его значения можно указать:

- `size` — потомки элемента-родителя должны адаптироваться к его ширине и высоте;
- `inline-size` — потомки родителя должны адаптироваться только к его ширине;
- `normal` — потомки не должны адаптироваться к размерам родителя.

Значение по умолчанию: `normal`.

Сам медиазапрос, проверяющий характеристики родителя, пишется с применением директивы `@container`:

```
@container [<имя родителя>] <набор характеристик родителя> {
    <стили, применяемые к потомкам родителя при совпадении его ↵
    размеров с заданным набором характеристик>
}
```

Набор характеристик родителя записывается так же, как и набор характеристик устройства (см. *разд. 31.2.2*). Поддерживаются лишь характеристики `width`, `height`, `aspect-ratio` и `orientation`.

Если *имя родителя* не задано, веб-обозреватель будет проверять характеристики элемента, удовлетворяющего двум условиям:

- являющегося ближайшим общим родителем всех элементов, на которые ссылаются *стили*, записанные в медиазапросе;
- содержащего в применяемом к нему стиле атрибут стиля `container-type` со значением `size` или `inline-size`.

Пример:

```
<div class="parent">
    <div class="child">Адаптируемый элемент</div>
</div>
. . .
.parent {
    container-type: inline-size;
    border: thin solid black;
}
.child {
    font-size: 24pt;
    text-decoration-line: underline;
```

```
}
@container (max-width: 500px) {
    .child { text-decoration-line: none; }
}
```

Результат:

Ширина родителя более 500 пикселов	Ширина родителя не более 500 пикселов
Адаптируемый элемент	Адаптируемый элемент

Если адаптируемые элементы последовательно вложены в несколько родителей, в стилях которых заданы атрибуты стиля со значениями `size` или `inline-size`, может потребоваться указать, к размерам какого из родителей они должны адаптироваться. Для этого следует дать нужному родителю уникальное имя.

Имя, которое необходимо дать родителю, указывается в его стиле, в атрибуте стиля `container-name`. В нем можно задать:

- собственно имя родителя. Оно может содержать лишь буквы, цифры, символы дефиса и подчеркивания и быть уникальным в пределах всех страниц сайта;
- `none` — родитель не будет иметь имени.

Значение по умолчанию: `none`.

Далее нужно указать это *имя* в медиазапросе, поставив его после слова `@container`.

Пример:

```
<div class="parent2">
    <div class="parent3">
        <div class="child2">Адаптируемый элемент</div>
    </div>
</div>
. . .
.parent2, .parent3 {
    container-type: inline-size;
    border: thin solid black;
    padding: 8pt;
```

```
}
.parent2 { container-name: par; }
.parent3 { width: 50%; }
.child2 {
    font-size: 24pt;
    text-decoration-line: underline;
    text-underline-offset: 8pt;
}
@container par (max-width: 500px) {
    .child2 { text-decoration-line: none; }
}
```

Результат:

Ширина родителя более 500 пикселов	Ширина родителя не более 500 пикселов
Адаптируемый элемент	Адаптируемый элемент

В стилях, записываемых в медиазапросах, которые проверяют характеристики родителя, можно использовать любые единицы измерения, описанные в *разд. 12.2.1.2.1*. Кроме того, доступны следующие специфические единицы измерения, основанные на размерах родительского элемента (табл. 31.1).

Таблица 31.1

Обозначение	Описание
`cqw`	$^{1}/_{100}$ ширины родителя
`cqh`	$^{1}/_{100}$ высоты родителя
`cqmin`	Наименьшее из значений: `cqw` или `cqh`
`cqmax`	Наибольшее из значений: `cqw` или `cqh`

Пример:

```
@container (max-width: 500px) {
    .child { width: 50cqw; }
}
```

31.5. Упражнение. Адаптация фотогалереи «The Beatles»

Теперь адаптируем для просмотра на мобильных устройствах сайт фотогалереи великой группы «The Beatles». Сделаем так, чтобы на экранах шириной не более 600 пикселов выводились по две фотографии в строке, а на экранах шириной не более 400 пикселов — по одной фотографии в строке.

На этот раз используем медиазапросы, проверяющие характеристики родителей.

1. Найдем в папке 29\s29.5 сопровождающего книгу файлового архива (см. *приложение 4*) папку the-beatles-photos и скопируем ее куда-либо на локальный диск.

2. Откроем главную страницу index.html в текстовом редакторе и вставим в ее код тег с настройками области просмотра, аналогичный написанному на *шаге 2 упражнения 31.3*.

   Все фото, отображаемые на странице, находятся в семантическом основном содержимом (теге `<main>`). Именно к ширине этого элемента мы будем адаптировать его потомки-фотографии.

   Поэтому сразу же запишем в CSS-код соответствующее предписание.

3. Откроем таблицу стилей styles\main.css в текстовом редакторе и дополним стиль основного содержимого:

```
main {
    . . .
    container-type: inline-size;
}
```

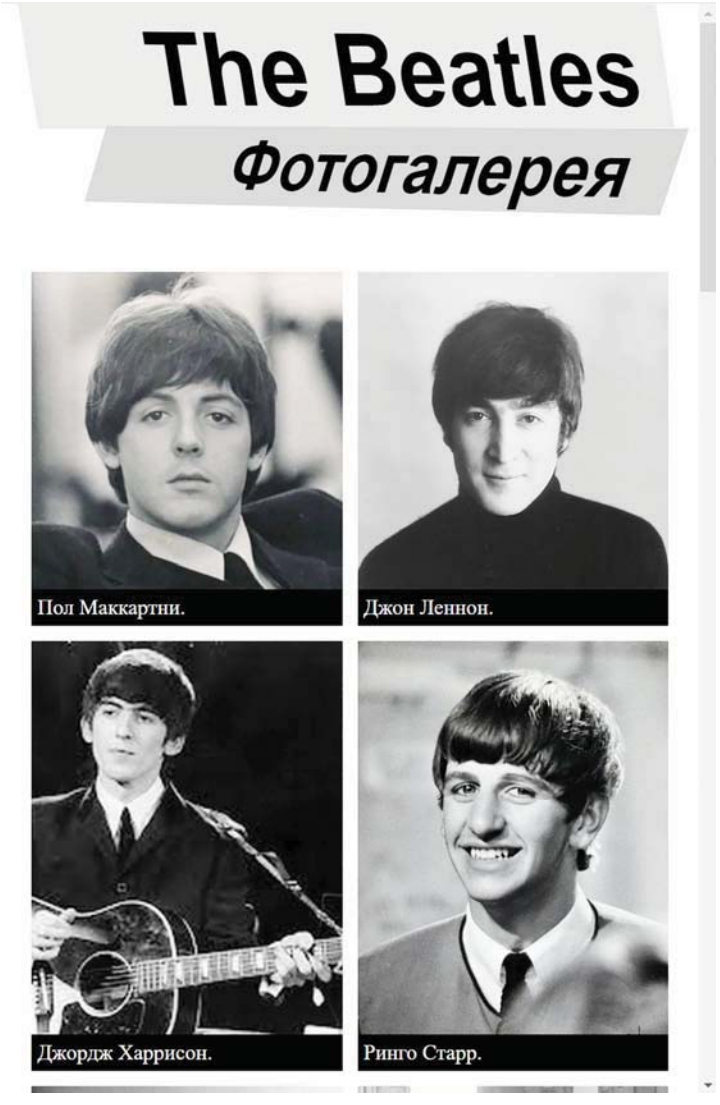
   Отдельные фото вместе с подписями к ним выводятся в элементах со стилевыми классами `photo`. Чтобы вывести по две фотографии в строке, требуется соответственно пересчитать значение ширины такого элемента. Учтем также, что теперь просвет между фото лишь один.

4. Добавим медиазапрос, который задаст ширину отдельной фотографии для экранов с шириной не более 600 пикселов:

```
@container (max-width: 600px) {
    main .photo {
        width: calc((100cqw - 10pt) / 2);
    }
}
```

   Уменьшим ширину окна веб-обозревателя, чтобы увидеть...

…что у нас получилось ⌄



Вывести одну фотографию в строке проще — достаточно указать у нее ширину равной ширине родителя.

5. Добавим еще один медиазапрос, задающий ширину фото для экранов с шириной не более 400 пикселов:

```
@container (max-width: 400px) {
    main .photo { width: 100cqw; }
}
```

Активизируем режим эмуляции мобильного устройства (см. *упражнение 31.3*), чтобы увидеть...

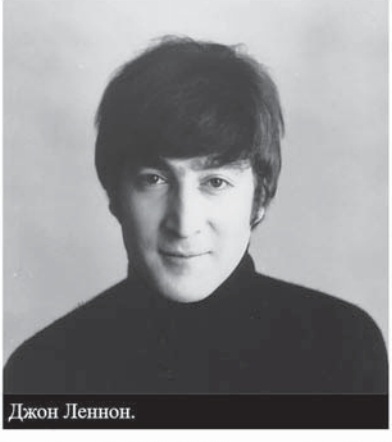
…результат ›

31.6. Адаптивные изображения

> ***Адаптивное изображение***
>
> Элемент страницы, выводящий разные изображения в зависимости от характеристик устройства.

В адаптивном изображении указывается набор источников, каждая позиция которого содержит ссылку на графический файл, хранящий одно из изображений, и медиазапрос. Веб-обозреватель проверяет устройство на совпадение медиазапросам, записанным в составе источников, и в случае совпадения загружает соответствующий файл и выводит содержащееся в нем изображение на экран.


Пол Маккартни.


Джон Леннон.

Само адаптивное изображение создается парным тегом `<picture>`. В нем записывается набор источников.

Отдельный источник представляется одинарным тегом `<source>`. Этот тег поддерживает следующие атрибуты:

- `srcset` — ссылка на графический файл, хранящий одно из изображений;
- `media` — медиазапрос, который записывается согласно правилам, приведенным в *разд. 31.2.2*.

Также тег `<source>` поддерживает атрибуты тега `width` и `height`, аналогичные имеющимся у тега `<img>` (см. *разд. 7.1.1*).

Теги источников <source> обрабатываются веб-обозревателем в том порядке, в котором они записаны в теге <picture>. Как только будет достигнут источник с совпавшим медиазапросом, обработка остальных источников прекращается.

Следует отметить, что сам тег <picture> не может вывести изображение на экран. Для этого он использует обычный тег <img>, который должен быть помещен в него. Как правило, в теге <img> указывается ссылка на изображение, которое должно быть выведено, если ни один из медиазапросов, приведенных в источниках, не совпал.

Пример адаптивного изображения, выводящего изображение из файла image-wide.jpg на экранах с шириной не менее 1024 пиксела, изображение image-standard.jpg — на экранах с шириной не менее 400 пикселов и изображение image-narrow.jpg — на экранах у́же 400 пикселов:

```
<picture>
    <source srcset="image-wide.jpg" media="(min-width: 1024px)">
    <source srcset="image-standard.jpg" media="(min-width: 400px)">
    <img src="image-narrow.jpg">
</picture>
```

31.7. Создание печатной редакции веб-страницы

31.7.1. Управление разбиением на страницы при печати и количеством висячих строк

Для управления разбиением веб-страницы на печатные страницы применяются три ненаследуемых атрибута стиля:

- `break-inside` — управляет разбиением содержимого текущего элемента на страницы:
  - `auto` — содержимое элемента может быть разбито на страницы при печати;
  - `avoid` — содержимое элемента по возможности не будет разбиваться на страницы:

    ```
    section.comment { break-inside: avoid; }
    ```

    Значение по умолчанию: `auto`;
- `break-before` — указывает, начать ли печать содержимого текущего элемента с новой страницы;

- `break-after` — указывает, начать ли печать *следующего соседа* текущего элемента с новой страницы.

  Оба этих атрибута стиля поддерживают одинаковый набор значений:

  - `auto` — содержимое элемента может начать печататься с новой страницы;
  - `page` — всегда начинать печатать содержимое элемента с новой страницы:

    ```
    h1 { break-before: page; }
    ```

  - `left` — всегда начинать печатать содержимое элемента с новой страницы, если следующая страница — левая;
  - `right` — всегда начинать печатать содержимое элемента с новой страницы, если следующая страница — правая;
  - `avoid` — по возможности никогда не начинать печать содержимого элемента с новой страницы. Пример указания не отрывать любой заголовок от следующего текста:

    ```
    h1, h2, h3, h4, h5, h6 { break-after: avoid; }
    ```

  Значения по умолчанию у обоих атрибутов стиля: `auto`.

Для задания допустимого количества висячих строк служат два наследуемых атрибута стиля:

- `widows` — минимально допустимое количество висячих строк *вверху* страницы. Задается в виде неотрицательного целого числа без единицы измерения. По умолчанию: 2;
- `orphans` — минимально допустимое количество висячих строк *внизу* страницы. Задается в виде неотрицательного целого числа без единицы измерения. По умолчанию: 2.

Пример:

```
body {
    widows: 3;
    orphans: 4;
}
```

Ранее веб-стандарты содержали атрибуты стиля `page-break-inside`, `page-break-before` и `page-break-after`, аналогичные приведенным ранее. Последние два атрибута стиля поддерживали значение `always`, аналогичное значению `page`.

Эти три атрибута стиля объявлены устаревшими и нерекомендованными к применению. Однако они до сих пор поддерживаются веб-обозревателями и встречаются в старом коде.

31.7.2. Управление содержанием печатаемой веб-страницы

Есть возможность задать какое-либо специфическое оформление для печатной редакции страницы: указать цветовую гамму «темный текст на светлом фоне», убрать графические фоны (что позволит сэкономить чернила), скрыть некоторые элементы, не нужные в печатной копии, и др.

Это можно сделать в медиазапросах, указав в самом начале простую характеристику `print` и при необходимости задав какие-либо дополнительные характеристики.

Пример указания черного цвета текста, белого фона и скрытия панели навигации (которая в печатной редакции страницы, скорее всего, не понадобится):

```
@media print {
    body {
        color: black;
        background: white;
    }
    nav { display: none; }
}
```

31.7.3. Задание параметров печатных страниц

Наконец, можно задать параметры страниц, на которых будет печататься веб-страница.

Параметры печатной страницы записываются в директиве `@page`:

```
@page [<псевдокласс страницы>|<имя страницы>] {
    <параметры страницы>
}
```

Параметры страницы записываются так же, как обычный CSS-стиль. В них можно использовать следующие атрибуты стиля:

- «семейство» `margin` — для указания просветов между краями бумажного листа и границами печатаемого содержания;
- `size` — размер и ориентация страницы. Поддерживаются следующие значения:
  - `[<размер>] [<ориентация>]` — задает *размер* и (или) *ориентацию* печатной страницы.

    Поддерживаются следующие значения *размера*: `A3`, `A4`, `A5`, `B4`, `B5`, `letter` `legal` и `ledger`.

    Размер также может быть указан в формате:
    `<ширина страницы> [<высота страницы>]`

Ширина и *высота* страницы задаются в виде числовых значений в любой из поддерживаемых единиц измерения. Если *высота* не задана, она принимается равной *ширине*.

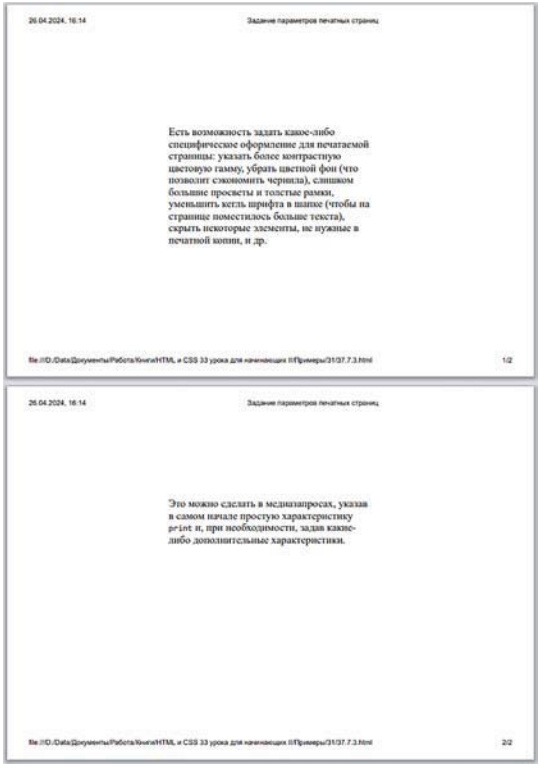
Если *размер* не указан, будет использован размер, заданный в системных параметрах принтера.

Ориентация страницы задается в виде слова `portrait` (портретная, она же вертикальная) или `landscape` (ландшафтная, или горизонтальная). Если *ориентация* не задана, они принимается равной `portrait`.

Пример:

```
@page {
    size: a5 landscape;
    margin: 30%;
}
```

Результат (взят из окна предварительного просмотра веб-страницы, предоставляемого веб-обозревателем):



- `auto` — будут использованы размер и ориентация бумаги, заданные в системных параметрах принтера.

Значение по умолчанию: `auto`.

Можно задать разные параметры для первой, левых и правых печатных страниц, записав несколько директив `@page` и использовав в них следующие *псевдоклассы*:

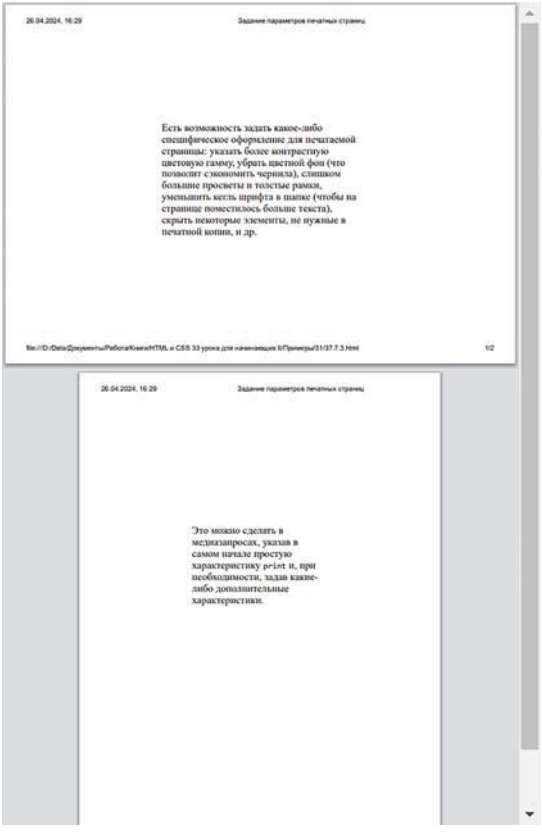
- `:first` — самая первая из печатных страниц;
- `:left` — все левые страницы;
- `:right` — все правые страницы.

Директивы `@page` с псевдоклассами имеют больший приоритет, чем директивы `@page` без псевдоклассов, и перекрывают заданные в них параметры страниц.

Пример:

```
/* И у левых, и у правых страниц указываем одинаковые просветы… */
@page { margin: 30%; }
/* …но разную ориентацию */
@page :left { size: a5 portrait; }
@page :right { size: a5 landscape; }
```

Результат:



Также можно указать отдельные параметры печатных страниц для разных элементов печатаемой веб-страницы. Для этого следует записать несколько директив `@page`, указав в каждой уникальные *имена страниц*. *Имя страницы* выбирается произвольно, может содержать лишь буквы, цифры, знаки дефиса и подчеркивания.

Далее останется записать стиль для элемента, который должен печататься на отдельной странице со специфическими параметрами, и указать в этом стиле имя этой страницы. Имя страницы задается в ненаследуемом атрибуте стиля `page`.

В нем можно задать:

- собственно, имя требуемой страницы;
- `auto` — будут взяты параметры страниц, которые записаны в директивах, не содержащих *имен страниц*.

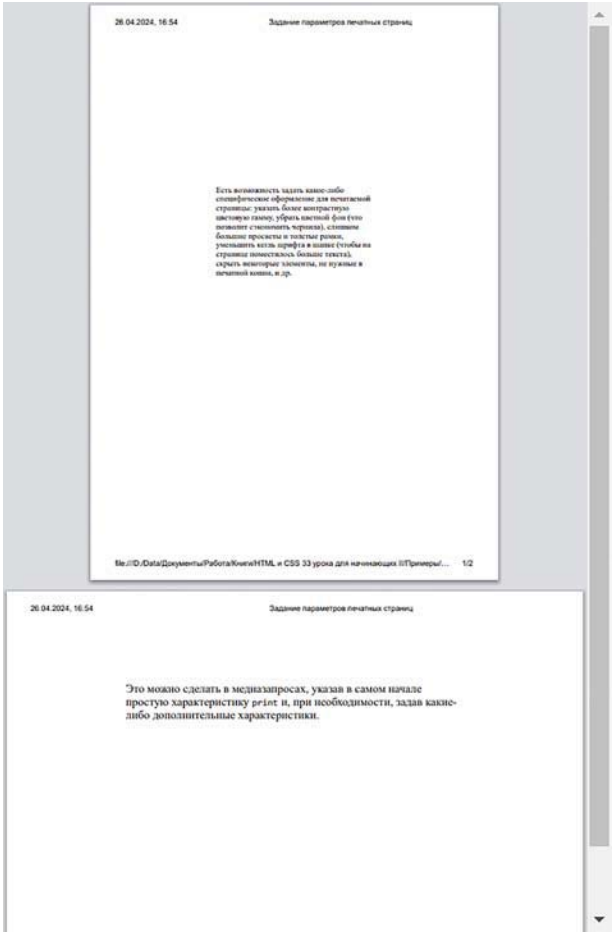
Значение по умолчанию: `auto`.

Директивы `@page` с именами страниц имеют больший приоритет, чем директивы с псевдоклассами, и значительно больший, чем директивы без псевдоклассов.

Пример:

```
/* Параметры всех печатных страниц, кроме последней */
@page {
    size: a4;
    margin: 30%;
}
/* Параметры последней печатной страницы */
@page last-page {
    size: a5 landscape;
    margin: 20%;
}
p:last-of-type { page: last-page; }
```

Результат:



31.8. Самостоятельные упражнения

- На страницах каталогов суши-бара — при выводе на экранах с шириной не более 400 пикселов — укажите высоту изображений блюд равной 150 пикселов (сейчас они великоваты, и их нужно уменьшить) и увеличьте кегль шрифта, которым выводятся названия блюд, до 38 пунктов.
- На странице фотогалереи группы «The Beatles» — при выводе на экранах с шириной не более 550 пикселов — уменьшите кегль шрифта, которым выводятся заголовки в шапке, чтобы они не вылезали за ее пределы. Точное значение кегля подберите самостоятельно.
- Не забудьте записать в стиле шапки указание на то, что ее потомки должны адаптироваться к ее ширине.
- Создайте печатную редакцию сайта суши-бара. Уберите у всех страниц фоны, скройте панель навигации и уберите большой внешний просвет слева у основного содержимого. У поддона отмените печать с новой страницы.

Урок 32. Фреймы

Фреймы.
Вывод веб-страниц во фреймах по щелчкам на гиперссылках.

> ***Фрейм***
>
> Своего рода небольшой веб-обозреватель, встроенный в текущую страницу и выводящий содержание другой страницы — с текущего или другого сайта.

32.1. Создание фреймов

Фрейм создается парным тегом `<iframe>`, не имеющим содержимого. Этот тег поддерживает следующие важные атрибуты:

- `src` — ссылка на веб-страницу, которая должна выводиться во фрейме изначально, сразу после загрузки текущей страницы.

  Атрибут тега не является обязательным. Если его не указать, будет создан пустой фрейм, в котором ничего не выводится. Можно создать гиперссылку, при щелчке на которой в таком фрейме будет выведена какая-либо страница (как это сделать, описано в *разд. 32.2*);

- `width` и `height` — соответственно, ширина и высота фрейма в виде целых положительных чисел, измеряемые в пикселах. Если не указаны, фрейм будет иметь размеры 300 × 150 пикселов;

- `loading` — указывает, когда веб-обозревателю следует загрузить страницу, выводимую во фрейме. Можно задать одно из следующих значений:

  - `eager` — загрузить страницу немедленно, даже если выводящий ее фрейм в текущий момент не присутствует на экране (например, находится за пределами области просмотра);

- `lazy` — загрузить страницу только тогда, когда фрейм присутствует на экране (когда посетитель, прокручивая страницу, выведет фрейм в области просмотра).

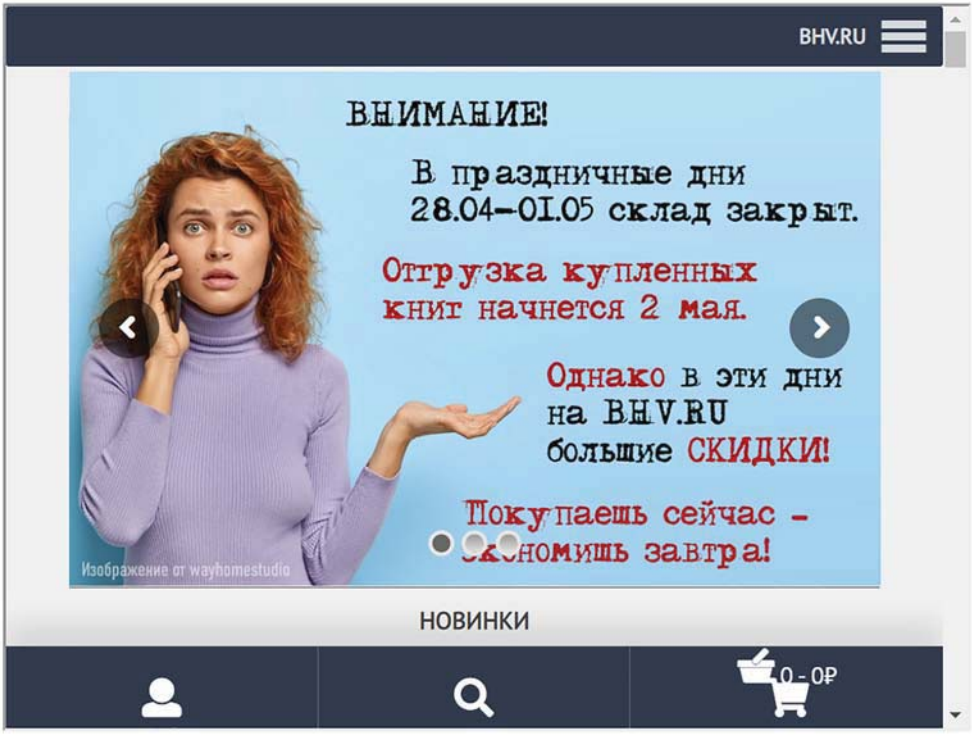
Значение по умолчанию: `eager`.

Фрейм является встроенно-блочным элементом страницы.

Пример фрейма, выводящего одну из страниц сайта издательства «БХВ»:

```
<iframe src="https://bhv.ru/" width="640" height="480"></iframe>
```

Результат:



По умолчанию фрейм выводится в трехмерной «вдавленной» рамке. Разумеется, средствами CSS можно задать другую рамку или вообще убрать ее (см. *разд. 15.1*).

При переходах по гиперссылкам сайта, открытого во фрейме, целевые страницы также будут открываться во фрейме.

Фреймы применятся в случае, если на странице требуется вывести содержание другой страницы, как правило, принадлежащей другому сайту.

Будучи фактически полноценным веб-обозревателем, помещенным на страницу, фрейм отнимает довольно много системных ресурсов. Поэтому использовать фреймы следует лишь тогда, когда без них нельзя обойтись.

32.2. Открытие веб-страниц во фрейме по щелчкам на гиперссылках

Может понадобиться сделать так, чтобы при щелчке на гиперссылке целевая страница открывалась не в окне веб-обозревателя, а в заданном фрейме. Выполняется это в два шага:

1. Фрейму, в котором должны открываться страницы, дается наименование — в атрибуте `name`, поддерживаемом всеми тегами.
2. Наименование требуемого фрейма записывается в гиперссылке, в атрибуте `target` тега `<a>` (см. *разд. 9.1*).

Пример:

```
<a href="https://bhv.ru/" target="output">
    Издательство &laquo;БХВ&raquo;
</a>
. . .
<iframe name="output"></iframe>
```

Урок 33. Полезные мелочи

Метаданные.
Привязка данных к веб-странице.
Форма курсора мыши.
Распределение системных ресурсов.

33.1. Метаданные веб-страницы

Метаданные — это данные, описывающие саму страницу[1]. Из *разд. 4.3* мы знаем, что к ним относятся, в частности, обозначение кодировки, в которой написан HTML-код страницы, и ее название. Обозначение кодировки записывается в одинарном теге `<meta>`, в его атрибуте `charset`, а название страницы — в парном теге `<title>`. Оба тега включаются в секцию заголовка страницы (в тег `<head>`).

Однако в код страницы в качестве метаданных можно дополнительно занести имя автора страницы, ее краткое описание, набор ключевых слов и др.

Эти метаданные записываются тоже с помощью одинарных тегов `<meta>` и также в секции заголовка страницы. Для записи отдельной единицы метаданных в отдельном теге `<meta>` применяются следующие атрибуты тега:

- `name` — название единицы метаданных, записываемой в теге;
- `content` — сама единица метаданных.

Веб-стандарты предусматривают ряд типовых единиц метаданных, которые можно записать в коде страницы (табл. 33.1).

[1] Или, в более общем случае, метаданные — это данные, описывающие другие данные.

Таблица 33.1

Название	Описание
`author`	Имя автора страницы
`description`	Краткое описание страницы и опубликованного на ней материала
`keywords`	Набор ключевых слов, максимально полно описывающих страницу, разделенных запятыми или комбинациями из запятой и пробела
`generator`	Название программы, в которой была создана страница
`color-scheme`	Название поддерживаемой цветовой схемы: `normal` (любая цветовая схема), `light` (светлая), `dark` (темная), `light dark` (изначально светлая, но может быть установлена темная), `dark light` (изначально темная, но может быть установлена светлая) или `only light` (то же самое, что и `light`). По умолчанию: `normal`

Веб-обозреватели используют в работе, по крайней мере, часть стандартных метаданных. Так, название цветовой схемы применяется для установки цветовой схемы, а описание страницы записывается в составе закладки в избранном.

Пример:

```
<head>
    . . .
    <title>Йокогама: суши-бар</title>
    <meta name="author" content="Читатели книги">
    <meta name="description"
          content="Суши-бар Йокогама. Сашими, суши, роллы,
напитки">
    <meta name="keywords"
          content="суши-бар,Йокогама,сашими,суши,ролл,напиток">
    <meta name="generator" content="Visual Studio Code">
    <meta name="color-scheme" content="light">
</head>
```

Кроме стандартных единиц метаданных, можно записать в странице любые произвольные единицы, дав им уникальные названия. Однако такие метаданные не будут поддерживаться веб-обозревателями и фактически окажутся бесполезными.

33.2. Привязка данных к веб-странице

Помимо внешних таблиц стилей, мы можем привязать к странице и другие данные, в частности, значок сайта. Для этого используется одинарный тег

<link>, знакомый нам по *разд. 11.1.1.1*. Привязываемые данные описываются в следующих атрибутах этого тега:

- rel — обозначение привязываемых данных (например, stylesheet обозначает, что привязывается внешняя таблица стилей);
- href — ссылка на файл с привязываемыми данными.

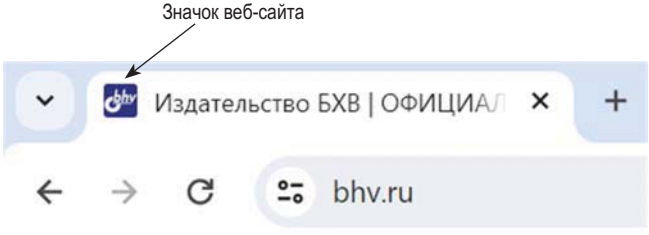
Все теги <link> записываются в секции заголовка страницы, чтобы веб-обозреватель обработал их сразу после начала загрузки страницы.

Количество привязываемых к странице данных не ограничено.

Далее будут рассмотрены практические примеры привязки к странице различных данных.

33.2.1. Значок веб-сайта

К страницам веб-сайта можно привязать значок, который будет выводиться в левой части вкладки окна веб-обозревателя (*значок веб-сайта*):



Чтобы привязать к странице значок, следует добавить в секцию заголовка страницы тег <link> со следующими атрибутами тега:

- rel — со значением icon;
- href — со ссылкой на файл со значком.

  Традиционно файлу со значком сайта дается имя favicon. Значок может быть сохранен как в формате ICO, используемом в Windows, так и в любом из интернет-форматов (как правило, в GIF или PNG, подробности — в *разд. 7.1.2*).

Пример:

```
<head>
    . . .
    <title>Йокогама: суши-бар </title>
    <link rel="icon" href="favicon.png">
</head>
```

33.2.2. Предварительная загрузка данных

Файлы, связанные со страницей (изображения, аудио-, видеоролики и др.), загружаются, когда в них возникает непосредственная необходимость. Так, файл с изображением будет загружен, когда веб-обозреватель в процессе обработки HTML-кода страницы дойдет до того места, где присутствует тег `<img>`, выводящий это изображение.

Загрузка любого файла приостанавливает вывод страницы на экран до момента, пока этот файл не будет загружен. Если со страницей связано много файлов (например, она содержит много изображений), ее вывод может занять много времени.

Отображение страницы можно ускорить, предписав веб-обозревателю начать загрузку связанных файлов в самом начале обработки кода страницы, задолго до того момента, когда в них возникнет необходимость. В результате ко времени вывода эти файлы уже будут загружены. Такой подход носит название *предварительной загрузки*.

Чтобы выполнить предварительную загрузку какого-либо файла, его следует привязать к странице с помощью тега `<link>` с атрибутами:

- `rel` — со значением `preload`;
- `href` — со ссылкой на предварительно загружаемый файл;
- `as` — обозначает разновидность данных, хранящихся в предварительно загружаемом файле. Поддерживаются следующие значения:
  - `image` — изображение, выводимое в теге `<img>`, указанное в теге `<source>` (см. *разд. 31.6*) или используемое в таблицах стилей (например, в качестве фонового);
  - `audio` — аудиоролик;
  - `video` — видеоролик;
  - `embed` — файл, выводимый в теге `<embed>` (см. *разд. 7.7.1*);
  - `object` — файл, выводимый в теге `<object>` (см. *разд. 7.7.2*);
  - `document` — страница, выводящаяся во фрейме (см. *урок 32*).

Пример:

```
<head>
    . . .
    <title>Суши :: Йокогама: суши-бар </title>
    <link rel="preload" href="../images/sushi-chukka.jpg" as="image">
    <link rel="preload" href="../images/sushi-maguro.jpg" as="image">
```

```
    <link rel="preload" href="../images/sushi-tobiko.jpg" as="image">
</head>
```

Также поддерживается атрибут тега `media`, в котором можно указать медиазапрос. Тогда файл будет загружен лишь в случае, если устройство совпадет с заданным медиазапросом. Пример:

```
<head>
    . . .
    <link rel="preload" href="image-wide.jpg" as="image"
                        media="(min-width: 1024px)">
    <link rel="preload" href="image-standard.jpg" as="image"
                        media="(min-width: 400px)">
    <link rel="preload" href="image-narrow.jpg" as="image"
                        media="(max-width: 399px)">
</head>
. . .
<body>
    . . .
    <picture>
        <source srcset="image-wide.jpg" media="(min-width: 1024px)">
        <source srcset="image-standard.jpg" media="(min-width: 400px)">
        <img src="image-narrow.jpg">
    </picture>
    . . .
</body>
```

33.3. Задание формы курсора мыши

CSS позволяет указать форму, которую примет курсор мыши при наведении на элемент страницы. Для этого применяется ненаследуемый атрибут стиля `cursor`. Он поддерживает довольно много значений, а вот наиболее часто используемые:

- `auto` — формой курсора мыши управляет веб-обозреватель;
- `default` — обычная стрелка;
- `pointer` — «указующий перст»:

  ```
  .element1 { curson: pointer; }
  ```

- `text` — текстовый курсор;
- `progress` — стрелка с вращающимся индикатором процесса;
- `wait` — вращающийся индикатор процесса;

- `help` — стрелка с небольшим вопросительным знаком;
- `crosshair` — перекрестье;
- `not-allowed` — перечеркнутый красный круг;
- `none` — курсор вообще не выводится.

Полный список значений, поддерживаемых атрибутом стиля `cursor`, вместе с изображениями соответствующих курсоров, можно найти на странице: **https://developer.mozilla.org/en-US/docs/Web/CSS/cursor**.

Также можно использовать курсор, хранящийся в заданном файле формата CUR. Для этого в атрибуте стиля `cursor` следует записать функцию `url()` в формате:

`url(<`*`ссылка на файл с курсором`*`>) [<`*`гориз. коорд.`*`> <`*`верт. коорд.`*`>]`

После функции `url()` можно указать *горизонтальную* и *вертикальную координаты* точки курсора, которая будет служить собственно указателем. Они исчисляются в пикселах, отсчитываются от левого верхнего угла курсора вправо и вниз. Если *координаты* не заданы, точка-указатель будет иметь координаты [0, 0].

Пример:

```
.element2 { cursor: url(../cursors/my-cursor.cur) 4 6; };
```

После функции `url()` и *координат* точки-указателя рекомендуется поставить запятую и записать наименование одного из стандартных курсоров — на тот случай, если указанный файл с курсором по какой-то причине не будет загружен:

```
.element3 { cursor: url(../cursors/my-cursor.cur) 4 6, pointer; };
```

Значение атрибута стиля `cursor` по умолчанию: `auto`.

33.4. Управление распределением системных ресурсов

Обычно веб-обозреватель при выводе страницы сам распределяет системные ресурсы компьютера, решая, сколько ресурсов выделить на вывод содержания страницы, изменившегося в результате действий пользователя, сколько — на ее прокрутку, а сколько — на воспроизведение анимации. Но иногда его «самоуправство» не приводит ни к чему хорошему — например, важная анимация воспроизводится рывками, поскольку все ресурсы брошены на вывод содержимого страницы.

Указать, на что в первую очередь следует направить системные ресурсы, можно посредством ненаследуемого атрибута стиля `will-change`. Он поддерживает следующие значения:

- `scroll-position` — ресурсы следует направить на выполнение прокрутки страницы;
- `contents` — на вывод содержания страницы, изменившегося в результате действий пользователя (например, гиперссылок, меняющих цвет при наведении на них курсора мыши);
- *<атрибут стиля>* — на анимирование заданного *атрибута стиля* (анимации посвящен *урок 30*):

```
.animated1 {
    position: absolute;
    left: 10px;
    top: 20px;
    will-change: left;
}
.animated1:hover {
    left: 200px;
    transition-duration: 3s;
}
```

- *<атрибут стиля 1>*, *<атрибут стиля 2>*, . . ., *<атрибут стиля N>* — на анимирование заданных *атрибутов стиля*:

```
.animated6 {
    position: absolute;
    left: 10px;
    top: 20px;
    background-color: grey;
    will-change: left, top;
}
.animated6:hover {
    left: 200px;
    top: 40px;
    background-color: yellow;
    transition-property: left, background-color, top;
    transition-duration: 3s, 4s, 2s;
}
```

- `auto` — веб-обозреватель будет распределять ресурсы по своему усмотрению.

Значение по умолчанию: `auto`.

33.5. Управление выделением

По умолчанию веб-обозреватель позволяет выделять содержимое элементов страницы с целью скопировать его в буфер обмена. Однако это поведение можно изменить.

Для управления выделением содержимого элемента применяется ненаследуемый атрибут стиля `user-select`. Вот поддерживаемые им значения:

- `text` — текстовое содержимое элемента доступно для выделения;
- `all` — элемент может быть выделен только целиком, вместе со всем его содержимым (включая потомки);
- `none` — содержимое элемента недоступно для выделения:

```
.only-see-not-copy { user-select: none; }
```

- `auto`:
  - генерируемое содержание и содержимое элементов, у родителей которых явно отключена возможность выделения, — недоступно для выбора (как если бы было задано значение `none`);
  - содержимое элементов, у родителей которых явно включен режим выделения только целиком, — доступно для выделения только целиком (аналог значения `all`);
  - содержимое прочих элементов — доступен для выделения текст (`text`).

Значение по умолчанию: `auto`.

33.6. Создание элементов с изменяемыми размерами

Ненаследуемый атрибут стиля `resize` указывает, может ли пользователь менять размеры элемента страницы. Доступные для указания значения:

- `both` — можно менять ширину и высоту элемента;
- `horizontal` — можно менять только ширину;
- `vertical` — только высоту;
- `none` — размеры менять нельзя.

Значение по умолчанию: `both` (у областей редактирования) или `none` (у остальных элементов).

Чтобы настройка, заданная в атрибуте стиля `resize`, вступила в силу, следует также задать один из атрибутов стиля «семейства» `overflow` со значением, отличным от `visible` (см. *разд. 15.8*).

Пример:

```
.resizeable {
    overflow: auto;
    resize: both;
}
```

Результат:

33.7. Редактируемые элементы веб-страниц

Любой элемент страницы можно сделать редактируемым, после чего пользователь сможет дополнять и править его содержимое. Для этого следует вставить в тег, создающий этот элемент, атрибут `contenteditable`, поддерживаемый всеми тегами. Пример:

```
<div contenteditable>Редактируемый элемент веб-страницы</div>
```

В редактируемом элементе работают следующие комбинации клавиш:

- <Ctrl>+<B> — делает выделенный текст полужирным;
- <Ctrl>+<I> — курсивным;
- <Ctrl>+<U> — подчеркнутым.

Вообще, атрибут тега `contenteditable` может быть использован как:

- атрибут без значения — превращает элемент в редактируемый;
- обычный атрибут с указанием одного из следующих значений:
  - `""` (пустая строка) или `true` — превращает элемент в редактируемый;
  - `false` — делает элемент обычным, нередактируемым.

Впрочем, эта возможность скорее курьезна, нежели практически полезна.

Заключение

На этом учебный курс по языкам HTML, CSS и основам веб-верстки завершен.

Вы изучили все необходимое для того чтобы начать работать самостоятельно: продолжить эксперименты с веб-технологиями, изготовить свой собственный сайт или даже сделать карьеру профессионального веб-верстальщика. Вы получили практический опыт веб-верстки и теперь знаете, как использовать впечатляющие инструменты HTML и CSS для достижения нужного результата. Вы даже создали под руководством автора несколько сайтов, вполне пригодных для публикации в Сети.

Автор постарался описать в книге все более или менее востребованные инструменты HTML и CSS, сделав ее тем самым как можно более полным руководством по этим языкам, которое вы можете использовать также в качестве справочника. За рамками книги остались лишь:

- крайне специфические языковые средства, применяемые на практике очень редко;
- инструменты, не поддерживаемые основными игроками рынка веб-обозревателей (к ним относятся Google Chrome, Mozilla Firefox, Microsoft Edge Chromium, Opera и Apple Safari).

  Хотя, вполне возможно, поддержка этих инструментов появится во всех основных веб-обозревателях уже в ближайшем будущем.

Если же вы, уважаемые читатели, хотите узнать об HTML и CSS, что называется, всю подноготную, то вот вам перечень интернет-ресурсов, где можно найти дополнительную информацию:

- **https://developer.mozilla.org/en-US/docs/Web/HTML/** — исчерпывающий ресурс об HTML, созданный командой разработчиков Mozilla Firefox. Содержит введение в язык, руководства и справочники;
- **https://developer.mozilla.org/en-US/docs/Web/CSS/** — аналогичный ресурс о CSS;

- **https://www.w3schools.com/** — один из лучших зарубежных ресурсов, посвященных веб-технологиям, включая HTML и CSS. Руководства, справочники, много примеров;
- **https://htmlbook.ru/** — хороший русский ресурс, аналог предыдущего;
- **https://html5book.ru/** — еще один русский ресурс, достойный посещения. Помимо всего прочего, содержит полезные советы по дизайну веб-страниц.

На этом автор прощается с вами, уважаемые читатели. Успехов вам!

Владимир Дронов

ПРИЛОЖЕНИЯ

Приложение 1. Одностраничные веб-сайты

Одностраничные веб-сайты.
Навигация по одностраничным веб-сайтам.

> ***Одностраничный веб-сайт***
>
> Веб-сайт, все содержание которого находится на одной странице.

Одностраничные сайты обычно невелики по объему и служат рекламным целям.

П1.1. Упражнение. Одностраничный веб-сайт

Создадим еще одну редакцию сайта о суши-баре «Йокогама», на этот раз одностраничную.

1. Найдем в папке A1\!sources сопровождающего книгу файлового архива (см. *приложение 4*) папку one-page и скопируем ее куда-либо на локальный диск.

   Дальнейшая работа будет протекать в только что созданной копии этой папки.

   Поместим на страницу (в текущий момент пустую) также пустые семантические шапку, статью, посвященную сашими (пока одну — остальные добавим позже), и поддон. А потом наполним их содержимым.

2. Откроем страницу index.html в текстовом редакторе и вставим в секцию тега (в тег <body>) необходимый HTML-код:

```
<header>
    <section>
    </section>
</header>
<article>
    <section>
    </section>
</article>
<footer>
    <section>
    </section>
</footer>
```

В каждый из тегов: <header>, <article> и <footer>, создающих, соответственно, шапку, статью и поддон, мы вставили тег семантической части <section> (см. *разд. 5.2*). В него мы потом поместим содержимое шапки, статьи и поддона, после чего, применив знакомый нам по *разд. 22.3.2* трюк, выровняем его по центру.

3. Откроем пустую таблицу стилей styles\main.css (она уже привязана к странице) в текстовом редакторе и запишем стили, оформляющие созданные ранее элементы:

```
body {
    margin: 0;
    font-family: serif;
}
header, article, footer {
    box-sizing: border-box;
    height: 100vh;
    border: 10pt solid blanchedalmond;
    display: flex;
    justify-content: center;
    align-items: center;
}
```

Шапку, статью и поддон мы растянули по вертикали на всю высоту области просмотра, записав в атрибуте стиля height значение 100vh (единица измерения vh обозначает $^{1}/_{100}$ от высоты окна веб-обозревателя). Еще мы указали, чтобы атрибут стиля height задавал полную высоту элемента, а не высоту его содержимого, добавив атрибут стиля box-sizing

со значением `border-box`, — так нам будет проще в дальнейшем. И, наконец, мы указали у всех этих элементов флекс-разметку и задали выравнивание их единственного потомка по центру по горизонтали и вертикали.

4. Переключимся на страницу index.html и вставим в тег `<section>`, находящийся в шапке (`<header>`), код, создающий содержимое шапки:

```
<header>
    <section>
        <h1>Суши-бар</h1>
        <h1>Йокогама</h1>
        <aside><img src="images/sushi.jpg"></aside>
        <address>Ул. Зеленая, 17.</address>
        <address>Часы работы: 12:00-22:00.</address>
    </section>
</header>
```

5. Переключимся на таблицу стилей styles\main.css и добавим стили, которые зададут оформление шапки:

```
h1, h2, h3 {
    font-family: sans-serif;
    margin: 0;
}
header h1:first-of-type {
    font-size: 48pt;
    text-align: right;
}
header h1:last-of-type {
    font-size: 64pt;
    font-variant: small-caps;
    letter-spacing: 8pt;
    margin-bottom: 12pt;
    text-align: center;
}
header aside {
    width: 40%;
    float: left;
}
header address {
    font-size: 28pt;
    text-align: right;
    margin: 12pt 8pt;
}
```

Чтобы указать разное оформление у первого и второго заголовков первого уровня («Суши-бар» и «Йокогама» соответственно), мы применили псевдоклассы :first-of-type и :last-of-type. Врезку (<aside>) с изображением суши мы превратили в плавающий элемент и сдвинули к левому краю родителя, в результате чего адрес суши-бара и часы его работы будут выводиться справа от нее.

✓ Что у нас получилось? ⌄

Показана шапка.

Получилось лаконично и довольно стильно.

6. Переключимся на страницу и вставим в тег <section>, находящийся в статье (<article>), содержимое первой статьи, повествующей о сашими:

```
<article>
    <section>
        <h2>Сашими</h2>
        <table>
            <tr>
```

```
                <td><img src="images/sashimi-unagi.jpg"></td>
                <td>
                    <h3>Унаги</h3>
                    <p>Копченый угорь, свежий салат, кунжут и
                       одноименный соус.</p>
                    <p class="price">340 &#8381;</p>
                </td>
            </tr>
            <tr>
                <td><img src="images/sashimi-hamachi.jpg"></td>
                <td>
                    <h3>Хамачи</h3>
                    <p>Готовится из желтохвоста, лакедры,
                       зеленого лука, риса и дайкона.</p>
                    <p class="price">400 &#8381;</p>
                </td>
            </tr>
        </table>
    </section>
</article>
```

7. Переключимся на таблицу стилей и добавим стили, которые оформят все это:

```
article h2 {
    font-size: 26pt;
    margin-bottom: 4pt;
}
article table {
    width: 80vw;
    margin: 0 auto;
}
article table td { padding: 0 6pt; }
article img { height: 20vh; }
article h3 {
    font-size: 20pt;
    margin-bottom: 4pt;
}
article p { font-size: 16pt; }
article p.price {
    font-weight: bold;
    text-align: right;
}
```

У изображений блюд мы задали высоту равную $^1/_5$ от высоты области просмотра — чтобы каталог не слишком растягивался в высоту (не забываем, что нам еще писать статьи о суши и роллах, более объемные, чем статья о сашими...).

✓ Что у нас получилось? ⌄

Показана статья, посвященная сашими.

8. Переключимся на страницу и впишем в тег `<section>`, что находится в поддоне (`<footer>`), призыв к будущим посетителям:

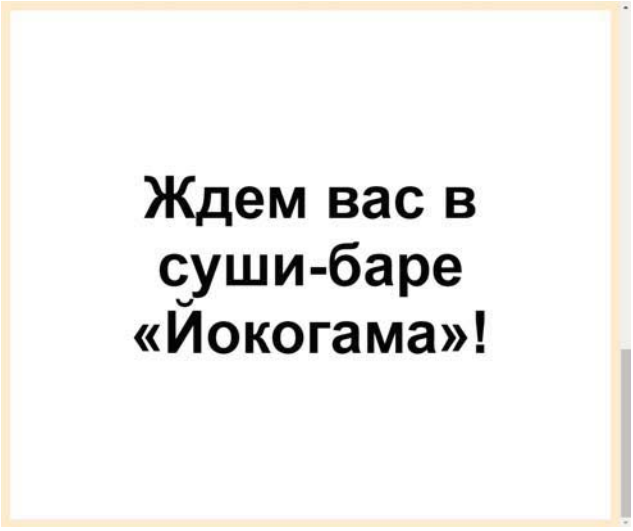
```
<footer>
    <section>
        <h1>Ждем вас в суши-баре &laquo;Йокогама&raquo;!</h1>
    </section>
</footer>
```

9. В таблицу стилей не забудем добавить стиль для поддона:

```
footer h1 {
    font-size: 64pt;
    text-align: center;
    margin-left: 10%;
    margin-right: 10%;
}
```

✓ Что у нас получилось? ⌄

Показан поддон.



Вот такой у нас получился сайтик. Все его содержание находится на одной странице, и посетитель сможет просматривать его, просто прокручивая страницу в окне веб-обозревателя.

Но, по-хорошему, нужно предусмотреть более удобные и привычные средства навигации по сайту — гиперссылки.

П1.2. Упражнение. Панель навигации для одностраничного веб-сайта

Панель навигации на одностраничном сайте создается так же, как у обычного, «многостраничного» сайта. Единственное исключение: помещаемые в ней гиперссылки ссылаются не на страницы, а на якоря. Для этого каждому из разделов сайта (шапке, отдельной статье, поддону) нужно дать якорь.

Сделаем панель навигации для нового сайта суши-бара. Поместим ее в верхней части каждого раздела сайта.

1. Найдем в папке A1\ex1.1 сопровождающего книгу файлового архива (см. *приложение 4*) папку one-page и скопируем ее куда-либо на локальный диск.

   Дальнейшая работа будет протекать в только что созданной копии этой папки.

2. Откроем страницу index.html и найдем в ее коде теги: `<header>`, `<article>` и `<footer>`. Укажем у них якоря: `hd`, `sashimi` и `ft` соответственно:

```
<header>
<header id="hd">
    . . .
</header>
<article>
<article id="sashimi">
    . . .
</article>
<footer>
<footer id="ft">
    . . .
</footer>
```

3. Вставим в начало шапки (тег `<header>`) код, создающий панель навигации:

```
<header id="hd">
    <nav>
        <div>Главная</div>
        <a href="#sashimi">Сашими</a>
        <a href="#">Суши</a>
        <a href="#">Роллы</a>
        <a href="#ft">Поддон</a>
    </nav>
        . . .
</header>
```

Статьи о суши и роллах у нас еще не готовы, поэтому мы временно сделали гиперссылки **Суши** и **Роллы** пустыми.

Панель навигации будет располагаться вдоль верхней стороны шапки. Расположить панель таким образом проще всего, превратив ее в свободно позиционируемый элемент. А чтобы разместить в панели гиперссылки, мы укажем у нее флекс-разметку.

Для успешного позиционирования панелей навигации в шапке, статье и поддоне последние следует предварительно превратить в относительно позиционируемые элементы.

4. Переключимся на таблицу стилей styles\main.css, найдем стиль, оформляющий шапку, статьи и поддон, и добавим в него код:

```
header, article, footer {
    . . .
    position: relative;
}
```

5. Добавим стили, оформляющие панель навигации и ее гиперссылки:

```
nav {
    position: absolute;
    left: 0;
    top: 0;
    right: 0;
    display: flex;
    justify-content: center;
    background-color: blanchedalmond;
}
nav div, nav a {
    font: bold 24pt sans-serif;
    color: maroon;
    display: block;
    padding: 10pt;
    text-decoration: none;
}
nav a:active, nav a:hover {
    color: blanchedalmond;
    background-color: maroon;
}
```

✓ Результат

Вот теперь созданный нами одностраничный сайт выглядит почти так же, как традиционный многостраничный. Разницу заметит разве что знаток, увидев, что при щелчках на гиперссылках страница прокручивается, открывая запрошенный материал.

П1.3. Самостоятельные упражнения

- Подготовьте остальные две статьи — с каталогами суши и роллов. Оформите их так же, как и статью о сашими.
- Поместите созданную при выполнении *упражнения П1.2* панель навигации во все остальные статьи. Не забывайте в каждой копии панели навигации оформлять пункт, указывающий на текущий материал, в виде блока (тега `<div>`).

✓ У вас должно получиться ⌄

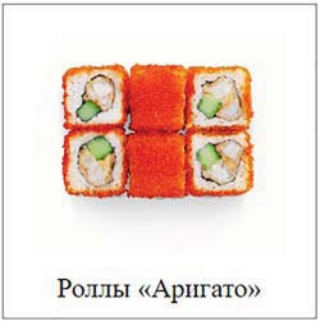
Приложение 2. В копилку веб-верстальщика

Фотография в стиле Polaroid.
Фотогалерея в стиле Polaroid.
Круглая виньетка.
Спойлер с анимацией.
Меню-гамбургер.
Аккордеон.

П2.1. Фотография в стиле Polaroid

Внешний вид

Приведены три элемента.



В таких «фотографиях» рекомендуется использовать изображения с соотношением сторон 4:3.

HTML

Приведен код, создающий один такой элемент.

```
<div class="polar">
    <img src="roll-mexican.jpg">
    <div>Роллы &laquo;Аригато&raquo;</div>
</div>
```

CSS

```
.polar {
    width: 182px;
    height: 182px;
    border: 1px solid grey;
    position: relative;
}
.polar img {
    display: block;
    width: 100%;
    height: 146px;
    object-fit: contain;
}
.polar div {
    position: absolute;
    left: 0;
    bottom: 0;
    right: 0;
    height: 18px;
    padding: 8px;
    text-align: center;
}
```

Размеры тега <img>, выводящего изображение, и блока (тега), содержащего подпись, подобраны таким образом, чтобы эти элементы выводились вплотную, не налезая друг на друга и не вылезая за пределы родительского блока.

П2.2. Фотогалерея в стиле Polaroid

Внешний вид

HTML

```
<div class="gallery">
    <div class="polar">
        <img src="roll-mexican.jpg">
        <div>Роллы &laquo;Аригато&raquo;</div>
    </div>
    <div class="polar">
        <img src="sushi-tobiko.jpg">
        <div>Суши &laquo;Тобико&raquo;</div>
    </div>
    <div class="polar">
        <img src="roll-kunsei-batakon.jpg">
        <div>Ролл &laquo;Кунсэй Батакон&raquo;</div>
    </div>
    . . .
</div>
```

CSS

```
.polar {
    width: 182px;
    height: 182px;
```

```
    border: 1px solid grey;
    position: relative;
}
.polar img {
    display: block;
    width: 100%;
    height: 146px;
    object-fit: contain;
}
.polar div {
    position: absolute;
    left: 0;
    bottom: 0;
    right: 0;
    height: 18px;
    padding: 8px;
    text-align: center;
}
.gallery {
    display: flex;
    flex-wrap: wrap;
}
.polar { margin: 20px; }
.polar:nth-child(n+1) { transform: rotate(-5deg); }
.polar:nth-child(2n+1) { transform: rotate(10deg); }
.polar:nth-child(3n+1) { transform: rotate(-15deg); }
```

П2.3. Круглая виньетка

Внешний вид

HTML

```
<div class="rounded" style="background-image: url(john-lennon.jpg);">
</div>
```

Изображение, на основе которого создается круглая виньетка, указывается во встроенном стиле (в атрибуте тега `style`) в качестве графического фона.

CSS

```
.rounded {
    width: 200px;
    height: 200px;
    background-size: cover;
    background-position: center;
    /*
        Чтобы превратить элемент страницы в круг, достаточно
        указать у него радиус скругления углов в половину (50%)
        его размеров
    */
    border-radius: 50%;
}
```

П2.4. Спойлер с анимацией

Спойлер, создаваемый тегом `<details>` (см. *разд. 10.5*), разворачивается и сворачивается рывком. Анимировать его разворачивание и сворачивание средствами CSS невозможно.

Предлагаемый спойлер разворачивается и сворачивается плавно.

Внешний вид

Свернут	Развернут
[+] «Магуро»	[-] «Магуро» Классические суши. Готовятся из тунца и риса.

HTML

```
<div class="spoiler">
    <input type="checkbox" id="sptoggler">
    <label for="sptoggler"><span></span>&laquo;Магуро&raquo;</label>
```

```
    <div>
        <p>Классические суши.</p>
        <p>Готовятся из тунца и риса.</p>
    </div>
</div>
```

Спойлер формируется следующими элементами (их вложенность друг в друга показана отступами):

- блоком (тегом `<div>`) со стилевым классом `spoiler` — он создает сам спойлер;
  - флажком с якорем `sptoggler` — он хранит текущее состояние спойлера (свернут или развернут) и не выводится на экран;
  - надписью (тегом `<label>`), связанной с флажком `sptoggler`, — она выступает в качестве заголовка спойлера;
    - пустым встроенным контейнером (`<span>`) — он служит для вывода индикатора состояния спойлера: `[+]` — если свернут, `[-]` — если развернут.

      После этого встроенного контейнера в теге `<label>` записывается текст заголовка спойлера;
  - блоком (`<div>`) — в нем помещается содержимое спойлера.

По щелчку на заголовке спойлера (теге `<label>`) связанный с ним флажок устанавливается, в результате чего к блоку с содержимым спойлера применяется стиль, который выводит его на экран. При повторном щелчке флажок сбрасывается, к блоку с содержимым применяется другой стиль, и содержимое скрывается.

CSS

Чтобы сделать спойлер свернутым, в стиле, действующем на его содержимое, необходимо указать высоту, равную `0`. Кроме того, у блока с содержимым спойлера следует указать скрытие той части содержимого, которая выходит за пределы этого блока (средствами, описанными в *разд. 15.8*).

Наиболее очевидный способ развернуть спойлер: указать в стиле, разворачивающем его содержимое, значение высоты, равное высоте содержимого спойлера, выраженное числовой величиной в любой единице измерения, кроме процентов. Это, кстати, позволило бы без проблем анимировать спойлер посредством анимации с двумя состояниями (см. *разд. 30.1*).

Проблема в том, что высоту содержимого спойлера не всегда удается определить точно. Кроме того, у разных спойлеров содержимое может иметь разную высоту.

Поэтому вместо значения точной высоты содержимого удобнее указывать в стиле его максимальную высоту, подобрав ее таким образом, чтобы содержимое даже самого «объемистого» спойлера полностью выводилось на экран. Такой подход позволит очень просто анимировать спойлер.

```
.spoiler { border: 1px solid grey; }
/* Не забываем скрыть флажок */
.spoiler > input { display: none; }
.spoiler > label {
    display: block;
    color: white;
    background-color: grey;
    margin: 1px;
    padding: 3px;
    cursor: pointer;
}
.spoiler > div {
    overflow: hidden;
    padding: 3px;
}
.spoiler > label > span { font-family: monospace; }
/*
    Если спойлер свернут (флажок сброшен), задаем у его
    содержимого максимальную высоту, равную 0, и сразу же создаем
    обратную анимацию максимальной высоты продолжительностью 2 с.
    Для ссылки на сброшенный флажок мы используем псевдокласс :not
    (см. разд. 13.4.1).
*/
.spoiler > :not(input:checked) ~ div {
    max-height: 0;
    transition: max-height 2s;
}
/*
    Также выводим во встроенном контейнере <span> индикатор [+].
    Для этого удобно использовать генерируемое содержание
    (см. разд. 14.6).
*/
.spoiler > :not(input:checked) ~ label > span:before {
    content: "[+] ";
}
/*
```

```
    Если спойлер развернут (флажок установлен), задаем у его
    содержимого максимальную высоту, равную 200 пикселов
    (подобрана опытным путем), и создаем анимацию максимальной
    высоты продолжительностью 2 с.
*/
.spoiler > input:checked ~ div {
    max-height: 200px;
    transition: max-height 2s;
}
/* Не забываем вывести индикатор [-] */
.spoiler > input:checked ~ label > span:before {
    content: "[-] ";
}
```

Если на веб-странице нужно разместить несколько спойлеров...

...в помещенных в них флажках и надписях следует задать разные якоря (выделены полужирным шрифтом):

```
<div class="spoiler">
    <input type="checkbox" id="sptoggler1">
    <label for="sptoggler1"><span></span> . . . </label>
    <div>
        . . .
    </div>
</div>
<div class="spoiler">
    <input type="checkbox" id="sptoggler2">
    <label for="sptoggler2"><span></span> . . . </label>
    <div>
        . . .
    </div>
</div>
<div class="spoiler">
    <input type="checkbox" id="sptoggler3">
    <label for="sptoggler3"><span></span> . . . </label>
    <div>
        . . .
    </div>
</div>
```

П2.5. Меню-гамбургер

> ***Меню-гамбургер***
>
> Панель навигации, выводящаяся на экран по нажатию особой кнопки и этим напоминающая стандартное меню Windows-приложений. Закрывается нажатием той же кнопки.
>
> Свое название получила из-за картинки в виде трех горизонтальных линий («гамбургера»), традиционно отображаемой на кнопке вызова меню.

Для вывода знака «гамбургера» на кнопке можно использовать специальный символ `&#9776;`.

Внешний вид

Меню скрыто, видна только кнопка вывода	Меню выведено на экран
☰ Суши-бар «Йокогама»	☰ «Йокогама» Главная Сашими Суши Роллы Прайс-лист Заказ

HTML

```
<div class="hamburger">
    <input type="checkbox" id="hmtoggler">
    <label for="hmtoggler">&#9776;</label>
    <nav>
        <a href="#">Главная</a>
        <a href="#">Сашими</a>
        <a href="#">Суши</a>
        <a href="#">Роллы</a>
        <a href="#">Прайс-лист</a>
        <a href="#">Заказ</a>
    </nav>
</div>
```

Меню-гамбургер формируется такими элементами (их вложенность друг в друга обозначена отступами):

- блоком (тегом `<div>`) со стилевым классом `hamburger` — он создает сам элемент страницы, включая меню и выводящую его кнопку;
  - флажком с якорем `hmtoggler` — он хранит текущее состояние меню (скрыто или открыто) и не выводится на экран;
  - надписью (тегом `<label>`), связанной с флажком `hmtoggler`, — она содержит знак «гамбургера» и формирует кнопку, выводящую меню;
  - панелью навигации (`<nav>`) с набором гиперссылок (`<a>`) — она создает само меню.

Здесь применяется тот же самый принцип, что и в спойлере из *разд. П2.4*. По щелчку на надписи (теге `<label>`) связанный с ней флажок устанавливается, в результате чего к панели навигации применяется стиль, выводящий ее на экран. При повторном щелчке флажок сбрасывается, стиль перестает применяться, и панель навигации скрывается.

CSS

Кнопку-«гамбургер» и само меню (тег `<nav>`) следует превратить в свободно позиционируемые элементы — так их будет проще разместить в левом верхнем углу страницы. Кнопку-«гамбургер» следует вывести поверх тега `<nav>`, чтобы эта кнопка оставалась доступной для щелчка и при выведенном меню.

Стиль, скрывающий меню (тег `<nav>`), указывает у него нулевые ширину, высоту, уровень непрозрачности и внутренние просветы. Кроме того, у меню необходимо указать скрытие части содержимого, выходящей за его пределы.

Для вывода меню к нему достаточно применить стиль, который задаст ширину и высоту, пригодные для вывода всех имеющихся в меню гиперссылок, уровень непрозрачности, равный `1.0`, и необходимые величины внутренних просветов. Точные значения размеров и внутренних просветов придется подобрать экспериментально.

Поскольку все эти значения задаются в виде числовых величин в единицах измерения, не являющихся процентами, появляется возможность анимировать вывод и скрытие меню.

При скрытом меню у выводящей его кнопки-«гамбургера» нужно указать полную рамку — это сделает кнопку заметнее. При выведенном же меню у кнопки следует задать рамку только из левой и верхней сторон (чтобы меню приняло вид, представленный на рисунке ранее).

```
div.hamburger > input { display: none; }
div.hamburger > label {
    display: block;
    position: absolute;
    left: 0;
    top: 0;
    width: 34pt;
    height: 34pt;
    z-index: 1;
    padding: 0px;
    font-size: 24pt;
    text-align: center;
    background-color: white;
    cursor: pointer;
}
div.hamburger > nav {
    position: absolute;
    left: 0;
    top: 0;
    overflow: hidden;
    background-color: white;
    border: 1px solid grey;
}
div.hamburger > nav > a {
    display: block;
    font-size: 24pt;
    text-decoration: none;
    margin: 10px;
}
div.hamburger > :not(input:checked) ~ nav {
    width: 0;
    height: 0;
    padding: 0;
    opacity: 0.0;
    transition-property: width, height, padding, opacity;
    transition-duration: .5s;
}
div.hamburger > :not(input:checked) ~ label {
    border: 1px solid grey;
}
div.hamburger > input:checked ~ nav {
```

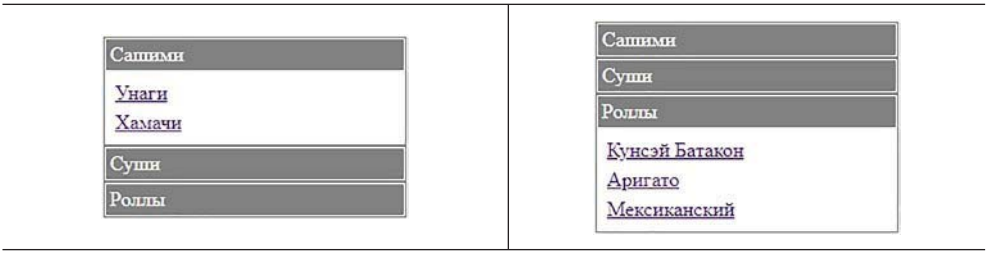
```
    width: 200px;
    height: 300px;
    padding: 52pt 5px 5px 20px;
    opacity: 1.0;
    transition-property: width, height, padding, opacity;
    transition-duration: .5s;
}
div.hamburger > input:checked ~ label {
    border-left: 1px solid grey;
    border-top: 1px solid grey;
}
```

CSS-код меню-гамбургера достаточно велик, но не так уж и сложен. Вы, уважаемые читатели, сами сможете в нем разобраться.

П2.6. Аккордеон

> ***Аккордеон***
>
> Набор спойлеров, из которых в развернутом состоянии может находиться только один (при развертывании другого спойлера ранее развернутый сворачивается).
>
> Обычно используется в качестве панели навигации.

Внешний вид



HTML

```
<div class="accordion">
    <input type="radio" name="accordion" id="actoggler1" checked>
    <label for="actoggler1">Сашими</label>
    <nav>
        <a href="#">Унаги</a>
        <a href="#">Хамачи</a>
    </nav>
</div>
```

```
<div class="accordion">
    <input type="radio" name="accordion" id="actoggler2">
    <label for="actoggler2">Суши</label>
    <nav>
        <a href="#">Чукка</a>
        <a href="#">Магуро</a>
        <a href="#">Тобико</a>
    </nav>
</div>
<div class="accordion">
    <input type="radio" name="accordion" id="actoggler3">
    <label for="actoggler3">Роллы</label>
    <nav>
        <a href="#">Кунсэй Батакон</a>
        <a href="#">Аригато</a>
        <a href="#">Мексиканский</a>
    </nav>
</div>
```

Каждый спойлер, входящий в состав аккордеона (его *отдел*), формируется следующими элементами (их вложенность друг в друга обозначена отступами):

- блоком (тегом `<div>`) со стилевым классом `accordion` — он создает сам отдел;
  - переключателем — он хранит текущее состояние отдела (свернут или развернут) и не выводится на экран.

    У всех переключателей, входящих в состав отделов одного аккордеона, задается одинаковое наименование (в атрибуте тега `name`) и уникальные якоря (обычно в формате *<наименование><порядковый номер>*).

    Переключатель, находящийся в отделе, который должен быть развернут изначально, делается изначально установленным (в его тег `<input>` записывается атрибут `checked`);
  - надписью (тегом `<label>`), связанной с переключателем того же отдела, — она выступает в качестве заголовка отдела;
  - панелью навигации (`<nav>`) с набором гиперссылок (`<a>`) — она создает содержимое отдела.

По щелчку на заголовке отдела связанный с ним переключатель устанавливается, в результате чего к содержимому соответствующего отдела аккордеона (тегу `<nav>`) применяется стиль, который выводит его на экран. При этом

ранее установленный переключатель сбрасывается, стиль перестает действовать на содержимое соответствующего ему отдела, и отдел сворачивается.

CSS

```
.accordion { border: 1px solid grey; }
.accordion > input { display: none; }
.accordion > label {
    display: block;
    color: white;
    background-color: grey;
    margin: 1px;
    padding: 3px;
    cursor: pointer;
}
.accordion > nav {
    overflow: hidden;
    padding: 3px;
}
.accordion > nav > a {
    display: block;
    margin: 5px;
}
.accordion> :not(input:checked) ~ nav {
    max-height: 0;
    transition: max-height 2s;
}
.accordion> input:checked ~ nav {
    max-height: 200px;
    transition: max-height 2s;
}
```

CSS-код аккордеона практически аналогичен таковому у спойлера (см. *разд. П2.4*).

Приложение 3. HTML-теги

Список HTML-тегов с переводом их имен приведен в табл. П3.1

Таблица П3.1

Тег	Расшифровка имени	Перевод имени
<!doctype>	Document type	Тип документа
<a>	Anchor	Якорь (подразумевается, что текущая страница как бы бросает якорь на другую)
<abbr>	Abbreviation	Сокращение, аббревиатура
<acronym>	Acronym	Акроним
<address>	Address	Адрес
<area>	Area	Область
<article>	Article	Статья
<aside>	Aside	Сбоку
<audio>	Audio	Аудио
<b>	Bold	Полужирный
<blockquote>	Block quote	Цитата
<body>	Body	Тело
 	Break	Разрыв
<button>	Button	Кнопка
<caption>	Caption	Подпись
<cite>	To cite	Ссылаться
<code>	Code	Код (например, на языке HTML или CSS)
<col>	Column	Колонка

Таблица П3.1 *(продолжение)*

Тег	Расшифровка имени	Перевод имени
<colgroup>	Column group	Группа колонок
<datalist>	Data list	Перечень данных
<dd>	Definition description	Описание определения (термина)
<del>	Deleted	Удаленный
<details>	Details	Подробности
<dfn>	Definition	Термин
	Division	Раздел
<dl>	Description list	Список описаний
<dt>	Definition term	Определение термина
	Emphasis	Акцент
<embed>	To embed	Вставлять, встраивать
<fieldset>	Field set	Набор полей (ввода)
<figcaption>	Figure caption	Подпись к иллюстрации
<figure>	Figure	Иллюстрация
<footer>	Footer	Поддон, нижний колонтитул
<form>	Form	Форма
<h1> ... <h6>	Heading	Заголовок
<head>	Head	Голова
<header>	Header	Шапка, верхний колонтитул
<hgroup>	Heading group	Группа заголовков
<hr>	Horizontal rule	Горизонтальная линия
<html>	HTML	
<i>	Italic	Курсивный
<iframe>	Inline frame	Встроенный фрейм
<img>	Image	Изображение
<input>	Input	Ввод
<ins>	Inserted	Вставленный

Таблица П3.1 *(продолжение)*

Тег	Расшифровка имени	Перевод имени
<kbd>	Keyboard	Клавиатура
<label>	Label	Пометка
<legend>	Legend	Легенда
	List item	Пункт списка
<link>	Link	Связь
<main>	Main	Главный
<map>	Map	Карта
<mark>	Mark	Отметка
<menu>	Menu	Меню
<meta>	Metadata	Метаданные
<meter>	Meter	Метр
<nav>	Navigation	Навигация
<object>	Object	Объект
	Ordered list	Упорядоченный список
<optgroup>	Option group	Группа вариантов
<option>	Option	Вариант
<output>	Output	Вывод
<p>	Paragraph	Абзац
<picture>	Picture	Изображение
<pre>	Preformatted	Предварительно отформатированный
<progress>	Progress	Прогресс
<q>	Quotation	Цитирование
<s>	Strikethrough	Зачеркнутый
<samp>	Sample	Пример
<search>	Search	Поиск
<section>	Section	Раздел, часть, глава, параграф, секция
<select>	To select	Выбирать

Таблица П3.1 *(окончание)*

Тег	Расшифровка имени	Перевод имени
<small>	Small	Маленький
<source>	Source	Источник
	Span	Интервал
	Strong	Веский
<style>	Style	Стиль
<sub>	Subscript	Нижний индекс
<summary>	Summary	Краткое содержание, сводка
<sup>	Superscript	Верхний индекс
<table>	Table	Таблица
<tbody>	Table body	Тело таблицы
<td>	Table datum	Величина (значение) в таблице
<textarea>	Text area	Область текста
<tfoot>	Table footer	Поддон таблицы
<th>	Table heading	Заголовок таблицы
<thead>	Table header	Шапка таблицы
<title>	Title	Название, заглавие
<tr>	Table row	Строка таблицы
<u>	Underlined	Подчеркнутый
	Unordered list	Неупорядоченный список
<var>	Variable	Переменная
<video>	Video	Видео
<wbr>	Word break	Разрыв слова

Приложение 4. Описание файлового архива

Файловый архив, сопровождающий книгу, размещен на сайте издательства «БХВ» по интернет-адресу: **https://zip.bhv.ru/9785977520072.zip**. Ссылка на него доступна и со страницы книги на сайте **https://bhv.ru/**.

Список папок и файлов, имеющихся в архиве (вложенность папок и файлов показана отступами):

- <*номер урока*> — папка с материалами урока с указанным *номером*. Состав папки:
  - !sources — папка с исходными материалами, необходимыми для выполнения упражнений текущего урока;
  - ex<*номер раздела*> — результаты выполнения упражнения, приведенного в разделе с указанным *номером*;
  - s<*номер раздела*> — результаты выполнения самостоятельных упражнений из раздела с указанным *номером*;
- A<*номер приложения*> — папка с материалами приложения с заданным *номером*. Состав этих папок такой же, как и у папок с материалами уроков (см. ранее);
- readme.txt — текстовый файл с описанием электронного архива.

Предметный указатель

З

И

К

Л

М

Н

О

П

Р

С

Т

У

Ф