Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
**«Владимирский государственный университет
имени А.Г. и Н.Г. Столетовых»**

ЯКУБОВИЧ Д.А.
ЕРОПОВА Е.С.
ЕРОПОВ И.А.

Основы WEB-разработки

Учебно-методическое пособие для лабораторных занятий

Владимир 2017

УДК 004.738
ББК 32.973-018.2
Я49

Якубович, Д. А

Я49 Основы WEB-разработки: учебно-методическое пособие для проведения лабораторных занятий / Д. А. Якубович, Е. С. Еропова, И. А. Еропов / Владимирский гос. ун-т имени Александра Григорьевича и Николая Григорьевича Столетовых. – Владимир : Издательство «Шерлок-пресс», 2017. – 102 с.

ISBN 978-5-9909534-6-8

В учебно-методическом пособии представлены 15 тем по основам WEB-верстки на базе технологий HTML5 и CSS3. Пособие предназначено для проведения лабораторных занятий со студентами педагогических вузов по дисциплинам «WEB-технологии», «Современные информационные технологии», «Информатика», «Информационные технологии в образовании».

Материал систематизирован и содержит многочисленные примеры. Учебное пособие может быть использовано для организации самостоятельной работы студентов и самообучения.

УДК 004.738
ББК 32.973-018.2

Рецензент: кандидат физико-математических наук, доцент кафедры вычислительной техники и систем управленияинститута информационных технологий и радиоэлектроники Владимирского государственного университета им. Александра Григорьевича и Николая Григорьевича Столетовых (ВлГУ)
А.В. Шутов

ISBN 978-5-9909534-6-8

Оглавление

Введение

В настоящее время WEB-технологии предоставляют широкий спектр возможностей пользователям и разработчикам. Благодаря высокой степени интерактивности и возможности работы с мультимедиа WEB-технологии активно занимают нишу сферы современного образования.

Среди указанных технологий важное место занимают вопросы верстки WEB-документов и сайтостроения. Наиболее распространенным инструментом по созданию сайтов являются технологии HTML и CSS.

В текущем пособии раскрыты базовые вопросы и приемы работы с указанными технологиями. Для удобства изучения материал поделен на два раздела.

В первом разделе рассматриваются основные вопросы WEB верстки на базе современной спецификации HTML5. Читатель познакомится с понятием технологии, способами и инструментами верстки сайтов, структуризации документа.

Второй раздел раскрывает базовые вопросы технологии каскадных таблиц стилей CSS3. Здесь изложены наиболее важные возможности технологии, ориентированные на качественное оформление WEB-страниц.

Каждое занятие сопровождается необходимым теоретическим минимумом, приведено множество примеров и иллюстраций.

Методическое пособие ориентировано на выработку системного подхода к планированию работы, развитие навыков рационального структурирования данных и поэтапной реализации проектов.

Материал предназначен для студентов педагогических ВУЗов; также он может быть рекомендован в качестве пособия для самообучения.

Раздел 1 Технология HTML5

1.1 Технология HTML. Спецификация HTML5

Понятие HTML

За последние 15 лет совершен большой скачек в области развития и продвижения глобальной сети Интернет. Сейчас она является неотъемлемой частью жизни, поскольку предоставляет оперативный доступ к необходимой информации, осуществляет функции коммуникации и многое другое.

Основными «точками», предоставляющими информацию в сети, являются *сайты*. По существу это развитие формы представления документации, но уже с использованием современных мультимедийных технологий и средств коммуникации. Обработка данных осуществляется посредством *браузера* – посредника между пользователем и сетью. Каким же образом браузер способен интерпретировать сайт именно в той форме, которая представлена разработчиком?

Очевидно, должен существовать общий механизм построения таких документов.

> ***HTML*** *(от англ.* ***HyperText Markup Language*** *— «язык гипертекстовой разметки») — стандартный язык разметки документов во Всемирной паутине.*

Большинство веб-страниц содержат описание разметки именно на языке HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа в удобной для человека форме.

> *Возможно, читатель ни разу не сталкивался с разработкой документов с помощью языка разметки. Между тем, подобная практика широко используется в типографиях и разработке достаточно сложных типов документации (и не только).*

Основным строительным элементом HTML является *тег*.

> ***Тег*** *– правило, согласно которому информация отображается в браузере. Теги окружены символами <>.*

По существу – теги и задают всю структуру вашего сайта, размечая его основные элементы (заголовки, абзацы, текст, блоки, таблицы, изображения и т.д.).

Выделяют парные и унарные теги.

Парные теги	Унарные теги
Представляют собой открывающий и закрывающий теги: `<тег> . . . </тег>` Как правило, парные теги определяют действие для внутреннего блока (установка размера шрифта, абзацев и т.п.), т.е. являются контейнерами.	Унарный тег не имеет закрывающей пары: `<тег>` Обычно он производит действие, не привязанное к конкретному блоку (например, переход на новую строку, прорисовка разделительной линии, вставка изображения).

HTML5

Во всемирной паутине длительное время использовались стандарты HTML 4.01, XHTML 1.0 и XHTML 1.1. Многие веб-страницы были свёрстаны с использованием смеси особенностей, представленных различными спецификациями.

HTML5 был создан как единый язык разметки, который мог бы сочетать синтаксические нормы HTML и XHTML. Он расширяет, улучшает и рационализирует разметку документов, а также добавляет огромный спектр возможностей для работы с мультимедиа (звуковое проигрывание, видео, анимация).

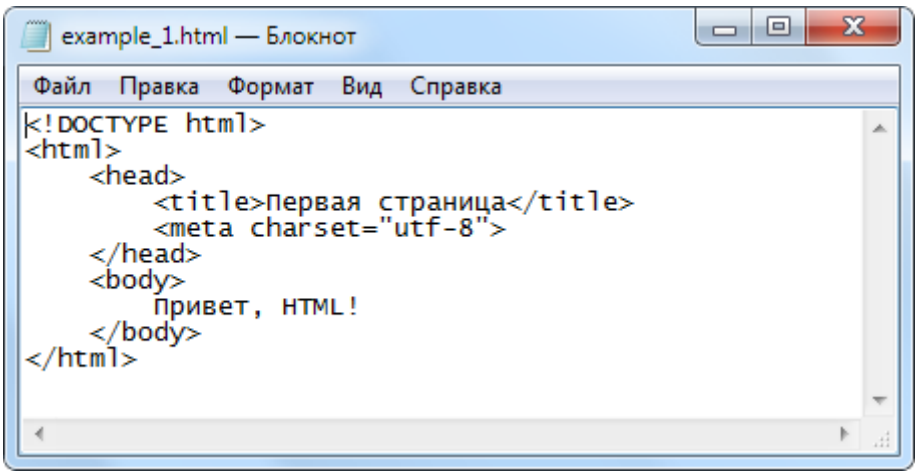
С 28 октября 2014 года консорциум W3C[1] официально рекомендует использовать стандарт языка HTML5.

Набор кода

Любой HTML-документ представляет собой текстовый файл, содержащий разметку будущего документа.

Набирать текст можно в простейшем текстовом редакторе – блокноте:

[1] W3C (World Wide Web Consortium) – организация, разрабатывающая и внедряющая технологические стандарты для Всемирной паутины.



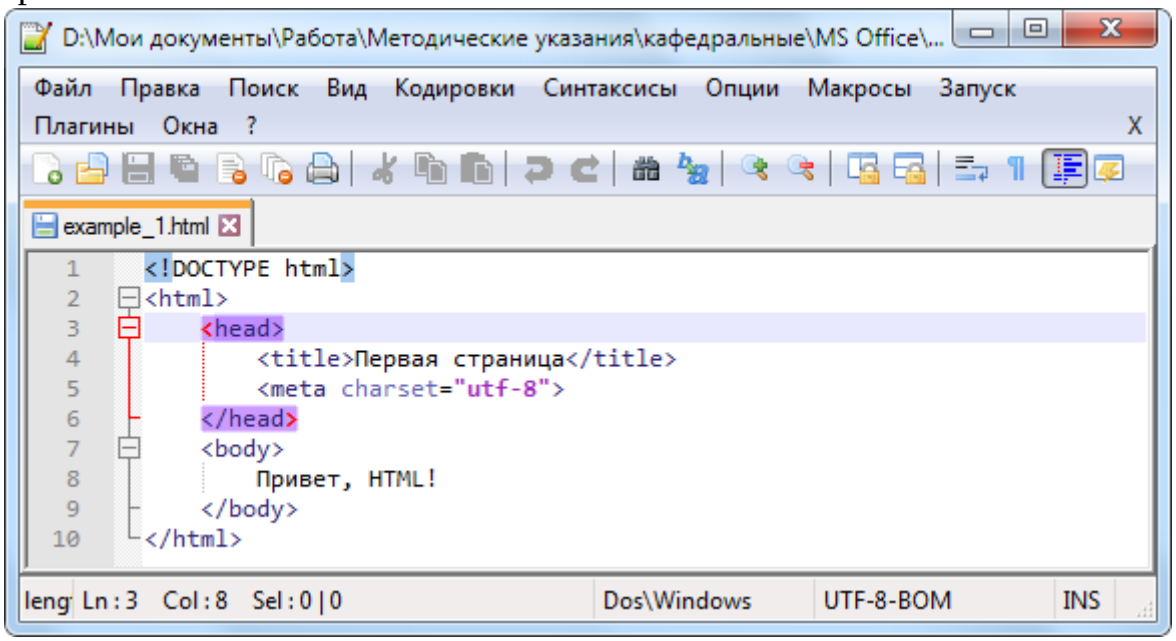
Использовать текстовые процессоры (например, Word) нельзя, поскольку такие файлы сохраняются с дополнительными данными.

> *Любая веб-страница имеет расширения* ***.htm*** *или* ***.html****. Поэтому обязательно необходимо изменить расширение файла с кодом на любое из указанных, чтобы затем запустить страницу в браузере.*

В дальнейшем такие файлы можно открывать, редактировать и сохранять в любом текстовом редакторе уже не меняя расширения.

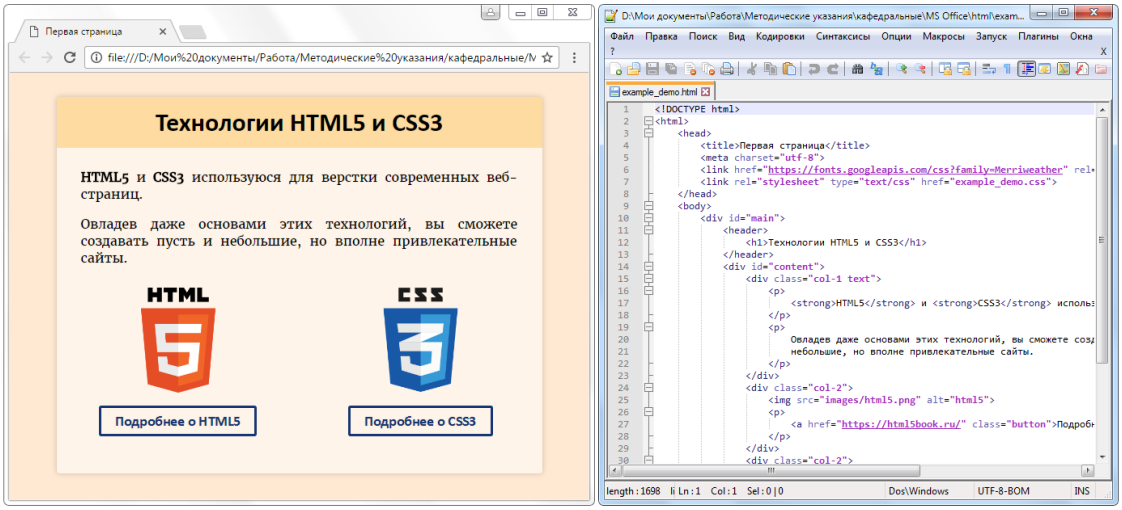
Для повышения эффективности работы можно использовать продвинутые текстовые редакторы: Notepad++, Sublime Text 3, Geany, Vim, Emacs и др. В частности, они поддерживают подсветку синтаксиса, автоподстановку и автозавершение команд, сценарии запуска в браузере и другие возможности.

Простым, удобным и бесплатно распространяемым является редактор Notepad++:



Набранную страничку можно открыть напрямую из Notepad++: для этого необходимо выбрать в контекстном меню «Запуск» и один из четырех представленных браузеров.

В процессе разработке удобно держать открытыми параллельно два окна: окно с кодом разметки и страницу, открытую в браузере. После написания логически завершенной части кода производится сохранение, а страница в браузере обновляется (F5), чтобы просмотреть текущий результат:



Стиль оформления кода

Браузеру безразлично, каким регистром набраны теги. Например, <HTML>, <html> или <HtMl> считается одной и той же командой.

> *Консорциум W3C рекомендует набирать теги строчными буквами.*

Сами теги можно писать подряд, через пробелы и на разных строчках: главное не разрывать команду тега. Однако лучше придерживаться общепринятого правила: вложенные блоки размещаются с некоторым отступом вправо. Таким образом лучше прослеживается структура документа.

Хороший стиль

```
<!DOCTYPE html>
<html>
    <head>
        <title>Первая страница</title>
        <meta charset="utf-8">
    </head>
    <body>
        Привет, HTML!
    </body>
</html>
```

Плохой стиль

```
<!DOCTYPE html>
<html><head>
        <title>Первая страница</title>
            <meta charset="utf-8">

</head>
<body>
Привет, HTML!</body>
</html>
```

Задания для самостоятельной работы

1. Откройте блокнот и наберите в нем код из примера выше. Сохраните документ в своей папке и переименуйте его в First.html (Если не отобража-

ются расширения документов, поставьте соответствующую галочку в настройках свойства папок).

2. Откройте документ любым доступным браузером.
3. Не закрывая браузер, измените в коде текст вкладки на свое имя. Сохраните изменения в блокноте и обновите страницу браузера.
4. Откройте файл с помощью Notepad++. Исследуйте дополнительные возможности редактора. Для запуска браузера в меню нажмите «Запуск» («Run») и выберете браузер, в котором будет отрыт документ. Настройка редактора доступна в разделе «Опции» («Options»). Также в контекстном меню доступно изменение кодировки документа (ANSI, UTF8 и т.д.). В случае отображения браузером русских символов кракозябрами добавьте в раздел <head> тег <meta charset="utf-8">, а в Notepad++ смените кодировку документа с ANSI на UTF-8.

1.2 Структура HTML-документа

Структура простого HTML-документа

Минимальный шаблон разметки документа согласно спецификации HTML5 следующий:

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title></title>
    </head>
    <body>

    </body>
</html>
```

Раздел DOCTYPE

HTML следует правилам, которые содержатся в файле объявления типа документа (Document Type Definition, или DTD). DTD представляет собой XML[2]-документ, определяющий, какие теги, атрибуты и их значения действительны для конкретного типа HTML. Для каждой версии HTML определен свой DTD.

[2] XML – расширяемый язык разметки. Можно рассматривать как формальный язык представления документации, понятный как человеку, так и эффективно обрабатываемый компьютером. В отличии от HTML теги могут задаваться любыми. Сам язык используется в разных сферах ИТ.

DOCTYPE-объявление располагается в самом начале кода HTML-документа и необходимо для переключения браузера в режим соответствия стандартам.

Для HTML5 указывается стандарт html.

Раздел HTML

Раздел HTML определяет специфику документа, содержание которого будет интерпретироваться браузером. Это корневой элемент документа. Он описывается парным тегом <html></html> и информирует браузер о том, что внутри тегов описан код WEB-страницы.

Все, что содержится вне этого блока, не является кодом разметки.

Раздел HEAD

Раздел описывается парным тегом <head></head>, является преамбулой документа. Здесь описывается важнейшая информация о способе представления данных и стилях, исполняемые скрипты, данные для поисковых систем.

Внутри раздела HEAD располагаются следующие элементы:

Тег	Описание
`<title></title>`	Обязателен. Предназначен для указания имени электронному документу. Указанный текст отображается на вкладке.
`<meta>`	Предоставляет необходимую информацию для браузеров и поисковых систем. В частности, задает кодировку страницы. Допускает многократное применение.

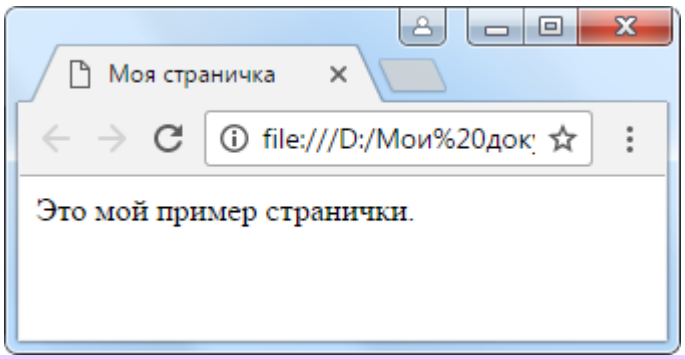
Раздел BODY

В разделе BODY реализовано «тело» документа. Здесь разметка задает то, что непосредственно отображается в окне браузера.

Пример

Пример простейшего документа с кодировкой UTF-8.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>
    <body>
        Это мой пример странички.
    </body>
</html>
```



Следите, чтобы кодировка .html-файла совпадала с кодировкой, указанной в теге <meta>! Для кодирования кириллических символов рекомендуется кодировка UTF-8.
Если символы в браузере отображаются кракозябрами, это верный признак того, что кодировки не совпадают.

Снова об HTML5

Исторически сложилось так, что парные теги <html>, <head> и <body> являлись важнейшими и обязательными в документе. Однако стандарт HTML5 менее строг в этом плане: указанная тройка может отсутствовать в разметке, и такой код является корректным!

В HTML5 присутствие тегов <html>, <head> и <body> необязательно! Однако в дальнейшем они будут присутствовать, чтобы лучше отражать структуру документа.

Атрибуты тегов

Тег – это команда. У тегов могут быть дополнительные свойства, называемые *атрибутами*.

Атрибуты *– дополнительные параметры, влияющие на работу тега.*

Их перечисляют после имени тега через пробел с определенными значениями; значения пишут в кавычках:

```
<тег атрибут1="значение" атрибут2="значение" ...>
```

Порядок перечисления атрибутов не важен.

Обычно у тегов есть собственные атрибуты. Существуют также *глобальные атрибуты*, доступные всем тегам (например, style, id, class).

Так, тег <body> обладает рядом атрибутов, задающих глобальные параметры фона, отступов, текста и гиперссылок.

Тег	Атрибуты	Описание
<body></body>		Тело документа.

	`bgcolor`	Задает цвет фона.

	`topmargin`
	`bottommargin`
	`leftmargin`
	`rightmargin`
		Задают размеры отступов от соответствующих границ. Значение можно задавать в процентах.

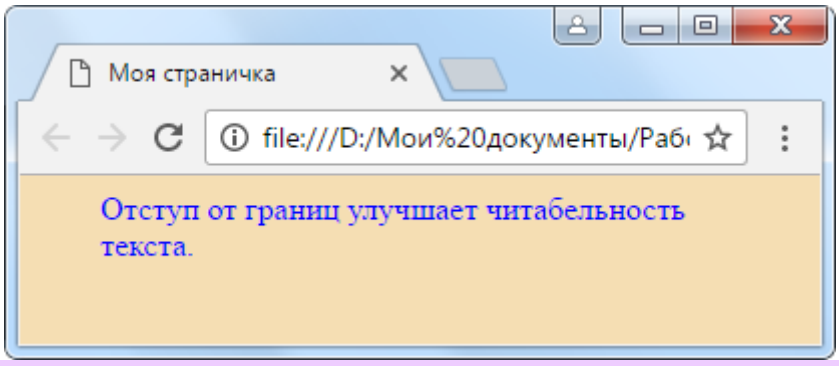
	`text`	Глобальный цвет текста.

	`link`
	`alink`
	`vlink`
		Цвета гиперссылок (основной, активной, посещенной).

Пример

Глобальное изменение цвета фона, текста и отступов от левой и правой границы.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>
    <body bgcolor="Wheat" text="Blue" leftmargin="40"
        rightmargin="40">
        Отступ от границ улучшает читабельность текста.
    </body>
</html>
```



Однако необходимо учитывать, что по стандарту HTML5 многие атрибуты считаются устаревшими, хотя и поддерживаются браузерами. К этому вопросу мы вернемся позднее.

Цвета в HTML

Цвета можно задавать тремя способами:

- по имени ("green");
- шестнадцатеричным кодом ("#A0A0A0");
- в формате RGB ("rgb(230,108,100)").

Первый способ наиболее ограничен по количеству возможных цветов; названия можно найти в HTML справочниках или таблицах. Второй и третий способ взаимоэквивалентны и работают по принципу смешивания трех компо-

нент: красный-зеленый-голубой. Минимальное значение каждой компоненты – 0 (00), максимальное – 255 (FF). Например, следующие команды определяют одинаковый цвет:

```
"rgb(179,36,175)"
"#B324AF"
```

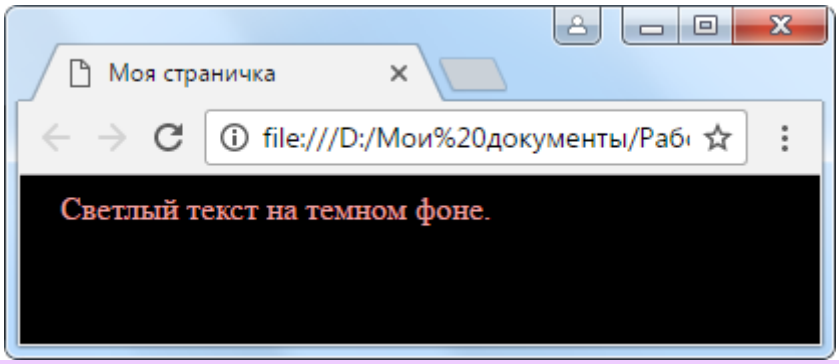
(в некоторых браузерах их работа может не совпадать, поэтому обычно используется шестнадцатеричное представление цвета).

Для получения кода необходимого цвета можно воспользоваться любым приложением, работающем с палитрой цветов (Word, Excel, PowerPoint, Paint, другой утилитой) или онлайн ресурсом, реализующим генератор цветов.

Пример

Разные способы определения цветов.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>
    <body bgcolor="Black" text="#F2A2A2" leftmargin="5%"
        rightmargin="5%">
        Светлый текст на темном фоне.
    </body>
</html>
```



Задания для самостоятельной работы

1. В сети Интернет найти 2-3 сайта, предоставляющих возможность получить коды цветов по таблицам или палитре.
2. Используя атрибуты тега <body> установите темный фон с синеватым оттенком, белым цветом текста, отступом слева на 100 и справа на 80 пикселей. Цвет подобрать в палитре.

1.3 Форматирование текста

Логическое и физическое форматирование текста

Теги для работы с текстом можно разделить на две категории:

- логическое форматирование (задают логическую структуру документа);
- физическое форматирование (определяют свойства отображаемых объектов).

Теги логического форматирования

Тег	Атрибуты	Описание
`<cite></cite>`	нет	Отмечает цитаты, высказывания, сноски. По умолчанию текст курсивом.
`<strong></strong>`	нет	Текст, на который требуется акцентировать внимание; браузеры отображают его жирным начертанием.
`<del></del>`	нет	Помечает текст как удаленный (например, устаревший). Браузеры отображают его как зачеркнутый.

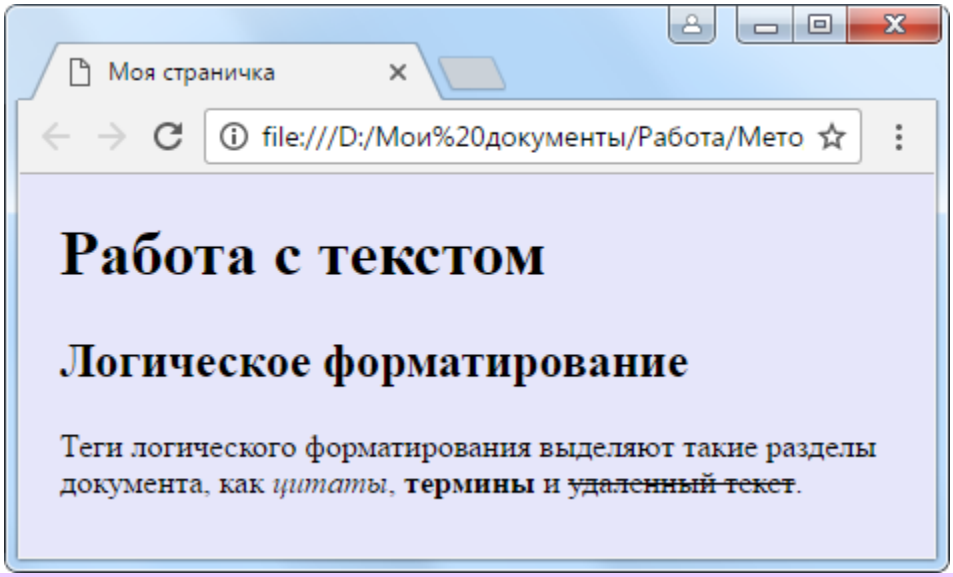
`<h1></h1>`
`<h2></h2>`
`<h3></h3>`
`<h4></h4>`
`<h5></h5>`
`<h6></h6>`
	нет	Теги заголовка. Блочный элемент, т.е. занимает доступную ширину. По умолчанию уровень 1 – наибольший по размеру текст, 6 – наименьший.

Пример

Логическое форматирование.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Моя страничка</title>
    </head>

    <body bgcolor="Lavender" leftmargin="20" rightmargin="20">
        <h1>Работа с текстом</h1>
        <h2>Логическое форматирование</h2>
        Теги логического форматирования выделяют такие разделы
        документа, как
        <cite>цитаты</cite>, <strong>термины</strong> и
        <del>удаленный текст</del>.
    </body>
```

```
</html>
```



Теги физического форматирования

Тег	Атрибуты	Описание
	нет	Жирное начертание шрифта.
	нет	Курсив.
<u></u>	нет	Подчеркивание.
<s></s>	нет	Зачеркивание.
<big></big>	нет	Текст несколько крупнее. Удален из HTML5.
	нет	Текст несколько мельче.
	нет	Текст относительно нижней линии (нижний индекс).
	нет	Текст относительно верхней линии (верхний индекс).

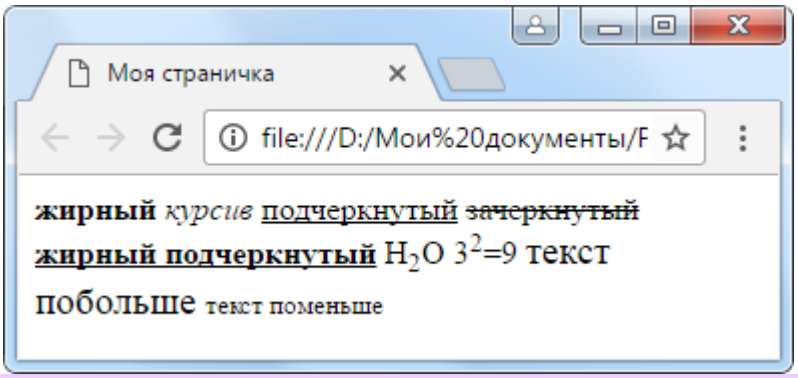
Теги можно группировать, вкладывая одни в другие. При этом обязательно необходимо соблюдать порядок вложения открывающих и закрывающих тегов. Сам порядок вложения пар не влияет на результат.

Пример

Физическое форматирование.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Моя страничка</title>
    </head>

    <body>
        <b>жирный</b>
        <i>курсив</i>
        <u>подчеркнутый</u>
```

```
        <s>зачеркнутый</s>
        <b><u>жирный подчеркнутый</u></b>
        H<sub>2</sub>O
        3<sup>2</sup>=9
        <big>текст побольше</big>
        <small>текст поменьше</small>
    </body>
</html>
```



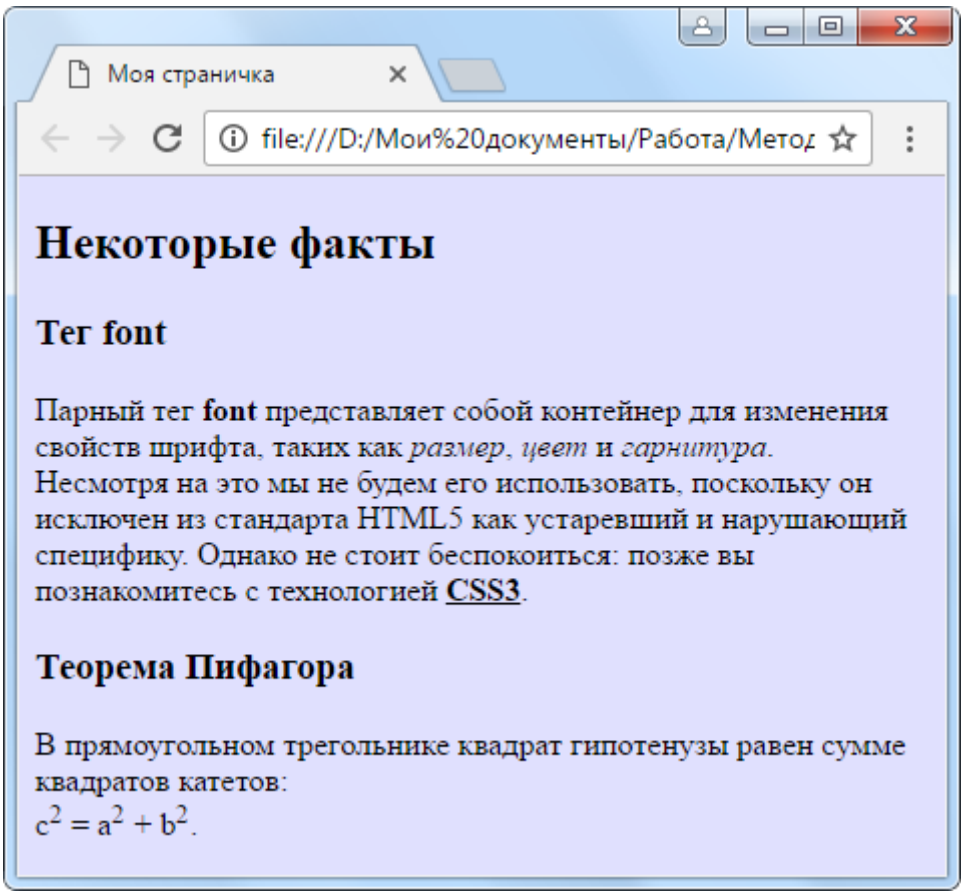
Разграничение ролей

У читателя может возникнуть вопрос: чем отличается, например, тег <cite> от <i>, либо <del> от <s>? Ведь браузеры отображают результат абсолютно одинаково.

Принципиальное отличие кроется в роли тегов. Так, первостепенная роль тегов логического форматирования помечать, что выделенный фрагмент текста имеет определенный смысл (заголовок, цитата, важное замечание и т.д.). Это важно, прежде всего, поисковым системам, которые собирают информацию согласно запросам пользователей. Теги физического форматирования не несут смысловой нагрузки: их главная задача – изменить оформление. Соответственно, поисковику они ни о чем не говорят.

Задания для самостоятельной работы

1. С помощью тегов логического и физического форматирования набрать нижеследующий документ. При необходимости для перехода на следующую строку используйте унарный тег <br>. Цвет фона и текста заданы через атрибуты тега <body>.



1.4 Структурное форматирование текста Специальные символы

Структурное форматирование в HTML – разбиение текстовых фрагментов электронного документа на логические блоки с информацией, которым соответствует определенный формат: абзац, текстовый блок, перенос строки, горизонтальный разделитель, предварительно отформатированный текст и комментарии.

Абзац, выравнивание, пробелы и новая строка

При работе с примерами вы могли заметить, что перевод каретки в коде на новую строку никак не отразится при отображении текста в браузере. Равно как и несколько пробелов между словами заменяются одним. В HTML это задается с помощью специальных тегов.

Тег	Атрибуты	Описание
`<p></p>`		Тег задает один абзац. По умолчанию браузеры добавляют небольшой отступ сверху и снизу.
	`align`	Задает выравнивание текста внутри абзаца. Доступны четыре режима:

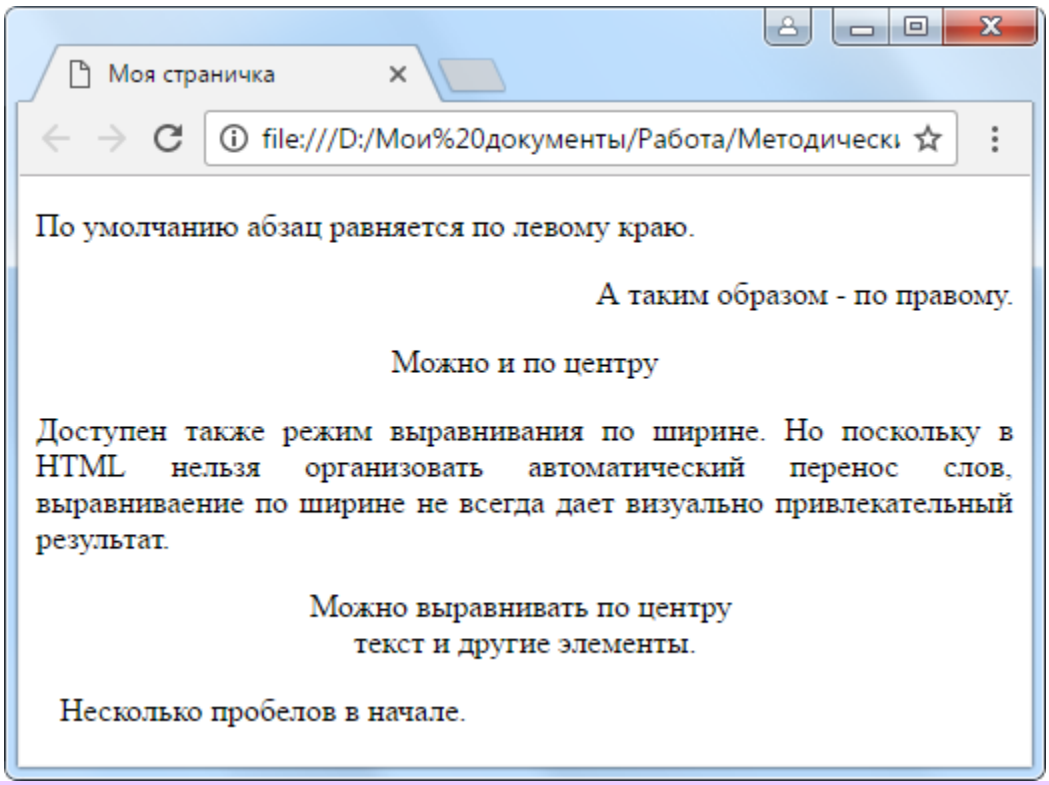
	• left (по левому краю (по умолчанию)); • right (по правому краю); • center (по центру); • justify (по ширине).
`<center></center>`	Выравнивает текст по центру. Удален из HTML5.
` `	Осуществляет перенос каретки на новую строчку.
` `	Кодовая инструкция, заменяющая один пробел.
`<nobr></nobr>`	Запрет разрыва фразы

Для центрирования элемента можно использовать как тег абзаца <p>, так и тег . Последний поддерживается браузерами, но из спецификации HTML5 удален.

Пример

Структурное форматирование текста.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Моя страничка</title>
    </head>

    <body>
        <p>По умолчанию абзац равняется по левому краю.</p>
        <p align="right">А таким образом - по правому.</p>
        <p align="center">Можно и по центру</p>
        <p align="justify">
            Доступен также режим выравнивания по ширине.
            Но поскольку в HTML нельзя организовать автоматический
            перенос слов, выравниваение по ширине не всегда дает
            визуально привлекательный результат.
        </p>
        <center>
            Можно выравнивать по центру
            <br>
            текст и другие элементы.
        </center>
        <p>   Несколько пробелов в начале.</p>
    </body>
</html>
```



Разделитель, дословный вывод и комментарии

Тег	Атрибуты	Описание
`<hr>`		Горизонтальная черта (разделитель).
	`width`	Длина (в пикселях или в процентах).
	`size`	Толщина (в пикселях).
	`align`	Выравнивание: • left; • right; • center.
	`color`	Цвет.
	`noshade`	Убрать эффект рельефности.
`<pre></pre>`		Предварительно отформатированный текст. Содержимое выводится в окне браузера так же, как и набрано в коде (со всеми пробелами и новыми строками). Браузер отображает его машинописным шрифтом.
`<!--  -->`		Комментарии в программном коде. Не отображаются в окне браузера и ни на что не влияют.
`&lt; &gt;`		Заменители угловых скобок < и >. В этом случае говорят, что производится экранирование символов. Оно необходимо, чтобы браузер смог отличить их от скобок тега или в случае вывода

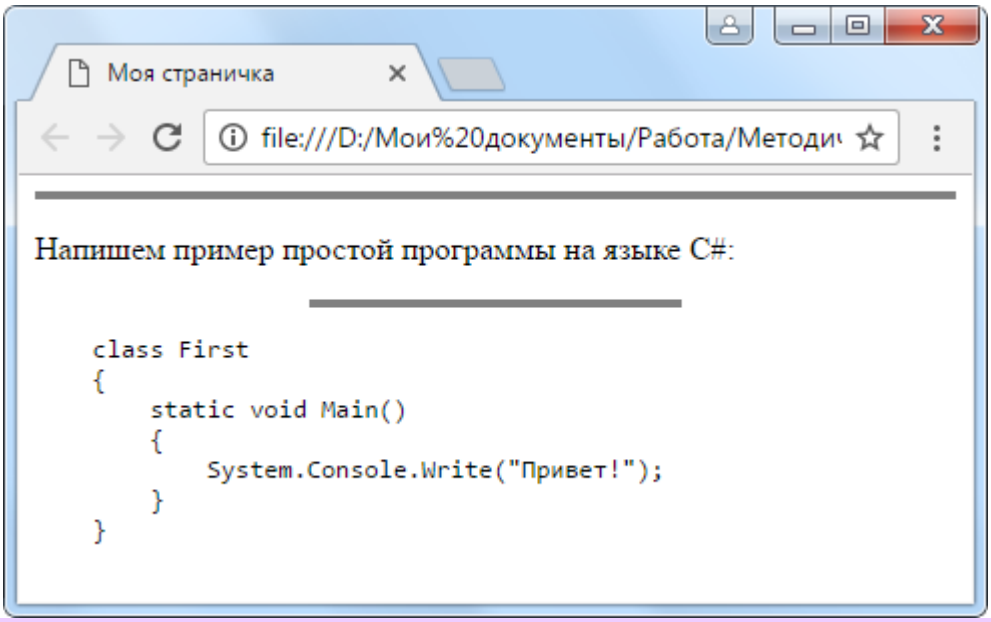
листинга HTML кода.

Комментарии обычно оставляет разработчик, поясняя себе или другим разработчикам происходящее в текущем блоке кода. Комментарии лишь влияют на размер файла с кодом WEB-страницы, поэтому перед релизом проекта их можно удалить.

Пример

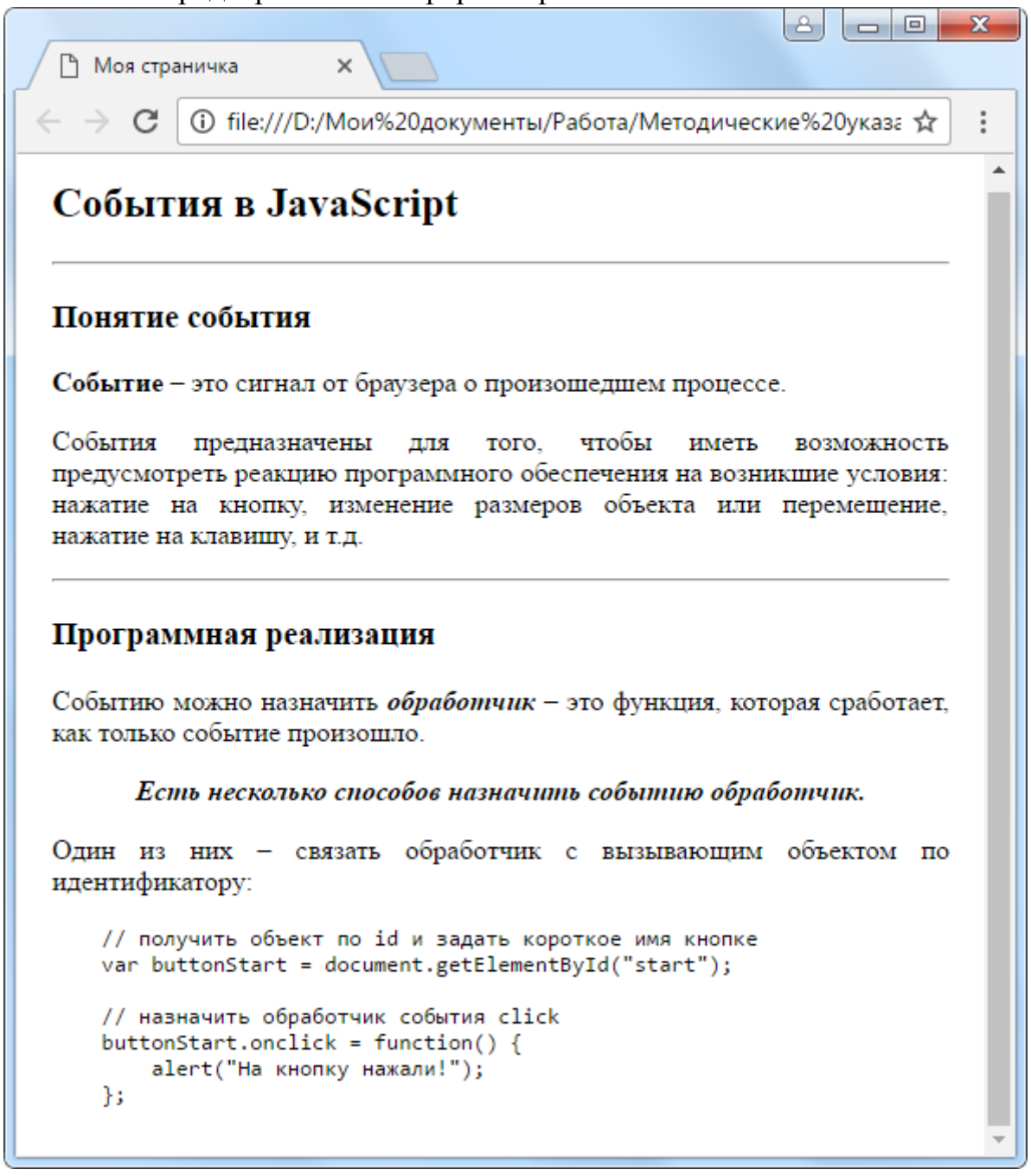
Структурное форматирование.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Моя страничка</title>
    </head>

    <body>
        <hr size="4" noshade>
        <p>Напишем пример простой программы на языке C#:</p>
        <hr width="40%" size="4" noshade>

        <!-- далее листинг программы -->
        <pre>
    class First
    {
        static void Main()
        {
            System.Console.Write("Привет!");
        }
    }
        </pre>
    </body>
</html>
```



Задания для самостоятельной работы

1. Наберите следующий ниже документ, используя абзацы, разделительные линии и предварительно отформатированный текст:



1.5 Списки

Списки

Спецификация HTML предусматривает три основных типа списков:

- нумерованные;
- маркированные;
- списки определений.

Нумерованные списки

Для создания нумерованного списка используется тег-контейнер (Ordered List), внутри которого располагаются пункты перечня, окруженные тегом (List Item):

Тег	Атрибуты	Описание
`<ol></ol>`		Нумерованный список.
	`start`	Номер, с которого начинается список (по умолчанию равен 1).
	`type`	Устанавливает вид маркера. Доступны варианты: • 1 (арабские числа (по умолчанию)); • i (строчные римские); • I (заглавные римские); • a (строчные буквы); • A (заглавные буквы).
	`reversed`	Установить обратный порядок отсчета.
`<li></li>`		Контейнер одного элемента списка.
	`value`	Начинает новый отсчёт с указанного номера.
	`type`	Устанавливает вид маркера (см. выше).

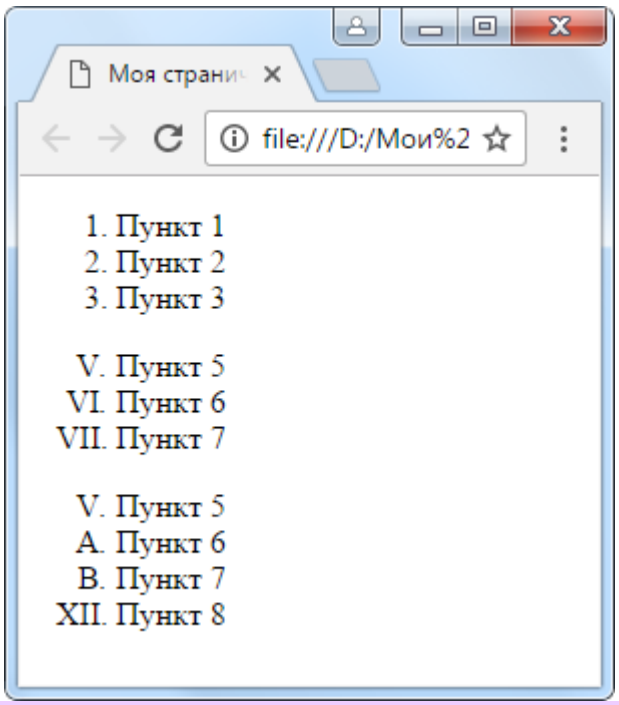
По умолчанию каждый элемент списка отображается в новой строке.

Пример

Нумерованные списки

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>

    <body>
        <!-- Стандартный список с автонумерацией-->
        <ol>
            <li>Пункт 1</li>
            <li>Пункт 2</li>
            <li>Пункт 3</li>
        </ol>

        <!-- Автонумерация римскими цифрами, начиная с V -->
        <ol start="5" type="I">
            <li>Пункт 5</li>
            <li>Пункт 6</li>
            <li>Пункт 7</li>
        </ol>
```

```
        <!-- Немного хаоса -->
        <ol start="5" type="I">
            <li>Пункт 5</li>
            <li value="1" type="A">Пункт 6</li>
            <li value="2" type="A">Пункт 7</li>
            <li value="12" type="I">Пункт 8</li>
        </ol>
    </body>
</html>
```



Маркированные списки

Маркированный список представляет собой неупорядоченный перечень (Unordered List) элементов. В качестве маркеров выступают специальные символы.

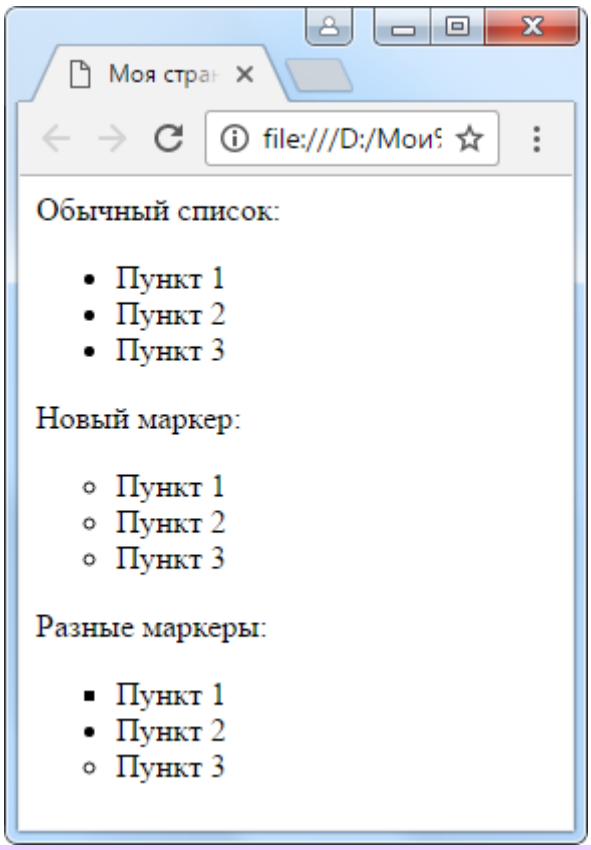
Для создания маркированных списков применяется тег-контейнер , внутри которого располагаются элементы самого списка, окруженные тегом-контейнером :

Тег	Атрибуты	Описание
`<ul></ul>`		Маркированный список.
	`type`	Устанавливает вид маркера. Доступны варианты: • disc (закрашенный круг (по умолчанию)); • circle (кольцо); • square (закрашенный квадрат).
`<li></li>`		Контейнер одного элемента списка.
	`type`	Устанавливает вид маркера (см. выше).

Пример

Маркерованные списки

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>

    <body>
         Обычный список:
         <ul>
             <li>Пункт 1</li>
             <li>Пункт 2</li>
             <li>Пункт 3</li>
         </ul>

         Новый маркер:
         <ul type="circle">
             <li>Пункт 1</li>
             <li>Пункт 2</li>
             <li>Пункт 3</li>
         </ul>

         Разные маркеры:
         <ul type="square">
             <li>Пункт 1</li>
             <li type="disc">Пункт 2</li>
             <li type="circle">Пункт 3</li>
         </ul>
    </body>
</html>
```



Списки определений

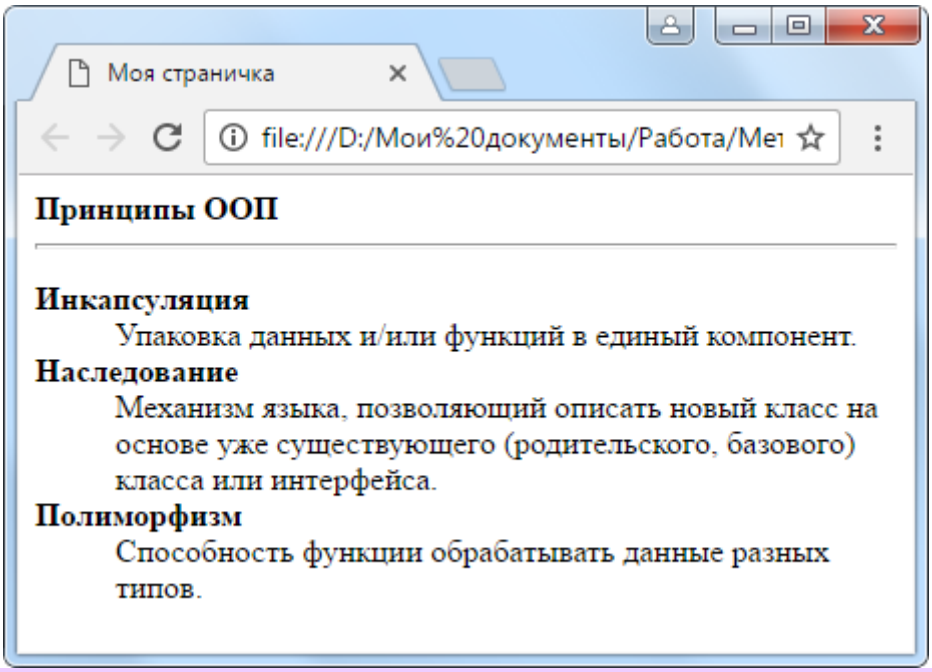
Список определений состоит из двух компонент – термина и определения. Область списка задается с помощью контейнера <dl></dl>, термин – тегом <dt></dt>, а его определение – с помощью тега <dd></dd>.

Пример

Списки определений.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>

    <body>
        <b>Принципы ООП</b>
        <hr size="3">

        <dl>
            <dt><b>Инкапсуляция</b><dt>
            <dd>Упаковка данных и/или функций в единый компонент.</dd>

            <dt><b>Наследование</b><dt>
            <dd>Механизм языка, позволяющий описать новый класс на
            основе уже существующего (родительского, базового)
            класса или интерфейса.</dd>
```

```
            <dt><b>Полиморфизм</b><dt>
            <dd>Способность функции обрабатывать данные разных
            типов.</dd>
        <dl>
    </body>
</html>
```



Вложенные списки

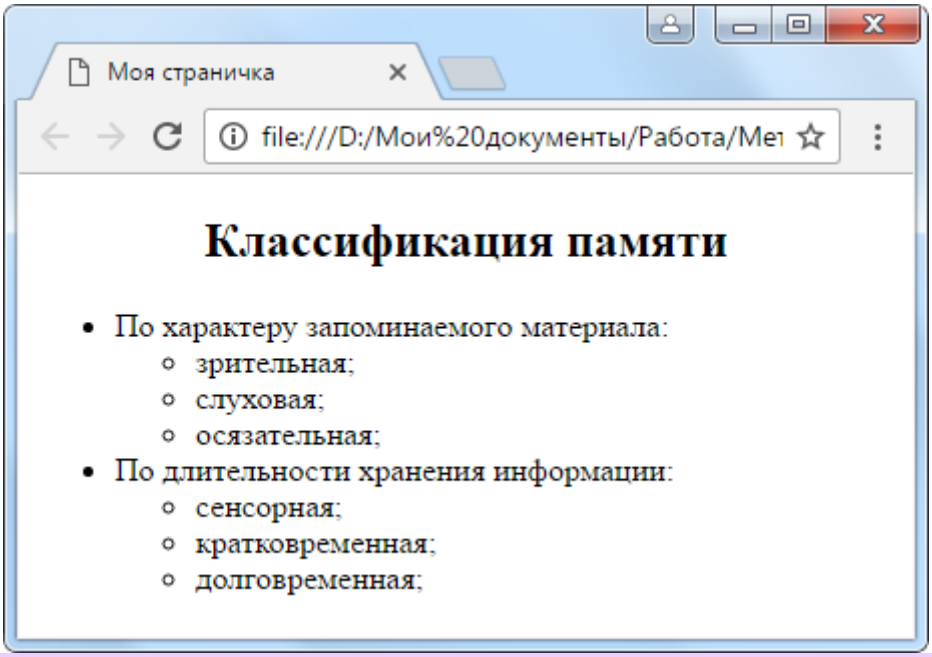
Вложение списков позволяет организовать списки сложной структуры. В частности, допускается вложение маркированных списков в нумерованные и наоборот.

Пример

Вложение списков.

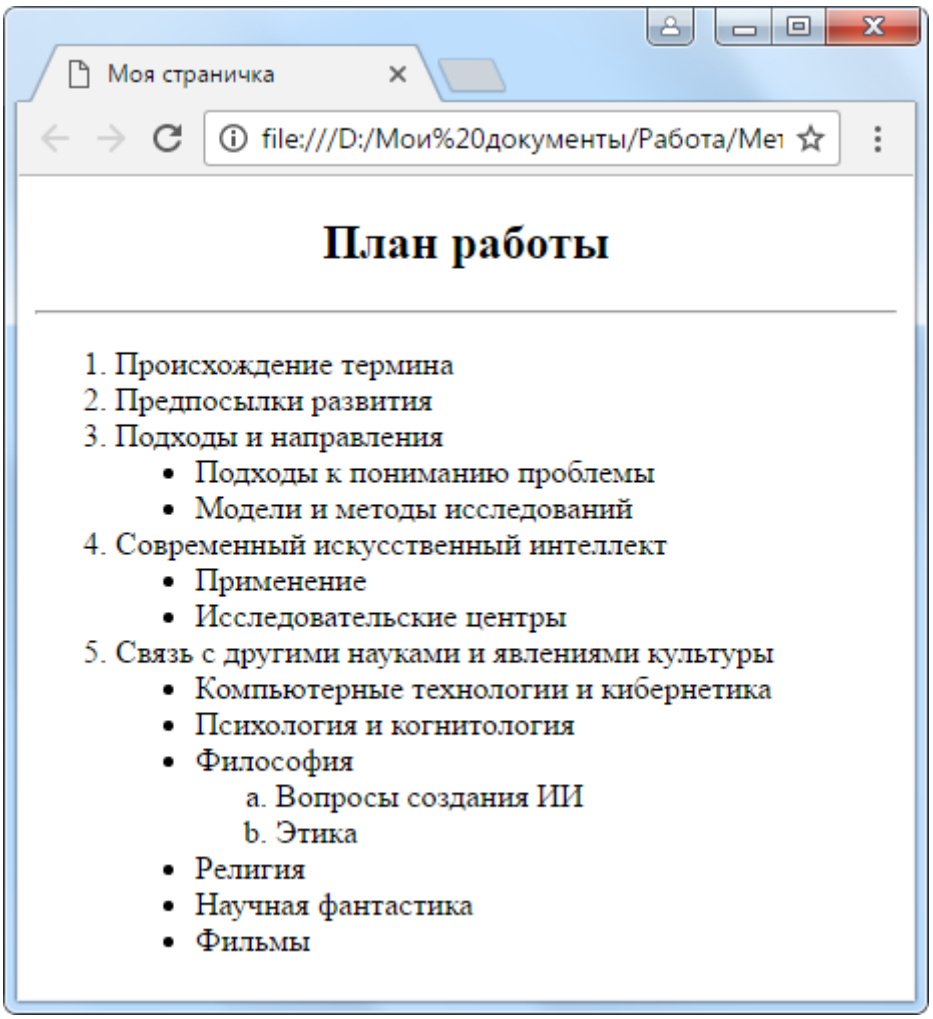
```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>

    <body>
        <h2><center>Классификация памяти</center></h2>

        <ul>
            <li>По характеру запоминаемого материала:
                <ul>
                    <li>зрительная;</li>
                    <li>слуховая;</li>
                    <li>осязательная;</li>
                </ul>
```

```
            </li>

            <li>По длительности хранения информации:
                <ul>
                    <li>сенсорная;</li>
                    <li>кратковременная;</li>
                    <li>долговременная;</li>
                </ul>
            </li>
        </ul>
    </body>
</html>
```



Задания для самостоятельной работы

1. Оформить следующий список:



1.6 Гиперссылки

Внешняя ссылка

Гипертекстовая ссылка *– это указатель адреса в глобальной сети, по которому можно перейти из окна браузера.*

Корректный переход по ссылке возможен, если:

1) ресурс, на который ссылаемся, функционирует в реальном времени;
2) ссылка корректно оформлена.

Любая гиперссылка состоит из двух частей:

- указателя ссылки («якоря»);
- адреса ресурса.

По умолчанию браузеры выделяют ссылку подчеркиванием и синим цветом. При наведении курсора мыши на ссылку указатель принимает вид руки с указательным пальцем. В качестве указателя ссылки может выступать текст (отдельное слово, фразы и страницы текста), изображение, таблица и т.д. Допускается также объединение графики и текста в рамках единого указателя ссылки.

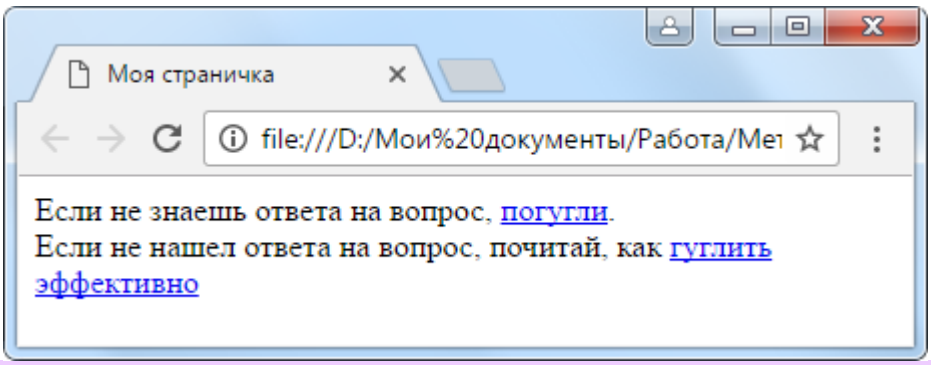
Указатель ссылки описывается тегом <a>.

Тег	Атрибуты	Описание
`<a></a>`		Задает ссылку.
	`download`	При его наличии браузер не переходит по ссылке, а предлагает скачать указанный по ссылке документ.
	`href`	Задает адрес, по которому следует перейти. Обязательный атрибут. Адрес может быть абсолютным, т.е. задавать весь путь к внешнему ресурсу (`href="http://vk.com"`) и относительный, содержащий ссылку на некоторую страницу (`href="secondPage.html"`).
	`name`	Задает имя якорю. Используется для организации внутренних переходов по странице.
	`target`	Указывает, в каком окне открыть страницу: • _blank (в новом окне); • _self (в текущем окне).
	`title`	Всплывающая подсказка.

Пример

Открытие ссылки в текущем и новом окне.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>

    <body>
        Если не знаешь ответа на вопрос,
        <a href="http://www.google.com">погугли</a>.
        <br>
        Если не нашел ответа на вопрос, почитай, как
        <a
href="http://www.runetica.com/profiles/google/cheatsheet.html"
              target="_blank">гуглить эффективно</a>
    </body>
</html>
```



Внутренняя ссылка

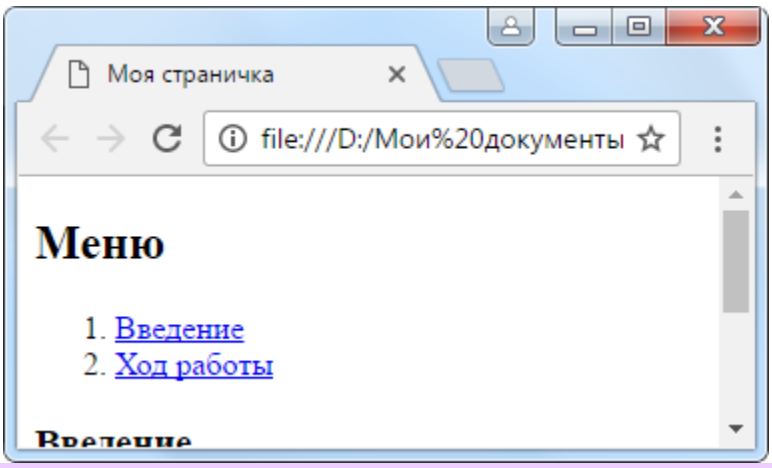
Внутренние ссылки организовывают переходы внутри страницы (это может потребоваться, когда на ней много материала и используется полоса прокрутки).

Структура внутренней гиперссылки включает две части – сама ссылка и ее именной идентификатор (# плюс имя элемента), называемый *якорем*. По нажатию на ссылку содержимое окна прокручивается до положения якоря.

Пример

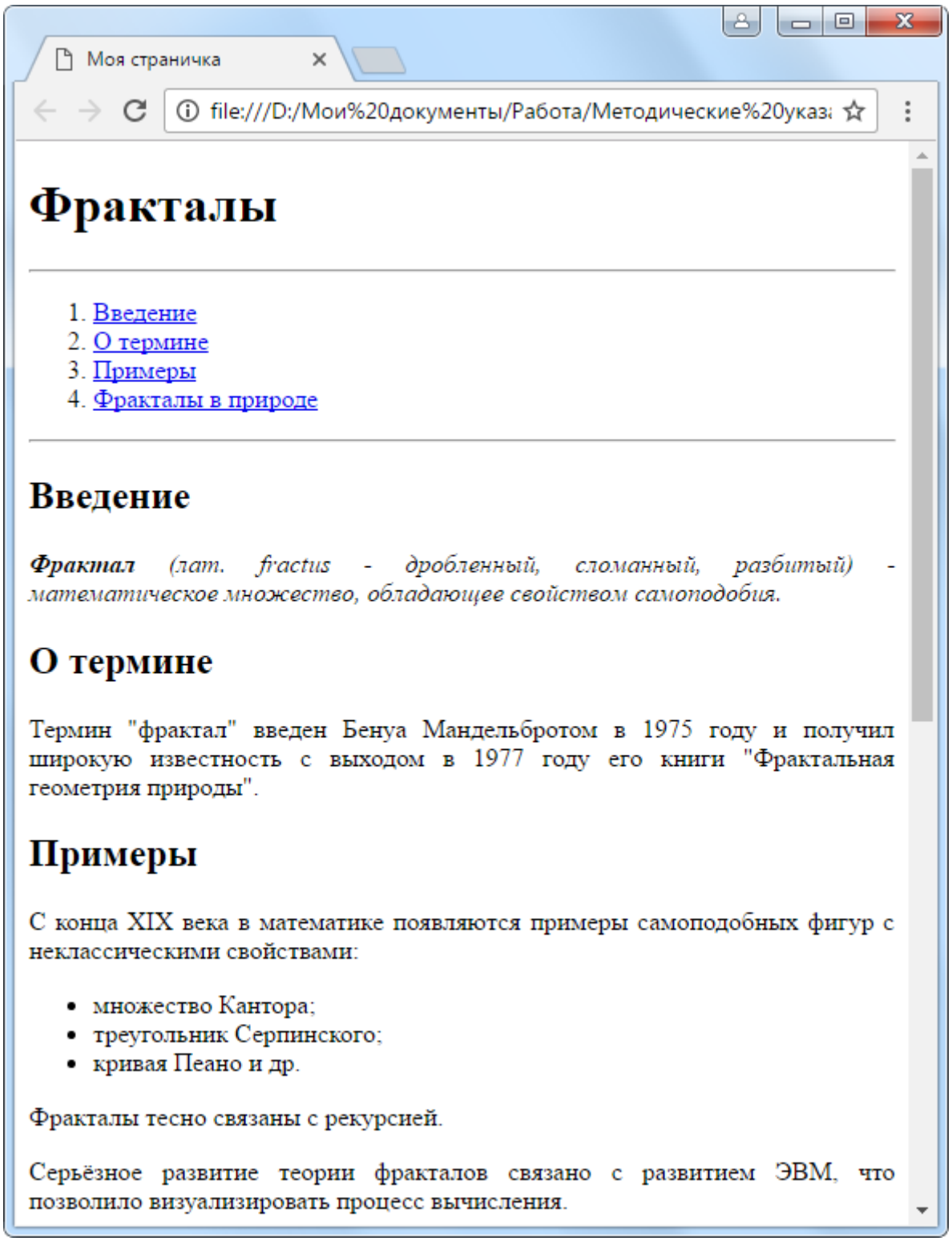
Открытие ссылки в текущем и новом окне.

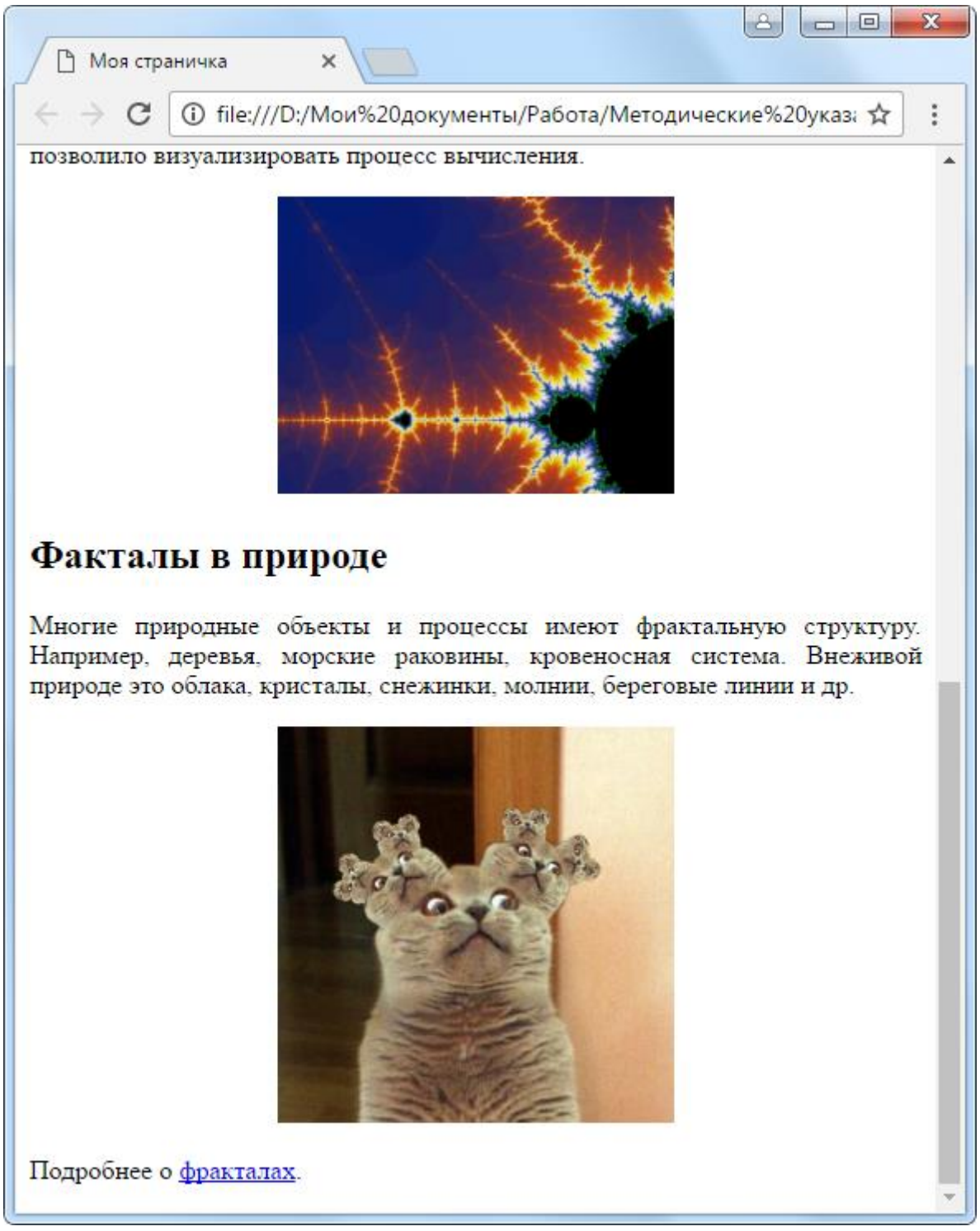
```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>

    <body>
        <h2>Меню</h2>
        <ol>
            <li><a href="#theme1">Введение</a></li>
            <li><a href="#theme2">Ход работы</a></li>
        </ol>
        <a name="theme1"></a>
        <h3>Введение</h3>
        . . . . . .
        <a name="theme2"></a>
        <h3>Ход работы</h3>
        . . . . . .
    </body>
</html>
```



Задания для самостоятельной работы

1. Внимательно изучите механизмы создания внешних и внутренних гиперссылок на примерах.
2. Наберите код из примера по внешним гиперссылкам.
3. Используя полученные ранее знания, набрать следующий документ. Переход на подпункты меню должен осуществляться через внутренние ссылки. Ссылка в конце осуществляет переход на статью в Wikipedia. Тег <img> для вставки изображения описан в разделе 3.8.


Моя страничка
file:///D:/Мои%20документы/Работа/Методические%20указ
Фракталы
1. Введение
2. О термине
3. Примеры
4. Фракталы в природе
Введение
Фрактал (лат. fractus - дробленный, сломанный, разбитый) - математическое множество, обладающее свойством самоподобия.
О термине
Термин "фрактал" введен Бенуа Мандельбротом в 1975 году и получил широкую известность с выходом в 1977 году его книги "Фрактальная геометрия природы".
Примеры
С конца XIX века в математике появляются примеры самоподобных фигур с неклассическими свойствами:
• множество Кантора;
• треугольник Серпинского;
• кривая Пеано и др.
Фракталы тесно связаны с рекурсией.
Серьёзное развитие теории фракталов связано с развитием ЭВМ, что позволило визуализировать процесс вычисления.



1.7 Таблицы

Основным тегом таблицы является <table></table>. Любая таблица состоит из строк (<tr></tr>) и столбцов (<td></td>). Ячейке <td> есть альтернатива <th>, выполняющая роль заголовка; по умолчанию браузеры отображают ее текст жирным начертанием и выравнивают по центру.

Атрибуты таблицы

Тег	Атрибуты	Описание
`<table></table>`		
	`align`	Выравнивает таблицу: • left (по умолчанию); • center; • right.
	`border`	Толщина рамки в пикселах. По умолчанию равна 0 (нет рамки).
	`bgcolor`	Цвет фона таблицы.
	`background`	Ссылка на изображение в качестве фона (всегда в натуральный размер).
	`bordercolor`	Цвет границы.
	`cellpadding`	Расстояние между границей ячейки и ее содержимым. По умолчанию 0.
	`cellspacing`	Задает расстояние между внешними границами ячеек.
	`height`	Высота таблицы.
	`width`	Ширина таблицы.

По умолчанию таблица минимизирует ячейки согласно их содержимому и отступам. Если задана определенная ширина таблицы, размеры ячеек не указаны фиксировано, а их содержимое не прилегает к границам плотно, то колонки задаются равной ширины.

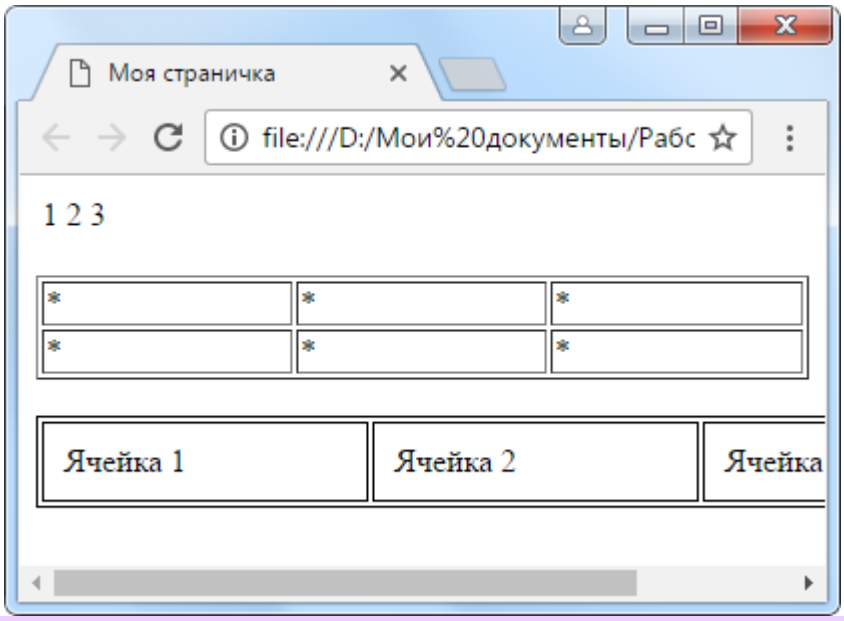
Отметим, что ширину колонок достаточно задать ячейкам первой строки.

Пример

Таблицы.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>

    <body>
        <table>
            <tr>
                <td>1</td>
                <td>2</td>
                <td>3</td>
            </tr>
        </table>

        <br>
```

```
        <table border="1" width="100%">
            <tr>
                <td>*</td>
                <td>*</td>
                <td>*</td>
            </tr>
            <tr>
                <td>*</td>
                <td>*</td>
                <td>*</td>
            </tr>
        </table>

        <br>

        <table align="center" border="1" bordercolor="black"
            cellpadding="10" width="500">
            <tr>
                <td>Ячейка 1</td>
                <td>Ячейка 2</td>
                <td>Ячейка 3</td>
            </tr>
        </table>

    </body>
</html>
```



Атрибуты строк и ячеек

Тег	Атрибуты	Описание
`<tr></tr>`		Задает одну строку таблицы.
	`align`	Горизонтальное выравнивание содержимого ячеек в строке:

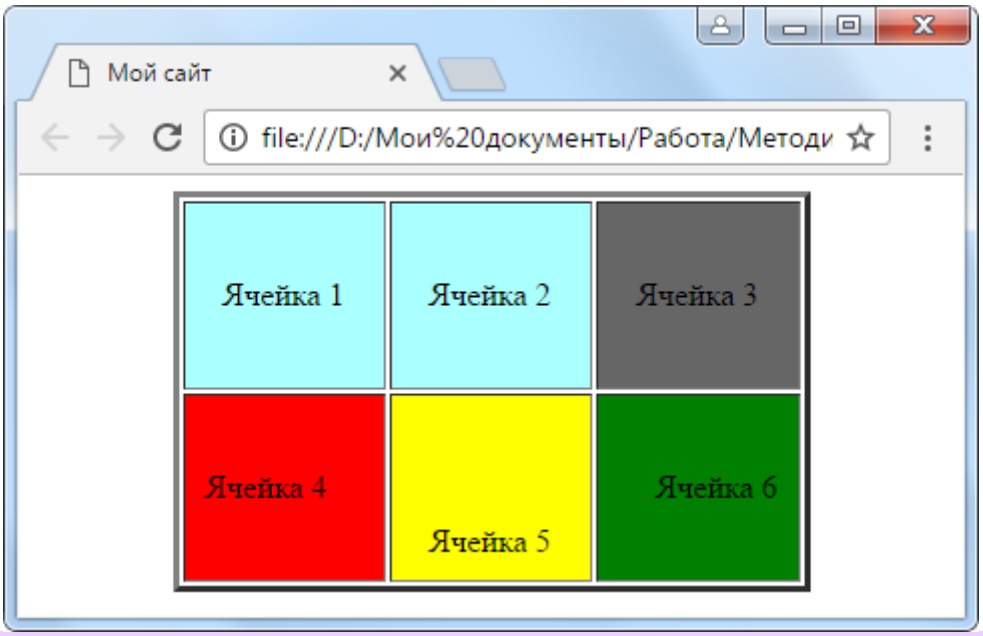
		• left (по умолчанию); • center; • right; • justify.
	bgcolor	Цвет фона строки.
	bordercolor	Цвет рамки вокруг строки.
	valign	Вертикальное выравнивание содержимого ячеек в строке: • top; • middle (по умолчанию); • bottom; • baseline.
<td></td>		Задает одну ячейку.
	align	Горизонтальное выравнивание текста в ячейке: • left (по умолчанию); • center; • right; • justify.
	bgcolor	Цвет фона ячейки.
	background	Ссылка на изображение в качестве фона ячейки.
	bordercolor	Цвет рамки ячейки.
	height	Высота ячейки.
	width	Ширина ячейки.
	valign	Вертикальное выравнивание содержимого ячейки: • top; • middle (по умолчанию); • bottom; • baseline.

Пример

Таблицы.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Мой сайт</title>
    </head>

    <body>
        <table align="center" border="3" cellpadding="10"
            width="70%" height="200">
```

```
            <tr align="center" bgcolor="#AAFFFF">
                <td>Ячейка 1</td>
                <td>Ячейка 2</td>
                <td bgcolor="#666666">Ячейка 3</td>
            </tr>
            <tr>
                <td align="left" bgcolor="red">Ячейка 4</td>
                <td align="middle" valign="bottom" bgcolor="yellow">
                    Ячейка 5</td>
                <td align="right" bgcolor="green">Ячейка 6</td>
            </tr>
        </table>
    </body>
</html>
```



Объединение ячеек в таблице

Атрибуты <td> colspan и rowspan используются при построении таблиц, когда возникает необходимость объединения ячеек таблицы.

- colspan показывает, на сколько ячеек по горизонтали следует расширить <td> (<th>);
- rowspan объединяет ячейки заданного столбца по строкам.

Чтобы проще было прорисовывать структуру такой таблицы, постройте изначально всю сетку. Далее расширяйте требуемую ячейку по строке / столбцу и сразу же удаляйте лишние ячейки (которые вошли в объединение). Также в сети Интернет можно найти ресурсы, способные генерировать код таблицы по визуально заданной форме.

Пример

Объединение ячеек таблицы.

```
<!DOCTYPE html>
<html>
    <head>
```

```
    <meta charset="utf-8">
    <title>Мой сайт</title>
</head>

<body>

    Искомая таблица:
    <table align="center" border="1" width="100px">
        <tr>
            <td>1</td>
            <td>2</td>
        </tr>
        <tr>
            <td>3</td>
            <td>4</td>
        </tr>
    </table>

    Объединяем два столбца первой строки:
    <table align="center" border="1" width="100px">
        <tr>
            <td colspan="2">1</td>
        </tr>
        <tr>
            <td>3</td>
            <td>4</td>
        </tr>
    </table>

    Объединяем две строки второго столбца:
    <table align="center" border="1" width="100px">
        <tr>
            <td>1</td>
            <td rowspan="2">2</td>
        </tr>
        <tr>
            <td>3</td>
        </tr>
    </table>

    Комбинируем возможности (из таблицы 4x4):
    <table align="center" border="1" width="100px">
        <tr>
            <td colspan="2">*</td>
            <td colspan="2">*</td>
        </tr>
        <tr>
            <td rowspan="3">*</td>
            <td rowspan="3">*</td>
            <td colspan="2">*</td>
        </tr>
```

```
            <tr>
                <td>*</td>
                <td>*</td>
            </tr>
            <tr>
                <td>*</td>
                <td>*</td>
            </tr>
        </table>

    </body>
</html>
```

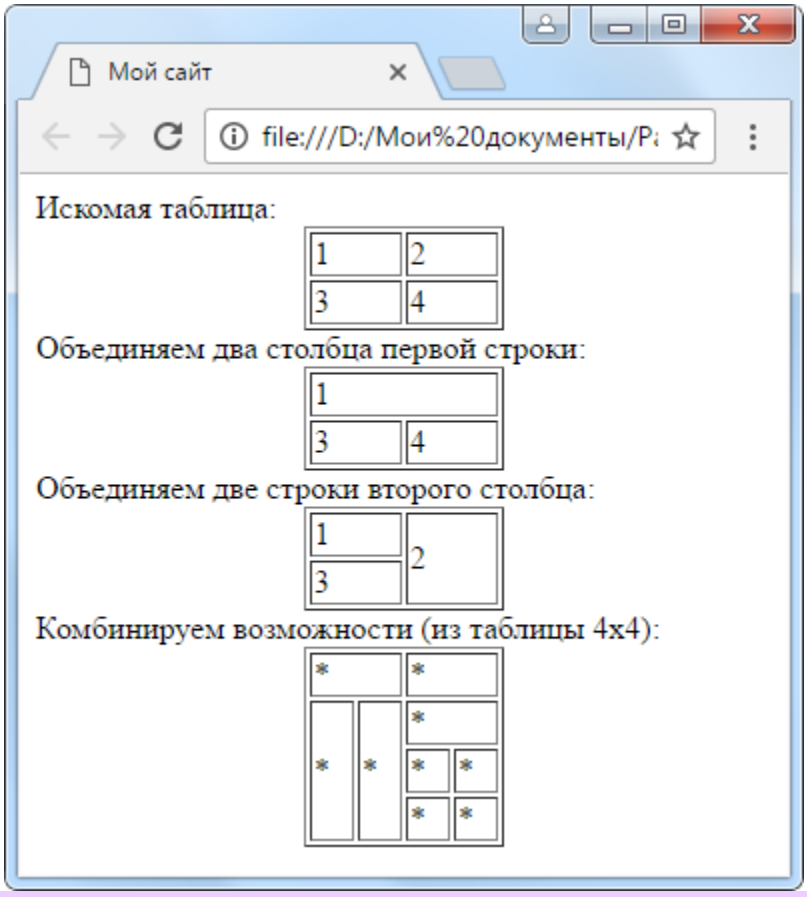


Таблица в разметке структуры страницы

Таблицы можно использовать в качестве каркаса всего сайта. Обычно в этом случае толщину рамки берут равной нулю. Таким образом пользователь не видит таблицу, однако элементы в ячейках будут располагаться требуемым образом.

> *Использовать таблицы в качестве инструмента разметки сайта не рекомендуется. Эту задачу сейчас решает блочная разметка по технологии CSS. Таблицы необходимо использовать по назначению – как носитель структурированных данных.*

Задания для самостоятельной работы

1. Наберите следующий документ:

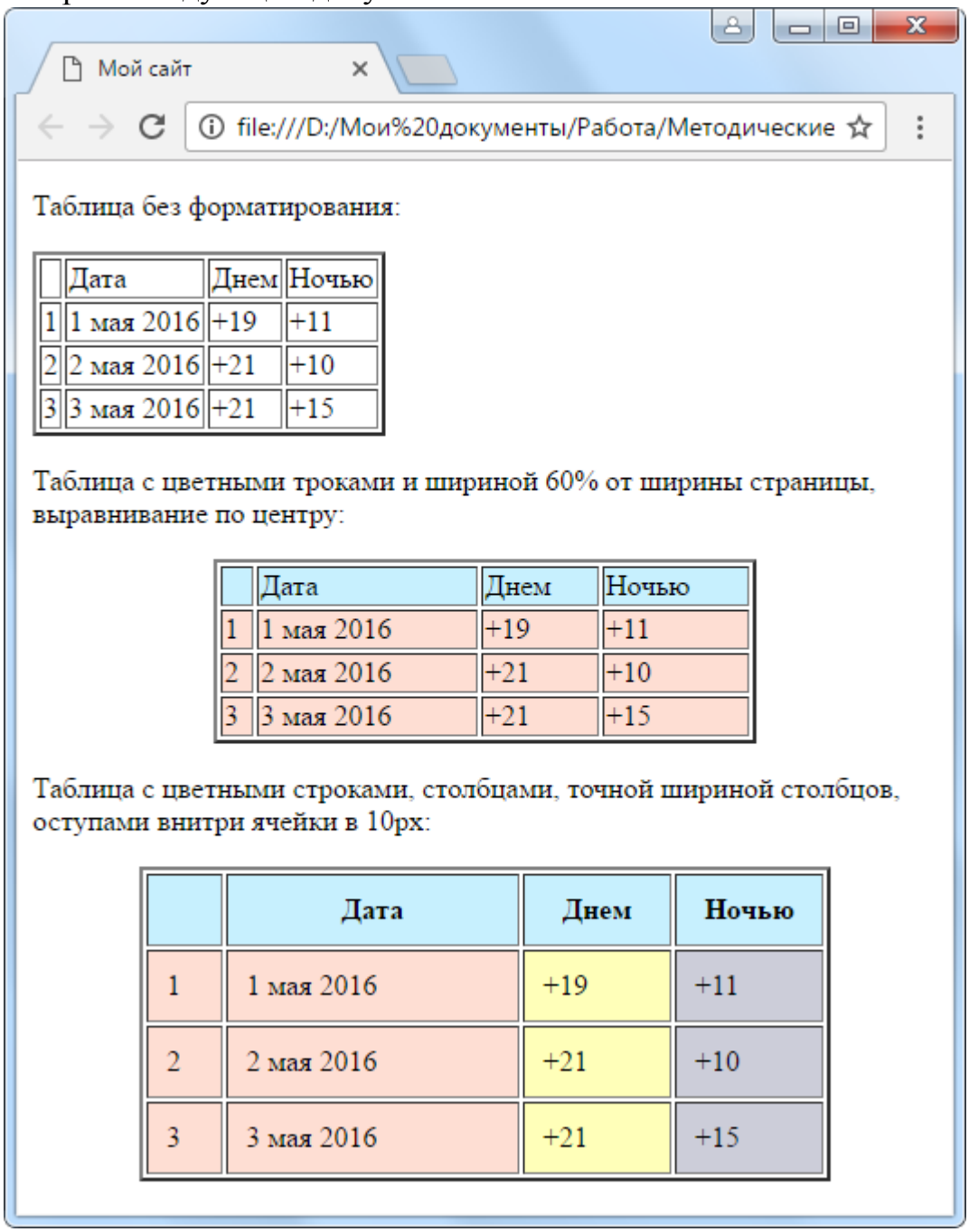


Таблица без форматирования:

	Дата	Днем	Ночью
1	1 мая 2016	+19	+11
2	2 мая 2016	+21	+10
3	3 мая 2016	+21	+15

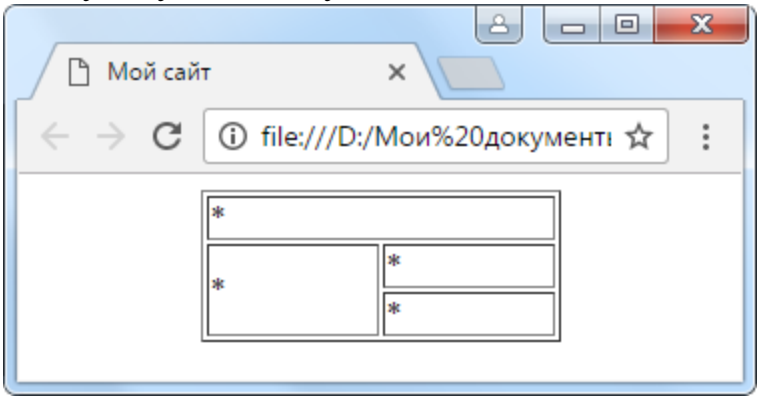
Таблица с цветными троками и шириной 60% от ширины страницы, выравнивание по центру:

	Дата	Днем	Ночью
1	1 мая 2016	+19	+11
2	2 мая 2016	+21	+10
3	3 мая 2016	+21	+15

Таблица с цветными строками, столбцами, точной шириной столбцов, оступами внутри ячейки в 10px:

	Дата	Днем	Ночью
1	1 мая 2016	+19	+11
2	2 мая 2016	+21	+10
3	3 мая 2016	+21	+15

2. Постройте следующую таблицу:



1.8 Изображения

Для вставки графических изображений в HTML-документы используется унарный тег <img>. Изображение является плавающим элементом, поэтому в случае выравнивания другие объекты его будут обтекать.

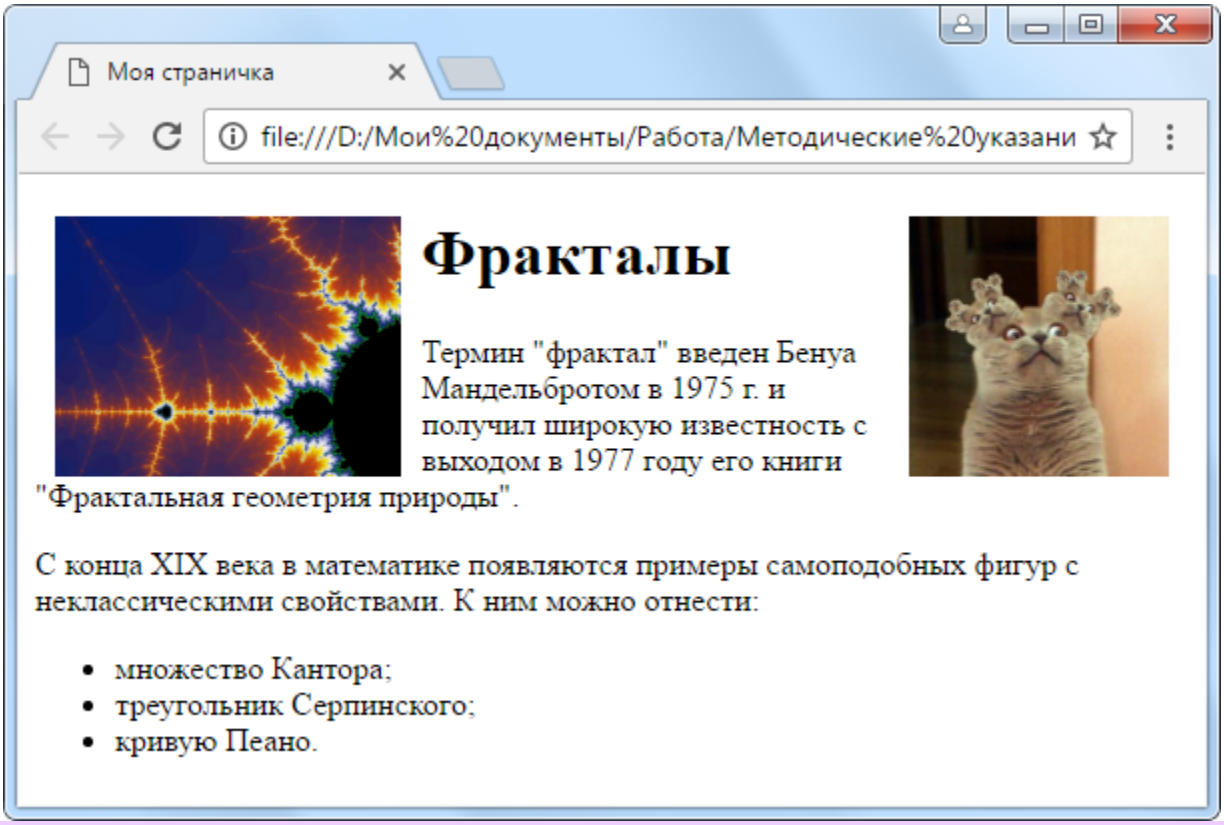
Тег	Атрибуты	Описание
<img>		Вставляет изображение.
	align	Задает положение относительно обтекающего текста: • left; • middle; • right; • top; • bottom (по умолчанию).
	alt	Альтернативный текст. Отображается вместо изображения, если оно не найдено или не полностью загружено.
	border	Задает толщину границы (по умолчанию равен 0).
	height	Высота изображения (в пикселах или процентах).
	hspace	Горизонтальный невидимый отступ от границы.
	src	Имя или ссылка на изображение.
	vspace	Вертикальный невидимый отступ от границы.
	width	Ширина таблицы (в пикселах или процентах).

Пример

Вставка изображения.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>

    <body>
        <img src="fractal_1.jpg" height="130" align="left"
            hspace="10">
        <img src="fractal_2.jpg" height="130" align="right"
            hspace="10">
        <h1>Фракталы</h1>
        <p>
```

```
            Термин "фрактал" введен Бенуа Мандельбротом в 1975 г.
            и получил широкую известность с выходом в 1977 году
            его книги "Фрактальная геометрия природы".
        </p>
        <p>
            С конца XIX века в математике появляются примеры
            самоподобных фигур с неклассическими свойствами.
            К ним можно отнести:
        </p>
        <ul>
            <li>множество Кантора;</li>
            <li>треугольник Серпинского;</li>
            <li>кривую Пеано.</li>
        </ul>
    </body>
</html>
```

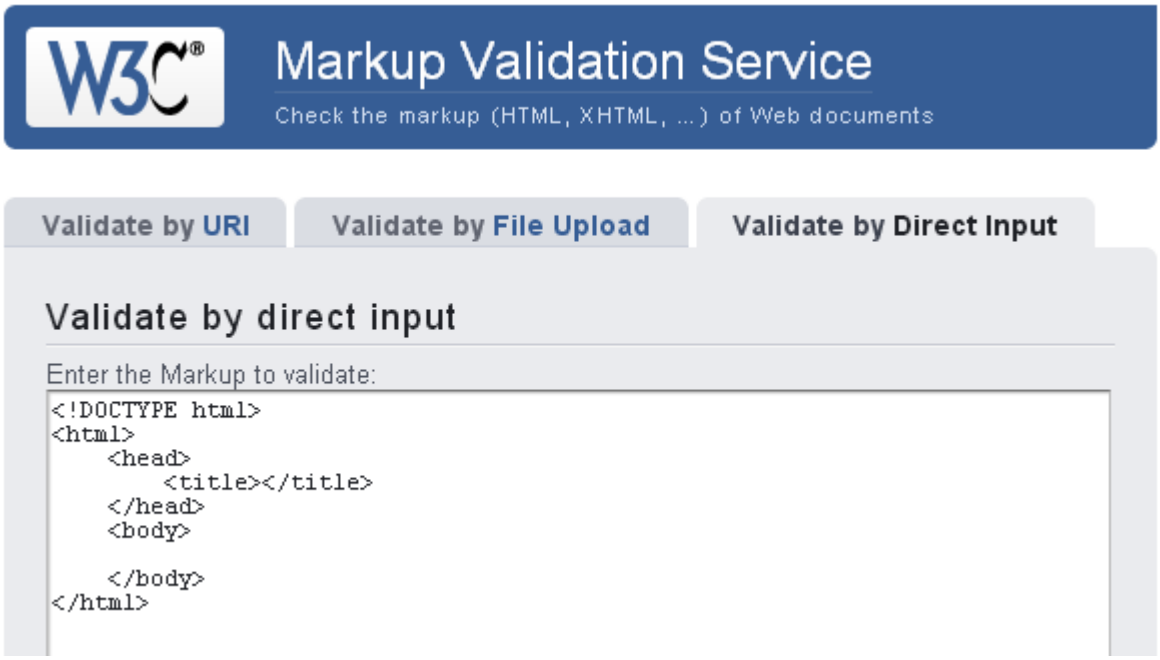


Валидаторы

До сих пор не был раскрыт вопрос о проверке написанного кода на корректность, соответствие стандарту спецификации, или по-другому *валидность*. Существуют специальные сервисы, позволяющие произвести валидацию кода и выдать предупреждения и ошибки, которые следует устранить.

Прежде всего, такой сервис предоставляет консорциум W3C:

https://validator.w3.org/#validate_by_input



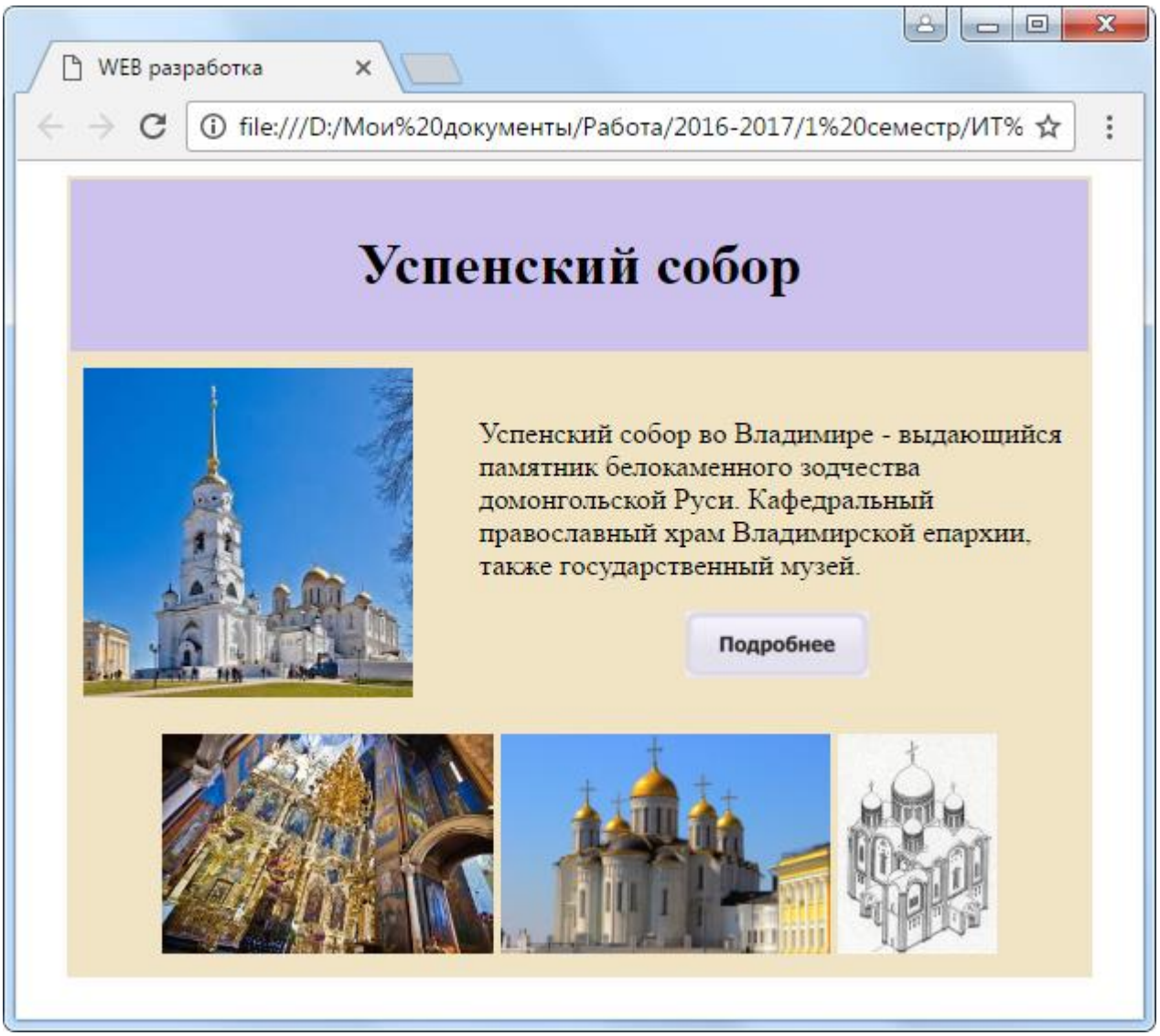
Задания для самостоятельной работы

1. Скопируйте седующий следующий макет:

Пример

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
    </head>

    <body>
        <table border="1" width="560" align="center"
            bgcolor="#F1E4C5" cellpadding="7">
            <tr bgcolor="#CCC2ED">
                <td colspan="2">Заголовок</td>
            </tr>
            <tr>
                <td width="200">Изображение</td>
                <td>Описание</td>
            </tr>
            <tr>
                <td colspan="2">Изображения</td>
            </tr>
        </table>
    </body>
</html>
```

2. Заполните ячейки макета соотвествующим содержимым и оформите его надлежащим образом (кнопка является изображением с ссылкой на статью в Wikipedia):



3. Проверить набранный в первом задании код любым online-валидатором. При необходимости исправить допущенные ошибки или недочеты.

1.9 Новые теги спецификации HTML5

Ранее отмечалось, что в HTML5 реализовано немало новых тегов, позволяющих повысить эффективность обработки сайта поисковыми системами, добавить элементы мультимедиа и вцелом улучшить логическую структурированность работы.

> *Разметка может быть построена и без указанных ниже тегов. Однако необходимо помнить, что они предоставляют важную информацию поисковым системам, повышая тем самым релевантностью сайта при индексировании.*

Ключевые элементы

К этой категории тегов можно отнести теги контейнеры <header>, <footer>, <aside>, и <nav>.

Теги header и footer

Предоставляют область верхнего и нижнего информационного блока. Обычно в <header> располагаются логотипы, основные разделы сайта, элементы управления, поиска, регистрации и т.п. Раздел <footer> (т.н. «подвал») часто содержит контактную информацию.

Тег aside

Контейнер удобно использовать для оформления блока меню, любой другой боковой панели («сайдбара») либо поясняющего раздела.

Тег nav

Отводится на область навигации по сайту или краткой ее реализации. Обычно в этом блоке присутствуют самые важные ссылки на другие разделы сайта.

Секционирование

Часто содержимое сайта можно рассматривать как набор статей похожей структуры и оформления. Для таких целей удобно использовать теги <main>, <article> и <section>.

Тег main

Определяет область для основных данных, которые не должны дублироваться. Он может определять область контента, но используется только один раз.

Тег article

Определяет область вывода новости, статьи, поста в блоге и любого другого автономного блока информации. Может включать в себя такие разделы, как <header>, <footer> и <article>.

Тег section

Используется для секционирования определенных смысловых частей (пунктов, параграфов). Внутри первым тегом рекомендуется брать один из заголовочных.

В следующей таблице демонстрируется пример корректной по HTML5 стандарту разметки, а также ее аналог в блочной div-модели (классы и идентификаторы рассматриваются в следующих занятиях). Благодаря новым тегам код становится и более коротким:

Разметка с тегами HTML5	Альтернативная блочная разметка

`<header>`
`  Верхняя панель`
`  <nav>Ссылки</nav>`
`</header>`
	`<div id="header">`
	`  Верхняя панель`
	`  <div id="header-nav">Ссылки</div>`
	`</div>`

`<aside>Левая панель</aside>`	`<div id="left-aside">Левая панель</div>`
`<main>`	`<div id="main">`

```
    <article>                          <div class="article">
        <section>                          <div class="section">
            <h1>Тема 1</h1>                    <h1>Тема 1</h1>
            . . .                              . . .
        </section>                         </div>
        <section>                          <div class="section">
            <h1>Тема 2</h1>                    <h1>Тема 2</h1>
            . . .                              . . .
        </section>                         </div>
        . . . . . .                        . . . . . .
    </article>                         </div>
</main>                            </div>

<footer>                           <div id="footer">
    Нижняя панель                      Нижняя панель
</footer>                          </div>
```

Информационные теги

Дополнительную, но важную информацию желательно включать в теги <address> и <time>.

Тег address

Предназначение тега – включать информацию об авторе (проекта, статьи, блога).

```
<address>
    Разработчик:
    <a href="mailto:yakubovich_project@mail.ru">Якубович Д.А.</a>
</address>
```

Тег time

Указанный тег используется для указания даты (статьи, последних изменений и т.д.).

> *Перечисленные теги не предоставляют каких-либо дополнительных стилевых настроек. Свойства и позиционирование объектов задается через технологию CSS.*

Разметка других объектов

Тег figure

Подходит для разметки области таких элементов как изображения, схемы, таблицы. Обычно содержит в себе тег <figcaption>, приводящий пояснение:

```
<figure>
    <img src="pic.jpg">
    <figcaption>Рис. 1.</figcaption>
</figure>
```

Тег для графического и звукового воспроизведения

В HTML5 окончательно внедрена возможность вставки видео- и звукопроигрывателей.

Обеспечивают эту работу теги <video> и <audio>. Тег <sourse> используется указанными тегами для указания источника для воспроизведения.

Например:

```
<video width="400" height="320" controls="controls">
    <source src="video/My_cat.ogv"
        type='video/ogg; codecs="theora, vorbis"'>
    <source src="video/My_cat.mp4"
        type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
</video>
<audio controls>
    <source src="audio/M_Jackson.mp3" type="audio/mpeg">
    Тег audio не поддерживается вашим браузером.
</audio>
```

Задания для самостоятельной работы

1. Найдите в сети Интернет более подробное описание новых тегов, а также рекомендуемую область их применения. Законспектируйте материал и покажите преподавателю.

Раздел 2 Технология CSS3

2.1 Каскадные таблицы стилей Технология CSS3

Понятие CSS

> ***Стилем*** *или* ***CSS (Cascading Style Sheets, каскадные таблицы стилей)*** *называется набор параметров форматирования, который применяется к элементам документа для изменения их внешнего вида.*

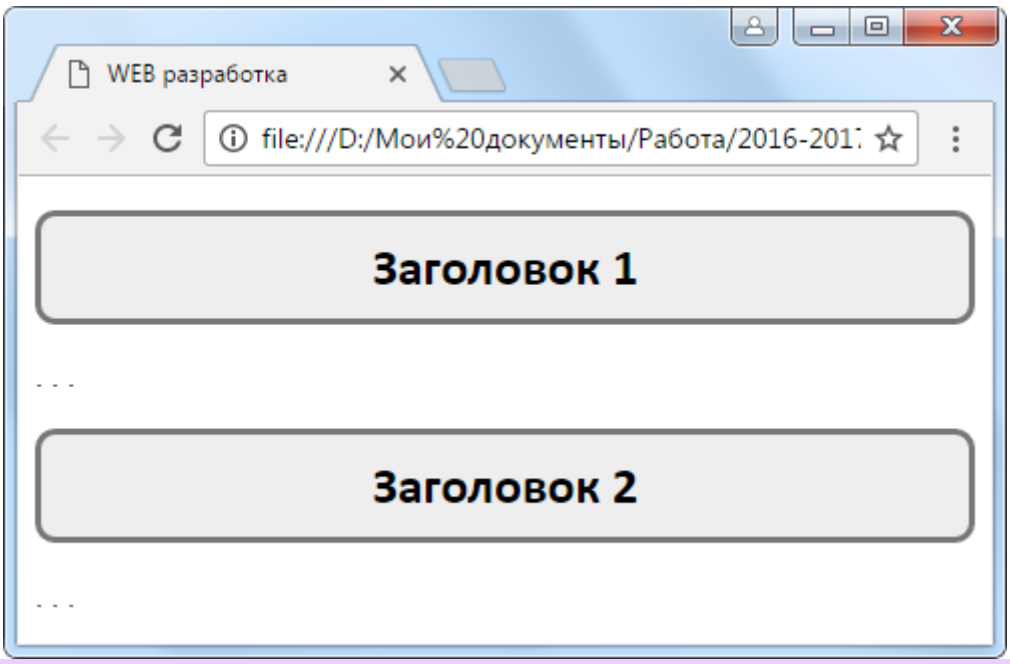
Как известно, HTML является языком разметки электронного документа. Основная его задача – определить логическую структуру документа. Однако если встает вопрос оформления страницы, то HTML не может предложить богатого выбора средств. Но теперь от него это и не требуется.

Серьезное преимущество каскадных таблиц стилей – гораздо более широкий спектр возможностей форматирования, чего не может позволить HTML.

Обратите внимание на следующий код и результат:

Пример

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>WEB разработка</title>
        <style>
            h1 {
                background: #EEE;
                font: bold 160% Calibri;
                text-align: center;
                padding: 10px;
                border: 3px solid #777;
                border-radius: 10px;
            }
        </style>
    </head>

    <body>
        <h1>Заголовок 1</h1>
        . . .
        <h1>Заголовок 2</h1>
        . . .
    </body>
</html>
```



Согласно разметке у тегов <h1> отсутствуют какие-либо атрибуты, а всё оформление сосредоточено внутри нового тега <style>. Так, в нашем примере необходимые свойства тегу <h1> задаются следующим кодом:

```
h1 {
    background: #EEE;
    font: bold 160% Calibri;
    text-align: center;
    padding: 10px;
    border: 3px solid #777;
    border-radius: 10px;
}
```

Преимущество стилей

- Разграничение кода и оформления. В идеале HTML должен определять только логическую структуру документа, а вид и настройки элементов задаются через CSS-стили.
- Разное оформление для разных устройств. С помощью стилей можно определить вид веб-страницы для разных устройств вывода: монитора, принтера, смартфона, планшета и др. При этом без изменений HTML-разметки.
- Расширенные по сравнению с HTML способы оформления элементов.
- Ускорение загрузки сайта.
- Единое стилевое оформление множества документов.
- Хранение стилей в отдельных файлах.

Внедрение CSS естественным образом разграничивает вэб-разработку:

- *HTML задает разметку документа;*
- *CSS оформляет элементы.*

Последнее замечание очень важно. С приходом CSS использование атрибутов тегов, отвечающих за визуальное оформление (цвет, шрифт, фон и т.д.), сведено к нулю.

CSS3

Главная особенность CSS3 – возможность работы с эффектами анимации без использования языка JavaScript, работа с градиентами, тенями, эффектами сглаживания и многое другое (то, чего не было в ранних версиях).

Технология CSS3 – отдельная технология, напрямую не связанная с HTML5, однако сейчас рекомендуется их совместное использование.

Несмотря на то, что проблемы совместимости технологий HTML5 и CSS3 с современными браузерами постепенно сводятся на нет, стандартный браузер Internet Explorer может не поддерживать многие новые свойства или теги.
Подробнее информацию по решению проблем с браузерами IE можно найти на сайте консорциума W3C.

Основы синтаксиса CSS

Если в HTML ключевым элементом являлся тег, то в CSS таким элементом является *селектор*.

***Селектор** – это некоторое имя стиля, для которого добавляются параметры форматирования. В качестве селектора могут выступают теги, атрибуты тегов, классы и идентификаторы.*

Теги-селекторы позволяют определять оформление для тегов; селекторы атрибутов способны задавать стиль тега в зависимости от значения его атрибутов или их наличия у тега. Классы и идентификаторы повышают гибкость использования стилей и фактически делают стиль независимым от конкретного тега.

Общий способ записи селектора (на примере тега <h1>) имеет следующий вид:

CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции, поэтому форма записи зависит от вкуса разработчика.

Хорошим считается стиль, когда каждое свойство указывается на отдельной строке:

```
h1 {
    background: #EEE;
    font: bold 160% Calibri;
    text-align: center;
    padding: 10px;
    border: 3px solid #777;
    border-radius: 10px;
}
```

В случае одного свойства часто пишут в одну строчку:

```
body { background: #F5E2D0; }
```

> *Алгоритм использования CSS стилей очень прост и однотипен. Названия свойств и описание их работы читатель может найти в справочнике текущего учебного пособия либо в сети Интернет.*
> *Дальнейшие вопросы будут связаны с развертыванием механизмов работы каскадных таблиц стилей совместно с HTML. Поэтому для изучения примеров и реализации заданий пользуйтесь справочником пособия.*

Комментарии

Для комментирования части CSS-кода используют скобки /* */. Такой комментарий является многострочным.

Например,

```
/*
    Стилевой шаблон: Article-style
    Дата послед. кор.: от 28.05.16
    Редактор: Якубович Д.А.
*/
h1 {
    background: #EEE;          /* цвет фона */
    font: bold 160% Calibri;   /* гарнитура шрифта */
```

```
        text-align: center;          /* выравнивание */
        padding: 10px;               /* внутренний оступ */
        border: 3px solid #777;      /* граница */
        border-radius: 10px;         /* радиус скругления углов */
    }
```

Виды стилей

Можно выделить следующие виды стилей, определяющие способ их подключения к основному документу:

- встроенные стили;
- внутренние стили;
- связанные стили.

Каждый из них имеет определенные преимущества в плане гибкости использования и эффективностью загрузки страницы.

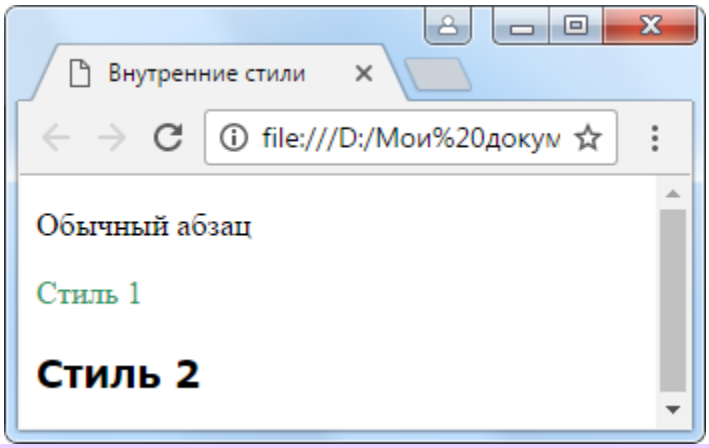
Встроенные стили

Встроенный стиль расширяет возможности одного тега, но работает только внутри него. Для его применения тегам добавлен глобальный атрибут style. В нем перечисляются требуемые свойства, фигурные скобки не указываются.

Пример

Встроенные стили.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Внутренние стили</title>
    </head>

    <body>
        <p>Обычный абзац</p>
        <p style="font-size: 12pt; color: #339966">Стиль 1</p>
        <p style="font: bold 14pt Verdana">Стиль 2</p>
    </body>
</html>
```



Такой способ отличается малой гибкостью («одноразовостью»), а также приводит к визуальному нагромождению и дублированию кода.

> *Настоятельно рекомендуется отказаться от практики использования встроенных стилей. Они лишь приведены в ознакомительных целях.*

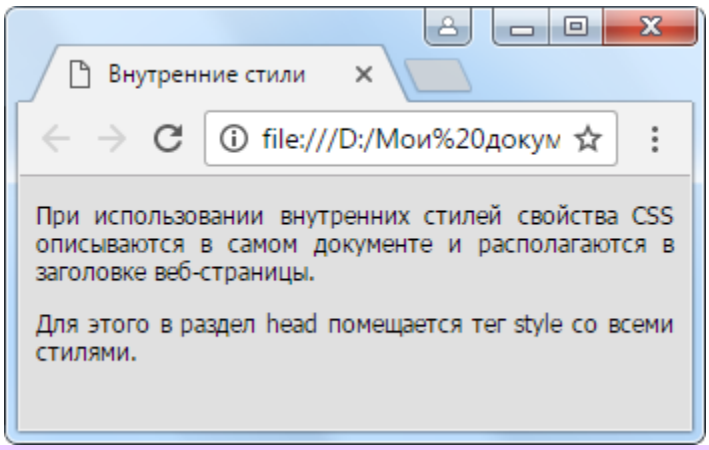
Внутренний стиль

При использовании внутренних стилей свойства CSS описываются в самом документе и располагаются в заголовке веб-страницы. Для этого в раздел <head> помещается тег <style></style> со всеми стилями. Отметим, что указанный парный тег может неоднократно использоваться, причем даже в разделе <body>.

Пример

Внутренние стили меняют свойства для тегов-селекторов <body> и <p>.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Внутренние стили</title>
        <style>
            body {
                background: #E0E0E0;
                font: 12px Tahoma;
            }

            p { text-align: justify; }
        </style>
    </head>

    <body>
        <p>
            При использовании внутренних стилей свойства CSS
            описываются в самом документе и располагаются в
            заголовке веб-страницы.
        </p>
        <p>
            Для этого в раздел head помещается тег style со
            всеми стилями.
        </p>
    </body>
</html>
```



Связанные стили

При использовании связанных стилей описание селекторов и их значений располагается в отдельном файле с расширением .css, а для связывания документа с этим файлом применяется тег <link>. Данный тег помещается в контейнер <head></head>.

> *В .css-файле могут присутствовать только стили. HTML код в нем запрещен!*

Тег	Атрибуты	Описание
`<link>`		Определяет отношение между страницей и другими документами.
	`href`	Основной атрибут тега, в качестве значения выступает путь к файлу со стилями.
	`rel`	Отношение к подключаемому документу. При подключении CSS-таблицы указывается значение «stylesheet».
	`type`	Определяет MIME-тип документа, на который идет ссылка. В данном случае он принимает значение «text/css». В HTML5 можно не указывать.

Пример

Связанный стиль, описанный в отдельном файле.

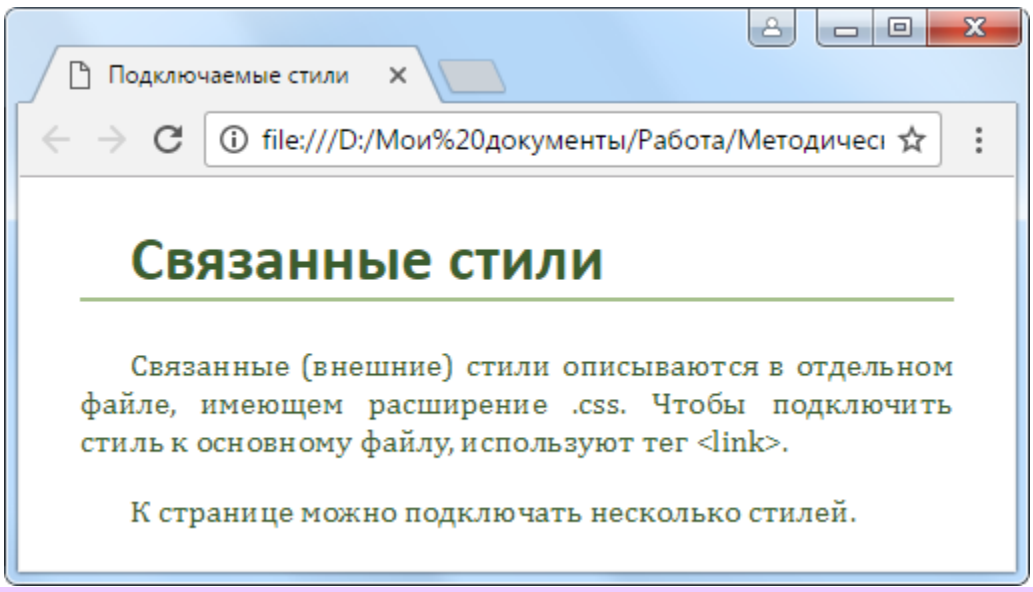
1) В начале описываем основной документ:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Подключаемые стили</title>
        <link rel="stylesheet" type="text/css" href="MyStyles.css">
```

```
    </head>

    <body>
        <h1>Связанные стили</h1>
        <p>
            Связанные (внешние) стили описываются в отдельном
            файле, имеющем расширение .css. Чтобы подключить
            стиль к основному файлу, используют тег &lt;link&gt;.
        </p>
        <p>
            К странице можно подключать несколько стилей.
        </p>
    </body>
</html>
```

2) Затем создаем файл MyStyles.css:

```
body {
    font: 16px Cambria;
    color: #3C5D2E;
    margin: 0 30px;
}

h1 {
    font-family: Calibri;
    padding-left: 25px;
    border-bottom: 2px solid #A7C18E;
}

p {
    text-align: justify;
    text-indent: 25px;
}
```



Значение атрибута тега <link> – rel остаётся неизменным независимо от кода, как приведено в данном примере. Значение href задаёт путь к CSS-файлу, он может быть задан как относительно, так и абсолютно. Заметьте, что таким

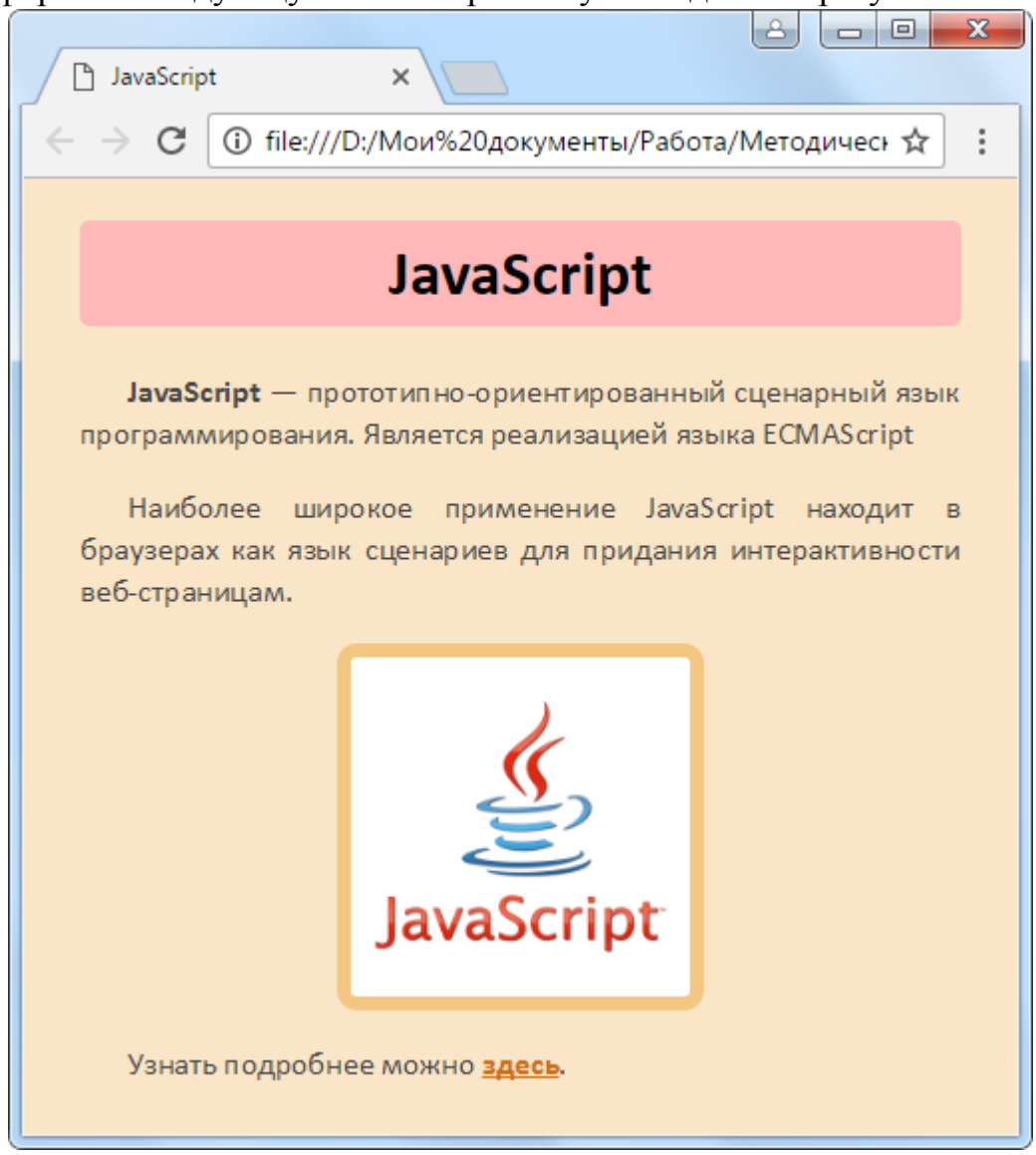
образом можно подключать таблицу стилей, которая находится на другом сайте.

Все типы стилей можно комбинировать в одном документе.

> *Связанные стили – наиболее мощный и эффективный способ подключения стилей, поскольку они позволяют достаточно быстро поменять оформление сайта без вмешательства в основной документ. Не менее важно то, что стилевую таблицу (и не одну) могут раздельно использовать разные страницы сайта.*

Задания для самостоятельной работы

1. С помощью CSS задать стили для тегов <body>,<h1>, <p>, <img>, <a> и оформить следующую ниже страничку. Ожидаемый результат:



2. Реализовать первое задание, используя:
   - внутренние стили;

- внешние стили.

2.2 Классы и идентификаторы

Единицы измерения

HTML достаточно сильно ограничивает возможность выбора числовых значений. Однако в CSS вы можете определять размер шрифта, границ (рамок), полей и отступов, используя различные единицы измерений.

Единица измерений	Описание	Тип
px	Пиксели	Относительная
pt	Пункты	Абсолютная
in	Дюймы	Абсолютная
cm	Сантиметры	Абсолютная
mm	Миллиметры	Абсолютная
pc	Пика	Относительная
em	Em	Относительная
ex	Ex	Относительная
%	Проценты	Относительная

Пиксели (px)

Наиболее часто используется в веб-дизайне. Пиксель не является абсолютной величиной: его размер определяется размером и разрешением экрана пользователя.

Типографский пункт (pt)

Пункты должны использоваться для печати. Один дюйм равен 72 пунктам.

Дюймы (in), сантиметры (cm), миллиметры (mm)

Несмотря на то, что это одни из самых распространенных единиц измерений, в веб-дизайне абсолютные значения лучше не использовать.

Пики (pc)

Пика (Picas) – единица измерений, которая используется для печати. Один дюйм равен 6 пикам.

Em

Em, или Мутт – это относительная единица измерений, определяющая размер буквы «M» в шрифте. Часто используется в веб-дизайне.

Ex

Ex, или «x-высота», определяет высоту строчной «x» шрифта. Ex применяется для установки размера контента в зависимости от размера окружающего шрифта.

%

Относительная величина, задающая размер пропорционально размеру родительского элемента. 100% соответствует искомому размеру.

При выборе единицы измерения стоит обратить внимание на тот факт, что абсолютные величины могут давать несколько отличный результат на разных типах устройств вывода. Относительные же не зависят от устройств вывода и сохраняют пропорции, поэтому рекомендуются верстальщиками в качестве основных.

Классы

Основная идея применения стилей CSS – разграничение логической структуры страницы и стиля ее оформления. Для создания индивидуальных стилей и повышения гибкости разметочного кода используются *классы*.

Классы, привязанные к тегу

Синтаксис класса:

```
Тег.Имя класса {
    свойство1: значение;
    свойство2: значение;
    ...
}
```

Внутри стиля вначале пишется желаемый тег-селектор, а затем через точку пользовательское имя класса. Имена классов должны начинаться с латинского символа и могут содержать в себе символ дефиса и подчеркивания. Использование кириллицы в именах классов недопустимо.

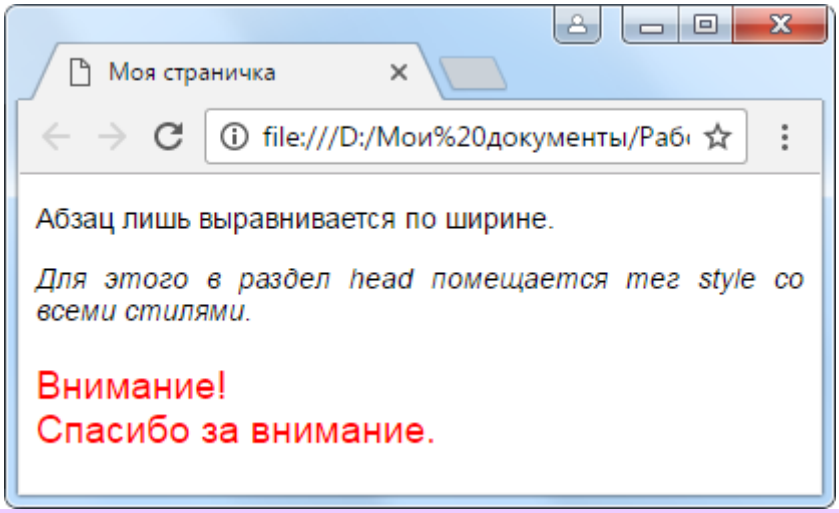
Чтобы свойства вступили в силу, тег должен применяться с указанным классом. Для этого тегам доступен глобальный атрибут class, в котором указывается имя класса.

Пример

Классы тега.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
        <style>
            p {
                font: 14px Arial;
                text-align: justify;
            }

            p.Quote { font-style: italic; }
```

```
            p.Warning {
                color: #F00;
                font-size: 120%;
            }
        </style>
    </head>

    <body>
        <p>
            Абзац лишь выравнивается по ширине.
        </p>
        <p class="Quote">
            Для этого в раздел head помещается тег style со
            всеми стилями.
        </p>
        <p class="Warning">
            Внимание! <br> Спасибо за внимание.
        </p>
    </body>
</html>
```



> *Помните, что классы наследуют свойства своего тега.*
> *Если у тега и его класса описано одинаковое свойство, то в качестве значения берется то, что указано в классе.*

Автономные классы

Класс необязательно привязывать к тегу. Его можно описывать как самостоятельный стиль:

Синтаксис автономного класса:

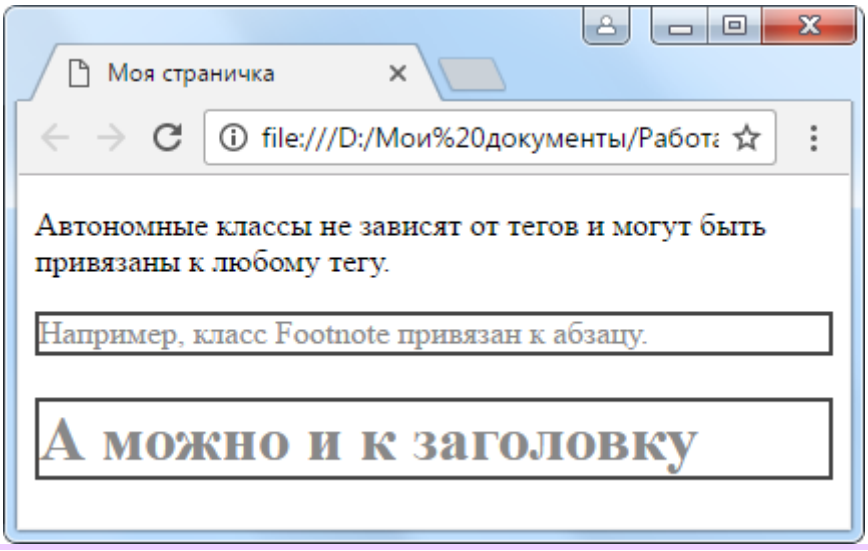
```
.Имя_класса {
    свойство1: значение;
    свойство2: значение;
    ...
}
```

Автономный класс можно привязать к любому тегу (если это имеет смысл).

Пример

Классы тега.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
        <style>
            .Footnote {
                color: #888;
                border: 2px solid #444;
            }
        </style>
    </head>

    <body>
        <p>
            Автономные классы не зависят от тегов и могут
            быть привязаны к любому тегу.
        </p>
        <p class="Footnote">
            Например, класс Footnote привязан к абзацу.
        </p>
        <h1 class="Footnote">А можно и к заголовку</h1>
    </body>
</html>
```



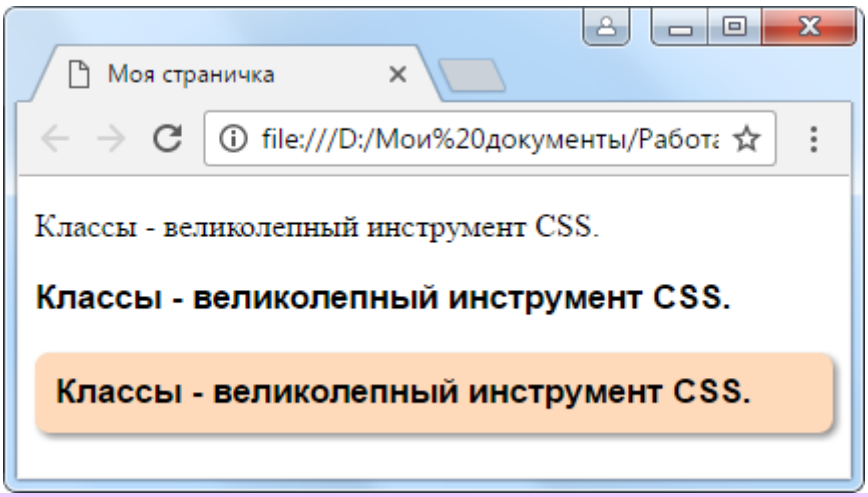
Использование нескольких классов

К любому тегу можно добавить несколько стилей, перечислив их в атрибуте class через пробел. Если при этом в разных стилях имелись одинаковые свойства, то используется значение последнего.

Пример

Классы тега.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
        <style>
            p { text-align: justify; }

            .Definition { font: bold 105% Arial; }

            .Block {
                background: #FED9BA;
                border-radius: 7px;
                box-shadow: 2px 2px 5px #888;
                padding: 10px;
            }
        </style>
    </head>

    <body>
        <p>
            Классы - великолепный инструмент CSS.
        </p>
        <p class="Definition">
            Классы - великолепный инструмент CSS.
        </p>
        <p class="Definition Block">
            Классы - великолепный инструмент CSS.
        </p>
    </body>
</html>
```



Идентификаторы

Еще один важный элемент в CSS, похожий на класс – идентификатор.

Синтаксис идентификатора:

```
#Имя_идентификатора {
    свойство1: значение;
    свойство2: значение;
    ...
}
```

Идентификатор также может быть привязан к тегу, либо быть автономным. Чтобы применить идентификатор, тегам доступен глобальный атрибут id.

От классов идентификатор отличает следующее:

- Идентификатор можно использовать только один раз.
- Идентификаторы имеют более высокий приоритет (специфичность) стиля.

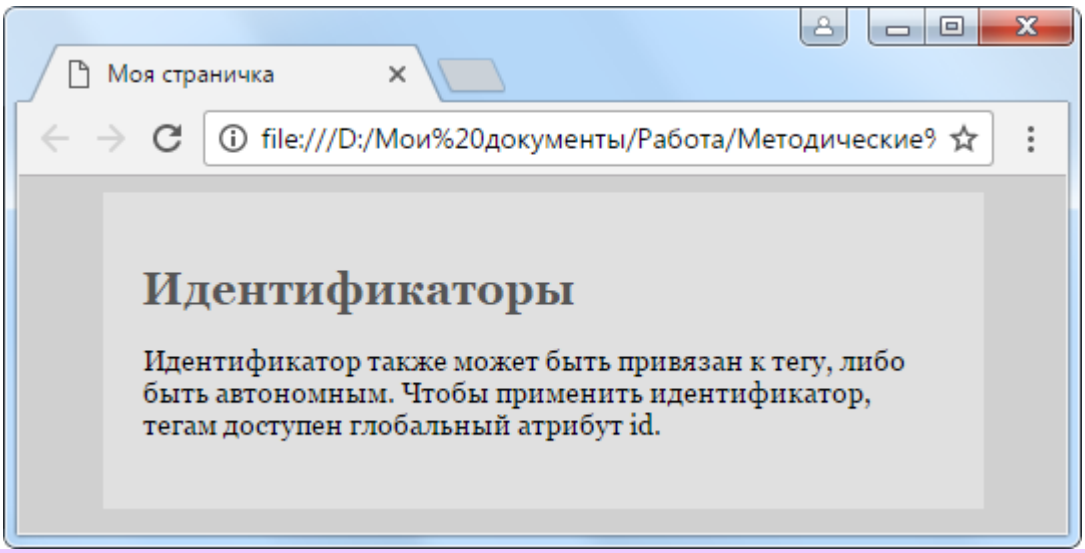
Идентификаторы следует использовать для элементов, уникальных в своем роде: блока меню, верхней и нижней панели, области контента и т.п.

Пример

Идентификаторы тега.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Моя страничка</title>
        <meta charset="utf-8">
        <style>
            body {
                background: #D0D0D0;
                font: 14px Georgia, serif;
            }

            #Content {
```

```
            background: #E0E0E0;
            width: 400px;
            margin: 0 auto;     /* т.о. центрует блок */
            padding: 20px;
        }

        h1 {
            font-size: 160%;
            color: #555;
        }
    </style>
</head>

<body>
    <div id="Content">
        <h1>Идентификаторы</h1>
        <p>
            Идентификатор также может быть привязан к тегу,
            либо быть автономным. Чтобы применить
            идентификатор, тегам доступен глобальный
            атрибут id.
        </p>
    </div>
</body>
</html>
```



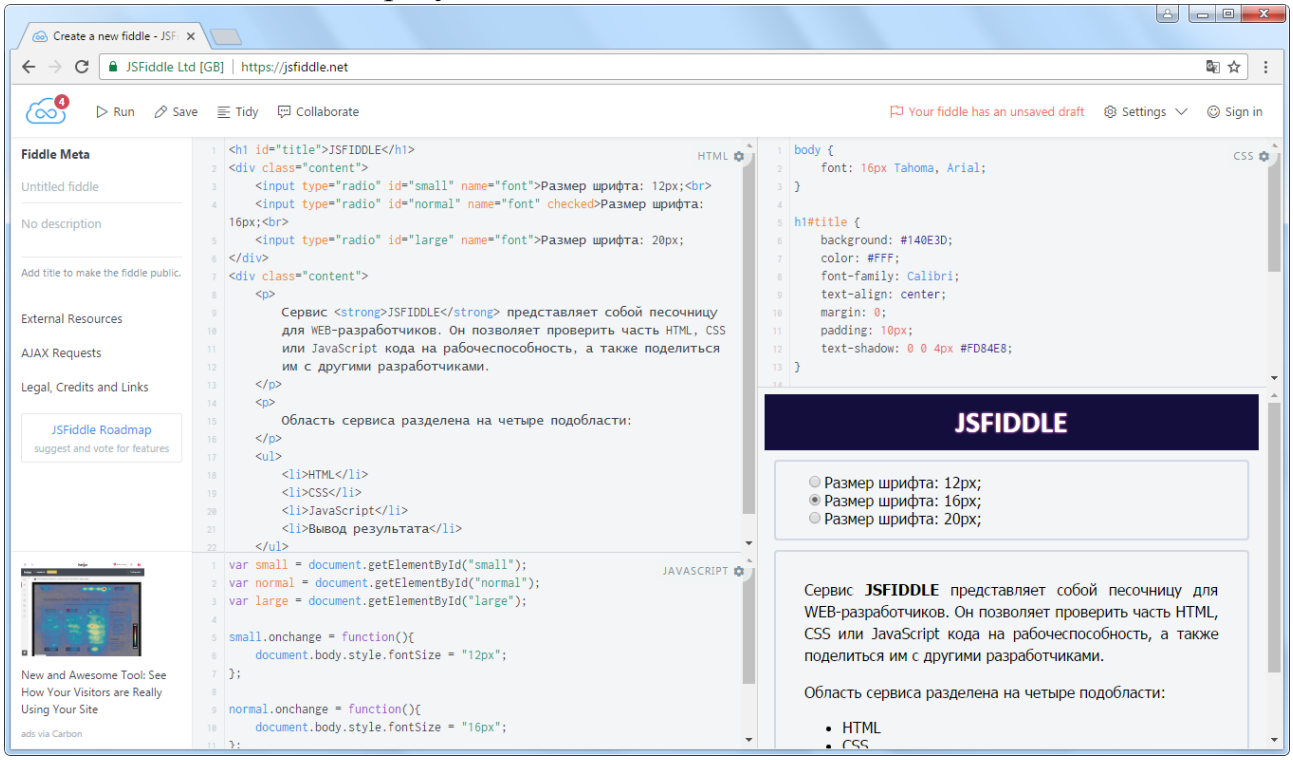
В примере используется тег , выполняющий роль контейнера. О нем поговорим подробнее в следующем занятии.

Онлайн проекты для тестирования возможностей

В сети Интернет можно найти интересные проекты, помогающие просмотреть результат работы части HTML и CSS кода., а также обсудить код с другими разработчиками.

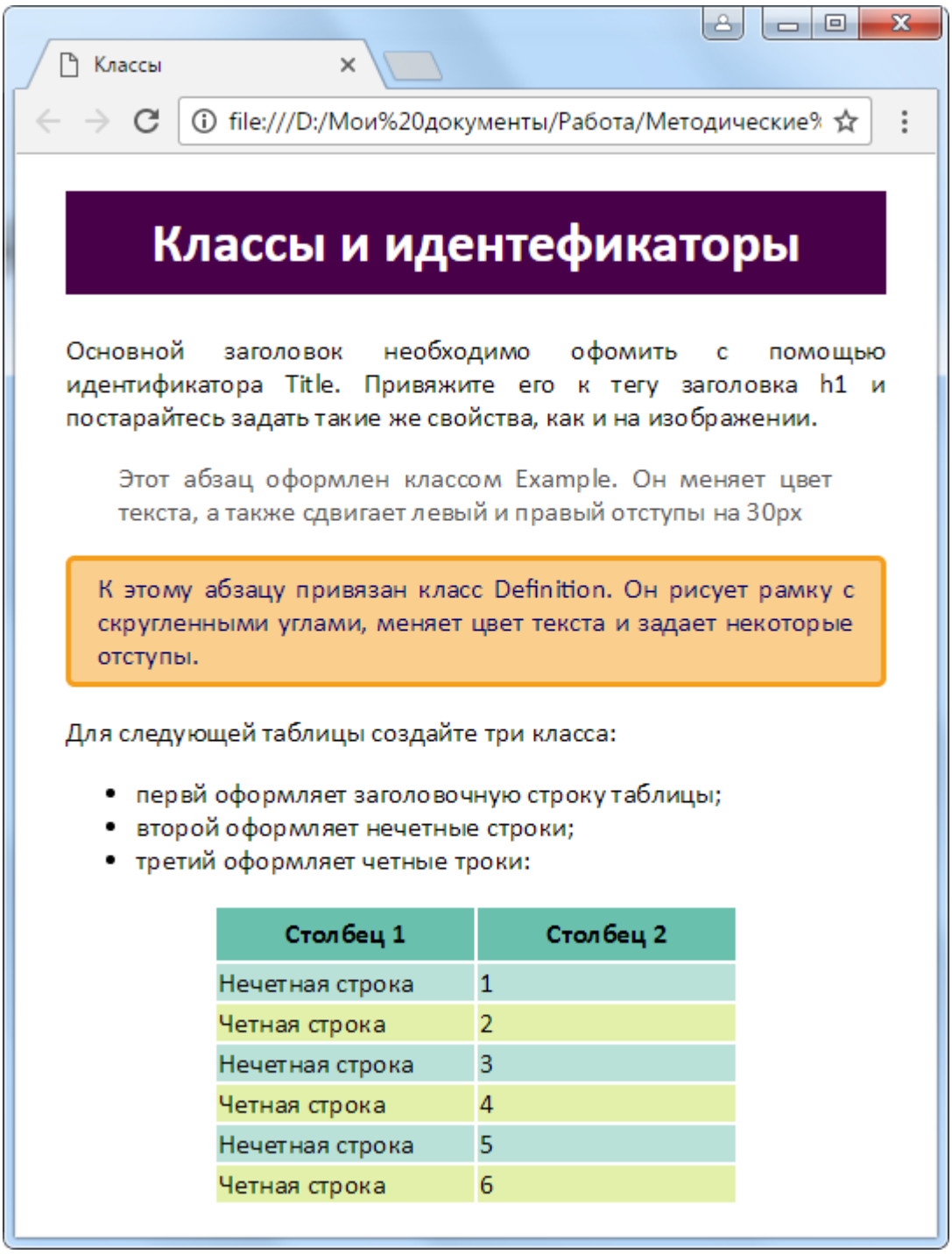
Достаточно удобным в этом плане является проект JSFiddle. Здесь предоставлены четыре окна:

- окно HTML кода;
- окно CSS кода;
- окно JavaScript кода;
- окно вывода результата.



Задания для самостоятельной работы

1. Используя стили для селекторов-тегов <body>, <h1>, <p>, а также создав указанные в контексте задания классы, оформите следующий документ:

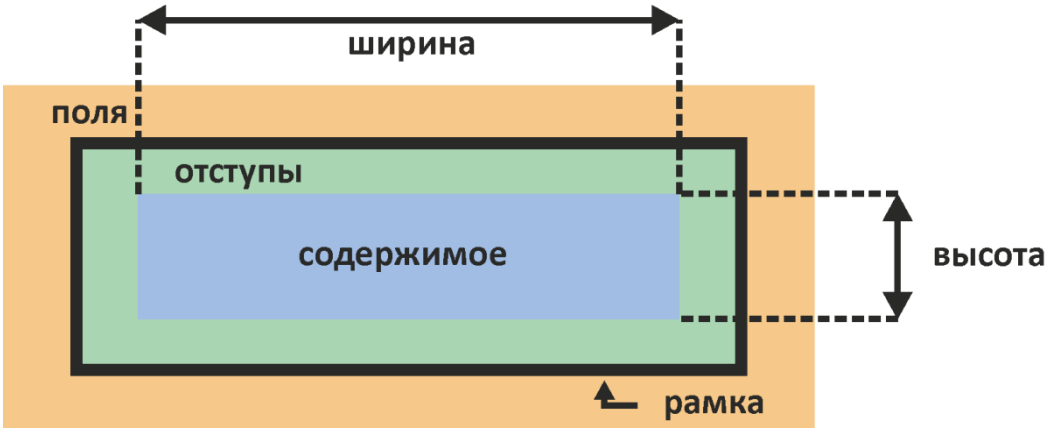


2.3 Блоки

Блочная структура

Каждый объект разметки генерирует прямоугольный блок, называемый **контейнером элемента**. Контейнер определяют следующие параметры:

- содержимое;
- отступ (padding);
- рамка (border);
- поле (margin).



В явной форме каждый параметр может быть задан определенным значением, либо равен нулю. Последние три характеристики задаются соответственно свойствами padding, border и margin или их вариациями для каждой из четырех сторон (см. справочник). Ширина и высота объекта определяется свойствами width и height соответственно[3].

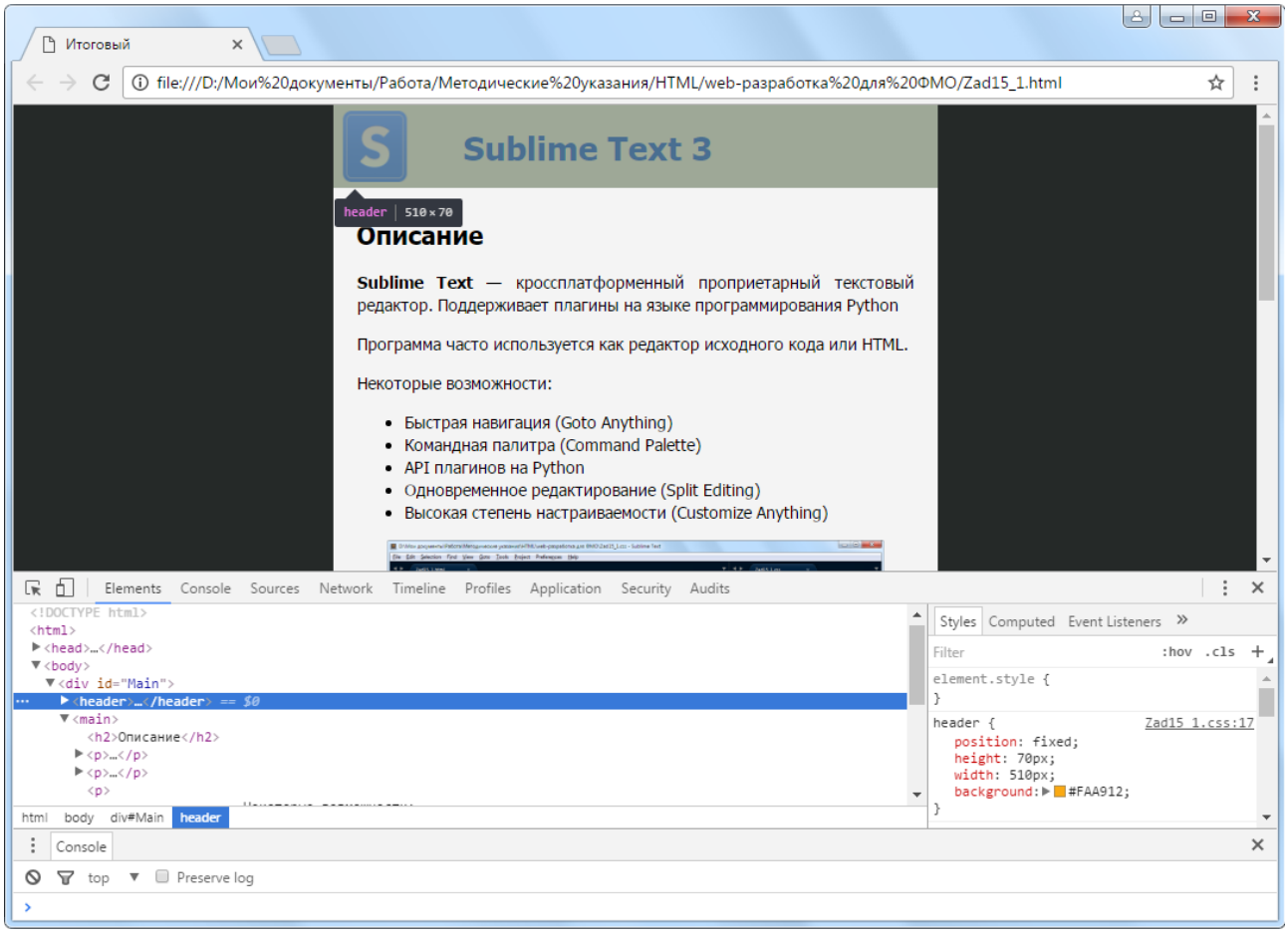
Элементы можно разделить на две основные категории:

Блочные	Строковые
Занимают всю доступную ширину окна или родительского элемента. Высота определяется согласно содержимому, отступам, рамке и полям. Примеры: , <p>, <h1-h6> , и т.д.	Фактическая ширина зависит от содержимого, отступов, рамок и полей. Аналогично с высотой. Примеры: , <b>, <s>, <u>, <i>, <sub> и т.д.

Чтобы лучше понять принцип работы, а также научиться контролировать общие размеры объекта в процессе разработки, можно воспользоваться *инспектором кода*, доступному в современных браузерах.

Например, в браузерах Google Chrome инспектор кода вызывается так: правая кнопка мыши / Просмотреть код:

[3] Важно отметить, что ширина и высота берется именно по содержимому, без учета отступов и рамок.



Особенности браузеров

Если изучить несколько веб-страничек с помощью инспектора кода, то можно заметить, что у тегов <h1>, <p>, <ul> есть дополнительные поля и / или отступы. Причем даже в том случае, когда мы их не задавали.

Это объясняется особенностью браузеров, которые по умолчанию способны самостоятельно задавать элементам некоторые свойства. Красноречиво о «чужих» свойствах говорит надпись:

```
user agent stylesheet
```

где агент – это и есть браузер.

Разумеется, если мы сами укажем свойства в разметке, то стандартные значения сбрасываются.

> *Если необходимо просто сбросить стандартные значения отступов и полей селектора, укажите следующее:*
>
> ```
> margin: 0;
> padding: 0;
> ```

Теги div и span

В HTML есть два парных тега-контейнера, по умолчанию отвечающие за блочные и строковые элементы. Обычно их применяют в случае, когда по ряду причин другие теги применять логически неоправданно.

Тег представляет блочный элемент. В него можно поместить как строковые, так и блочные элементы, в частности, другие блоки . С приходом CSS тег является наиболее важным и часто используемым в разметке.

Тег является строчечным элементом. Обычно он отвечает за форматирование части текста.

> *Теги и не приводят к каким-либо изменениям содержимого сами по себе. Необходимые настройки задаются в классах и идентификаторах, которые подключаются к этим тегам.*

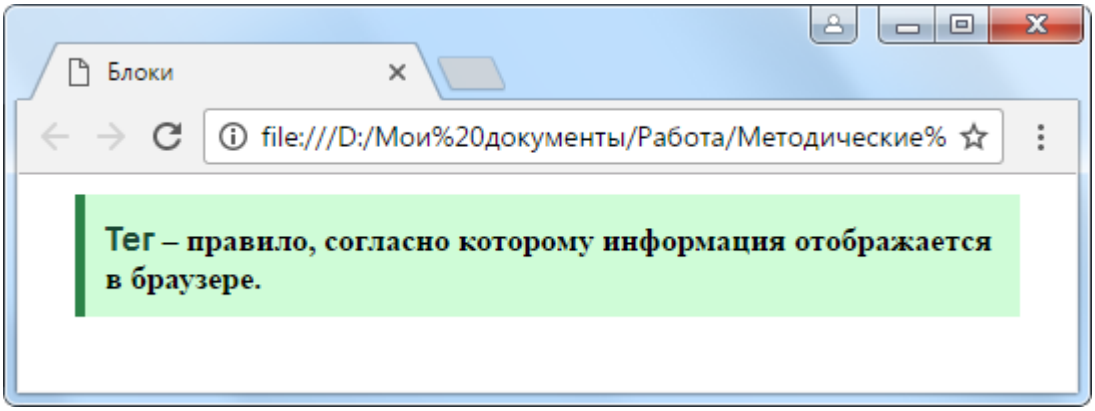
Использование тегов div и span.

HTML

```
<div class="Definition">
  <span class="Termin">Тег</span>
    – правило, согласно которому
    информация отображается в
    браузере.
</div>
```

CSS

```
div.Definition {
  background: #CFFCD7;
  font-weight: bold;
  border-left: 5px solid #2C8348;
  margin: 10px 20px;
  padding: 10px;
}

span.Termin {
  font: bold 120% Calibri, Arial;
  color: #174631;
}
```



Разумеется, классы Defin и Term можно сделать автономными.

Вложение блоков

Необходимо понять, что блочная разметка отличается от табличной и имеет множество специфических моментов.

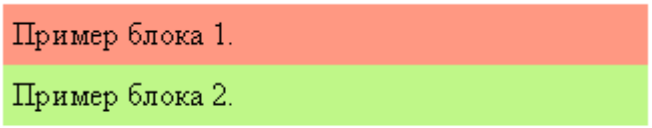
Ранее было отмечено, что блоки занимают всю доступную ширину.

HTML

```
<div class="BlockA">
  Пример блока 1.
</div>
<div class="BlockB">
  Пример блока 2.
</div>
```

CSS

```
/* отступ для двух блоков */
.BlockA, .BlockB { padding: 5px; }

.BlockA { background: #FF9882; }
.BlockB { background: #BFF788; }
```



Можно задать фиксированную ширину первому блоку, но второй блок все равно останется на своем уровне:

HTML

```
<div class="BlockA">
  Пример блока 1.
</div>
<div class="BlockB">
  Пример блока 2.
</div>
```

CSS

```
.BlockA, .BlockB { padding: 5px; }

.BlockA {
  background: #FF9882;
  width: 250px;
}

.BlockB { background: #BFF788; }
```

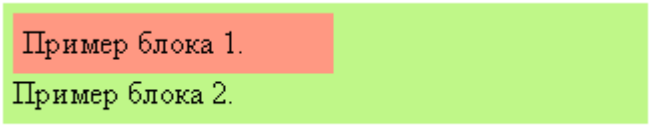
Вставим первый блок во второй. При этом второй блок сохранит отступ в 5px для содержимого (т.е. для первого блока):

HTML

```
<div class="BlockB">
  <div class="BlockA">
    Пример блока 1.
  </div>
  Пример блока 2.
</div>
```

CSS

```
.BlockA, .BlockB { padding: 5px; }

.BlockA {
  background: #FF9882;
  width: 250px;
}

.BlockB { background: #BFF788; }
```



А теперь зададим конкретное значение высоты для блоков:

HTML

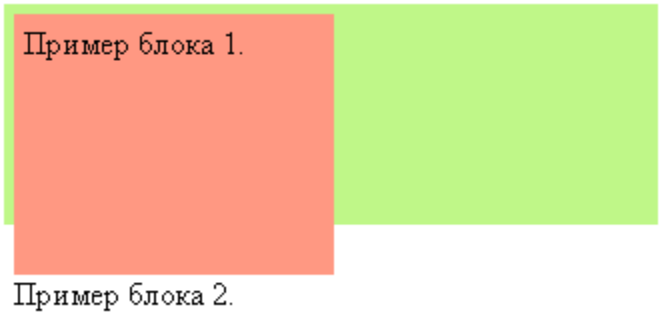
```
<div class="BlockB">
  <div class="BlockA">
    Пример блока 1.
```

CSS

```
.BlockA, .BlockB { padding: 5px; }

.BlockA {
```

```
  </div>
  Пример блока 2.
</div>
```

```
  background: #FF9882;
  width: 150px;
  height: 120px;
}

.BlockB {
  background: #BFF788;
  height: 100px;
}
```



Пример блока 2.

Для читателя результат может быть неожиданным, однако по заданным свойствам он весьма ожидаемый: первый блок выше, и он фактически вытолкнул содержимое второго.

Таким образом, при вложении блоков следует заранее определить, как содержимое может повлиять на результат. Хорошей практикой является:

- не фиксировать высоту родительского блока (она будет подбираться согласно высоте контента);
- фиксировать высоту родительского блока при условии, что высота контента заранее просчитана.

Свойства margin и padding

Указанные свойства являются краткой формой в своих семействах свойств и позволяют задать значения сразу для четырех сторон объекта. Каждое из них может иметь одно, два, три или четыре значения.

Рассмотрим пример использования свойства margin; для padding правила аналогичны.

Четыре значения

Задают параметры в следующем порядке: верх, право, низ, лево:

```
margin: 10px 20px 40px 20px;
```

Вышеприведенный код в полной форме свойства margin:

```
margin-top: 10px;
margin-right: 20px;
margin-bottom: 40px;
margin-left: 20px;
```

Одно значение

В этом случае значение задается равным со всех сторон:

```
margin: 10px;  /* поле в 10px сверху, справа, снизу и слева */
margin: 10px 10px 10px 10px;  /* то же самое */
```

Два значения

Когда присутствуют два числа, то они задают значения сверху и справа, а низ и левая сторона их копируют:

```
/* поле в 10px сверху/снизу и 20px справа/слева */
margin: 10px 20px;
margin: 10px 20px 10px 20px; /* то же самое */
```

Три значения

Задают верх, правую сторону и низ; а левая часть копирует значение у правой:

```
margin: 10px 20px 30px;  /* сверху 10px, справа/слева – 20px, */
                         /* снизу – 30px */
margin: 10px 20px 30px 20px;  /* то же самое */
```

Значение auto

Кроме того, каждое свойство может принимать значение auto, которое определяет ширину или высоту согласно содержимому. Но чаще всего auto используется для выравнивания объекта по горизонтали:

```
/* отступ сверху и снизу 0, слева и справа - auto */
margin: 0 auto;
/* отступ сверху 10px, снизу 20px, слева и справа - auto */
margin: 10px auto 20px;
```

Свойство border

Свойство border также является укороченной командой для установки границ. Обычно ему указывают три параметра:

```
border: толщина тип_линии цвет;
```

однако минимально допускается указать только тип линии.

Задания для самостоятельной работы

1. С помощью инспектора кода исследуйте практическое задание предыдущего занятия. Обратите внимание на отступы, границы и поля объектов в схеме.
2. Измените локально некоторые размеры отступов и цвета. Сохраните страницу из окна браузере к себе в каталог.

2.4 Вложенные и дочерние селекторы Группировка и каскадирование

Группировка

При описании селекторов часто приходится неоднократно указывать одинаковые свойства. Чтобы предотвратить дублирование кода, CSS предоставляет возможность группировки общих свойств. Для этого достаточно перечислить через запятую селекторы и описать для них стиль. А свойства, принимающие разные значения, указать ниже отдельно.

Например, следующий код

до группировки

```
h1 {
    text-align: center;
    font-weight: bold;
    font-size: 200%;
}

h2 {
    text-align: center;
    font-weight: bold;
    font-size: 170%;
}

h3 {
    text-align: center;
    font-weight: bold;
    font-size: 145%;
    color: #AAA;
}
```

после группировки

```
h1, h2, h3 {
    text-align: center;
    font-weight: bold;
}

h1 { font-size: 200%; }

h2 { font-size: 180%; }

h3 {
    font-size: 145%;
    color: #AAA;
}
```

Контекстные селекторы

Мощный инструмент CSS, которым следует овладеть – *контекстные* (вложенные) селекторы. Он позволяет задать указанные свойства только тем селекторам, которые вложены в указанный (родительский). Конечно же, в качестве селекторов могут выступать теги, классы, идентификаторы и их комбинации.

Чтобы показать зависимость, дочерний элемент ставится через пробел.

Например, следующий код

```
p a {
    text-decoration: none;
    color: #0080C0;

}
```

установит стиль только тем ссылкам, которые находятся внутри тега абзаца <p>. При этом тег ссылки может быть вложен и в другие теги; изменения его

также коснутся. Однако если тег ссылки лежит вне p, то применяются стандартные свойства.

В следующим примере демонстрируется более сложное вложение:

```
div#Menu ul li {
    font: bold 120% Calibri;
    padding: 4px;
}
```

Это означает, что указанные свойства применяются только для тех элементов списка (<li>), которые находятся внутри маркированного списка (<ul>), который в свою очередь расположен в контейнере , описанном с идентификатором #Menu, т.е. свойства заработают, например в таком случае:

```
<div id="Menu">
    <ul>
        <li>Пункт 1</li>
        <li> . . . </li>
    </ul>
</div>
```

> *Глубина вложения контекстных селекторов не важна.*

Дочерние селекторы

Дочерние селекторы работают по принципу, похожему на работу контекстных селекторов. Но есть одно важное отличие: свойства распространяются только на прямых потомков первого уровня; более глубокие вложения уже «неподвластны» изменениям.

Для указания зависимости используется знак «>».

Например,

```
p > strong { color: #F00; }
```

задает цвет только тому тексту внутри , который непосредственно является дочерним для тега p. Так, в случае

```
<p>
    <strong>Дочерние селекторы</strong>.
    <br>
    <b><strong>Дочерние селекторы</strong></b>
</p>
```

изменениям подвергается только первая строка

> *Дочерние селекторы работают по принципу «вассал моего вассала не мой вассал».*

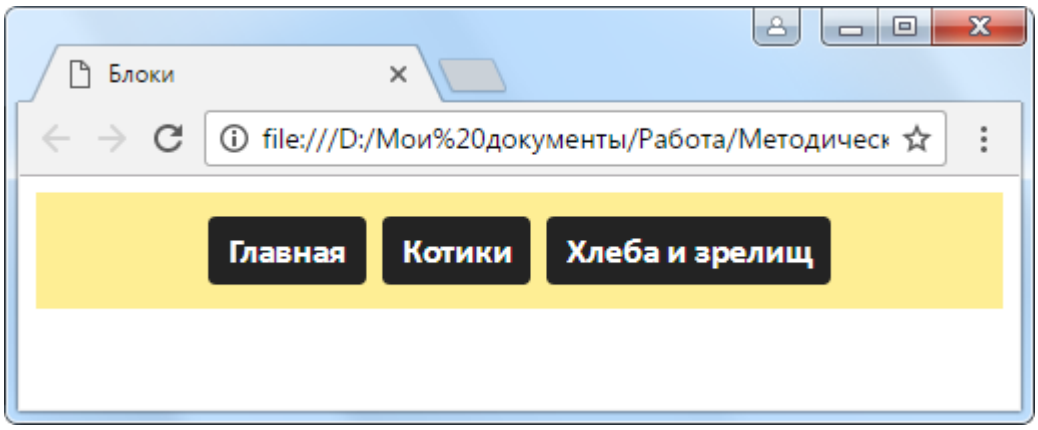
Дочерние селекторы удобно применять для элементов, которые обладают иерархической структурой (таблицы, списки и т.д.).

Пример

Контекстные и дочерние селекторы. Оформлена панель навигации <nav>,

невиди-мый блок-контейнер (класс Center-box), который центрует содержимое любой ширины, а также задан стиль для ссылок, работающий исключительно внутри блока с идентификатором Main-panel.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Блоки</title>
        <style>
            /* ограничить "плавающую" ширину страницы */
            body { min-width: 420px; }

            /* область навигации */
            #Main-panel {
                background: #FEEE94;
                padding: 18px 0;
            }

            /* класс центрует блок любой ширины */
            .Center-box {
                display: table;     /* ведет себя как таблица */
                margin: 0 auto;
            }

            /* оформление ссылки внутри главной панели ссылок */
            #Main-panel a {
                background: #232323;
                color: #FFF;
                text-decoration: none;
                border-radius: 4px;
                font: bold 110% Calibri;
                padding: 6px 10px;
                margin: 0 4px;
            }
        </style>
    </head>

    <body>

        <nav id="Main-panel">
            <div class="Center-box">
                <a href="pages/page1.html">Главная</a>
                <a href="pages/page2.html">Котики</a>
                <a href="pages/page3.html">Хлеба и зрелищ</a>
            </div>
        </nav>

    </body>
</html>
```



Заметим, что можно было написать:

```
nav#Main-panel {  }
```

и

```
nav#Main-panel a {  }
```

Однако тег-селектор nav указывать необязательно, поскольку у нас больше нет тегов с идентификаторами Main-panel.

Универсальный селектор

CSS предоставляет специальный селектор *, который соответствует любому элементу разметки. Свойства, указанные в нем, будут распространяться на все элементы, что не всегда желательно.

Однако есть задача, которая может оказаться полезной. Так вам известно, что браузеры по умолчанию могут устанавливать поля или отступы некоторым блочным элементам. Их можно сбросить, добавив стиль:

```
* {
    margin: 0;
    padding: 0;
}
```

Задания для самостоятельной работы

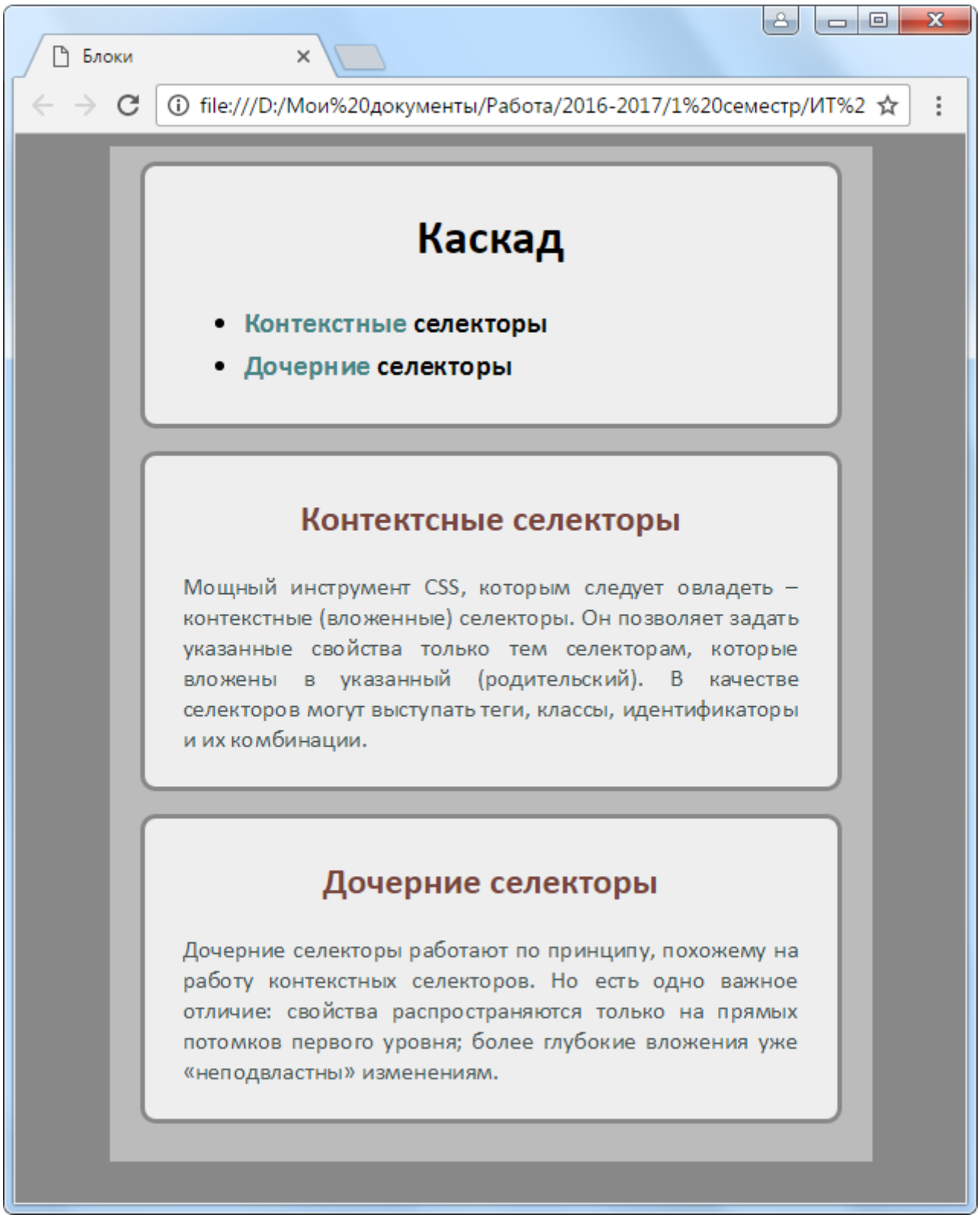
1. Пользуясь ранее полученными знаниями о блоках div, классах и идентификаторах, дочерних и контекстных селекторах, методах группировки общих свойств, скопируйте и изучите следующий код:

Пример

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Блоки</title>
        <style>
            body {
                background: #888;
                font: 16px Calibri, Arial;
```

```
            }

            /*  основной блок  */
            #Box {
                background: #BBB;
                width: 560px;
                padding: 10px 20px;
                margin: 0 auto;
            }

            /*  список в разделе меню  */
            ul#Menu {  }

            /*  ссылки в разделе меню  */
            ul#Menu > li > a {  }

            /*  блоки разделов  */
            .Container {  }

            /*  основные элементы разметки текста  */
            h1, h2 {  }

            p {  }
        </style>
    </head>

    <body>

        <div id="Box">
            <div class="Container">
                <h1>Каскад</h1>
                <ul id="Menu">
                    <li><a href="#">Контекстные</a> селекторы</li>
                    <li><a href="#">Дочерние</a> селекторы</li>
                </ul>
            </div>
            <div class="Container">
                <h2>Контектсные селекторы</h2>
            </div>
            <div class="Container">
                <h2>Дочерние селекторы</h2>
            </div>
        </div>

    </body>
</html>
```

2. Допишите свойства для соответсвующих селекторов, чтобы получить следующий результат:


Блоки
file:///D:/Мои%20документы/Работа/2016-2017/1%20семестр/ИТ%2
Каскад
Контекстные селекторы
Дочерние селекторы
Контектсные селекторы
Мощный инструмент CSS, которым следует овладеть – контекстные (вложенные) селекторы. Он позволяет задать указанные свойства только тем селекторам, которые вложены в указанный (родительский). В качестве селекторов могут выступать теги, классы, идентификаторы и их комбинации.
Дочерние селекторы
Дочерние селекторы работают по принципу, похожему на работу контекстных селекторов. Но есть одно важное отличие: свойства распространяются только на прямых потомков первого уровня; более глубокие вложения уже «неподвластны» изменениям.

2.5 Псевдоклассы Плавающие элементы

Песевдоклассы

Псевдоклассы определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа. С помощью псевдоклассов можно добавить дополнительные эффекты или упростить некоторые рутинные операции.

Синтаксис псевдокласса:

```
селектор:псевдокласс { свойства }
```

:hover
Определяет стиль элемента при наведении на него курсора.

:link
Применяется для ссылок, которые еще не посещались. Необходимо учитывать, что браузер сохраняет информацию о переходах, поэтому в дальнейшем ссылка помечается как посещенная.

:visited
Задает свойства уже посещенным ссылкам.

:active
Устанавливает стиль активированной ссылке.

При работе с ссылками псевдоклассы рекомендуется описывать в следующем порядке:

```
a:link { }
a:visited { }
a:hover { }
a:active { }
```

Оформление ссылок. Непосещенная и посещенная ссылки будут отображаться одним цветом. При наведении курсора ссылка становится ярче.

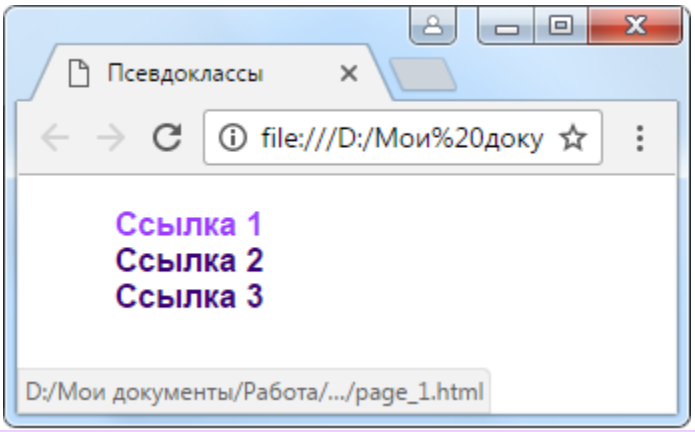
HTML

```
<ul id="Menu">
  <li><a href="page_1.html">
    Ссылка 1</a></li>
  <li><a href="page_2.html">
    Ссылка 2</a></li>
  <li><a href="page_3.html">
    Ссылка 3</a></li>
</ul>
```

CSS

```
#Menu { list-style: none; }

#Menu a {
  font: bold 12pt Arial;
  text-decoration: none;
}

#Menu a:link,
#Menu a:visited {
```

```
  color: #3A0075;
}

#Menu a:hover {
  color: #9D3CFF;
}
```



Свойство display

Ранее мы говорили о том, что формально элементы можно разделить на две категории: блочные и строковые. Однако далеко не всегда так требуется.

Например, для оформления красивого меню желательно, чтобы ссылкой выступал не только текст, но и окружающий ее блок. Однако ссылка по умолчанию относится к строковым элементам.

С помощью свойства display можно переопределить особенность элемента. Оно имеет много разных значений, мы же рассмотрим наиболее часто используемые:

Свойство	Значения и описание
display	block | inline | inline-block
	Определяет способ отображения элемента: • block (элемент отображается блоком и занимает доступную ширину); • inline (элемент становится строковым); • inline-block (элемент становится блоком, но другие элементы, в частности текст его уже будут обтекать).

Последнее значение может оказаться полезным, например, при оформлении ссылок в строку. Сами ссылки удобно поместить в список, которому задано свойство

```
display: inline-block;
```

Оформление меню. За основу взят список. Чтобы ссылкой стала вся область пункта, делаем ее блоком. Также продемонстрирован прием декоративного

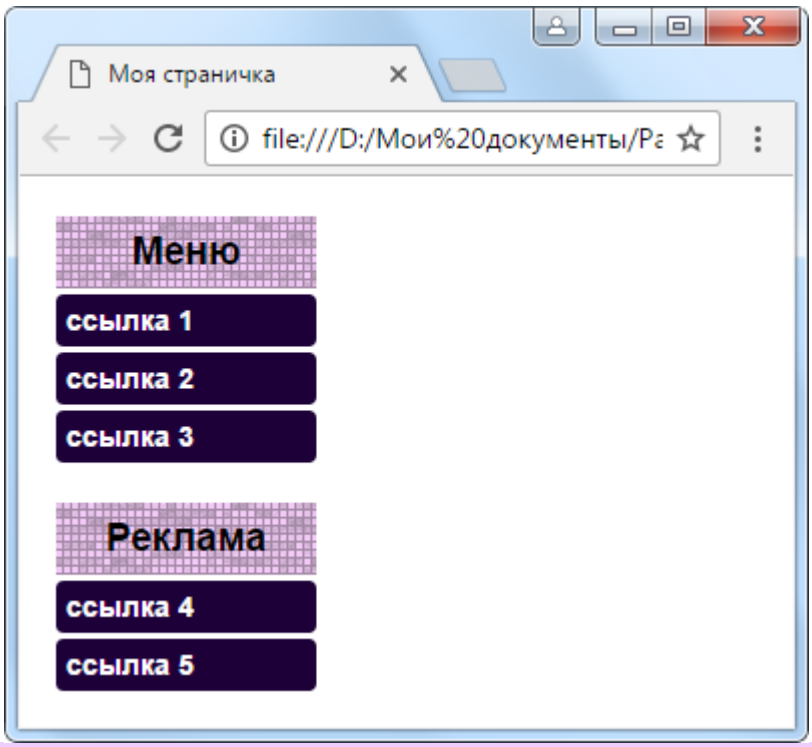
оформления заголовков меню с помощью полупрозрачного фонового изображения.

HTML

```
<ul id="Menu">
  <li class="Title">Меню</li>
  <li><a href="1.html">
    ссылка 1</a></li>
  <li><a href="2.html">
    ссылка 2</a></li>
  <li><a href="3.html">
    ссылка 3</a></li>

  <li class="Title">Реклама</li>
  <li><a href="4.html">
    ссылка 4</a></li>
  <li><a href="5.html">
    ссылка 5</a></li>
</ul>
```

CSS

```
/* общие свойства списка-меню */
ul#Menu {
    /* убрать маркеры пунктов */
    list-style-type: none;
    width: 130px;
    padding-left: 10px;
}

ul#Menu li { margin: 3px 0; }

/* оформление подзаголовка
   меню */
ul#Menu li.Title {
    background: #F3CAF7
url('fon.png');
    font: bold 120% Arial;
    text-align: center;
    padding: 7px 15px;
    margin-top: 20px;
}

/* оформление ссылок в виде
   кнопок */
ul#Menu li a {
    /* делаем ссылку блоком */
    display: block;
    background: #1C0037;
    font: bold 14px Arial;
    color: #FFF;
    border-radius: 4px;
    padding: 5px;
    text-decoration: none;
}
```



Плавающие элементы

CSS предоставляет разные подходы к позиционированию элементов. Но прежде всего обратим внимание на возможность создания плавающих элементов. Подобными свойствами, например, обладают изображения.

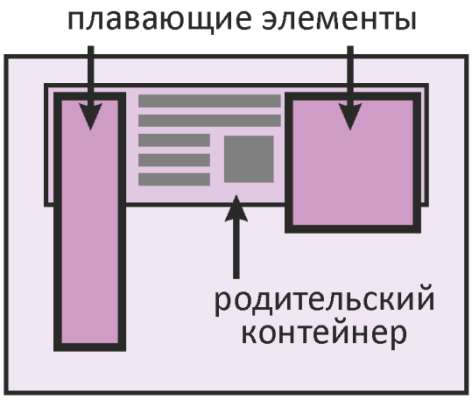
Соответствующую настройку осуществляет свойство float.

Свойство	Значения и описание
`float`	`left | right | none`
	Цепляет элемент к указанному краю родительского элемента. При этом другие элементы вынуждены его обтекать.

Если плавающие элементы идут подряд, то плотно цепляются уже друг к другу с указанной стороны. Однако если плавающий элемент не влезает по ширине, он вынужден сместиться вниз.

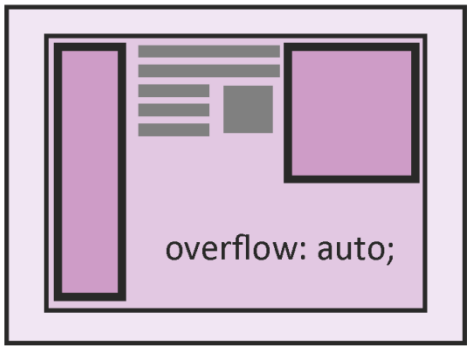
Плавающие элементы активно используют при верстке страницы.

Но у плавающих элементов есть одна особенность: их высота не влияет на высоту родительского блока. И может получиться так, что они вылезут за границу своего родителя:



Чтобы исключить подобное, воспользуйтесь свойством overflow в родительском блоке:

```
overflow: auto;
```



Запрет обтекания

Бывает необходимо запретить обтекание элементам, следующим после плавающего блока. Для этого можно воспользоваться свойством clear.

Свойство	Значения и описание
`clear`	`left | right | both`
	Запрещает обтекание слева, справа или с обоих сторон. Указывается объекту, который может быть подвергнут обтеканию.

Сконструировать шаблон следующего вида:

HTML

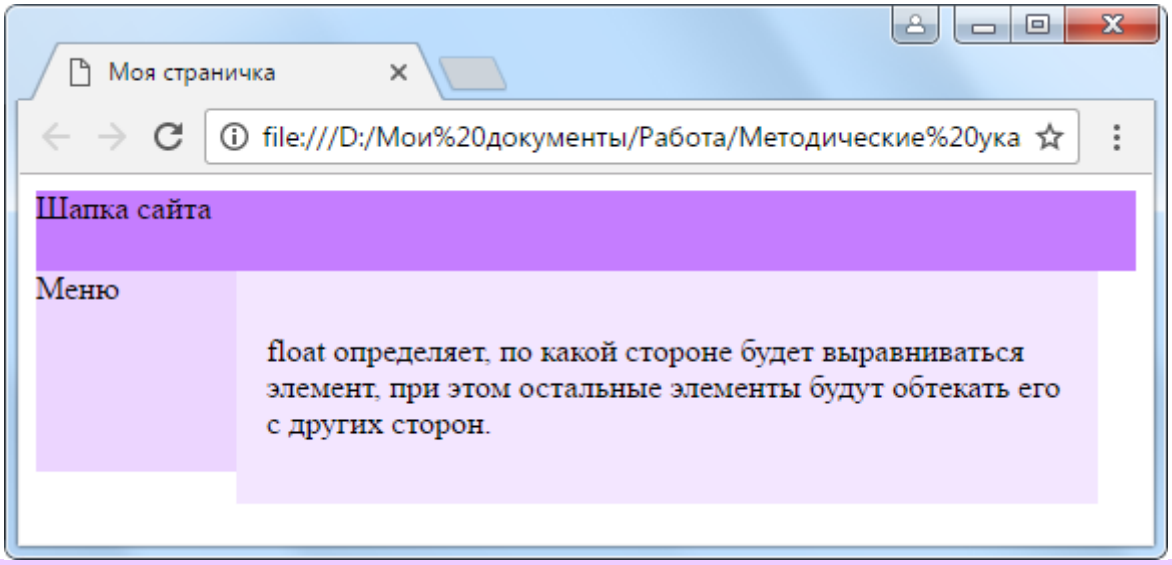
```
<header id="Head">
```

CSS

```
#Head {
```

```
  Шапка сайта
</header>
<div id="Menu">
  Меню
</div>
<div id="Content">
  <p>
    float определяет, по какой
    стороне будет выравниваться
    элемент, при этом остальные
    элементы будут обтекать его
    с других сторон.
  </p>
</div>
```

```
  background: #C57DFF;
  height: 40px;
}

#Menu {
  float: left;
  width: 100px;
  height: 100px;
  background: #ECD5FF;
}

#Content {
  background: #F3E6FF;
  float: left;
  width: 400px;
  padding: 15px;
}
```



В случае примера выше есть существенный недостаток: если сузить окно к блоку #Content, то оно не влезет по ширине и ему придется опуститься вниз (т.н. «выпадающий блок»). Решить такую проблему можно по разному:

- Поместить контейнеры #Menu и #Content в блок div фиксированной ширины. В нашем случае это можно сделать с точностью до пикселя, т.к. ширина обоих блоков известна.
- Задавать ширину блоков в процентах. Это т.н. «резиновая верстка». Однако в этом случае при сужении/растяжении окна по ширине результат может быть не очень приятным.
- Воспользоваться свойством box-sizing, способным поменять модель вычисления ширины блока.

Последний способ весьма интересен, ведь стандартный подход CSS – понимать под шириной блока только ширину контента. Часто же гораздо удобнее в ширину включать отступы и границы.

Свойство	Значения и описание
box-sizing	content-box | padding-box | border-box Задает способ вычисления ширины и высоты элемента. • content-box – только контент (по умолчанию); • padding-box – ширина контента + ширина отступов; • border-box – ширина контента + отступов + границ.

Значение border-sizing: border-box лучшим образом подходит для решения проблемы.

Обычно его задают универсальному селектору, чтобы распространить на все остальные элементы:

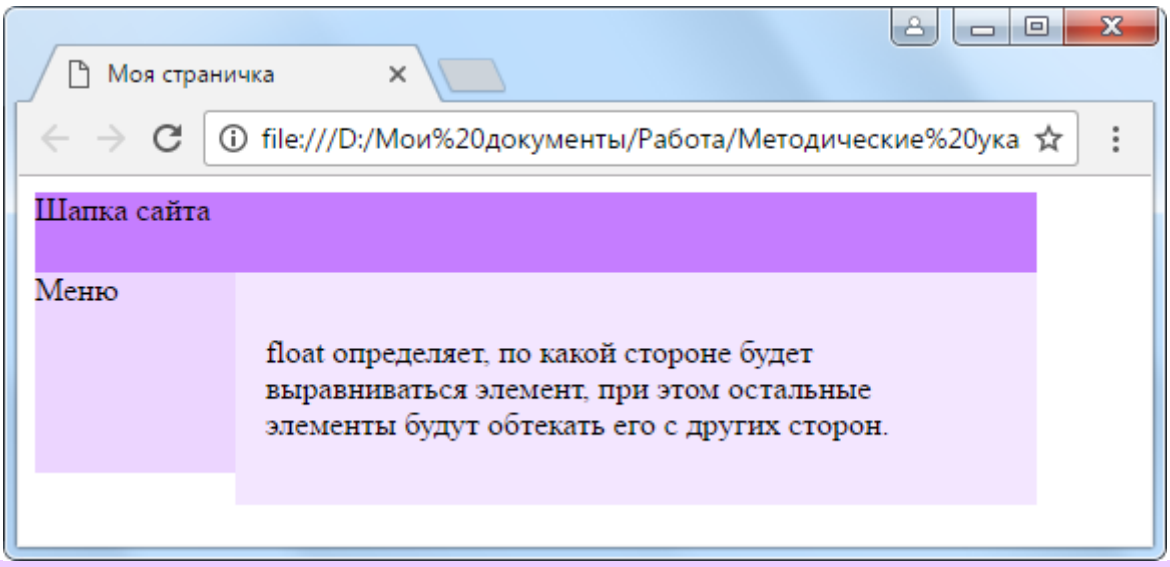
```
* { box-sizing: border-box; }
```

HTML

```
<div id="Main">

  <header id="Head">
    Шапка сайта
  </header>

  <div id="Menu">
    Меню
  </div>

  <div id="Content">
    <p>
      float определяет, по какой
      стороне будет выравниваться
      элемент, при этом остальные
      элементы будут обтекать его
      с других сторон.
    </p>
  </div>

</div>
```

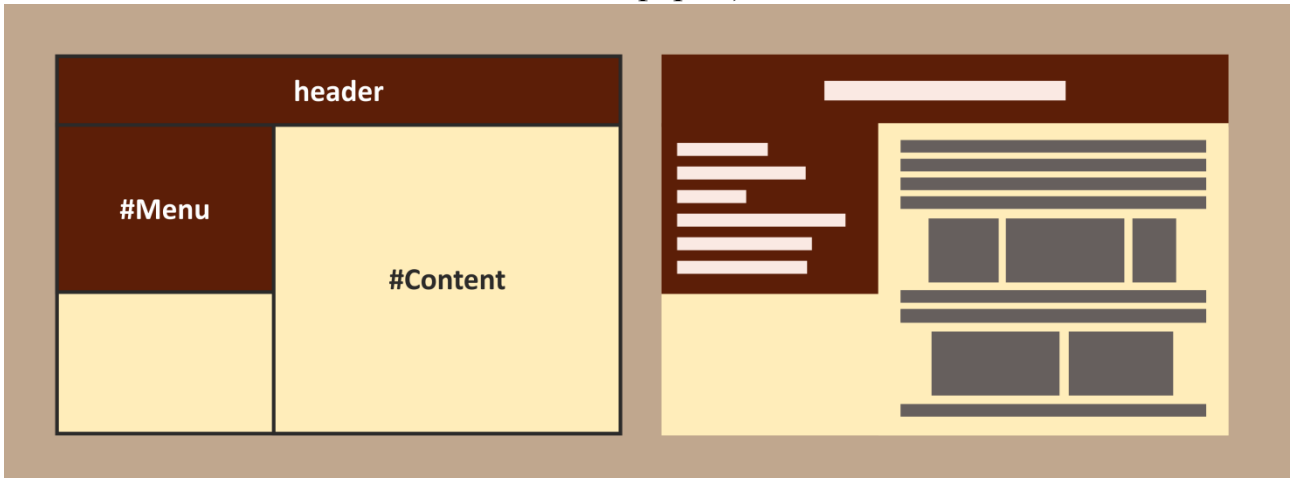
CSS

```
* {
  box-sizing: border-box;
}

/* блок фикс. ширины
для всего контента */
#Main { width: 500px; }

#Head {
  background: #C57DFF;
  height: 40px;
}

#Menu {
  float: left;
  width: 100px;
  height: 100px;
  background: #ECD5FF;
}

#Content {
  background: #F3E6FF;
  float: left;
  width: 400px;
  padding: 15px;
}
```



Задания для самостоятельной работы

1. Разработайте макет сайта следующего вида (на изображении представлен макет блоков и эскиз в конечной форме):



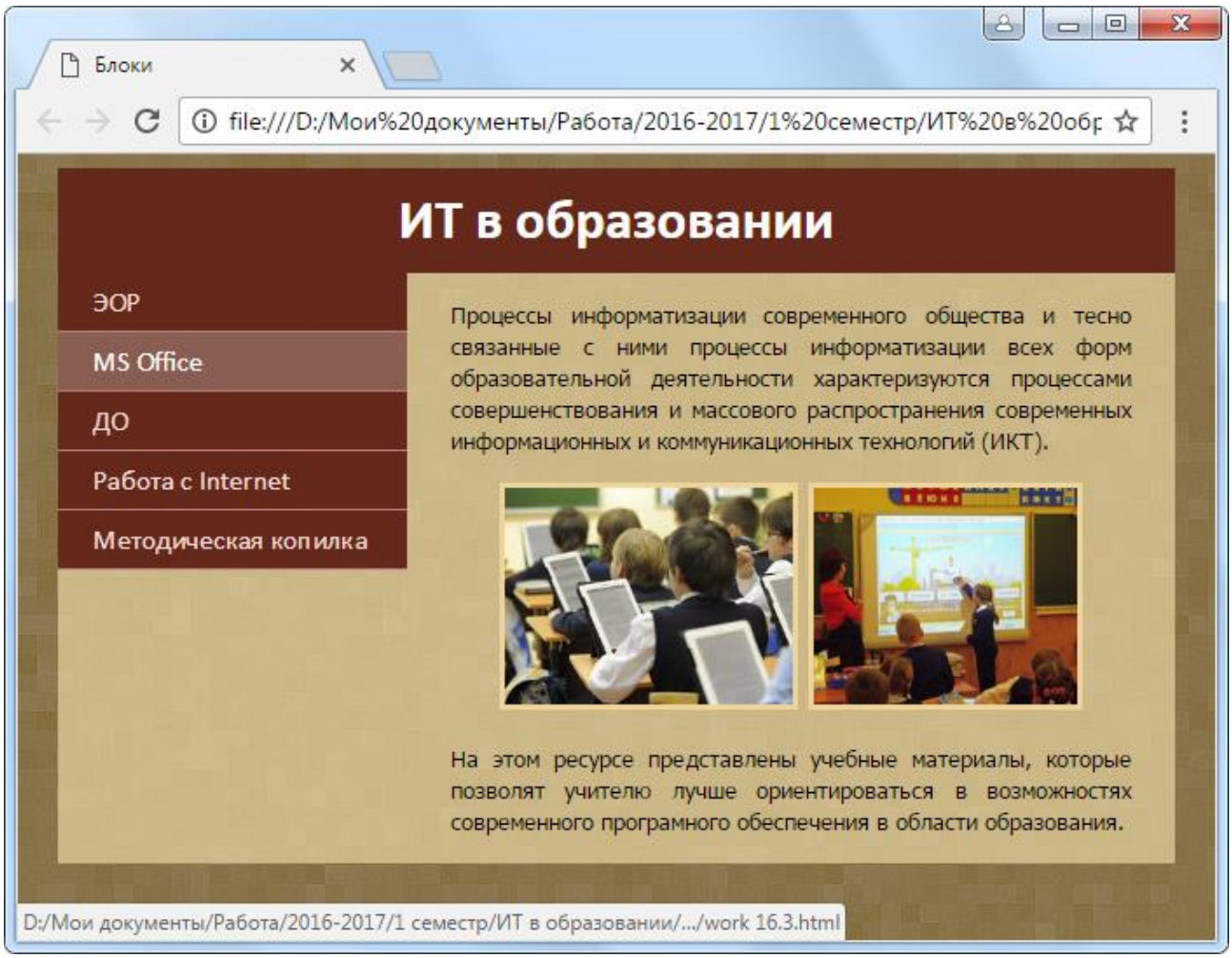
Для этого воспользуйтесь следующим кодом разметки:

Пример

```
<!DOCTYPE html>
<html>
    <head>
        <title>Блоки</title>
        <meta charset="utf-8">
        <style>
            * { box-sizing: border-box; }

            body {  }

            /* основной блок */
            #Main {
                width: 640px;
                background: rgba(251,231,179,0.6);
                margin: 0 auto;
```

```
            overflow: auto;
        }

        header {
            height: 60px;
        }

        #Menu, #Content { float: left; }

        #Menu { width: 200px; }

        /* пункты меню в форме стилизованных кнопок */
        #Menu a {
            display: block;

        }

        #Content {
            width: 440px;
        }
    </style>
</head>

<body>
    <div id="Main">
        <header>
            <h1 id="Title">Макет</h1>
        </header>

        <aside id="Menu">
            <a href="#">Раздел 1</a>
            <a href="#">Раздел 2</a>
            <a href="#">Раздел 3</a>
            <a href="#">Раздел 4</a>
            <a href="#">Раздел 5</a>
        </aside>

        <div id="Content">
            Здесь присутствует описание.
        </div>
    </div>
</body>
</html>
```

2. Допишите свойства селекторам и при необходимости добавьте новые, улучшающие отображение элементов. Ориентируйтесь на следующий результат:



Дополнительно сделайте так, чтобы при наведении на пункт ссылки в меню ее фон несколько осветлялся.

2.6 Позиционирование

Ранее мы убедились, что блоки выстраиваются последовательно в общем потоке следования. При вложении блоков заданной ширины можно получить определенный результат, но далеко не всегда этого достаточно. Частично проблему позиционирования помогло решить свойство display и модель плавающего блока, организованная свойством float.

Однако существуют другие способы позиционирования элементов на странице.

Свойство position

Свойство	Значения и описание
`position`	`absolute | fixed | relative | static`
	Определяет способ позиционирования элемента относительно окна браузера или родительского элемента.

Свойство position часто сопровождается следующей группой свойств:

Свойство	Значения и описание

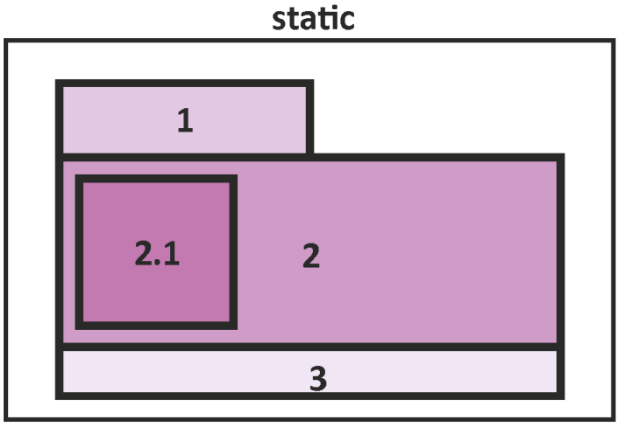
`left`
`right`
`top`
`bottom`
	`<значение>`

	Задают смещение элемента от указанной стороны

position: static

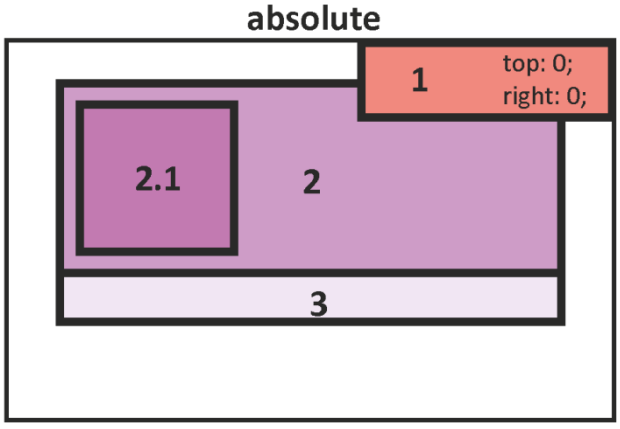
По умолчанию элементы располагаются статически, т.е. в том порядке, который и определен HTML-разметкой. В таком случае говорят, что элементы следуют в *нормальном потоке*. Поэтому указывать значение static вовсе необязательно.

Единственный случай практического использования static – вернуть позицию блока в нормальный поток (если она была изменена, например, в результате наследования).



position: absolute

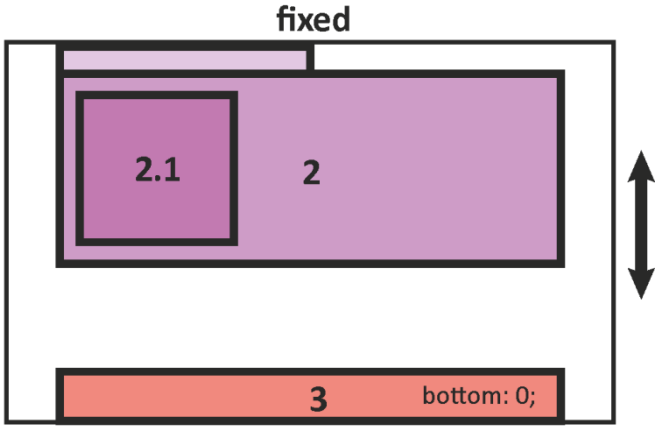
При абсолютном позиционировании элемент «выходит» из нормального потока, а другие блоки могут занять освободившееся место. Самому элементу можно задать определенное положение относительно окна или родительского элемента с помощью свойств left, right, top и bottom.



Абсолютное позиционирование чаще применяется в совокупности с относительным.

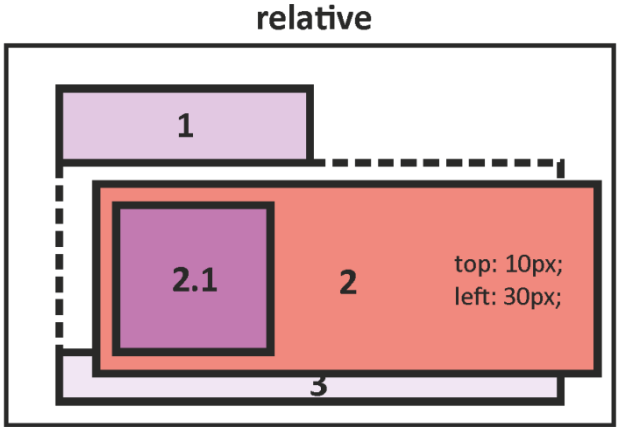
position: fixed

Фиксированное позиционирование аналогично абсолютному, однако элемент всегда зафиксирован в определенной позиции экрана при прокрутке. Фиксированными, например, делают «шапку» и «подвал» сайта, меню быстрого доступа и другие элементы, которые должны всегда быть «на виду».



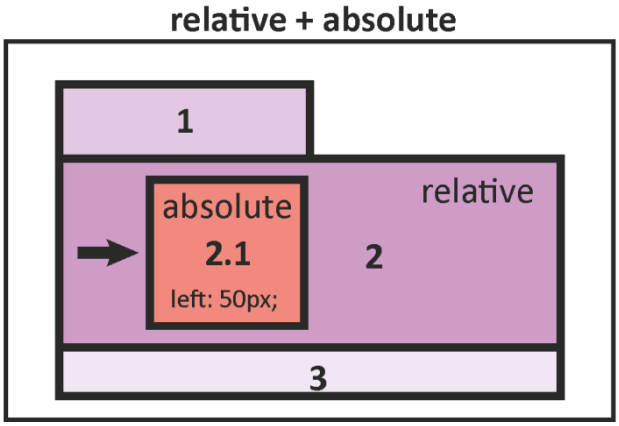
position: relative

Относительное позиционирование работает со свойствами left, right, top и bottom, которые смещают элемент относительно его нормального положения в потоке. При этом другие блоки не занимают освободившееся пространство, как это происходит при абсолютном позиционировании.



position: relative + position: absolute

Если родительский блок позиционирован как относительный, а его дочерний(ие) – абсолютно, то свойства left, right, top и bottom дочерних блоков будут сдвигать их уже относительно родитедя, а не окна браузера. Т.е. в этом случае начало координат локально переносится в родительский блок.



Комбинирование

На самом деле среди всех перечисленных типов позиционирования нет универсального. Каждый подход имеет свои достоинства и недостатки, возможности и ограничения. Поэтому при верстке страницы приходится применять разные подходы или их комбинации.

Z-порядок

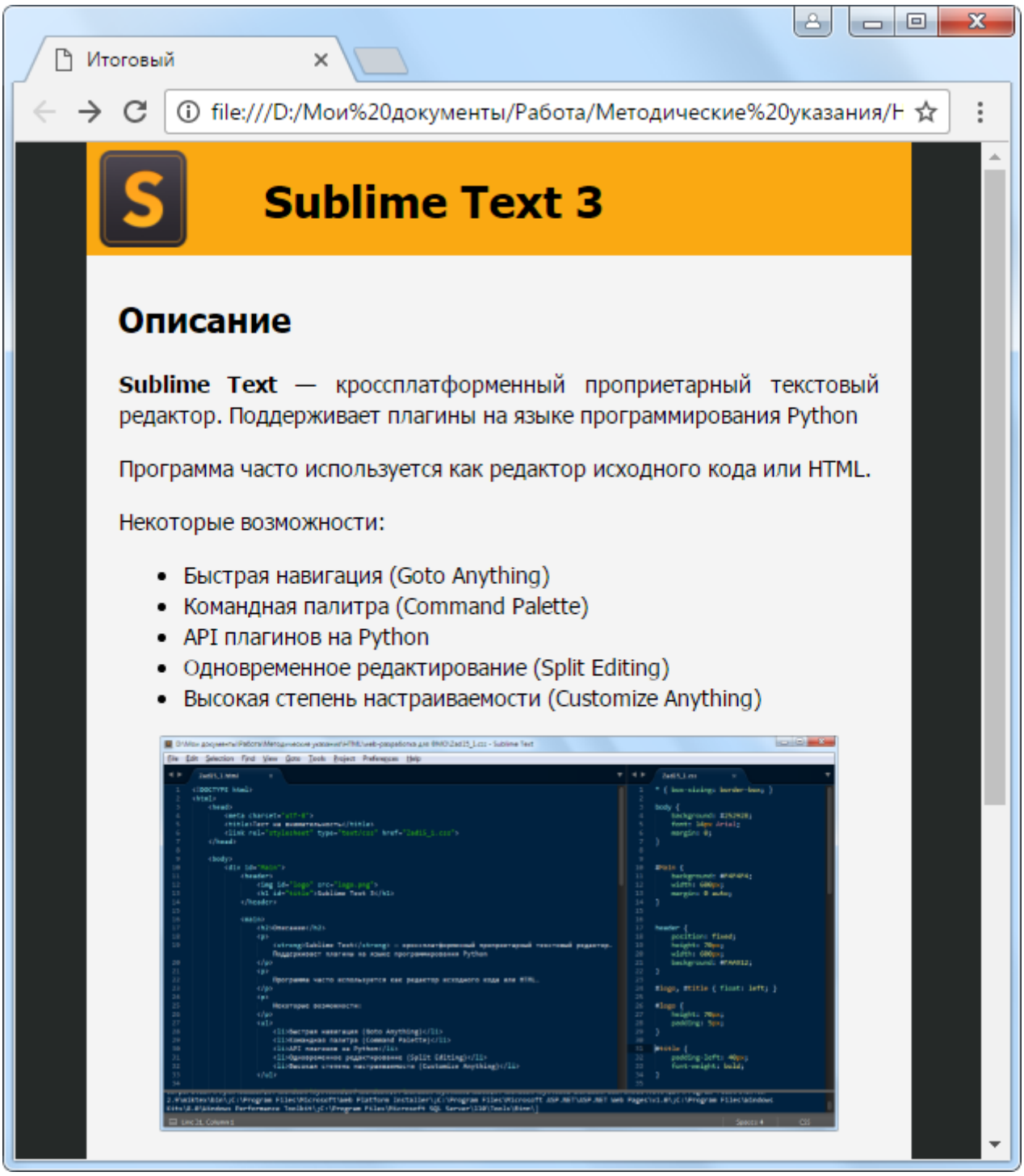
Z-порядок определяет порядок наложения элементов на плоскости экрана. Иными словами, если представить экран как систему координат OXY, то перпендикулярная к экрану ось Z задает уровни наложения слоев.

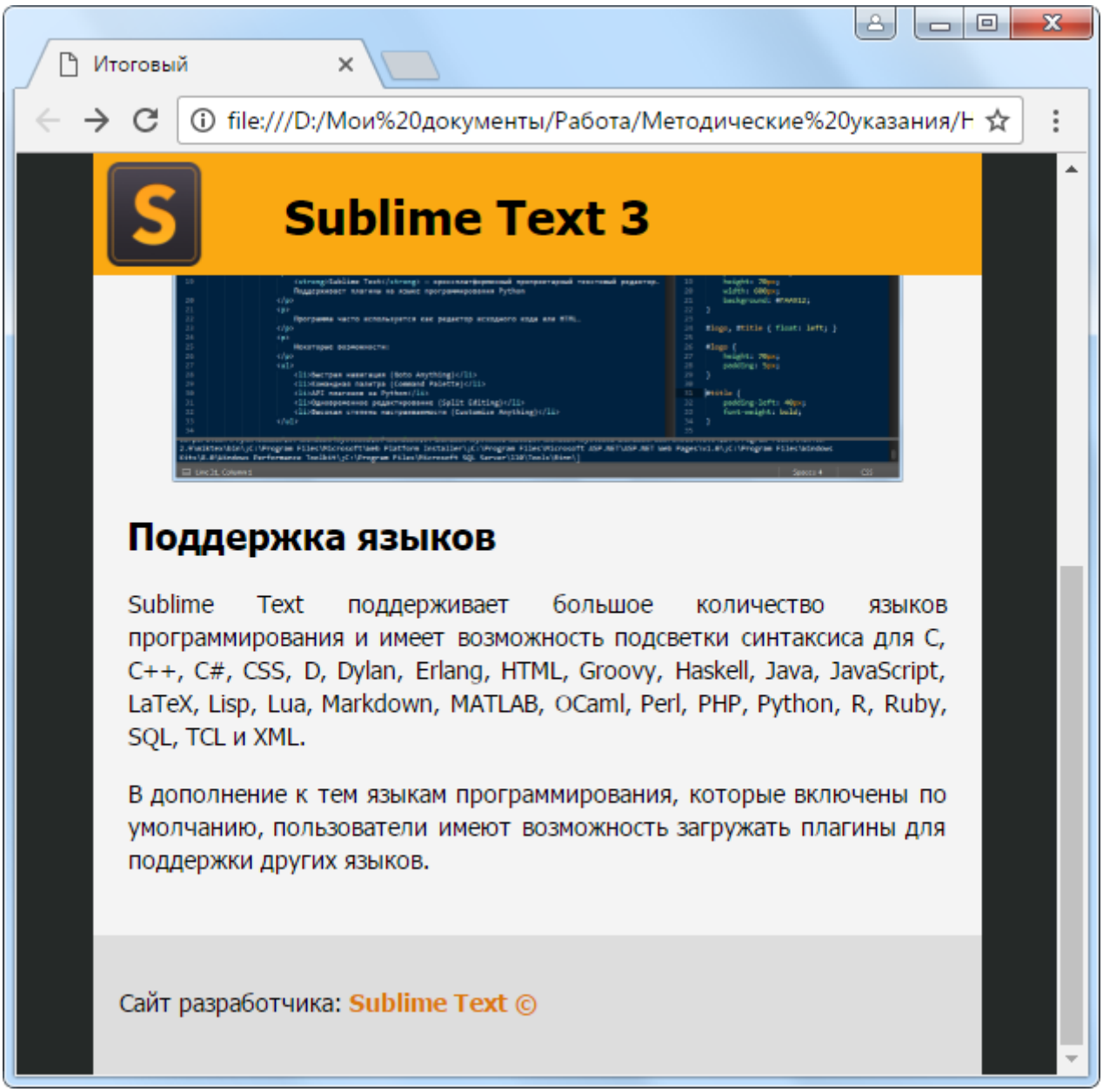
Для определения правильного порядка вывода слоя в CSS используется свойство z-index.

Свойство	Значения и описание
`z-index`	`<значение>`
	Определяет порядок наложения слоя. Чем больше значение, тем выше элемент.

Задания для самостоятельной работы

1. Используя полученные навыки верстки HTML-документов и технологию CSS, реализуйте документ (текст взят из Wikipedia), представленный на изображениях ниже.
    - Основной блок фиксированной ширины и отцентрован.
    - В разметке должны присутствовать «шапка», содержание и «подвал».
    - Блок заголовка зафиксирован при прокрутке.


Итоговый
file:///D:/Мои%20документы/Работа/Методические%20указания/Н
Sublime Text 3
Описание
Sublime Text — кроссплатформенный проприетарный текстовый редактор. Поддерживает плагины на языке программирования Python
Программа часто используется как редактор исходного кода или HTML.
Некоторые возможности:
Быстрая навигация (Goto Anything)
Командная палитра (Command Palette)
API плагинов на Python
Одновременное редактирование (Split Editing)
Высокая степень настраиваемости (Customize Anything)



Итоговый проект

Руководствуясь полученными знаниями по технологиям HTML5 и CSS3, спроектируйте сайт, состоящий из основной страницы и не менее двух дополнительных, вызываемых из главной. Тематика сайта может быть посвящена:

- вашему профилю обучения;
- направлению научной деятельности;
- учебно-педагогической деятельности;
- работе.

В разметочном коде должны присутствовать не менее половины рассмотренных тегов и свойств, реализованы различные приемы позиционирования элементов, каскадирования свойств.

Разработанные проекты могут быть вынесены на отдельное занятие для обсуждения либо развернуты в рамках научной работы студента.

Использование справочника

Все свойства разбиты по категориям.

Символ | означает «ЛИБО» – допустимо одно из перечисленных значений.

Символ || означает «ИЛИ» – допустимо одно или более из указанных значений.

Необязательные параметры указаны в [квадратных скобках].

Значение none указывает на необходимость отмены действия свойства.

Некоторые группы свойств можно объединить одном кратким универсальным (следить в таблицах). Например, код

```
body {
    font-family: Arial;
    font-size: 12pt;
    font-weight: bold;
}
```

эквивалентен коду

```
body { font: bold 12pt Arial; }
```

Свойства шрифта

Свойство	Значения и описание
`font-family`	`[шрифт 1] [шрифт 2] [...] serif | sans-serif | monospace | cursive`
	Задает семейство шрифтов. Можно указывать как имена шрифтов, так и базовые типы: • serif (шрифты с засечками); • sans-serif (рубленные шрифты); • monospace (моноширные, печатная машинка); • cursive (рукописный). Базовые значения можно использовать для подстраховки, если указанные шрифты не будут найдены.
`font-size`	`xx-small | x-small | small | medium | large | x-large | xx-large | smaller | larger | <размер> | <%>`
	Задает размер шрифта. По умолчанию значение medium.
`font-weigth`	`normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900`
	Задает насыщенность. Числовые значения могут не давать результат, поскольку многое может зависеть от гарнитуры шрифта и других фак-

	торов. По умолчанию значение normal.
`font-style`	`italic | oblique | normal`
	Стиль шрифта. Доступен • курсив; • наклонный. По умолчанию значение normal.
`font-variant`	`small-caps | normal`
	Вариант шрифта. Позволяет перевести буквы в малые прописные (капитель). По умолчанию значение normal.
`font`	`[стиль || вариант || насыщенность] размер [/ высота_строки] шрифт`
	Универсальное свойство представляет краткую форму записи перечисленных свойств. Высота строки не относится к свойствам шрифта и обычно настраивается свойством line-height, но допускает использование в font.

Свойства текста

Свойство	Значения и описание
`text-align`	`left | center | right | justify`
	Определяет способ выравнивания текста. По умолчанию значение left.
`text-indent`	`<длина> | <%>`
	Задает отступ красной строки. Начальное значение равно нулю.
`line-height`	`<длина> | <%> | <коэффициент> | normal`
	Задает высоту строк (междустрочечный интервал). Коэффициент высчитывает интервал пропорционально размеру шрифта. По умолчанию значение normal.
`vertical-align`	`baseline | sub | super | top | middle | bottom | text-top | text-bottom | <длина> | <%>`
	Задает вертикальное выравнивание. По умолчанию значение baseline.

word-spacing	<длина> | normal
	Расстояние между словами. По умолчанию значение normal.
letter-spacing	<длина> | normal
	Расстояние между буквами. По умолчанию значение normal.
text-transform	uppercase | lowercase | capitalize | none
	Преобразовывает регистр букв. Значение по умолчанию none.
text-decoration	none | [underline || overline || line-through || blink]
	Оформление текста. Определяет подчеркивание, перечеркивание или мерцание. Значение по умолчанию none.
white-space	normal | nowrap | pre | pre-wrap | pre-line
	Указывает способ обработки пробелов. Значение pre требует отображения всех пробелов. Значение по умолчанию normal.

Фон и цвет

Свойство	Значения и описание
color	<цвет>
	Задает цвет элемента (или текста). Доступны: • табличные названия цветов; • шестнадцатеричный код; • модель rgb(x,x,x); • модель rgba(x,x,x,x) (полупрозрачный цвет).
background-color	<цвет> | transparent
	Задает цвет фона элемента. Значение transparent устанавливает наследование фона предка. Значение по умолчанию transparent.
background-image	url('путь_к_файлу') | none

	Задает фоновое изображение. Если картинка не заполняет весь фон, то укладывается плиткой.
`background-position`	`[ [<длина> | <%> | left | center | right]` `[<длина> | <%> | top | center | bottom] ]`
	Задает позицию фонового изображения (относительно левого верхнего угла) по горизонтали и вертикали. Допускается непосредственно определение координат или с помощью констант.
`background-repeat`	`repeat | repeat-x | repeat-y | no-repeat`
	Регулирует режим повтора фона. • no-repeat - запретить повтор, показать только одно изображение; • repeat - повторять изображение; • repeat-x - повторять только по горизонтали; • repeat-y - повторять только по вертикали. Начальное значение no-repeat.
`background-attachment`	
	Позиционирование относительно прокрутки: • fixed - фиксированный фон; • scroll - подвижный фон (по умолчанию). По умолчанию значение scroll.
`background`	`[цвет || изображение || повтор || скроллинг || положение]`
	Универсальное свойство представляет краткую форму записи перечисленных свойств.

Поля и отступы, размеры

Свойство	Значения и описание
`margin`	`<значение {1,4}>`
	Задает поля (внешний отступ от рамки). Допустимо использовать одно, два, три или четыре значения, перечисляя их через пробел.

`margin-left`
`margin-right`
`margin-top`
`margin-bottom`
	`<значение>`

	Задает соответствующее поле.

`padding`	`<значение {1,4}>`
	Задает отступы (от рамки к контенту). Допустимо использовать одно, два, три или четыре значения, перечисляя их через пробел.

`padding-left`
`padding-right`
`padding-top`
`padding-bottom`
	`<значение>`

	Задает соответствующий отступ.
`width`	`<значение>`
	Задает ширину объекта.
`height`	`<значение>`
	Задает высоту объекта.
`box-sizing`	`content-box | padding-box | border-box`
	Задает способ вычисления ширины и высоты элемента. • content-box – учитывается только контент (по умолчанию); • padding-box – ширина контента + ширина отступов; • border-box – ширина контента + отступов + границ.

Границы

Свойство	Значения и описание

`border`
`border-left`
`border-right`
`border-top`
`border-bottom`
	`<толщина> || <тип_линии> || <цвет>`

	Универсальное свойство задает границу для всей области или для одной из четырех сторон. Доступные типы линии: • none (граница отсутствует); • dotted (граница из точек); • dashed (пунктирная граница); • solid (сплошная граница); • double (двойная сплошная граница); • groove (граница «бороздка»);

	• ridge (граница «гребень»); • inset (вдавленная граница); • outset (выдавленная граница).

`border-left-width`
`border-right-width`
`border-top-width`
`border-bottom-width`
	`<значение> | thin | medium | thick`

	Задает толщину соответствующей границы. Для встроенных значений: • thin (2px); • medium (4px); • thick (6px).

`border-left-style`
`border-right-style`
`border-top-style`
`border-bottom-style`
	`none | dotted | dashed | solid | double | groove | ridge | inset | outset`

	Задает стиль линии соответствующей границы.

`border-left-color`
`border-right-color`
`border-top-color`
`border-bottom-color`
	`<цвет> | transparent`

	Задает цвет соответствующей границы.

`border-top-left-radius`
`border-top-right-radius`
`border-bottom-left-radius`
`border-bottom-right-radius`
	`[<значение>] [<значение>]`

	Устанавливает радиус скругления соответствующего угла. Первое значение – радиус по горизонтали, второе – по вертикале.
`border-radius`	`<значение {1,4}> [/ <значение {1,2}>]`
	Устанавливает радиус скругления уголков рамки. Допустимо использовать одно, два, три или четыре значения, перечисляя их через пробел. Также допустимо писать два значения через слэш (/).

Списки

Свойство	Значения и описание
`list-style-type`	`none | circle | disc | square | armenian | decimal | decimal-leading-zero | georgian | lower-alpha | lower-greek | lower-latin | lower-roman | upper-alpha |upper-latin | upper-roman`
	Задает маркер для маркированного либо нумерованного списка.
`list-style-image`	`none | url('путь_к_изображению')`
	Задает изображение вместо маркера.
`list-style-position`	`outside | inside`
	Определяет способ размещения маркера относительно текста: • outside (маркер за границей элементов списка); • inside (маркер обтекается текстом).
`list-style`	`<тип_маркера> || <позиция_маркера> || <изображение>`
	Универсальное свойство задает настройки списка.

Позиционирование и отображение

Свойство	Значения и описание
`display`	`block | inline | inline-block`
	Определяет способ отображения элемента: • block (элемент отображается блоком и занимает доступную ширину); • inline (элемент становится строковым); • inline-block (элемент становится блоком, но другие элементы, в частности текст его уже будут обтекать).
`float`	`left | right | none`
	Цепляет элемент к указанному краю родительского элемента. При этом другие элементы вынуждены его обтекать.
`clear`	`left | right | both`
	Запрещает обтекание слева, справа или с обоих сторон. Указывается объекту, который может быть подвергнут обтеканию.

`position`	`absolute | fixed | relative | static`
	Определяет способ позиционирования элемента относительно окна браузера или родительского элемента. • absolute (положение задается относительно сторон); • fixed (фиксируется при прокрутке); • relative (смещение относительно нормального положения); • static (по умолчанию).

`left`
`right`
`top`
`bottom`
	`<значение>`

	Задают смещение элемента от указанной стороны
`z-index`	`<значение>`
	Определяет порядок наложения слоя. Чем больше значение, тем выше элемент.

Эффекты

Свойство	Значения и описание
`text-shadow`	`none | <тень>`
	Задает тень тексту. Определяется следующим образом: `<X> <Y> <радиус_размытия> <цвет>` Допускается перечисление нескольких теней через запятую.
`box-shadow`	`none | <тень>`
	Задает тень элементу. Определяется следующим образом: `[inset] <X> <Y> <радиус_размытия> <растяжение> <цвет>` По умолчанию растяжение равно нулю. Допускается перечисление нескольких теней через запятую.

Список литературы

1. **Роббинс Дж.** HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Дженифер Роббинс; [пер. с англ. М.А. Райтман]. — 4-е издание. М.: Эксмо, 2014 — 528 с.
2. **Пьюривал С**. Основы разработки веб-приложений. СПб.: Питер, 2016. – 272 с.: ил. — (Серия «Бестселлеры O'Reilly»).
3. **Дронов В. А.** HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. — СПб.: БХВ-Петербург, 2011. — 416 с.: ил. — (Профессиональное программирование).
4. **Хольцнер, Стивен**. HTML5 за 10 минут, 5-е изд. : Пер. с англ. — М.: ООО «И.Д. Вильямс», 20111. — 240 с.: ил. — Парал. тит. англ.
5. **Роббинс, Дженифер**. HTML5: карманный справочник, 5-е издание.: Пер. с англ. — М.: ООО «И.Д. Вильямс», 2015. — 192 с.: ил. — Парал. тит. англ.
6. **Макфарланд Д.** Большая книга CSS3. 3-е изд. — СПб.: Питер, 2014. — 608 с.: ил. — (Серия «Бестселлеры O'Reilly»).
7. **Мейер Э.** CSS — каскадные таблицы стилей. Подробное руководство. 3-е изд. — Пер. с англ. — СПб: Символ-Плюс, 2008. — 576 с.: ил.

Учебное издание

Якубович Денис Андреевич, **Еропова** Елена Станиславовна,
Еропов Илья Александрович

Основы WEB-разработки

Подписано в печать 28.02.2017. Формат 60x84 1/16. Усл.-печ. л 5,93.
Тираж 200 экз. Заказ 39.

Отпечатано с готового оригинала-макета
ООО «Издательство «Шерлок-пресс»
г. Владимир, ул. Девическая, д. 11