Кафедра информатики, информационных систем и технологий

Петракова Н.В.

ОСНОВЫ CSS

Часть 2

Учебно-методическое пособие
по дисциплине Web-программирование
для самостоятельной работы студентов
по направлению подготовки
09.03.03 Прикладная информатика

Брянск
2023

УДК 004.9 (076)
ББК 32.81
П 30

Петракова, Н. В. Основы CSS: учебно-методическое пособие по дисциплине Web-программирование для самостоятельной работы студентов по направлению подготовки 09.03.03 Прикладная информатика / Н. В. Петракова. – Брянск: Изд-во Брянский ГАУ, 2023. – Ч. 2. – 107 с.

В учебно-методическом пособии раскрыты основы CSS, рассматривается, как средствами CSS оформлять веб-страницы, использовать блоки, списки и таблицы, изображения и гиперссылки, элементы управления, фон и колонки, располагать, выводить и скрывать элементы, делать макеты веб-страниц, выполнять преобразования, анимацию и др.

Учебно-методическое пособие «Основы CSS» предназначено для изучения дисциплины Web-программирование по направлению подготовки 09.03.03 Прикладная информатика. Кроме того, учебное пособие будет полезно студентам других специальностей и направлений, изучающих информационные технологии.

Рецензенты:

к.э.н., доцент кафедры информатики, информационных систем и технологий Федькова Н.А.

к.т.н., доцент кафедры технического сервиса Феськов С.А.

Рекомендовано к изданию решением учебно-методической комиссии института энергетики и природопользования Брянского ГАУ, протокол №1 от 28 сентября 2023 года.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Одной из наиболее динамично развивающихся областей использования информационных технологий является разработка веб-страниц и веб-сайтов различной тематики, направленности и степени сложности.

Вместе с глобальным развитием сети Интернет в программировании все более отчетливо выделяется отдельная его отрасль – Web-программирование.

Учебно-методическое пособие «Основы CSS» предназначено для изучения дисциплины Web-программирование по направлению подготовки 09.03.03 Прикладная информатика, которое посвящено языку описания стилей внешнего отображения веб-документов CSS. Строгое изложение разделов официальной спецификации языка сопровождается конкретными примерами и заданиями использования технологии.

Использование CSS требует знания основ HTML (см. учебно-методическое пособие «Основы HTML. Часть 1»).

В учебно-методическом пособии рассматривается разработка веб-страниц с использованием такого мощного инструмента, как каскадные таблицы стилей CSS. Описаны способы управления селекторами стилей, свойства стилей, отвечающие за оформление различных элементов web-страницы. Также приведены методы и способы создания web-страниц на основе блочной разметки.

Полученные теоретические и практические навыки позволят разрабатывать статическое содержимое веб-страниц и веб-сайтов с использованием современных методов и способов web- программирования.

Пособие соответствует рабочей программе по дисциплине «Web-программирование» и предназначено для студентов очной и заочной форм обучения.

1. КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ: БАЗОВЫЕ ПОНЯТИЯ

1.1. Введение в CSS

Для описания внешнего вида документа используется **язык CSS** (каскадные таблицы стилей), о котором впервые заговорили в 1995 году.

Каскадные таблицы стилей CSS (Cascading Style Sheets) или **язык иерархических стилевых спецификаций** – набор правил отображения, применяемых в документе, к которому присоединяется соответствующая таблица стилей.

Главная цель CSS – отделить структуру документа от его оформления и позволить автору самому решать, как должен выглядеть тот или иной элемент содержания HTML-страницы. Другим достоинством таблиц стилей является возможность обеспечить единый стиль оформления некоторого набора HTML-документов.

Стиль – правило, сообщающее браузеру, как интерпретировать отдельные теги HTML.

Данная технология используется для форматирования документов HTML.

Возможности CSS:

a) определение начертания и размера шрифта, цвет фона и переднего плана, фоновое изображение, размеры границ и другие характеристики;

b) управление отображением стандартных и новых элементов с наложением ограничений;

c) сочетание установок браузера, используемых по умолчанию, с непосредственными инструкциями автора документа и пользователя.

Преимущество CSS в том, что с помощью одного стиля можно оформить сразу все однотипные элементы страницы или целого сайта (сразу все ссылки, абзацы, списки). CSS-стилем один раз определяется, как должен выглядеть тот или иной элемент, например, картинки, и они меняют свое оформление сразу во всех документах. Чтобы изменить форматирование текста по всему сайту, достаточно изменить CSS-код всего один раз.

Основные элементы CSS

Как HTML состоит из тегов, атрибутов и значений, так и **CSS** состоит из своих **собственных элементов**. Суть конструкций CSS можно объяснить так: «Указать, какой элемент страницы оформить, и указать, как его оформлять».

Составляющие конструкции CSS:

- **Селектор**. Идентификатор, который указывает браузеру, к какому именно элементу страницы применить оформление. Благодаря ему браузер «понимает», что стиль предназначен, например, для оформления списков или таблиц.
- **Блок объявления стилей**. Пишется после селектора и заключается в фигурные скобки. В нем задается стиль элемента (его оформление). Блок объявления стилей состоит из двух частей:

– **Свойство**. Аналог атрибута в HTML. Определяет, какое именно свойство оформления будет изменено.

– **Значение**. Задается свойству через двоеточие и определяет, как именно свойство будет изменено.

Свойств и значений в блоке объявления стилей может быть несколько, в таком случае они перечисляются через точку с запятой.

Типы селекторов

Селекторы представляют структуру веб-страницы. С их помощью создаются правила для форматирования элементов веб-страницы.

В зависимости от того, свойства каких элементов страницы они определяют, селекторы бывают разных типов.

- **Универсальный селектор** выбирает все элементы, а сам селектор имеет вид звездочки: ***{блок объявлений}**.

Например, *** {font-size: 14px;}** устанавливает размер шрифта 14 пикселей всем элементам документа, для которых не заданы другие правила с помощью других селекторов.

- **Селектор элемента** позволяет форматировать все элементы данного типа на всех страницах сайта. Название селектора совпадает с именем элемента (тега).

Например, код **h2 {color: red;}** устанавливает красный цвет текста для всех заголовков второго уровня, то есть содержимого тегов **<h2></h2>**.

- **Селектор атрибута** выбирает элементы на основе имени атрибута или значения атрибута. Этим можно пользоваться, чтобы делать оформление более индивидуальным.
- **Селектор класса** позволяет задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы или на разных страницах сайта. Сам селектор по классу начинается с точки, после которой следует имя селектора (класса).

Например, для создания заголовка с классом **headline** необходимо добавить атрибут **class** со значением **headline** в открывающий тег **<h1>** и задать стиль для указанного класса.

HTML-код, который привяжет элемент к правилу, будет следующим:

```
<h1 class="headline">Инструкция пользования персональным компьютером</h1>
```

CSS-код будет таким:

```
.headline {
text-transform: uppercase;
color: lightblue;
}
```

Если элемент имеет несколько атрибутов класса, их значения объединяются с пробелами.

- **Селектор идентификатора** позволяет форматировать один конкретный элемент. Значение **id** должно быть уникальным, на одной странице может встречаться только один раз и должно содержать хотя бы один символ. Значение не должно содержать пробелов. Сам селектор начинается со знака решетки **#,** после которого следует идентификатор (т.е. значение атрибута id).

HTML-код:

```
<div id="sidebar"></div>
```

CSS-код:

```
#sidebar {
width: 300px;
float: left;
}
```

- **Контекстный селектор**. В HTML одни теги часто находятся внутри других, и контекстные селекторы помогают задать правила как раз на такой

случай. С помощью них, например, можно отформатировать элементы внутри нумерованных списков или курсивный текст внутри абзацев:

p i {font-family: Century;}

Остальные группы селекторов основаны на совмещении перечисленных типов, а также на принципе наследования, когда дочерний элемент берет свойства родительского.

Комбинировать и **группировать селекторы** удобно во многих ситуациях.

Например, чтобы задать правила класса **step** только для ссылок, нужно указать оба селектора через точку (сначала тег, потом класс):

a.step {margin-left: 15px;}

Один и тот же стиль можно одновременно применить к нескольким элементам. Для этого необходимо в левой части объявления перечислить через запятую нужные селекторы:

```
h1,
h2,
p,
span {
color: tomato;
background: white;
}
```

Комментарии в CSS используются для добавления поясняющих заметок или для того, чтобы предотвратить интеграцию части кода в браузер.

Синтаксис:

```
/* Комментарий */
```

Примеры:

```
/* Однострочный комментарий */
/* Комментарий,
   который содержит
   несколько
   строк */
```

***Замечания*:**

Данный /* */ синтаксис комментария используется для обоих вариантов, и однострочного и многострочного комментария. Нет других способов добавить комментарий во внешнюю таблицу стилей. Как и в большинстве языков программирования, которые используют синтаксис комментариев /* */, комментарии нельзя вкладывать друг в друга.

1.2. Виды каскадных таблиц стилей

Внешняя таблица стилей представляет собой текстовый файл с расширением **.css**, в котором находится набор CSS-стилей элементов. Файл создается в редакторе кода, так же как и HTML-страница. Внутри файла могут содержатся только стили, без HTML-разметки. Внешняя таблица стилей подключается к веб-странице с помощью элемента **<link>**, расположенного внутри раздела **<head></head>**. Такие стили работают для всех страниц сайта.

К каждой веб-странице можно присоединить несколько таблиц стилей, добавляя последовательно несколько элементов **<link>**, указав в атрибуте **media** назначение данной таблицы стилей.

rel="stylesheet" указывает тип ссылки (ссылка на таблицу стилей).

Например:

```
<head>
<link rel="stylesheet" href="css/style.css">
<link rel="stylesheet" href="css/assets.css" media="all">
</head>
```

Внутренние (глобальные) стили встраиваются в раздел **<head></head>** HTML-документа и определяются внутри элемента **<style></style>**. Внутренние стили имеют приоритет над внешними, но уступают встроенным стилям (заданным через атрибут **style**).

HTML-код:

```
<head>
<style>
h1,
h2 {
color: red;
font-family: "Times New Roman", Georgia, Serif;
line-height: 1.3em;
}
</style>
</head>
<body>
...
</body>
```

Встроенные стили – CSS-код помещается в HTML-файл, непосредственно внутри элемента с помощью атрибута **style**:

```
<p style="font-weight: bold; color: red;">Обратите внимание на этот текст.</p>
```

Такие стили действуют только на тот элемент, для которого они заданы.

Правило @import (импортированные стили) позволяет загружать внешние таблицы стилей. Чтобы директива **@import** работала, она должна располагаться в таблице стилей (внешней или внутренней) перед всеми остальными правилами.

Код ниже импортирует в документ таблицу файла **mobile.css**, которая находится в папке с редактируемым HTML-документом:

```
@import url(mobile.css);
```

Команда прописывается строкой ниже открывающего тега **<style>**, до первой строки внутреннего стиля:

HTML-код:

```
<style>
@import url(mobile.css);
p {
font-size: 0.9em;
color: grey;
}
</style>
```

Правило @import также используется для подключения веб-шрифтов:

CSS-код:

```
@import
url(https://fonts.googleapis.com/css?family=Open+Sans&subset=latin,cyrillic);
```

1.3. Наследование и каскад

Наследование и каскад – два фундаментальных понятия в CSS, которые тесно связаны между собой.

Наследование заключается в том, что элементы наследуют свойства от своего родителя (элемента, их содержащего).

Каскад проявляется в том, как разные виды таблиц стилей применяются к документу, и как конфликтующие правила переопределяют друг друга.

Наследование является механизмом, с помощью которого определенные свойства передаются от предка к его потомкам. Спецификацией CSS предусмотрено наследование свойств, относящихся к текстовому содержимому страницы, таких как **color**, **font**, **letter-spacing**, **line-height**, **list-style**, **text-align**, **text-indent**, **text-transform**, **visibility**, **white-space** и **word-spacing**. Во многих случаях это удобно, так как не нужно задавать размер шрифта и семейство шрифтов для каждого элемента веб-страницы.

Свойства, относящиеся к форматированию блоков, не наследуются. Это **background**, **border**, **display**, **float и clear**, **height и width**, **margin**, **min-max-height** и **-width**, **outline**, **overflow**, **padding**, **position**, **text-decoration**, **vertical-align** и **z-index**.

Принудительное наследование

С помощью ключевого слова **inherit** можно принудить элемент наследовать любое значение свойства родительского элемента. Это работает даже для тех свойств, которые не наследуются по умолчанию.

Как задаются и работают CSS-стили

Стили могут наследоваться от родительского элемента (наследуемые свойства или с помощью значения **inherit**).

Стили, расположенные в таблице стилей ниже, отменяют стили, расположенные в таблице выше.

К одному элементу могут применяться стили из разных источников. Проверить, какие стили применяются, можно в режиме разработчика браузера. Для этого над элементом нужно щелкнуть правой кнопкой мыши и выбрать пункт «Посмотреть код» (или что-то аналогичное). В правом столбце будут перечислены все свойства, которые заданы для этого элемента или наследуются от родительского элемента, а также файлы стилей, в которых они указаны, и порядковый номер строки кода.

При определении стиля можно использовать любую комбинацию селекторов – селектор элемента, класса или идентификатора элемента.

HTML-код:

```
<div id="wrap" class="box clear"></div>
```

CSS-код:

```
div {border: 1px solid #eee;}
#wrap {width: 500px;}
.box {float: left;}
.clear {clear: both;}
```

Каскадирование – это механизм, который управляет конечным результатом в ситуации, когда к одному элементу применяются разные CSS-правила. Существует три критерия, которые определяют порядок применения свойств – правило !important, специфичность и порядок, в котором подключены таблицы стилей.

Правило !important

Вес правила можно задать с помощью ключевого слова **!important**, которое добавляется сразу после значения свойства, например, **span {font-weight: bold!important;}**. Правило необходимо размещать в конец объявления перед закрывающей скобкой, без пробела. Такое объявление будет иметь приоритет над всеми остальными правилами. Это правило позволяет отменить значение свойства и установить новое для элемента из группы элементов в случае, когда нет прямого доступа к файлу со стилями.

Специфичность

Для каждого правила браузер вычисляет специфичность селектора, и если у элемента имеются конфликтующие объявления свойств, во внимание принимается правило, имеющее наибольшую специфичность. Значение специфичности состоит из четырех частей: 0, 0, 0, 0. Специфичность селектора определяется следующим образом:

- для **id** добавляется 0, 1, 0, 0;
- для **class** добавляется 0, 0, 1, 0;
- для каждого элемента добавляется 0, 0, 0, 1;
- для встроенного стиля, добавленного непосредственно к элементу – 1, 0, 0, 0;
- универсальный селектор не имеет специфичности.

CSS-код:

```
h1 {color: lightblue;} /*специфичность 0, 0, 0, 1*/
em {color: silver;} /*специфичность 0, 0, 0, 1*/
h1 em {color: gold;} /*специфичность: 0, 0, 0, 1 + 0, 0, 0, 1 = 0, 0, 0, 2*/
div#main p.about {color: blue;} /*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 + 0,
0, 0, 1 + 0, 0, 1, 0 = 0, 1, 1, 2*/
.sidebar {color: grey;} /*специфичность 0, 0, 1, 0*/
#sidebar {color: orange;} /*специфичность 0, 1, 0, 0*/
li#sidebar  {color: aqua;} /*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 = 0, 1, 0,
1*/
```

В результате к элементу применятся те правила, специфичность которых больше. Например, если на элемент действуют две специфичности со значениями 0, 0, 0, 2 и 0, 1, 0, 1, то выиграет второе правило.

Порядок подключенных таблиц

Можно создать несколько внешних таблиц стилей и подключить их к одной веб-странице. Если в разных таблицах будут встречаться разные значения свойств одного элемента, то в результате к элементу применится правило, находящееся в таблице стилей, идущей в списке ниже.

1.4. Псевдоэлементы и псевдоклассы

Псевдоэлементы позволяют задать стиль для части элемента (например, для первой буквы или первой строки абзаца), а также генерировать содержимое, которого нет в HTML-коде документа.

Псевдоэлемент состоит из двоеточия (:) и имени.

Синтаксис:

```
селектор:псевдоэлемент
  {CSS свойство: значение;...
  }
```

Псевдоэлементы **:before** и **:after** используется вместе со свойством **content**, которое необходимо для вставки сгенерированного контента.

Псевдоэлемент **:before** добавляет определенное содержимое перед каждым указанным элементом.

Псевдоэлемент **:after** добавляет определенное содержимое после каждого элемента.

Псевдоэлемент **:first-line** – оформляет первую строку абзаца, позволяя определить свойства шрифта (font), цвета и заднего плана (color и background), интервал между словами (word-spacing), интервал между символами (letter-spacing), декоративное оформление текста (text-decoration), вертикальное выравнивание (vertical-align), трансформацию текста (text-transform), высоту строки (line-height) и параметры обтекания (clear).

Псевдоэлемент **:first-letter** – оформляет первый символ абзаца, позволяя определить свойства шрифта (font), цвета и заднего плана (color и background), декоративное оформление текста (text-decoration), вертикальное выравнивание (vertical-align) – только если свойство float установлено в "none" (по умолчанию), трансформацию текста (text-transform), высоту строки (line-height), параметры обтекания (clear), отступы (margin), расстояние от обрамления (padding), параметры обрамления (border) и отделение от остального текста (float).

Псевдокласс – это специфическая характеристика элемента, которая создается с помощью CSS.

Псевдокласс определяет динамическое состояние элемента и применяется тогда, когда пользователь выполняет определенное действие.

Синтаксис:

```
селектор:псевдокласс
  {CSS свойство: значение;...
  }
```

В CSS существует более тридцати псевдоклассов.

Наиболее распространенные псевдоклассы:

`:link` `:visited` `:hover` `:active` `:focus` `:first-child` `:nth-child`

Стилизация ссылок

В CSS существуют четыре псевдокласса, они позволяют работать со ссылками. У ссылок есть четыре состояния: простая, активная, посещенная и та, на которую наведен курсор. Состояние ссылок зависит от действия пользователя, и браузер, в зависимости от этих действий может применять разные стили. Для описания этих стилей и существуют псевдоклассы.

a:link – задает стиль обычной ссылки.

a:active – задает стиль активной ссылки.

a:visited – задает стиль посещенной ссылки.

a:hover – задает стиль ссылки, на которую наведен курсор.

Псевдокласс **:hover** может быть применен не только к ссылкам.

При наведении указателя мыши можно подсветить строку или ячейку таблицы.

Фокус на элементе

Данный псевдокласс **:focus** происходит, когда элемент HTML получает фокус. Это состояние обычно используется для полей форм.

Структурные псевдоклассы

Псевдокласс **:nth-child** используется для добавления стиля к элементам на основе нумерации в дереве элементов.

Синтаксис:

:nth-child(номер) – определяет элемент по его номеру в дереве элементов.

```
Значения номера:

 1. :nth-child(odd) - Все нечетные номера элементов.

 2. :nth-child(even) - Все четные номера элементов.

 3. :nth-child(число) - Порядковый номер дочернего элемента относительно своего родителя.
    Нумерация начинается с 1, это будет первый элемент в списке.

 4. :nth-child(выражение) - Выражение задается в виде an+b, где a и b целые числа. n — счетчик,
    который автоматически принимает значение 0, 1, 2...
```

2. СВОЙСТВА CSS

2.1. Форматирование текста

Свойства CSS для форматирования текста позволяют оформить содержимое страницы, не затрагивая HTML-код.

Выравнивание по горизонтали

Для него используется свойство **text-align**. Выровнять с его помощью можно только блочный текст (теги ** **, **<p>**). Свойству может быть задано одно из четырех значений:

left – выравнивание по левому краю.

right – по правому краю.

center – по центру.

justify – по ширине.

Если текст выровнен по ширине (**text-align: justify**), то можно использовать свойство **text-align-last**, чтобы задать отличное от основного содержимого выравнивание последней строчки текста элемента.

Выравнивание по вертикали

Может задаваться только для строчных элементов (картинок, форм), определяется свойством **vertical-align**. С его помощью выравнивается не содержимое, а сами элементы, кроме случая с ячейкой – использование **vertical-align** выравнивает не ее саму, а только расположенный в ней текст.

Значения могут быть следующими:

- **baseline**. Задается свойству по умолчанию и выравнивает базовую линию элемента по базовой линии родителя. Если у родителя ее нет, то выравнивание происходит по нижней границе.
- **top** и **bottom**. Если задано первое значение, то верхний край элемента будет совпадать с верхним краем самого высокого элемента строки. Можно сказать, что **top** – это выравнивание по верхнему краю. Второе свойство выполняет противоположную функцию – совмещает нижний край оформляемого элемента с нижней частью элемента, расположенного в строке ниже всех, то есть происходит выравнивание по нижнему краю.
- **text-top** и **text-bottom**. От предыдущих свойств отличаются тем, что выравнивание происходит по самым нижним и верхним текстовым элементам, а не любым.

- **sub** и **super**. Аналоги HTML-тегов **<sub>** и **<sup>**. Первое свойство делает элемент подстрочным, второе – надстрочным. Шрифт текста при этом не меняется.

  middle. Выравнивание по центру относительно элемента-родителя.

Также с помощью **vertical-align** можно переместить элемент вверх или вниз, указав значение в пикселях, единицах или процентах. Положительная цифра переместит его вверх, отрицательная – вниз.

Отступ первой строки

Свойство **text-indent** позволяет задать отступ первой строки текста. Например, так можно отформатировать абзацы, чтобы лучше визуально отделить их друг от друга. В качестве значения используется цифра, задающая длину в процентах, единицах или пикселях. Отрицательное число превратит отступ в выступ.

Междустрочный интервал

Задается свойством **line-height**, в качестве значения которого может указываться:

Процент. Высчитывается от размера шрифта элемента.

Число. Определяется как множитель от размера шрифта, который принимается за единицу. Например, line-height: 1.5; установит полуторный интервал.

Пиксели или ***пункты***. Определяют не переменное, как предыдущие варианты, а постоянное расстояние.

Декорирование текста

Значение свойства **text-decoration** позволяет сделать текст зачеркнутым (**line-through**), подчеркнутым (**underline**) – линия появляется под текстом, надчеркнутым (**overline**) – линия появляется над текстом, или отменить эффекты (**none**).

Интервал между символами и словами

Расстояние между словами можно изменить с помощью свойства **word-spacing**. Межсимвольное расстояние задается свойством **letter-spacing**. В качестве значений используются любые принятые в CSS единицы длины.

Смена регистра

Указав свойство **text-transform**, вы можете сделать так, чтобы все буквы текста были заглавными (значение **uppercase**), строчными (**lovercase**), или чтобы каждое слово начиналось с большой буквы (**capitalize**).

2.2. Оформление шрифтов

Свойствами CSS можно задавать разные шрифты разным элементам веб-страницы, а также указать их начертание, размер, цвет и другие параметры.

Рассмотрим доступные свойства.

font-family

Позволяет определить, каким шрифтом будет написан текст.

Для группировки шрифтов в CSS используется два типа имен: **generic-family** и **family-name**.

generic-family содержит пять базовых семейств шрифтов:

- **sans-serif** – шрифты без засечек. Написанный ими текст воспринимается лучше других.
- **serif** – шрифты с засечками.
- **monospace** – шрифты, символы которых имеют фиксированную ширину. Их принято использовать для отображения программного кода.
- **cursive** – рукописные шрифты.
- **fantasy** – декоративные (художественные) шрифты.

family-name определяет не семейство, а один шрифт: Arial Black, Verdana.

В качестве значений свойства **font-family** семейства и шрифты перечисляются через запятую. Браузер определяет, установлен ли на ПК пользователя первый в списке шрифт, и если да, то отображает текст им, если нет – переходит к следующему и т. д. Если имя шрифта содержит спецсимволы или состоит из нескольких слов (содержит пробел), то его надо брать в кавычки.

```
h1 {font-family: Arial, sans-serif;}
h2 { font-family: "Times New Roman", serif; }
```

font-style

Задает курсивное (значение **italic**), наклонное (**oblique**) или стандартное (**normal**) начертание текста.

font-size

Важное свойство, которое определяет размер шрифта.

Его можно задавать разными способами.

- ***Пиксели*** (**px**). Используются довольно часто, потому что позволяют указать размер максимально точно. По умолчанию браузер отображает текст размером 16 пикселей. Между числом и единицами измерения пробела быть не должно.

```
p { font-size: 12px; }
```

- ***Проценты*** (**%**). Высчитываются от размера шрифта родительского элемента. Если у родителя он установлен по умолчанию, то можно вспомнить про 16 пикселей и принять их за 100%.

```
p { font-size: 120%; }
```

- ***Пункты*** (**pt**). Тоже используются очень часто. Более того, когда вы выбираете число, выставляя размер шрифта в текстовых редакторах (Word, Блокнот, Notepad и т. д.), то также используете пункты.

```
p { font-size: 15pt; }
```

- ***Относительная высота шрифта*** (**em**). Высота шрифта родителя принимается за единицу, и относительно него устанавливается высота шрифта текущего элемента.

```
p { font-size: 1.2em; }
```

- ***Константы***. Считается, что значения **xx-small**, **x-small**, **small**, **medium**, **large**, **x-large**, **xx-large** (от меньшего к большему) задают абсолютный размер шрифта, хотя на самом деле при их использовании размер все равно будет зависеть от настроек операционной системы и браузера. Есть также и относительные константы: **smaller** (меньший) и **larger** (больший), которые уменьшают или увеличивают шрифт относительно элемента-родителя.

```
p { font-size: x-large; }
```

- ***Другие единицы измерения***. Для установки шрифта можно использовать все, что доступно в CSS: ***миллиметры*** (**mm**), ***сантиметры*** (**cm**), ***дюймы*** (**in**), ***пики*** (**pc**, 1 pc = 12 pt), ***размер символа x*** (**ex**).

```
p { font-size: 1.5pc; }
```

font-variant

Предлагает два значения:
small-caps – строчные буквы меняются на прописные малого размера.
normal – обычный текст.

```
h1 { font-variant: small-caps; }
h2 { font-variant: normal; }
```

font-weight

Влияет на насыщенность (жирность) шрифта.

Значения могут быть такими:

- **Числа от 100 до 900 с шагом в сотню**. С их помощью можно задать жирность точнее всего. Насыщенность нормального шрифта, который вы обычно видите на страницах, равна значению 400, полужирного – 700. Проблема в том, что многие браузеры не поддерживают весь этот ассортимент, и поэтому использование числовых значений часто бывает бессмысленным.
- **normal** – обычный шрифт.
- **bold** – устанавливает полужирное начертание.

```
div { font-weight: bold; }
```

- **bolder и lighter** – делает шрифт соответственно жирнее или тоньше шрифта текста родителя.

color

Определяет цвет шрифта. Может задаваться следующими способами:

- **По названию**. Используются зарезервированные слова, обозначающие имена цветов (*например*, blue – синий, yellow – желтый). Самый простой, но вместе с тем и самый ограниченный метод, потому что поддерживает только 140 цветов, хотя обычно и их хватает.

```
p { color: red; }
```

- **По шестнадцатеричному коду (HEX)**. Шестнадцатеричная система счисления включает в себя 16 цифр, последние шесть из которых обозначаются буквами латинского алфавита от A до F, причем десятичное число 255 в этой системе записывается как FF. Как известно, все цвета создаются путем смешивания всего трех: красного, зеленого и синего. Степень присутствия каждого из них и задается числами от 0 до 255, записанными в шестнадцатеричной системе. Перед числом ставится символ «решетка (#). Белый цвет – #000000, черный – #ffffff.

  Пример ниже задает абзацу серый цвет:

```
p { color: #808080; }
```

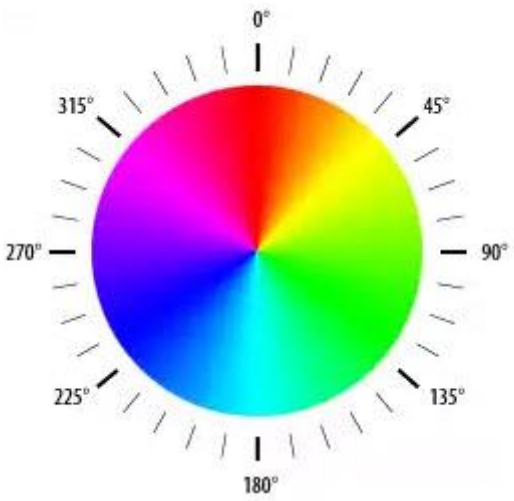
- **В системе RGB**. Все те же красный, зеленый и синий (red, green, blue), но задающиеся уже числами от 0 до 255 через запятую. Присутствие каждого из трех цветов можно обозначать и в процентах. Тогда 100% будет соответствовать числу 255.

```
a {color: rgb(0,128,201);}
```

- **RGBA**. Все, как в RGB, только в конце прибавляется еще одно число от 0 до 1 – альфа-канал, задающий тексту прозрачность.

```
a {color: rgba(0,128,201,0.5);}
```

- **HSL**. Значение в этом формате задается тремя параметрами через запятую.
  1) **H (hue – оттенок)**. Определяется в градусах от 0 до 359° в зависимости от цвета на цветовом круге (рисунок ниже).



  2) **S (Saturate – оттенок)**. Указывается в процентах. Понятно, что от 0 до 100%, и понятно, что чем больше процент, тем насыщеннее цвет.
  3) **L (lightness – яркость)**. Так же, как и насыщенность, устанавливается в процентах.

  Насыщенный красный цвет задается:

```
a {color: hsl(0,100%,100%);}
```

- **HSLA**. То же, что и HSL, но с добавлением альфа-канала (как RGBA).

```
a {color: hsla(0,100%,100%,0.7);}
```

font

Позволяет объединить все свойства шрифта в одной строке и может тем самым сильно сократить код.

Имеет следующую структуру:

```
font: font-style font-variant font-weight font-size/line-height font-family
```

Свойства должны записываться именно в такой последовательности (чтобы не задавать значение, его можно просто пропустить).

Например:

```
div {font: oblique small-caps 12pt/1.2 Arial, sans-serif;}
```

2.3. Оформление списков

Для списков предусмотрены следующие CSS-правила.

list-style-type

Задает маркер или нумерацию списков вместо атрибута type в HTML-коде. Значения свойств для маркированных списков могут быть:

disk – кружок, установлен по умолчанию.

circle – окружность.

square – квадрат.

Для нумерованных списков свойству обычно присваиваются значения:

decimal – арабские цифры, значение установлено по умолчанию.

lower-roman – маленькие римские цифры.

upper-roman – заглавные римские цифры.

lower-alpha – строчные латинские буквы.

upper-alpha – прописные латинские буквы.

Также для любого типа списка свойству **list-style-type** можно указать значение **none**, которое вообще уберет маркер.

Для нумерованных списков доступны и другие значения, *например*, **cjk-ideographic** задает идеографическую нумерацию, **armenian –** традиционную армянскую, а **decimal-leading-zero** установит нумерацию римскими цифрами, но с нулем в начале: 01, 02, 03… 09, однако на практике эти и подобные им значения используются крайне редко.

Цвет маркеров совпадает с цветом текста в списке, указанного свойством color.

list-style-image

Позволяет установить в качестве маркера списка собственное изображение.

Например, если в папке с содержащей список страницей находится файл marker.png, то код оформления будет следующим:

```
ul { list-style-image: url("marker.png"); }
```

list-style-position

Определяет положение маркера: либо он вынесен за границу элемента списка (**list-style-position: outside**), либо текст его обтекает (**list-style-position: inside**).

list-style

Позволяет сократить код, записав все три перечисленных свойства одной строкой.

Записываются правила через пробел:

```
ul { list-style: square inside; }
```

2.4. Оформление ссылок

Для ссылок в таблицах стилей предусмотрено четыре специальных псевдокласса. От классов псевдоклассы отличаются тем, что по факту они прикреплены не к тегам, а к событиям или правилам.

Изменить рамку поля, в которое сейчас вводится текст, подчеркнуть красным неправильно заполненные поля формы, задать отдельное оформление тексту, написанному на другом языке – все это и многое другое могут псевдоклассы. От прочих типов селекторов они отличаются двоеточием в начале: **:read-only**.

Для оформления ссылок используется ***четыре псевдокласса***:

:link – ссылка, по которой еще не переходили.

:hover – ссылка, над которой сейчас находится курсор мыши.

:active – ссылка, которую в данный момент нажимает пользователь.

:visited – посещенная ссылка, то есть та, по которой уже переходили.

Стилевое оформление:

color: цвет – меняет цвет ссылки в зависимости от того, нажали ее или нет, только подвели к ней курсор или уже по ней перешли.

text-decoration: none; – убирает заданное ссылкам по умолчанию подчеркивание текста.

font-size: 18px; – меняет размер шрифта ссылки в момент, когда к ней подводится курсор. В момент нажатия ссылки размер шрифта не меняется и остается равным 18px, так как пользователь, нажимая ссылку, по-прежнему не убирает с нее курсор мыши, а псевдоклассу **:active** другой размер шрифта не задан.

text-decoration: overline; – зато свойству **:active** задано отличное от псевдокласса **:hover** правило декорирования текста. Именно поэтому в момент нажатия ссылки появляется надчеркивание, тогда как при наведении курсора без щелчка ничего нет.

2.5. Оформление таблиц

Для форматирования таблиц каскадные стили используют перечисленные ниже свойства.

width и height

Задают соответственно ширину и высоту таблицы.

Без этих свойств параметры определяются автоматически и зависят от содержимого контейнера **<table>**. Значения устанавливаются в любых единицах длины CSS, но зачастую используются пиксели (px) и проценты (%). Последние настраивают ширину относительно родительского элемента, первые же задают абсолютную величину.

```
table {width: 450px; height: 80%;}
```

caption-side

Указывает, где будет размещен заголовок таблицы, описанный тегом **<caption>**.

Свойству можно задавать значения:

top – расположить над таблицей.

bottom – разместить под таблицей.

Эксклюзивно для обозревателя Firefox доступны значения **left** (заголовок слева) и **right** (справа от таблицы), но другие браузеры их не понимают.

```
table {caption-side: top;}
```

border-collapse

Помогает избежать ситуаций, когда границы ячеек образуют двойные рамки.

Так рамки ячеек отображаются по умолчанию. Правило **border-collapse: separate;** дает такой же эффект. Чтобы решить проблему, нужно объявить **border-collapse: collapse;**

border-spacing

Определяет расстояние между границами ячеек.

Правило задается сразу для всей таблицы. Если значение одно, то оно установит расстояние и по горизонтали, и по вертикали. Если значения два, то первое

задаст расстояние по горизонтали, второе – по вертикали. Свойство несовместимо с правилом **table {border-collapse: collapse;}**.

CSS-код:

```
table {
 border: 4px double #FCA360;
 border-collapse: separate;
 border-spacing: 10px 20px;
}
td {
 padding: 3px;
 border: 1px solid #FCA360;
}
```

empty-cells

Указывает, будет ли отображаться фон и границы ячейки, если она пуста. Значение у свойства может быть одно из двух:

show – показывать границы и фон (по умолчанию).

hide – скрыть их. Если все ячейки строки пусты, то будет скрыта, соответственно, вся строка. Если таблице задано правило **border-collapse: collapse;**, то свойство игнорируется.

table-layout

Указывает браузеру, как определять ширину ячеек таблицы, основываясь на их содержимом.

- **auto**. Ширина определяется автоматически. При этом либо суммируется ширина всех столбцов, либо берется значение свойства width, если таблице оно задано. Браузер сначала загружает таблицу, потом анализирует ее, определяя ширину, и только после этого отображает.
- **fixed**. Фиксированная ширина, которая определяется по первой строке.

Пример оформления таблицы:

```
<!DOCTYPE html>
<html>
 <head>
  <title>border-collapse</title>
  <style>
   table {
    width: 50%;
    caption-side: bottom;
    border: 4px solid #006400;
    border-collapse: collapse;
    table-layout: fixed;
   }
   th {
    font-size: 13px;
    font-weight: bold;
    background: #ADFF2F;
    border-top: 4px solid #006400;
    border-bottom: 3px solid #FF8C00;
    color: #039;
    padding: 8px;
   }
   td {
    background: #E0FFFF;
    border-bottom: 1px solid #FF8C00;
    border-top: 1px solid transparent;
    padding: 8px;
   </style>
 </head>
 <body>
  <table>
   <caption>Пример таблицы</caption>
   <tr>
    <th>Цены</th><th>2014</th>
    <th>2015</th><th>2016</th>
   </tr>
   <tr>
    <td>Хлеб</td><td>16</td>
    <td>18</td><td>21</td>
   </tr>
   <tr>
    <td>Сахар</td><td>35</td>
    <td>44</td><td>50</td>
   </tr>
   <tr>
    <td>Соль</td><td>8</td>
    <td>8,50</td><td>9</td>
   </tr>
  </table>
 </body>
</html>
```

Цены	2014	2015	2016
Хлеб	16	18	21
Сахар	35	44	50
Соль	8	8,50	9

Пример таблицы

width: 50%;

Ширина таблицы в половину от родительской.

50% берется от ширины контейнера **<body>**, потому что других родителей у нее нет. То есть таблица будет занимать ровно половину окна браузера.

caption-side: bottom;

Заголовок снизу, под таблицей.

border: 4px solid #006400;

Цветная рамка толщиной 4 пикселя.

border-collapse: collapse;

Объединение границы ячеек.

table-layout: fixed;

Настраиваем способ определения браузером ширины таблицы.

font-size: 13px;

Задаем размер шрифта заглавных ячеек.

font-weight: bold;

Делаем текст внутри них жирным.

background: #ADFF2F;

Устанавливаем цвет фона ячеек.

border-top: 4px solid #006400; border-bottom: 3px solid #FF8C00;

Настраиваем верхние и нижние рамки.

color: #039;

Определяем цвет текста.

padding: 8px;

Задаем расстояние от содержимого ячеек до их границ равным восьми пикселям.

2.6. CSS-фон

background-color

Устанавливает задний фон любого HTML-элемента: таблицы, абзаца, списка и т. д.

Например, цвет фона всей страницы можно настроить так:

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

background-image

Устанавливает в качестве фона картинку. Если изображение подобрано правильно, то страница на его фоне будет выглядеть гораздо красивее, чем на фоне одноцветном.

Например, картинка **background.png** хранится в той же папке, что и веб-страница, и нужно установить ее как фон параграфам этого HTML-документа. Код будет таким:

```
p { background-image: url(background.png); }
```

Для одного и того же веб-элемента можно установить несколько фоновых рисунков. Для этого достаточно перечислить их все в свойстве **background-image** через запятую.

Очевидно, что без определенного позиционирования выбранные рисунки при отображении будут накладываться друг на друга, причем первое из перечисленных будет самым верхним, последнее – под всеми, задним фоном.

background-repeat

По умолчанию фоновое изображение, если его размер меньше размера элемента, будет заполнять объект полностью, повторяясь с левого верхнего к правому нижнему углу.

Бывают ситуации, когда такое заполнение неприемлемо. Как раз для них и существует свойство **background-repeat**, которому можно указать следующие значения:

- **repeat**. Задано по умолчанию.
- **repeat-x**. Картинка будет повторяться только слева направо, по горизонтали. По вертикали не будет, так что заполнит только первый слой.
- **repeat-y**. Рисунок будет повторяться по вертикали, но не по горизонтали, то есть идти вдоль левой границы страницы.
- **no-repeat**. Фон не будет повторяться вообще.

background-position

Позволяет задать позицию фоновому изображению, ведь не всегда нужно, чтобы оно отображалось с левого верхнего угла окна браузера. Для позиционирования достаточно указать два значения через пробел: координату по оси X (горизонтальное позиционирование) и координату по оси Y (позиционирование вертикальное). Задавать их можно в любых единицах длины, но для абсолютных значений рекомендуется использовать пиксели (px), а для относительных – проценты (%).

```
background-position: 200px 350px;
```

Также картинку можно позиционировать зарезервированными словами.

- Для горизонтального расположения используются слова **left**, **center** и **right** (по левому краю, по центру и по правому краю соответственно).
- По вертикали положение задают значения **top**, **center** и **bottom** – позиционирование сверху, по центру и снизу.

Свойство **background-position** позволяет устанавливать в качестве фона несколько картинок и задавать правила так, чтобы изображения не накладывались друг на друга. Чтобы это сделать, достаточно перечислить пары координат для всех фоновых рисунков в одном правиле.

Например, позиционировать три картинки можно так:

```
background-position: center left, 0% 50%, 800px 0px;
```

background-attachment

Определяет, будет ли фоновая картинка прокручиваться вместе с содержимым страницы или будет оставаться неподвижной. Значения могут быть такими:

- **fixed** – фон зафиксирован.
- **scroll** – фон разблокирован (прокручивается)
- **local** – рисунок прокручивается только с содержимым элемента, но дальше, если элемент уже закончился, не идет.

Если изображений несколько, для них можно указать правила, перечислив их в одном свойстве **background-attachment** через запятую:

```
background-attachment: fixed, scroll, local;
```

background-clip

Определяет, как фоновое изображение или цвет фона будут выводиться относительно границ элемента.

- **content-box** – фон отображается только под содержимым.
- **border-box** – выводится и под контентом, и под границами.
- **padding-box** – отображается внутри границ.

background-origin

То же, что и **background-clip**, с такими же значениями, только задает положение относительно границ не текущего элемента, а элемента-родителя.

- **content-box** – фон располагается относительно контента родительского элемента.
- **border-box** – фон позиционируется относительно границы родителя, при этом рамки могут фон перекрывать.
- padding-box – фон отображается до границ элемента-родителя, при этом границы на него не заходят.

background-size

Свойство указывает браузеру размеры фонового изображения. Их можно задать в любых единицах длины (через пробел указывается сначала ширина, потом высота картинки). Если прописано только одно значение, то оно определит ширину, высота при этом останется исходной, то есть такой, как в файле изображения. Чтобы указать только высоту, а ширину оставить оригинальной, свойству **background-size** нужно задать значения **auto** размер (*например,* **background-size: auto 300px**).

В правиле можно использовать еще два значения.

- **cover**. Масштабирует фон по размерам блока с сохранением исходных пропорций картинки, то есть рисунок не растянется и не станет слишком узким.
- **contain**. Помещает изображение внутрь блока, сохраняет пропорции.

background

Позволяет объединить значения перечисленных свойств в одной строке:

```
background: url("background.jpg") center center / 100px 100px no-repeat content-box;
```

В примере между свойствами **background-position** и **background-size** стоит слеш (/), которым их значения друг от друга необходимо отделять, чтобы не запутывать браузер.

3. БЛОКИ В CSS. CSS-ПОЗИЦИОНИРОВАНИЕ.

3.1. Блочная модель CSS

Блочная модель – это способ отображения элементов браузерами, которые обрабатывают все теги как небольшие прямоугольные блоки. Для браузеров любой элемент – это контейнер с содержимым: текстом, изображениями, другими тегами и т.д.

Выделяют две основные категории HTML-элементов, которые соответствуют типам их содержимого и поведению в структуре веб-страницы – ***блочные*** и ***строчные*** элементы. С помощью ***блочных*** элементов можно создавать структуру веб-страницы, ***строчные*** элементы используются для форматирования текстовых фрагментов.

В CSS модель документа представляется блоком. Каждый элемент в дереве элементов документа представляет собой самостоятельный блок. Причем, есть блоки, структурно отделенные от остальных, а есть строчные блоки, которые могут находиться внутри структурных блоков.

Блок имеет прямоугольную форму:

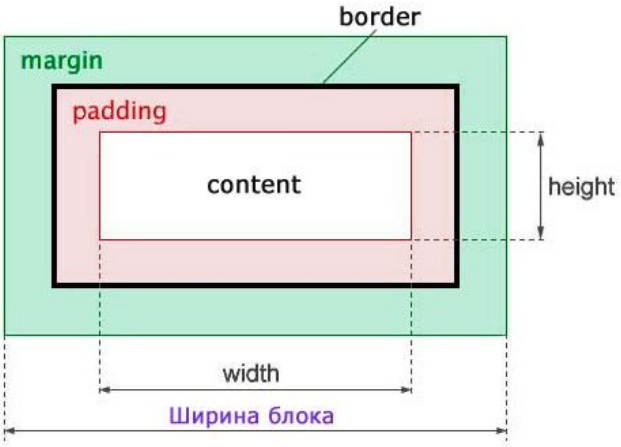


Рис. 1. Области и края блока

Ширина блока – это комплексная величина и складывается из нескольких значений свойств:

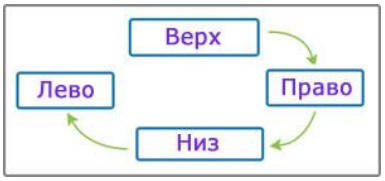
- **width** – ширина контента, т.е. содержимого блока;
- **padding-left** и **padding-right** – поле слева и справа от контента;
- **border-left** и **border-right** – толщина границы (рамки) слева и справа;
- **margin-left** и **margin-right** – отступ слева и справа.

Padding – устанавливает значение полей вокруг содержимого элемента (контента).

Margin – устанавливает отступ (расстояние) от границы текущего элемента до внутренней границы его родительского элемента.

Если у элемента нет родителя, отступом будет расстояние от края элемента до края окна браузера с учетом того, что у самого окна по умолчанию тоже установлены отступы. Чтобы от них избавиться, следует устанавливать значение **margin** для селектора **<body>** равное нулю.

Свойства **margin** и **padding** могут принимать от одного до четырех значений.

Число значений	Результат
значение1	Поля будут установлены одновременно от каждого края элемента.
значение1 значение2	Первое значение устанавливает поля от верхнего и нижнего края, второе - от левого и правого.
значение1 значение2 значение3	Первое значение задает поле от верхнего края, второе - одновременно от левого и правого края, а третье - от нижнего края.
значение1 значение2 значение3 значение4	Поочередно устанавливается поля от верхнего, правого, нижнего и левого края. 

padding-top	**значение**	Устанавливает значение поля от верхнего края содержимого элемента.
margin-top	**значение**	Устанавливает расстояние от верхнего края текущего элемента до внутренней границы его родительского элемента.
padding-left	**значение**	Устанавливает значение поля от левого края содержимого элемента.
margin-left	**значение**	Устанавливает расстояние от края левой границы текущего элемента до внутренней границы его родительского элемента.
padding-bottom	**значение**	Устанавливает значение поля от нижнего края содержимого элемента.
margin-bottom	**значение**	Устанавливает расстояние от края нижней границы текущего элемента до внутренней границы его родительского элемента.
padding-right	**значение**	Устанавливает значение поля от правого края содержимого элемента.
margin-right	**значение**	Устанавливает расстояние от правого края текущего элемента до внутренней границы его родительского элемента.

Ширина и высота элемента в CSS

По умолчанию **для блочных элементов** используется автоширина. Это означает, что элемент будет растянут по горизонтали ровно на столько, сколько есть свободного места. Высота блочных элементов по умолчанию устанавливается автоматически, т.е. браузер растягивает область содержимого в вертикальном направлении так, чтобы отобразилось все содержимое (контента). Чтобы установить собственные размеры для области содержимого элемента используются свойства **width** и **height**.

В CSS свойство **width** позволяет задать ширину области содержимого элемента, а свойство **height** высоту области содержимого.

Ширина (**width**) и высота (**height**) содержимого блока (контента) не включают ширину отступов, полей и толщину границы.

Значения высоты и ширины задаются в единицах, принятых в CSS (см. абсолютные и относительные единицы измерения).

Для свойств **height** и **width** значение можно указать в % относительно размеров родительского элемента или окна браузера. При значении **auto** высота или ширина блока автоматически подстраиваются под содержимое (значение по умолчанию).

Рис. 2. Ширина и высота элемента

Управление содержанием элемента

Содержимое, расположенное внутри элемента, может превысить указанный размер блока.

Для содержания блочного элемента, если оно целиком не помещается и выходит за область заданных размеров применяется свойство **overflow.**

Свойство **overflow** может иметь следующие значения:

visible – элемент растягивается до необходимых размеров (по умолчанию).

hidden – содержание элемента «обрезается», видна лишь та его часть, что помещается в элементе.

scroll – добавляются полосы прокрутки (всегда! даже если содержание помещается в пределах элемента).

auto – полосы прокрутки добавляются только при необходимости.

Границы в CSS

Параметр	Значение	Описание
border-top-color **border-right-color** **border-left-color** **border-bottom-color** **border-color**	**цвет** **нет цвета** (по умолчанию)	Параметр определяет **цвет** четырех сторон рамок. Если вместо цвета подставить URL-адрес изображения, оно будет повторяться, образуя рамку.
border-top-style **border-right-style** **border-left-style** **border-bottom-style** **border-style**	**none** (по умолчанию) **solid** **double** **groove** **ridge** **inset** **outset**	Определяют **стиль** четырех сторон рамок.
border-top **border-right** **border-left** **border-bottom** **border**	**ширина_рамки,** **стиль_рамки,** **цвет** (по умолчанию: **medium, none, нет цвета**)	Определяют свойства четырех сторон рамок. **ширина_рамки** может быть значением medium, thin или thick или задана в единицах измерения. **стиль_рамки** может иметь значение none или solid. Аргумент **color** используется для определения цвета заполнения фона элемента, пока он загружается, а также позади прозрачных частей элемента. Если передать вместо него адрес изображения, то рисунок будет повторяться, заполняя контур рамки.

border-top-width **border-right-width** **border-left-width** **border-bottom-width** **border-width**	**medium** (по умолчанию) **thin** **thick** **длина**	Определяют ширину рамки вокруг элемента. Каждая сторона может быть задана соответствующим параметром. Можно заменить установку четырех индивидуальных свойств установкой одного свойства **border-width** так же, как и для свойств отступа **margin**.
ciip	**rect** **auto** (по умолчанию)	Определяет, какая часть элемента видна (/top/right/bottom/left/). Все, что находится за пределами области, указанной этим параметром, увидеть нельзя.
display	**"", none**	Этот параметр указывает, будет ли отображен элемент.
float	**none** (по умолчанию) **left** **right**	Определяет обтекание элемента другими элементами слева или справа вместо помещения их под ними. Поддерживается тегами ** , ** ...
height	**auto** (по умолчанию) **длина**	Устанавливают высоту элемента и измеряют ее, если это необходимо. Значение возвращаются в виде строки, включающей единицы измерения (px, % и т.д.). Для получения числового значения используется свойство posHeight.
left	**auto** (по умолчанию) **длина** **проценты**	Задают горизонтальную координату элемента, позволяя корректно устанавливать и передвигать элементы. Значение возвращается как строка, включающая единицы измерения (px, % и т.д.). Для получения значения в виде числа используется свойство posLeft.

Стиль рамки

Значениями свойства **style** могут быть следующие ключевые слова:

none – граница отсутствует.

dotted – граница состоит из точек.

dashed – граница в виде пунктирной линии.

solid – граница отображается сплошной линией.

double – граница отображается двойной сплошной линией.

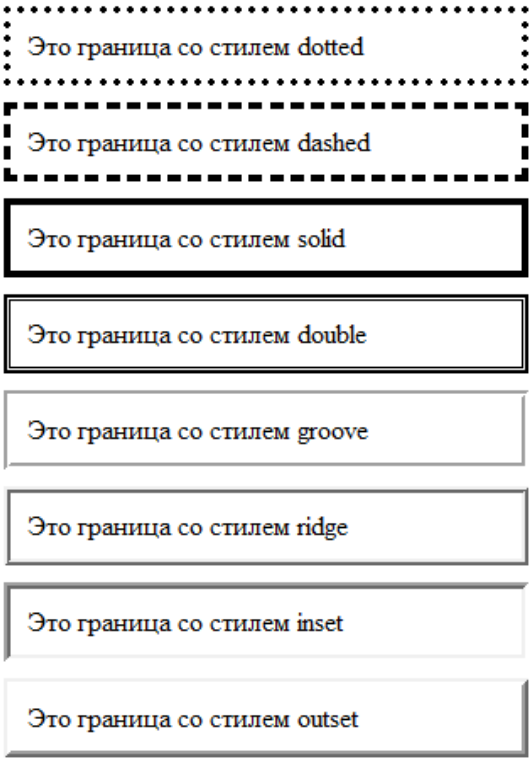
groove – граница отображается вдавленной объемной линией.

ridge – граница отображается выпуклой объемной линией.

inset – граница отображается так, что весь блок выглядит вдавленным.

outset – граница отображается так, что весь блок выглядит выпуклым.

В браузере разные стили границ выглядят так:

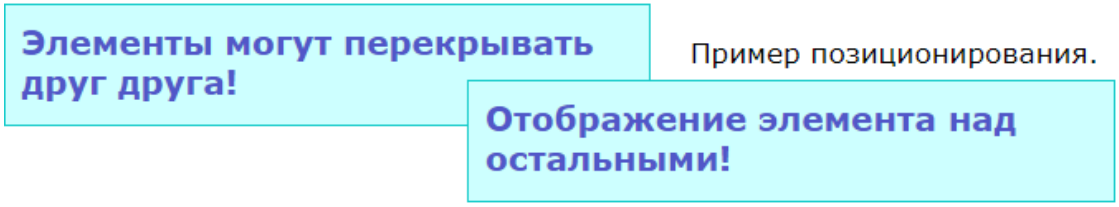


Иногда возможность задавать стили для разных секторов границы очень выручает, но чаще требуется задать единый стиль для всех границ и тогда удобнее пользоваться сокращенной записью **border**, в которой через пробел указываются: *толщина*, *тип* и *цвет* (именно в таком порядке).

3.2. Позиционирование блоков в CSS

Позиционирование – это управление местом расположения (позицией) элемента на веб-странице, оно контролируется с помощью CSS свойства **position**.

Для указания точного места расположения позиционированных элементов, используются CSS свойства: **top**, **right**, **bottom** и **left**. Они работают со всеми позиционированными элементами, кроме статических.



Поток документа

По умолчанию элементы на веб-странице отображаются в том порядке, в котором они представлены в HTML-документе, т. е. ***блочные элементы*** занимают всю доступную для них ширину и укладываются вертикально один под другим. ***Строчные элементы*** выстраиваются по горизонтали до тех пор, пока не будет занята вся доступная ширина, после того как вся ширина будет занята, будет сделан перенос строки и всё пойдет по новой. Такой порядок расположения элементов называется **нормальным потоком** (его также называют **потоком документа** или **общим потоком**).

С помощью свойства **float** или **position** можно убрать элемент из нормального потока. Если элемент "выпадает" из нормального потока, то элементы, которые расположены в коде ниже этого элемента будут смещены на его место на веб-странице.

Существуют следующие виды позиционирования блоков:

Абсолютное позиционирование. Элемент размещается по указанным координатам. Координаты отсчитываются от границ страницы или от границ внешнего блока, если этот блок тоже имеет позиционирование.

Относительное позиционирование. Элемент смещается относительно того места, которое он занимал в основном потоке документа. Другие элементы не занимают его место. Оно остаётся пустым, если его не перекроют позиционированные элементы.

Фиксированное позиционирование. Элемент размещается по координатам относительно границ окна браузера. При прокрутке страницы такой элемент не движется вместе со всеми элементами, а остаётся на одном месте в окне браузера.

Статическое позиционирование. Это расположение элемента в основном потоке документа. То есть, элемент находится там, где и должен находиться без позиционирования. Если ему заданы координаты, то они игнорируются.

Позиционирование устанавливает свойство **position**.

Оно принимает значения:

position: absolute; – абсолютное

position: relative; – относительное

position: fixed; – фиксированное

position: static; – статическое

Для указания координат используются свойства:

left – смещение левого края элемента относительно левой границы окна браузера или внешнего блока.

right – смещение правого края элемента относительно правой границы окна браузера или внешнего блока.

top – смещение верхнего края элемента относительно верхней границы окна браузера или внешнего блока.
bottom – смещение нижнего края элемента относительно нижней границы окна браузера или внешнего блока.

Эти свойства могут принимать следующие значения:
left: auto; – без смещения (по умолчанию)
left: величина в единицах измерения CSS
left: величина в процентах. Если элемент смещается относительно окна браузера, то проценты рассчитываются от размеров окна браузера. А если элемент смещается относительно внешнего блока, то проценты рассчитываются от размеров блока.
left: inherit; – значение принимается от родительского элемента

У остальных свойств значения указываются так же, как у **left**.

Перекрывающие элементы

Когда элементы находятся вне общего потока страницы, они могут перекрывать друг друга. Обычно порядок расположения элементов соответствует их порядку в HTML-коде страницы, однако есть возможность управлять порядком перекрытия с помощью CSS свойства **z-index**, чем больше его значение, тем выше будет элемент.

Это свойство может быть установлено только позиционированным элементам. Оно принимает значения:
z-index: auto; – координата Z не установлена (по умолчанию)
z-index: целое число
z-index: inherit – значение принимается от родительского элемента

Обтекание элементов

В CSS есть возможность установить обтекание элемента. При этом элемент смещается максимально к левому либо к правому краю страницы или внешнего блока. А элементы, следующие за ним, обтекают его.
Обтекание устанавливается с помощью свойства **float**. Значения свойства:
float: left; – элемент смещается влево, другие элементы обтекают его справа
float: right; – элемент смещается вправо, другие элементы обтекают его слева
float: none; – отмена обтекания
float: inherit; – значение принимается от родительского элемента

Пример использования свойства **float** – это обтекание картинки текстом. Выглядит это так:

Взрослый лев имеет вес в среднем до 250 килограмм при длине туловища около 2,5 м. Гибкое, подвижное тело льва обладает отлично развитой мускулатурой шеи и передних лап. Когти льва достигают в длинну 7 см. Массивная голова льва с вытянутой мордой имеет сильные челюсти. У льва 30 зубов, клыки размером до 8 см позволяют ему охотиться на достаточно крупных животных: косуль, кабанов, зебр и антилоп.

Чтобы сделать такое обтекание, картинку нужно поместить в абзац или другой блок и установить ей свойство **float**.

Элементы со свойством **float** имеют ряд особенностей:

- Не влияют на высоту внешнего контейнера
- Если размеры не установлены, то блоки имеют ширину по содержимому
- Если элементов несколько, то они располагаются в строку
- Блоки, следующие за элементом с **float**, не обращают на него внимания и занимают его место. А строковые элементы учитывают элемент с **float** и располагаются за ним.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. ПОДКЛЮЧЕНИЕ CSS К HTML

1. Создать html-страницу с кодом:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Подключение CSS к HTML</title>
  </head>
  <body>
    <h1>Это заголовок первого уровня</h1>
    Здесь просто текст
    <h2>Это заголовок второго уровня</h2>
    Здесь просто текст
  </body>
</html>
```

2. Создать страницу (пока пустую) и сохраните ее как style.css в папку с html-страницей. Это и будет страница стилей. Подключить страницу style.css к html-странице. Для этого добавить в html-страницу тег <link>, который и отвечает за подключение внешних файлов:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Подключение CSS к HTML</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>Это заголовок первого уровня</h1>
    Здесь просто текст
    <h2>Это заголовок второго уровня</h2>
    Здесь просто текст
  </body>
</html>
```

Внутренние таблицы стилей

Так называют таблицу стилей, заданную внутри элемента HTML, при помощи атрибута **style**. Пример кода:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Подключение CSS к HTML</title>
  </head>
  <body>
    <h1 style="color:red">Это заголовок первого уровня</h1>
    Здесь просто текст
    <h2>Это заголовок второго уровня</h2>
    Здесь просто текст
  </body>
</html>
```

Встроенные таблицы стилей

При этом способе таблица стилей встраивается в заголовок html-страницы. Для этого в HTML существуют теги **<style></style>**, с параметром **type**, который указывает, что подключается именно таблица стилей CSS.

Пример кода:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Подключение CSS к HTML</title>

    <style type="text/css">
    h1{
        color:red
    }
    </style>

  </head>
  <body>
    <h1>Этот заголовок будет красного цвета</h1>
    <h1>Этот заголовок будет красного цвета</h1>
    <h1>Этот заголовок будет красного цвета</h1>
  </body>
</html>
```

Все заголовки **h1** на странице будут красного цвета. А, если нужно, чтобы один из них был синего цвета, то воспользуемся для него внутренней таблицей стилей:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Подключение CSS к HTML</title>

    <style type="text/css">
    h1{
        color:red
    }
    </style>

  </head>
  <body>
    <h1>Этот заголовок будет красного цвета</h1>
    <h1 style="color:blue">Этот заголовок будет синего цвета</h1>
    <h1>Этот заголовок будет красного цвета</h1>
  </body>
</html>
```

В этом случае и применяется **принцип каскадирования**, который разрешит конфликт: в данном случае внутренняя таблица имеет высший приоритет, поэтому заголовок и станет синим.

Недостаток этого способа: таблицу стилей придется создавать для каждой страницы.

2. ПРАВИЛА И СЕЛЕКТОРЫ CSS

Правило CSS

Сам блок объявления стилей состоит из свойств и их значений, разделенных точкой с запятой:

1. Открыть css-страницу (файл style.css), задать странице голубой фон, за это отвечает тег **<body>**, для страницы style.css записать код:

```
body{
    background: blue;
}
```

Открыть html-страницу в браузере и убедиться, что ее фон стал синим.

2. Оформить текст на странице белым цветом:

```
body{
    background: blue;
    color: white;
}

```

Просмотреть результат.

3. Оформить цвета заголовков красным (для h1) и желтым (для h2):

```
body{
    background: blue;
    color: white;
}
h1{
    color:red;
}
h2{
    color:yellow;
}
```

Обновить html-страницу в браузере и просмотреть результат.

Селекторы CSS

Селекторы по идентификатору

В вышеуказанном примере в качестве селекторов использовались элементы страницы: **body**, **h1**, **h2**. Если на html-странице есть несколько одинаковых элементов (например, абзацев), и необходимо, чтобы текст всех абзацев был черным, а один из них - розовый. Тогда следует указать уникальный идентификатор для этого абзаца и создать для него стиль.

В HTML идентификатор элемента задается при помощи параметра **id**, в качестве значения которого указывается уникальное имя. Например:

```
<p id="pink">Текст абзаца с идентификатором (id)</p>
```

Имена можно давать любые, кроме зарезервированных слов (к ним относятся имена тегов и параметров элементов HTML и CSS).

1. Добавить на html-страницу пару абзацев и одному из них присвоить идентификатор:

```
<!DOCTYPE html>
<html>
  <head>
    <title>CSS id</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>Это заголовок первого уровня</h1>
    Здесь просто текст
    <h2>Это заголовок второго уровня</h2>
    Здесь просто текст
    <p>Простой абзац</p>
    <p id="pink">Текст абзаца с идентификатором (id)</p>
  </body>
</html>
```

На странице в браузере текст абзацев белого цвета.

2. Добавить в таблицу стилей (style.css) стили для абзацев:

```
body{
    background: blue;
    color: white;
}
h1{
    color:red;
}
h2{
    color:yellow;
}
p{
    color:black;
}
p#pink{
    color:pink;
}
```

Сначала указан черный цвет для всех абзацев, а цвет текста абзаца с **id "pink"** – розовый.

В данном случае селектор состоит из элемента (**p**), разделителя (#) и имени идентификатора (**pink**).

Важно помнить, что на странице может быть только один идентификатор (**id**). Т.е. для нашего примера нельзя создать два абзаца с **id "pink"**, абзац с таким **id** может быть только один. Но каждый абзац может иметь свой идентификатор, *например*, можно создать абзац с **id="green"** и задать стиль для этого абзаца в таблице стилей.

Селекторы по классу

В примере выше создан абзац с розовым цветом текста и указано, что такой **id** может быть только один. Но, если необходимо, чтобы розовый цвет текста был у двух или трех абзацев. Для этого в HTML существует параметр **class**, в качестве значения которого указывается его имя.

1. Добавить на html-страницу еще пару абзацев и присвоить им **class="pink"**:

```
<!DOCTYPE html>
<html>
  <head>
    <title>CSS class</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>Это заголовок первого уровня</h1>
    Здесь просто текст
    <h2>Это заголовок второго уровня</h2>
    Здесь просто текст
    <p>Простой абзац</p>
    <p id="pink">Абзац с идентификатором</p>
    <p class="pink">Абзац с классом (class) pink</p>
    <p class="pink">Абзац с классом (class) pink</p>
  </body>
</html>
```

Для того, чтобы указать стиль для этого класса, в таблице стилей указать правило, где в качестве селектора снова будет использоваться элемент **p** и имя **pink**, но в данном случае это имя класса, поэтому в качестве разделителя будет использоваться точка (**.**):

```
body{
    background: blue;
    color: white;
}
h1{
    color:red;
}
h2{
    color:yellow;
}
p{
    color:black;
}
p#pink{
    color:pink;
}
p.pink{
    color:pink;
}
```

Абзацев с таким классом может быть сколько угодно.

Таким образом:

- если все подобные элементы (например, все заголовки h1) должны иметь один стиль, то селектор состоит только из этого элемента *например*, **p{color:black;}**
- если элемент (любой: абзац, заголовок...) должен отличаться от всех остальных, то ему присваивается идентификатор (**id**) и разделителем в таблице стилей является знак решетки (#), *например*, **p#pink{color:pink;}**.
- если же на странице будет несколько элементов с одинаковым стилем, то им присваивается класс (**class**) и разделителем в таблице стилей является знак точки (**.**), *например*, **p.pink{color:pink;}**.
- идентификатор имеет более высокий приоритет, чем класс. Поэтому, если для какого-либо элемента будет указан и класс, и идентификатор (что не противоречит принципам CSS), то применен будет стиль идентификатора.

Идентификаторы и классы можно задавать любым элементам html. Но часто бывает так, что необходимо несколько разных элементов выделить одним стилем, например, зеленым цветом. В таком случае можно воспользоваться **унифицированным селектором**. В таких селекторах имя элемента не указывается, указываются точка или решетка, как признак класса или идентификатора и имя.

Например:

```
1  .red{
2      color:red;
3  }
4  #yellow{
5      color:yellow;
6  }
```

Таким образом, какому бы элементу мы не задали **class="red"** (заголовку, абзацу или ссылке), цвет текста у него станет красным. Одному элементу на странице (любому) можно задать **id="yellow"** и цвет текста этого элемента станет желтым.

Контекстный селектор

1. Создать html-страницу с кодом:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Селекторы по элементу</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <p>Этот текст находится в абзаце</p>
    Это просто текст.
    <i>Этот текст выделен курсивом</i>
    <p>Этот текст находится в абзаце, но <i>эта часть выделена курсивом</i></p>
  </body>
</html>
```

2. Оформить текст зеленым цветом, выделенный курсивом, для этого в таблицу стилей записать селектор по элементу:

```
i{
    color:green;
}
```

Страница в браузере:

Этот текст находится в абзаце

Это просто текст. *Этот текст выделен курсивом*

Этот текст находится в абзаце, но *эта часть выделена курсивом*

3. Оформить текст зеленым цветом, выделенный курсивом, только тот, который находится в абзацах. Для этого внести изменения в таблицу стилей:

```
p i{
    color:green;
}
```

Данный стиль применен к элементам **i**, которые находятся в элементах **p**. Названия элементов при этом отделяются пробелом. Такие селекторы называют ***контекстными***. Теперь страница в браузере должна выглядеть так:

Этот текст находится в абзаце

Это просто текст. *Этот текст выделен курсивом*

Этот текст находится в абзаце, но *эта часть выделена курсивом*

Группировка селекторов

Если блоки объявления стилей для нескольких селекторов совпадают (например, мы хотим, чтобы заголовки первых трех уровней были зеленого цвета), то их можно сгруппировать. Для этого селекторы, к которым будет применяться один стиль, нужно перечислить через запятую.

Пример:

```
1  h1, h2, h3{
2      color:green;
3  }
```

Если кроме цвета, необходимо задать заголовкам размер, тогда в таблицу стилей следует дописать:

```
1  h1, h2, h3{
2      color:green;
3  }
4  h1{
5      font-size:18px;
6  }
7  h2{
8      font-size:16px;
9  }
10 h3{
11     font-size:14px;
12 }
```

У заголовков будет применен указанный размер, но все они будут зеленого цвета.

Насчет группировок имеются разногласия. Одни считают правильным вышеприведенный код, т.к. группировка помогла избежать повтора одинакового свойства для трех элементов, это сокращает код.

Другие считают, что таким образом, ухудшается логичность кода. Так как, найдя селектор для заголовка h3, не сразу понятно, почему текст в нем зеленого цвета. Сторонники логики рекомендуют группировать только элементы, у которых блоки описаний полностью совпадают.

В общем, группировать или нет, дело вкуса. Но необходимо помнить о разных подходах, когда будете читать чужой код, например, в шаблонах.

3. ПСЕВДОЭЛЕМЕНТЫ И ПСЕВДОКЛАССЫ

Псевдоэлементы

CSS применяется к элементам HTML. Но есть несколько элементов, которых не существует в HTML, но они присутствуют на странице (первая буква слова и первая строка абзаца). Такие элементы и называют **псевдоэлементами**. Им можно задавать стиль, также, как и элементам HTML.

К **псевдоэлементам** относят:

- **first-letter** – задает стиль первой букве слова
- **first-line** – задает стиль первой строке абзаца

1. Ввести код html-страницы:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Заголовок документа</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <p>
    Это текст абзаца, первую букву этого абзаца выделить красным цветом.
    </p>
  </body>
</html>
```

2. Выделить первую букву абзаца красным цветом, для этого в таблице стилей написать:

```
p:first-letter{
    color:red;
}
```

3. Страница в браузере:

Это текст абзаца, первую букву этого абзаца выделить красным цветом.

4. Выделить первую строку абзаца синим цветом, для этого в таблице стилей написать:

```
p:first-letter{
    color:red;
}
p:first-line{
    color:blue;
}
```

5. Страница в браузере:

Это текст абзаца, первую букву этого абзаца выделить красным цветом.

Первая буква осталась красного цвета, т.к. для нее задан отдельный селектор.

Псевдоклассы

В CSS существуют **четыре псевдокласса**, они позволяют работать со **ссылками**. Как вы знаете, у ссылок есть четыре состояния: простая, активная, посещенная и та, на которую наведен курсор. Состояние ссылок зависит от действия пользователя, и браузер, в зависимости от этих действий может применять разные стили. Для описания этих стилей и существуют **псевдоклассы**.

- **a:link** – задает стиль обычной ссылки.
- **a:active** – задает стиль активной ссылки.
- **a:visited** – задает стиль посещенной ссылки.
- **a:hover** – задает стиль ссылки, на которую наведен курсор.

1. Код html-страницы со ссылкой:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Заголовок документа</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <a href="#">ссылка</a>
  </body>
</html>
```

Результат: ссылка

2. Сделать ссылку зеленого цвета и убрать подчеркивание:

```
a{
   color:green;
   text-decoration:none;
}
```

Свойство **text-decoration** отвечает как раз за подчеркивание, а его значение **none** указывает, что подчеркивать не надо.

Результат: ссылка

3. Задать стиль для ссылки, на которую наведен курсор. Пусть она становится красного цвета:

```
a{
   color:green;
   text-decoration:none;
}
a:hover{
   color:red;
}
```

Обновить страницу в браузере и посмотреть результат.

4. СВОЙСТВА CSS: ЦВЕТ И ФОН

Цвет – color

Свойство цвета – **color** (задает цвет текста внутри элемента), является наследуемым.

1. Ввести код html-страницы:

```
<!DOCTYPE html>
<html>
  <head>
    <title>css color (цвет)</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>Заголовок</h1>
    <h2>Заголовок</h2>
    <p>Здесь текст абзаца.</p>
    Здесь просто текст.
  </body>
</html>
```

2. Задать стиль для элемента body:

```
body{
    color:green;
}
```

Теперь весь текст на странице зеленого цвета. Если требуется изменить цвет какого-либо элемента, то ему нужно будет задать свой стиль.

3. Сделать цвет заголовка красным:

```
body{
    color:green;
}
h1{
    color:red;
}
```

Теперь цвет заголовка красный, т.к. мы задали ему свойство **color**, а до этого он, как и все остальные элементы, был зеленым, потому что унаследовал свойство **color** от своего "предка" – элемента **body**.

Таким образом, если у элемента не задано свойство **color**, то оно наследуется от элемента "предка". Если оно не задано и для предка, то будет осуществлен поиск вверх по дереву элементов, пока не будет найден элемент, для которого это свойство задано.

Дерево элементов – структура всех элементов html-страницы, отражающая их вложенность друг в друга.

Схематично дерево элементов для нашего примера выглядит так:

В нашем примере для элементов **h1** и **p** предком является элемент **body**, для которого предком является элемент **html**. Это и есть **принцип наследования**.

Значениями свойства **color** могут быть **именные цвета** (red, blue...), **шестнадцатеричные коды цветов** (#FF0000) и **десятичные коды цвета в модели RGB** (rgb(255, 0, 0)).

Итак, задать цвет текста для элемента можно тремя способами:

```
1  body{
2      color:green;
3  }
4  h1{
5      color:#FF0000;
6  }
7  h2{
8      color:rgb(255,0,0);
9  }
```

Фон – background

Группа свойств, так или иначе связанная с фоном. При помощи CSS фон можно задать не только странице, но и заголовку, и абзацу, и любому другому элементу.

1. Ввести код html-страницы:

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>css background (фон)</title>
5      <link rel="stylesheet" type="text/css" href="style.css">
6    </head>
7    <body>
8      Здесь содержимое документа
9    </body>
10 </html>
```

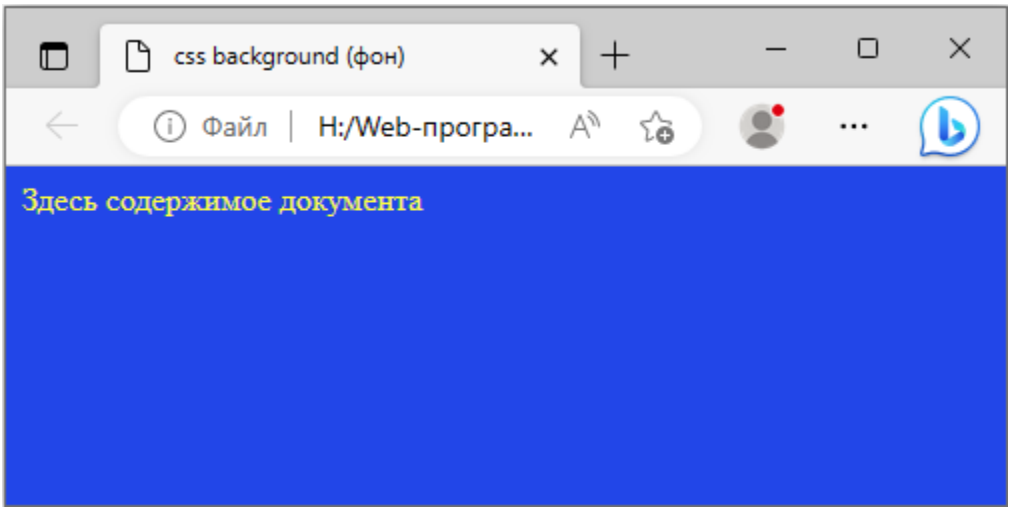
Рассмотрим группу свойств **background**:

- **background-color** – задает **цвет фона**. По умолчанию не наследуется, но его можно сделать наследуемым, если в качестве значения указать значение **inherit**.

Пример:

```
1  body{
2      background-color:#243CED;
3      color:yellow;
4  }
```

Страница в браузере:



- **background-image** – задает **фоновое изображение**. Значением свойства является URL графического файла. Формат задания следующий: сначала идет обозначение функции url, а затем в круглых скобках указывается путь к файлу. Путь к файлу указывается относительно таблицы стилей.

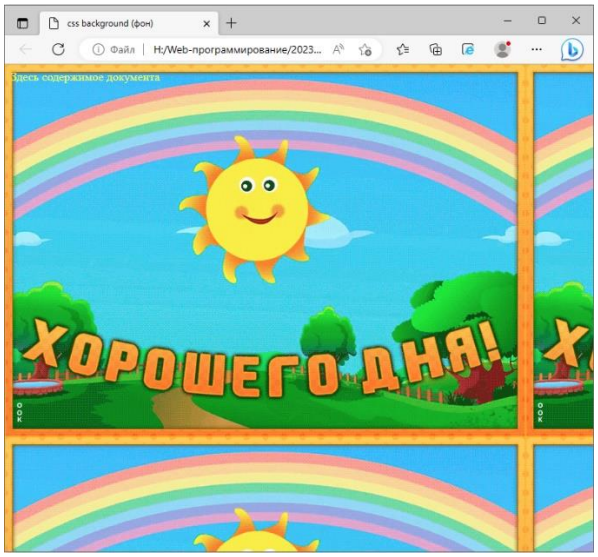
Пример:

```
1  body{
2      background-image:url(den.gif);
3      background-color:#243CED;
4      color:yellow;
5  }
```

Задано оба свойства: **background-image** и **background-color**, это рекомендуется делать на случай, если фоновое изображение по тем или иным причинам окажется недоступным. При задании обоих свойств фоновое изображение будет лежать поверх фонового цвета.

Страница стилей style.css лежит в той же папке, что и изображение den.gif.

Страница в браузере:



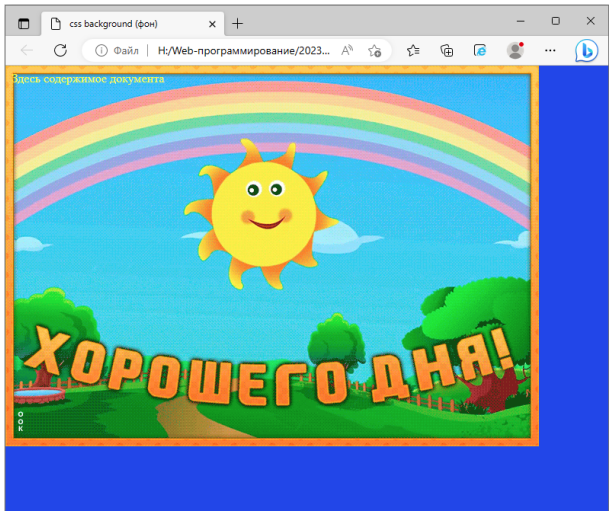
- **background-repeat** – задает **возможность повторения фонового изображения**. В качестве фонового изображения может выступать как цельное изображение (например, шапка сайта), так и маленькое изображение, которое должно замостить собой все пространство элемента. Данное свойство как раз и указывает, повторять ли изображение и, если да, то, как именно повторять. Возможны 4 варианта:
  repeat – повторять изображение по горизонтали и вертикали.
  repeat-x – повторять изображение только по горизонтали.
  repeat-y – повторять изображение только по вертикали.
  no-repeat – не повторять изображение.

По умолчанию используется значение **repeat**, как мы и убедились в предыдущем примере.

Страница стилей style.css в случае не повторять изображение:

```
1  body{
2      background-image:url(den.gif);
3      background-repeat:no-repeat;
4      background-color:#243CED;
5      color:yellow;
6  }
```

Страница в браузере:



Рассмотреть **самостоятельно**: **repeat-x**, **repeat-y** (файл изображения можно использовать любой).

- **background-attachment** – указывает, должно ли **фоновое изображение прокручиваться вместе с текстом элемента** или оно должно оставаться неподвижным. Может принимать два значения:

scroll – фон прокручивается вместе с текстом. Это значение используется по умолчанию.

fixed – фоновое изображение фиксируется относительно окна браузера, и во время прокрутки текста оно остается неподвижным.

Пример:

```
body{
    background-image:url(den.gif);
    background-repeat:no-repeat;
    background-attachment:fixed;
    background-color:#243CED;
    color:yellow;
}
```

Результат действия значения **fixed** непривычно для пользователя, поэтому применяйте его только когда это действительно обоснованно.

- **background-position** – задает **расположение элемента относительно окна браузера**. Значения можно задавать в процентах, в единицах длины и при помощи ключевых слов. Рассмотрим на примерах:

```
body{
    background-image:url(den.gif);
    background-repeat:no-repeat;
    background-color:#243CED;
    background-position:10% 30%;
    color:yellow;
}
```

Левый верхний угол изображения будет смещен на 10% от ширины окна по горизонтали и на 30% от высоты окна по вертикали.

Следует заметить, что это свойство по-разному воспринимается разными браузерами (**проверить!**).

Выполнить:

1. Левый верхний угол изображения сместить на 50 пикселей (**50px**) по горизонтали и на 50 пикселей (**50px**) по вертикали.
2. Изображение выровнять по горизонтали – по центру, а по вертикали – по верхнему краю страницы.

Для **выравнивания по горизонтали** (*первый параметр*) можно использовать следующие ключевые слова: **left** (по левому краю), **center** (по центру) и **right** (по правому краю). Для **выравнивания по вертикали** (*второй параметр*): **top** (по верхнему краю), **center** (по центру) и **bottom** (по нижнему краю).

Сокращенная запись свойства background

В CSS для многих свойств существует сокращенная запись. В этом случае значения всех свойств перечисляются через пробел в произвольном порядке.

Пример:

```
1  body{
2      background:url(den.gif) no-repeat #33CCFF center top;
3      color:yellow;
4  }
```

5. СВОЙСТВА CSS: СТИЛЕВЫЕ ПАРАМЕТРЫ ШРИФТА И ТЕКСТА

Свойства CSS: шрифты

Для задания параметров шрифтов в CSS используется свойство **font**.

font-family

Это свойство CSS задает **гарнитуру шрифта**. Все шрифты можно условно разделить на несколько групп:

serif – шрифты с засечками, например, Times New Roman.

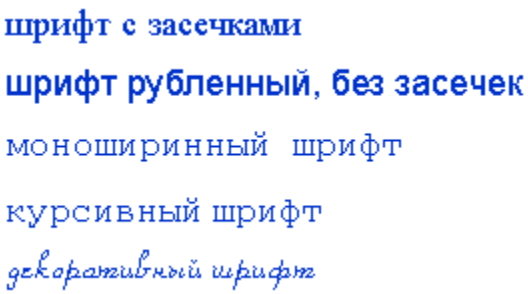
sans-serif – шрифты рубленные, без засечек, например, Arial.

monospace – моноширинные шрифты, например, Courier New.

cursive – курсивные шрифты, например, Calisto MT.

fantasy – декоративные шрифты, например, Torhok.

Примеры написания разными группами шрифтов:



В качестве значения свойства **font-family** можно указать шрифт и группу шрифтов.

Пример:

```
1  body{
2      font-family: Verdana, sans-serif;
3  }
```

Весь текст страницы будет написан шрифтом Verdana, но если на компьютере пользователя такого шрифта не окажется, то будет использоваться любой другой из группы **sans-serif**. То есть будет подобран шрифт наиболее близкий ему по виду, и внешний вид страницы будет не очень отличаться от задуманного.

Можно указать несколько шрифтов через запятую, в порядке убывания приоритета.

font-style

Это свойство CSS задает **стиль шрифта**: **normal** (обычный), **oblique** (наклонный), **italic** (курсивный).

1. Ввести код html-страницы с тремя абзацами, каждому задать уникальный идентификатор:

```
<!DOCTYPE html>
<html>
  <head>
    <title>css шрифты</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <p id="sn">Текст в абзаце с идентификатором sn.</p>
    <p id="so">Текст в абзаце с идентификатором so.</p>
    <p id="si">Текст в абзаце с идентификатором si.</p>
  </body>
</html>
```

2. Задать в таблице стилей шрифт для всех абзацев:

```
p#sn, p#so, p#si{
   font-family: Verdana, sans-serif;
}
```

Страница в браузере:

Текст в абзаце с идентификатором sn.

Текст в абзаце с идентификатором so.

Текст в абзаце с идентификатором si.

3. Задать каждому абзацу свой стиль:

```
p#sn, p#so, p#si{
   font-family: Verdana, sans-serif;
}
p#sn{
   font-style:normal;
}
p#so{
   font-style:oblique;
}
p#si{
   font-style:italic;
}
```

Страница в браузере теперь выглядит так:

Текст в абзаце с идентификатором sn.

Текст в абзаце с идентификатором so.

Текст в абзаце с идентификатором si.

Чем отличается наклонный стиль от курсивного?

Дело в том, что различные шрифты включают в себя различные варианты начертания (**bold**, **italic**, **regular** и другие). Но, если в какой-либо шрифт не включен вариант **italic**, то он имитируется простым наклоном букв, что и соответствует значению **oblique**.

По умолчанию используется стиль **normal**.

font-variant

Это свойство CSS задает **вариант написания букв** из двух возможных: **normal** (*обычный*) и **small-caps** (*малые прописные буквы*). По умолчанию это свойство имеет значение **normal**.

1. Изменить таблицу стилей предыдущего примера на такую:

```
p#sn, p#so, p#si{
   font-family: Verdana, sans-serif;
}
p#so{
   font-variant:small-caps;
}
```

2. Результат просмотреть в браузере.

font-weight

Это свойство CSS задает **толщину букв шрифта**. В качестве значений выступают числа: 100, 200, 300, 400, 500, 600, 700, 800 и 900. А также ключевые слова: **normal** (*нормальный*), **bold** (*полужирный*), **bolder** (*более жирный по отношению к базовому, унаследованному от предка*) и **lighter** (*менее жирный по отношению к базовому, унаследованному от предка*).

При этом значению **normal** соответствует числовое значение 400, а значению **bold** – 700.

1. Задать свойство **font-weight** второму абзацу:

```
p#sn, p#so, p#si{
   font-family: Verdana, sans-serif;
}
p#so{
   font-weight:bold;
}
```

Страница в браузере:

Текст в абзаце с идентификатором sn.

Текст в абзаце с идентификатором so.

Текст в абзаце с идентификатором si.

font-size

Это свойство CSS задает **размер шрифта**. Задавать размер в CSS можно тремя способами:

– с помощью *ключевых слов* (xx-small, x-small, small, medium, large, x-large, xx-large, smaller, large),

– с помощью *относительных единиц* (% и em),
– с помощью *единиц измерения длины* (пикселы, пункты, сантиметры и миллиметры).

Использовать ключевые слова пока не рекомендуется, так как разные браузеры по-разному их отображают.

С помощью относительных единиц задаются размеры относительно элемента-предка.

Существует три ***относительных единицы измерения***:

px – равен одной точке на экране, называемой пикселем

em – равен высоте шрифта, используемого в данном элементе

ex – равен высоте строчной буквы "x" шрифта, используемого в данном элементе

И пять ***абсолютных единиц измерения***:

in – равен 1 дюйму (2,54 см)

pt – равен 1/72 дюйма

pc – равен 1/6 дюйма

mm – равен 1 миллиметру

sm – равен 1 сантиметру

Абсолютные единицы имеют фиксированный размер, а относительные устанавливают размер относительно какой-то другой единицы.

Для web-страниц лучше использовать только три единицы измерения:

pt – для "фиксированного" дизайна сайта

% – для "резинового" дизайна

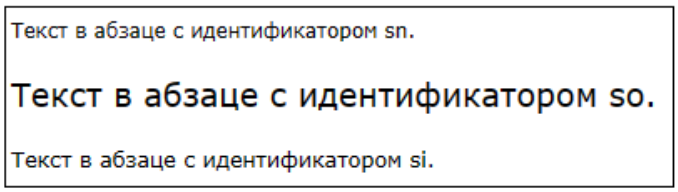
em – для пропорционального изменения размера

1. Задать для текста абзацев размер в 12 пикселей, для второго абзаца увеличить его на 20%, а для третьего – уменьшить на 10% от базового (т.е. от 12 пикселей):

```
1  p#sn, p#so, p#si{
2      font-family:  Verdana, sans-serif;
3      font-size:12px;
4  }
5  p#so{
6      font-size:1.2em;
7  }
8  p#si{
9      font-size:0.8em;
10 }
```

У величины **em** в качестве разделителя выступает точка.

Страница в браузере:



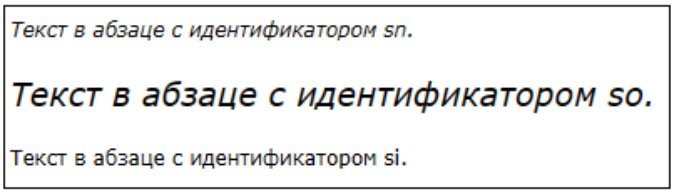
Сокращенная запись свойства font

В этом случае значения всех свойств перечисляются через пробел в следующем порядке: **font-style**, **font-variant**, **font-weight**, **font-size**, **font-family**. Причем, любое из свойств, кроме **font-size** и **font-family**, можно не указывать.

1. Задать абзацам следующие свойства:

```
p#sn, p#so, p#si{
    font:italic 12px Verdana, sans-serif;
}
p#so{
    font-size:1.2em;
}
p#si{
    font-style:normal;
}
```

Таким образом, сначала сокращенной записью заданы значения свойства **font** для всех абзацев, а затем заданы отличия для второго и третьего абзацев. Теперь страница выглядит так:



Свойства CSS: текст

Свойства текста позволяют задавать параметры слов и предложений, а также их взаимное расположение.

text-decoration

Это свойство CSS отвечает за **оформление текста**. Может принимать следующие значения:

none – у текста нет оформления

underline – текст подчеркивается

overline – текст надчеркивается линией, расположенной над текстом

line-through – текст отображается зачеркнутым

blink – текст становится мигающим.

Достаточно распространенный прием убрать подчеркивание у ссылок, а при наведении курсора подчеркивание будет снова появляться. Для этого в таблице стилей нужно написать:

```
a{
    text-decoration:none;
}
a:hover{
    text-decoration:underline;
}
```

text-align

Это свойство CSS отвечает за **горизонтальное выравнивание текста**. Может принимать следующие значения:

left – выравнивание по левому краю

center – выравнивание по центру

right – выравнивание по правому краю

justify – выравнивание по ширине

Пример кода:

```
a{
    text-decoration:none;
    text-align:center;
}
a:hover{
    text-decoration:underline;
}
```

text-indent

Это свойство CSS отвечает за **отступы в абзацах**. Отступы задаются в **единицах измерения** и **%**, могут быть как **положительными** (*красная строка*), так и **отрицательными** (*висячая строка*).

1. Ввести код html-страницы с тремя абзацами:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Заголовок документа</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <p>Текст в абзаце, где первая строка будет с отступом.
    Текст в абзаце, где первая строка будет с отступом.</p>
    <p>Текст в абзаце, где первая строка будет с отступом.
    Текст в абзаце, где первая строка будет с отступом.</p>
    <p>Текст в абзаце, где первая строка будет с отступом.
    Текст в абзаце, где первая строка будет с отступом.</p>
  </body>
</html>
```

2. Добавить для абзацев свойство **text-indent**, чтобы задать красную строку:

```
p{
    text-indent:1.2em;
}
```

Здесь лучше использовать единицу **em**, чтобы отступ был пропорционален размеру шрифта.

Страница в браузере:

Текст в абзаце, где первая строка будет с отступом. Текст в абзаце, где первая строка будет с отступом.

Текст в абзаце, где первая строка будет с отступом. Текст в абзаце, где первая строка будет с отступом.

Текст в абзаце, где первая строка будет с отступом. Текст в абзаце, где первая строка будет с отступом.

text-transform

Это свойство CSS отвечает за **видоизменение текста**, точнее за смену регистра. Используются следующие значения:

capitalize – меняет первую букву каждого слова на заглавную

uppercase – меняет все буквы на заглавные

lowercase – меняет все буквы на строчные

none - изменений не происходит

1. Ввести код html-страницы и присвоить каждому абзацу идентификатор:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Заголовок документа</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <p id="sc">Текст в абзаце со свойством capitalize.</p>
    <p id="su">Текст в абзаце со свойством uppercase.</p>
    <p id="sl">Текст в абзаце со свойством lowercase.</p>
  </body>
</html>
```

2. Изменить таблицу стилей, задав для каждого абзаца свойство **text-transform**:

```
p#sc{
    text-transform:capitalize;
}
p#su{
    text-transform:uppercase;
}
p#sl{
    text-transform:lowercase;
}
```

Страница в браузере:

Текст В Абзаце Со Свойством Capitalize.

ТЕКСТ В АБЗАЦЕ СО СВОЙСТВОМ UPPERCASE.

текст в абзаце со свойством lowercase.

Интервалы

В CSS есть несколько свойств, регулирующих **расстояния между словами, буквами в словах** и **строками.**

word-spacing – задает интервал между словами

letter-spacing – задает интервал между буквами

line-height – задает интервал между строками

Значения этих свойств задаются в **единицах измерения** и **%.**

1. Ввести код html-страницы:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Заголовок документа</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <p id="sc">Текст в абзаце с интервалами по умолчанию.</p>
    <p id="su">Текст в абзаце со свойством word-spacing.</p>
    <p id="sl">Текст в абзаце со свойством letter-spacing.</p>
  </body>
</html>
```

2. Изменить таблицу стилей:

```
p#su{
   word-spacing:10px;
}
p#sl{
   letter-spacing:5px;
}
```

Страница в браузере:

Текст в абзаце с интервалами по умолчанию.

Текст в абзаце со свойством word-spacing.

Текст в абзаце со свойством letter-spacing.

3. Увеличить интервал между строками абзаца, для этого добавить в таблицу стилей свойство **line-height**:

```
p{
   line-height:200%
}
p#su{
   word-spacing:10px;
}
p#sl{
   letter-spacing:5px;
}
```

Страница в браузере:

Текст в абзаце с интервалами по умолчанию.

Текст в абзаце со свойством word-spacing.

Текст в абзаце со свойством letter-spacing.

Использование свойств текста

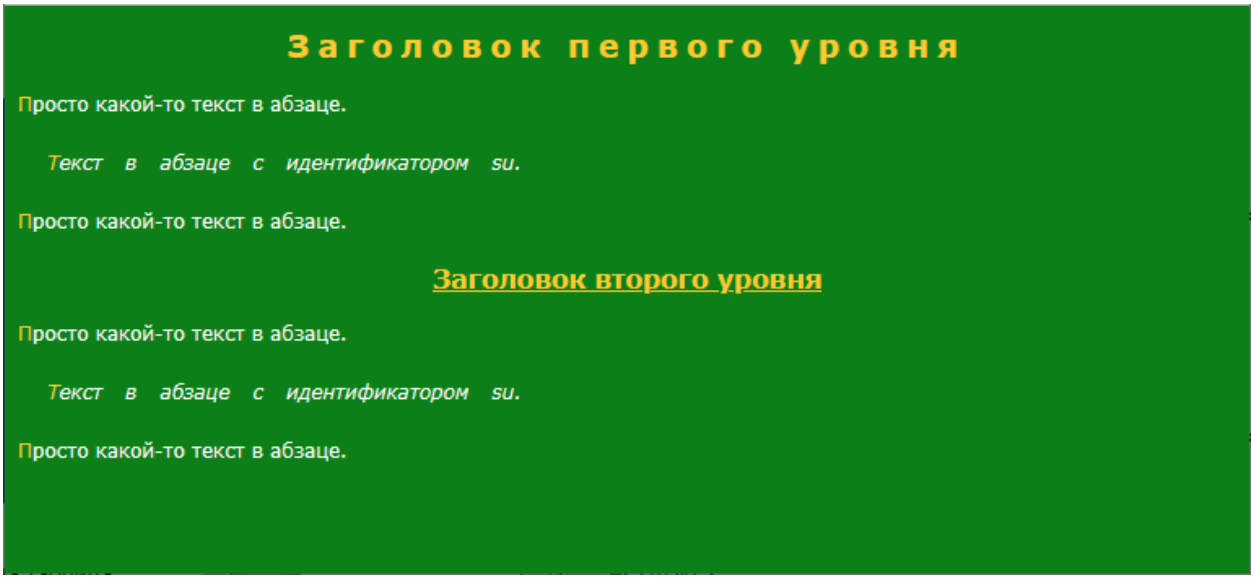
1. Ввести код html-страницы:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Заголовок документа</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>

    <h1>Заголовок первого уровня</h1>

    <p>Просто какой-то текст в абзаце.</p>

    <p id="su">Текст в абзаце с идентификатором su.</p>

    <p>Просто какой-то текст в абзаце.</p>

    <h2>Заголовок второго уровня</h2>

    <p>Просто какой-то текст в абзаце.</p>

    <p id="su">Текст в абзаце с идентификатором su.</p>

    <p>Просто какой-то текст в абзаце.</p>

  </body>
</html>
```

2. Задать для этой страницы следующее оформление:

```
body{
    background:green;
    color:#FFFFFF;
    font:12px Verdana, sans-serif;
}
h1{
    font-size:1.6em;
    color:#FFCC00;
    letter-spacing:5px;
    text-align:center;
}
h2{
    font-size:1.3em;
    color:#FFCC00;
    text-decoration:underline;
    text-align:center;
}
p:first-letter{
    color:#FFCC00;
}
p#su{
    text-indent:1.5em;
    word-spacing:10px;
    font-style:italic;
}
```

Страница в браузере:



6. СВОЙСТВА CSS: СТИЛЕВЫЕ ПАРАМЕТРЫ СПИСКОВ

Для оформления списков существует три свойства:

- **list-style-type** – определяет внешний вид маркера или номера
- **list-style-image** – определяет пользовательское изображение маркера
- **list-style-position** – определяет положение маркеров относительно блока

list-style-type

Для маркированных списков используются те же значения, что и в HTML:

- **disc** – закрашенный круг
- **circle** – незакрашенный круг
- **square** – закрашенный квадрат

1. Использование **встроенных стилей CSS** внутри маркированного списка:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Пример изменения типа маркера маркированного списка</title>
    </head>
    <body>
        <ul style = "list-style-type:none">  <!-- маркер отсутствует -->
            <li>Первый пункт</li>
            <li>Второй пункт</li>
            <li>Третий пункт</li>
        </ul>
        <ul style = "list-style-type:disc">  <!-- маленький черный круг -->
            <li>Первый пункт</li>
            <li>Второй пункт</li>
            <li>Третий пункт</li>
        </ul>
        <ul style = "list-style-type:circle">  <!-- круг пустой внутри -->
            <li>Первый пункт</li>
            <li>Второй пункт</li>
            <li>Третий пункт</li>
        </ul>
        <ul style = "list-style-type:square">  <!-- маркер в форме квадрата -->
            <li>Первый пункт</li>
            <li>Второй пункт</li>
            <li>Третий пункт</li>
        </ul>
    </body>
</html>
```

Результат просмотреть в браузере.

2. Использование **внешних стилей CSS** внутри маркированного списка:

```html
<!DOCTYPE html>
<html>
  <head>
    <title>CSS списки</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <ul id="spisok1">
      <li>Первый пункт</li>
      <li>Второй пункт</li>
      <li>Третий пункт</li>
    </ul>
    <ul id="spisok2">
      <li>Первый пункт</li>
      <li>Второй пункт</li>
      <li>Третий пункт</li>
    </ul>
    <ul id="spisok3">
      <li>Первый пункт</li>
      <li>Второй пункт</li>
      <li>Третий пункт</li>
    </ul>
  </body>
</html>
```

3. Код страницы CSS:

```css
#spisok1{
  list-style-type:disk;
}
#spisok2{
  list-style-type:circle;
}
#spisok3{
  list-style-type:square;
}
```

Результат просмотреть в браузере.

Для нумерованных списков перечень шире, чем в HTML, но не все значения поддерживаются всеми браузерами. Поэтому рекомендуется использовать только следующие:

- **decimal** – десятичные числа
- **lower-roman** – строчные римские цифры
- **upper-roman** – прописные римские цифры
- **lower-alpha** – строчные латинские буквы

1. Создать четыре одинаковых списка, но каждому в стилях задать свое значение свойства.

Код html-страницы:

```
<!DOCTYPE html>
<html>
  <head>
    <title>CSS списки</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <ol id="spisok1">
      <li>Первый пункт</li>
      <li>Второй пункт</li>
      <li>Третий пункт</li>
    </ol>
    <ol id="spisok2">
      <li>Первый пункт</li>
      <li>Второй пункт</li>
      <li>Третий пункт</li>
    </ol>
    <ol id="spisok3">
      <li>Первый пункт</li>
      <li>Второй пункт</li>
      <li>Третий пункт</li>
    </ol>
    <ol id="spisok4">
      <li>Первый пункт</li>
      <li>Второй пункт</li>
      <li>Третий пункт</li>
    </ol>
  </body>
</html>
```

Код страницы style.css:

```
#spisok1{
  list-style-type:decimal;
}
#spisok2{
  list-style-type:lower-roman;
}
#spisok3{
  list-style-type:upper-roman;
}
#spisok4{
  list-style-type:lower-alpha;
}
```

Результат просмотреть в браузере.

list-style-image

Это свойство позволяет задать **свой вид маркера**.

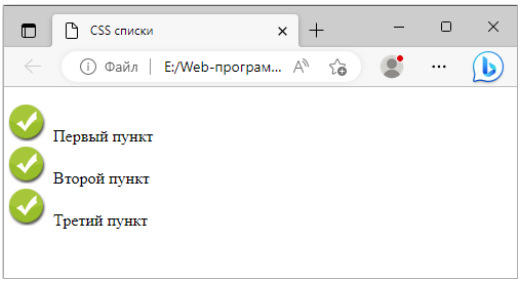
1. Создать список. Код html-страницы:

```
<!DOCTYPE html>
<html>
  <head>
    <title>CSS списки</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <ul id="spisok">
      <li>Первый пункт</li>
      <li>Второй пункт</li>
      <li>Третий пункт</li>
    </ul>
  </body>
</html>
```

Код страницы CSS:

```css
#spisok{
list-style-image:url(marker.png);
}
```

Картинку в виде маркера использовать свою.
Результат:



На внешний вид списка будет влиять размер картинки.

list-style-position

Это свойство определяет **положение маркера**: внутри блока – **inside** или снаружи – **outside**.

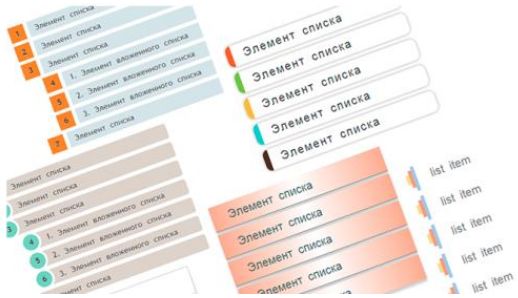
1. Создать два списка, расположенных в **div**:

```html
<!DOCTYPE html>
<html>
  <head>
    <title>CSS списки</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <div id="sp">

      <ul id="spisok1">
       <li>Первый пункт первого списка</li><br>
       <li>Второй пункт первого списка</li><br>
       <li>Третий пункт первого списка</li><br>
      </ul>

      <ul id="spisok2">
       <li>Первый пункт второго списка</li><br>
       <li>Второй пункт второго списка</li><br>
       <li>Третий пункт второго списка</li><br>
      </ul>

   </div>
  </body>
</html>
```

2. Задать стили для списков с разными значениями:

```css
#sp{
  width:150px;
}
#spisok1{
  list-style-position:inside;
}
#spisok2{
  list-style-position:outside;
}
```

Результат просмотреть в браузере.



Примеры красивого оформления списков

1. Создать любой список из предложенных вариантов
2. Сделать к CSS-коду несколько комментариев

Вариант 1.

1. Элемент списка
2. Элемент списка
3. Элемент списка
4. Элемент списка
5. Элемент списка

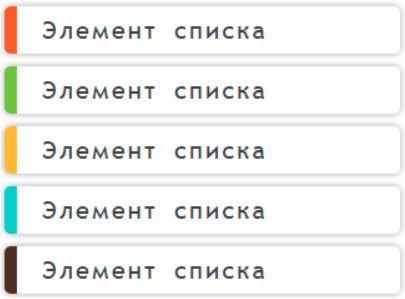
HTML

```
<ol class="rounded">
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
</ol>
```

CSS

```
.rounded {
counter-reset: li;
list-style: none;
font: 14px "Trebuchet MS", "Lucida Sans";
padding: 0;
text-shadow: 0 1px 0 rgba(255,255,255,.5);
}
.rounded a {
position: relative;
display: block;
padding: .4em .4em .4em 2em;
margin: .5em 0;
background: #DAD2CA;
color: #444;
text-decoration: none;
border-radius: .3em;
transition: .3s ease-out;
}
.rounded a:hover {background: #E9E4E0;}
.rounded a:hover:before {transform: rotate(360deg
.rounded a:before {
content: counter(li);
counter-increment: li;
position: absolute;
left: -1.3em;
top: 50%;
margin-top: -1.3em;
background: #8FD4C1;
height: 2em;
width: 2em;
line-height: 2em;
border: .3em solid white;
text-align: center;
font-weight: bold;
border-radius: 2em;
transition: all .3s ease-out;
}
```

Вариант 2.



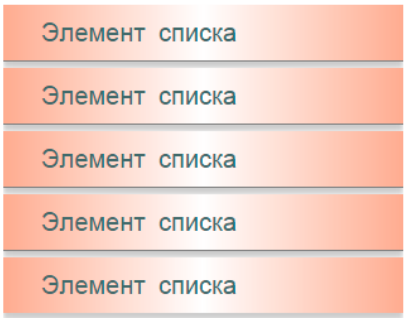
HTML

```
<ul class="border">
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
</ul>
```

CSS

```
.border {
list-style: none;
padding: 0;
}
.border li {
font-family: "Trebuchet MS", "Lucida Sans";
padding: 7px 20px;
margin-bottom: 10px;
border-radius: 5px;
border-left: 10px solid #f05d22;
box-shadow: 2px -2px 5px 0 rgba(0,0,0,.1),
     -2px -2px 5px 0 rgba(0,0,0,.1),
    2px 2px 5px 0 rgba(0,0,0,.1),
    -2px 2px 5px 0 rgba(0,0,0,.1);
font-size: 20px;
letter-spacing: 2px;
transition: 0.3s all linear;
}
.border li:nth-child(2){border-color: #8bc63e;}
.border li:nth-child(3){border-color: #fcba30;}
.border li:nth-child(4){border-color: #1ccfc9;}
.border li:nth-child(5){border-color: #493224;}
.border li:hover {border-left: 10px solid transparent;}
.border li:nth-child(1):hover {border-right: 10px solid #f05d22;}
.border li:nth-child(2):hover {border-right: 10px solid #8bc63e;}
.border li:nth-child(3):hover {border-right: 10px solid #fcba30;}
.border li:nth-child(4):hover {border-right: 10px solid #1ccfc9;}
.border li:nth-child(5):hover {border-right: 10px solid #493224;}
```

Вариант 3.



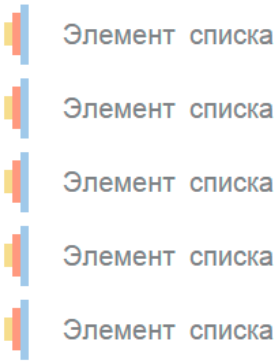
HTML

```
<ul class="first">
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
</ul>
```

CSS

```
.first  {
list-style: none;
padding: 0;
}
.first li {
padding: 10px 30px;
background: linear-gradient(to left, #f8ab8d 0%, white, #f8ab8d
border-bottom: 1px solid grey;
color: #506a6b;
font-size: 20px;
box-shadow: 0 5px 5px 0 rgba(0,0,0, .2);
margin-bottom: 5px;
}
.first li:last-child {border-bottom: none;}
```

Вариант 4.



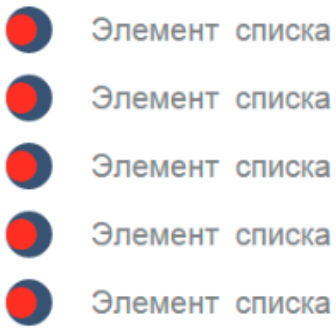
HTML

```
<ul class="dbl-border">
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
</ul>
```

CSS

```
.dbl-border {
list-style: none;
margin: 0;
}
.dbl-border li {
margin: 10px 0;
position: relative;
}
.dbl-border a {
width: 100%;
color: #808285;
text-decoration: none;
border-left: 6px solid #ABC7EA;
display: block;
padding-left: 25px;
height: 44px;
line-height: 44px;
font-size: 20px;
position: relative;
transition: 0.3s linear;
}
.dbl-border a:before {
content: "";
width: 6px;
height: 70%;
background: #EE997C;
position: absolute;
top: 15%;
left: -12px;
}
.dbl-border li:before {
content: "";
width: 6px;
height: 40%;
background: #EFDD89;
position: absolute;
top: 30%;
left: -12px;
}
.dbl-border a:hover {background: #D4
```

Вариант 5.



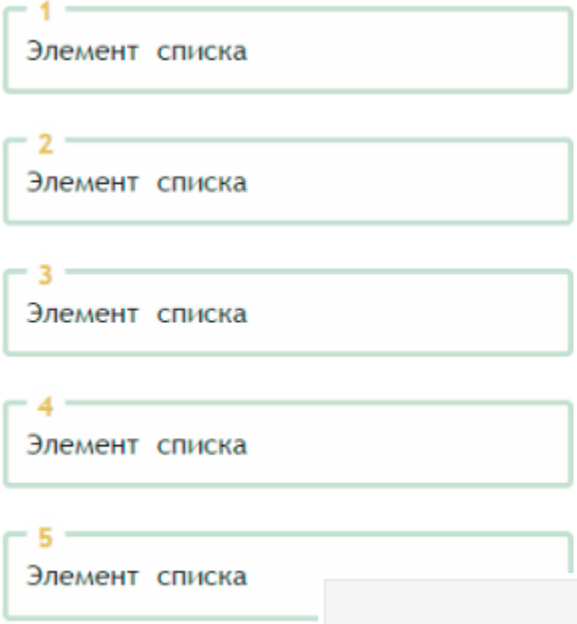
HTML

```
<ol class="ball">
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
</ol>
```

CSS

```
.ball {
list-style: none;
margin: 0;
}
.ball a {
width: 100%;
color: #808285;
text-decoration: none;
display: inline-block;
padding-left: 25px;
height: 44px;
line-height: 44px;
font-size: 20px;
position: relative;
transition: .3s linear;
}
.ball a:before {
content: "";
width: 30px;
height: 30px;
border-radius: 50%;
background: #425273;
position: absolute;
left: -30px;
top: 7px;
}
.ball li {position: relative;}
.ball li:before {
content: "";
width: 20px;
height: 20px;
border-radius: 50%;
background: #EC351D;
position: absolute;
top: 12px;
left: -30px;
z-index: 2;
transition: .4s ease-in-out;
}
.ball li:hover:before {left: -20
```

Вариант 6.



HTML

```
<ol class="bullet">
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
</ol>
```

CSS

```
.bullet {
margin-left: 0;
list-style: none;
counter-reset: li;
}
.bullet li {
position: relative;
margin-bottom: 1.5em;
border: 3px solid #CADFCF;
padding: 0.6em;
border-radius: 4px;
background: #FEFEFE;
color: #231F20;
font-family: "Trebuchet MS", "Lucida Sans";
}
.bullet li:before {
position: absolute;
top: -0.7em;
padding-left: 0.4em;
padding-right: 0.4em;
font-size: 16px;
font-weight: bold;
color: #DCC24B;
background: #FEFEFE;
border-radius: 50%;
counter-increment: li;
content: counter(li);
}
```

Вариант 7.

HTML

```
<ol class="pills">
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
</ol>
```

CSS

```
.pills {
  margin-left: 0;
  list-style: none;
  counter-reset: li;
  font-family: calibri;
}
.pills li {
  padding: 10px 0;
  position: relative;
  left: 1.5em;
  margin-bottom: 0.75em;
  padding-left: 1em;
  background: #E3DEDC;
}
.pills li:before {
  padding: 10px 0;
  position: absolute;
  top: 0;
  bottom: 0;
  left: -1.5em;
  width: 1.875em;
  text-align: center;
  color: white;
  font-weight: bold;
  background: #D66786;
  border-bottom-left-radius: 70em;
  border-top-left-radius: 70em;
  counter-increment: li;
  content: counter(li);
}
```

Вариант 8.

HTML

```
<ul class="push">
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
  <li>Элемент списка</li>
</ul>
```

CSS

```
.push {
  list-style: none;
  font-family: "Lucida Sans";
}
.push li {
  position: relative;
  padding: 20px 0 20px 40px;
  color: #D29D25;
  font-variant: small-caps;
  font-weight: bold;
  cursor: pointer;
}
.push li:before {
  position: absolute;
  width: 7px;
  height: 7px;
  border-radius: 50%;
  background: #4F5151;
  content: "";
  left: 0;
  transition: .3s ease-in-out;
  top: 27px;
}
.push li:after {
  position: absolute;
  border-left: 1px dotted #4F5151;
  width: 1px;
  bottom: -12px;
  content: "";
  left: 3px;
  top: 48px;
}
.push li:hover:before{box-shadow: 0 0 0 10px rgba(0,0,0,.2)}
.push li:last-child:after {content: none;}
```

Вариант 9.

1.04.15	День смеха
4.04.15	День веб-мастера
5.04.15	День геолога
7.04.15	День рождения РУНЕТА

HTML

```
<dl class="holiday">
  <dt>1.04.15</dt>
    <dd>День смеха</dd>
  <dt>4.04.15</dt>
    <dd>День веб-мастера</dd>
  <dt>5.04.15</dt>
    <dd>День геолога</dd>
  <dt>7.04.15</dt>
    <dd>День рождения РУНЕТА</dd>
</dl>
```

CSS

```
.holiday {
  overflow: hidden;
  font-size: 16px;
}
.holiday dt, .holiday dd {
  height: 2.5em;
  line-height: 2.5em;
  padding: 0 0.625em 0 0.875em;
  color: #4C565C;
  box-sizing: border-box;
}
dt {
  width: 30%;
  float: left;
  clear: right;
  background: #D3E6DD;
  font-weight: bold;
}
dd {
  width: 70%;
  float: right;
  margin-left: 0;
  margin-bottom: .3125em;
  border: 1px solid #BECFC7;
  border-left: none;
}
```

Вариант 10.

HTML

```
<ul class="older">
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
  <li><a href="#">Элемент списка</a></li>
</ul>
```

CSS

```
@import url(http://fonts.googleapis.com/css?family=Fredericka+the+Great|EB+Garamond);
.older {
  list-style: none;
  margin: 0 auto;
  width: 660px;
  counter-reset: li;
}
.older li {
  border-bottom: dashed 1px #006699;
  margin-top: 10px;
  box-shadow: 0 1px 4px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1) inset;
  padding: 5px;
}
.older a {
  text-decoration: none;
  padding: 10px;
  display: block;
  line-height: 30px;
  color: #3A3A3A;
  font-family: 'EB Garamond', serif;
  font-size: 20px;
}
.older a:before {
  display: inline-block;
  content: counter(li);
  counter-increment: li;
  height: 30px;
  width: 30px;
  text-align: center;
  border: solid 1px #dedede;
  margin-right: 20px;
  box-shadow: 0 1px 4px rgba(0, 0, 0, 0.3), 0 0 30px rgba(0, 0, 0, 0.1) inset;
  font-family: 'Fredericka the Great', cursive;
  font-size: 24px;
  -webkit-transition: .4s linear;
  transition: .4s linear;
}
.older a:hover:before {
  color:#D72F2C;
  -webkit-transform: scale(1.1);
  transform: scale(1.1);
}
```

7. СВОЙСТВА CSS: СТИЛЕВЫЕ ПАРАМЕТРЫ ТАБЛИЦЫ

1. Создать таблицу в HTML документе любого размера
2. Применить к таблице стиль

Границы таблицы border

Таблица и ячейки внутри нее по-умолчанию отображаются в браузере без видимых границ. Границы таблицы задаются свойством **border**:

```
table {
    border-collapse: collapse;
    /*убираем пустые промежутки между ячейками*/
    border: 1px solid grey;
    /*устанавливаем для таблицы внешнюю границу серого цвета толщиной 1px*/
}
```

Границы ячеек заголовка каждого столбца задаются для элемента **th**:

```
th {
    border: 1px solid grey;
}
```

Границы ячеек тела таблицы задаются для элемента **td**:

```
td {
    border: 1px solid grey;
}
```

Толщина рамок соседних ячеек не удваивается, поэтому задать границы для всей таблицы можно следующим способом:

```
th, td {
    border: 1px solid grey;
}
```

Внешнюю границу таблицы можно выделить, задав ей увеличенную ширину:

```
table {
    border: 3px solid grey;
}
```

Границы можно задавать частично:

```
/* устанавливаем для таблицы внешнюю границу серого цвета толщиной 3px */
table {
    border-top: 3px solid grey;
}
/* задаём для ячейки тела таблицы границу серого цвета толщиной 1px */
td {
    border-bottom: 1px solid grey;
}
```

Ширина и высота таблицы

По умолчанию **ширина и высота таблицы** определяется содержимым ее ячеек. Если ширина не задана, то она будет равна ширине самого широкого ряда (строки).

Ширина таблицы и столбцов задается с помощью свойства **width**. Если для таблицы задано **table {width: 100%;}**, то ширина таблицы будет равна ширине блока-контейнера, в котором она находится.

Ширину таблицы и столбцов обычно задают в **px** или **%**, *например*:

```
table {
    width: 600px;
}
th {
    width: 20%;
}
td:first-child {
    width: 30%;
}
```

Высота таблицы не задается. Высотой рядов таблицы можно управлять, добавив верхний и нижний **padding** для элементов **<td>** и **<th>**.

Фиксировать высоту с помощью свойства **height** не рекомендуется.

```
th, td {
    padding: 10px 15px;
}
```

Задать фон таблицы

По умолчанию фон таблицы и ячеек прозрачный. Если страница или блок, содержащие таблицу, имеют фон, то он будет просвечиваться сквозь таблицу. Если фон задан и для таблицы и для ячеек, то в местах наложения фона таблицы и ячеек будет виден фон только ячеек. В качестве фона для таблицы в целом и ее ячеек могут выступать:

- заливка сплошным цветом,
- градиентная заливка,
- фоновое изображение.

Заголовок в таблице

Добавить заголовок в таблицу можно с помощью элемента **<caption>**, а с помощью свойства **caption-side** его можно поместить перед таблицей или под ней. Для горизонтального выравнивания текста заголовка применяется свойство **text-align**.

Свойство наследуется.

caption-side	
Значения:	
top	Заголовок таблицы располагается над таблицей. Значение по умолчанию.
bottom	Располагает заголовок под таблицей.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

HTML-код

```
<table>
    <caption>Таблица № 1</caption>
        <tr>
            <th>Company</th>
            <th>Q1</th>
            <th>Q2</th>
            <th>Q3</th>
            <th>Q4</th>
        </tr>
...
</table>
```

CSS-код

```
caption {
    caption-side: bottom;
    text-align: right;
    padding: 10px 0;
    font-size: 14px;
}
```

Пример отображения заголовка под таблицей

Company	Q1	Q2	Q3	Q4
Microsoft	20.3	30.5	23.5	40.3
Google	50.2	40.63	45.23	39.3
Apple	25.4	30.2	33.3	36.7
IBM	20.4	15.6	22.3	29.3

Таблица № 1

Как убрать промежуток между рамками ячеек

Рамки ячеек таблицы по умолчанию разделены небольшим промежутком. Если задать для таблицы **border-collapse: collapse**, то промежуток уберется.

Свойство наследуется.

border-collapse	
Значения:	
separate	Рамки ячеек располагаются раздельно.
collapse	Рамки ячеек сливаются в одну, а промежутки между рамками убираются.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

CSS-код

```
7  table {
8      border-collapse: collapse;
9  }
```

Пример таблиц со сливающимися и раздельными рамками ячеек

Company	Q1	Q2	Q3
Microsoft	20.3	30.5	23.5
Google	50.2	40.63	45.23
Apple	25.4	30.2	33.3
IBM	20.4	15.6	22.3

Company	Q1	Q2	Q3
Microsoft	20.3	30.5	23.5
Google	50.2	40.63	45.23
Apple	25.4	30.2	33.3
IBM	20.4	15.6	22.3

Как увеличить промежуток между рамками ячеек

С помощью свойства **border-spacing** можно менять расстояние между рамками ячеек. Данное свойство применяется к таблице в целом.
Свойство наследуется.

border-spacing	
Значения:	
<длина> <длина>	Добавляет промежутки между рамками как по вертикали, так и по горизонтали. Если заданы две длины, то первая всегда определяет горизонтальный промежуток, а вторая — вертикальный.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

CSS-код

```
table {
    border-collapse: separate;
    border-spacing: 10px 20px;
}
table {
    border-collapse: separate;
    border-spacing: 10px;
}
```

Пример таблиц с увеличенными промежутками между рамками ячеек

Company	Q1	Q2	Q3
Microsoft	20.3	30.5	23.5
Google	50.2	40.63	45.23
Apple	25.4	30.2	33.3
IBM	20.4	15.6	22.3

Company	Q1	Q2	Q3
Microsoft	20.3	30.5	23.5
Google	50.2	40.63	45.23
Apple	25.4	30.2	33.3
IBM	20.4	15.6	22.3

Как скрыть пустые ячейки таблицы

Свойство **empty-cells** скрывает или показывает пустые ячейки. Действует только на ячейки, которые не содержат какой-либо контент. Если для ячейки задан фон, а для таблицы задано **table {border-collapse: collapse;}**, то ячейка не будет скрыта.

Свойство наследуется.

empty-cells	
Значения:	
show	Рамка и фон пустой ячейки будут отрисовываться так же, как для ячейки таблицы, имеющей содержимое.
hide	Если все ячейки строки пусты, то вся строка отображается так, если бы было задано значение display: none.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Пример скрытия пустой ячейки таблицы

Код HTML и CSS

```html
<table>
<tr>
  <th>Company</th>
  <th>Q1</th>
  <th>Q2</th>
  <th>Q3</th>
  </tr>
<tr>
  <td>Microsoft</td>
  <td>20.3</td>
  <td>30.5</td>
  <td></td>
</tr>
<tr>
  <td>Google</td>
  <td>50.2</td>
  <td>40.63</td>
  <td>45.23</td>
</tr>
</table>
```

Company	Q1	Q2	Q3
Microsoft	20.3	30.5	
Google	50.2	40.63	45.23

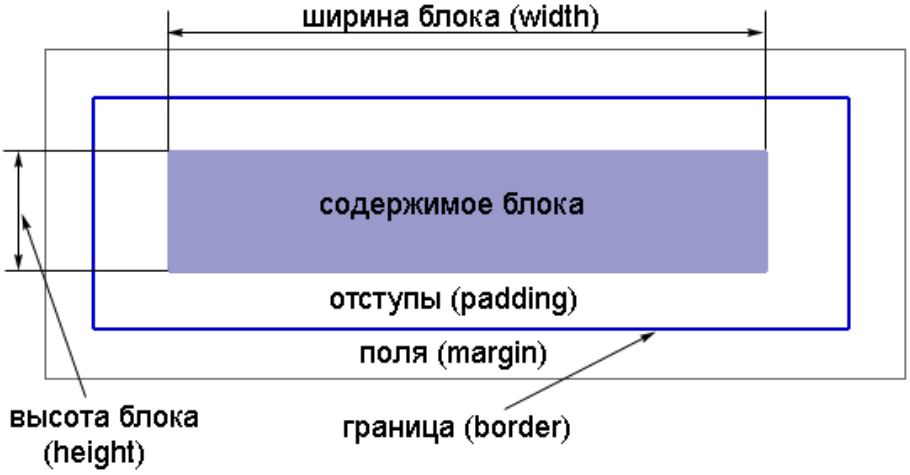
```css
table {
    border: 1px solid #69c;
    border-collapse: separate;
    empty-cells: hide;
}
th, td {
    border: 2px solid #69c;
}
```

Шпаргалка по работе с таблицами
https://html5book.ru/shpargalka-po-tablicam/

8. БЛОКИ В CSS

В CSS модель документа представляется **блоком**. Каждый элемент в дереве элементов документа представляет собой самостоятельный блок. Причем, есть блоки, структурно отделенные от остальных, а есть строчные блоки, которые могут находиться внутри структурных блоков.

Блок имеет прямоугольную форму:



У блока есть содержимое, например, для элемента **p** – это текст. Вокруг содержимого есть отступы (**padding**), они служат для того, чтобы текст не примыкал вплотную к границе блока. Фон отступов такой же, как и у содержимого.

Затем идет граница блока (**border**), которая может быть как видимой, так и невидимой.

Также блок имеет поля (**margin**), которые задают дополнительное свободное пространство вокруг блока. Фон полей прозрачный, т.е. сквозь него просвечивает фон родительского элемента.

Размер блока, т.е. его ширина (**width**) и высота (**height**), определяются содержимым, поля и отступы не учитываются в размере блока.

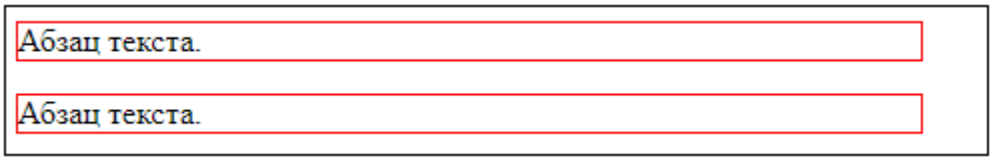
1. Создать HTML-страницу с двумя абзацами:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Заголовок документа</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <p>Абзац текста.</p>
    <p>Абзац текста.</p>
  </body>
</html>
```

2. Задать границу для абзацев (для того, чтобы увидеть отступы, поля и границы):

```
p{
    border:1px solid red;
}
```

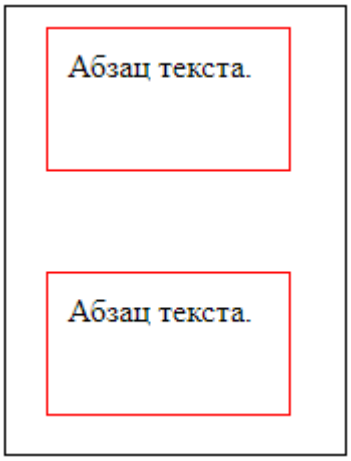
Страница в браузере:



3. Задать отступы от границы, используя свойство **padding**, поля (свойство **margin**), которые оделяют блоки друг от друга и размеры блоков для абзацев:

```
p{
    border:1px solid red;
    padding:10px;
    margin:50px;
    width:100px;
    height:50px;
}
```

Результат:



Свойства блоков

div – этот элемент является контейнером для остальных. Элемент div отделяется от остальных элементов абзацными отступами. Элемент **span**, в отличии от div, создает строчный блок.

1. Создать html-страницу со следующим кодом:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Заголовок документа</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>

    <div id="first">Это текст в тегах div с id="first".
    </div>

    <div id="second">
       <div id="third">Это текст в тегах div с id="third".</div>
       <div id="fourth">Это текст в тегах div с id="fourth".</div>
    </div>

  </body>
</html>
```

Страница в браузере:

Это текст в тегах div с id="first".
Это текст в тегах div с id="third".
Это текст в тегах div с id="fourth".

Содержимое тегов **div** располагается с абзацным отступом, т.е. одно под другим. На примере этой страницы рассмотрим свойства блоков.

border (граница)

Границы в CSS можно задавать отдельно для каждой стороны:

- **border-top** – верхняя граница
- **border-right** – правая граница
- **border-bottom** – нижняя граница
- **border-left** – левая граница

Каждый **сегмент границы** может иметь свои характеристики: **цвет**, **толщину** и **тип линии**. Для этого к свойству границы через дефис необходимо дописать ключевые слова: **color** (для цвета), **width** (для толщины) и **style** (для типа линии).

Например, **border-top-color** определяет цвет верхней границы, а **border-left-style** – тип линии для левой границы.

Если все четыре границы будут иметь одинаковые значения, то следует воспользоваться сокращенной записью:

- **border-color** – цвет всех границ
- **border-width** – толщина всех границ
- **border-style** – стиль всех границ

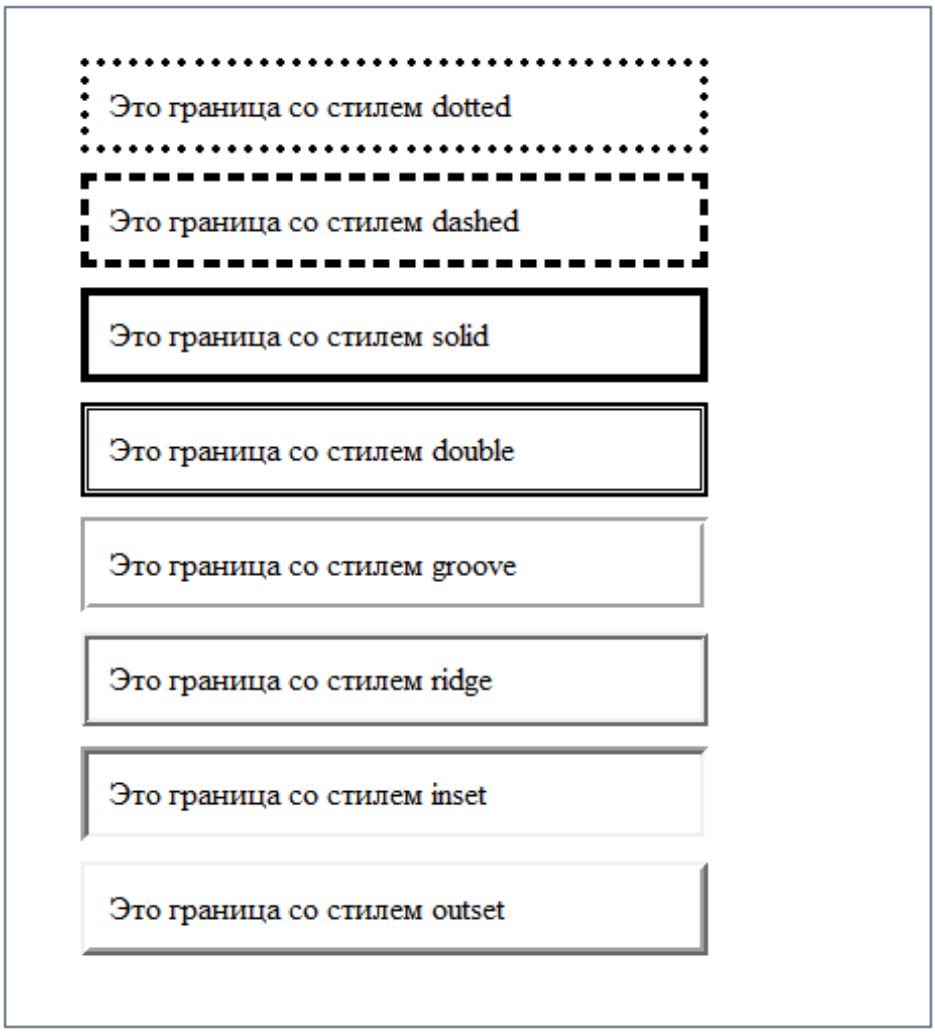
Значениями свойства **color** могут быть **именные цвета** (red, blue и т.д.), **шестнадцатеричные коды цветов** (типа #FFCCFF) и **десятичные коды в модели RGB** (типа rgb(255, 0, 0)).

Значениями свойства **width** могут быть ключевые слова: **thin** (тонкая граница), **medium** (средняя граница) и **thick** (толстая граница). А также любая единица измерения.

Значениями свойства **style** могут быть следующие ключевые слова:

- **none** – граница отсутствует
- **dotted** – граница состоит из точек
- **dashed** – граница в виде пунктирной линии
- **solid** – граница отображается сплошной линией
- **double** – граница отображается двойной сплошной линией
- **groove** – граница отображается вдавленной объемной линией
- **ridge** – граница отображается выпуклой объемной линией
- **inset** – граница отображается так, что весь блок выглядит вдавленным
- **outset** – граница отображается так, что весь блок выглядит выпуклым

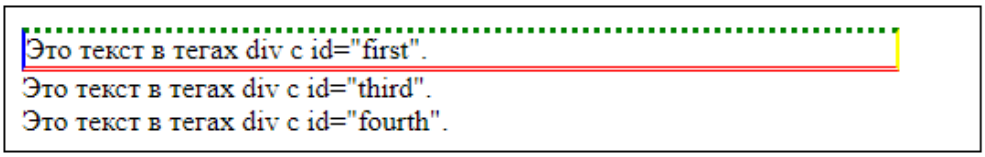
В браузере разные стили границ выглядят так:



2. Задать первому **div** разные границы:

```
#first{
    border-bottom-style:double;
    border-bottom-color:red;
    border-left-style:solid;
    border-left-width:2px;
    border-left-color:blue;
    border-right-style:solid;
    border-right-width:2px;
    border-right-color:yellow;
    border-top-style:dotted;
    border-top-color:green;
}
```

Страница в браузере:

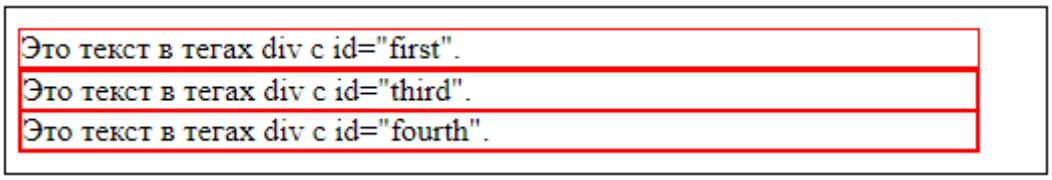


Чтобы задать единый стиль для всех границ удобнее пользоваться сокращенной записью **border**, в которой через пробел указываются: **толщина**, **тип** и **цвет** (именно в таком порядке).

3. Задать для всех элементов созданной страницы один стиль границ:

```
#first, #second, #third, #fourth{
    border: 1px solid red;
}
```

Страница в браузере:



В результате произошло наложение границ разных элементов друг на друга. Это происходит потому, что еще не заданы поля элементов.

margin (поля)

Поля задают свободное пространство вокруг элемента. Как и границы, поля в CSS можно определять отдельно для верхней, правой, нижней и левой сторон. Для этого используются свойства:

- **margin-top** – ширина верхнего поля
- **margin-right** – ширина правого поля
- **margin-bottom** – ширина нижнего поля
- **margin-left** – ширина левого поля

Чаще используется сокращенная запись **margin**, где через пробел указываются **ширина верхнего**, **правого**, **нижнего** и **левого полей**. Причем, именно в таком порядке.

Пример:

```
p{
    margin:5px 10px 15px 10px;
}
```

Если значения верхнего и нижнего полей совпадают, и значения правого и левого полей совпадают, то сокращенная запись выглядит еще короче:

```
p{
    margin:5px 10px;
}
```

В данном случае ширина верхнего и нижнего полей будет 5 пикселей, а правого и левого – по 10 пикселей.

Если же у всех полей ширина одинакова, то сокращенная запись выглядит так:

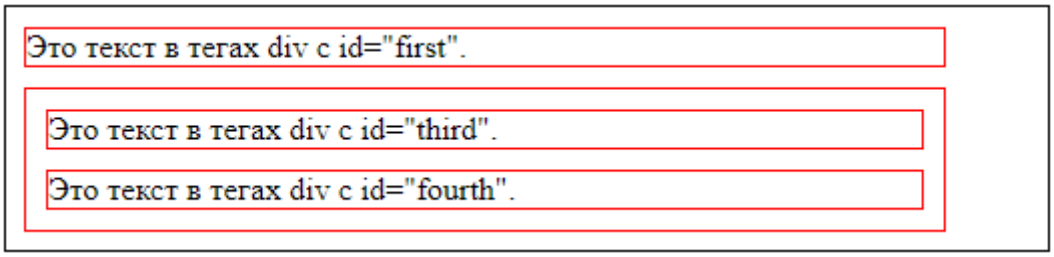
```
p{
    margin:5px;
}
```

Значения полей можно задавать и в других единицах длины, и в процентах. Также величина значения может иметь отрицательное значение, что в некоторых случаях очень удобно использовать.

4. Задать всем элементам одинаковую ширину полей - в 10 пикселей:

```
#first, #second, #third, #fourth{
    border: 1px solid red;
    margin:10px;
}
```

Страница в браузере:



В результате текст в самих элементах прижат к границам, чтобы это исправить, следует задать ему отступы.

padding (отступы)

Отступы позволяют отделить содержимое блока от границы. Параметры отступов в CSS можно задать для каждой стороны отдельно, используя следующие свойства:

padding-top – ширина верхнего отступа.

padding-right – ширина правого отступа.
padding-bottom – ширина нижнего отступа.
padding-left – ширина левого отступа.

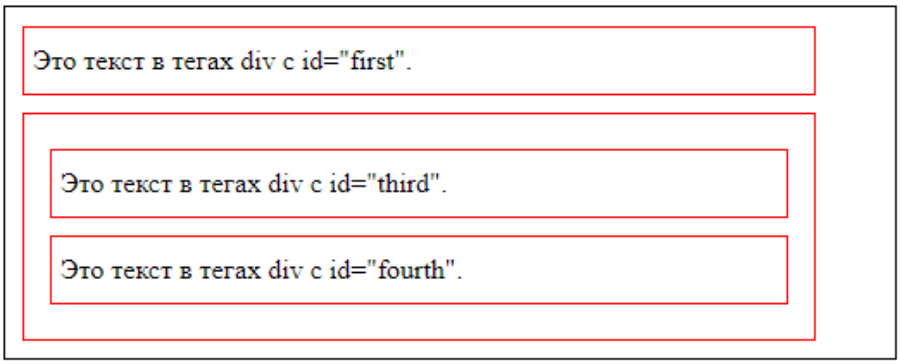
Значения свойств могут задаваться в различных единицах длины или в процентах. Проценты вычисляются относительно ширины блока. В качестве значения может выступать только положительная величина.

Как и с полями чаще удобнее использовать сокращенную запись, которая аналогична записи для полей.

5. Задать для элементов страницы отступы: сверху и снизу – по 10 пикселов, а справа и слева – по 5 пикселов.

```
#first, #second, #third, #fourth{
    border: 1px solid red;
    margin:10px;
    padding:10px 5px;
}
```

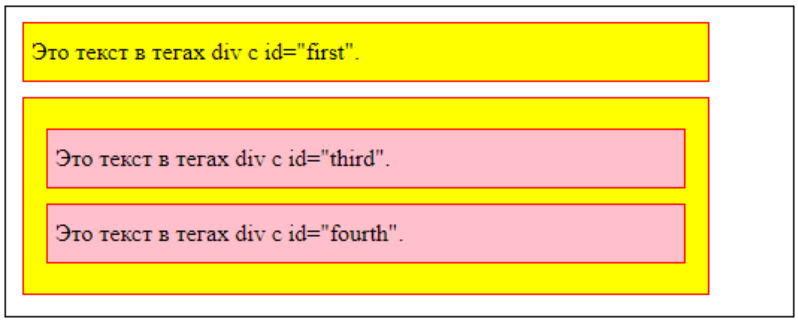
Страница в браузере:



6. Задать фон элементам, чтобы убедиться, что фон отступов совпадает с фоном элемента, а фон полей – прозрачный.

```
#first, #second, #third, #fourth{
    border: 1px solid red;
    margin:10px;
    padding:10px 5px;
}
#first, #second{
    background:yellow;
}
#third, #fourth{
    background:pink;
}
```

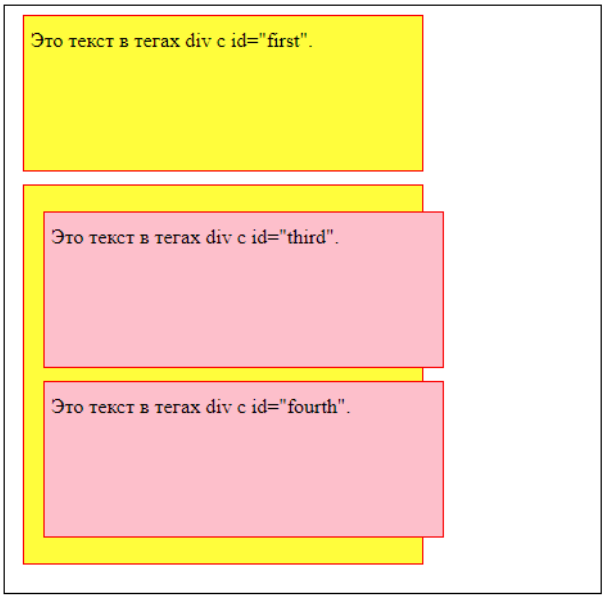
Страница в браузере:



7. Задать ширину и высоту блоков:

```
#first, #second, #third, #fourth{
    border: 1px solid red;
    margin:10px;
    padding:10px 5px;
}
#first{
    background:yellow;
    width:300px;
    height:100px;
}
#second{
    background:yellow;
    width:300px;
}
#third, #fourth{
    background:pink;
    width:300px;
    height:100px;
}
```

Страница в браузере:



Браузер отобразил блоки шириной в 300 пикселей, а потом задал им поля и отступы, именно на это количество пикселей и отличается ширина блоков.

8. Изменить ширину блоков розового цвета в 270 пикселей. Результат просмотреть в браузере.

9. Добавить на страницу style.css стили для наших элементов:

```
#first, #second, #third, #fourth{
    border: 1px solid red;
    margin:10px;
    padding:10px 5px;
    font-size:14px;
}
#first{
    background:yellow;
    width:300px;
    height:100px;
    text-align:center;
    color:blue;
}
#second{
    background:yellow;
    width:300px;
    text-align:center;
    color:blue;
}
#third, #fourth{
    background:pink;
    width:270px;
    height:100px;
    text-align:left;
    text-transform:capitalize;
    word-spacing:8px;
    color:red;
}
#first:first-letter{
    color:red;
    font-size:26px;
}
#third:first-letter, #fourth:first-letter{
    color:blue;
    font-size:26px;
}
```

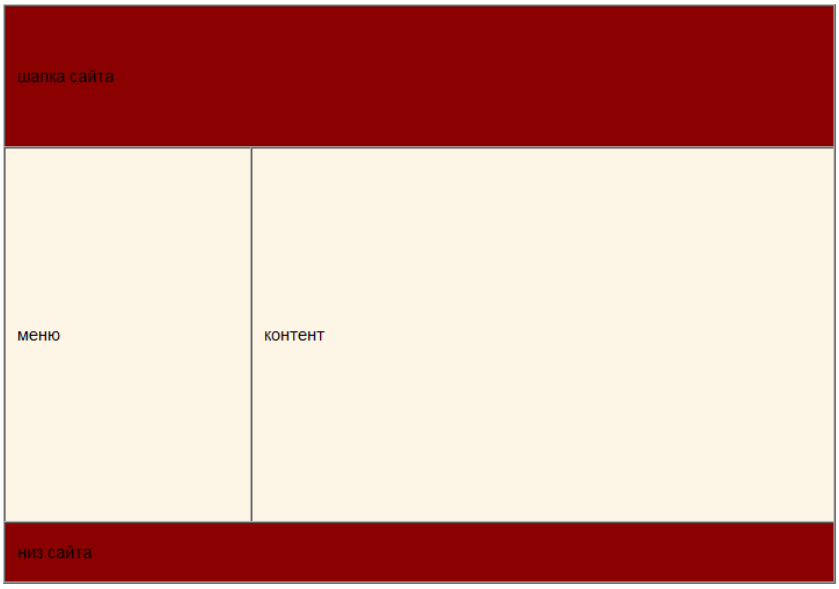
Страница в браузере:

9. ПОЗИЦИОНИРОВАНИЕ БЛОКОВ

В HTML для позиционирования элементов на странице используются таблицы. У таблиц есть как преимущества (легкость использования, одинаковое отображение браузерами), так и недостатки (объемный, нечитабельный код, нелогичность верстки и т.д.).

В CSS для позиционирования элементов используются блоки (**div**). Код при этом становится компактным, логичным и легко изменяемым. К недостаткам блочной верстки можно отнести неодинаковую поддержку браузерами, поэтому приходится писать кроссбраузерный код (т.е. код, который отображается разными браузерами почти одинаково).

1. Создать html-страницу:



Код html-страницы:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Заголовок документа</title>
    </head>
    <body>
      <table width="715"  border="1"
          align="center" cellspacing="0" cellpadding="10">
                <tr bgcolor="darkred">
            <td colspan="2" height="100">шапка сайта</td>
        </tr>
        <tr bgcolor="oldlace">
            <td  width="190" height="300">меню</td>
            <td>контент</td>
        </tr>
        <tr bgcolor="darkred">
            <td colspan="2" height="30">низ сайта</td>
        </tr>
     </table>
    </body>
</html>
```

2. Сверстать такую страницу средствами CSS.
    - Страница имеет четыре блока: шапка сайта, меню, контент и низ сайта. Таким образом, мы имеем четыре **div**. Написать html-код страницы с четырьмя **div** и каждому присвоить соответствующий идентификатор (**id**):

```
<!DOCTYPE html>
<html>
    <head>
        <title>CSS позиционирование</title>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>

    <body>
        <div id="header">шапка сайта</div>
        <div id="menu">меню</div>
        <div id="content">контент</div>
        <div id="footer">низ сайта</div>
    </body>

</html>
```

- Задать свойства: ширину, высоту и фон каждого блока.

```
#header{
background:darkred;
width:715px;
height:100px;
}
#menu{
background:oldlace;
width:190px;
height:300px;
}
#content{
background:oldlace;
width:525px;
height:300px;
}
#footer{
background:darkred;
width:715px;
height:30px;
}
```

- Страница в браузере:

Такое позиционирование элементов называется **позиционированием в нормальном потоке**. Это значит, что все элементы отображаются в окне браузера сверху вниз, по вертикали, в том порядке, в каком они следуют друг за другом в html-коде.

По своей сути нормальный поток ничем не отличается от позиционирования элементов в HTML. И для верстки такой страницы без CSS, нам пришлось бы использовать таблицу, за неимением других вариантов. В CSS же предоставляются и другие схемы позиционирования:

- **абсолютное позиционирование**
- **относительное позиционирование**
- **плавающая блоковая модель**

Для определения схемы позиционирования используется свойство **position**, оно может принимать четыре значения, соответствующие выбранной схеме позиционирования:

- **static** – блок позиционируется в нормальном потоке. Это значение по умолчанию.
- **relative** – относительное позиционирование (относительно нормального потока).
- **absolute** – абсолютное позиционирование
- **fixed** – фиксированное позиционирование (фиксируется относительно области просмотра).

Абсолютное позиционирование

При этой схеме позиционирования расположение блока на странице не зависит от того, в каком месте html-кода расположен этот блок. Расположение каждого блока задается указанием, в каком месте экрана отобразить данный блок. Для этого существуют четыре свойства:

- **left** – указывает на сколько надо сместить блок относительно левого края окна
- **right** – указывает на сколько надо сместить блок относительно правого края окна
- **top** – указывает на сколько надо сместить блок относительно верхнего края окна
- **bottom** – указывает на сколько надо сместить блок относительно нижнего края окна.

В вышеуказанном примере блоки **header**, **menu** и **footer** позиционируются в нормальном потоке, поэтому свойство **position** для них задавать не надо.

А вот блок **content** нужно расположить в другом месте, поэтому для него необходимо указать свойство **position:absolute** и задать смещение: от левого края окна на ширину блока **menu**, т.е. на 190 пикселей, а от верхнего края окна на высоту блока **header**, т.е. на 100 пикселей.

CSS-код

```
#header{
background:darkred;
width:715px;
height:100px;
}
#menu{
background:oldlace;
width:190px;
height:300px;
}
#content{
background:oldlace;
width:525px;
height:300px;
position:absolute;
left:190px;
top:100px;
}
#footer{
background:darkred;
width:715px;
height:30px;
}
```

Результат:

Блок расположился не совсем так, как ожидалось.

У браузеров есть свои, встроенные таблицы стилей. И, если не задано какое-либо свойство, то используется свойство по умолчанию.

Так, по умолчанию для элемента **body** определены поля, а они не были учтены при задании свойств смещения. Чтобы решить эту проблему, достаточно задать для **body** свойство **margin:0px**, т.е. явно указать размер полей (в нашем примере – их отсутствие).

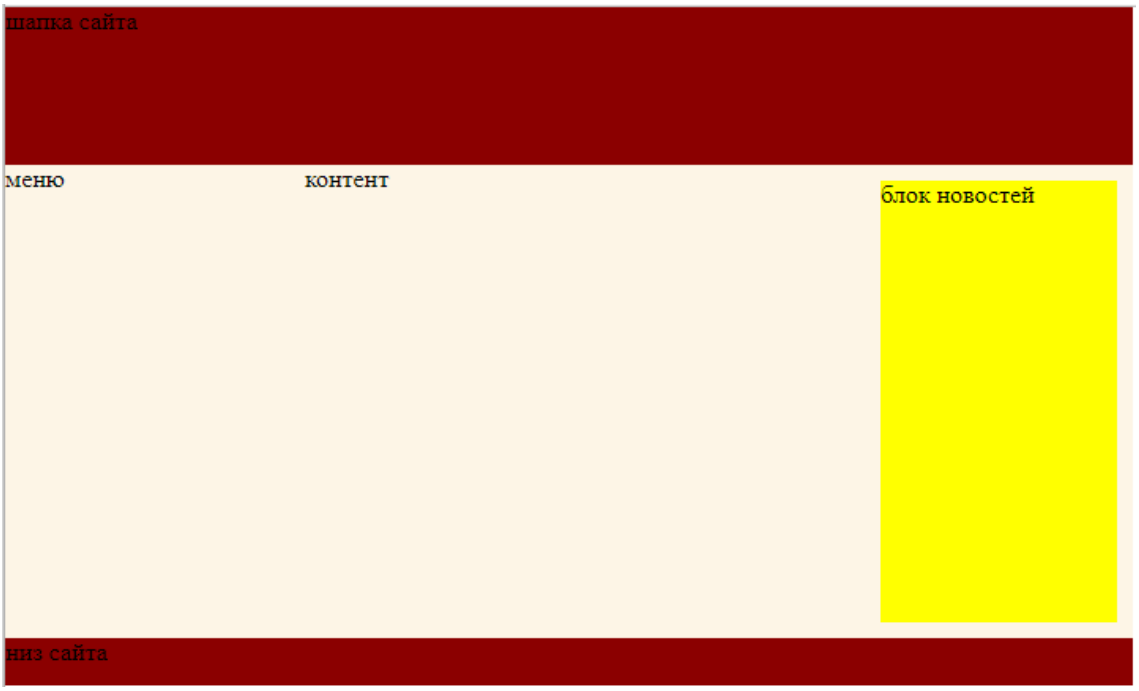
Добавить в таблицу стилей:

```
1  body{
2  margin:0px;
3  }
```

Просмотреть результат.

Размеры смещения можно было задать и для каждого блока, иногда это необходимо. Главное, что необходимо помнить: при абсолютном позиционировании следует задать для блока свойство **position:absolute** и свойства смещения относительно "родительского" элемента. В нашем примере родительским элементом для **div** было окно браузера.

3. Добавить блок новостей на нашу страницу и разместить его в блоке контента:



4. Для этого в html-страницу, в **div id="content"** добавить **div id="news"**:

```
<div id="content">
 контент
 <div id="news">блок новостей</div>
</div>
```

5. В таблице стилей смещение указать относительно блока **content**:

```
#news{
background:yellow;
width:150px;
height:280px;
position:absolute;
left:365px;
top:10px;
}
```

Ширина блока **content** равна 525 пикселей, а ширина блока **news** – 150 пикселей. Значит, смещение от левого края равно (525-150) 375 пикселей, но, чтобы блок не прилипал к правому краю, необходимо уменьшить смещение до 365 пикселей.

Аналогично рассчитывается смещение от верхнего края: высота блока **content** равна 300 пикселей, а высота блока **news** – 280 пикселей. Значит смещение от верхнего края может быть не более (300-280) 20 пикселей, мы сделали 10.

При абсолютном позиционировании, чтобы не запутаться с величинами смещения, необходимо определить сначала "родителя" и помнить, что смещение происходит относительно "родителя".

Относительное позиционирование

При относительном позиционировании блока надо задать свойство **position:relative** и свойства смещения. Смещение в этом случае будет происходить не относительно "родительского" элемента (как при абсолютном позиционировании), а относительно самого блока в нормальном потоке.

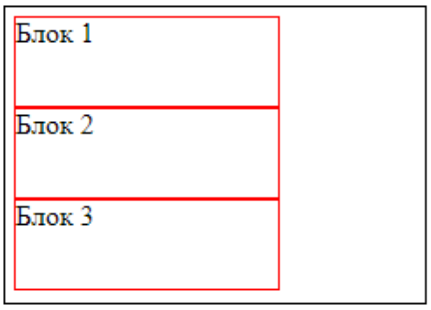
1. Создать html-страницу с тремя **div**:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Относительное позиционирование</title>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>

    <body>
        <div id="blok1">Блок 1</div>
        <div id="blok2">Блок 2</div>
        <div id="blok3">Блок 3</div>
    </body>
</html>
```

2. Задать в таблице стилей размеры и границы этих блоков:

```
#blok1, #blok2, #blok3 {
border:1px solid red;
width:150px;
height:50px;
}
```

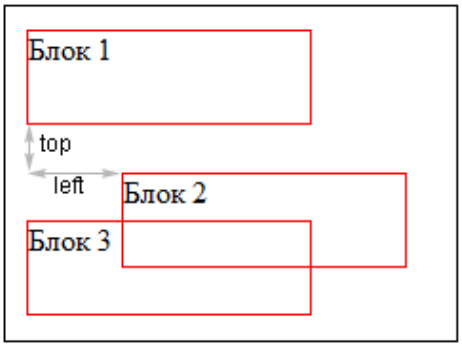
Страница в браузере:



3. Изменить положение второго блока, для этого добавить в страницу стилей правило:

```
#blok1, #blok2, #blok3 {
border:1px solid red;
width:150px;
height:50px;
}
#blok2{
position:relative;
left:50px;
top:25px;
}
```

Страница в браузере:



Второй блок сместился вниз и вправо относительно того места, где бы он находился в нормальном потоке. Остальные же блоки остались на своих местах. Практически относительное позиционирование применяется достаточно редко.

Плавающие блоки

Эти блоки нельзя позиционировать с точностью до пикселя, как в предыдущих схемах, но именно эта схема позиционирования очень распространенная. Без плавающих блоков обходится редкий сайт, а уж сделать "резиновую" верстку сайта без них и вовсе невозможно.

Такие блоки могут свободно перемещаться по странице, подобным образом ведут себя картинки в HTML, выровненные с помощью параметра **align**.

Плавающие блоки определяются свойством **float**, который определяет будет ли блок плавающим и в какую сторону он будет перемещаться. Возможны три варианта:

- **left** – блок прижимается к левому краю, остальные элементы обтекают его с правой стороны
- **right** – блок прижимается к правому краю, остальные элементы обтекают его с левой стороны
- **none** – блок не перемещается и позиционируется согласно свойству **position**

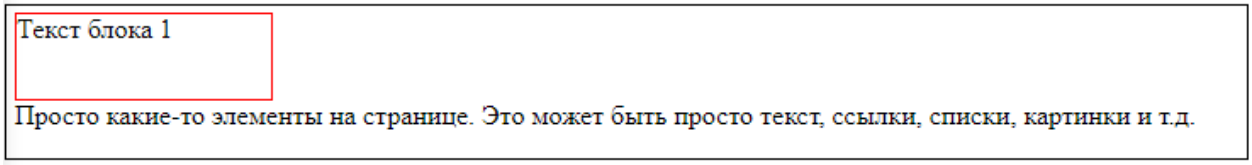
1. Создать html-страницу:

```
<!DOCTYPE html>
<html>

    <head>
      <title>Позиционирование блоков</title>
      <link rel="stylesheet" type="text/css" href="style.css">
    </head>

    <body>
      <div id="blok1">Текст блока 1</div>
      Просто какие-то элементы на странице. Это может быть просто
      текст, ссылки, списки, картинки и т.д.
    </body>

</html>
```

2. Код страницы style.css:

```
#blok1{
border:1px solid red;
width:150px;
height:50px;
}
```

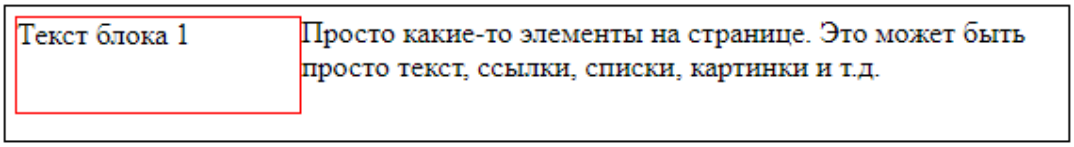
Страница в браузере:



3. Сделать блок плавающим и прижать его к левому краю:

```
#blok1{
border:1px solid red;
width:150px;
height:50px;
float:left;
}
```

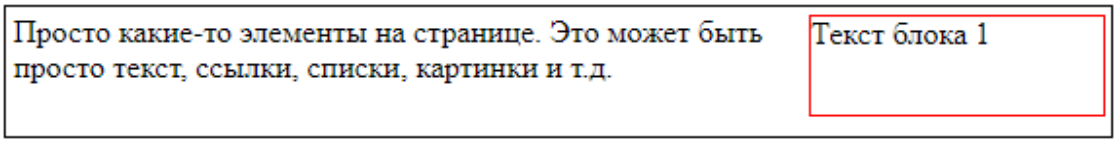
Страница в браузере:



4. Прижать блок к правому краю:

```
#blok1{
border:1px solid red;
width:150px;
height:50px;
float:right;
}
```

Страница в браузере:



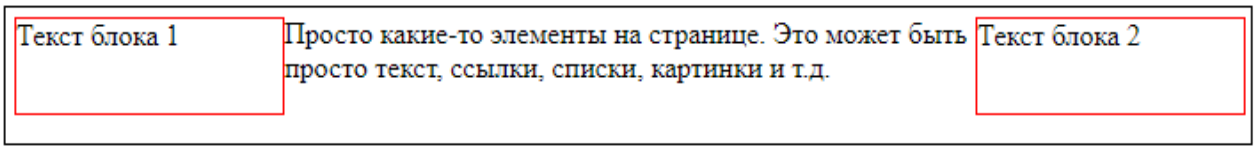
5. Добавить на html-страницу еще один блок:

```
<!DOCTYPE html>
<html>

    <head>
        <title>Позиционирование блоков</title>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>

    <body>
        <div id="blok1">Текст блока 1</div>
        <div id="blok2">Текст блока 2</div>
        Просто какие-то элементы на странице. Это может быть просто
        текст, ссылки, списки, картинки и т.д.
    </body>

</html>
```

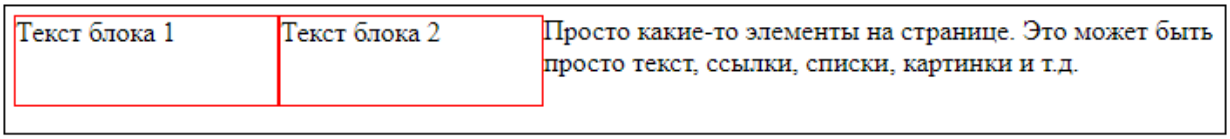
6. Задать блокам разные значения свойства **float**:

```
#blok1{
border:1px solid red;
width:150px;
height:50px;
float:left;
}
#blok2{
border:1px solid red;
width:150px;
height:50px;
float:right;
}
```

Страница в браузере:



7. Задать блокам одинаковые значения свойства **float:left**
   Тогда второй блок прижмется к правому краю первого блока:



Аналогична будет ситуация при одинаковых значениях **right**: сначала к правому краю прижмется блок 1, а уже к нему прижмется блок 2.

Если требуется, чтобы блоки были прижаты к правому краю, но располагались бы один под другим. Для этого существует свойство **clear**, которое определяет, какие стороны плавающего блока не могут соседствовать с другими плавающими блоками. У этого свойства может быть задано одно из четырех значений:

left – блок должен располагаться ниже всех левосторонних блоков.

right – блок должен располагаться ниже всех правосторонних блоков.

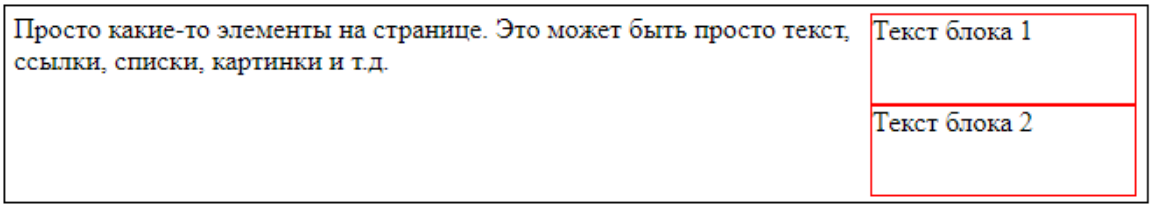
both – блок должен располагаться ниже всех плавающих блоков.

none – никаких ограничений нет, это значение по умолчанию.

8. В последнем примере задать свойство **clear**:

```
#blok1{
border:1px solid red;
width:150px;
height:50px;
float:right;
}
#blok2{
border:1px solid red;
width:150px;
height:50px;
float:right;
clear:right;
}
```

Страница в браузере:



ЛИТЕРАТУРА

1. CSS учебник [Электронный ресурс]. – Режим доступа: URL: https://html5css.ru/css/default.php
2. HTML5BOOK [Электронный ресурс]. – Режим доступа: URL: https://html5book.ru/css-css3/
3. Гумерова Л.З. Основы web-программирования: учеб. пособие. Красноярск: Научно-инновационный центр, 2019. 104 с. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – Режим доступа: URL: https://www.iprbookshop.ru/97112.html
4. Дронов В.А. HTML и CSS: 25 уроков для начинающих. СПб.: БХВ-Петербург, 2020. 400 с.
5. Ефромеев Н.М., Ефромеева Е.В. Основы web-программирования: учеб. пособие. Саратов: Вузовское образование, 2019. 128 с. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – Режим доступа: URL: https://www.iprbookshop.ru/86300.html
6. Маркин А.В. Web-программирование: учебник. М.: Ай Пи Ар Медиа, 2021. 286 с. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – Режим доступа: URL: https://www.iprbookshop.ru/104883.html
7. Основы Web-технологий [Электронный ресурс]: учеб. пособие / П.Б. Храмцов и др. М., Саратов: Интернет-университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. 375 с. – Режим доступа: http://www.iprbookshop.ru/67384.html
8. Самоучитель CSS [Электронный ресурс]. – Режим доступа: URL: https://htmlbook.ru/samcss
9. Современный учебник CSS [Электронный ресурс]. – Режим доступа: URL: https://idg.net.ua/blog/uchebnik-css
10. Учебник CSS для начинающих [Электронный ресурс]. – Режим доступа: URL: https://msiter.ru/tutorials/css-nachalnogo-urovnya

Учебное издание

Петракова Наталья Васильевна

ОСНОВЫ CSS
ЧАСТЬ 2

Учебно-методическое пособие
по дисциплине Web-программирование
для самостоятельной работы студентов
по направлению подготовки 09.03.03 Прикладная информатика

Редактор Лебедева Е.М.

Подписано к печати 11.10.2023 г. Формат 60x84 $^{1}/_{16}$.
Бумага офсетная. Усл. п. л. 6,21. Тираж 100 экз. Изд. №7578.

Издательство Брянского государственного аграрного университета
Брянская обл., Выгоничский район, с. Кокино, Брянский ГАУ