

# Лекция 3.

## ОСНОВЫ HTML и CSS

Web System Design and Development

# План лекции

- Введение в HTML
  - Что такое HTML? Основные понятия
  - Структура кода
  - Типы тегов
  - Семантические элементы
  - Основные элементы
  - Веб-формы
  - Элементы управления формой
  - Атрибуты событий
- Введение в CSS
  - Виды селекторов
  - Селекторы атрибутов
  - Псевдоклассы
  - Псевдоэлементы
  - Основные CSS свойства

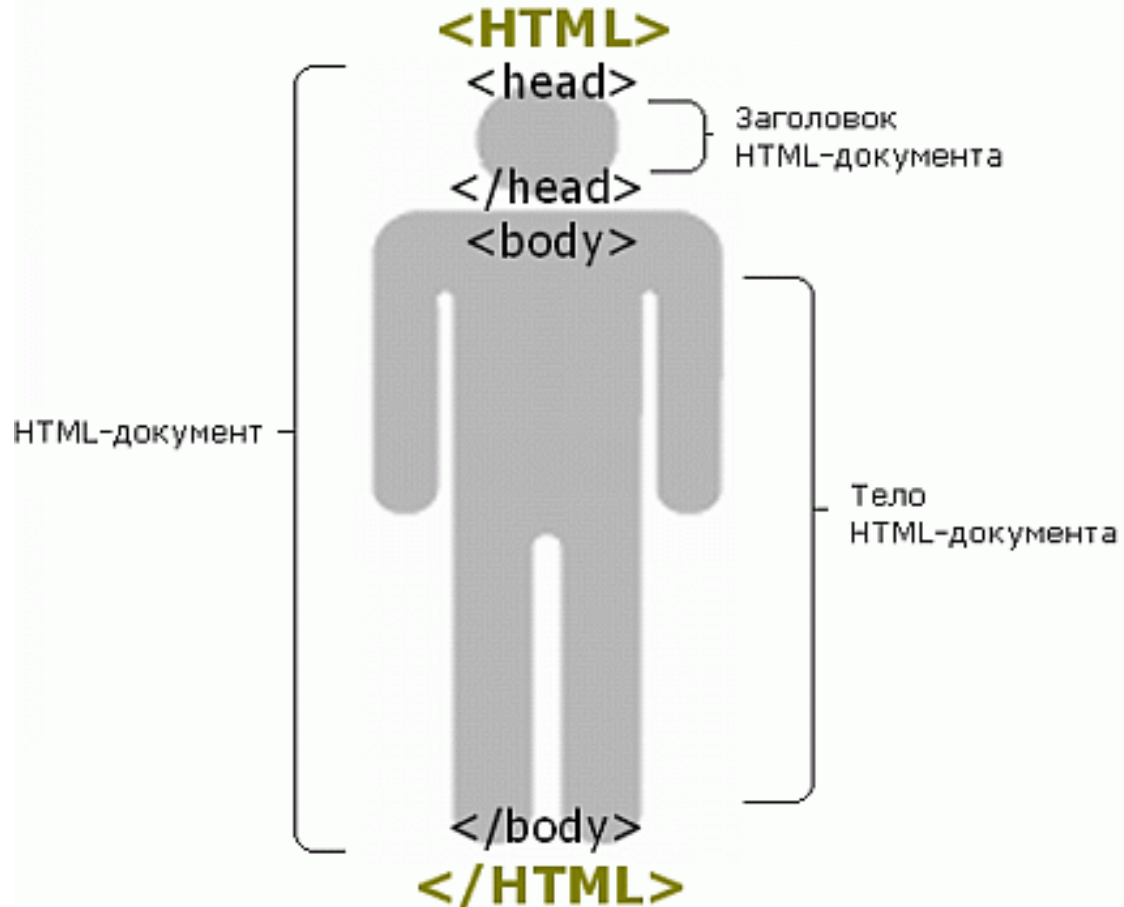
# Введение в HTML

# Что такое HTML? Основные понятия

## HTML – HyperText Markup Language

```
<!DOCTYPE html>
<html>
<!-- created 2010-01-01 -->
<head>
  <title>sample</title>
</head>
<body>
  <p>Voluptatem accusantium
    totam rem aperiam.</p>
</body>
</html>
```

HTML



Стандартный язык разметки документов.

Теговый язык.

HTML интерпретируется браузерами и отображается в виде документа в понятной для человека форме.

# Что такое HTML? Основные понятия

**Тег (tag)** — это специальный символ разметки, выделенный угловыми скобками (angle brackets, `<...>`), который применяется для вставки различных элементов на веб-страницу таких как: рисунки, таблицы, ссылки и др. и для изменения их вида.

**Контейнер** — парный тег, внутри которого могут располагаться другие теги. Контейнер требует закрывающего тега, обозначаемого `</тег>`. Таким образом, контейнер состоит из открывающего тега (`<тег>`) и закрывающего (`</тег>`).

`<h1> Теги в HTML </h1>`

`<!-- отобразит текст «Теги в HTML» как заголовок первого уровня -->`

**Элементы могут быть пустыми**, то есть не содержащими никакого текста и других данных (например, тег перевода строки `<br>`). В этом случае обычно не указывается закрывающий тег.

# Что такое HTML? Основные понятия

Начальный тег может содержать атрибуты — описание дополнительных свойств, уточняющих действие инструкций. Если атрибутов несколько, то они **отделяются друг от друга пробелом**.

Чтобы **указать атрибут**, сначала записывается его **полное имя** (иногда сокращенное), а затем **после знака равно значение**, которое оно должно принять.

```
<p style="color:blue">Списки</p>  
<!-- отобразит абзац «Списки» шрифтом синего цвета -->
```

Для **создания комментариев** в любой части документа используется парный тег **<!-- -->**.

# Что такое HTML? Основные понятия

Допускается одну пару тэгов заключать внутри другой пары. В этом случае инструкции внешней пары будут распространяться и на заключенную в нее **внутреннюю пару**.

`<b><i>`Форматирование текста`</i></b>`

`<!--отобразится курсивными и полужирными буквами:-->`

*Форматирование текста*

При написании тегов **строчные и прописные буквы не различаются**, **перенос строки** распознается как **пробел**, а **несколько подряд записанных пробелов** воспринимаются как **один**.

Исключением из этого правила является тег `<pre>`, внутри которого **любое число пробелов отображается** именно так, как оно указано в коде.

# Структура кода

**<!DOCTYPE html>  
must be the first line of  
code in all of your HTML  
documents.**

Document Type Definition -  
описание типа документа

**Минимальная структура HTML-страницы** включает следующие обязательные тэги (если они отсутствуют, то браузер подставляет их автоматически):

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



# Структура кода

HTML 4.01	
<code>&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"&gt;</code>	Строгий синтаксис HTML.
<code>&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"&gt;</code>	Переходный синтаксис HTML.
<code>&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd"&gt;</code>	В HTML-документе применяются фреймы.
HTML 5	
<code>&lt;!DOCTYPE html&gt;</code>	В этой версии HTML только один доктайп.

# Структура кода

```
<!DOCTYPE html>
<html>
  <head>
    <title>Моя первая страница </title>
  </head>
  <body>
    ...
  </body>
</html>
```

## <head>

содержит метаданные, может содержать текст и теги, но содержимое этого раздела не показывается напрямую на странице, за исключением контейнера <title>

Внутри контейнера <head> допускается размещать следующие элементы: <base>, <basefont>, <bgsound>, <link>, <meta>, <script>, <style>, <title>.

```
<meta charset="utf-8">
```

# Типы тегов

Теги верхнего уровня

Теги заголовка документа

Блочные элементы

Встроенные элементы

Универсальные элементы

Табличные элементы

Списки

# Типы тегов

## Теги верхнего уровня

Тег	Описание	
<b>&lt;html&gt;</b>	Является контейнером, который включает в себе всё содержимое веб-страницы, включая теги <head> и <body>. Открывающий и закрывающий теги <html> в документе необязательны, но хороший стиль диктует непременно их использование.	<html>  <head> ...  </head>  <body> ...  </body> </html>
<b>&lt;head&gt;</b>	Предназначен для хранения других элементов, цель которых — помочь браузеру в работе с данными. Также внутри контейнера <head> находятся метатеги, которые используются для хранения информации, предназначенной для браузеров и поисковых систем.	
<b>&lt;body&gt;</b>	Предназначен для хранения содержания веб-страницы, отображаемого в окне браузера. Информацию, которую следует выводить в документе, следует располагать именно внутри контейнера <body>.	

# Типы тегов

## Теги заголовка документа

Тег	Описание
<b>&lt;title&gt;</b>	Используется для отображения строки текста в левом верхнем углу окна браузера, а также на вкладке. Такая строка сообщает пользователю название сайта и другую информацию, которую добавляет разработчик.
<b>&lt;meta&gt;</b>	Метатеги используются для хранения информации, предназначенной для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных.

## Универсальные теги

Тег	Описание
<b>&lt;del&gt;</b>	Используется для выделения текста, который был удален в новой версии документа. Подобное форматирование позволяет отследить, какие изменения в тексте документа были сделаны. Браузеры обычно помечают текст в контейнере <b>&lt;del&gt;</b> как перечеркнутый.
<b>&lt;ins&gt;</b>	Тег <b>&lt;ins&gt;</b> предназначен для акцентирования вновь добавленного текста и обычно применяется наряду с тегом <b>&lt;del&gt;</b> . Браузеры помечают содержимое контейнера <b>&lt;ins&gt;</b> подчеркиванием текста.

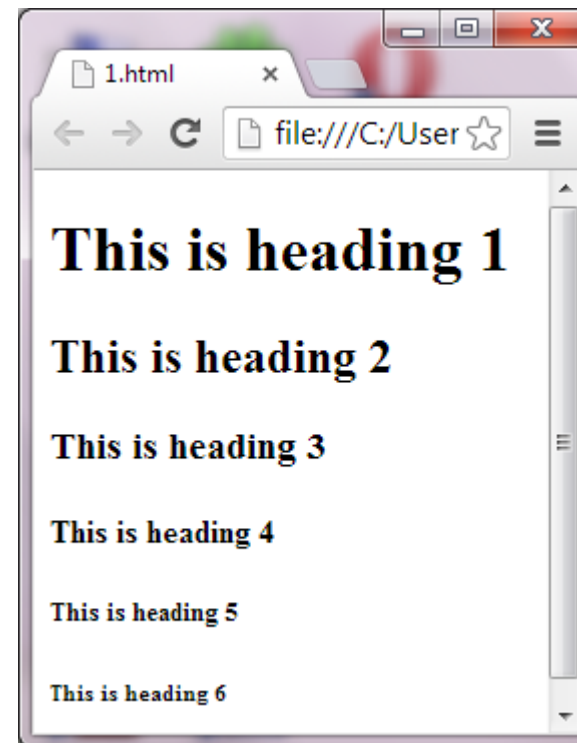
# Типы тегов

## Пример использования заголовков разного уровня

```
<!DOCTYPE html>
<html>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>

</body>
</html>
```



# Типы тегов

## Пример использования тега <pre>

```
<!DOCTYPE html>
<html>
<body>

<p>Тег pre удобен для отображения компьютерного кода:</p>

<pre>
for i = 1 to 10
    print i
next i
</pre>

</body>
</html>
```

Тег pre удобен для отображения компьютерного кода:

```
for i = 1 to 10
    print i
next i
```

# Типы тегов

## Блочные элементы

Тег	Описание	Синтаксис
<code>&lt;blockquote&gt;</code>	Предназначен для выделения длинных цитат внутри документа.	<code>&lt;blockquote&gt;Текст&lt;/blockquote&gt;</code>
<code>&lt;div&gt;</code>	Относится к универсальным блочным контейнерам и применяется в тех случаях, где нужны блочные элементы без дополнительных свойств.	<code>&lt;div&gt;...&lt;/div&gt;</code>
<code>&lt;h1&gt;, ..., &lt;h6&gt;</code>	Эта группа тегов определяет текстовые заголовки разного уровня, которые показывают относительную важность секции, расположенной после заголовка.	<code>&lt;h1&gt;Заголовок первого уровня&lt;/h1&gt;</code>
<code>&lt;hr&gt;</code>	Рисует горизонтальную линию, которая по своему виду зависит от используемых атрибутов.	
<code>&lt;p&gt;</code>	Определяет параграф (абзац) текста.	<code>&lt;p&gt;Текст&lt;/p&gt;</code>
<code>&lt;pre&gt;</code>	Задаёт блок предварительно форматированного текста. Такой текст отображается обычно моноширинным шрифтом и со всеми пробелами между словами.	<code>&lt;pre&gt;Текст&lt;/pre&gt;</code>



# Типы тегов

## Строчные элементы

Тег	Описание
<code>&lt;a&gt;</code>	Предназначен для создания ссылок. В зависимости от присутствия атрибутов <code>name</code> или <code>href</code> тег <code>&lt;a&gt;</code> устанавливает ссылку или якорь.
<code>&lt;b&gt;</code>	Определяет жирное начертание шрифта.
<code>&lt;big&gt;</code>	Тег <code>&lt;big&gt;</code> увеличивает размер шрифта на единицу по сравнению с обычным текстом. Т.е., добавление тега <code>&lt;big&gt;</code> увеличивает текст на одну условную единицу.
<code>&lt;br&gt;</code>	Тег <code>&lt;br&gt;</code> устанавливает перевод строки в том месте, где этот тег находится. В отличие от тега параграфа <code>&lt;p&gt;</code> , использование тега <code>&lt;br&gt;</code> не добавляет пустой отступ перед строкой.
<code>&lt;em&gt;</code>	Тег <code>&lt;em&gt;</code> предназначен для акцентирования текста. Браузеры отображают такой текст курсивным начертанием.
<code>&lt;i&gt;</code>	Устанавливает курсивное начертание шрифта.

# Типы тегов

## Строчные элементы

Тег	Описание
<b>&lt;img&gt;</b>	Тег <img> предназначен для отображения на веб-странице изображений в графическом формате GIF, JPEG или PNG.
<b>&lt;small&gt;</b>	Тег <small> уменьшает размер шрифта на единицу по сравнению с обычным текстом. По своему действию похож на тег <big>, но действует с точностью до наоборот.
<b>&lt;span&gt;</b>	Универсальный тег, предназначенный для определения строчного элемента внутри документа.
<b>&lt;strong&gt;</b>	Тег <strong> предназначен для акцентирования текста. Браузеры отображают такой текст жирным начертанием.
<b>&lt;sub&gt;</b>	Отображает шрифт в виде нижнего индекса. Текст при этом располагается ниже базовой линии остальных символов строки и уменьшенного размера — H <sub>2</sub> O.
<b>&lt;sup&gt;</b>	Отображает шрифт в виде верхнего индекса. По своему действию похож на <sub>, но текст отображается выше базовой линии текста — м <sup>2</sup> .

# Типы тегов

## Теги для списков

Тег	Описание	Синтаксис
<b>&lt;ol&gt;</b>	<b>Устанавливает нумерованный список</b> , т.е. каждый элемент списка начинается с числа или буквы и увеличивается по нарастающей.	<code>&lt;ul&gt;</code> <code>&lt;li&gt;элемент маркированного списка&lt;/li&gt;</code> <code>&lt;/ul&gt;</code>
<b>&lt;ul&gt;</b>	<b>Устанавливает маркированный список</b> , каждый элемент которого начинается с небольшого символа — маркера.	<code>&lt;ol&gt;</code> <code>&lt;li&gt;элемент нумерованного списка&lt;/li&gt;</code> <code>&lt;/ol&gt;</code>
<b>&lt;li&gt;</b>	<b>Определяет отдельный элемент списка.</b> Внешний тег <code>&lt;ul&gt;</code> или <code>&lt;ol&gt;</code> устанавливает тип списка — маркированный или нумерованный.	
<b>&lt;dd&gt;, &lt;dt&gt;, &lt;dl&gt;</b>	<b>Предназначены для создания списка определений.</b> Каждый такой список начинается с контейнера <code>&lt;dl&gt;</code> , куда входит тег <code>&lt;dt&gt;</code> создающий термин и тег <code>&lt;dd&gt;</code> задающий определение этого термина. Закрывающий тег <code>&lt;/dd&gt;</code> не обязателен, поскольку следующий тег сообщает о завершении предыдущего элемента.	<code>&lt;dl&gt;</code> <code>&lt;dt&gt;Термин 1&lt;/dt&gt;</code> <code>&lt;dd&gt;Определение термина 1&lt;/dd&gt;</code> <code>&lt;dt&gt;Термин 2&lt;/dt&gt;</code> <code>&lt;dd&gt;Определение термина 2&lt;/dd&gt;</code> <code>&lt;/dl&gt;</code>

# Типы тегов

## Теги для таблиц

Тег	Описание	Синтаксис
<b>&lt;table&gt;</b>	<b>Служит контейнером для элементов, определяющих содержимое таблицы.</b> Любая таблица состоит из строк и ячеек, которые задаются с помощью тегов <tr> и <td>.	<pre>&lt;table&gt;   &lt;tr&gt;     &lt;td&gt;...&lt;/td&gt;   &lt;/tr&gt; &lt;/table&gt;</pre>
<b>&lt;td&gt;</b>	<b>Предназначен для создания одной ячейки таблицы.</b> Тег <td> должен размещаться внутри контейнера <tr>, который в свою очередь располагается внутри тега <table>.	
<b>&lt;tr&gt;</b>	<b>Служит контейнером для создания строки таблицы.</b>	
<b>&lt;th&gt;</b>	<b>Предназначен для создания одной ячейки таблицы, которая обозначается как заголовочная.</b> Текст в такой ячейке отображается браузером обычно жирным шрифтом и выравнивается по центру.	<pre>&lt;table&gt;   &lt;tr&gt;     &lt;th&gt;...&lt;/th&gt;   &lt;/tr&gt; &lt;/table&gt;</pre>

# Особенности HTML5



28 October 2014

## Семантические элементы

`<video>`

`<audio>`

`<canvas>`

`<object>`

Было  
раньше

Стало в  
версии 5

`<section>`

`<article>`

`<nav>`

`<header>`

`<footer>`

`<div>,  
<span>`

Было  
раньше

Стало в  
версии 5

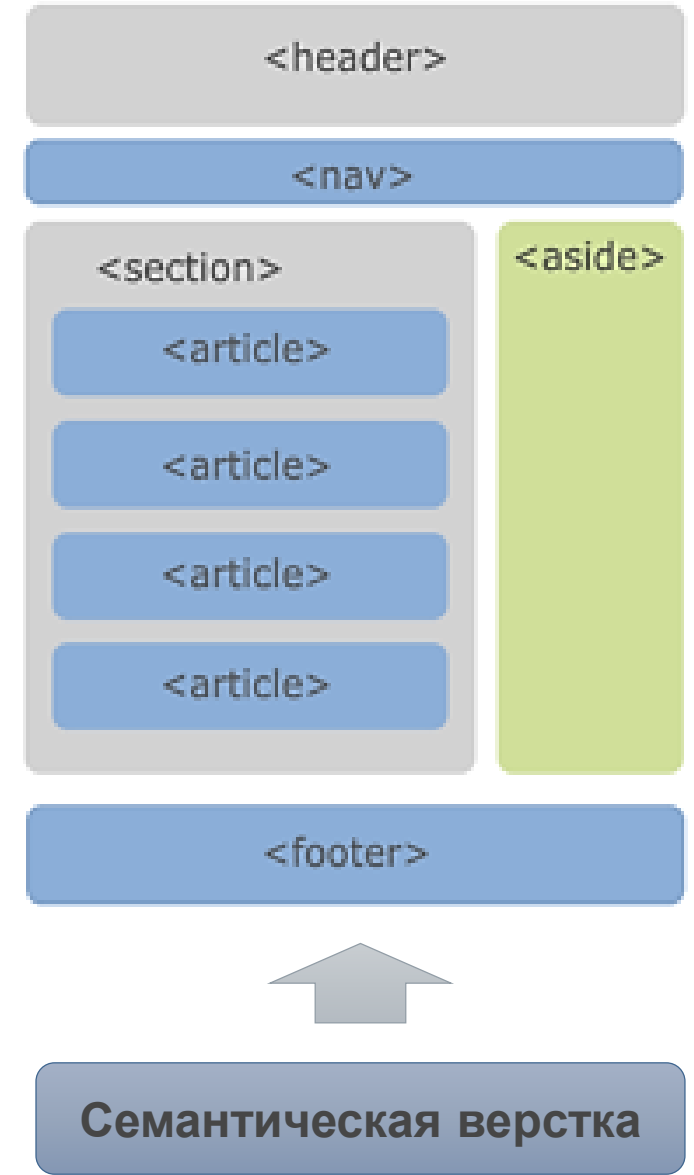
# Особенности HTML5

## Новые элементы:

<article>, <aside>, <audio>, <canvas>, <command>, <datalist>, <details>, <embed>, <figcaption>, <figure>, <footer>, <header>, <hgroup>, <keygen>, <mark>, <meter>, <nav>, <output>, <progress>, <rp>, <rt>, <ruby>, <section>, <source>, <summary>, <time>, <video>, <wbr>

## Элементы, которые исключены

<acronym>, <applet>, <basefont>, <big>, <center>, <dir>, <font>, <frame>, <frameset>, <isindex>, <noframes>, <strike>, <tt>



# Семантические элементы

## <header>

Группирует вводные и навигационные элементы, не является обязательным. Может содержать заголовки, оборачивать содержание раздела страницы, форму поиска или логотип. В html-документе может содержаться одновременно несколько элементов `<header>` и они могут располагаться в любой части страницы.

```
1 <header>
2   <h1>Site description</h1>
3   <nav>
4     <ul>
5       <li><a href="">About</a>
6       <li><a href="">Forum</a>
7       <li><a href="">Download</a>
8     </ul>
9   </nav>
10 </header>
```

Элемент `<header>` нельзя помещать внутрь элементов `<footer>`, `<address>` или другого элемента `<header>`.

# Семантические элементы

## <nav>

Предназначен для создания блока навигации веб-страницы или всего веб-сайта, при этом не обязательно должен находиться внутри `<header>`. На странице может быть несколько элементов `<nav>`. Не заменяет теги `<ul>` или `<ol>`, он просто их обрамляет. Не все группы ссылок на странице должны быть обернуты `<nav>`, этот элемент предназначен в первую очередь для разделов, которые состоят из главных навигационных блоков.

```
<nav>
  <ul>
    <li><a>...</a></li>
    <li><a>...</a></li>
    <li><a>...</a></li>
  </ul>
</nav>
```

В качестве элементов панели навигации можно использовать не только элементы списков:

```
<nav>
  <p><a href="">...</a></p>
  <p><a href="">...</a></p>
</nav>
```



# Семантические элементы

## <article>

- Используется для группировки записей — публикаций, статей, записей блога, комментариев. Представляет собой независимый обособленный блок, предназначенный для многократного использования, как правило, начинается с заголовка.
- Может дублироваться на других страницах сайта и содержать внутри другие элементы **<article>**, которые по содержанию имеют близкое отношение к содержанию внешней статьи.
- Если на странице присутствует только одна статья с заголовком и текстовым содержимым, она не нуждается в обёртке элементом **<article>**.

```
<article>
  <header>
    <h2>...</h2>
  </header>
  <p>...</p>
  <footer>
    Опубликовано в категории<a href="">Музыка</a>.
    <a href="">0 комментариев</a>
  </footer>
</article>
```

# Семантические элементы

## <section>

- Элемент представляет собой **универсальный раздел документа**. Группирует тематическое содержимое, не используется многократно и обычно содержит заголовков. Не является блоком-оберткой, для этих целей уместнее использовать элемент <div>.
- В качестве содержимого может выступать оглавление, разделы научных публикаций, программа мероприятия.
- Домашняя страница сайта также может быть поделена на секции — блок вводной информации, новости и контакты.

```
<article>
  <h1>...</h1>
  <section>
    <h2>...</h2>
    <p>...</p>
  </section>
  <section>
    <h2>...</h2>
    <p>...</p>
  </section>
  <p>...</p>
</article>
```

### <article> внутри <section>

Можно создавать родительские элементы <section> с вложенными элементами <article>, в которых есть один или несколько элементов <article>. Не все страницы должны быть устроены именно так, но это допустимый способ вложения элементов. Например, основная область контента страницы содержит два блока со статьями разной тематики. Можно сделать на этом акцент, поместив каждую статью одной тематики внутрь элемента <section>

# Семантические элементы

## <aside>

- Группирует содержимое, связанное с окружающим его контентом напрямую, но которое можно счесть отдельным (*т.е., удаление этого блока не повлияет на понимание основного содержимого*).
- Чаще всего элемент позиционируется как боковая колонка (как в книгах) и включает в себя группу элементов: <nav>, цифровые данные, цитаты, рекламные блоки, архивные записи.
- Не подходит для блоков, просто позиционированных в стороне.

```
<aside>
  <h2>...</h2>
  <p>...</p>
</aside>
```

```
1 <aside>
2   <h2>...</h2>
3   <p>...</p>
4   <blockquote>
5     <p>...<cite>...</cite>...</p>
6     <p>...</p>
7   </blockquote>
8 </aside>
```

# Семантические элементы

## <footer>

- Представляет собой нижний колонтитул содержащей его секции или корневого элемента. Обычно содержит информацию об авторе статьи, данные о копирайте и т.д. Если используется как колонтитул всей страницы, содержимое дополняется сведениями об авторских правах, ссылками на условия использования, контактную информацию, ссылками на связанное содержимое и т.п.
- В одном веб-документе может быть несколько элементов <footer>.
- Как каждая страница, так и каждая статья может иметь свой элемент <footer>, более того, <footer> можно поместить в элемент <blockquote>, чтобы указать источник цитирования.

```
<footer>  
  <address>...</address>  
  <small>©2014...</small>  
</footer>
```

# Основные элементы

## Ссылки (Link)

Для создания ссылки необходимо сообщить браузеру, что является ссылкой, а также указать адрес документа, на который следует сделать ссылку.

```
<a href="URL">текст ссылки</a>
```

- атрибут **href** определяет URL - адрес документа, на который следует перейти,
- содержимое контейнера **<a>** является ссылкой.

Пример:

```
<body>  
  <p><a href="dog.html">Собаки</a></p>  
  <p><a href="cat.html">Кошки</a></p>  
</body>
```

Результат в браузере:

[Собаки](#)

[Кошки](#)

# Основные элементы

## Пример создания относительных и абсолютных ссылок

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ter A, атрибут href</title>
  </head>
  <body>
    <p><a href="../../../example/knob.html">Относительная ссылка</a></p>
    <p><a href="http://htmlbook.ru/example/knob.html">Абсолютная ссылка</a></p>
  </body>
</html>
```

Относительная ссылка

Абсолютная ссылка

# Основные элементы

## Ссылка на адрес электронной почты

Создание ссылки на адрес электронной почты делается почти также как и ссылка на веб-страницу. Только вместо URL указывается **mailto:адрес электронной почты**.

Можно также автоматически добавить **тему сообщения**, присоединив к адресу электронной почты **?subject=тема сообщения**.

```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <title>Тема письма</title>
</head>
<body>
<p><a href="mailto:mail@ya.ru?subject=Вопрос по HTML">Задавайте
  вопросы по электронной почте</a></p>
</body>
</html>
```

# Основные элементы

## Другие ссылки

```
1  ссылка на телефонный номер
2  <a href="tel:+74951234567">+7 (495) 123-45-67</a>
3
4  ссылка на адрес электронной почты
5  <a href="mailto:example@mail.ru">example@mail.ru</a>
6
7  ссылка на скайп (позвонить)
8  <a href="skype:имя-пользователя?call">Skype</a>
9
10 ссылка на скайп (открыть чат)
11 <a href="skype:имя-пользователя?chat">Skype</a>
12
13 ссылка на скайп (добавить в список контактов)
14 <a href="skype:имя-пользователя?add">Skype</a>
15
16 ссылка на скайп (отправить файл)
17 <a href="skype:имя-пользователя?sendfile">Skype</a>
```



# Основные элементы

## Атрибуты ссылок

Элемент <a> поддерживает глобальные атрибуты и собственные.  
Один из собственных атрибутов ссылок:

target	Указывает на то, в каком окне должен открываться документ, к которому ведет ссылка. Принимает следующие значения: _self — страница загружается в текущее окно; _blank — страница открывается в новом окне браузера; _parent — страница загружается во фрейм-родитель; _top — страница загружается в полное окно браузера.
	...

# Основные элементы

## Якорь (Anchor)

**Закладка с уникальным именем на определенном месте веб-страницы, предназначенная для создания перехода к ней по ссылке.**

- Для **создания якоря** следует вначале сделать закладку в соответствующем месте и дать ей имя при помощи атрибута **name** тега **<a>**.
- В качестве значения href для перехода к этому якорю используется имя закладки с символом решетки (#) впереди.

```
<body>
  <p><a name="top"></a></p>
  <p>...</p>
  <p><a href="#top">Наверх</a></p>
</body>
```

Ссылку можно также сделать на закладку, находящуюся в другой веб-странице и даже другом сайте. Для этого в атрибуте href тега **<a>** надо указать адрес документа и в конце добавить символ решетки # и имя закладки.

# Основные элементы

## Изображения

**SRC** – обязательный атрибут, указывающий источник изображения.

**ALT** – альтернативный текст позволяет получить текстовую информацию о рисунке при отключенном в браузере показе картинок или во время их загрузки.

Текст в атрибуте alt обязательно должен быть взят в кавычки.

```

```

```
<body>
  <p><a href="index.html"></a></p>
</body>
```

Больше информации: <https://html5book.ru/images-in-html/>

# Основные элементы

## Изображения с активными областями

Тег **<map>** служит для представления графического изображения в виде карты с активными областями. Активные области определяются по изменению вида курсора мыши при наведении. Щелкая мышью на активных областях, пользователь может переходить к связанным документам.

Тег **<area>** описывает только одну активную область в составе карты изображений на стороне клиента.

```

<map name="flowers">
  <area shape="circle" coords="70,164,50" href="https://ru.wikipedia.org/wiki/Гербера"
    alt="gerbera" target="_blank"> <!--задаем активную область в виде круга и указываем координаты ее центра и радиус-->
  <area shape="poly" coords="191,13,240,98,143,98,191,13" href="https://ru.wikipedia.org/wiki/%C3%E8%E0%F6%E8%ED%F2"
    alt="hyacinth" target="_blank"> <!--задаем активную область в виде треугольника и указываем координаты его вершин-->
</map>
```

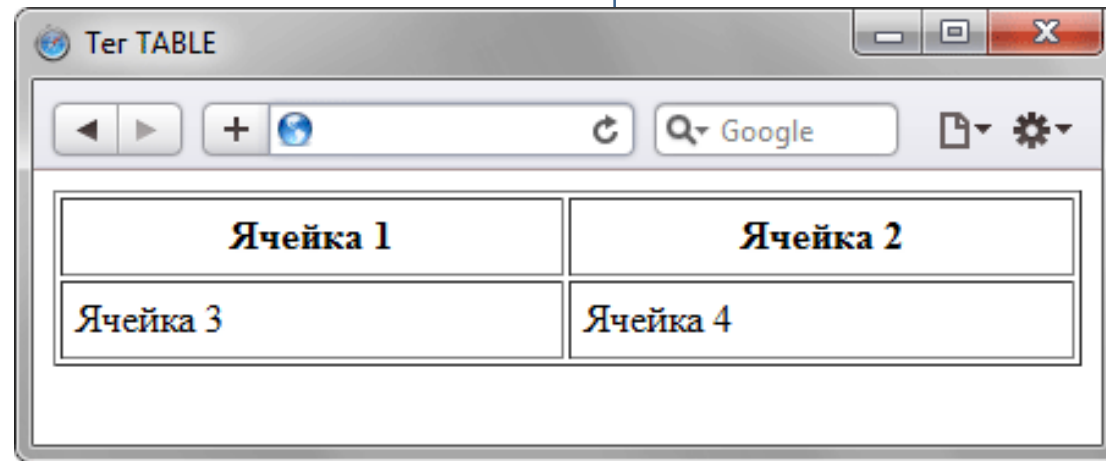
Больше информации: <https://html5book.ru/images-in-html/>

# Основные элементы

## Таблицы

Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. Обычно таблицы используются для упорядочения и представления данных. С помощью таблиц удобно верстать макеты страниц, расположив нужным образом фрагменты текста и изображений.

```
<body>
  <table border="1" width="100%" cellpadding="5">
    <tr>
      <th>Ячейка 1</th>
      <th>Ячейка 2</th>
    </tr>
    <tr>
      <td>Ячейка 3</td>
      <td>Ячейка 4</td>
    </tr>
  </table>
</body>
```



# Основные элементы

## Атрибуты тега <table>

Атрибут	Описание
<b>align</b>	Задаёт выравнивание таблицы по краю окна браузера. Допустимые значения: left — выравнивание таблицы по левому краю, center — по центру и right — по правому краю. Когда используются значения left и right, текст начинает обтекать таблицу сбоку и снизу.
<b>bgcolor</b>	Устанавливает цвет фона таблицы.
<b>border</b>	Устанавливает толщину границы в пикселах вокруг таблицы. При наличии этого атрибута также отображаются границы между ячейками.
<b>cellpadding</b>	Определяет расстояние между границей ячейки и её содержимым. Этот атрибут добавляет пустое пространство к ячейке, увеличивая тем самым её размеры. Без cellpadding текст в таблице «налипает» на рамку, ухудшая тем самым его восприятие. Добавление же cellpadding позволяет улучшить читабельность текста. При отсутствии границ особого значения этот атрибут не имеет, но может помочь, когда требуется установить пустой промежуток между ячейками.

# Основные элементы

## Атрибуты тега <table>

Атрибут	Описание
<b>cellspacing</b>	Задаёт расстояние между внешними границами ячеек. Если установлен атрибут border, толщина границы принимается в расчёт и входит в общее значение.
<b>cols</b>	Указывает количество столбцов в таблице. Без этого атрибута таблица будет показана только после того, как все её содержимое будет загружено в браузер и проанализировано. Использование атрибута cols позволяет несколько ускорить отображение содержимого таблицы.
<b>rules</b>	Сообщает браузеру, где отображать границы между ячейками. По умолчанию рамка рисуется вокруг каждой ячейки, образуя тем самым сетку. В дополнение можно указать отображать линии между колонками (значение cols), строками (rows) или группами (groups), которые определяются наличием тегов <thead>, <tfoot>, <tbody>, <colgroup> или <col>.
<b>width</b>	Задаёт ширину таблицы.

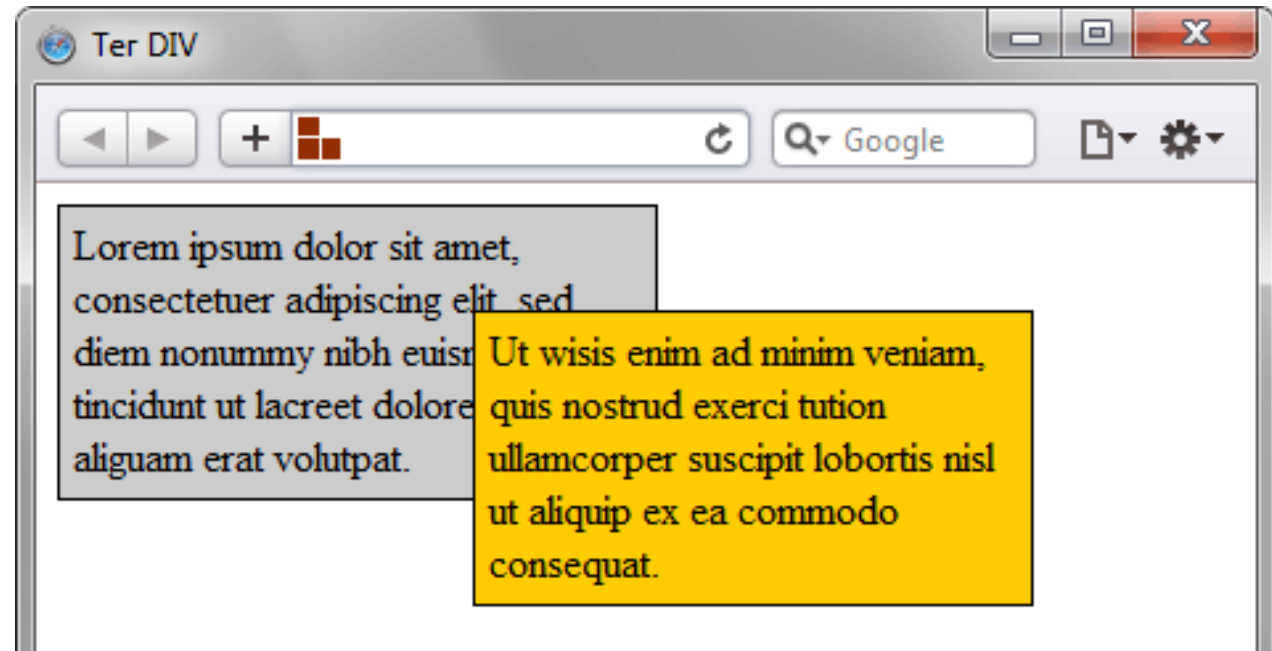
Больше: <https://html5book.ru/html-table/>

# Основные элементы

## Блочный элемент <div>

Предназначен для выделения фрагмента документа с целью изменения вида содержимого. Как правило, вид блока управляется с помощью **стилей**.

```
<div>...</div>
```





# Веб-формы

## Веб-форма:

- задается тегом **<form>**
- внутри - любые необходимые теги,
  - содержит обязательные и необязательные параметры:

адрес программы на веб-сервере, которая будет обрабатывать содержимое данных формы - **обязательный атрибут action**;

элементы управления формой, которые представляют собой стандартные поля для ввода информации пользователем;

кнопку отправки данных на сервер - Submit.

# Веб-формы

## Форма

для обмена данными между пользователем и сервером

с помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению.

документ может содержать любое количество форм, но одновременно на сервер может быть отправлена только одна форма. По этой причине данные форм должны быть независимы друг от друга.

```
<form action="http://example.com/app/profile" method="post">
  <p>
    <label for="username">Имя: </label>
    <input type="text" id="username" /><br />
    <label for="sex">Пол: </label>
    <input type="radio" name="sex" value="male">мужской<br />
    <input type="radio" name="sex" value="female">женский<br />
    <input type="submit" value="Отправить"> <input type="reset" value="Отменить">
  </p>
</form>
```

# Веб-формы

## Элемент `<form>`

Кроме общих для HTML атрибутов, в `<form>` могут присутствовать следующие:

- **action** (действие) — обязательный атрибут, содержащий URI обработчика формы;
- **method** (метод отправки формы) — атрибут, принимающий значения GET (по умолчанию) или POST;
- **enctype** (тип кодирования для содержимого) — по умолчанию `application/x-www-form-urlencoded` (всегда для метода GET), но обычно употребляется `multipart/form-data`;
- **accept** — список MIME-типов для загрузки файлов;
- **name** — имя формы;
- **onsubmit** — обработчик события «форма отправлена»;
- **onreset** — обработчик события: «форма очищена»;
- **accept-charset** — список поддерживаемых наборов символов.

# Элементы управления формой

## Виды элементов формы

Атрибут, определяющий вид элемента — **type**. Он позволяет задавать следующие типы элементов формы:

- текстовое поле (text);
- поле с паролем (password);
- переключатель (radio);
- флажок (checkbox);
- скрытое поле (hidden);
- кнопка (button);
- кнопка для отправки формы (submit);
- кнопка для очистки формы (reset);
- поле для отправки файла (file);

# Элементы управления формой

Появились в  
HTML5

Тип	Описание
<b>color</b>	Виджет для выбора цвета.
<b>date</b>	Поле для выбора календарной даты.
<b>datetime</b>	Указание даты и времени.
<b>datetime-local</b>	Указание местной даты и времени.
<b>email</b>	Для адресов электронной почты.
<b>number</b>	Ввод чисел.
<b>range</b>	Ползунок для выбора чисел в указанном диапазоне.
<b>search</b>	Поле для поиска.
<b>tel</b>	Для телефонных номеров.
<b>time</b>	Для времени.
<b>url</b>	Для веб-адресов.
<b>month</b>	Выбор месяца.
<b>week</b>	Выбор недели.

# Элементы управления формой

**Элементы управления** служат для взаимодействия пользователя с формой.

Атрибут **name** определяет имя элемента управления с областью видимости внутри данной формы.

С каждым элементом формы связано **начальное** и **текущее значение**. За некоторыми исключениями (textarea, object), **начальное значение** может быть **задано атрибутом value**. Значения, соответствующие элементам, могут изменяться при взаимодействии пользователя или скриптов (например, на Javascript) с формой.

При очистке формы элементы приобретают **начальные значения**.

Для **отправки** данных формы используются методы **GET** или **POST**.

# Элементы управления формой

## Элемент «Однострочное текстовое поле»

`<input type="text">`

предназначено для ввода символов с помощью клавиатуры

```
<form action="demo_form.asp">  
First name: <input type="text" name="FirstName" value="Mickey"><br>  
Last name: <input type="text" name="LastName" value="Mouse"><br>  
<input type="submit" value="Submit">  
</form>
```

Результат в браузере:

First name: Mickey

Last name: Mouse

Submit

# Элементы управления формой

Элемент «Пароль»

**`<input type="password">`**

обычное текстовое поле, но отличается от него тем, что все символы показываются звездочками.

```
<form>  
Password: <input type="password" name="pwd">  
</form>
```

Результат в браузере:

Password:



# Элементы управления формой

Элемент «Переключатель»

**<input type="radio">**

позволяет пользователю **выбрать только один вариант** из нескольких предложенных.

```
<input type="radio" name="Пол" value="male"> Мужской <br>  
<input type="radio" name="Пол" value="female"> Женский
```

Результат в браузере:



☐ Мужской  
☒ Женский

# Элементы управления формой

Элемент «Чекбокс (флажок)»

**`<input type="checkbox">`**

позволяет выбрать более одного варианта из предложенных или не одного.

```
<form>  
<input type="checkbox" name="vehicle" value="Bike">I have a bike<br>  
<input type="checkbox" name="vehicle" value="Car">I have a car  
</form>
```

Результат в браузере:

☐ I have a bike  
☐ I have a car

# Элементы управления формой

Элемент «Кнопка для отправки формы Submit»

**`<input type="submit">`**

кнопка для отправки данных формы на сервер

```
<form name="input" action="html_form_action.asp" method="get">  
Username: <input type="text" name="user">  
<input type="submit" value="Submit">  
</form>
```

Результат в браузере:

Username:

# Элементы управления формой

Элемент «Кнопка для очистки формы Reset»

**`<input type="reset">`**

кнопка **для очистки формы**, т.е. возвращения данных формы в первоначальное значение.

```
<form>
  <p><input value="Введите текст"></p>
  <p><input type="submit" value="Отправить">
    <input type="reset" value="Очистить"></p>
</form>
```

Результат в браузере:

# Элементы управления формой

Элемент «Кнопка с изображением»

**<input type="image" >**

кнопка с изображением.

При нажатии на рисунок данные формы отправляются на сервер.

```
<form action="input10.php">
  <p><input type="radio" name="drink" value="rad1"> Пиво<Br>
  <input type="radio" name="drink" value="rad2"> Чай<Br>
  <input type="radio" name="drink" value="rad3"> Кофе</p>
  <p><input type="image" src="images/imgbutton.gif"></p>
</form>
```

Результат в браузере:

- ☐ Пиво
- ☐ Чай
- ☐ Кофе

Отправить

# Элементы управления формой

## Элемент «Поле для отправки файла» <input type="file">

Прежде, чем использовать данное поле, в форме необходимо:


- задать метод отправки данных **POST (method="post")**;
- установить у атрибута **enctype** значение **multipart/form-data**.

```
<html>
<head>
  <meta charset="utf-8">
  <title>Отправка файла на сервер</title>
</head>
<body>
  <form enctype="multipart/form-data" method="post">
    <p><input type="file" name="f">
      <input type="submit" value="Отправить"></p>
  </form>
</body>
</html>
```

# Элементы управления формой

## Элемент «Скрытое текстовое поле»

**<input type="hidden">** - не отображается на странице и прячет своё содержимое от пользователя.

```
 <form>  
  <input type="hidden" value="hiddenvalue1" name="Hidden1" />  
</form>
```

## Элемент «Кнопка»

**<input type="button">** - кликабельная кнопка (в основном используется с JavaScript для активации скрипта).

```
<input type="button" value="Click me" onclick="msg()">
```

# Элементы управления формой

## Поле «Многострочный текст»

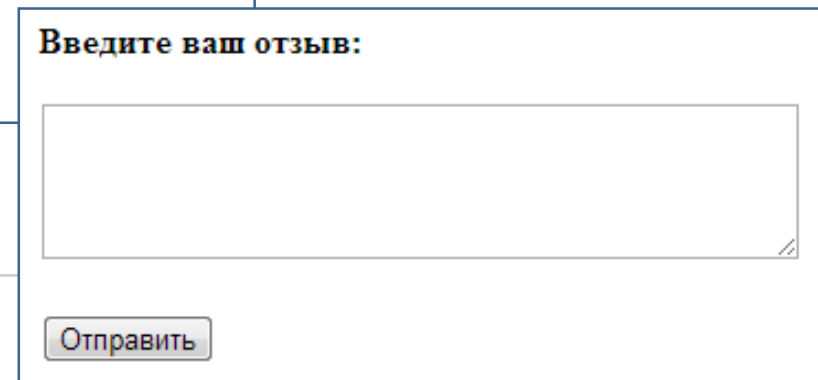
### <textarea атрибуты>

Элемент формы для создания области, в которую можно вводить несколько строк текста. В отличие от тега <input> в текстовом поле допустимо делать переносы строк, они сохраняются при отправке данных на сервер.

Между тегам <textarea> и </textarea> можно поместить любой текст, который будет отображаться внутри поля.

```
<form action="textareal.php" method="post">
  <p><b>Введите ваш отзыв:</b></p>
  <p><textarea rows="10" cols="45" name="text"></textarea></p>
  <p><input type="submit" value="Отправить"></p>
</form>
```

Результат в браузере:



Введите ваш отзыв:

Отправить



# Элементы управления формой

## Кнопка

- Тег **<button>** создает на веб-странице **кнопки** и по своему действию напоминает результат, получаемый с помощью тега `<input>` (с атрибутом `type="button | reset | submit"`). В отличие от этого тега, `<button>` предлагает **расширенные возможности по созданию кнопок**.

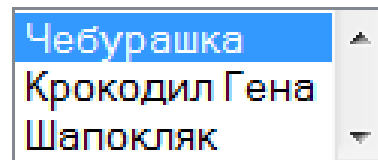
```
<form>  
  <button>...</button>  
</form>
```

```
<body>  
  <p style="text-align: center"><button>Кнопка с текстом</button>  
  <button> Кнопка с рисунком</button></p>  
</body>
```

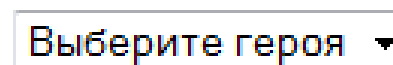
# Элементы управления формой

## Поле со списком

- Тег **<select>** позволяет создать элемент интерфейса в виде **раскрывающегося списка**, а также **список с одним или множественным выбором**.
- Конечный вид зависит от использования **атрибута size** тега **<select>**, который **устанавливает высоту списка**.
- **Ширина списка** определяется **самым широким текстом**, указанным в теге **<option>**, а также может изменяться с помощью стилей.
- **Каждый пункт** создается с помощью тега **<option>**, который должен быть вложен в контейнер **<select>**.



*Список множественного  
выбора*



*Раскрывающийся  
список*

# Элементы управления формой

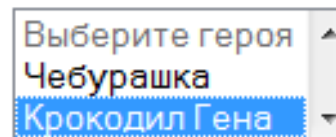
Синтаксис:

```
<select>
  <option>Пункт 1</option>
  <option>Пункт 2</option>
</select>
```

Пример создания списка множественного выбора:

```
<form action="select1.php" method="post">
  <p><select size="3" multiple name="hero[]">
    <option disabled>Выберите героя</option>
    <option value="Чебурашка">Чебурашка</option>
    <option selected value="Крокодил Гена">Крокодил Гена</option>
    <option value="Шапокляк">Шапокляк</option>
    <option value="Крыса Лариса">Крыса Лариса</option>
  </select></p>
  <p><input type="submit" value="Отправить"></p>
</form>
```

Результат в браузере:



Отправить

# Элементы управления формой

Тег, который часто используется в формах вместе с элементами управления формой

**<label>**

Тег <label> устанавливает связь между определенной меткой, в качестве которой обычно выступает текст, и элементом формы (<input>, <select>, <textarea>).

Существует два способа связывания объекта и метки:

- Использовать идентификатор **id** внутри элемента формы и указать его имя в качестве атрибута **for** тега **<label>**.
- элемент формы поместить **внутри** контейнера **<label>**.

```
<input id="идентификатор"><label for="идентификатор">Текст</label>  
<label><input type="..."> Текст</label>
```

# Атрибуты тега <input>

## disabled

Блокирует доступ и изменение поля формы. Оно в таком случае отображается серым и недоступным для активации пользователем.

```
<input type="..." disabled>
```

## checked

Определяет, **помечен** ли заранее такой элемент формы, как **флажок** или **переключатель**. В случае использования переключателей (radio button), может быть отмечен только один элемент группы, для флажков (checkbox) допустимо пометить **хоть все элементы**.

```
<input type="radio" checked>  
<input type="checkbox" checked>
```

# Атрибуты тега <input>

## readonly

Когда к тегу <input> добавляется атрибут **readonly**, **поле не может изменяться пользователем**, в том числе вводиться новый текст или модифицироваться существующий.

```
<input type="text" readonly>
```

## placeholder

Выводит **текст внутри поля** формы, который **исчезает при получении фокуса**.

```
<input placeholder="строка">
```

## autocomplete

Помогает заполнять поля форм текстом, который был введен в них ранее

```
<input type="text" autocomplete="on | off">  
<input type="password" autocomplete="on | off">  
<input type="email" autocomplete="on | off">
```

# Атрибуты тега <input>

## value

Определяет значение элемента формы, которое будет отправлено на сервер или получено с помощью клиентских скриптов.

На сервер отправляется пара «имя=значение», где имя задается атрибутом name тега <input>, а значение — атрибутом value.

- для кнопок (input type="button | reset | submit") устанавливает текстовую надпись на них;
- для текстовых полей (input type="password | text") указывает предварительно введенную строку.
- для флажков и переключателей (input type="checkbox | radio") уникально определяет каждый элемент, с тем, чтобы клиентская или серверная программа могла однозначно установить, какой пункт выбрал пользователь.
- для файлового поля (input type="file") не оказывает влияние.

# Атрибуты событий

**События HTML 5** - это специальные глобальные атрибуты, используемые в тегах для вызова обработчиков событий, написанных на различных языках сценариев таких, как JavaScript и вызываемых, когда на странице происходит какое-либо действие.

События позволяют сделать страницу динамической.

Атрибуты событий окна

События формы

События клавиатуры

События мыши

События медиа-файлов



# Атрибуты событий

## Атрибуты событий окна

События, вызываемые на объект окна (применяются с тегом <body>):

Атрибут	Описание
onbeforeunload	Скрипт запустится до загрузки документа.
onstorage	Скрипт запустится при загрузке документа.
onresize	Скрипт запустится при изменении размеров окна.
onerror	Скрипт запустится, если произойдет ошибка.
onoffline	Скрипт запустится, когда документ перейдет в автономный режим.
onafterprint	Скрипт запустится после того, как документ будет распечатан.

# Атрибуты событий

## События формы

События срабатывают на действия внутри HTML-формы (могут применяться ко всем элементам HTML, но чаще всего применяются к элементам формы):

Атрибут	Описание
onchange	Скрипт запустится при изменении элемента.
onformchange	Скрипт запустится при изменении формы.
onforminput	Скрипт запустится, когда пользователь будет вводить данные в форму.
oninput	Скрипт запустится, когда элемент станет получать пользовательский ввод.
onselect	Скрипт запустится при выборе элемента.
onsubmit	Скрипт запустится при отправке формы.

# Атрибуты событий

## События клавиатуры

События, вызываемые клавиатурой (могут применяться ко всем элементам HTML):

Атрибут	Описание
onkeydown	Скрипт запустится, когда будет нажата клавиша.
onkeypress	Скрипт запустится после того, как клавиша была нажата и отпущена.
onkeyup	Скрипт запустится при отпуске клавиши.

Полный перечень: [http://www.w3schools.com/tags/ref\\_eventattributes.asp](http://www.w3schools.com/tags/ref_eventattributes.asp)

# Атрибуты событий

## События мыши

События, вызываемые действиями мышкой или аналогичными действиями пользователя (могут применяться ко всем элементам HTML):

Атрибут	Описание
onclick	Код события будет выполнен по клику левой кнопкой мыши на элементе.
ondblclick	Скрипт запустится после двойного клика мыши.
ondrag	Скрипт запустится при перетаскивании элемента.
ondragend	Скрипт запустится после операции перетаскивания.
ondragstart	Скрипт запустится в начале операции перетаскивания элемента.
onmousedown	Скрипт запустится при нажатии кнопки мыши.
onmousemove	Скрипт запустится, когда указатель мыши начнет перемещаться.

# Атрибуты событий

## События медиа-файлов

События, вызываемые для медиа файлов, таких как видео, изображения и аудио (могут применяться ко всем элементам HTML, но чаще всего применяются к таким элементам, как `<audio>`, `<embed>`, `<img>`, `<object>`, и `<video>`):

Атрибут	Описание
onplaying	Скрипт запустится при воспроизведении файла.
onprogress	Скрипт запускается, когда браузер находится в процессе получения данных медиа файла.
onloadeddata	Скрипт запустится при загрузке данных медиа файла.
oncanplay	Скрипт запустится, когда файл будет готов, чтобы начать проигрывание (когда будет достаточно буферизирован).

**Больше:** [http://www.w3schools.com/tags/ref\\_eventattributes.asp](http://www.w3schools.com/tags/ref_eventattributes.asp)

# Атрибуты событий

## Примеры

```
<button onclick="copyText()">Copy Text</button>
```

```

```

```
<input type="text" onkeydown="displayResult()">
```

```
<body onresize="showMsg()">
```

# Введение в CSS

# Что такое CSS?

**CSS (Cascading Style Sheets) - каскадные таблицы стилей**



Стандарт на основе текстового формата, определяющий представление данных в браузере;

Набор параметров форматирования, который применяется к элементам документа, чтобы изменить их внешний вид;



# Что такое CSS?

Основной целью разработки CSS являлось **разделение описания логической структуры** веб-страницы (которое производится с помощью HTML или других языков разметки) **от описания внешнего вида этой веб-страницы** (которое теперь производится с помощью формального языка CSS).

Термин "каскадный" означает, что в одной странице HTML могут использоваться разные стили. Браузер, поддерживающий таблицы стилей, будет следовать их порядку (как по каскаду), интерпретируя информацию стилей.

# Базовый синтаксис

Селектор

h1

Описание

{color:blue; font-size:12px;}

Свойство 1

Значение

Свойство 2

Значение

**Селектор** — это HTML-элемент, для которого добавляются параметры форматирования. В качестве селектора выступают **теги, классы и идентификаторы**.

Каждое описание содержит **свойство** и **значение**, каждое свойство должно иметь значение.

Стилевые **свойства** **разделяются** между собой **точкой с запятой**, в конце этот символ можно опустить.

**CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции.**

**Приоритет имеет значение, указанное в коде ниже.**

# Базовый синтаксис

## Форма записи

Можно записывать так:

```
p {color:red;text-align:center;}
```

```
td { background: olive; }  
td { color: white; }  
td { border: 1px solid black; }
```

Или так:

```
p {  
  color:red;  
  text-align:center;  
}
```

```
td {  
  background: olive;  
  color: white;  
  border: 1px solid black;  
}
```

# Базовый синтаксис

## CSS комментарии

Чтобы выделить часть CSS документа как комментарий, используется конструкция `/* ... */`

**`/* Это комментарий */`**

```
/*  
    Стиль для сайта  
*/  
  
div {  
    width: 200px; /* Ширина блока */  
    margin: 10px; /* Поля вокруг элемента */  
    float: left; /* Обтекание по правому краю */  
}
```

# Применение стилевых правил



# Применение стилевых правил

## Связанные стили

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Стили</title>
    <link rel="stylesheet" type="text/css" href="mysite.css">
    <link rel="stylesheet" type="text/css" href="main.css">
  </head>
  <body>
    <h1>Заголовок</h1>
    <p>Текст</p>
  </body>
</html>
```

### Содержимое файла mysite.css

```
h1 {
  color: #000080;
  font-size: 200%;
  font-family: Arial, Verdana, sans-serif;
  text-align: center;
}
p {
  padding-left: 20px;
}
```

# Применение стилевых правил

## Глобальные стили

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Глобальные стили</title>
    <style type="text/css">
      h1 {
        font-size: 120%;
        font-family: Verdana, Arial, Helvetica, sans-serif;
        color: #333366;
      }
    </style>
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

# Применение стилевых правил

## Внутренние стили

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Внутренние стили</title>
  </head>
  <body>
    <p style="font-size: 120%; font-family: monospace;
    color: #cd66cc">Пример текста</p>
  </body>
</html>
```



# Применение стилевых правил

**Внутренние стили** рекомендуется применять на сайте ограниченно или вообще отказаться от их использования. Дело в том, что добавление таких стилей увеличивает общий объем файлов, что ведет к повышению времени их загрузки в браузере, и усложняет редактирование документов для разработчиков.

Использование **связанных стилей** является наиболее универсальным и удобным методом добавления стиля на сайт. Ведь стили хранятся в одном файле, а в HTML-документах указывается только ссылка на него.

Все описанные методы использования CSS могут применяться как самостоятельно, так и в сочетании друг с другом. В этом случае необходимо помнить об их иерархии. **Первым всегда применяется внутренний стиль, затем глобальный стиль и в последнюю очередь связанный стиль.**

# Классы и идентификаторы

**Классы** применяют, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега.

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

```
<style type="text/css">
  P { /* Обычный абзац */
    text-align: justify; /* Выравнивание текста по ширине */
  }
  P.cite { /* Абзац с классом cite */
    color: navy; /* Цвет текста */
    margin-left: 20px; /* Отступ слева */
    border-left: 1px solid navy; /* Граница слева */
    padding-left: 15px; /* Расстояние от линии */
  }
</style>
```

```
<body>
  <p>Обычный абзац</p>
  <p class="cite">Абзац со стилем</p>
</body>
```

# Классы и идентификаторы

Можно, также, использовать классы без указания тега:

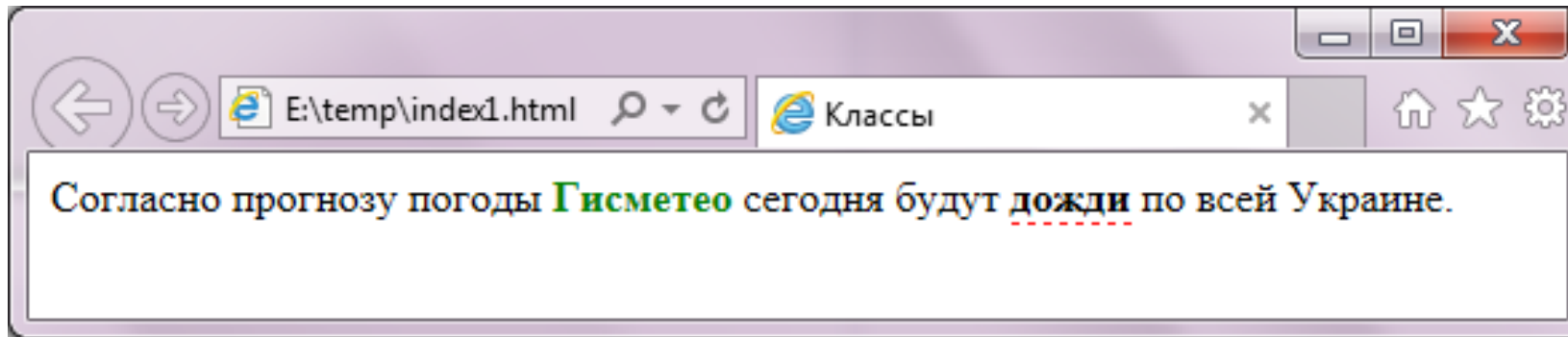
**.Имя класса { свойство1: значение; свойство2: значение; ... }**

При такой записи, класс можно применять к любому тегу:

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Классы</title>
  <style type="text/css">
    .gost {
      color: green; /* Цвет текста */
      font-weight: bold; /* Жирное начертание */
    }
    .term {
      border-bottom: 1px dashed red; /* Подчеркивание под текстом */
    }
  </style>
</head>
<body>
  <p>Согласно прогнозу погоды <span class="gost">Гисметео</span> сегодня будут
    <b class="term">дожди</b> по всей Украине.
  </p>
</body>
```

# Классы и идентификаторы

Результат в браузере:



Классы **удобно использовать**, когда нужно **применить стиль** к разным элементам веб-страницы: **ячейкам таблицы, ссылкам, абзацам и др.**

К любому тегу **одновременно можно добавить несколько классов**, перечисляя их **в параметре class через пробел**. В этом случае к элементу применяется стиль, описанный в правилах для каждого класса.

# Классы и идентификаторы

## Идентификатор ( «ID селектор» )

**определяет уникальное имя элемента**, которое используется для изменения его стиля и обращения к нему через скрипты.

**#Имя идентификатора { свойство1: значение; свойство2: значение; ... }**

**Обращение к идентификатору** происходит аналогично классам, но в качестве ключевого слова у тега используется параметр `id`, значением которого выступает имя идентификатора.

**<тег id="Имя идентификатора">**

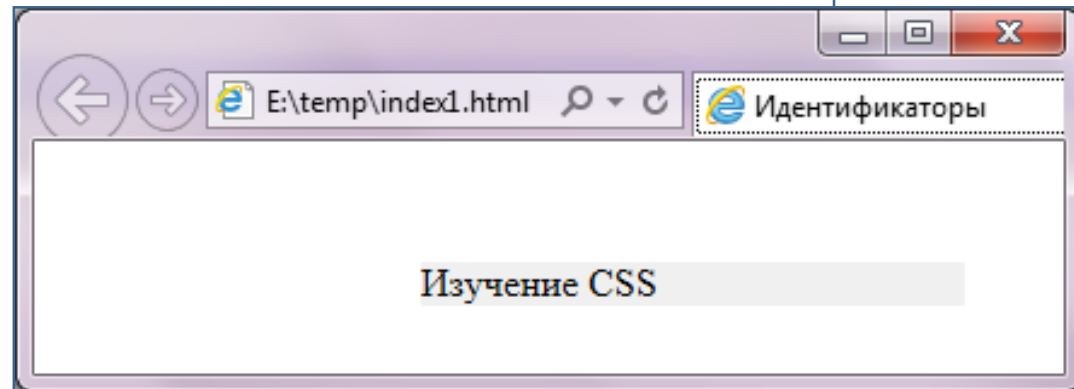
Идентификаторы можно применять к конкретному тегу:

**Тег#Имя идентификатора { свойство1: значение; свойство2: значение; ... }**

# Классы и идентификаторы

## Пример использования идентификатора:

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Идентификаторы</title>
  <style type="text/css">
    #help {
      position: absolute; /* Абсолютное позиционирование */
      left: 160px; /* Положение элемента от левого края */
      top: 50px; /* Положение от верхнего края */
      width: 225px; /* Ширина блока */
      padding: 5px; /* Поля вокруг текста */
      background: #f0f0f0; /* Цвет фона */
    }
  </style>
</head>
<body>
  <div id="help">
    Изучение CSS!
  </div>
</body>
```



# Классы и идентификаторы

## Идентификаторы

- В коде документа каждый идентификатор **уникален** и **должен быть включён лишь один раз**.
- Имя идентификатора **чувствительно к регистру**.
- Через метод **getElementById** можно получить доступ к элементу по его идентификатору и изменить свойства элемента.
- Стил для идентификатора имеет **приоритет выше, чем у классов**.

## Классы

- Классы могут использоваться в коде **неоднократно**.
- Имена классов **чувствительны к регистру**.
- Классы можно комбинировать между собой, добавляя **несколько классов к одному тегу**.

# Наследование свойств

Наследование — перенос правил форматирования для элементов, находящихся внутри других.

Такие элементы являются дочерними, и они наследуют некоторые стилевые свойства своих родителей, внутри которых располагаются.

Наследование позволяет задавать значения некоторых свойств единожды, определяя их для родителей верхнего уровня.

Допустим,  
требуется  
установить  
цвет и шрифт  
для основного  
текста:

```
<style type="text/css">
  BODY {
    font-family: Arial, Helvetica, sans-serif; /* Гарнитура шрифта */
    color: navy; /* Синий цвет текста */
  }
</style>
...
<body>
  <p>Цвет текста этого абзаца синий.</p>
</body>
```



# Каскадирование

**Каскадирование** — одновременное применение разных стилевых правил к элементам документа — с помощью подключения нескольких стилевых файлов, наследования свойств и других методов.

**Приоритеты (в списке по нарастанию):**

**Стиль браузера**

**Стиль автора**

**Стиль пользователя**

**Стиль автора с добавлением !important**

**Стиль пользователя с добавлением !important**

# Специфичность

Если к одному элементу одновременно применяются противоречивые стилевые правила, то более высокий приоритет имеет правило, у которого значение специфичности селектора больше.

**Специфичность** - это условная величина, вычисляемая следующим образом:

- за каждый **идентификатор** начисляется **100** (в примере - a)
- за каждый класс и **псевдокласс** начисляется **10** (в примере - b)
- за каждый **селектор тега** и **псевдоэлемент** начисляется **1** (в примере – c).

Складывая указанные значения в определенном порядке, получим значение специфичности для данного селектора.

<b>ul ol+li</b>	<b>{ /* a=0 b=0 c=3 -&gt; специфичность = 3 */</b>
<b>ul li.red</b>	<b>{ /* a=0 b=1 c=2 -&gt; специфичность = 12 */</b>
<b>li.red.level</b>	<b>{ /* a=0 b=2 c=1 -&gt; специфичность = 21 */</b>
<b>#t34</b>	<b>{ /* a=1 b=0 c=0 -&gt; специфичность = 100 */</b>

**Встроенный стиль, добавляемый к тегу через атрибут style, имеет специфичность 1000, поэтому всегда перекрывает связанные и глобальные стили. Однако добавление !important перекрывает в том числе и встроенные стили.**

# Виды селекторов

В качестве селектора может выступать **любой тег HTML** для которого **определяются правила форматирования**, такие как: **цвет, фон, размер и т.д.**

**Тег { свойство1: значение; свойство2: значение; ... }**

## Контекстные селекторы (селекторы потомков)

Для того, чтобы одновременно установить стиль для отдельного тега, а также для тега, который находится внутри другого.

**Контекстный селектор** состоит из простых селекторов разделенных пробелом:

**Тег1 Тег2 { ... }**

**Стиль будет применяться к Тегу2** когда он размещается внутри Тега1.

# Виды селекторов

Часто приходится **вкладывать одни теги внутрь других**. Чтобы стили для этих тегов использовались корректно, **помогут селекторы, которые работают только в определенном контексте**.

Например, задать стиль для тега `<b>` только когда он располагается внутри контейнера `<p>`.

Таким образом можно одновременно установить стиль для отдельного тега, а также для тега, который находится внутри другого.

# Виды селекторов

## Соседние селекторы

Для управления стилем **соседних элементов** используется символ плюса (+), который устанавливается между двумя селекторами:

Селектор 1 + Селектор 2 { Описание правил стиля }

Стиль при такой записи применяется к Селектору 2, но только в том случае, если он является **соседним** для Селектора 1 и **следует сразу после него**.

## Дочерние селекторы

Дочерним селектором считается такой, который в дереве элементов находится **прямо внутри родительского элемента**.

Селектор 1 > Селектор 2 { Описание правил стиля }

Стиль применяется к Селектору 2, но только в том случае, если он является **дочерним** для Селектора 1.

# Виды селекторов

По своей логике дочерние селекторы похожи на селекторы контекстные.  
**Разница** между ними следующая:

- Стиль к дочернему селектору применяется только в том случае, когда он является прямым потомком, иными словами, непосредственно располагается внутри родительского элемента.
- Для контекстного селектора допустим любой уровень вложенности.

# Виды селекторов

## Универсальный селектор

Применяется, чтобы установить **одновременно один стиль для всех элементов веб-страницы**, например, задать шрифт или начертание текста.

Для обозначения универсального селектора применяется символ звездочки:

**\* { Описание правил стиля }**

В некоторых случаях указывать универсальный селектор не обязательно. Так, например, записи \*.class и .class являются идентичными по своему результату.



# Селекторы атрибутов

Селекторы атрибутов позволяют установить стиль по присутствию определенного атрибута тега или его значения.

## Простой селектор атрибута

Устанавливает стиль для элемента, если задан специфичный атрибут тега. Его значение в данном случае не важно.

```
[атрибут] { Описание правил стиля }  
Селектор[атрибут] { Описание правил стиля }
```

Стиль применяется к тем тегам, внутри которых добавлен указанный атрибут.

```
<style type="text/css">  
  Q {  
    font-style: italic; /* Курсивное начертание */  
    quotes: "\00AB" "\00BB"; /* Меняем вид кавычек в цитате */  
  }  
  Q[title] {  
    color: maroon; /* Цвет текста */  
  }  
</style>
```



# Селекторы атрибутов

## Атрибут со значением

Устанавливает стиль для элемента в том случае, **если задано определенное значение специфичного атрибута.**

[атрибут="значение"] { Описание правил стиля }  
Селектор[атрибут="значение"] { Описание правил стиля }

В первом случае стиль применяется ко всем тегам, которые содержат указанное значение. А во втором — только к определенным селекторам.

```
<title>Селекторы атрибутов</title>
<style type="text/css">
  A[target="_blank"] {
    background: url(blank.gif) no-repeat 0 2px; /* Параметры фонового рисунка */
    padding-left: 15px; /* Смещаем текст вправо */
  }
</style>
</head>
<body>
  <p><a href="link1.html">Обычная ссылка</a> |
    <a href="link2" target="_blank">Ссылка в новом окне</a></p>
</body>
```

# Селекторы атрибутов

## Значение атрибута начинается с определенного текста

Устанавливает стиль для элемента в том случае, если значение атрибута тега начинается с указанного текста.

[атрибут^="значение"] { Описание правил стиля }  
Селектор[атрибут^="значение"] { Описание правил стиля }

В первом случае стиль применяется ко всем элементам, у которых значение атрибута начинаются с указанного текста. А во втором — только к определенным селекторам.

```
<style type="text/css">
  A[href^="http://"] {
    font-weight: bold; /* Жирное начертание */
  }
</style>
</head>
<body>
  <p><a href="link1.html">Обычная ссылка</a> |
  <a href="http://ya.ru" target="_blank">Внешняя ссылка на сайт ya.ru</a></p>
</body>
</html>
```

# Селекторы атрибутов

Значение атрибута оканчивается определенным текстом

Устанавливает стиль для элемента в том случае, **если значение атрибута оканчивается указанным текстом.**

[атрибут\$="значение"] { Описание правил стиля }  
Селектор[атрибут\$="значение"] { Описание правил стиля }

В первом случае стиль применяется ко всем элементам у которых значение атрибута завершается заданным текстом. А во втором — только к определенным селекторам.

```
<style type="text/css">
A[href$=".ru"] { /* Если ссылка заканчивается на .ru */
  background: url(ru.gif) no-repeat; /* Параметры фонового рисунка */
  padding-left: 10px; /* Смещаем текст вправо */
}
A[href$=".com"] { /* Если ссылка заканчивается на .com */
  background: url(com.gif) no-repeat;
  padding-left: 10px;
}
</style>
</head>
<body>
<p><a href="http://www.yandex.com">Yandex.Com</a> |
<a href="http://www.yandex.ru">Yandex.Ru</a></p>
```

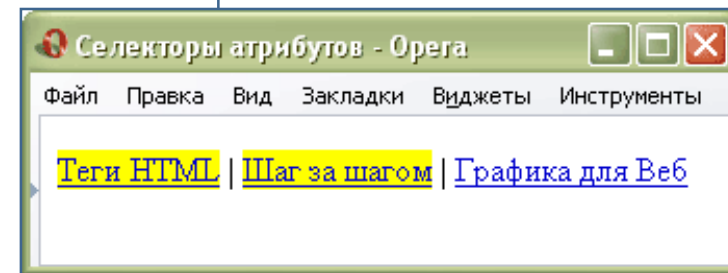
# Селекторы атрибутов

## Значение атрибута содержит указанный текст

Возможны варианты, когда стиль следует применить к тегу с определенным атрибутом, при этом частью его значения является некоторый текст. При этом точно не известно, в каком месте значения включен данный текст — в начале, середине или конце.

[атрибут\*="значение"] { Описание правил стиля }  
Селектор[атрибут\*="значение"] { Описание правил стиля }

```
<style type="text/css">
  [href*="htmlbook"] {
    background: yellow; /* Желтый цвет фона */
  }
</style>
</head>
<body>
  <p><a href="http://www.htmlbook.ru/html/">Теги HTML</a> |
  <a href="http://stepbystep.htmlbook.ru">Шаг за шагом</a> |
  <a href="http://webimg.ru">Графика для Веб</a></p>
</body>
</html>
```



# Селекторы атрибутов

## Одно из нескольких значений атрибута

Некоторые значения атрибутов могут перечисляться через пробел, например имена классов. Чтобы **задать стиль при наличии в списке требуемого значения** применяется следующий синтаксис:

```
[атрибут~="значение"] { Описание правил стиля }  
Селектор[атрибут~="значение"] { Описание правил стиля }
```

Стиль применяется в том случае, если **у атрибута имеется указанное значение или оно входит в список значений, разделяемых пробелом.**

```
<style type="text/css">  
  [class~="block"] h3 { color: green; }  
</style>  
</head>  
<body>  
  <div class="block tag">  
    <h3>Заголовок</h3>  
  </div>
```

# Селекторы атрибутов

## Дефис в значении атрибута

Для изменения стиля элементов, в значении которых применяется дефис, следует воспользоваться следующим синтаксисом:

[атрибут]="значение" { Описание правил стиля }  
Селектор[атрибут]="значение" { Описание правил стиля }

Стиль применяется к элементам, у которых атрибут начинается с указанного значения или с фрагмента значения, после которого идет дефис.

```
<style type="text/css">
  DIV[class|="block"] {
    background: #306589; /* Цвет фона */
    color: #acdb4c; /* Цвет текста */
    padding: 5px; /* Поля */
  }
  DIV[class|="block"] A {
    color: #fff; /* Цвет ссылок */
  }
</style>
</head>
<body>
  <div class="block-menu-therm">
    <h2>Термины</h2>
    <div class="content">
      <ul class="menu">
        <li><a href="t1.html">Буквица</a></li>
```

# Псевдоклассы

определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа.

Селектор:Псевдокласс { Описание правил стиля }

Пример: текстовая ссылка, которая меняет свой цвет при наведении на неё курсора мыши.

При использовании **псевдоклассов** браузер **не перегружает** текущий документ, поэтому с помощью псевдоклассов можно получить разные **динамические эффекты** на странице.

Условно все псевдоклассы делятся на три группы:

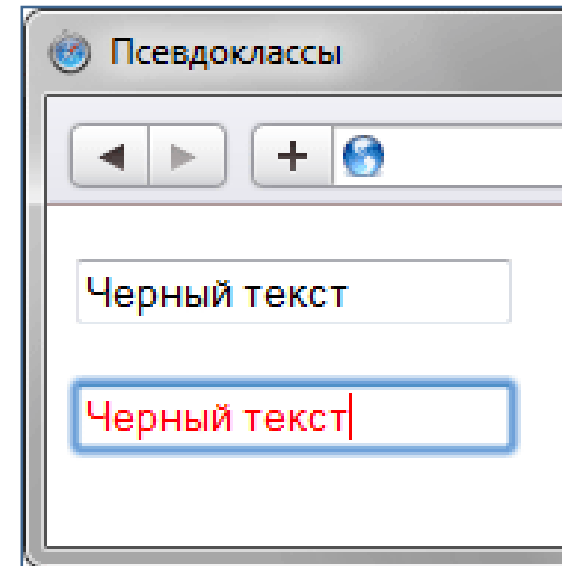
- **определяющие состояние элементов;**
- **имеющие отношение к дереву элементов;**
- **указывающие язык текста.**

# Псевдоклассы

## :focus

**Применяется** к элементу **при получении** им **фокуса**. Например, для текстового поля формы получение фокуса означает, что курсор установлен в поле, и с помощью клавиатуры можно вводить в него текст.

```
<style>
  INPUT:focus {
    color: red; /* Красный цвет текста */
  }
</style>
</head>
<body>
  <form action="">
    <p><input type="text" value="Черный текст"></p>
    <p><input type="text" value="Черный текст"></p>
  </form>
```





# Псевдоклассы

## :active

Происходит **при активации пользователем элемента**. Например, ссылка становится активной, если навести на неё курсор и щёлкнуть мышкой. Несмотря на то, что активным может стать практически любой элемент веб-страницы, псевдокласс :active **используется преимущественно для ссылок**.

## :link

Применяется к **непосещенным ссылкам**, т. е. таким ссылкам, на которые пользователь ещё не нажимал. Браузер некоторое время сохраняет историю посещений, поэтому ссылка может быть помечена как посещенная хотя бы потому, что по ней был зафиксирован переход ранее.

# Псевдоклассы

## :visited

Данный псевдокласс применяется к **посещённым ссылкам**. Обычно такая ссылка меняет свой цвет по умолчанию на фиолетовый, но с помощью стилей цвет и другие параметры можно задать самостоятельно.

## :hover

Псевдокласс :hover активизируется, когда **курсор мыши находится в пределах элемента, но щелчка по нему не происходит**.

**a:hover** должен быть расположен после **a:link** и **a:visited**!  
**a:active** должен быть расположен после **a:hover**!  
**Псевдоклассы не чувствительны к регистру.**

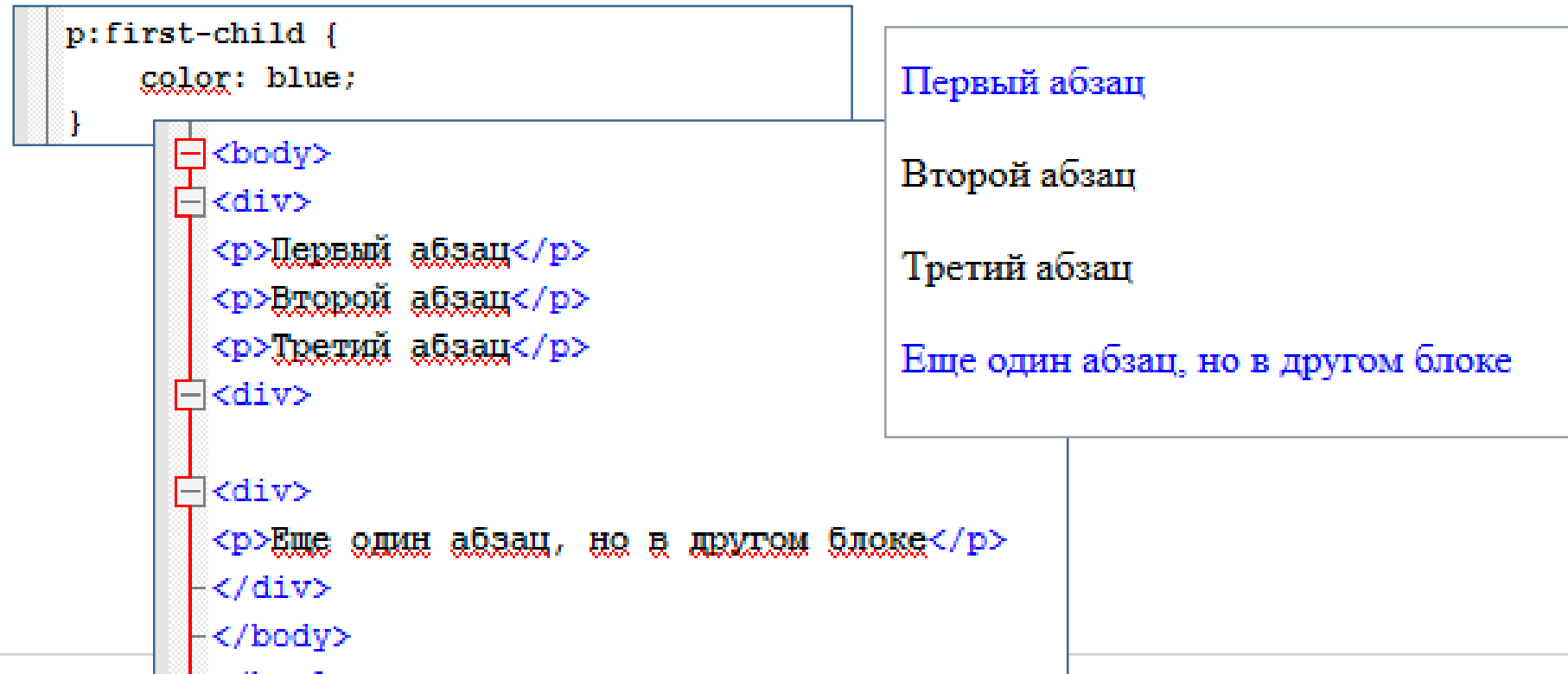
# Псевдоклассы

```
a {  
    color: red; /* Цвет непосещенных ссылок */  
}  
  
a:visited {  
    color: blue; /* Цвет посещенных ссылок */  
}  
  
a:hover {  
    color: grey; /* Цвет ссылок при наведении на них курсора мыши */  
}  
  
a:active {  
    color: black; /* Цвет активных ссылок */  
}
```

# Псевдоклассы

## :first-child

Применяется к **первому дочернему элементу селектора**, который расположен в дереве элементов документа.



# Псевдоклассы

## :lang

**Определяет язык**, который используется в документе или его фрагменте. В коде HTML язык устанавливается через атрибут lang, он обычно добавляется к тегу <html>. С помощью псевдокласса :lang можно задавать определённые настройки, характерные для разных языков, например, вид кавычек в цитатах.

В качестве языка могут выступать следующие значения: **ru** — русский; **en** — английский ; **de** — немецкий ; **fr** — французский; **it** — итальянский.

**Больше псевдо-классов:**

[http://www.w3schools.com/css/css\\_pseudo\\_classes.asp](http://www.w3schools.com/css/css_pseudo_classes.asp)

# Псевдоэлементы

Псевдоэлемент	Пример
<u>::after</u>	p::after
<u>::before</u>	p::before
<u>::first-letter</u>	p::first-letter
<u>::first-line</u>	p::first-line

# Псевдоэлементы

Селектор::Псевдоэлемент { Описание правил стиля }

## ::after

Применяется для вставки назначенного контента после содержимого элемента. Этот псевдоэлемент работает совместно со стилевым свойством **content**, которое определяет содержимое для вставки.

```
<p>HyperText Markup Language</p>
<p>Cascading Style Sheets</p>
```

```
<p><b>Важно:</b> Чтобы этот псевдоэлемент работал в IE8, DOCTYPE должен быть указан, <br>
и нужно использовать старый синтаксис :after вместо ::after.</p>
```

```
<style>
p::after {
    content: " - Читать далее...";
    background-color: yellow;
    color: red;
    font-weight: bold;
}
</style>
```

# Псевдоэлементы

HyperText Markup Language - Читать далее...

Cascading Style Sheets - Читать далее...

**Важно:** Чтобы этот псевдоэлемент работал в IE8, DOCTYPE должен быть указан, и нужно использовать старый синтаксис :after вместо ::after. - Читать далее...

## ::before

По своему действию **::before** аналогичен псевдоэлементу **::after**, но вставляет контент **до** содержимого элемента.



# Псевдоэлементы

## ::first-letter

Определяет **стиль первого символа в тексте элемента**, к которому добавляется. Это позволяет создавать в тексте буквицу и выступающий инициал.

## ::first-line

Определяет **стиль первой строки блочного текста**. Длина этой строки зависит от многих факторов, таких как используемый шрифт, размер окна браузера, ширина блока, языка и т.д.

# Основные CSS свойства

Свойство	Название
Цвет	color, background-color
Интервал	margin, padding, margin-left, margin-right, margin-top, margin-bottom
Границы	border-width, border-style, border-color, border (для установки ширины, стиля и цвета границ за один прием)
Выравнивание текста	text-align, text-indent, word-spacing, letter-spacing, line-height, white-space
Шрифты	font-family, font-size, font-weight, font-style, font-variant, text-decoration, @font-face (для использования вычурных шрифтов)
Размер	width, height
Позиция	position, left, right, float, center
Графика	background-image, background-repeat, background-position

# Основные CSS свойства

## Для форматирования текста

### color

Данный стиль **задает цвет текста**. Для задания цвета можно использовать как хекс-значение цвета (*color:#FFF*), так и ряд ключевых слов (*color:black, color:red ...*)

### text-align

Данный стиль **задает выравнивание текста внутри родительского блока**. Может иметь значения *left, right, center*. Есть еще значение стиля - *justify*, которое выравнивает текст по всей ширине родительского блока. Среди веб-дизайнеров стиль *justify* считается плохим тоном, т.к. выравнивание по всей ширине родительского блока приводит к появлению пробелов различной длины, что сильно ухудшает читабельность. Выравнивание, заданное свойством *text-align*, распространяется так же на графические элементы внутри блока.

# Основные CSS свойства

## Для форматирования текста

### line-height

Данный стиль задает **расстояние между строк в текстовом блоке** (или, иными словами, изменяет высоту строки текста, еще это называется «интерлиньяж»). Порой шрифт значительно приятнее смотрится при увеличении значения *line-height*, заданного по умолчанию.

Значение данного свойства задается в процентах (100%, 150% ...), множителем (1 - интерлиньяж по умолчанию, 1.5 — увеличен в полтора раза) или точным значением в пикселах (10px, 1.5 em...).

### letter-spacing

Межсимвольное расстояние. Значение данного свойства указывает в единицах длины (пиксели, дюймы, pt), либо относительные единицы — em.

# Основные CSS свойства

## Для форматирования текста

### font

Универсальное свойство, которое позволяет одновременно задать несколько характеристик шрифта и текста.

В качестве **обязательных** значений свойства font указывается **размер шрифта и его семейство**. Остальные значения являются опциональными и задаются при желании. Для подробного ознакомления смотрите информацию о каждом свойстве отдельно.

**font:** [font-style||font-variant||font-weight]  
font-size [/line-height] font-family | inherit

```
p { font: 12pt/10pt sans-serif; }
```

```
p { font: normal small-caps 12px/14px fantasy;
```

```
p { font: bold italic 110% serif; }
```

# Основные CSS свойства

## Для форматирования текста

### font-family

Данный стиль служит для задания гарнитуры шрифта. Название шрифтов перечисляются через запятую, в случае, если название состоит более чем из одно слово необходимо использовать кавычки.

**font-family:** (шрифт без засечек), Arial, Helvetica, sans-serif;  
**font-family:** (шрифт с засечками), «Times New Roman», Times, serif;  
**font-family:** (моноширинный шрифт), «Courier New», Courier, monospaced;

### font-weight

При желании сделать текст блока **жирным** — используйте стиль ***font-weight:bold***.

Если вы наоборот хотите **убрать жирное выделение** — тут все просто ***font-weight:normal***

# Основные CSS свойства

## Для фона

В качестве фона можно задать картинку: ***background-image***; либо просто задать фоновый цвет: ***background-color***. Фон может повторяться (по оси X или Y) - ***background-repeat***. А так же фон можно смещать — ***background-position***.

CSS2.1	<code>background: [<u>background-attachment</u>    <u>background-color</u>    <u>background-image</u>    <u>background-position</u>    <u>background-repeat</u>]   inherit</code>
CSS3	<code>background: [&lt;фон&gt;, ]* &lt;последний_фон&gt;</code>

```
background: url(images/hand.png) repeat-y #fc0;
```

# Основные CSS свойства

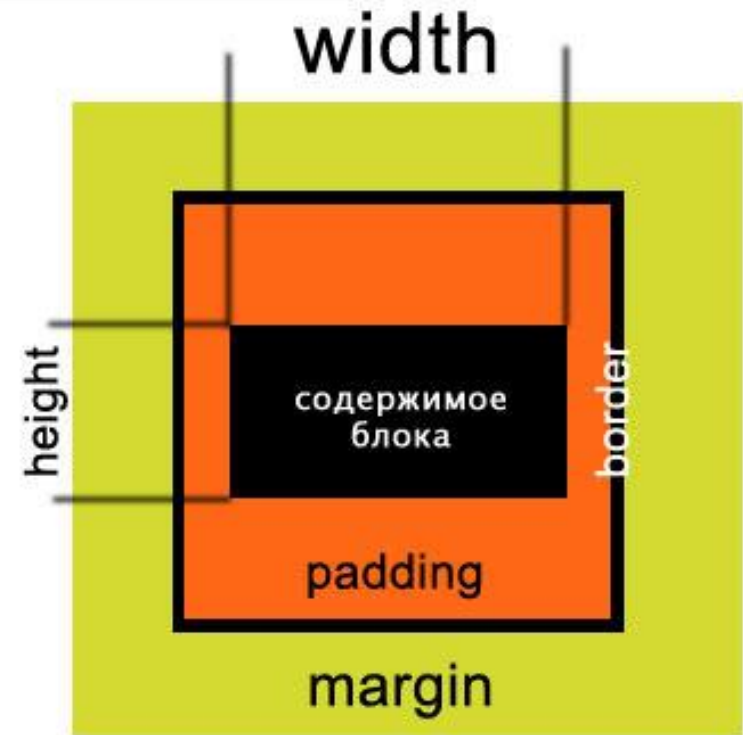
## Для отступов

### padding

Отступ внутри блока (значение в пикселах)

### margin

Отступы от границ блока



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

Данный стиль убирает отступы, задаваемые в браузере по умолчанию.  
Рекомендовано начинать написание файла CSS именно с этого стиля.



# Основные CSS свойства

## Позиционирование элементов

```
position: absolute | fixed | relative | static | inherit
```

### **position: absolute**

Данное свойство вырывает блок из нормального потока формирования страницы. И далее блок позиционируется относительно верхнего угла браузера путем указания свойств *left* и *top* в процентах или пикселях.

**Нормальный поток** — это когда блоки идут на странице один за одним, т.е. `div` под `div`-ом. В случае применения свойства ***position: absolute*** блок накладывается поверх остальных блоков нормального потока.

Используя абсолютное позиционирование, дабы избежать непредвиденных обстоятельств, необходимо задать значения ширины и высоты блока в пикселях (свойства *width* и *height*).

# Основные CSS свойства

## Позиционирование элементов

### **position:relative**

Разновидность абсолютного позиционирования. В данном случае блок смещается заданием значений *left* и *top* относительно места своего нормального положения.

Иными словами, блок выводится там, где он и должен быть в нормальном потоке и сдвигается на заданные значения.

Если у родительского блока указан стиль *position:relative*, то вложенный блок с указанным стилем *position:absolute* будет смещаться относительно левого верхнего угла родительского блока.

# Основные CSS свойства

## Позиционирование элементов

**float:left**

В случае, если необходимо разместить два блока *div* в одну линию друг за другом, у первого блока указывается стиль *float:left* (это означает что своей левой границей данный блок должен прилипнуть к предыдущему блоку в потоке). Первый блок оказывается прижатым, например к левой границе окна браузера. Если следующему блоку в потоке указать тоже самое значение в стилях, то два блока будут выводиться на одной линии. Первый блок будет прилипать к левой границе экрана, а второй, своей левой границей, к правой границе предыдущего блока.

Если у второго блока указать значения стиля *float:right*, то оба блока все так же окажутся расположенными на одной линии, но теперь первый блок будет прилипать к левой границе окна браузера, а правый — своей правой стороной к правой границе окна браузера.

# Материалы для самостоятельного изучения

- <http://www.w3schools.com/>
- <http://caniuse.com/>
- <http://htmlbook.ru/>
- <https://html5book.ru/>
- <https://www.codecademy.com>
- <https://htmlacademy.ru/>

Q&A

# Thank You

