HTML элементы. Полный справочник тегов

HTML теги упорядоченные по алфавиту

= Новый в HTML5.

Тег	Описание
<!--...-->	Определяет комментарий
<!DOCTYPE>	Определяет тип документа
<a>	Определяет гиперссылку
<abbr>	Определяет абревиатуру или акроним
<acronym>	Не поддерживается в HTML5. Используйте <abbr> вместо этого. Определяет акроним
<address>	Определяет контактную информацию автора / владельца документа
<applet>	Не поддерживается в HTML5. Используйте <embed> или <object> вместо этого. Определяет встроенный аплет
<area>	Определяет область внутри карты изображения
<article>	Определяет статью
<aside>	Определяет содержание, кроме содержания страницы (в стороне)
<audio>	Определяет звуковой контент
<b>	Определяет жирный текст
<base>	Указывает базовый URL-адрес / цель для всех относительных URL-адресов документа
<basefont>	Не поддерживается в HTML5. Используйте CSS вместо этого. Отпределяет цвет, размер и шрифт по умолчанию для всего текста документа
<bdi>	Изолирует часть текста, который может быть отформатирован в другом направлении от иного текста за его пределами
<bdo>	Переопределяет текущее направление текста
<big>	Не поддерживается в HTML5. Используйте CSS вместо этого. Определяет увеличенный текст
<blockquote>	Определяет раздел, который цитируется с другого источника
<body>	Определяет тело документа
 	Определяет разрыв строки
<button>	Определяет кнопку, которую можно нажимать
<canvas>	Используется для рисования на лету, с помощью сценариев (обычно на JavaScript)
<caption>	Определяет подпись к таблице
	Не поддерживается в HTML5. Используйте CSS вместо этого. Определяет центрирование текста
<cite>	Определяет название произведения при цитировании из него
<code>	Определяет фрагмент компьютерного кода
<col>	Указывает свойства столбцов для каждого столбца в элементе <colgroup>
<colgroup>	Определяет группу с одного или нескольких столбцов в таблице для форматирования
<data>	Связывает заданный контент с машиночитаемым переводом
<datalist>	Определяет список предварительно определённых параметров управления вводом
<dd>	Определяет описание / значение термина в списке описания
<del>	Определяет удалённый с документа текст
<details>	Определяет дополнительные детали, которые пользователь может просматривать или прятать
<dfn>	Задаёт термин, для которого будет дано определение (definition)
<dialog>	Определяет диалоговый бокс или окно
<dir>	Не поддерживается в HTML5. Используйте вместо этого. Определяет список каталогов
	Определяет раздел (блочный) в документе
<dl>	Определяет список описаний
<dt>	Определяет термин / имя в списке описания
	Определяет семантически подчёркнутый текст (empharized)

<embed>	Определяет контейнер для внешнего (не HTML) приложения
<fieldset>	Группы связанных элементов в форме
<figcaption>	Определяет заголовок для элемента <figure>
<figure>	Определяет автономное содержание
<font>	Не поддерживается в HTML5. Используйте CSS вместо этого. Определяет шрифт, цвет и размер текста
<footer>	Определяет нижний колонтитул (футер) для документа или раздела
<form>	Определяет HTML форму для ввода пользователем
<frame>	Не поддерживается в HTML5. Определяет окно (фрейм) в наборе фреймов
<frameset>	Не поддерживается в HTML5. Определяет набор фреймов
<h1> to <h6>	Определяет HTML заголовки
<head>	Определяет информацию о документе
<header>	Определяет заголовок для документа или раздела
<hr>	Определяет тематическую смену контента
<html>	Определяет корень HTML-документа
<i>	Определяет часть текста альтернативным голосом или настроением
<iframe>	Определяет встроенный фрейм
<img>	Определяет изображение
<input>	Определяет элемент управления вводом
<ins>	Определяет текст, который был вставлен в документ
<kbd>	Определяет ввод с клавиатуры
<label>	Определяет метку для элемента <input>
<legend>	Определяет заголовок для элемента <fieldset>
	Определяет элемент списка
<link>	Определяет взаимосвязь между документом и внешним ресурсом (обычно используется для ссылки на внешние таблицы стилей)
<main>	Определяет основное содержание документа
<map>	Определяет карту изображения на стороне клиента
<mark>	Определяет помеченный (маркированный) / выделенный текст
<meta>	Определяет метаданные HTML документа
<meter>	Определяет скалярное измерение в пределах известного диапазона (датчик)
<nav>	Определяет навигационные ссылки (навигация по сайту)
<noframes>	Не поддерживается в HTML5. Определяет альтернативное содержание для пользователей, которые не поддерживают фреймы
<noscript>	Определяет альтернативное содержание для пользователей, которые не поддерживают скрипты на стороне клиента
<object>	Определяет встроенный объект
	Определяет упорядоченный (нумерованный) список
<optgroup>	Определяет группу соответствующих параметров в выпадающем списке (выпадающем меню)
<option>	Определяет параметр в выпадающем списке
<output>	Определяет результат расчёт (калькуляции)
<p>	Определяет параграф (абзац)
<param>	Определяет параметр для объекта
<picture>	Определяет контейнер для нескольких ресурсов изображения
<pre>	Определяет предварительно отформатированный текст
<progress>	Представляет ход выполнения задания
<q>	Определяет короткую цитату
<rp>	Определяет, что показывать в браузерах, которые не поддерживают ruby аннотации
<rt>	Определяет пояснения / произношение символов (для восточноазиатской типографики)
<ruby>	Определяет аннотацию ruby (для восточноазиатской типографики)
<s>	Определяет текст, который больше не является правильным
<samp>	Определяет исходные данные с компьютерной программы
<script>	Определяет скрипт на стороне клиента
<section>	Определяет раздел (секцию) в документе

<select>	Определяет выпадающий список
<small>	Определяет меньший текст
<source>	Определяет несколько медиа-ресурсов для медиа-элементов (<video> и <audio>)
	Определяет раздел (строчный) в документе
<strike>	Не поддерживается в HTML5. Используйте <del> или <s> вместо этого. Определяет перечеркнутый текст
	Определяет семантически важный текст
<style>	Определяет информацию о стиле в документе
<sub>	Определяет подстрчный текст (нижний индекс)
<summary>	Определяет видимый заголовок для элемента <details>
<sup>	Определяет надстрочный текст (верхний индекс)
<svg>	Определяет контейнер для графики SVG
<table>	Определяет таблицу
<tbody>	Группирует содержание тела в таблице
<td>	Определяет клетку (ячейку) в таблице
<template>	Определяет шаблон
<textarea>	Определяет многострочный элемент управления вводом (текстовая область)
<tfoot>	Группирует содержание нижнего колонтитула в таблице
<th>	Определяет ячейку заголовка в таблице
<thead>	Группирует содержание заголовка в таблице
<time>	Определяет дату / время
<title>	Определяет заголовок документа
<tr>	Определяет строку в таблице
<track>	Определяет текстовые дорожки для медиа-элементов (<video> и <audio>)
<tt>	Не поддерживается в HTML5. Используйте CSS вместо этого. Определяет текст телетайпа
<u>	Определяет текст, который должен быть стилистически отличным от обычного текста
	Определяет неупорядоченный (ненумерованный) список
<var>	Определяет переменную
<video>	Определяет видео или фильм
<wbr>	Определяет возможный разрыв строки

Сколько всего HTML тегов?

Как видно из вышеприведённой таблицы, по состоянию на 2019 год в спецификации **HTML5** официально утверждено и поддерживается **107 HTML тегов**. 12 тегов, приведённых в таблице, уже не поддерживаются в спецификации **HTML5**, но поддерживаются в большинстве браузеров. Даже если вы раньше использовали эти теги при создании своих веб-сайтов, они ещё будут работать во всех современных веб-браузерах. Но все-таки устаревшие **HTML теги** не рекомендуется использовать в новых веб-проектах. Лучше придерживаться рекомендаций **W3C** и официально утверждённой спецификации **HTML5**. Кроме того, стандарты постоянно меняются. Со временем некоторые теги могут исчезать, а новые теги появляются. Для того, чтобы оставаться в курсе всех изменений, которые происходят в веб-стандартах, необходимо посещать официальные веб-ресурсы, где все эти изменения публикуются и утверждаются, в т.ч. сайт Консорциума Всемирной Паутины **W3C**: https://www.w3.org/.

□ Home Next □

HTML5 Аудио

□ Prev Next □

Аудио в Интернете

До появления стандарта **HTML5** аудиофайлы можно было воспроизводить в веб-браузере лишь с помощью плагина (на подобие flash).

HTML5 элемент `<audio>` указывает стандартный способ встраивания аудио в веб-страницу.

Поддержка браузерами

Цифры в таблице определяют первую версию браузера, которая полностью поддерживает элемент `<audio>`.

Элемент					
<audio>	4.0	9.0	3.5	4.0	10.5

HTML элемент <audio>

Чтобы воспроизвести аудиофайл в HTML, используйте элемент `<audio>`:

Пример:

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Ваш веб-браузер не підтримує елемент audio.
</audio>
```

Попробуйте самі »

Результат:

Ваш веб-браузер не поддерживает элемент audio.

HTML Аудио - Как это работает

Атрибут `controls` добавляет элементы управления звуком, такие как воспроизведение, пауза и громкость.

Элемент `<source>` позволяет указать альтернативные аудиофайлы, с которых браузер может выбирать. Браузер будет использовать первый распознанный формат.

Текст между `<audio>` и `</audio>` тегами будет отображаться лишь в браузерах, которые не поддерживают элемент `<audio>`.

HTML Аудио - Поддержка браузерами

В **HTML5** есть три поддерживаемых аудиоформата: MP3, WAV и OGG.

Поддержка браузерами разных форматов:

Браузер	MP3	WAV	OGG
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

HTML Аудио - Типы медиа

Формат файла	Тип медиа
MP3	audio/mpeg
OGG	audio/ogg
WAV	audio/wav

HTML Аудио - Методы, свойства и события

HTML5 определяет методы, свойства и события DOM для элемента `<audio>`.

Это позволяет загружать, воспроизводить и приостанавливать аудио, а также устанавливать продолжительность и громкость.

Есть также события DOM, которые могут извещать вас, когда звук начинает воспроизводиться, приостанавливаться и т.д.

Для получения полной справки на DOM перейдите к HTML5 Аудио/Видео DOM Справочника.

HTML5 Аудио теги

Тег	Описание
<audio>	Определяет звуковой контент
<source>	Определяет множество мультимедийных ресурсов для элементов медиа, например <video> и <audio>

Prev Next

HTML5 Поддержка браузерами

Prev Next

Вы можете научить старые браузеры правильно обрабатывать **HTML5**.

HTML5 Поддержка браузерами

HTML5 поддерживается всеми современными браузерами.

Кроме того, все браузеры, старые и новые, автоматически обрабатывают неопознанные элементы как встроенные элементы. Благодаря чему вы можете "научить" старые браузеры обрабатывать "неизвестные" для него элементы HTML.

Вы даже можете научить IE6 (Windows XP 2001), как обрабатывать неизвестные элементы HTML.

Определение семантических элементов как блочных элементов

HTML5 определяет восемь новых **семантических** элементов. Все они являются **блочными** элементами.

Чтобы обеспечить корректное поведение в старых браузерах, нужно установить CSS свойство **display** для этих HTML элементов **block**:

```
header, section, footer, aside, nav, main, article, figure {
  display: block;
}
```

Как добавить новые элементы в HTML

Также можно добавить новые элементы в HTML-страницы с помощью обмана браузера.

Этот пример добавляет новый элемент с названием <myHero> на HTML страницу и определяет его стиль:

Пример:

```
<!DOCTYPE html>
<html>
<head>
<script>document.createElement("myHero")</script>
<style>
myHero {
  display: block;
  background-color: #dddddd;
  padding: 50px;
  font-size: 30px;
}
</style>
```

```
</head>
<body>

<h1>A Heading</h1>
<myHero>My Hero Element</myHero>

</body>
</html>
```
Попробуйте сами »

JavaScript оператор `document.createElement("myHero")` нужен для создания нового элемента в IE 9 и более ранних версиях.

Проблемы с Internet Explorer 8

Вы можете воспользоваться описанным выше решением для всех новых елементов **HTML5**.

При этом **IE8 (и более ранние версии) не позволяет стилизовать неизвестные элементы!**

Но, слава Богу, Sjoerd Visscher создал **HTML5Shiv**! HTML5Shiv - это обходной путь JavaScript для включения стилизации элементов HTML5 в версиях Internet Explorer до версии 9.

Вам будет необходим HTML5shiv для обеспечения совместимости для IE браузеров старших от IE 9.

Синтаксис для HTML5Shiv

HTML5Shiv располагается в пределах тега `<head>`.

HTML5Shiv - это файл JavaScript, на который ссылается тег `<script>`.

Вы можете использовать **HTML5Shiv**, если вы используете новые элементы **HTML5**, такие как: `<article>`, `<section>`, `<aside>`, `<nav>`, `<footer>`.

Вы можете загрузить последнюю версию **HTML5shiv** из github или по ссылке CDN версии на https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js

Синтаксис

```
<head>
  <!--[if lt IE 9]>
    <script src="/js/html5shiv.js"></script>
  <![endif]-->
</head>
```

HTML5Shiv Пример

Если вы не желаете загружать и сохранять **HTML5Shiv** на своём сайте, можно ссылаться на версию, которая находится на сайте CDN.

Скрипт **HTML5Shiv** должен быть расположен в элементе `<head>` после любых таблиц стилей:

Пример:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<!--[if lt IE 9]>
  <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
<![endif]-->
</head>
<body>

<section>
```

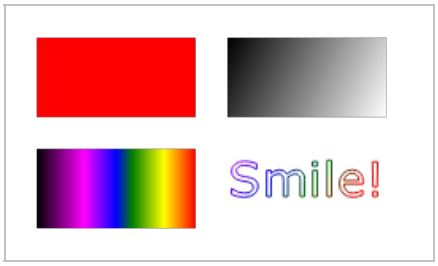
```
<h1>Famous Cities</h1>

<article>
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
</article>

<article>
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
</article>

<article>
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.</p>
</article>

</section>

</body>
</html>
```

Попробуйте сами »

□ Prev Next □

HTML5 Canvas

□ Prev Next □



HTML элемент `<canvas>` используется для рисования графики на веб-странице.

Графический объект слева создаётся с помощью `<canvas>`. Он показывает четыри элемента: красный прямоугольник, градиентный прямоугольник, многоцветный прямоугольник и многоцветный текст.

Что такое HTML Canvas?

HTML элемент `<canvas>` (*с анг. canvas - холст*) используется для рисования графики на лету с помощью JavaScript.

Элемент `<canvas>` является лишь контейнером для графики. На самом деле необходимо использовать JavaScript, чтобы нарисовать графику.

Canvas имеет несколько методов для рисования дорожек, полей, кругов, текста и добавления изображения.

Поддержка браузерами

Цифры в таблице определяют первую версию браузера, которая полностью поддерживает элемент `<canvas>`.

Элемент				
<canvas> 4.0	9.0	2.0	3.1	9.0

Canvas Примеры

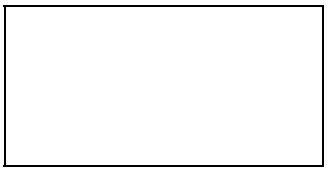
Canvas (холст) - это прямоугольная область на HTML странице. По умолчанию холст не имеет границы и содержания.

Разметка выглядит так:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Примечание: Всегда указывайте атрибут `id` (на который ссылается скрипт), атрибуты `width` (ширина) и `height` (высота) для определения размера холста. Чтобы добавить границу, воспользуйтесь атрибутом `style`.

Вот пример основного, пустого холста:

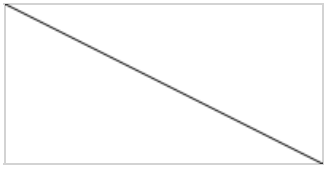


Пример:

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
```

Попробуйте сами »

Нарисовать линию

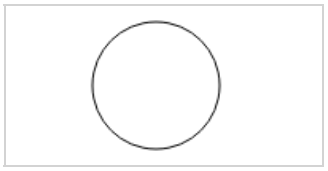


Пример:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
```

Попробуйте сами »

Нарисовать круг

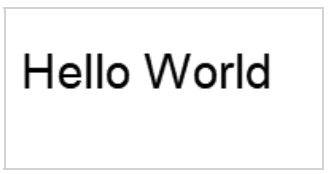


Пример:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
```

Попробуйте сами »

Нарисовать текст



Пример:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
```

```
ctx.fillText("Hello World", 10, 50);
```
Попробуйте сами »

Обведенный текст

Пример:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World", 10, 50);
```
Попробуйте сами »

Нарисовать линейный градиент

Пример:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// Create gradient
var grd = ctx.createLinearGradient(0, 0, 200, 0);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
```
Попробуйте сами »

Нарисовать круговой градиент

Пример:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// Create gradient
var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
```
Попробуйте сами »

Нарисовать изображение

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = document.getElementById("scream");
ctx.drawImage(img, 10, 10);
```

Попробуйте сами »

HTML Canvas учебник

Чтобы узнать больше про HTML `<canvas>`, посетите Учебник по HTML Canvas.

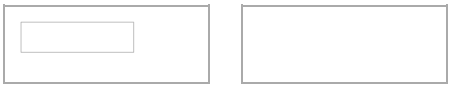
❮ Prev Next ❯

HTML5 Drag and Drop

❮ Prev Next ❯

Что такое Drag and Drop?

Drag and Drop - дословный перевод с английского языка - "Перетянуть и кинуть"



Перетяните мышкой изображение W3Schools с одного прямоугольника в другой прямоугольник.

Drag and Drop - Перетянуть и кинуть

Перетягивание (Drag and drop) - очень распространённая функция. Это когда вы "захватываете" мышкой объект и перетягиваете его в другое место.

В **HTML5** перетягивание является частью стандарта: любой элемент может быть перетянутым. Для этого используется JavaScript.

Поддержка браузерами

Цифры в таблице определяют первую версию браузера, которая полностью поддерживает перетягивание (Drag and Drop).

API				
Drag and Drop	4.0	9.0	3.5	6.0	12.0

HTML Drag and Drop Пример

Ниже приведён пример простого перетягивания:

Пример:

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}

function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
```

```
}

function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementByld(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height="69">

</body>
</html>
```

Попробуйте сами »

Это может показаться сложным, но давайте рассмотрим все разные части перетягивания.

Создать элемент для перетягивания

Прежде всего: Чтобы сделать элемент перетягиваемым, установите атрибут `draggable` на true (правда):

```
<img draggable="true" >
```

Чтобы перетянуть - ondragstart и setData()

Потом укажите, что должно произойти, когда элемент перетягивается.

В приведённом выше примере атрибут `ondragstart` вызывает функцию drag(event), которая указывает, какие данные необходимо перетягивать.

Метод `dataTransfer.setData()` устанавливает тип данных и значения перетягиваемых данных:

```
function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}
```

В этом случае тип данных - "text", а значение - id элемента перетягивания ("drag1").

Куда кинуть перетянувши - ondragover

Событие `ondragover` указывает, куда можно перетянуть и скинуть данные.

По умолчанию данные/элементы не могут быть скинуты в другие элементы. Чтобы позволить скидывание, мы должны предотвратить обработку элемента по умолчанию.

Это делается путём вызова метода `event.preventDefault()` для ondragover события:

```
event.preventDefault()
```

Сделать скидывание - ondrop

Когда вытягиваются перетягиваемые данные, происходит событие скидывания.

В приведённом выше примере атрибут ondrop вызывает функцию drop(event):

```
function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementByld(data));
}
```

Объяснение кода:

- Вызвать функцию preventDefault(), чтобы запретить обработку данных по умолчанию браузером (по умолчанию открывается как ссылка при скидывании);
- Получить перетянутые данные с помощью метода dataTransfer.getData(). Этот метод возвратит любые данные, которые были установлены в том же типе в методе setData();
- Перетянутые данные - это id перетянутого элемента ("drag1");
- Добавьте перетянутый элемент в элемент drop.

Больше примеров

Перетяните с помощью мышки изображение вперёд и назад

Как перетянуть (и скинуть) изображение вперёд и назад между двумя элементами (с одного прямоугольника в другой):

Попробуйте сами »

□ Prev Next □

HTML5 Геолокация

□ Prev Next □

---

px	em	процент
5px	0.3125em	31.25%
6px	0.3750em	37.50%
7px	0.4375em	43.75%
8px	0.5000em	50.00%
9px	0.5625em	56.25%
10px	0.6250em	62.50%
11px	0.6875em	68.75%
12px	0.7500em	75.00%
13px	0.8125em	81.25%
14px	0.8750em	87.50%
15px	0.9375em	93.75%
16px	1.0000em	100.00%
17px	1.0625em	106.25%
18px	1.1250em	112.50%
19px	1.1875em	118.75%
20px	1.2500em	125.00%
21px	1.3125em	131.25%
22px	1.3750em	137.50%
23px	1.4375em	143.75%
24px	1.5000em	150.00%
25px	1.5625em	156.25%

Попробуй это

Найдите позицию пользователя

API геолокации HTML используется для получения географического положения пользователя.

Поскольку это может нарушить конфиденциальность, позиция недоступна до тех пор, пока пользователь не согласиться с этим.

Примечание: Геолокация является наиболее точной для устройств с GPS, таких как смартфон.

Поддержка браузерами

Цифры в таблице определяют первую версию браузера, которая полностью поддерживает геолокацию.

API					
Geolocation	5.0 - 49.0 (http) 50.0 (https)	9.0	3.5	5.0	16.0

Примечание: Начиная с Chrome 50, API Геолокация будет работать только при безопасном соединении, таком как HTTPS. Если ваш сайт размещён на незащищенном источнике (например, HTTP), запросы на получение местоположения пользователей больше не будут функционировать.

Использование HTML Геолокации

Метод `getCurrentPosition()` используется для возвращения позиции пользователя.

Приведённый ниже пример возвращает широту и долготу позиции пользователя:

Пример:

```
<script>
var x = document.getElementById("demo");
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}

function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
</script>
```

Попробуйте сами »

Объяснение примера:

- Проверяем, поддерживается ли геолокация;
- Если поддерживается, запускаем метод getCurrentPosition(). Если нет, то отображаем сообщение пользователю;
- Если метод getCurrentPosition() успешен, он возвращает объект координат к функции, указанной в параметре (showPosition);
- Функция showPosition() выводит широту и долготу.

Приведённый выше пример является очень простым базовым скриптом геолокации без обработки ошибок.

Обработка ошибок и отказов

Другой параметр метода `getCurrentPosition()` используется для обработки ошибок. Он определяет функцию для запуска, если не удалось получить местонахождение пользователя:

Пример:

```
function showError(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED:
      x.innerHTML = "User denied the request for Geolocation."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML = "Location information is unavailable."
      break;
    case error.TIMEOUT:
      x.innerHTML = "The request to get user location timed out."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML = "An unknown error occurred."
      break;
```

```
  }
}
```
Попробуйте сами »

Отображение результата на карте

Чтобы отобразить результат на карте, нужен доступ к сервису карт, например, до Карт Google.

В приведённом ниже примере возвращённая широта и долгота используются для отображения местонахождения на карте Google (с помощью статического изображения):

Пример:

```
function showPosition(position) {
  var latlon = position.coords.latitude + "," + position.coords.longitude;

  var img_url = "https://maps.googleapis.com/maps/api/staticmap?center=
  "+latlon+"&zoom=14&size=400x300&sensor=false&key=YOUR_:KEY";

  document.getElementById("mapholder").innerHTML = "<img src='"+img_url+"'>";
}
```
Попробуйте сами »

Информация, которая касается местонахождения

На этой странице показано, как отобразить позицию пользователя на карте.

Геолокация также очень полезна для конкретной информации о местонахождении, например:

- Актуальная местная информация;
- Отображение интересных мест возле пользователя;
- Пошаговая навигация (GPS).

Метод *getCurrentPosition()* - возвращение данных

Метод `getCurrentPosition()` возвращает объект в случае успеха. Свойства широты, долготы и точности всегда возвращаются. Другие свойства возвращаются, если доступны:

Свойство	Возвращение
coords.latitude	Широта как десятичное число (всегда возвращается)
coords.longitude	Долгота как десятичное число (всегда возвращается)
coords.accuracy	Точность позиции (всегда возвращается)
coords.altitude	Высота в метрах выше среднего уровня моря (возвращается, если есть)
coords.altitudeAccuracy	Точность позиции высоты (возвращается, если доступна)
coords.heading	Направление в градусах по часовой стрелке от Севера (возвращается, если доступно)
coords.speed	Скорость в метрах в секунду (возвращается, если доступно)
timestamp	Дата / время ответа (возвращается, если доступно)

Объект геолокации - другие интересные методы

Объект геолокации - также другие интересные методы:

- `watchPosition()` - Возвращает текущую позицию пользователя и продолжает возвращать обновленную позицию, когда пользователь перемещается (например, GPS в автомобиле).
- `clearWatch()` - останавливает метод `watchPosition()`.

Приведенный ниже пример показывает метод `watchPosition()`. Для тестирования требуется точное устройство GPS (например, смартфон):

Пример:

```
<script>
var x = document.getElementById("demo");
```

```
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.watchPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}
function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
</script>
```

Попробуйте самі »

❮ Prev Next ❯

HTML5 Введение

❮ Prev Next ❯

Что такое HTML5?

HTML5 - это действующая версия языка **HTML**. В состав рабочей группы по **HTML5** вошли AOL, Apple, Google, IBM, Microsoft, Mozilla, Nokia, Opera и ещё несколько сотен других производителей.

Существует некоторая путаница о версиях, поскольку существуют две независимые группы разработчиков - WHATWG и W3C.

В WHATWG отказались от принципа «версийности» в пользу «вечной разработки» при принятии **HTML** спецификации. Такое решение было вызвано попыткой ускорить воплощение стандарта в жизнь, то есть разработчикам веб браузеров не нужно ждать, пока выйдет официально утверждённая версия спецификации (то есть, спецификация перейдет в состояние recomendation), они могут воплощать определённые части спецификации уже сейчас. Поэтому по версии WHATWG существует только одна спецификация, которая постоянно развивается - **HTML**.

Эти две группы работали в тандеме, в WHATWG писали спецификации в режиме «живого стандарта» («Living Standard»), а в W3C принимали эти спецификации как «снимки», и внедряли их в чёткие версии своей спецификации. Группа W3C работала значительно медленнее, потому что необходимо было обеспечивать требования большего спектра пользователей, а не только веб-браузеров.

28 октября 2014 года Консорциум Всемирной Паутины (W3C) объявил о предоставлении набору спецификаций **HTML5** статус рекомендованного стандарта. Интересно, что в этом виде спецификации **HTML 5.0** были сформированы ещё два года до того, после чего работа была сосредоточена на проведении тестирования и оценки совместимости доступных реализаций. На время стандартизации **HTML5** уже давно стал стандартом де-факто и активно использовался в веб-приложениях. Фактическое утверждение стандарта лишь формально поставило точку в продвижении **HTML5** и подтвердило вездесущность и корректность его реализации.

Спецификация **HTML5** не ограничивается только разметкой и включает в себя ряд веб-технологий, которые в совокупности формируют открытую Веб-платформу - программное окружение для работы кросс-платформенных приложений, способных взаимодействовать с оборудованием, и которые поддерживают средства для работы с видео, графикой и анимацией, что даёт расширенные сетевые возможности.

Что нового в HTML5?

Объявление **DOCTYPE** для **HTML5** очень простое:

```
<!DOCTYPE html>
```

Объявление кодировки символов (charset) также очень простое:

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

HTML5 пример:

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

```
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>

<body>
Content of the document.....
</body>

</html>
```

Попробуйте сами »

Кодировка символов по умолчанию в **HTML5** - это **UTF-8**.

Новые элементы в HTML5

Наиболее интересными являются новые **HTML5** элементы:

Новые **семантические элементы**, такие как `<header>`, `<footer>`, `<article>` и `<section>`.

Новые **атрибуты элементов формы**, такие как number, date, time, calendar и range.

Новые **графические элементы**: `<svg>` и `<canvas>`.

Новые **мультимедиа элементы**: `<audio>` и `<video>`.

В следующем разделе HTML5 Поддержка браузерами вы узнаете, "как научить" старые браузеры обрабатывать "неизвестные" для них (новые) HTML элементы.

Новые HTML5 API (интерфейсы прикладного программирования)

Наиболее интересными являются новые API в **HTML5**:

- HTML Геолокация
- HTML Drag and Drop
- HTML Локальное хранилище
- HTML Кеш приложения
- HTML Веб-работники
- HTML SSE

Совет: Локальное хранилище является мощной заменой для файлов cookie.

Удалённые элементы в HTML5

В **HTML5** удалены такие элементы **HTML4**:

Удалено элемент	Используют вместо этого
<acronym>	<abbr>
<applet>	<object>
<basefont>	CSS
<big>	CSS
	CSS
<dir>	
<font>	CSS
<frame>	
<frameset>	
<noframes>	
<strike>	CSS, <s> або <del>
<tt>	CSS

В разделе HTML5 Миграция с HTML4 вы узнаете, как легко перейти с **HTML4** на **HTML5**.

HTML История. Как это было?

С первых дней существования Всемирной паутины существует много версий **HTML**:

Год	Версия
1989	Tim Berners-Lee изобрёл www
1991	Tim Berners-Lee изобрёл HTML
1993	Dave Raggett разработал HTML+
1995	Рабочая группа HTML разработала HTML 2.0
1997	W3C Рекомендация: HTML 3.2
1999	W3C Рекомендация: HTML 4.01
2000	W3C Рекомендация: XHTML 1.0
2008	WHATWG HTML5 Первый публичный проект
2012	WHATWG HTML5 Living Standard (Живой стандарт)
2014	W3C Рекомендация: HTML5
2016	W3C Кандидат рекомендации: HTML 5.1
2017	W3C Рекомендация: HTML5.1 2nd Edition
2017	W3C Рекомендация: HTML5.2

С 1991 по 1999 год HTML разрабатывался с версии 1 до версии 4.

В 2000 году Консорциум Всемирной паутины - World Wide Web (**W3C**) - рекомендовал XHTML 1.0. Синтаксис XHTML был строгим, и разработчики были вынуждены писать валидный и "хорошо сформированный" код.

В 2004 году группа W3C решила закрыть разработку HTML в пользу XHTML.

В 2004 году была сформирована **WHATWG** (Web Hypertext Application Technology Working Group - Рабочая группа веб-технологий по использованию гипертекста). WHATWG хотела разработать HTML, согласованный с тем, как использовался веб-сайт, когда он был совместим со старыми версиями HTML.

В 2004 - 2006 годах компания WHATWG получила поддержку от основных производителей браузеров.

В 2006 году группа W3C объявила про поддержку WHATWG.

В 2008 году был выпущен первый публичный проект **HTML5**.

В 2012 году WHATWG и W3C приняли решение о разделении:

Группа WHATWG хотела разработать HTML как "Living Standard" ("Живий Стандарт"). "Living Standard" всегда обновляется и усовершенствуется. Можно добавить новые функции, но старую функциональность нельзя удалить.

WHATWG HTML5 Living Standard был опубликован в 2012 году и постоянно обновляется.

Группа W3C хотела разработать окончательный стандарт HTML5 и XHTML.

W3C HTML5 Рекомендацию было выпущено 28 октября 2014 года.

W3C HTML5.1 2nd Edition Рекомендацию было выпущено 3 октября 2017 года.

W3C HTML5.2 Рекомендацию было выпущено 14 декабря 2017 года.

Примечание: Получить актуальную и более полную информацию о "Living Standard" WHATWG и действующие Рекомендации W3C вы всегда можете на официальных сайтах WHATWG и W3C.

□ Prev Next □

Переход с HTML4 на HTML5

□ Prev Next □

Как перейти с HTML4 на HTML5?

В этом разделе рассказывается лишь о том, **как перейти с HTML4 на HTML5**.

Этот раздел демонстрирует, как переделать страницу из HTML4 на страницу HTML5, не удаляя ничего с начального содержания или структуры веб-страницы.

Вы можете перейти с **XHTML** на **HTML5**, используя точно такой же рецепт.

Типичный HTML4	Типичный HTML5
	<header>
	<nav>
	<section>
	<article>
	<footer>

Типичная HTML4 страница

Пример:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
<title>HTML4</title>
<style>
body {
  font-family: Verdana,sans-serif;
  font-size: 0.9em;
}

div#header, div#footer {
  padding: 10px;
  color: white;
  background-color: black;
}

div#content {
  margin: 5px;
  padding: 10px;
  background-color: lightgrey;
}

div.article {
  margin: 5px;
  padding: 10px;
  background-color: white;
}

div#menu ul {
  padding: 0;
}

div#menu ul li {
  display: inline;
  margin: 5px;
}
```

```
</style>
</head>
<body>

<div id="header">
  <h1>Monday Times</h1>
</div>

<div id="menu">
  <ul>
    <li>News</li>
    <li>Sports</li>
    <li>Weather</li>
  </ul>
</div>

<div id="content">
  <h2>News Section</h2>
  <div class="article">
    <h2>News Article</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>
  </div>
  <div class="article">
    <h2>News Article</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>
  </div>
</div>

<div id="footer">
  <p>&amp;amp;copy; 2016 Monday Times. All rights reserved.</p>
</div>

</body>
</html>
```

Попробуйте сами »

Изменить на HTML5 Doctype

Изменить **doctype**:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

на HTML5 **doctype**:

Пример:

```
<!DOCTYPE html>
```

Попробуйте сами »

Изменить на кодировку HTML5

Изменить информацию **кодировки**:

```
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
```

на HTML5 **кодировку**:

Пример:

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

Попробуйте сами »

Добавить HTML5Shiv

Новые семантические элементы **HTML5** поддерживаются во всех современных браузерах. Кроме того, вы можете "научить" старые браузеры, как обрабатывать "неизвестные элементы".

Однако IE8 по-прежнему не допускает стилизации неизвестных элементов. Таким образом, HTML5Shiv - это обходной путь для JavaScript, чтобы включить стилизацию элементов **HTML5** в версиях Internet Explorer до версии 9.

Добавить HTML5Shiv:

Пример:

```
<!--[if lt IE 9]>
  <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
<![endif]-->
```

Попробуйте сами »

Прочитать больше про **HTML5Shiv** можно в разделе HTML5. Поддержка браузерами.

Изменить на семантические HTML5 элементы

Существующий CSS содержит идентификаторы и классы для стилизации элементов:

```
body {
  font-family: Verdana,sans-serif;
  font-size: 0.9em;
}

div#header, div#footer {
  padding: 10px;
  color: white;
  background-color: black;
}

div#content {
  margin: 5px;
  padding: 10px;
  background-color: lightgrey;
}

div.article {
  margin: 5px;
  padding: 10px;
  background-color: white;
}

div#menu ul {
  padding: 0;
}

div#menu ul li {
  display: inline;
  margin: 5px;
}
```

Необходимо заменить CSS стили для семантических элементов **HTML5**:

```
body {
  font-family: Verdana,sans-serif;
  font-size: 0.9em;
}

header, footer {
  padding: 10px;
  color: white;
  background-color: black;
}

section {
  margin: 5px;
```

```
  padding: 10px;
  background-color: lightgrey;
}

article {
  margin: 5px;
  padding: 10px;
  background-color: white;
}

nav ul {
  padding: 0;
}

nav ul li {
  display: inline;
  margin: 5px;
}
```

И, наконец, меняем элементы на семантические элементы **HTML5**:

Пример:

```
<body>

<header>
  <h1>Monday Times</h1>
</header>

<nav>
  <ul>
    <li>News</li>
    <li>Sports</li>
    <li>Weather</li>
  </ul>
</nav>

<section>
  <h2>News Section</h2>
  <article>
    <h2>News Article</h2>
    <p>Lorem ipsum dolor sit amet..</p>
  </article>
  <article>
    <h2>News Article</h2>
    <p>Lorem ipsum dolor sit amet..</p>
  </article>
</section>

<footer>
  <p>&copy; 2014 Monday Times. All rights reserved.</p>
</footer>

</body>
```

Попробуйте сами »

Разница между <article> <section> и

Существует путаница (отсутствие отличия) в стандарте **HTML5** между `<article>`, `<section>` и `<div>`.

В стандарте **HTML5** элемент `<section>` определяется как блок связанных элементов.

Элемент `<article>` определяется как полный, автономный блок связанных элементов.

Элемент `<div>` определяется как блок дочерних элементов.

Как интерпретировать это?

В приведённом выше примере мы использовали `<section>` как контейнер для связанных статей `<articles>`.

Но мы могли бы использовать `<article>` как контейнер для завершенной статьи.

Вот несколько разных примеров:

<article> в <article>:

```
<article>

<h2>Famous Cities</h2>

<article>
  <h2>London</h2>
  <p>London is the capital city of England.</p>
</article>

<article>
  <h2>Paris</h2>
  <p>Paris is the capital and most populous city of France.</p>
</article>

<article>
  <h2>Tokyo</h2>
  <p>Tokyo is the capital of Japan.</p>
</article>

</article>
```

Попробуйте сами »

в <article>:

```
<article>

<h2>Famous Cities</h2>

<div class="city">
  <h2>London</h2>
  <p>London is the capital city of England.</p>
</div>

<div class="city">
  <h2>Paris</h2>
  <p>Paris is the capital and most populous city of France.</p>
</div>

<div class="city">
  <h2>Tokyo</h2>
  <p>Tokyo is the capital of Japan.</p>
</div>

</article>
```

Попробуйте сами »

в <section> в <article>:

```
<article>

<section>
  <h2>Famous Cities</h2>

  <div class="city">
    <h2>London</h2>
    <p>London is the capital city of England.</p>
  </div>

  <div class="city">
    <h2>Paris</h2>
    <p>Paris is the capital and most populous city of France.</p>
  </div>
```

```
  <div class="city">
    <h2>Tokyo</h2>
    <p>Tokyo is the capital of Japan.</p>
  </div>
</section>

<section>
  <h2>Famous Countries</h2>

  <div class="country">
    <h2>England</h2>
    <p>London is the capital city of England.</p>
</div>

  <div class="country">
    <h2>France</h2>
    <p>Paris is the capital and most populous city of France.</p>
</div>

  <div class="country">
    <h2>Japan</h2>
    <p>Tokyo is the capital of Japan.</p>
  </div>
</section>

</article>
```

Попробуйте сами »

❮ Prev Next ❯

HTML5 Новые элементы

❮ Prev Next ❯

Новые элементы в HTML5

Ниже приведён список новых **HTML5** элементов и описание того, для чего они используются.

Новые семантические / структурные элементы

HTML5 предлагает новые элементы для лучшей структуры документов:

Тег	Описание
<article>	Определяет статью в документе
<aside>	Определяет содержание, кроме содержания страницы
<bdi>	Изолирует часть текста, который может быть отформатирован в другом направлении от другого текста за его пределами
<details>	Определяет дополнительные детали, которые пользователь может просматривать или прятать
<dialog>	Определяет диалоговый бокс или окно
<figcaption>	Определяет заголовок для элемента <figure>
<figure>	Определяет автономное содержание
<footer>	Определяет нижний колонтитул для документа или раздела
<header>	Определяет заголовок для документа или раздела
<main>	Определяет основное содержание документа
<mark>	Определяет помеченный / выделенный текст
<meter>	Определяет скалярное измерение в пределах определённого диапазона (датчик)
<nav>	Определяет навигационные ссылки
<progress>	Представляет ход выполнения задания
<rp>	Определяет, что показывать в браузерах, которые не поддерживают ruby аннотации
<rt>	Определяет пояснения / произношение символов (для восточноазиатской типографики)
<ruby>	Определяет аннотацию ruby (для восточноазиатской типографики)

<section>	Определяет раздел в документе
<summary>	Определяет видимый заголовок для элемента <details>
<time>	Определяет дату / время
<wbr>	Определяет возможный разрыв строки

Читать больше про HTML5 Семантика.

Новые элементы формы

Тег	Описание
<datalist>	Определяет список предварительно определённых параметров управления вводом
<output>	Определяет результат расчёта

Прочитайте всё о старых и новых элементах формы в разделе HTML Элементы формы.

Новые типы ввода

Новые типы ввода

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

Новые атрибуты ввода

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max
- multiple
- pattern (regexp)
- placeholder
- required
- step

Узнайте о старых и новых типах ввода данных в разделе HTML Типы ввода.

Узнайте всё об атрибутах ввода в разделе HTML Атрибуты ввода.

HTML5 - Новый синтаксис атрибутов

HTML5 позволяет использовать четыре разных синтаксиса атрибутов.

Этот пример демонстрирует разные синтаксисы, которые используются в теге `<input>`:

Тип	Пример
Пустой	<input type="text" value="John" **disabled**>
Без кавычек	<input type="text" **value=John**>
Двойные кавычки	<input type="text" **value="John Doe"**>
Одинарные кавычки	<input type="text" **value='John Doe'**>

В **HTML5** можно использовать все четыре синтаксиса, в зависимости от того, что необходимо для атрибута.

HTML5 Графика

Тег	Описание
<canvas>	Нарисуйте графику на лету с помощью скриптов (обычно JavaScript)
<svg>	Нарисуйте масштабированную векторную графику

Читать больше про HTML5 Canvas.

Читать больше про HTML5 SVG.

Новые медиа элементы

Тег	Описание
<audio>	Определяет звуковой контент
<embed>	Определяет контейнер для внешнего (не HTML) приложения
<source>	Определяет несколько медиа-ресурсов для медиа-элементов (<video> и <audio>)
<track>	Определяет текстовые дорожки для медиа-элементов (<video> и <audio>)
<video>	Определяет видео или фильм

Читать больше про HTML5 Video.

Читать больше про HTML5 Audio.

□ Prev Next □

HTML5 Семантическиие элементы

□ Prev Next □

Семантика - это изучение значений слов и фраз в языке.

Семантические элементы = элементы со значением.

Что такое семантические элементы?

Семантический элемент чётко описывает его значение как для браузера, так и для разработчика.

Например, **несемантические** элементы: `<div>` и `<span>` - нечого не говорят о своём содержании.

Например, **семантические** элементы: `<form>`, `<table>` и `<article>` - чётко определяют своё содержание.

Поддержка браузерами

Да Да Да Да Да

Семантические элементы HTML5 поддерживаются во всех современных браузерах.

Кроме того, вы можете "научить" старые браузеры обрабатывать "неизвестные элементы".

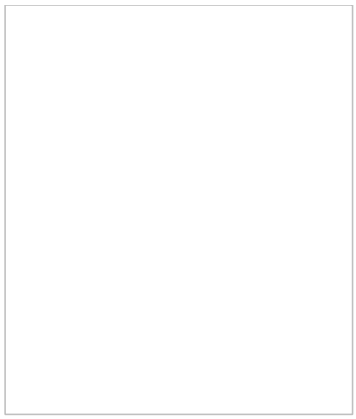
Читайте об этом в разделе HTML5 Поддержка браузерами.

Новые семантические элементы в HTML5

Многие веб-сайты содержат HTML-код:
для обозначения навигации, заголовка и нижнего колонтитула.

HTML5 предлагает новые семантические элементы для определения разных частей веб-страницы:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



HTML5 элемент <section>

Элемент `<section>` определяет раздел в документе.

В соответствие с документацией HTML5 W3C: "Раздел является тематической группировкой содержания, как правило, из заголовком."

Домашнюю страницу обычно можно разделить на разделы для представления, содержания и контактной информации.

Пример:

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p>
</section>
```

Попробуйте сами »

HTML5 элемент <article>

Элемент `<article>` определяет независимое, автономное содержание (статью).

Статья должна иметь смысл сама по себе, так, чтобы её можно было читать независимо от остального сайта.

Примеры использования элемента `<article>`:

- Сообщение форума
- Публикация в блоге
- Газетная статья

Пример:

```
<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

Попробуйте сами »

Вложение <article> в <section> или наоборот?

Элемент`<article>`определяет независимое, автономное содержание.

Элемент `<section>` определяет раздел в документе.

Можем ли мы использовать определение, чтобы решить, как вставить эти элементы? Нет, мы не можем!

Итак, в Интернете вы найдете HTML-страницы с элементами `<section>` которые содержат элементы `<article>` и элементы `<article>`, которые содержат элементы `<section>`.

Вы также найдёте страницы с элементами `<section>`, которые содержат элементы `<section>` и элементы `<article>`, которые содержат элементы `<article>`.

Пример из газеты: Спортивная статья <article> в спортивном разделе **section**, может содержать технический раздел **section** в каждой статье <article>.

HTML5 элемент <header>

Элемент <header> определяет заголовок для документа или раздела.

Элемент <header> должен использоваться как контейнер для вступительного содержания.

Вы можете иметь несколько элементов <header> в одном документе.

Следующий пример определяет заголовок статьи:

Пример:

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

Попробуйте сами »

HTML5 элемент <footer>

Элемент <footer> определяет колонтитул для документа или раздела.

Элемент <footer> должен содержать информацию, которая содержится в нём.

В нижнем колонтитуле (футере) обычно содержится информация об авторе документа, информация об авторских правах, ссылка на условия использования, контактная информация и т.д.

Может быть несколько элементов <footer> на одной веб-странице.

Пример:

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

Попробуйте сами »

HTML5 элемент <nav>

Элемент <nav> определяет набор навигационных ссылок.

Обратите внимание, что НЕ все ссылки документа должны находиться внутри элемента <nav>. Элемент <nav> предназначен лишь для основного блока навигационных ссылок.

Пример:

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

Попробуйте сами »

HTML5 элемент <aside>

Элемент <aside> определяет некоторое содержание, кроме содержания, в котором он расположен (как боковая панель)

Содержание <aside> должно быть связано с окружающим содержанием.

Пример:

```
<p>My family and I visited The Epcot center this summer.</p>

<aside>
  <h4>Epcot Center</h4>
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

Попробуйте сами »

HTML5 элементы <figure> и <figcaption>

Подпись к изображению делается для того, чтобы добавить объяснение, что на нём изображено.

В HTML5 изображения и подпись к нему можна сгруппировать в элемент <figure>:

Пример:

```
<figure>
  <img src="pic_trulli.jpg" alt="Trulli">
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

Попробуйте сами »

Элемент <img> определяет изображение, элемент <figcaption> определяет подпись к изображению.

Почему семантические элементы?

Начиная с HTML4, разработчики использовали свои собственные названия id/class для стилизации элементов: header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav и др.

Это значительно усложняло поисковым системам определять правильное содержание веб-страницы.

С новыми HTML5 элементами (<header> <footer> <nav> <section> <article>) это стало намного легче.

Согласно определения с W3C, Semantic Web: "Позволяет обмениваться данными и повторно использовать их в разных программах, на предприятиях и в сообществах".

Семантические элементы в HTML5

Ниже приведён **алфавитный список новых семантических элементов в HTML5**.

Ссылка на полный HTML5 Справочник тегов.

Тег	Описание
<article>	Определяет статью
<aside>	Определяет содержание, кроме содержания страницы
<details>	Определяет дополнительные детали, которые пользователь может просматривать или скрывать
<figcaption>	Определяет подпись для элемента <figure>
<figure>	Определяет автономное содержание, например, иллюстрации, диаграммы, фотографии, списки кодов и т.д.
<footer>	Определяет нижний колонтитул для документа или раздела
<header>	Определяет заголовок для документа или раздела
<main>	Определяет основное содержание документа
<mark>	Определяет помеченный / выделенный текст
<nav>	Определяет навигационные ссылки
<section>	Определяет раздел в документе
<summary>	Определяет видимый заголовок для элемента <details>
<time>	Определяет дату/время

HTML5 Server-Sent Events (SSE) - События, отправленные сервером

□ Prev Next □

Для чего нужны Server-Sent Events (SSE) - События, отправленные сервером?

События, отправленные сервером позволяют веб-странице получать обновления с сервера.

События, отправленные сервером - это одностороние сообщения

События, отправленные сервером - это когда веб-страница автоматически получает обновления с сервера.

Это также было возможным и раньше, но на веб-странице необходимо было сделать запрос, доступны ли обновления. Из событиями, отправленными сервером, обновления отправляются автоматически.

Примеры: Facebook/Twitter обновления, обновления цен на акции, новостные ленты, спортивные результаты и т.д.

Поддержка браузерами

Цифры в таблице определяют первую версию браузера, которая полностью поддерживает события, отправленные сервером.

API					
SSE	6.0	Не поддерживается	6.0	5.0	11.5

Получать уведомления об отправленных сервером событиях

Объект EventSource используется для получения сообщений о событиях, которые отправляются сервером:

Пример:

```
var source = new EventSource("demo_sse.php");
source.onmessage = function(event) {
  document.getElementById("result").innerHTML += event.data + "<br>";
};
```

Попробуйте сами »

Объяснение примера:

- Создайте новый объект EventSource и укажите URL-адрес страницы, которая отправляет обновления (в этом примере "demo_sse.php")
- Каждый раз при получении обновления происходит событие onmessage
- Когда происходит событие onmessage, поместите полученные данные в элемент с id = "result"

Проверьте поддержку событий, отправленных сервером

В приведённом выше примере "Попробуйте сами" было несколько дополнительных строк кода для проверки поддержки браузером событий, отправленных сервером:

```
if(typeof(EventSource) !== "undefined") {
  // Да! Server-sent events поддерживается!
  // Какой-то код.....
} else {
  // Извините! Server-sent events не поддерживается..
}
```

Пример кода на стороне сервера

Для приведённого выше примера нужен сервер, который может отправлять обновления даных (например, PHP или ASP).

Синтаксис потока событий на стороне сервера простой. Установите заголовок "Content-Type" на "text/event-stream". Теперь вы можете начать отправлять потоки событий.

Код на PHP (demo_sse.php):

```
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');

$time = date('r');
echo "data: The server time is: {$time}\n\n";
flush();
?>
```

Код на ASP (VB) (demo_sse.asp):

```
<%
Response.ContentType = "text/event-stream"
Response.Expires = -1
Response.Write("data: The server time is: " & now())
Response.Flush()
%>
```

Объяснение кода:

- Установите заголовок "Content-Type" в "text/event-stream"
- Укажите, что страница не должна кешироваться (not cache)
- Выведите данные, которые необходимо отправить (**Всегда** нужно начинать с "data: ")
- Скиньте исходные данные назад на веб-страницу

Объект EventSource

В приведённых выше примерах мы использовали событие *onmessage* для получения сообщений. Но доступны и другие события:

События	Описание
onopen	Когда открыто соединение с сервером
onmessage	Когда получено сообщение
onerror	Когда возникла ошибка

□ Prev Next □

HTML5 SVG - Scalable Vector Graphics

□ Prev Next □

Что такое SVG?

- **SVG** расшифровывается как Scalable Vector Graphics - Масштабируемая векторная графика
- **SVG** используется для определения Web-графики
- **SVG** рекомендуется W3C (Консорциумом всемирной паутины)

HTML элемент <svg>

HTML элемент `<svg>` является контейнером для SVG-графики.

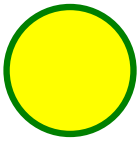
SVG имеет несколько методов для рисования контуров, боксов, кругов, текста и графических изображений.

Поддержка браузерами

Цифры в таблице определяют первую версию браузера, которая полностью поддерживает элемент `<svg>`.

Элемент					
<svg>	4.0	9.0	3.0	3.2	10.1

SVG Круг



Пример:

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

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
</svg>

</body>
</html>
```

Попробуйте сами »

SVG Прямоугольник

Пример:

```
<svg width="400" height="100">
  <rect width="400" height="100" style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />
</svg>
```

Попробуйте сами »

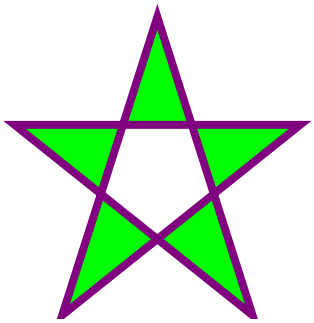
SVG Закругленный прямоугольник

Пример:

```
<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
```

```
</svg>
```
Попробуйте сами »

SVG Звезда



Пример:

```
<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
```
Попробуйте сами »

SVG Лого

Пример:

```
<svg height="130" width="500">
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />
      <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
    </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-family="Verdana" x="50" y="86">SVG</text>
  К сожалению, ваш браузер не поддерживает встроенный SVG.
</svg>
```
Попробуйте сами »

Отличия между SVG и Canvas

SVG - это язык для описания 2D-графики в XML.

Canvas рисует 2D графику на лету (с помощью JavaScript).

SVG базируется на XML, что означает, что каждый элемент доступен в SVG DOM. Вы можете присоединить обработчики событий JavaScript для элемента.

В SVG каждая нарисованная форма запоминается как объект. Если атрибуты объекта SVG изменены, браузер может автоматически повторно воспроизвести форму.

Canvas визуализируется пиксель за пикселем. В canvas, когда рисуется рисунок, он забывается браузером. Если его позицию надо изменить, то необходимо перерисовать всю сцену, включая любые объекты, которые могли быть покрыты графикой.

Сравнение Canvas и SVG

В таблице ниже показаны некоторые важные отличия между Canvas и SVG:

Canvas	SVG
• Зависит от разрешения • Нет поддержки обработчиков событий • Низкие возможности отображения текста • Вы можете сохранить полученное изображение как .png или .jpg • Хорошо подходит для графических интенсивных игр	• Не зависит от разрешения • Поддержка обработчиков событий • Лучше всего подходит для программ из большими областями рендеринга (Карты Google) • Медленный рендеринг, если комплекс (все, что использует DOM) будет очень медленным • Не подходит для игровых программ

SVG Учебник

Узнать больше про SVG можно, посетив Учебник по SVG.

□ Prev Next □

HTML5 Гид по стилю и соглашение по кодированию

□ Prev Next □

HTML. Как правильно писать код? Соглашение по кодированию

Веб-разработчики часто не уверены в использовании стиля кодирования и синтаксиса в **HTML**.

Между 2000 и 2010 годами многие веб-разработчики перешли с **HTML** на **XHTML**.

Начиная с **XHTML** разработчики были вынуждены писать валидный и "хорошо сформированный" код.

HTML5 является более небрежным, когда идёт речь о проверке (валидации) кода.

Будьте разумны и будущее это докажет

Последовательное использование определённого стиля облегчает другим понимание вашего **HTML**.

В будущем программы, такие, как считыватели XML, могут захотеть прочитать ваш **HTML**.

Использование хорошо сформированного синтаксиса "близкого к XHTML" является разумным выбором.

Держите свой код аккуратным, чистым и хорошо сформированным.

Используйте правильный Doctype

Всегда объявляйте Doctype (тип документа) в первой строке документа:

```
<!DOCTYPE html>
```

Если вам необходима согласованность с тегами нижнего регистра, вы можете использовать:

```
<!doctype html>
```

Используйте названия в нижнем регистре

HTML5 позволяет смешивать большие и маленькие буквы в названиях элементов.

Мы рекомендуем использовать названия элементов буквами лишь в нижнем регистре, поскольку:

- Смешивание названий буквами в верхнем и нижнем регистре плохо для восприятия

- Разработчики обычно используют названия с помощью букв в нижнем регистре (как в XHTML)
- Нижний регистр выглядит чище
- Буквы в нижнем регистре легче и быстрее писать

Плохо:

```
<SECTION>
  <p>This is a paragraph.</p>
</SECTION>
```

Очень плохо:

```
<Section>
  <p>This is a paragraph.</p>
</SECTION>
```

Хорошо:

```
<section>
  <p>This is a paragraph.</p>
</section>
```

Закрывайте все HTML элементы

В HTML5 нет необходимости закрывать все элементы (например, элемент <p>).

W3C рекомендует закрывать все HTML элементы.

Плохо:

```
<section>
  <p>This is a paragraph.
  <p>This is a paragraph.
</section>
```

Хорошо:

```
<section>
  <p>This is a paragraph.</p>
  <p>This is a paragraph.</p>
</section>
```

Закрывайте пустые HTML элементы

В HTML5 не обязательно закрывать пустые элементы.

Разрешено:

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

Также разрешено:

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

При этом закрывающая косая черта (/) НЕОБХОДИМА в XHTML и XML.

Если вы ожидаете, что программное обеспечение XML будет иметь доступ к вашей странице, то стоит сохранить закрытую косую черту (слэш)!

Используйте названия атрибутов в нижнем регистре

HTML5 позволяет смешивать большие и маленькие буквы в названиях атрибутов.

W3C рекомендует использовать названия атрибутов в нижнем регистре, поскольку:

- Смешивание букв верхнего и нижнего регистров в названиях плохо воспринимается
- Разработчики обычно используют названия в нижнем регистре (как в XHTML)
- Нижний регистр выглядит чище
- Буквы в нижнем регистре легче и быстрее писать

Плохо:

```
<div CLASS="menu">
```

Хорошо:

```
<div class="menu">
```

Значение атрибутов в кавычках

HTML5 позволяет писать значение атрибутов без кавычек.

W3C рекомендует писать значение атрибутов в кавычках, поскольку:

- Разработчики обычно пишут значение атрибутов в кавычках (как в XHTML)
- Значение в кавычках легче читать
- Если значение содержит пробелы, вы ДОЛЖНЫ использовать кавычки

Очень плохо:

Это не будет работать, поскольку значение содержит пробелы:

```
<table class=table striped>
```

Плохо:

```
<table class=striped>
```

Хорошо:

```
<table class="striped">
```

Атрибуты изображения

Всегда добавляйте атрибут `alt` к изображениям. Этот атрибут является важным, когда изображение по какой-то причине не может быть отображено. Также всегда определяйте ширину и высоту изображения. Это уменьшает мерцание, поскольку браузер может резервировать пространство для изображения перед загрузкой.

Плохо:

```
<img src="html5.gif">
```

Хорошо:

```
<img src="html5.gif" alt="HTML5" style="width:128px;height:128px">
```

Пробелы и знаки равенства

HTML5 позволяет размещать пробелы вокруг знаков равенства. Но проще читать, когда объекты сгруппированы вместе.

Плохо:

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

Хорошо:

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

Избегайте длинных строк кода

Используя редактор HTML, неудобно прокручивать влево и вправо, чтобы читать HTML-код.

Старайтесь избегать строк кода длиной больше 80 символов.

Пустые строки и отступ

Не добавляйте пустые строки без крайней необходимости.

Для удобства чтения добавляйте пустые строки для разделения больших или логических блоков кода.

Для удобства чтения добавьте два пробела отступа. Не используйте клавишу табуляции.

Не используйте ненужные пустые строки и отступы. Нет необходимости делать отступы для каждого элемента:

Ненужно:

```
<body>

  <h1>Famous Cities</h1>

  <h2>Tokyo</h2>

  <p>
    Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
    and the most populous metropolitan area in the world.
    It is the seat of the Japanese government and the Imperial Palace,
    and the home of the Japanese Imperial Family.
  </p>

</body>
```

Лучше:

```
<body>

<h1>Famous Cities</h1>

<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.
It is the seat of the Japanese government and the Imperial Palace,
and the home of the Japanese Imperial Family.</p>

</body>
```

Пример таблицы:

```
<table>
  <tr>
    <th>Name</th>
    <th>Description</th>
  </tr>
  <tr>
    <td>A</td>
    <td>Description of A</td>
  </tr>
  <tr>
    <td>B</td>
    <td>Description of B</td>
  </tr>
</table>
```

Пример списка:

```
<ul>
  <li>London</li>
  <li>Paris</li>
  <li>Tokyo</li>
```

```
</ul>
```

Опускание <html> и <body>?

В HTML5 тег `<html>` и тег `<body>` могут быть опущены.

Следующий код будет считаться валидным согласно спецификации HTML5:

Пример:

```
<!DOCTYPE html>
<head>
  <title>Page Title</title>
</head>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

Попробуйте сами »

Однако мы не рекомендуем пропускать теги `<html>` и `<body>`.

Элемент `<html>` указывает загрузку HTML-документа. Также это рекомендуемое место для определения языка страницы:

```
<!DOCTYPE html>
<html lang="en-US">
```

Объявление языка страницы имеет важное значение для программ с доступностью (программы для чтения с экрана) и поисковых систем.

Опускание `<html>` или `<body>` может обрушить DOM и программное обеспечение XML.

Опускание `<body>` может вызвать ошибки в старых браузерах (IE9).

Опускание <head>?

В HTML5 тег `<head>` также может быть опущен.

По умолчанию браузеры будут добавлять все элементы, которые находятся перед `<body>` до `<head>` элемента.

Вы можете упростить HTML, опустив тег `<head>`:

Пример:

```
<!DOCTYPE html>
<html>
<title>Page Title</title>

<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body>

</html>
```

Попробуйте сами »

Однако мы не рекомендуем опускать теги `<head>`.

Опускание тегов неизвестно для веб-разработчиков. Необходимо время, чтобы это стало ориентиром (стандартом).

Метаданные

Элемент `<title>` обязателен в HTML5. Необходимо делать название как можно более содержательным:

```
<title>HTML5 Syntax and Coding Style</title>
```

Для обеспечения надлежащей интерпретации и правильного индексирования поисковыми системами, как язык страницы,

так и кодирование символов должны быть определены как можно раньше в документе:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>HTML5 Syntax and Coding Style</title>
</head>
```

Настройка окна просмотра - Viewport

В HTML5 появился метод, который позволил веб-дизайнерам взять под свой контроль окно просмотра с помощью тега `<meta>`.

Окно просмотра (viewport) - видимая область пользователя веб-страницы. Она меняется в зависимости от устройства и будет меньше на мобильном телефоне, чем на экране компьютера.

Вам следует включить следующий элемент `<meta>` на ваших веб-страницах:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Элемент `<meta>` viewport предоставляет инструкции браузеру, как контролировать размеры и масштабирование страницы.

Часть width=device-width устанавливает ширину страницы, чтобы она соответствовала ширине экрана устройства (которая зависит от устройства).

Часть initial-scale=1.0 устанавливает начальный уровень масштабирования, когда страница сначала загружается браузером.

Вот пример веб-страницы *без* viewport метатега, и той же веб-страницы *из* viewport метатегом:

Совет: Если вы просматриваете эту страницу с помощью телефона или планшета, можно нажать две ссылки ниже, чтобы увидеть разницу.

Без viewport метатега

Из viewport метатегом

HTML Коментарии

Короткие комментарии должны быть написани на одной строке:

```
<!-- Это комментарий -->
```

Комментарии, которые охватывают более одной строки, должны быть написаны так:

```
<!--
  Это пример длинных комментариев. Это пример длинных комментариев.
  Это пример длинных комментариев. Это пример длинных комментариев.
-->
```

Длинные комментарии легче увидеть, если они имеют отступ в два пробела.

Таблица стилей

Используйте простой синтаксис для ссылки на таблицы стилей (атрибут type не является необходимым):

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

Короткие правила стиля могут быть написаны кратко:

```
p.intro {font-family: Verdana; font-size: 16em;}
```

Длинные правила стиля лучше писать на нескольких строках:

```
body {
  background-color: lightgrey;
  font-family: "Arial Black", Helvetica, sans-serif;
  font-size: 16em;
  color: black;
}
```

- Разместите открывающую скобку на той же строке, что и селектор
- Используйте один пробел перед открывающей скобкой
- Используйте два пробела для отступа
- Используйте точку с запятой после каждой пары свойства-значения, включая последнюю
- Используйте всегда кавычки вокруг значений, если значение содержит пробелы
- Поместите закрывающую скобку на новую строку, без сопутствующих пробелов
- Избегайте строк более 80 символов

Подключение JavaScript в HTML

Используйте простой синтаксис для подключения внешних скриптов (атрибут type не нужен):

```
<script src="myscript.js">
```

Доступ к элементам HTML с помощью JavaScript

Следствием использования "нечистых" стилей HTML может стать ошибка JavaScript.

Эти два оператора JavaScript дадут разные результаты:

Пример:

```
var obj = getElementById("Demo")
```

```
var obj = getElementById("demo")
```

Попробуйте сами »

Посетите Учебник по стилю JavaScript.

Используйте имена файлов в нижнем регистре

Некоторые веб-серверы (Apache, Unix) чувствительны к регистру имен файлов: "london.jpg" не доступен как "London.jpg".

Другие веб-серверы (Microsoft, IIS) не чувствительны к регистру: "london.jpg" доступен как "London.jpg", так и в качестве "london.jpg".

Если вы используете смесь верхнего и нижнего регистра, вы должны быть очень последовательными.

Если вы перейдете с нечувствительного к регистру сервера на чувствительный к регистру сервер, то даже небольшие ошибки сломают вашу сеть!

Чтобы избежать этих проблем, всегда используйте имена файлов в нижнем регистре.

Расширение файлов

HTML файлы всегда должны иметь расширение **.html** или **.htm**.

CSS файлы должны иметь расширение **.css**.

JavaScript файлы должны иметь расширение **.js**.

Разница между .htm и .html

Нет разницы между расширениями .htm и .html. Оба расширения веб-браузерами или веб-серверами будут рассматриваться как HTML.

Отличия культурные:

.htm "чувствуют" ранние системы DOS, где система ограничивала расширение до 3 символов.

.html "чувствуют" операционные системы Unix, которые не имеют этого ограничения.

Технические отличия

Если URL-адрес не указывает имя файла (например, https://www.w3schools.com/css/), сервер возвращает имя файла по умолчанию. Общие названия файлов по умолчанию - index.html, index.htm, default.html и default.htm.

Если ваш сервер настроен лишь для работы с "index.html" как типичным (стандартным) именем файла, ваш файл должен быть назван "index.html", а не "index.htm".

Однако, серверы могут быть настроены на более чем одно имя файла по умолчанию, и обычно вы можете установить столько названий файлов по умолчанию, сколько необходимо.

Так или иначе, полное расширение для HTML-файлов .html, и нет никаких причин его не использовать.

❮ Prev Next ❯

HTML5 Видео

❮ Prev Next ❯

HTML Пример видео. Взято из Big Buck Bunny.

Ваш браузер не поддерживает HTML5 видео.

Попробуйте сами »

Воспроизведение видео в HTML

До появления спецификации HTML5 видео можно было воспроизвести лишь в веб-браузере с плагином (например, flash).

HTML5 элемент `<video>` указывает стандартный способ встраивания видео в веб-страницу.

Поддержка браузерами

Цифры в таблице определяют первую версию браузера, которая полностью поддерживает элемент `<video>`.

Элемент					
<video>	4.0	9.0	3.5	4.0	10.5

HTML элемент <video>

Чтобы показать видео в HTML, используйте элемент `<video>`:

Пример:

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Ваш браузер не поддерживает тег video.
</video>
```

Попробуйте сами »

Как это работает

Атрибут `controls` добавляет элементы управления видео, такие как воспроизведение, приостановка и уровень звука.

Рекомендуется всегда включать атрибуты `width` и `height`. Если height (высота) и width (ширина) не заданы, страница может мерцать при проигрывании видео.

Элемент `<source>` позволяет указывать альтернативные видеофайлы, с которых браузер может выбирать. Браузер будет использовать первый распознанный формат.

Текст между тегами `<video>` и `</video>` будет отображаться лишь в браузерах, которые не поддерживают элемент `<video>`.

HTML <video> Autoplay

Чтобы проигрывание видео начиналось автоматически, используйте атрибут `autoplay`:

Пример:

```
<video width="320" height="240" autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Ваш браузер не поддерживает тег video.
</video>
```

Попробуйте сами »

Атрибут autoplay не работает на мобильных устройствах, таких как iPad и iPhone.

HTML Video - Поддержка браузерами

В HTML5 есть три поддерживаемых видеоформата: MP4, WebM и Ogg.

Поддержка браузерами разных форматов:

Браузер	MP4	WebM	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	NO	NO
Opera	YES (from Opera 25)	YES	YES

HTML Видео - Типы Медиа

Формат файла	Тип медиа
MP4	video/mp4
WebM	video/webm
Ogg	video/ogg

HTML видео - методы, свойства и события

HTML5 определяет методы, свойства и события DOM для элемента `<video>`.

Это позволяет загружать, воспроизводить и приостанавливать видео, а также устанавливать продолжительность и громкость.

Есть также события DOM, которые могут сообщать вам, когда видео начинает воспроизводиться, приостанавливаться и т.д.

Пример: использование JavaScript

Play/Pause Big Small Normal

Ваш браузер не поддерживает HTML5 видео.

Видео взято из Big Buck Bunny.

Попробуйте сами »

Для полной DOM справки, перейдите на HTML5 Audio/Video DOM Справочник.

HTML5 Видео Теги

Тег	Описание
<video>	Определяет видео или фильм
<source>	Определяет множество мультимедийных ресурсов для элементов медиа, например <video> и <audio>
<track>	Определяет текстовые дорожки в медиаплеерах

Prev Next

HTML5 Web Storage - Веб-хранилище

Prev Next

HTML веб-хранилище; лучше чем cookies.

Что такое HTML веб-хранилище?

С помощью веб-хранилища веб-программы могут сохранять данные локально в браузере пользователя.

До появления спецификации HTML5, данные приложений должны были сохраняться в куках (cookies), включенных в каждом запросе сервера. Веб-хранилище является более безопасным, и большие объёмы данных могут сохраняться локально, не влияя на работу веб-сайта.

В отличие от файлов cookie, лимит хранилища намного больший (по крайней мере 5 МБ), и информация никогда не передаётся на сервер.

Веб-хранилище - для каждого источника (для домена и протокола). Все страницы с одного источника (одинакового происхождения) могут сохранять и получать доступ к этим самым данным.

Поддержка браузерами

Цифры в таблице определяют первую версию браузера, которая полностью поддерживает веб-хранилище.

API				
Веб-хранилище 4.0	8.0	3.5	4.0	11.5

HTML Объекты веб-хранилища

Веб-хранилище HTML обеспечивает два объекта для хранения данных на стороне клиента:

- `window.localStorage` - сохраняет данные без даты окончания строка действия;
- `window.sessionStorage` - сохраняет данные для одного сеанса (данные теряются при закрытии вкладки веб-браузера).

Перед использованием веб-хранилища проверьте поддержку браузера localStorage и sessionStorage:

```
if (typeof(Storage) !== "undefined") {
  // Код для localStorage/sessionStorage.
} else {
  // Извините! Поддержка веб-хранилища отсутствует.
}
```

Объект localStorage

Объект **localStorage** сохраняет данные без даты окончания срока действия. Данные не будут удалены во время закрытия браузера и будут доступны на следующий день, неделю или год.

Пример:

```
// Сохраняем
localStorage.setItem("lastname", "Smith");

// Получаем
document.getElementById("result").innerHTML = localStorage.getItem("lastname");
```
Попробуйте сами »

Объяснение примера:

- Создаём пару localStorage name/value (имя/значение) с name="lastname" и value="Smith";
- Получаем значение "lastname" и вставляем его в элемент с id="result" (результат).

Приведённый выше пример также может быть написан таким образом:

```
// Сохраняем
localStorage.lastname = "Smith";
// Получаем
document.getElementById("result").innerHTML = localStorage.lastname;
```

Синтаксис для удаления элемента "lastname" localStorage является следующим:

```
localStorage.removeItem("lastname");
```

Примечание: Пары name/value всегда сохраняются как строки. Не забывайте конвертировать их в другой формат, когда это необходимо!

Следующий пример подсчитывает количество нажатий пользователем кнопки. В этом коде строка значений преобразуется в число, чтобы иметь возможность увеличить счетчик:

Пример:

```
if (localStorage.clickcount) {
  localStorage.clickcount = Number(localStorage.clickcount) + 1;
} else {
  localStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "Вы нажали кнопку " +
localStorage.clickcount + " раз.";
```
Попробуйте сами »

Объект sessionStorage

Объект `sessionStorage` соответсвует объекту localStorage, **кроме того**, что он сохраняет данные лишь для одного сеанса. Данные удаляются, когда пользователь закрывает определённую вкладку веб-браузера.

В следующем примере подсчитывается количество раз, когда пользователь нажал кнопку в текущей сесии:

Пример:

```
if (sessionStorage.clickcount) {
  sessionStorage.clickcount = Number(sessionStorage.clickcount) + 1;
} else {
  sessionStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "Вы нажали кнопку " +
sessionStorage.clickcount + " раз в этой сесии.";
```
Попробуйте сами »

❮ Prev Next ❯

HTML5 Web Workers - Веб-работники

□ Prev Next □

Веб-работник - это JavaScript, который работает в фоновом режиме, не влияя на продуктивность страницы.

Что такое веб-работник на веб-странице?

Во время выполнения скриптов на странице HTML страница перестаёт отвечать, пока сценарий (выполнение скрипта) не закончится.

Веб-работник - это JavaScript, который работает в фоновом режиме, независимо от других скриптов, не влияя на продуктивность страницы. Вы можете продолжать делать всё, что хотите: нажимать, выбирая вещи и т.д., а веб-работник будет работать в фоновом режиме.

Поддержка браузерами

Цифры в таблице определяют первую версию браузера, которая полностью пподдерживает Web Workers (веб-работники).

API				
Web Workers 4.0	10.0	3.5	4.0	11.5

HTML Веб-работник. Пример

Приведённый ниже пример создаёт простого веб-работника, который подсчитывает числа в фоновом режиме:

Пример:

Count numbers:

Start Worker Stop Worker

Попробуйте сами »

Проверьте поддержку Веб-работника

Перед тем, как создать веб-работника, проверьте, поддерживает ли его браузер пользователя:

```
if (typeof(Worker) !== "undefined") {
  // Да! Веб-работник поддерживается!
  // Какой-то код.....
} else {
  // Извините! Веб-работник не поддерживается.
}
```

Создать файл Веб-работника

Теперь давайте создадим нашего веб-работника во внешнем JavaScript файле.

Здесь мы создаём скрипт, который считает. Скрипт сохраняется в файле "demo_workers.js":

```
var i = 0;

function timedCount() {
  i = i + 1;
  postMessage(i);
  setTimeout("timedCount()",500);
}
```

timedCount();

Важной частью вышеуказанного кода является метод `postMessage()`, который используется для публикации сообщения на HTML-странице.

Примечание: Обычно веб-работники не используются для таких простых скриптов, а используются для более ресурсоёмких задач.

Создать объект Веб-работника

Теперь, когда у нас есть файл веб-работника, нам необходимо вызвать его с HTML-страницы.

Следующие строки проверяют, существует ли уже работник, если нет - он создаёт новый веб-объект и выполняет код в "demo_workers.js":

```
if (typeof(w) == "undefined") {
  w = new Worker("demo_workers.js");
}
```

Потом мы можем отправлять и получать сообщение от веб-работника.

Добавьте к веб-работнику слушателя событий "onmessage".

```
w.onmessage = function(event){
  document.getElementById("result").innerHTML = event.data;
};
```

Когда веб-работник отправляет сообщение, выполняется код в слушателе события. Данные веб-работника сохраняются в event.data.

Завершить работу Веб-работника

Когда объект веб-работника создан, он будет продолжать прослушивать сообщение (даже после завершения внешнего скрипта), пока он не будет завершен.

Для прекращения работы веб-работника и свободных ресурсов браузера/компьютера используйте метод `terminate()`:

```
w.terminate();
```

Повторное использование Веб-работника

Если для рабочей переменной задано значение undefined, после её завершения вы можете повторно использовать код:

```
w = undefined;
```

Полный пример кода Веб-работника

Мы уже видели код Веб-работника в файле .js. Ниже приведён код для HTML-страницы:

Пример:

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

<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>

<script>
var w;

function startWorker() {
  if (typeof(Worker) !== "undefined") {
    if (typeof(w) == "undefined") {
```

```
    w = new Worker("demo_workers.js");
   }
   w.onmessage = function(event) {
    document.getElementById("result").innerHTML = event.data;
   };
  } else {
   document.getElementById("result").innerHTML = "Sorry! No Web Worker support.";
  }
}

function stopWorker() {
  w.terminate();
  w = undefined;
}
</script>

</body>
</html>
```

Попробуйте сами »

Веб-работники и DOM

Поскольку веб-работники находятся во внешних файлах, они не имеют доступа к таким объектам JavaScript:

- Объект окна
- Объект документа
- Родительский объект

Prev Next

HTML Доступность

Prev HTML Справочник

HTML Доступность

Пишите HTML с учетом доступности. Предоставьте пользователю хороший способ навигации и взаимодействия с вашим сайтом. Сделайте ваш HTML код как можно более **семантичным**, чтобы его было легко понять как посетителям, так и считывателям экрана (скринридерам).

Семантический HTML

Семантический HTML означает максимально возможное использование правильных элементов HTML для их правильного назначения. Семантические элементы являются элементами со значением; если вам необходима кнопка, используйте элемент `<button>` (а не `<div>`).

Семантический

```
<button>Click Me</button>
```

Попробуйте сами »

Не семантический

```
<div>Click Me</div>
```

Попробуйте сами »

Семантический HTML дает контекст для скринридеров, которые читают содержимое веб-страницы вслух.

С примером кнопки ввиду:

- По умолчанию кнопки имеют более соответствующий стиль
- Скринридер идентифицирует его как кнопку
- Фокусированы
- Кликабельны

Кнопка также доступна для людей, которые полагаются на навигацию с помощью клавиатуры; на неё можно нажимать как мышкой, так и клавишами на клавиатуре, а также можно вводить вкладки между ними (с помощью клавиши табуляции на клавиатуре).

Примеры **не семантических** элементов: `<div>` и `<span>` - ничего не рассказывают о своём содержании.

Примеры **семантических** элементов: `<form>`, `<table>` и `<article>` - чётко определяют своё содержание.

Заголовки важны

Заголовки определяются с помощью тегов от `<h1>` до `<h6>`:

Пример:

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Попробуйте сами »

Поисковые системы используют заголовки для индексирования структуры и содержания ваших веб-страниц.

Пользователи просматривают ваши страницы с заголовками. Важно использовать заголовки, чтобы показать структуру документов и связь между различными разделами.

`<h1>` заголовки следует использовать для основных заголовков, после чего следует использовать `<h2>` заголовки, далее менее важные `<h3>` и так далее.

Примечание: Используйте заголовки HTML лишь для заголовков. Не используйте заголовки просто для создания текста **БОЛЬШИМ** или **жирным**.

Альтернативный текст

Атрибут `alt` предоставляет альтернативный текст для изображения, если пользователь по какой-то причине не может его посмотреть (из-за медленного Интернет-соединения, ошибки в атрибуте `src` или если пользователь использует скринридер).

Значение атрибута `alt` должно описывать изображение:

Пример:

```
<img src="img_chania.jpg" alt="Flowers in Chania">
```

Попробуйте сами »

Если браузер не может найти изображение, он отобразит значение атрибута `alt`:

Пример:

```
<img src="wrongname.gif" alt="Flowers in Chania">
```

Попробуйте сами »

Объявление языка

Объявление языка является важным для скринридеров и поисковых систем, и объявляется атрибутом `lang`.
Используйте приведённый ниже код для отображения веб-страницы на русском языке:

```
<!DOCTYPE html>
<html lang="ru">
<body>

...

</body>
```

```
</html>
```

Используйте "чистый" язык

Используйте чистый язык, который легко понять, и старайтесь избегать символов, которые невозможно прочитать чётко с помощью программы для чтения с экрана (скринридером). Например:

- Сделайте предложения как-можно более короткими.
- Избегайте тире там, где возможно. Вместо того, чтобы писать 1-3, запишите так: от 1 до 3
- Избегайте сокращений там, где возможно. Вместо того, чтобы писать Feb, пишите February
- Избегайте сленговых слов

Пишите хорошие ссылки

Ссылка должна чётко объяснять, какую информацию читатель получит, нажав на эту ссылку.

Примеры хороших и плохих ссылок:

Хорошо:

Узнать больше о языке HTML

Узнать больше о том, как правильно изучать HTML

Приобрести билеты на Марс здесь

Попробуйте сами »

Плохо:

Нажмите здесь

Узнать больше...

Купить билеты на Марс здесь

Попробуйте сами »

Названия ссылок

Атрибут `title` определяет дополнительную информацию об элементе. Эта информация наиболее часто отображается как текст подсказки, когда мышка находится над элементом.

Пример:

```
<a href="https://www.w3schools.com/html/" title="Перейти на W3Schools HTML раздел">Посетите наш HTML Учебник</a>
```

Попробуйте сами »

❮ Prev HTML Справочник ❯

HTML Основы. Атрибуты

❮ Prev Next ❯

Атрибуты предоставляют дополнительную информацию об элементах **HTML**.

Атрибуты HTML. Для чего необходимы атрибуты?

- Все элементы HTML могут иметь атрибуты
- Атрибуты предоставляют дополнительную информацию про элемент
- Атрибуты всегда задаются в начальном теге
- Атрибуты обычно входят в пару имя атрибута/значение, например: name="value" (имя атрибута="значение")

Атрибут href

HTML ссылки определяются тегом `<a>`. Адрес ссылки указан в атрибуте `href`:

Пример:

```
<a href="https://www.w3schools.com">This is a link</a>
```
Попробуйте сами »

Подробнее о ссылках и теге `<a>` можно узнать в следующих разделах этого учебника.

Атрибут src

Изображения в **HTML** определяются тегом `<img>`.

Название файла источника изображения указывается в атрибуте `src`:

Пример:

```
<img src="img_girl.jpg">
```
Попробуйте сами »

Атрибуты width (ширина) и height (высота)

Изображения в **HTML** имеют набор атрибутов размера, который определяет ширину (*width*) и высоту (*height*) изображения:

Пример:

```
<img src="img_girl.jpg" width="500" height="600">
```
Попробуйте сами »

Размер изображения задаётся в пикселях: width = "500" означает ширину 500 пикселей.

Вы узнаете больше об изображениях в разделе Изображения в HTML.

Атрибут alt

Атрибут `alt` указывает альтернативный текст для использования, когда изображение не может быть отображено. Значение атрибута можно прочитать считывателями экрана (скринридерами). Таким образом, кто-то "слушает" веб-страницу, например, человек с нарушениями зрения может "услышать" элемент.

Пример:

```
<img src="img_girl.jpg" alt="Girl with a jacket">
```
Попробуйте сами »

Атрибут `alt` также полезен, если изображение не существует. Посмотрите, что случится, если мы попробуем отобразить изображение, которое не существует:

Пример:

```
<img src="img_typo.jpg" alt="Girl with a jacket">
```
Попробуйте сами »

Атрибут style

Атрибут `style` используется для определения стиля элемента, например цвета, шрифта, размера и т.д.

Пример:

```
<p style="color:red">I am a paragraph</p>
```
Попробуйте сами »

Примечание: Вы узнаете больше про стилизацию в учебнике по CSS.

Атрибут lang

Язык веб-страницы можно объвить в теге `<html>`. Язык объявляется атрибутом `lang`. Объявление языка имеет важное значение для приложений доступности (программы для чтения с экрана) и поисковых систем:

Пример:

```
<!DOCTYPE html>
<html lang="en-US">
<body>
... Content
</body>
</html>
```

Первые две буквы определяют язык (например, en - английский). Если есть диалект, то используют ещё две буквы (US - США). Для русского языка используют буквы ru.

Атрибут title

Здесь атрибут `title` добавляется к элементу `<p>`. Значение атрибута заголовка будет отображаться как всплывающая подсказка, когда вы передвигаете курсор мыши над параграфом:

Пример:

```
<p title="I'm a tooltip">
This is a paragraph.
</p>
```

Попробуйте сами »

Используйте атрибуты нижнего регистра

Стандарт **HTML5** не требует имён атрибутов в нижнем регистре. Атрибут `title` может быть записан в верхнем или нижнем регистре – **title** или **TITLE**.

Примечание: W3C рекомендует нижний регистр в **HTML** и требует нижний регистр для более строгих типов документов, таких как **XHTML**.

Значения атрибутов в кавычках

Стандарт **HTML5** не требует кавычек вокруг значений атрибутов. Атрибут `href`, показаный выше, может быть написан без кавычек:

Пример **(Плохо)**:

```
<a href=https://www.w3schools.com>
```

Попробуйте сами »

Пример **(Хорошо)**:

```
<a href="https://www.w3schools.com">
```

Попробуйте сами »

W3C рекомендует кавычки в **HTML** и требует кавычки для более строгих типов документов, таких как **XHTML**. Иногда просто необходимо использовать кавычки. Этот пример не будет отображать правильно атрибут `title`, поскольку он содержит пробел:

Пример:

```
<p title=About W3Schools>
```

Попробуйте сами »

Примечание: Наиболее распространённым написанием **HTML-кода** является с использованием кавычек. Опускание кавычек может привести к ошибкам. В **W3Schools** всегда используют кавычки вокруг значений атрибутов.

Одинарные или двойные кавычки?

Двойные кавычки вокруг значений атрибутов более распространены в **HTML**, но также можно использовать и одинарные

кавычки. В некоторых ситуациях, когда значение атрибута само по собе содержит двойные кавычки, необходимо использовать одинарные кавычки:

Пример:

```
<p title='John "ShotGun" Nelson'>
```

Или наоборот:

Пример:

```
<p title="John 'ShotGun' Nelson">
```

Попробуйте сами »

Резюме раздела

- Все элементы **HTML** могут иметь **атрибуты**
- Атрибут `title` предоставляет дополнительную информацию в виде "подсказки"
- Атрибут `href` содержит информацию об адресе для ссылок
- Атрибуты `width` (ширины) и `height` (высоты) предоставляют информацию о размере изображений
- Атрибут `alt` предоставляет альтернативный текст для скринридеров (считывателей экрана)
- В **W3Schools** всегда используют имена атрибутов **нижнего регистра**
- В **W3Schools** всегда пишут значения атрибутов с **двойными кавычками**

Проверьте себя с помощью упражнений

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 »

HTML Атрибуты

Ниже приведён алфавитный список некоторых атрибутов, которые часто используются в **HTML**:

Атрибут	Описание
alt	Определяет альтернативный текст для изображения, когда изображение не может быть отображено
disabled	Указывает, что входной элемент должен быть отключен
href	Указывает URL-адрес (веб-адрес) для ссылки
id	Указывает уникальный идентификатор элемента
src	Указывает URL-адрес (веб-адрес) изображения
style	Определяет встроенный CSS стиль для элемента
title	Определяет дополнительную информацию про элемент (отображается как всплывающая подсказка)

Примечание: Полный список всех атрибутов для каждого **HTML** элемента приведён в Справочник HTML атрибутов.

□ Prev Next □

HTML Основы. Основные примеры

□ Prev Next □

Основные примеры написания HTML-кода объектов веб-страницы

В этой теме приводятся примеры только некоторых основных элементов на веб-странице: **заголовки**, **параграфы**, **ссылки**, **изображения**, **кнопки**, **списки**. Они лишь дают общее представление того, как создаются различные объекты на веб-страницах.

Не волнуйтесь, если эти примеры используют теги, которые вы ещё не знаете. Вы узнаете о них в следующих разделах.

HTML документы

Все **HTML** документы должны начинаться с объявления типа документа: `<!DOCTYPE html>`.

Сам **HTML** документ начинается с тега `<html>` и заканчивается тегом `</html>`.

Видимая часть **HTML** документа находится между тегами `<body>` и `</body>`.

Пример:

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

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Попробуйте сами »

Заголовки в HTML

Заголовки (heading - заголовок) в **HTML** определяются тегами от `<h1>` до `<h6>`.

Тег `<h1>` определяет самый важный заголовок. Тег `<h6>` определяет наименее важный заголовок:

Пример:

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
```

Попробуйте сами »

Параграфы (абзацы) в HTML

Параграфы (*paragraph - абзац, параграф*) в **HTML** определяются тегом `<p>`.

Примечание: Слово *paragraph* обычно переводится на русский язык как *абзац*, но в среде веб-разработчиков принято использовать слово *параграф* - так, как оно звучит в оригинале, чтобы не было путаницы.

Пример:

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

Попробуйте сами »

Ссылки в HTML

Ссылки в **HTML** определяются тегом `<a>` (от слова *anchor - якорь*):

Пример:

```
<a href="https://www.w3schools.com">This is a link</a>
```

Попробуйте сами »

Назначение ссылки указывается в атрибуте `href`. Атрибуты используются для предоставления дополнительной информации про элементы **HTML**.

Вы узнаете больше об атрибутах в следующей главе учебника.

Изображения в HTML

Изображения в **HTML** определяются тегом `<img>` (сокращенно от слова image - изображение).

Выходной файл (`src`), альтернативный текст (`alt`), `width` (ширина) и `height` (высота) предоставляются как атрибуты:

Пример:

```
<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">
```

Попробуйте сами »

Кнопки в HTML

Кнопки в **HTML** определяются тегом `<button>` (с анг. button - кнопка):

Пример:

```
<button>Click me</button>
```
Попробуйте сами »

Списки в HTML

Списки в **HTML** определяются тегом `<ul>` (от unordered list - неупорядоченный список / маркер ядро) или `<ol>` (от ordered list - упорядоченный / нумерованный список), за которыми следуют теги `<li>` (элементы списка):

Пример:

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```
Попробуйте сами »

Более подробно об этих и других примерах вы узнаете в следующих разделах сайта.

□ Prev Next □

HTML. Блочные и строчные (встроенные) элементы

□ Prev Next □

HTML. Как создать блочные и встроенные (строчные) элементы на веб-странице?

Каждый HTML элемент имеет значение по умолчанию, которое зависит от типа элемента. Значение по умолчанию для большинства элементов является **блочным** или **встроенным (строчным)**.

Блочные HTML элементы

Блочный элемент всегда начинается с новой строки и занимает всю доступную ширину страницы (вытягивается влево и вправо, насколько это возможно).

Элемент `<div>` является блочным элементом.

Пример:

```
<div>Hello</div>
<div>World</div>
```
Попробуйте сами »

Блочные HTML элементы:

`<address>`
`<article>`
`<aside>`
`<blockquote>`
`<canvas>`
`<dd>`
`<div>`
`<dl>`

<dt>
<fieldset>
<figcaption>
<figure>
<footer>
<form>
<h1>-<h6>
<header>
<hr>

<main>
<nav>
<noscript>

<p>
<pre>
<section>
<table>
<tfoot>

<video>

Встроенные (строчные) HTML элементы

Встроенный (или **строчный**) элемент не запускается на новой строке и занимает лишь столько ширины, сколько необходимо.

Таковым является встроенный в параграф `<span>` элемент.

Пример:

```
<span>Hello</span>
<span>World</span>
```

Попробуйте сами »

Встроенные (строчные) HTML элементы

<a>
<abbr>
<acronym>
<b>
<bdo>
<big>

<button>
<cite>
<code>
<dfn>

<i>
<img>
<input>
<kbd>
<label>
<map>
<object>

<output>
<q>
<samp>
<script>
<select>
<small>

<sub>
<sup>
<textarea>
<time>
<tt>
<var>

Элемент

Элемент часто используется как контейнер для других элементов **HTML**. Элемент не имеет обязательных атрибутов, но style, class и id являются общими. При использовании вместе с **CSS**, элемент может использоваться для стилизации содержимого блоков:

Пример:

```
<div style="background-color:black;color:white;padding:20px;">
  <h2>London</h2>
  <p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
</div>
```

Попробуйте сами »

Элемент

Элемент часто используется как контейнер для некоторого текста внутри строки. Элемент не имеет обязательных атрибутов, но style, class и id являются общими. При использовании вместе с **CSS**, элемент может использоваться для стилизации частей текста:

Пример:

```
<h1>My <span style="color:red">Important</span> Heading</h1>
```

Попробуйте сами »

HTML теги группирования

Тег	Описание
	Определяет блочный элемент в документе
	Определяет встроенный (строчный) элемент в документе

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

□ Prev Next □

HTML Наборы символов. Кодировка страниц

□ Prev Next □

HTML. Кодировка веб-страниц (наборы символов)

Чтобы отобразить **HTML**-страницу правильно, веб-браузер должен знать, какой набор символов (кодировку символов) использовать.

Что такое кодировка символов?

ASCII был первым **стандартом кодировки символов** (также назывался набором символов). ASCII определил 128 разных алфавитно-цифровых символов, которые можно использовать в Интернете: цифры (0-9), английские буквы (A-Z), а также некоторые специальные символы ! $ + - ( ) @ < > .

ANSI (Windows-1252) был оригинальным набором символов Windows с поддержкой 256 разных кодов символов.

Стандарт ISO-8859-1 был типичным набором символов для HTML4. Этот набор символов также поддерживал 256 разных кодов символов.

Поскольку ANSI и ISO-8859-1 были довольно ограничены, **HTML4** также поддерживал **UTF-8**.

Примечание: UTF-8 (Unicode) охватывает почти все знаки и символы в мире.

Кодировка символов по умолчанию для **HTML5** - это **UTF-8**.

HTML атрибут набора символов

Чтобы правильно отобразить **HTML**-страницу, веб-браузер должен знать набор символов (кодировку), которая используется на веб-странице. Это указано в теге `<meta>`:

Для HTML4:

```
<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">
```

Для HTML5:

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

Примечание: Если браузер обнаруживает ISO-8859-1 на веб-странице, он по умолчанию имеет ANSI, поскольку ANSI идентичный ISO-8859-1, за исключением того, что ANSI имеет 32 дополнительных символа.

Отличия между наборами символов

Следующая таблица отображает отличия между наборами символов, описанными выше:

Число	ASCII	ANSI	8859	UTF-8	Описание
32					space
33	!	!	!	!	exclamation mark
34	"	"	"	"	quotation mark
35	#	#	#	#	number sign
36	$	$	$	$	dollar sign
37	%	%	%	%	percent sign
38	&	&	&	&	ampersand
39	'	'	'	'	apostrophe
40	(	(	(	(	left parenthesis
41	)	)	)	)	right parenthesis
42	*	*	*	*	asterisk
43	+	+	+	+	plus sign
44	,	,	,	,	comma
45	-	-	-	-	hyphen-minus
46	.	.	.	.	full stop
47	/	/	/	/	solidus
48	0	0	0	0	digit zero
49	1	1	1	1	digit one
50	2	2	2	2	digit two
51	3	3	3	3	digit three

52	4	4	4	4	digit four
53	5	5	5	5	digit five
54	6	6	6	6	digit six
55	7	7	7	7	digit seven
56	8	8	8	8	digit eight
57	9	9	9	9	digit nine
58	:	:	:	:	colon
59	;	;	;	;	semicolon
60	<	<	<	<	less-than sign
61	=	=	=	=	equals sign
62	>	>	>	>	greater-than sign
63	?	?	?	?	question mark
64	@	@	@	@	commercial at
65	A	A	A	A	Latin capital letter A
66	B	B	B	B	Latin capital letter B
67	C	C	C	C	Latin capital letter C
68	D	D	D	D	Latin capital letter D
69	E	E	E	E	Latin capital letter E
70	F	F	F	F	Latin capital letter F
71	G	G	G	G	Latin capital letter G
72	H	H	H	H	Latin capital letter H
73	I	I	I	I	Latin capital letter I
74	J	J	J	J	Latin capital letter J
75	K	K	K	K	Latin capital letter K
76	L	L	L	L	Latin capital letter L
77	M	M	M	M	Latin capital letter M
78	N	N	N	N	Latin capital letter N
79	O	O	O	O	Latin capital letter O
80	P	P	P	P	Latin capital letter P
81	Q	Q	Q	Q	Latin capital letter Q
82	R	R	R	R	Latin capital letter R
83	S	S	S	S	Latin capital letter S
84	T	T	T	T	Latin capital letter T
85	U	U	U	U	Latin capital letter U
86	V	V	V	V	Latin capital letter V
87	W	W	W	W	Latin capital letter W
88	X	X	X	X	Latin capital letter X
89	Y	Y	Y	Y	Latin capital letter Y
90	Z	Z	Z	Z	Latin capital letter Z
91	[	[	[	[	left square bracket
92	\	\	\	\	reverse solidus
93	]	]	]	]	right square bracket
94	^	^	^	^	circumflex accent
95	_	_	_	_	low line
96	`	`	`	`	grave accent
97	a	a	a	a	Latin small letter a
98	b	b	b	b	Latin small letter b
99	c	c	c	c	Latin small letter c
100	d	d	d	d	Latin small letter d
101	e	e	e	e	Latin small letter e
102	f	f	f	f	Latin small letter f
103	g	g	g	g	Latin small letter g
104	h	h	h	h	Latin small letter h
105	i	i	i	i	Latin small letter i
106	j	j	j	j	Latin small letter j
107	k	k	k	k	Latin small letter k

108	l	l	l	l	Latin small letter l
109	m	m	m	m	Latin small letter m
110	n	n	n	n	Latin small letter n
111	o	o	o	o	Latin small letter o
112	p	p	p	p	Latin small letter p
113	q	q	q	q	Latin small letter q
114	r	r	r	r	Latin small letter r
115	s	s	s	s	Latin small letter s
116	t	t	t	t	Latin small letter t
117	u	u	u	u	Latin small letter u
118	v	v	v	v	Latin small letter v
119	w	w	w	w	Latin small letter w
120	x	x	x	x	Latin small letter x
121	y	y	y	y	Latin small letter y
122	z	z	z	z	Latin small letter z
123	{	{	{	{	left curly bracket
124	|	|	|	|	vertical line
125	}	}	}	}	right curly bracket
126	~	~	~	~	tilde
127	DEL				
128		€			euro sign
129					NOT USED
130		‚			single low-9 quotation mark
131		ƒ			Latin small letter f with hook
132		„			double low-9 quotation mark
133		…			horizontal ellipsis
134		†			dagger
135		‡			double dagger
136		ˆ			modifier letter circumflex accent
137		‰			per mille sign
138		Š			Latin capital letter S with caron
139		‹			single left-pointing angle quotation mark
140		Œ			Latin capital ligature OE
141					NOT USED
142		Ž			Latin capital letter Z with caron
143					NOT USED
144					NOT USED
145		‘			left single quotation mark
146		’			right single quotation mark
147		“			left double quotation mark
148		”			right double quotation mark
149		•			bullet
150		–			en dash
151		—			em dash
152		˜			small tilde
153		™			trade mark sign
154		š			Latin small letter s with caron
155		›			single right-pointing angle quotation mark
156		œ			Latin small ligature oe
157					NOT USED
158		ž			Latin small letter z with caron
159		Ÿ			Latin capital letter Y with diaeresis
160					no-break space
161		¡	¡	¡	inverted exclamation mark
162		¢	¢	¢	cent sign
163		£	£	£	pound sign

164	¤	¤	¤	currency sign
165	¥	¥	¥	yen sign
166	¦	¦	¦	broken bar
167	§	§	§	section sign
168	¨	¨	¨	diaeresis
169	©	©	©	copyright sign
170	ª	ª	ª	feminine ordinal indicator
171	«	«	«	left-pointing double angle quotation mark
172	¬	¬	¬	not sign
173				soft hyphen
174	®	®	®	registered sign
175	¯	¯	¯	macron
176	°	°	°	degree sign
177	±	±	±	plus-minus sign
178	²	²	²	superscript two
179	³	³	³	superscript three
180	´	´	´	acute accent
181	µ	µ	µ	micro sign
182	¶	¶	¶	pilcrow sign
183	·	·	·	middle dot
184	¸	¸	¸	cedilla
185	¹	¹	¹	superscript one
186	º	º	º	masculine ordinal indicator
187	»	»	»	right-pointing double angle quotation mark
188	¼	¼	¼	vulgar fraction one quarter
189	½	½	½	vulgar fraction one half
190	¾	¾	¾	vulgar fraction three quarters
191	¿	¿	¿	inverted question mark
192	À	À	À	Latin capital letter A with grave
193	Á	Á	Á	Latin capital letter A with acute
194	Â	Â	Â	Latin capital letter A with circumflex
195	Ã	Ã	Ã	Latin capital letter A with tilde
196	Ä	Ä	Ä	Latin capital letter A with diaeresis
197	Å	Å	Å	Latin capital letter A with ring above
198	Æ	Æ	Æ	Latin capital letter AE
199	Ç	Ç	Ç	Latin capital letter C with cedilla
200	È	È	È	Latin capital letter E with grave
201	É	É	É	Latin capital letter E with acute
202	Ê	Ê	Ê	Latin capital letter E with circumflex
203	Ë	Ë	Ë	Latin capital letter E with diaeresis
204	Ì	Ì	Ì	Latin capital letter I with grave
205	Í	Í	Í	Latin capital letter I with acute
206	Î	Î	Î	Latin capital letter I with circumflex
207	Ï	Ï	Ï	Latin capital letter I with diaeresis
208	Ð	Ð	Ð	Latin capital letter Eth
209	Ñ	Ñ	Ñ	Latin capital letter N with tilde
210	Ò	Ò	Ò	Latin capital letter O with grave
211	Ó	Ó	Ó	Latin capital letter O with acute
212	Ô	Ô	Ô	Latin capital letter O with circumflex
213	Õ	Õ	Õ	Latin capital letter O with tilde
214	Ö	Ö	Ö	Latin capital letter O with diaeresis
215	×	×	×	multiplication sign
216	Ø	Ø	Ø	Latin capital letter O with stroke
217	Ù	Ù	Ù	Latin capital letter U with grave
218	Ú	Ú	Ú	Latin capital letter U with acute
219	Û	Û	Û	Latin capital letter U with circumflex

220	Ü	Ü	Ü	Latin capital letter U with diaeresis
221	Ý	Ý	Ý	Latin capital letter Y with acute
222	Þ	Þ	Þ	Latin capital letter Thorn
223	ß	ß	ß	Latin small letter sharp s
224	à	à	à	Latin small letter a with grave
225	á	á	á	Latin small letter a with acute
226	â	â	â	Latin small letter a with circumflex
227	ã	ã	ã	Latin small letter a with tilde
228	ä	ä	ä	Latin small letter a with diaeresis
229	å	å	å	Latin small letter a with ring above
230	æ	æ	æ	Latin small letter ae
231	ç	ç	ç	Latin small letter c with cedilla
232	è	è	è	Latin small letter e with grave
233	é	é	é	Latin small letter e with acute
234	ê	ê	ê	Latin small letter e with circumflex
235	ë	ë	ë	Latin small letter e with diaeresis
236	ì	ì	ì	Latin small letter i with grave
237	í	í	í	Latin small letter i with acute
238	î	î	î	Latin small letter i with circumflex
239	ï	ï	ï	Latin small letter i with diaeresis
240	ð	ð	ð	Latin small letter eth
241	ñ	ñ	ñ	Latin small letter n with tilde
242	ò	ò	ò	Latin small letter o with grave
243	ó	ó	ó	Latin small letter o with acute
244	ô	ô	ô	Latin small letter o with circumflex
245	õ	õ	õ	Latin small letter o with tilde
246	ö	ö	ö	Latin small letter o with diaeresis
247	÷	÷	÷	division sign
248	ø	ø	ø	Latin small letter o with stroke
249	ù	ù	ù	Latin small letter u with grave
250	ú	ú	ú	Latin small letter u with acute
251	û	û	û	Latin small letter with circumflex
252	ü	ü	ü	Latin small letter u with diaeresis
253	ý	ý	ý	Latin small letter y with acute
254	þ	þ	þ	Latin small letter thorn
255	ÿ	ÿ	ÿ	Latin small letter y with diaeresis

Набор символов ASCII

ASCII использует значение от 0 до 31 (и 127) для управляющих символов.

ASCII использует значение от 32 до 126 для букв, цифр и символов.

ASCII не использует значение от 128 до 255.

Набор символов ANSI (Windows-1252)

ANSI идентичный ASCII для значений от 0 до 127.

ANSI имеет собственный набор символов для значений от 128 до 159.

ANSI идентичный UTF-8 для значений от 160 до 255.

Набор символов ISO-8859-1

8859-1 идентичный ASCII для значений от 0 до 127.

8859-1 не использует значения от 128 до 159.

8859-1 идентичный UTF-8 для значений от 160 до 255.

Набор символов UTF-8

UTF-8 идентичный ASCII для значений от 0 до 127.

UTF-8 не использует значения от 128 до 159.

UTF-8 идентичный как для ANSI, так и для 8859-1 для значений от 160 до 255.

UTF-8 продолжает со значения 256 с более чем 10 000 разных символов.

Для более детального рассмотрения, посетите Справочник полного набора символов HTML.

CSS правило @charset

Вы можете использовать CSS правило @charset, чтобы указать кодировку символов, которая используется в таблице стилей:

Пример:

Установите кодировку таблицы стилей Unicode UTF-8:

```
@charset "UTF-8";
```

Подробнее про **CSS** правило @charset можно узнать в Справочнике CSS.

□ Prev Next □

HTML. Классы. Атрибут class

□ Prev Next □

HTML. Для чего нужен атрибут класса - class? Использование атрибута class

HTML атрибут `class` используется для определения одинаковых стилей для элементов с тем же именем класса. Таким образом, все **HTML-элементы** с одинаковыми атрибутами класса будут иметь один и тот же формат и стиль.

Например, мы имеем три элемента `<div>`, которые указывают одно имя класса:

Пример:

```
<!DOCTYPE html>
<html>
<head>
<style>
.cities {
  background-color: black;
  color: white;
  margin: 10px;
  padding: 10px;
}
</style>
</head>
<body>

<div class="cities">
  <h2>London</h2>
  <p>London is the capital of England.</p>
</div>

<div class="cities">
```

```
  <h2>Paris</h2>
  <p>Paris is the capital of France.</p>
</div>

<div class="cities">
  <h2>Tokyo</h2>
  <p>Tokyo is the capital of Japan.</p>
</div>

</body>
</html>
```

Результат:

Попробуйте сами »

Использование атрибута class на встроенных элементах

HTML атрибут `class` также можно использовать для встроенных элементов:

Пример:

```
<!DOCTYPE html>
<html>
<head>
<style>
span.note {
  font-size: 120%;
  color: red;
}
</style>
</head>
<body>

<h1>My <span class="note">Important</span> Heading</h1>
<p>This is some <span class="note">important</span> text.</p>

</body>
</html>
```

Попробуйте сами »

Совет: Атрибут `class` можно использовать на любом элементе **HTML**.

Примечание: Название класса чувствительно к регистру!

Совет: Вы можете узнать больше про **CSS** в Учебнике по CSS.

Выберите элементы с определённым классом

В **CSS**, чтобы выбрать элементы с определённым классом, напишите символ точки (.), за которым следует имя класса:

Пример:

Используйте **CSS** для стилизации всех элементов с названием класса "city":

```
<style>
.city {
  background-color: tomato;
  color: white;
  padding: 10px;
}
</style>

<h2 class="city">London</h2>
<p>London is the capital of England.</p>

<h2 class="city">Paris</h2>
<p>Paris is the capital of France.</p>

<h2 class="city">Tokyo</h2>
<p>Tokyo is the capital of Japan.</p>
```

Результат:

London

London is the capital of England.

Paris

Paris is the capital of France.

Tokyo

Tokyo is the capital of Japan.

Попробуйте сами »

Несколько классов

HTML элементы могут иметь больше одного имени класса, каждое имя класса может быть разделено пробелом.

Пример:

Элементы стиля с названием класса "city", также стилевые элементы с названием класса "main":

```
<h2 class="city main">London</h2>
<h2 class="city">Paris</h2>
<h2 class="city">Tokyo</h2>
```

Попробуйте сами »

В приведённом выше примере первый элемент `<h2>` принадлежит как классу "city", так и классу "main".

Разные теги могут использовать один и тот же класс

Разные теги, такие как `<h2>` и `<p>`, могут иметь одинаковое имя класса и тем самым использовать один и тот же стиль:

Пример:

```
<h2 class="city">Paris</h2>
```

```
<p class="city">Paris is the capital of France</p>
```
Попробуйте сами »

Использование атрибута class в JavaScript

Название класса может также использоваться **JavaScript** для выполнения некоторых задач для элементов с заданным именем класса. **JavaScript** может получать доступ к элементам из указанным именем класса с помощью метода `getElementsByClassName()`:

Пример:

Когда пользователь нажимает кнопку, прячутся все элементы с названием класса "city":

```
<script>
function myFunction() {
  var x = document.getElementsByClassName("city");
  for (var i = 0; i < x.length; i++) {
    x[i].style.display = "none";
  }
}
</script>
```
Попробуйте сами »

Примечание: Узнать больше про JavaScript можно в разделе HTML JavaScript или в Учебнике по JavaScript.

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 »

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

□ Prev Next □

HTML Цвета

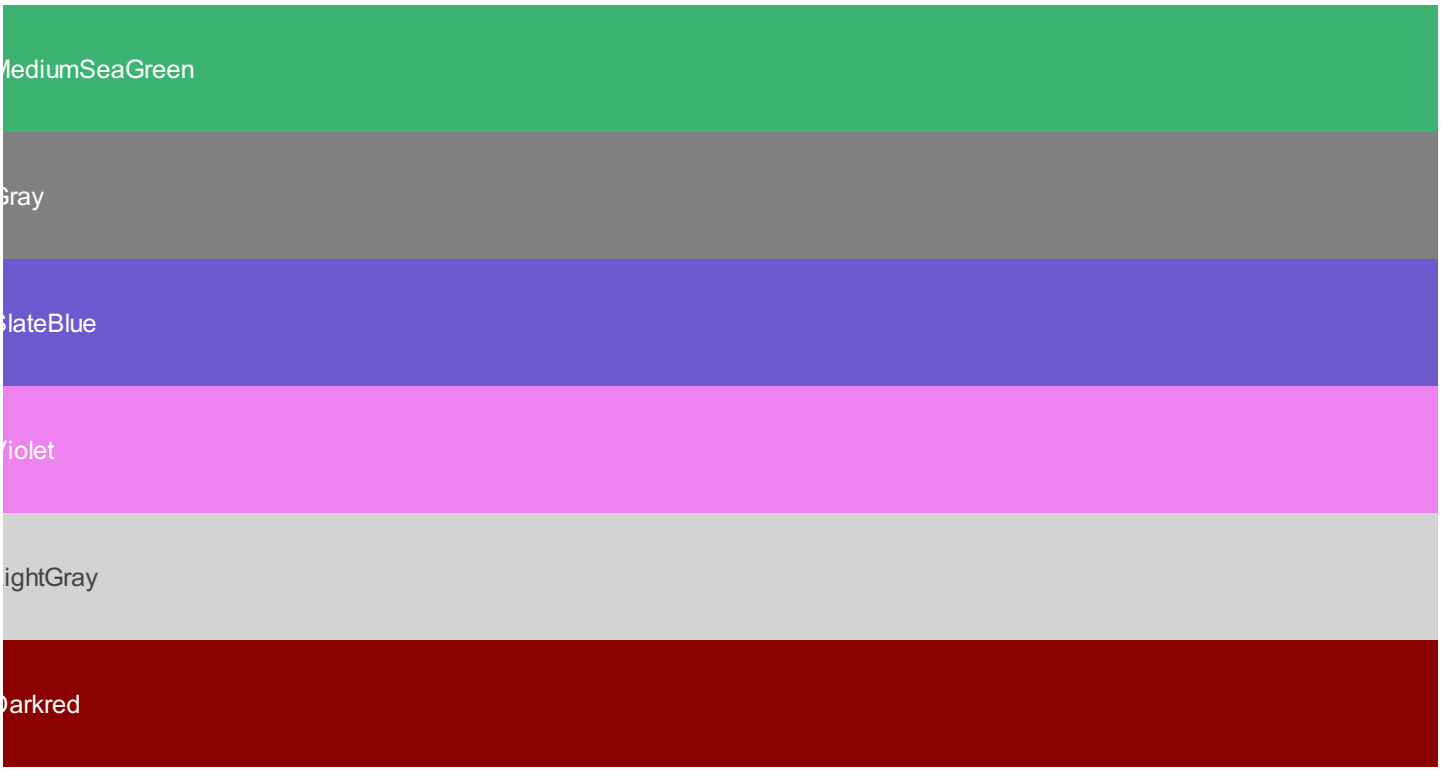
□ Prev Next □

HTML цвета. Как добавлять цвета на веб-страницах? Какие бывают цвета в веб-дизайне?

HTML цвета задаются с помощью предварительно определённых названий цветов или значений *RGB*, *HEX*, *HSL*, *RGBA*, *HSLA*.

Названия цветов

В **HTML** можно указать цвет с помощью названия цвета:



Попробуйте сами »

HTML поддерживает 140 стандартных названия цвета.

HTML Background Color - Цвет фона

Можно установить **цвет фона** для **HTML** элементов:

Hello World

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Пример:

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

Попробуйте сами »

HTML Text Color - Цвет текста

Можно установить **цвет текста**:

Hello World

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Пример:

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
```

```
<p style="color:MediumSeaGreen;">Ut wisienim...</p>
```
Попробуйте сами »

HTML Border Color - Цвет границы

Можно установить **цвет границы**:

Hello World

Hello World

Hello World

Пример:

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```
Попробуйте сами »

Цветовые значения

В **HTML** можно также указать цвета, используя значения **RGB**, значения **HEX**, значения **HSL**, значения **RGBA** и значения **HSLA**:

То ж самое, что и название цвета "Tomato":

```
rgb(255, 99, 71)
```

```
#ff6347
```

```
hsl(9, 100%, 64%)
```

То же самое, что и название цвета "Tomato", но 50% прозрачности:

```
rgba(255, 99, 71, 0.5)
```

```
hsla(9, 100%, 64%, 0.5)
```

Пример:

```
<h2 style="background-color:rgb(255, 99, 71);">...</h2>
<h2 style="background-color:#ff6347;">...</h2>
<h2 style="background-color:hsl(9, 100%, 64%);">...</h2>

<h2 style="background-color:rgba(255, 99, 71, 0.5);">...</h2>
<h2 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h2>
```
Попробуйте сами »

Значение RGB

В HTML цвет можно указывать как значение RGB, используя эту формулу:

rgb (red, green, blue)

что означает: красный, зелёний, синий.

Каждый параметр (красный, зелёний и синий) определяет интенсивность цвета от 0 до 255.

Например, rgb (255, 0, 0) отображается в виде красного цвета, потому что красный устанавливается на его самое большое значение (255), а другие устанавливаются на 0.

Для отображения черного цвета все параметры цвета должны быть установлены в 0, например: rgb (0, 0, 0).

Для отображения белого цвета все параметры цвета должны быть установлены на 255, например: rgb (255, 255, 255).

Поэкспериментируйте самостоятельно, смешивая значения RGB.

Пример:

```
rgb(255, 0, 0)
```

```
rgb(0, 0, 255)
```

```
rgb(60, 179, 113)
```

```
rgb(238, 130, 238)
```

```
rgb(255, 165, 0)
```

```
rgb(106, 90, 205)
```

Попробуйте сами »

Оттенки серого часто определяются с использованием равных значений для всех трёх источников света:

Пример:

```
rgb(0, 0, 0)
```

```
rgb(60, 60, 60)
```

```
rgb(120, 120, 120)
```

```
rgb(180, 180, 180)
```

```
rgb(240, 240, 240)
```

```
rgb(255, 255, 255)
```

Попробуйте сами »

Значение HEX

В **HTML** можно указать цвет, используя шестнадцатеричное значение в виде:

#rrggbb

где **rr** (красный), **gg** (зелёний) и **bb** (синий) являются шестнадцятеричными значениями между **00** и **ff** (такими же, как десятичные 0-255).

Например, #ff0000 отображается в виде красного цвета, поскольку красный устанавливается на его самое большое значение (ff), а другие устанавливаются на самые низкие значения (00).

Пример:

```
#ff0000
```

```
#0000ff
```

```
#3cb371
```

```
#ee82ee
```

```
#ffa500
```

```
#6a5acd
```

Попробуйте сами »

Оттенки серого часто определяются с использованием равных значений для всех трёх источников света:

Пример:

```
#000000
```

```
#3c3c3c
```

```
#787878
```

```
#b4b4b4
```

```
#f0f0f0
```

```
#ffffff
```

Попробуйте сами »

Значения HSL

В **HTML** можно указать цвет с помощью оттенка, насыщенности и лёгкости (осветленности) - (HSL) в виде:

hsl (оттенок, насыщенность, лёгкость)

Оттенок (Hue)

Оттенок - это степень на цветовом колесе от 0 до 360. 0 - красный, 120 - зелёний, а 240 - синий.

Насыщенность - это процентное значение, 0% означает оттенок серого, а 100% - полный цвет.

Лёгкость также составляет процент, 0% - чёрный, 50% - на половину светлый или тёмный, 100% - белый.

Пример:

```
hsl(0, 100%, 50%)
```

```
hsl(240, 100%, 50%)
```

```
hsl(147, 50%, 47%)
```

```
hsl(300, 76%, 72%)
```

```
hsl(39, 100%, 50%)
```

```
hsl(248, 53%, 58%)
```

Попробуйте сами »

Насыщенность (Saturation)

Насыщенность может быть описана как интенсивность цвета.

100% - это чистый цвет, нет серых оттенков.

50% - это 50% серого цвета, но вы всё ещё можете видеть цвет.

0% полностью серый, вы не можете больше видеть цвет.

Пример:

```
hsl(0, 100%, 50%)
```

```
hsl(0, 80%, 50%)
```

```
hsl(0, 60%, 50%)
```

```
hsl(0, 40%, 50%)
```

```
hsl(0, 20%, 50%)
```

```
hsl(0, 0%, 50%)
```

Попробуйте сами »

Лёгкость / Осветлённость / Яркость (Lightness)

Лёгкость цвета можно охарактеризовать как свет, который вы хотите придать цвету, где 0% означает отсутствие света (черный), 50% означает 50% света (ни тёмный, ни светлый), 100% означает полную лёгкость (белый цвет).

Пример:

```
hsl(0, 100%, 0%)
```

```
hsl(0, 100%, 25%)
```

```
hsl(0, 100%, 50%)
```

```
hsl(0, 100%, 75%)
```

```
hsl(0, 100%, 90%)
```

```
hsl(0, 100%, 100%)
```

Попробуйте сами »

Оттенки серого часто определяются установкой оттенка и насыщенности в 0, а также регулируют яркость от 0% до 100%, чтобы получить более тёмные / более светлые оттенки:

Пример:

```
hsl(0, 0%, 0%)
```

```
hsl(0, 0%, 24%)
```

```
hsl(0, 0%, 47%)
```

```
hsl(0, 0%, 71%)
```

```
hsl(0, 0%, 94%)
```

```
hsl(0, 0%, 100%)
```

Попробуйте сами »

Значения RGBA

Значения цветов **RGBA** является расширением значений цветов **RGB** с альфа-каналом, который определяет непрозрачность цвета.

Значение цвета **RGBA** указано с помощью:

rgba (red - красый, green - зелёный, blue - синий, alpha - альфа)

Параметр "alpha" - это число между 0.0 (полностью прозрачным) и 1.0 (совсем не прозрачным):

Пример:

```
rgba(255, 99, 71, 0)
```

```
rgba(255, 99, 71, 0.2)
```

```
rgba(255, 99, 71, 0.4)
```

```
rgba(255, 99, 71, 0.6)
```

```
rgba(255, 99, 71, 0.8)
```

```
rgba(255, 99, 71, 1)
```

Попробуйте сами »

Значение HSLA

Значение цветов **HSLA** является расширением значений цветов **HSL** с помощью альфа-канала, который определяет непрозрачность цвета.

Значение цвета **HSLA** указано с помощью:

hsla (hue - оттенок, saturation - насыщенность, lightness - лёгкость, alpha - альфа)

Параметр "альфа" - это число между 0.0 (полностью прозрачным) и 1.0 (совсем не прозрачным):

Пример:

```
hsla(9, 100%, 64%, 0)
```

```
hsla(9, 100%, 64%, 0.2)
```

```
hsla(9, 100%, 64%, 0.4)
```

```
hsla(9, 100%, 64%, 0.6)
```

```
hsla(9, 100%, 64%, 0.8)
```

```
hsla(9, 100%, 64%, 1)
```

Попробуйте сами »

Таким образом, вы можете использовать разные способы записи цвета в **HTML-разметке**. Но наиболее часто используется написание цвета в шестнадцатеричном значении **HEX** - как в наиболее универсальном и компактном стиле.

Существует множество разных программ и плагинов к браузерам, с помощью которых можно подобрать необходимый вам цвет в любом из цветовых значений: **по названию**, **HEX, RGB, HSL**. Например, онлайн сервисы Color Picker на сайте https://www.w3schools.com/, https://colorscheme.ru/ и др., программы ColorMania, Microsearch Color Picker, Pixie и др., расширения ColorZilla, ColorPicker в Chrome и др.

Справочник HTML тегов

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

□ Prev Next □

HTML Комментарии

□ Prev Next □

HTML комментарии. Как добавлять комментарии на веб-страницах? Как закомментировать HTML код?

Теги комментариев используются для вставки комментариев в исходный **HTML** код.

HTML теги комментариев

Вы можете добавить комментарии на веб-странице к **HTML**-коду, используя следующий синтаксис:

```
<!-- Напишите здесь свои комментарии -->
```

Обратите внимание, что в начальном теге есть восклицательный знак (!), но его нет в закрывающем теге.

Примечание: В браузере не отображаются комментарии, но они помогают документировать исходный **HTML-код**.

С помощью комментариев можно размещать оповещения и напоминания в своём **HTML-коде**:

```
<!-- Это комментарий -->

<p>Это параграф.</p>

<!-- Не забудьте добавить дополнительную информацию здесь -->
```

Попробуйте сами »

Комментарии также чудесно подходят для настройки **HTML-кода**, поскольку вы можете комментировать строки **HTML-кода** по очереди, для поиска ошибок, т.е., закомментировав часть кода, просматривать отображение результата в браузере:

```
<!-- Do not display this at the moment
<img border="0" src="pic_trulli.jpg" alt="Trulli">
-->
```

Попробуйте сами »

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 »

Справочник HTML тегов

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

❮ Prev Next ❯

HTML Компьютерный код

❮ Prev Next ❯

Элементы компьютерного кода на веб-сайте

Компьютерный код:

```
<code>
x = 5;<br>
y = 6;<br>
z = x + y;
</code>
```

Попробуйте сами »

HTML элемент <kbd> для введения с клавиатуры

HTML элемент `<kbd>` представляет введение пользователя, например, введение с клавиатуры или голосовые команды. Текст, окруженный тегами `<kbd>`, обычно отображается в стандартном шрифте браузера по умолчанию:

Пример:

```
<p>Сохраните документ, нажав <kbd>Ctrl + S</kbd></p>
```

Результат:

Сохраните документ, нажав `Ctrl + S`
Попробуйте сами »

HTML элемент <samp> для вывода программы

HTML элемент `<samp>` представляет собой вывод программы или вычислительной системы. Текст, окружённый тегами `<samp>`, обычно отображается в стандартном шрифте браузера по умолчанию:

Пример:

```
<p>Если вы ввели неправильное значение, программа вернёт <samp>Error!</samp></p>
```

Результат:

Если вы ввели неправильное значение, программа вернёт `Error!`
Попробуйте сами »

HTML элемент <code> для компьютерного кода

HTML элемент `<code>` определяет фрагмент компьютерного кода. Текст, который окружен тегами `<code>`, обычно отображается стандартным шрифтом браузера по умолчанию:

Пример:

```
<code>
```

```
x = 5;
y = 6;
z = x + y;
</code>
```

Результат:

```
x = 5; y = 6; z = x + y;
```

Попробуйте сами »

Обратите внимание, что элемент `<code>` не сохраняет дополнительных пробелов и разрывов строк. Чтобы исправить это, вы можете поместить элемент `<code>` в элемент `<pre>`:

Пример:

```
<pre>
<code>
x = 5;
y = 6;
z = x + y;
</code>
</pre>
```

Результат:

```
x = 5;

y = 6;

z = x + y;
```

Попробуйте сами »

HTML элемент <var> для переменных

HTML элемент `<var>` определяет переменную. Переменная может быть переменной в математическом выражении или переменной в контексте программирования:

Пример:

```
Эйнштейн написал: <var>E</var> = <var>mc</var><sup>2</sup>.
```

Результат:

Эйнштейн написал: $E = mc^2$.

Попробуйте сами »

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 »

HTML элементы компьютерного кода

Тег	Описание
`<code>`	Определяет программный код
`<kbd>`	Определяет введение пользователя с клавиатуры
`<samp>`	Определяет исходный компьютерный код
`<var>`	Определяет переменную
`<pre>`	Определяет предварительно отформатированный текст

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

HTML Каскадные таблицы стилей - CSS

□ Prev Next □

HTML/CSS. Как подключить каскадную таблицу стилей (CSS) к веб-странице?

Стилизация HTML с помощью CSS

CSS - (англ. *Cascading Style Sheets — каскадные таблицы стилей*). **CSS** описывает, как элементы **HTML** должны отображаться на экране, бумаге или на других носителях. **CSS** экономит много работы. Таблица стилей может управлять размещением нескольких веб-страниц одновременно.

CSS можно добавить к **HTML** элементам тремя способами:

- Inline (встроенный или строчный) - с помощью атрибута `style` в **HTML**-элементах
- Internal (внутренний) - с помощью элемента `<style>` в разделе `<head>`
- External (внешний) - с помощью внешнего файла **CSS**

Наиболее распространённым способом добавления **CSS** является сохранение стилей в отдельных файлах **CSS**. Однако здесь мы будем использовать встроенный и внутренний стиль, потому что это проще продемонстрировать, и проще попробовать его самостоятельно.

Примечание: Вы можете узнать больше о **CSS** в Учебнике по CSS на нашем сайте на русском языке или на сайте W3schools в оригинале.

Встроенный (inline) CSS

Встроенный (строчный) **CSS** используется для применения уникального стиля к одному элементу HTML. Встроенный **CSS** использует атрибут *style* элемента **HTML**.

Этот пример задаёт цвет тексту элемента `<h1>` синим цветом:

Пример:

```
<h1 style="color:blue;">This is a Blue Heading</h1>
```

Попробуйте сами »

Внутренний (internal) CSS

Внутренний **CSS** используется для определения стиля одной страницы **HTML**, т.е. пишется и применяется только на одной странице.

Внутренний **CSS** определяется в разделе `<head>` **HTML** страницы в элементе `<style>`:

Пример:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```
Попробуйте сами »

Внешний (external) CSS

Для определения стиля для нескольких **HTML** страниц используется внешняя таблица стилей. С помощью внешней таблицы стилей вы можете изменить вид всего веб-сайта, изменив лишь один **CSS** файл!

Чтобы использовать внешнюю таблицу стилей, добавьте ссылку на **CSS** файл в разделе `<head>` **HTML** страницы:

Пример:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```
Попробуйте сами »

В любом текстовом редакторе можно написать внешнюю таблицу стилей. Файл не должен содержать никакого **HTML**-кода и должен быть сохранён с расширением **.css**.

Какие современные текстовые редакторы можно использовать для создания и редактирования CSS кода, вы можете узнать на нашем сайте в теме Редакторы кода

Вот как выглядит файл стилей "styles.css":

Пример:

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

CSS Fonts - Шрифты

Свойство **CSS** `color` определяет цвет текста, который будет использоваться.

Свойство **CSS** `font-family` определяет используемый шрифт (семейство шрифтов).

Свойство **CSS** `font-size` определяет размер текста, который будет использоваться.

Пример:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: blue;
  font-family: verdana;
  font-size: 300%;
}
p  {
  color: red;
  font-family: courier;
  font-size: 160%;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Попробуйте сами »

CSS Border - Граница

Свойство **CSS** `border` определяет границу вокруг **HTML** элемента:

Пример:

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

Попробуйте сами »

CSS Padding - Внутренний отступ

Свойство **CSS** `padding` определяет добавление пространства (пробела) между текстом и границей **border** (внутренний отступ):

Пример:

```
p {
  border: 1px solid powderblue;
  padding: 30px;
}
```

Попробуйте сами »

CSS Margin - Внешний отступ

CSS свойство `margin` определяет пространство (пробел) за пределами границы **border** (внешний отступ):

Пример:

```
p {
  border: 1px solid powderblue;
  margin: 50px;
}
```

Попробуйте сами »

Атрибут Id (идентификатора)

Чтобы определить определённый стиль для одного специального элемента, добавьте к этому элементу атрибут `id`:

```
<p id="p01">I am different</p>
```

потом определите стиль для элемента с определённым идентификатором:

Пример:

```
#p01 {
  color: blue;
}
```
Попробуйте сами »

Примечание: Идентификатор элемента должен быть уникальным в пределах страницы, поэтому селектор идентификаторов **Id** используется для выбора одного уникального элемента!

Атрибут Class (класс)

Чтобы определить стиль для множества специальных типов элементов, добавьте к элементу атрибут `class`:

```
<p class="error">I am different</p>
```

потом определите стиль для элементов с определённым классом:

Пример:

```
p.error {
  color: red;
}
```
Попробуйте сами »

Внешние ссылки на таблицы стилей

На внешние таблицы стилей можно ссылаться с полным URL-адресом или путём, связанным с текущей веб-страницей.

Этот пример использует полный URL-адрес для ссылки на таблицу стилей:

Пример:

```
<link rel="stylesheet" href="https://www.w3schools.com/html/styles.css">
```
Попробуйте сами »

Этот пример ссылается на таблицу стилей, расположенную в папке **html** на текущем веб-сайте:

Пример:

```
<link rel="stylesheet" href="/html/styles.css">
```
Попробуйте сами »

Этот пример ссылается на таблицу стилей, расположенную в той же папке, что и текущая страница:

Пример:

```
<link rel="stylesheet" href="styles.css">
```
Попробуйте сами »

Примечание: Подробнее про пути к файлам вы можете прочитать в разделе HTML Пути к файлам.

Резюме раздела

- Используйте HTML атрибут `style` для встроенного стиля
- Используйте HTML элемент `<style>` для определения внутренних **CSS**
- Используйте HTML элемент `<link>`, чтобы ссылаться на внешний **CSS** файл
- Используйте HTML элемент `<head>` для сохранения элементов `<style>` и `<link>`
- Используйте CSS свойство `color` для задания цвета тексту
- Используйте CSS свойство `font-family` для задания тексту семейства шрифтов
- Используйте CSS свойство `font-size` для задания размеров тексту
- Используйте CSS свойство `border` для задания границы элементу
- Используйте CSS свойство `padding` для задания внутреннего отступа

- Используйте CSS свойство `margin` для задания внешнего отступа (за пределами border)

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 » Упражнение 5 » Упражнение 6 »

HTML теги стиля

Тег	Описание
`<style>`	Определяет информацию про стиль для HTML-документа
`<link>`	Определяет связь между документом и внешним ресурсом

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

Prev Next

HTML Основы. Редакторы кода

Prev Next

Редакторы кода. На чём писать HTML-код?

Писать **HTML** код можно, используя стандартные программы **Блокнот** (на Windows) и **TextEdit** (на MacOS). Но сейчас существует довольно большое количество разнообразных профессиональных текстовых редакторов, с помощью которых можно писать **HTML**-код. Хотя для изучения языка **HTML** рекомендуется сначала писать код как-раз с помощью простых текстовых редакторов - стандартных **Блокнота** или **TextEdit**. А научившись писать простой **HTML** код, потом перейти до более профессиональных редакторов.

Выполните четыре шага ниже, чтобы создать свою первую веб-страницу с помощью **Блокнота** или **TextEdit**.

Шаг 1. Откройте программу Блокнот (на Windows)

Windows 8 или более поздней версии:

Откройте начальный экран (символ окна внизу слева на экране). Выберите программу **Блокнот**.

Windows 7 или более ранней версии:

Нажмите меню **Пуск** > **Программы** > **Стандартные** > **Блокнот**

Шаг 1. Откройте TextEdit (Mac)

Откройте **Finder** > **Программы** > **TextEdit**.

Также измените некоторые настройки, чтобы программа могла правильно сохранять файлы. В меню **Параметры** > **Формат** выберите **"Обычный текст"**.

Потом в разделе "Открыть и сохранить" установите флажок **"Показывать HTML-файлы как HTML-код вместо форматированного текста"**.

Потом откройте новый документ, чтобы разместить код.

Шаг 2. Напишите HTML код

Перепишите или скопируйте данный **HTML**-код в **Блокнот**:

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

<h1>My First Heading</h1>
```

```
<p>My first paragraph.</p>

</body>
</html>
```


Отображение HTML-кода в Блокноте

Шаг 3. Сохраните HTML страницу

Сохраните файл на компьютере. Выберите **Файл** > **Сохранить как** в меню **Блокнота**.

Назовите файл **"index.htm"** и установите кодировку **UTF-8** (которая является лучшей кодировкой для файлов **HTML**).

Сохраниение текстового файла в виде html-страницы в Блокноте

Примечание: Вы можете использовать *.htm* или *.html* как расширение файла. Они идентичны. Разницы нет, выбор зависит от вас.

Шаг 4. Откройте и посмотрите созданную HTML-страницу в браузере

Откройте сохранённый **HTML**-файл в своём любимом веб-браузере (дважды кликните файл или кликните правой кнопкой мыши на данном файле и выберите "Открыть с помощью", выбрав необходимый браузер).

Результат будет выглядеть так:

HTML-страница в браузере

Онлайн-редактор W3Schools

С бесплатным онлайн-редактором на сайте W3Schools вы можете редактировать **HTML**-код и просматривать результат в вашем браузере. Это идеальный инструмент для быстрого тестирования кода. Редактор также имеет цветное кодирование и возможность сохранять и совместно использовать код с другими пользователями:

Пример:

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

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Попробуй сам »

Нажмите кнопку "Попробуй сам", чтобы узнать, как она работает.

Примечание: В сети существует множество различных онлайн-редакторов кода, которые вы также можете использовать (бесплатно). Например, кроме онлайн-редактора от w3schools.com, есть такие сервисы, как: jsfiddle.net, codepen.io, thimble.mozilla.org, jsbin.com, rapprogtrain.com, liveweave.com

---

С помощью каких програм-редакторов пишут код профессональные веб-программисты и веб-верстальщики?

- Какие текстовые редакторы используют профессиональные веб-разработчики?
- Редактор **Notepad ++**
- Редактор **Brackets**
- Редактор **Sublime Text 3**
- Редактор **Atom**

- IDE **Visual Studio Code**
- IDE **WebStorm / PHPStorm**
- IDE **Dreamweaver**
- IDE **NetBeans**

Сейчас существует довольно много разных редакторов **HTML**-кода. Ещё каких-то 12-15 лет назад многих нынешних самых популярных **HTML**-редакторов даже не существовало. Многие из бывших начинающих веб-дизайнеров и веб-верстальщиков начинали писать свои первые веб-страницы с помощью стандартной программы на Windows - **Блокнота**. Но сейчас, конечно же, никто с помощью **Блокнота HTML-код** уже не пишет. Для этого существуют другие более удобные редакторы кода с подсветкой синтаксиса и разными дополнительными функциями, которые облегчают написание кода.

Первым **HTML**-редактором, который раньше использовали после стандартного **Блокнота** Windows часто становился **Notepad ++**. Это свободный текстовый редактор с открытым исходным кодом для Windows с подсветкой синтаксиса большого количества языков программирования и разметки. Поддерживает открытие более 100 форматов. Базовая функциональность программы может быть расширена как за счет плагинов, так и сторонних модулей, таких как компиляторы и препроцессоры. Именно с помощью **Notepad ++** можно начинать писать свои первые веб-страницы любым новичкам. Программа небольшая, довольно удобная, имеет русский и украинский интерфейс, её можно скачать бесплатно с официального сайта. Есть портативный вариант (portable) программы, который не требует установки. Её достаточно скачать и распаковать в любую директорию на жестком диске вашего компьютера и пользоваться.

Второй редактор **HTML**-кода - **Brackets** - также бесплатный редактор из открытым кодом для веб-разработчиков. **Brackets** ориентирован на работу с **HTML**, **CSS** и **JavaScript**. Эти же технологии лежат в основе самого редактора, что обеспечивает его кроссплатформенность, т.е. совместимость с операционными системами Mac, Windows и Linux. Большую функциональность этому редактору дают множество расширений (плагинов), что добавляет необходимые инструменты для роботы с кодом. Скачать **Brackets** можно с официального сайта.

Ещё один чудесный редактор **HTML**-кода - это **SublimeText3**. Программа условно-бесплатна. Её можно скачать с официального сайта (или отсюда) и пользоваться абсолютно бесплатно. Единственное неудобство в этом случае - это периодическое появление сообщения о необходимости купить программу. Хотя на просторах Интернета можно найти **ключи активации для SublimeText3**. Также на официальном сайте **SublimeText3** есть только англоязычная версия программы. Но в Интернете легко можно найти и скачать пакеты с переводом на русский или украинский языки и, придерживаясь рекомендаций, самостоятельно их установить в **SublimeText3**. В целом этот **HTML редактор** довольно быстрый, простой и понятный в использовании, но всё-таки рекомендуется именно для более опытных пользователей и верстальщиков сайтов, потому что для расширения функциональности в **SublimeText3** необходимо дополнительно доустановить некоторые дополнения - плагины, которых существует огромное количество. Но именно плагины и дают все те удобства для этого редактора **HTML-кода**.

Следующий редактор **HTML-кода**, и он же интегрированная среда разработки - это **Atom**. Редактор **Атом** от команды *GitHub* предоставляет средства для крос-платформенного редактирования кода, имеет интеллектуальную систему автодополнения ввода и многое другое. Есть множество дополнений и в этого редактора. Хотя для новичков он будте всё-таки довольно сложным, поэтому рекомендуется для более опытных пользователей. **Редактор Atom** можно скачать бесплатно с официального сайта.

В Интернете можно, конечно же, найти и другие **HTML-редакторы**. Но все они, как правило, очень похожи на вышеперечисленные. Каждый может выбрать себе редактор по своему вкусу и степени сложности для освоения.

Кроме текстовых редакторов, которые используются для написания **HTML/CSS** - кода, профессиональные веб-разработчики используют также **IDE** - (сокращенно от англ. Integrated development environment) - Интегрированную среду разработки. Интегрированные среды разработки созданы для того, чтобы максимизировать продуктивность программиста, предоставив ему связанные инструменты разработки из похожими интерфейсами как одну программу, в которой происходит весь процесс разработки и которая предоставляет необходимые функции. **IDE** помагают увеличить продуктивность разработчика и ускорить процесс разработки и написания кода.

Наиболее популярными **IDE** среди веб-разработчиков по состоянию на 2019 год являются такие: **VS Code** - (бесплатный), **WebStorm** - (платный), **PHPStorm** - (платный), **Dreamweaver** - (платный), **NetBeans** - (бесплатный) и др.

Некоторые из этих IDE есть бесплатными, а некоторые платными. Хотя **WebStorm** и **PHPStorm** имеют довольно большой период бесплатной работы (trial) - 30 дней, а для студентов и преподавателей можно получить эти IDE на 2 года бесплатно в целях обучения. Также на просторах Интернета можно найти **ключи активации PHPStorm**.

HTML Основы. Элементы

Prev Next

Элементы HTML-кода. С чего состоит веб-страница?

Элемент в HTML обычно состоит из **начального** тега и **конечного** тега, содержание которого вставлено между ними:

`<Начальный тег>` Содержание размещается здесь ... `</Конечный тег>`

Элемент HTML - это всё от начального тега до конечного тега:

```
<p>My first paragraph.</p>
```

Начальный тег	Содержание элемента	Конечный тег
<h1>	My First Heading	</h1>
	My first paragraph.	

Примечание: Элементы **HTML** без содержания называются пустыми элементами. Пустые элементы не имеют конечного тега, например элемент `<br>`, который указывает на разрыв строки (*break – разрыв*)

Вложенные элементы HTML

Элементы **HTML** могут быть вложенными (элементы могут содержать элементы). Все документы **HTML** состоят из вложенных элементов **HTML**.

Этот пример содержит четыре **HTML элемента**:

Пример:

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

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Попробуйте сами »

Объяснение примера

Элемент `<html>` определяет весь документ (от *HyperText Markup Language*). Он имеет **начальный тег** `<html>` и **конечный тег** `</html>`. Содержание (**контент**) элемента - это другой элемент **HTML** (элемент `<body>`).

Пример:

```
<html>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Элемент `<body>` определяет тело документа (*body - тело*). Он имеет **начальный тег** `<body>` и **конечный тег** `</body>`. Содержание (**контент**) элемента - это два других элемента **HTML** (`<h1>` и `<p>`).

Пример:

```
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
```

Элемент `<h1>` определяет заголовок (сокращенно от *heading - заголовок*). Он имеет **начальный тег** `<h1>` и **конечный тег** `</h1>`. Содержание (**контент**) элемента: My First Heading.

Пример:

```
<h1>My First Heading</h1>
```

Элемент `<p>` определяет параграф (сокращенно от *paragraph - абзац, параграф*). Он имеет начальный тег `<p>` и конечный тег `</p>`. Содержание (**контент**) элемента: My first paragraph.

Пример:

```
<p>My first paragraph.</p>
```

Не забывайте конечный тег

Некоторые элементы **HTML** будут отображаться правильно, даже если вы забыли **конечный тег**:

Пример:

```
<html>
<body>

<p>This is a paragraph
<p>This is a paragraph

</body>
</html>
```

Попробуйте сами »

Приведённый выше пример работает во всех браузерах, поскольку **закрывающий тег** считается необязательным.

Никогда не полагайтесь на это. Иначе это может привести к неожиданным результатам и / или ошибкам, если вы забудете **конечный тег**.

Пустые элементы HTML

Элементы **HTML** без содержания называются **пустыми элементами**.

`<br>` - это пустой элемент без закрывающего тега (тег **
** определяет разрыв строки):

Пример:

```
<p>This is a <br> paragraph with a line break.</p>
```

Попробуйте сами »

Пустые элементы могут быть "закрыты" в начальном теге: `<br />`.

HTML5 не требует закрытия пустых элементов. Но если вы хотите более суровой проверки, или если вам необходимо сделать ваш документ читаемым XML-парсерами, вы должны закрыть все **HTML-элементы** надлежащим образом.

Используйте строчные теги

Теги **HTML** не чувствительны к регистру: `<P>` (в большом регистре) означает то же, что и `<p>` (в маленьком регистре).

Стандарт **HTML5** не требует использования тегов в нижнем регистре, но *W3C* рекомендует нижний регистр для написания **HTML-кода** и требует нижнего регистра для более строгих типов документов, таких как **XHTML**.

Примечание: В **W3Schools** всегда используют теги нижнего регистра.

□ Prev Next □

HTML Символьные объекты

□ Prev Next □

Символьные объекты на веб-страницах

Зарезервированные символы в **HTML** должны быть заменены символьными объектами. Символы, которых нет на клавиатуре, также могут быть заменены объектами.

HTML символьные объекты

Некоторые символы зарезервированы в **HTML**. Если вы используете в вашем тексте знаки "меньше чем" (<) или "больше чем" (>), браузер может смешивать их с тегами. Символьные объекты используются для отображения зарезервированных символов в **HTML**.

Структура написания символа выглядит так:

&*название_символа*;

или

&#*номер_символа*;

Чтобы отобразить знак «меньше» (<), мы должны написать: `<` или `&#60;`

Примечание: Преимущество использования имени объекта - название объекта легко запоминается. Недостаток использования названия объекта - браузеры могут не поддерживать все имена объектов, но поддержка номеров хорошая.

Неразрывное пространство

Общим символом, используемым в **HTML**, является неразрывное пространство ` `

Неразрывное пространство - это пространство, которое не прорывается в новую строку. Два слова, разделённые неразрывным пространством, слипаются вместе (не разбиваются на новую строку). Это удобно, когда слова разделяются.

Примеры:

- § 10
- 10 км / час
- 10 PM

Другое распространённое использование неразрывного пространства состоит в том, чтобы запретить браузерам отсекать пробелы в HTML-страницах. Если вы пишете 10 пробелов в вашем тексте (с помощью клавиши "Пробел"), браузер удалит 9 из них. Чтобы добавить настоящие пробелы к тексту, можно воспользоваться символьным объектом ` `.

Примечание: Неразрывный дефис (‑) позволяет использовать дефис (-), который не разрывается.

Некоторые другие полезные символьные HTML объекты

Результат	Описание	Название символьного объекта	Номер символьного объекта
	неразрывное пространство		
<	меньше чем	<	<
>	больше чем	>	>
&	амперсанд	&	&
"	двойные кавычки	"	"

'	одинарные кавычки (апостроф)	&apos;	&#39;
¢	цент	&cent;	&#162;
£	фунт	&pound;	&#163;
¥	ена	&yen;	&#165;
€	евро	&euro;	&#8364;
©	авторское право	&copy;	&#169;
®	зарегистрированный товарный знак	&reg;	&#174;

Примечание: Названия символьных объектов чувствительны к регистру.

Объединение диакритических знаков

Диакритический знак - это "глиф", который добавляется к букве. Некоторые диакритические знаки, такие как важность (̀) и ударение (́), называются акцентами.

Диакритические знаки могут появляться как над, так и под буквой, внутри буквы и между двумя буквами.

Диакритические знаки могут использоваться в соединении с буквенно-цифровыми символами для создания символа, который отсутствует в наборе символов (кодировке), используемом на странице.

Вот несколько примеров.

Знак	Символ	Конструкция	Результат
̀	a	à	à
́	a	á	á
̂	a	â	â
̃	a	ã	ã
̀	O	Ò	Ò
́	O	Ó	Ó
̂	O	Ô	Ô
̃	O	Õ	Õ

В следующем разделе этого учебника вы увидите больше **HTML-символов**.

□ Prev Next □

W3Schools HTML Certificate. Сертификат

□ Prev Next □

Как это работает?

- Учитесь бесплатно на сайте W3Schools.com (или на его русскоязычной версии);
- Учитесь со своей собственной скоростью;
- Проверьте свои навыки с онлайн-викторинами (тестами) W3Schools;
- Подайте заявку на получение сертификата, оплатив стоимость экзамена;
- Сдайте свой экзамен онлайн в любое удобное для вас время и находясь в любом месте.

❮ Prev Next ❯

HTML Примеры

❮ Prev Next ❯

HTML Основы

HTML документ HTML заголовки HTML параграфы HTML ссылки HTML изображения HTML кнопки HTML списки

Объяснение примеров

HTML Атрибуты

Атрибут title Атрибут href Атрибути width и height Атрибут alt Атрибут без кавычек Атрибут без кавычек не работает

Объяснение примеров

HTML Заголовки

HTML заголовки HTML горизонтальная разделительная линия HTML голова

Объяснение примеров

HTML Параграфы

HTML параграфы Больше HTML параграфов Использование разрывов строк в HTML Проблемы с отображением стихов (некоторые проблемы с форматированием HTML) Как контролировать разрывы строк и пробелы с помощью тега <pre>

Объяснение примеров

HTML Стили

HTML стили HTML цвет фона HTML цвет текста HTML семейство шрифта HTML размер текста HTML выравнивание текста

Объяснение примеров

HTML Форматирование текста

Жирное форматирование с помощью элемента <b> Семантически сильное форматирование (выделение) с помощью элемента Форматирование курсивом с помощью элемента <i> Семантически подчёркнутое форматирование с помощью элемента Уменьшение шрифта с помощью элемента <small> Пометка с помощью элемента <mark> Пометка зачёркиванием с помощью элемента <del> Пометка подчёркиванием с помощью элемента <ins> Пометка зачёркиванием и подчёркиванием с помощью <del> и <ins> Форматирование нижнего индекса с помощью элемента <sub> Форматирование верхнего индекса с помощью элемента <sup>

Объяснение примеров

HTML Цитаты и цитирование

Форматирование коротких цитат с помощью элемента <q>. Форматирование цитируемых разделов с элементом

<blockquote>. Форматирование информации про автора/владельца документа с помощью элемента <address> Форматирование сокращений и аббревиатур с помощью элемента <abbr> Форматирование названия работ с помощью элемента <cite> Форматирование направления текста с помощью элемента <bdo>

Объяснение примеров

HTML Комментарии

Скрытые комментарии Условные комментарии Комментарии для настройки

Объяснение примеров

HTML CSS

HTML из встроенным CSS HTML с внутренним CSS HTML из внешним CSS HTML из шрифтами CSS HTML из CSS с использованием атрибута id HTML из CSS с использованием атрибута class HTML и CSS границы HTML и CSS внутренний отступ (padding) HTML и CSS внешний отступ (margin) HTML и CSS полная демонстрация

Объяснение примеров

HTML Ссылки

Ссылка с использованием абсолютного URL Ссылка с использованием относительного URL Смена цвета ссылок Удаление подчеркивания из ссылок Смена цели ссылок Изображение как ссылка Создание ссылки на закладку Ссылка, которая выходит с фрейма Ссылка на электронную почту Ссылка на почту с темой

Объяснение примеров

HTML Изображение

Изображение Высота и ширина изображения с использованием атрибутов Высота и ширина изображения с помощью CSS Высота и ширина изображения с помощью атрибутов и CSS Изображение в другой папке Изображение с неработающей ссылкой Изображение на другом сервере Использование изображения как ссылки Движущее изображение Карта изображений из интерактивными областями, которые можно нажимать Плавающее изображение

Объяснение примеров

HTML Таблицы

Основы HTML таблиц Таблица. Границы Таблица. Свёрнутые границы Таблица. Отступ в ячейках Таблица. Заголовки Таблица. Выравнивание заголовков Горизонтальные/вертикальные заголовки таблицы Таблица. Подпись Ячейки таблицы, которые охватывают больше одного столбца Ячейки таблицы, которые охватывают больше одной строки Таблица. Интервалы между ячейками Таблица с HTML-тегами внутри Таблицы с другим стилем с использованием id I Таблицы с другим стилем с использованием id II Таблицы с разным стилем с использованием класса I Таблицы с другим стилем с использованием класса II

Объяснение примеров

HTML Списки

Неупорядоченный список (по умолчанию) Неупорядоченный список - disc маркеры Неупорядоченный список - circle маркеры Неупорядоченный список - square маркеры Неупорядоченный список без маркеров Упорядоченный список (по умолчанию) Упорядоченный список с числами Упорядоченный список с буквами Упорядоченный список с маленькими буквами Упорядоченный список с римскими числами Упорядоченный список с маленькими римскими числами Список описания Вложенный список I Вложенный список II Горизонтальный список Горизонтальный список - меню

Объяснение примеров

HTML Блочные и строчные элементы

Стилизация элементов Стилизация элементов

Объяснение примеров

HTML Классы

Стилизируйте все элементы с указанным именем класса Доступ к элементам с заданным именем класса с JavaScript Несколько классов Тот же класс, другой тег

Объяснение примеров

HTML Id

Стиль элемента с определённым Id Разница между class и id Доступ к елементу с определённым Id с JavaScript

Объяснение примеров

HTML Макет

Макет с использованием float Макет с использованием flexbox Макет с использованием flexbox 2 Макет с использованием flexbox 3

Объяснение примеров

HTML IFrame

Встроенный фрейм (фрейм в HTML странице) Объяснение примеров

HTML Элементы внутри head

Валидный HTML документ при отсутствии <html> <body и <head> Валидный HTML документ при отсутствии элемента <head> Элемент <title> определяет название документа Элемент <style> содержит информацию про стиль Элемент <link> определяет отношение к внешнему ресурсу Элемент <meta> определяет специальную мета-информацию Элемент <script> определяет на стороне клиента JavaScripts Элемент <base> определяет основной URL-адрес сайта для всех URL-адресов

Объяснение примеров

HTML Скрипты

Вставка скрипта Использование тега <noscript>

Объяснение примеров

HTML Элементы компьютерного кода

Форматирование введения с клавиатуры с помощью элемента <kbd> Форматирование вывода компьютера с помощью элемента <samp> Форматирование программного кода с помощью элемента <code> Форматирование программного кода из сохранением пробелов и разрывов строк Сменное форматирование с помощью элемента <var>

Объяснение примеров

HTML Формы

Форма с вводом текста Форма ввода с радио-кнопкой Форма с текстовыми полями и кнопкой отправки Форма с текстовыми полями без атрибута name Группировка данных формы

Объяснение примеров

HTML Элементы формы

Простой раскрывающийся (выпадающий) список Выпадающий список из предварительно выбранным значением Текстовая область (многострочное поле ввода текста) Кнопка ввода Использование элемента <datalist> Использование элемента <output>

Объяснение примеров

HTML Типы ввода

Тип ввода text Тип ввода password Тип ввода radio Тип ввода checkbox Тип ввода button Тип ввода number - с ограничениями Тип ввода number - с шагами Тип ввода date - с выбором даты Тип ввода date - с ограничениями Тип ввода color - с выбором цвета Тип ввода range Тип ввода month Тип ввода week Тип ввода time Тип ввода datetime Тип ввода datetime-local Тип ввода email Тип ввода search Тип ввода tel Тип ввода url

Объяснение примеров

HTML Атрибуты ввода

Атрибут autocomplete Атрибут novalidate Атрибут autofocus Атрибут form Атрибут formaction Атрибут formenctype Атрибут formmethod Атрибут formnovalidate Атрибут formtarget Атрибути height и width Атрибут list Атрибути min и max Атрибут multiple Атрибут pattern Атрибут placeholder Атрибут required Атрибут step

Объяснение примеров

HTML5 Canvas

Рисовать на холсте с помощью JavaScript Рисовать строку с lineTo() Рисовать круг с arc() Рисовать текст с fillText() Рисовать текст с strokeText() Рисовать линейный градиент Рисовать круговой градиент Рисовать изображение с drawImage()

Объяснение примеров

HTML5 SVG

SVG круг SVG прямоугольник SVG закругленный прямоугольник SVG звезда SVG лого

Объяснение примеров

HTML5 Медиа

Воспроизведение видео с кроликом Воспроизведение видео медведя с элементами управления Воспроизведение видео медведя с автозапуском Воспроизведение звука коня с элементами управления

Объяснение примеров

HTML5 Геолокация

Получить координаты геолокации Обработать ошибки геолокации Получить геолокацию с картой Получить геолокацию с помощью скрипта карты Google Получить геолокацию и наблюдать за позицией

Объяснение примеров

HTML5 Локальное хранилище

Сохранять название постоянно Сохранять счетчик постоянно Сохранять счетчик для одного сеанса

Объяснение примеров

HTML5 Медиа

Воспроизвести видео файл в HTML Воспроизвести аудио файл в HTML Воспроизвести YouTube видео в HTML

Объяснение примеров

Больше HTML5 примеров

HTML Упражнения. Проверочные задания

❮ Prev Next ❯

Вы можете проверить свои навыки (скилы) HTML с упражнениями W3Schools.

Упражнения

Мы собрали разные HTML упражнения (с ответами) для каждого HTML-раздела.

Попробуйте решить упражнение, отредактировав некоторый код. Получите "намёк", если вы застряли, или нажмите "показать ответ", чтобы увидеть, что вы сделали неправильно.

Посчитайте свои баллы

Вы получаете 1 балл за каждый правильный ответ. Ваш показатель и общий балл всегда будут отображаться.

Начало HTML упражнений

Удачи Вам!

Начать HTML упражнения ❯

Если вы ещё не знаете HTML, предлагаем вам прочитать HTML Учебник с самого начала.

HTML Пути к файлам

Как установить пути к файлам на веб-сайте?

Путь	Описание
<img src="picture.jpg">	picture.jpg находится в той же папке, что и текущая страница
<img src="images/picture.jpg">	picture.jpg находится в папке images в текущей папке
<img src="/images/picture.jpg">	picture.jpg находится в папке images в корне текущего сайта
<img src="../picture.jpg">	picture.jpg находится в папке на один уровень выше текущей папки

Путь к HTML файлу

Путь к файлу описывает размещение файла в структуре папок веб-сайта.

Пути к файлам используются при ссылке на внешние файлы, такие как:

- Веб-страницы
- Изображения
- Таблицы стилей
- JavaScript

Абсолютные пути к файлам

Абсолютный путь к файлу - это полный URL-адрес к файлу на просторах Интернета:

Пример:

```
<img src="https://www.w3schools.com/images/picture.jpg" alt="Mountain">
```
Попробуйте сами »

Примечание: Тег `<img>` и атрибуты `src` и `alt` объясняются в разделе HTML изображения.

Относительные пути к файлам

Относительный путь к файлу указывает на файл относительно текущей страницы.

В этом примере путь к файлу указывает на файл в папке **images**, расположенную в корне текущего веб-сайта:

Пример:

```
<img src="/images/picture.jpg" alt="Mountain">
```
Попробуйте сами »

В этом примере путь к файлу указывается на файл в папке **images**, который находится в текущей папке:

Пример:

```
<img src="images/picture.jpg" alt="Mountain">
```
Попробуйте сами »

В этом примере путь к файлу указывается на файл в папке **images**, расположенную в папке на один уровень выше над текущей папкой:

Пример:

```
<img src="../images/picture.jpg" alt="Mountain">
```
Попробуйте сами »

Наилучшая практика

Лучше всего на практике использовать относительные пути к файлам (если возможно). При использовании относительных путей к файлам ваши веб-страницы не будут связаны с текущим базовым URL-адресом. Все ссылки будут работать на вашем собственном компьютере (на localhost), а также на вашем текущем публичном домене и ваших

будущих публичных доменах.

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

Prev Next

HTML Форматирование текста

Prev Next

Форматирование текста в HTML. Как сделать форматирование на веб-страницах?

Форматирование текста:

This text is bold

This text is italic

This is subscript and superscript

Попробуйте сами »

Элементы форматирования в HTML

В предыдущем уроке вы узнали об **HTML** атрибуте *style*. **HTML** также определяет специальные элементы для придания тексту **особого значения**. **HTML** использует такие элементы, как `<b>` и `<i>` для форматирования вывода, например **жирний** или *курсивный* текст. Элементы форматирования предназначены для отображения специальных типов текста.

- `<b>` - жирный текст
- `<strong>` - важный текст (семантически важный, выводится жирным)
- `<i>` - курсивный текст
- `<em>` - подчёркнутый текст (семантически важный, выводится курсивом)
- `<mark>` - помеченный (маркированный) текст
- `<small>` - уменьшенный текст
- `<del>` - удаленный текст
- `<ins>` - вставленный текст
- `<sub>` - текст подстрочный (нижний индекс)
- `<sup>` - текст надстрочный (верхний индекс)

HTML элементы <b> и

HTML элемент `<b>` определяет жирный текст без дополнительного значения.

Пример:

```
<b>This text is bold</b>
```

Попробуйте сами »

HTML элемент `<strong>` определяет важный текст с добавлением семантически «сильного» значения.

Пример:

```
<strong>This text is strong</strong>
```

Попробуйте сами »

HTML элементы <i> и

HTML элемент `<i>` определяет текст курсивом без дополнительного значения.

Пример:

```
<i>This text is italic</i>
```
Попробуйте сами »

HTML элемент `<em>` определяет подчёркнутый текст с дополнительным смысловым значением.

Пример:

```
<em>This text is emphasized</em>
```
Попробуйте сами »

Примечание: Браузеры отображают `<strong>` как `<b>` (выделяют жирным шрифтом), а `<em>` как `<i>` (выделяют курсивным шрифтом). Однако, существует разница в значении этих тегов: `<b>` и `<i>` лишь отображаются жирным и курсивным текстом, но теги `<strong>` и `<em>` означают, что текст "важный" по значению.

HTML элемент <small>

HTML элемент `<small>` определяет меньший текст (по сравнению с текстом по умолчанию):

Пример:

```
<h2>HTML <small>Small</small> Formatting</h2>
```
Попробуйте сами »

HTML элемент <mark>

HTML элемент `<mark>` определяет помеченный (маркированный) текст (по умолчанию - выделяется желтым цветом):

Пример:

```
<h2>HTML <mark>Marked</mark> Formatting</h2>
```
Попробуйте сами »

HTML элемент <del>

HTML элемент `<del>` определяет удалённый (изъятый) текст. По умолчанию отображается зачёркнутым.

Пример:

```
<p>My favorite color is <del>blue</del> red.</p>
```
Попробуйте сами »

HTML элемент <ins>

HTML элемент `<ins>` определяет вставленный (добавленный) текст. По умолчанию отображается подчёркнутым.

Пример:

```
<p>My favorite <ins>color</ins> is red.</p>
```
Попробуйте сами »

HTML элемент <sub>

HTML элемент `<sub>` определяет подстрочный текст (нижнего индекса).

Пример:

```
<p>This is <sub>subscripted</sub> text.</p>
```
Попробуйте сами »

HTML элемент <sup>

HTML элемент `<sup>` определяет надстрочный текст (верхнего индекса).

Пример:

```
<p>This is <sup>superscripted</sup> text.</p>
```

Попробуйте сами »

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 » Упражнение 5 »

HTML элементы форматирования текста

Тег	Описание
<b>	Определяет жирный текст
	Определяет важно подчёркнутый текст
<i>	Определяет курсивный текст
<small>	Определяет уменьшенный текст
	Определяет важный по значению текст
<sub>	Определяет подстрочный текст
<sup>	Определяет надстрочный текст
<ins>	Определяет вставленный текст
<del>	Определяет удаленный текст
<mark>	Определяет помеченный/маркированный текст

Справочник HTML тегов

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

Prev Next

HTML Формы

Prev Next

Что такое HTML форма?

HTML форма. Пример

First name:
Mickey
Last name:
Mouse

Submit

Попробуйте сами »

Элемент <form>

HTML элемент <form> определяет форму, которая используется для сбора данных пользователя:

```
<form>
.
form elements
.
</form>
```

HTML форма содержит **элементы формы**.

Элементы формы - это разные типы элементов ввода, такие как текстовые поля, флажки, переключатели, кнопки отправки и многое другое.

Элемент <input>

Элемент `<input>` является самым важным элементом формы. Элемент `<input>` может отображаться несколькими способами, в зависимости от **типа** атрибута.

Вот несколько примеров.

Тип	Описание
<input type="text">	Определяет одно строчное поле для ввода текста
<input type="radio">	Определяет radio-переключатель (для выбора одного из множества вариантов)
<input type="submit">	Определяет кнопку отправки (для отправки формы)

Более подробно о типах ввода вы узнаете далее в этом учебнике.

Ввод текста - text input

`<input type="text">` определяет одно строчное поле input для **ввода текста**:

Пример:

```
<form>
  First name:<br>
  <input type="text" name="firstname"><br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```

Попробуйте сами »

Так будет выглядеть в браузере:

First name:

Last name:

Примечание: Саму форму не видно. Также обратите внимание, что типовая (по умолчанию) ширина текстового поля составляет 20 символов.

Кнопка radio для ввода

`<input type="radio">` определяет **radio-кнопку**.

Radio-кнопки позволяют пользователю выбрать один с ограниченного числа вариантов:

Пример:

```
<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```

Попробуйте сами »

Приведённый выше **HTML**-код так будет отображаться в веб-браузере:

- Male
- Female
- Other

Кнопка Submit (Отправить)

`<input type="submit">` определяет кнопку для **отправки** данных формы к **обработчику форм**. Обработчик форм обычно является страницей сервера из сценарием для обработки входных данных. Обработчик форм (обычно скрипт на языке PHP) указан в атрибуте `action` формы:

Пример:

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```

Попробуйте сами »

Приведённый выше **HTML**-код так будет отображаться в веб-браузере:

First name:
Mickey
Last name:
Mouse

Submit

Атрибут action (действия)

Атрибут `action` определяет действие, которое следует выполнить после отправки формы. Обычно данные формы отправляются на веб-страницу на сервере, когда пользователь нажимает на кнопку "Submit" ("Отправить").

В приведённом выше примере данные формы отправляются на страницу на сервере "/action_page.php". Эта страница содержит серверный сценарий (скрипт на языке PHP), который обрабатывает данные формы:

Пример:

```
<form action="/action_page.php">
```

Попробуйте сами »

Если атрибут `action` опущен, действие выполняется на текущей странице.

Атрибут target (цель)

Атрибут `target` определяет, будет ли открыт отправленный результат в новом окне, на вкладке веб-браузера, фрейме или в текущем окне. Значение по умолчанию - "`_self`", что означает, что форма будет открыта в текущем окне. Чтобы открыть результат в новой вкладке веб-браузера, воспользуйтесь значением "`_blank`":

Пример:

```
<form action="/action_page.php" target="_blank">
```

Попробуйте сами »

Другими действительными значениями является "`_parent`", "`_top`" или название, которое представляет название фрейма.

Атрибут method (метод)

Атрибут `method` указывает метод HTTP (**GET** или **POST**), который будет использоваться во время отправки данных формы:

Пример:

```
<form action="/action_page.php" method="get">
```

Попробуйте сами »

или:

Пример:

```
<form action="/action_page.php" method="post">
```

Попробуйте сами »

Когда использовать GET?

Метод по умолчанию при передаче данных формы **GET**. Однако, когда **GET** используется, данные, отосланные формой, будут **видимы в поле адреса страницы**:

Пример:

/action_page.php?firstname=Mickey&lastname=Mouse

Примечания к GET:

- Добавляет данные в форме отправки к URL-адресу в парах имя/значение
- Длина URL-адреса ограничена (около 3000 символов)
- Никогда не используйте GET для отправки конфиденциальных данных! (будет отображаться в URL-адресе)
- Полезно для подачи форм, где пользователь хочет сделать закладку результата
- GET является лучшим вариантом для незащищённых данных, например строк запросов в Google

---

Когда использовать POST?

Всегда используйте **POST**, если данные формы содержат конфиденциальную или личную информацию. Метод **POST** не отображает отправленные данные формы в адресной строке страницы.

Примечание к POST:

- POST не имеет ограничений по размерам и может использоваться для отправки больших объёмов данных
- Отправленные с POST формы нельзя добавить в закладки

Атрибут name

Для каждого поля ввода должен быть подан атрибут `name` (название). Если атрибут `name` пропущен, данные этого поля ввода вообще не будут отправляться.

В этом примере будет отправлено лишь поле ввода "Last name" ("Фамилия"):

Пример:

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```

Попробуйте сами »

Группировка данных форм с помощью <fieldset>

Элемент `<fieldset>` используется для группировки связанных данных в форме. Элемент `<legend>` определяет заголовок для элемента `<fieldset>`.

Пример:

```
<form action="/action_page.php">
  <fieldset>
    <legend>Personal information:</legend>
    First name:<br>
    <input type="text" name="firstname" value="Mickey"><br>
    Last name:<br>
    <input type="text" name="lastname" value="Mouse"><br><br>
    <input type="submit" value="Submit">
```

```
  </fieldset>
</form>
```
Попробуйте сами »

Так этот HTML код будет выглядеть в браузере:

Personal information:

First name:
Mickey

Last name:
Mouse

Submit

Проверьте себя с помощью упражнений

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 »

Список всех атрибутов `<form>`:

Атрибут	Описание
accept-charset	Указывает набор символов, что используется в отправленной форме (по умолчанию: набор символов страницы).
action	Указывает адрес (url), куда отправить форму (по умолчанию: страница отправки).
autocomplete	Указывает, должен ли браузер автоматически заполнять форму (по умолчанию: включено).
enctype	Определяет кодировку отправленных данных (по умолчанию: url-encoded).
method	Указывает метод HTTP, который используется во время отправки формы (по умолчанию: GET).
name	Указывает название, которое используется для идентификации формы (для использования DOM: document.forms.name).
novalidate	Указывает, что браузер не должен проверять форму.
target	Указывает цель адреса в атрибуте action (по умолчанию:_self).

Примечание: Вы узнаете больше про атрибуты форм в следующих разделах.

Prev Next

HTML Input Attributes. Атрибуты ввода

Prev Next

Атрибут value (значения)

Атрибут `value` определяет начальное значениие для поля ввода:

Пример:

```
<form action="">
  First name:<br>
  <input type="text" name="firstname" value="John">
</form>
```
Попробуйте сами »

Атрибут readonly (только для чтения)

Атрибут `readonly` указывает, что поле ввода только для чтения (не может быть изменено):

Пример:

```
<form action="">
  First name:<br>
  <input type="text" name="firstname" value="John" readonly>
</form>
```
Попробуйте сами »

Атрибут disabled (выключено)

Атрибут `disabled` указывает, что поле ввода выключено.

Выключенное поле ввода нельзя использовать и невозможно кликнуть, а его значение не отправляется во время отправки формы:

Пример:

```
<form action="">
  First name:<br>
  <input type="text" name="firstname" value="John" disabled>
</form>
```
Попробуйте сами »

Атрибут size (размер)

Атрибут `size` определяет размер (в символах) поля ввода:

Пример:

```
<form action="">
  First name:<br>
  <input type="text" name="firstname" value="John" size="40">
</form>
```
Попробуйте сами »

Атрибут maxlength (максимальная длина)

Атрибут `maxlength` определяет максимально допустимую длину поля ввода:

Пример:

```
<form action="">
  First name:<br>
  <input type="text" name="firstname" maxlength="10">
</form>
```
Попробуйте сами »

С помощью атрибута `maxlength` поле ввода не будет принимать больше допустимого числа символов.

Атрибут `maxlength` не предоставляет никакого отклика. Если вы хотите отправить сообщение пользователю, вы должны написать JavaScript код.

Примечание: Ограничение на входящие данные не является надёжным, а JavaScript предоставляет множество способов добавить незаконный вход. Чтобы безопасно ограничить ввод, он также должен быть проверен принимающей стороной (сервером)!

HTML5 Атрибуты

HTML5 добавил следующие атрибуты для `<input>`:

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate

- formtarget
- height and width
- list
- min and max
- multiple
- pattern (regexp)
- placeholder
- required
- step

и следующие атрибуты для `<form>`:

- autocomplete
- novalidate

Атрибут autocomplete (автозаполнение)

Атрибут `autocomplete` указывает, должна ли форма или поле использовать автозаповнение.

Когда автозаполнение включено, браузер автоматически завершает вводимые значения на основе значений, введенных пользователем ранее.

Совет: Можно иметь автозаполнение "включено" для формы и "выключено" для определённых полей ввода данных, или наоборот.

Атрибут `autocomplete` работает с `<form>` и следующими `<input>` типами: text, search, url, tel, email, password, datepickers, range и color.

Пример:

Форма HTML с автозаполнением (и без автозаполнения для одного поля ввода):

```
<form action="/action_page.php" autocomplete="on">
  First name:<input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  E-mail: <input type="email" name="email" autocomplete="off"><br>
  <input type="submit">
</form>
```

Попробуйте сами »

Совет: В некоторых браузерах вам может понадобиться активировать функцию автозаполнения для этого.

Атрибут novalidate (не проверяется)

Атрибут `novalidate` является атрибутом `<form>`.

При наличии `novalidate` указывается, что данные формы не должны проверяться, когда они отправляются.

Пример:

Указывается, что форма не проверяется при отправке:

```
<form action="/action_page.php" novalidate>
  E-mail: <input type="email" name="user_email">
  <input type="submit">
</form>
```

Попробуйте сами »

Атрибут autofocus (автофокус)

Атрибут `autofocus` указывает, что поле ввода должно автоматически получать фокус при загрузке страницы.

Пример:

Пусть поле ввода "First name" автоматически получает фокус, когда страница загружается:

```
First name:<input type="text" name="fname" autofocus>
```
Попробуйте сами »

Атрибут form (форма)

Атрибут `form` определяет одну или больше форм, к которым принадлежит элемент `<input>`

Совет: Чтобы ссылаться более чем на одну форму, используйте разделённый пробелами список идентификаторов форм.

Пример:

Поле ввода, расположенное вне формы HTML (но которое всё ещё является частью формы):

```
<form action="/action_page.php" id="form1">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
</form>

Last name: <input type="text" name="lname" form="form1">
```
Попробуйте сами »

Атрибут formaction (формирование)

Атрибут `formaction` определяет URL-адрес файла, который будет обрабатывать элемент управления вводом, когда форма будет отправлена.

Атрибут formaction переопределяет атрибут элемента `<form>`.

Атрибут formaction используется с `type="submit"` та `type="image"`.

Пример:

HTML форма с двумя кнопками отправки, с разными действиями:

```
<form action="/action_page.php">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formaction="/action_page2.php"
  value="Submit as admin">
</form>
```
Попробуйте сами »

Атрибут formenctype (тип формата)

Атрибут `formenctype` указывает, как будут кодироваться данные формы после отправки (лишь для форм с method="post").

Атрибут `formenctype` переопределяет атрибут `enctype` элемента `<form>`.

Атрибут `formenctype` используется с `type="submit"` и `type="image"`.

Пример:

Отправляются данные формы, кодированные по умолчанию (первая кнопка submit), и кодированные как "multipart/form-data" (вторая кнопка submit):

```
<form action="/action_page_binary.asp" method="post">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
  value="Submit as Multipart/form-data">
</form>
```
Попробуйте сами »

Атрибут formmethod (метод формы)

Атрибут `formmethod` определяет метод HTTP для отправки данных формы действующему URL.

Атрибут `formmethod` переопределяет атрибут метода элемента `<form>`.

Атрибут `formmethod` можно использовать с `type="submit"` и `type="image"`.

Пример:

Вторая кнопка submit заменяет метод HTTP формы::

```
<form action="/action_page.php" method="get">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formmethod="post" value="Submit using POST">
</form>
```

Попробуйте сами »

Атрибут formnovalidate (форма без проверки)

Атрибут `formnovalidate` заменяет атрибут novalidate элемента `<form>`.

Атрибут `formnovalidate` можно использовать с `type="submit"`.

Пример:

Форма с двумя кнопками submit (с проверкой и без):

```
<form action="/action_page.php">
  E-mail: <input type="email" name="userid"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formnovalidate value="Submit without validation">
</form>
```

Попробуйте сами »

Атрибут formtarget (цель формы)

Атрибут `formtarget` указывает название или ключевое слово, которое указывает, где будет отображаться ответ, полученный после отправки формы.

Атрибут `formtarget` переопределяет целевой атрибут элемента `<form>`.

Атрибут `formtarget` можно использовать с `type="submit"` и `type="image"`.

Пример:

Форма с двумя кнопками submit, с разными целевыми окнами:

```
<form action="/action_page.php">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit as normal">
  <input type="submit" formtarget="_blank"
  value="Submit to a new window">
</form>
```

Попробуйте сами »

Атрибуты height и width (высота и ширина)

Атрибуты `height` и `width` определяют высоту и ширину элемента `<input type="image">`.

Всегда указывайте размер изображений. Если веб-браузер не знает размера, страница будет мерцать во время загрузки изображений.

Пример:

Определить изображение как кнопку submit, с атрибутами высоты и ширины:

```
<input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
```

Попробуйте сами »

Атрибут list (список)

Атрибут `list` ссылается на элемент `<datalist>` который содержит предварительно определённые параметры для элемента `<input>`.

Пример:

Элемент `<input>` с предварительно определёнными значениями в `<datalist>`:

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

Попробуйте сами »

Атрибуты min и max (минимум и максимум)

Атрибуты `min` и `max` определяют минимальное и максимальное значение для элемента `<input>`.

Атрибуты `min` и `max` работают с такими типами input: number, range, date, datetime-local, month, time и week.

Пример:

<input> элементы с min и max значениями:

```
Enter a date before 1980-01-01:
<input type="date" name="bday" max="1979-12-31">

Enter a date after 2000-01-01:
<input type="date" name="bday" min="2000-01-02">

Quantity (between 1 and 5):
<input type="number" name="quantity" min="1" max="5">
```

Попробуйте сами »

Атрибут multiple (несколько значений)

Атрибут `multiple` указывает, что пользователю разрешено вводить более чем одно значение в элемент `<input>`.

Атрибут `multiple` работает с такими типами input: email и file.

Пример:

Поле загрузки файла, которое принимает несколько значений:

```
Select images: <input type="file" name="img" multiple>
```

Попробуйте сами »

Атрибут pattern (шаблон)

Атрибут `pattern` определяет регулярное выражение, в котором проверяется значение элемента `<input>`.

Атрибут `pattern` работает с такими типами input: text, search, url, tel, email и password.

Совет: Используйте глобальный атрибут title для описания шаблона, чтобы помочь пользователю.

Совет: Узнать больше про регулярные выражения можно в учебнике по JavaScript.

Пример:

Поле ввода, которое может содержать лишь три буквы (без номеров или специальных символов):

```
Country code: <input type="text" name="country_code" pattern="[A-Za-z]{3}" title="Three letter country code">
```
Попробуйте сами »

Атрибут placeholder (заполнитель)

Атрибут `placeholder` определяет подсказку, которая описывает ожидаемое значение поля ввода (значение выборки или короткое описание формата).

Подсказка отображается в поле ввода, прежде чем пользователь введёт значение.

Атрибут `placeholder` работает с такими типами input: text, search, url, tel, email и password.

Пример:

Поле ввода с текстом placeholder:

```
<input type="text" name="fname" placeholder="First name">
```
Попробуйте сами »

Атрибут required (требование)

Атрибут `required` указывает, что поле ввода необходимо заполнить перед подачей формы.

Атрибут `required` работает с такими типами input: text, search, url, tel, email, password, date pickers, number, checkbox, radio и file.

Пример:

Поле ввода required:

```
Username: <input type="text" name="usrname" required>
```
Попробуйте сами »

Атрибут step (шаг)

Атрибут `step` указывает интервалы действительного числа для элемента `<input>`.

Пример: если step="3", действительными числами могут быть -3, 0, 3, 6 и т.д.

Совет: Атрибут step можно использовать вместе с атрибутами max и min, чтобы создать диапазон действительных значений.

Атрибут `step` работает с такими типами input: number, range, date, datetime-local, month, time и week.

Пример:

Поле ввода с точно определёнными промежутками интервалов:

```
<input type="number" name="points" step="3">
```
Попробуйте сами »

Проверьте себя с помощью упражнений

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 »

HTML формы и элементы ввода

Тег	Описание
<form>	Определяет HTML форму для ввода пользователем
<input>	Определяет элемент управления вводом

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

Prev Next

HTML Элементы формы

Prev Next

Элементы форм на веб-страницах

Этот раздел описывает все **элементы форм в HTML**.

Элемент <input>

Самым важным элементом формы является элемент <input> (т.е. ввод). Элемент <input> может отображаться несколькими способами, в зависимости от атрибута type.

Пример:

```
<input name="firstname" type="text">
```
Попробуйте сами »

Примечание: Если атрибут type опущен, поле ввода получает тип по умолчанию: "text".

Все разные типы входных данных рассматриваются в следующем разделе.

Элемент <select>

Элемент <select> определяет выпадающий список:

Пример:

```
<select name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```
Попробуйте сами »

Элементы <option> определяют опцию, которую можно выбрать. По умолчанию выбирается первый элемент у выпадающем списке. Чтобы определить предварительно выбранный параметр, добавьте атрибут selected к параметру:

Пример:

```
<option value="fiat" selected>Fiat</option>
```
Попробуйте сами »

Видимые значения:

Используйте атрибут size, чтобы указать количество видимых значений:

Пример:

```
<select name="cars" size="3">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

Попробуйте сами »

Позволить выбор нескольких позиций:

Используйте атрибут `multiple`, чтобы позволить пользователю выбрать более одного значения:

Пример:

```
<select name="cars" size="4" multiple>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

Попробуйте сами »

Элемент <textarea>

Элемент `<textarea>` определяет многострочное поле ввода (**текстовая область**):

Пример:

```
<textarea name="message" rows="10" cols="30">
The cat was playing in the garden.
</textarea>
```

Попробуйте сами »

Атрибут `rows` определяет видимое число строк в текстовой области. Атрибут `cols` определяет видимую ширину текстовой области.

Таким образом HTML-код будет отображаться в браузере:

The cat was playing in the garden.

Вы также можете определить размер области текста с помощью **CSS**:

Пример:

```
<textarea name="message" style="width:200px; height:600px;">
The cat was playing in the garden.
</textarea>
```

Попробуйте сами »

Элемент <button> (кнопка)

Элемент `<button>` определяет кликабельную **кнопку** (которую можно нажать):

Пример:

```
<button type="button" onclick="alert('Hello World!')">Click Me!</button>
```

Попробуйте сами »

Таким образом данный **HTML**-код будет отображаться в браузере:

Click Me!

Примечание: Всегда указывайте атрибут `type` элемента кнопки. Разные браузеры могут использовать разные `type` по умолчанию для элемента кнопки.

Элементы формы в HTML5

HTML5 добавил такие элементы формы:

- `<datalist>`
- `<output>`

Примечание: Браузеры не показывают неизвестные элементы. Новые элементы, которые не поддерживаются в старых браузерах, не "портят" вашу веб-страницу.

HTML5 элемент <datalist>

Элемент `<datalist>` указывает список предварительно определённых параметров для элемента `<input>`. Пользователи видят раскрывающий список предварительно определённых параметров во время ввода данных. Атрибут `list` элемента `<input>` должен ссылаться на атрибут `id` элемента `<datalist>`.

Пример:

```
<form action="/action_page.php">
  <input list="browsers">
  <datalist id="browsers">
    <option value="Internet Explorer">
   <option value="Firefox">
    <option value="Chrome">
   <option value="Opera">
    <option value="Safari">
  </datalist>
</form>
```

Попробуйте сами »

HTML5 элемент <output>

Элемент `<output>` представляет собой результат вычисления (подобно тому, как выполняется скрипт).

Пример:

Выполнить расчёт и показать результат в элементе `<output>`:

```
<form action="/action_page.php"
  oninput="x.value=parseInt(a.value)+parseInt(b.value)">
  0
  <input type="range"  id="a" name="a" value="50">
  100 +
  <input type="number" id="b" name="b" value="50">
  =
  <output name="x" for="a b"></output>
  <br><br>
  <input type="submit">
</form>
```

Попробуйте сами »

Проверьте себя с помощью упражнений

Упражнение 1 » Упражнение 2 » Упражнение 3 »

HTML элементы формы

= новые в HTML5.

Тег	Описание
<form>	Определяет **HTML** форму для ввода пользователем
<input>	Определяет элемент управления вводом
<textarea>	Определяет многострочный элемент управления вводом
<label>	Определяет метку для элемента <input>

<fieldset>	Группы связанных элементов в форме
<legend>	Определяет заголовок для элемента <fieldset>
<select>	Определяет выпадающий (раскрывающийся) список
<optgroup>	Определяет группу соответствующих параметров в раскрывающемся списке
<option>	Определяет параметр в раскрывающемся списке
<button>	Определяет кликабельную кнопку (которую можно нажимать)
<datalist>	Определяет список предварительно определённых параметров для элементов управления вводом
<output>	Определяет результат расчёта

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

Prev Next

HTML Типы ввода. Input type

Prev Next

HTML Типы ввода

Этот раздел описывает разные **типы ввода** для элемента `<input>`.

HTML Input Type

Это разные типы ввода, которые можно использовать в **HTML**:

- `<input type="button">`
- `<input type="checkbox">`
- `<input type="color">`
- `<input type="date">`
- `<input type="datetime-local">`
- `<input type="email">`
- `<input type="file">`
- `<input type="hidden">`
- `<input type="image">`
- `<input type="month">`
- `<input type="password">`
- `<input type="radio">`
- `<input type="range">`
- `<input type="reset">`
- `<input type="search">`
- `<input type="submit">`
- `<input type="tel">`
- `<input type="text">`
- `<input type="time">`
- `<input type="url">`
- `<input type="week">`

Тип ввода Text (текст)

`<input type="text">` определяет однострочное поле ввода текста:

Пример:

```
<form>
  First name:<br>
  <input type="text" name="firstname"><br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```

Попробуйте сами »

Так HTML-код будет отображаться в браузере:

First name:
Last name:

Тип ввода Password (пароль)

`<input type="password">` определяет **поле ввода пароля**:

Пример:

```
<form>
  User name:<br>
  <input type="text" name="username"><br>
  User password:<br>
  <input type="password" name="psw">
</form>
```

Попробуйте сами »

Так HTML-код будет отображаться в браузере:

User name:
User password:

Символы в поле пароля маскируются (показано как звёздочки или черные кружочки).

Тип ввода Submit (Отправить)

`<input type="submit">` определяет кнопку для **отправки** данных формы к **обработчику форм**.

Обработчик форм обычно является страницей сервера из сценарием (скриптом) для обработки входных данных.
Обработчик форм указан в атрибуте `action` формы:

Пример:

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```

Попробуйте сами »

Так HTML-код будет отображаться в браузере:

First name:
Mickey
Last name:
Mouse

Submit

Если вы не укажете атрибут значения кнопки "Отправить", кнопка получит текст по умолчанию:

Пример:

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
```

```
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit">
</form>
```
Попробуйте сами »

Тип ввода Reset (Сброс)

`<input type="reset">` определяет кнопку сброса, которая сбрасывает все значения формы к значениям по умолчанию.

Пример:

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
  <input type="reset">
</form>
```
Попробуйте сами »

Так HTML-код будет отображаться в браузере:

First name:
Mickey
Last name:
Mouse

Submit Reset

Если изменить входные значения, а потом нажать кнопку "Submit" ("Отправить"), данные формы будут сброшены до значений по умолчанию.

Тип ввода Radio

`<input type="radio">` определяет переключатель (**кнопку radio**).

Кнопки radio позволяют пользователю выбрать ТОЛЬКО ОДИН с ограниченного числа вариантов:

Пример:

```
<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```
Попробуйте сами »

Так HTML-код будет отображаться в браузере:

- (•) Male
- ( ) Female
- ( ) Other

Тип ввода Checkbox (флажок)

`<input type="checkbox">` определяет флажок.

Флажки позволяют пользователю выбрать опции ZERO или MORE для ограниченного выбора.

Пример:

```
<form>
  <input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>
  <input type="checkbox" name="vehicle2" value="Car"> I have a car
```

```
</form>
```
Попробуйте сами »

Так HTML-код будет отображаться в браузере:

I have a bike
I have a car

Тип ввода Button (кнопка)

`<input type="button">` определяет **кнопку**:

Пример:

```
<input type="button" onclick="alert('Hello World!')" value="Click Me!">
```
Попробуйте сами »

Так HTML-код будет отображаться в браузере:

Click Me!

HTML5 Input Types (типы ввода)

HTML5 добавил несколько новых типов ввода:

- color
- date
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

Новые типы ввода данных, которые не поддерживаются более старыми веб-браузерами, будут вести себя как `<input type="text">`.

Тип ввода Color (цвет)

`<input type="color">` используется для полей ввода, которые должны содержать цвет.

В зависимости от поддержки веб-браузера в поле ввода может отображаться выбор цветов.

Пример:

```
<form>
  Select your favorite color:
  <input type="color" name="favcolor">
</form>
```
Попробуйте сами »

Тип ввода Date (дата)

`<input type="date">` используется для полей ввода, которые должны содержать дату.

В зависимости от поддержки веб-браузера в поле ввода может отображаться выбор даты.

Пример:

```
<form>
  Birthday:
```

```
  <input type="date" name="bday">
</form>
```
Попробуйте сами »

Можно также использовать атрибуты `min` и `max` для добавления ограничений для дат:

Пример:

```
<form>
  Enter a date before 1980-01-01:
  <input type="date" name="bday" max="1979-12-31"><br>
  Enter a date after 2000-01-01:
  <input type="date" name="bday" min="2000-01-02"><br>
</form>
```
Попробуйте сами »

Тип ввода Datetime-local

`<input type="datetime-local">` указывает поле ввода даты и времени, в котором нет временного пояса.

В зависимости от поддержки веб-браузера в поле ввода может отображаться выбор даты.

Пример:

```
<form>
  Birthday (date and time):
  <input type="datetime-local" name="bdaytime">
</form>
```
Попробуйте сами »

Тип ввода email (электронная почта)

`<input type="email">` используется для полей ввода, которые должны содержать адрес электронной почты.

В зависимости от поддержки браузера, адрес электронной почты может быть автоматически проверен во время отправки.

Некоторые смартфоны распознают тип электронной почты и добавляют ".com" на клавиатуру, чтобы отвечать вводу электронной почты.

Пример:

```
<form>
  E-mail:
  <input type="email" name="email">
</form>
```
Попробуйте сами »

Тип ввода File (файл)

`<input type="file">` определяет поле выбора файла и кнопку "Просмотр" для загрузки файлов.

Пример:

```
<form>
  Select a file: <input type="file" name="myFile">
</form>
```
Попробуйте сами »

Тип ввода Month (месяц)

`<input type="month">` позволяет пользователю выбрать месяц и год.

В зависимости от поддержки веб-браузера в поле ввода может отображаться выбор даты.

Пример:

```
<form>
  Birthday (month and year):
  <input type="month" name="bdaymonth">
</form>
```
Попробуйте сами »

Тип ввода Number (номер)

`<input type="number">` определяет **числовое поле** ввода.

Вы также можете установить ограничение того, какие числа принимаются.

В следующем примере отображается числовое поле ввода, в котором можно ввести значения от 1 до 5:

Пример:

```
<form>
  Quantity (between 1 and 5):
  <input type="number" name="quantity" min="1" max="5">
</form>
```
Попробуйте сами »

Ограничения ввода

Это список некоторых общих ограничений ввода (некоторые из них новые в **HTML5**):

Атрибут	Описание
checked	Указывает, что поле ввода должно быть предварительно выбрано при загрузке страницы (для type="checkbox" или type="radio")
disabled	Определяет, что поле ввода должно быть выключено (запрещено)
max	Определяет максимальное значение для поля ввода
maxlength	Определяет максимальное количество символов для поля ввода
min	Определяет минимальное значение для поля ввода
pattern	Определяет регулярное выражение для проверки входящего значения
readonly	Определяет, что поле ввода лишь для чтения (не может быть изменено)
required	Определяет, что поле ввода обязательно (должно быть заполнено)
size	Указывает ширину (в символах) поля ввода
step	Определяет допустимые интервалы номеров для поля ввода
value	Определяет значения по умолчанию для поля ввода

Вы узнаете больше про ограничения ввода в следующем разделе.

В следующем примере отображается числовое поле ввода, в котором можно ввести значения от 0 до 100, шагами 10. Значение по умолчанию - 30:

Пример:

```
<form>
  Quantity:
  <input type="number" name="points" min="0" max="100" step="10" value="30">
</form>
```
Попробуйте сами »

Тип ввода Range (диапазон)

`<input type="range">` определяет элемент управления для ввода числа, точное значение которого не является важным (например, ползунок). Стандартный диапазон от 0 до 100. Но вы можете установить ограничение того, какие числа принимаются с атрибутами `min`, `max` и `step`:

Пример:

```
<form>
  <input type="range" name="points" min="0" max="10">
</form>
```
Попробуйте сами »

Тип ввода Search (поиск)

`<input type="search">` используется для поисковых полей (поле поиска ведёт себя как обычное текстовое поле).

Пример:

```
<form>
  Search Google:
  <input type="search" name="googlesearch">
</form>
```

Попробуйте сами »

Тип ввода Tel (телефон)

`<input type="tel">` используется для полей ввода, которые должны содержать номер телефона.

Пример

```
<form>
  Telephone:
  <input type="tel" name="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```

Попробуйте сами »

Тип ввода Time (время)

`<input type="time">` позволяет пользователю выбрать время (без часового пояса).

В зависимости от поддержки веб-браузера в поле ввода может отображаться выбор времени.

Пример:

```
<form>
  Select a time:
  <input type="time" name="usr_time">
</form>
```

Попробуйте сами »

Тип ввода Url (веб-адрес)

`<input type="url">` используется для полей ввода, которые должны содержать URL-адрес.

В зависимости от поддержки веб-браузера, поле URL может быть автоматически проверено во время отправки. Некоторые смартфоны распознают тип URL-адреса и добавляют ".com" к набору с клавиатуры, чтобы соответствовать вводу URL.

Пример:

```
<form>
  Add your homepage:
  <input type="url" name="homepage">
</form>
```

Попробуйте сами »

Тип ввода Week (неделя)

`<input type="week">` позволяет пользователю выбрать неделю и год.

В зависимости от поддержки веб-браузера в поле ввода может отображаться выбор даты.

Пример:

```
<form>
  Select a week:
  <input type="week" name="week_year">
```

```
</form>
```
Попробуйте сами »

Проверьте себя с помощью упражнений

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 » Упражнение 5 »

HTML Атрибут Input Type

Тег	Описание
<input type="">	Определяет тип ввода для отображения

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

Prev Next

HTML Head. Голова веб-страницы

Prev Next

HTML элемент <head>

Элемент `<head>` является контейнером для метаданных (служебных данных) и располагается между тегом `<html>` и тегом `<body>`.

Метаданные HTML - это служебные данные про **HTML** документ. Метаданные не отображаются на веб-странице и их не видно пользователям. Метаданные обычно определяют название документа, набор символов (кодировку), стили, ссылки, скрипты и другую мета-информацию.

Следующие теги описывают метаданные: `<title>`, `<style>`, `<meta>`, `<link>`, `<script>` и `<base>`.

HTML элемент <title>

Элемент `<title>` определяет название документа и является обязательным для всех HTML/XHTML документов согласно спецификации **HTML5**.

Элемент `<title>`:

- определяет заголовок на вкладке веб-браузера
- предоставляет название страницы, когда она добавлена к избранному
- отображает название страницы в результатах поиска

Простой HTML документ:

Пример:

```
<!DOCTYPE html>
<html>

<head>
  <title>Page Title</title>
</head>

<body>
The content of the document.....
</body>

</html>
```
Попробуйте сами »

HTML элемент <style>

Элемент `<style>` используется для определения информации про стиль для одной **HTML** страницы:

Пример:

```
<style>
  body {background-color: powderblue;}
  h1 {color: red;}
  p {color: blue;}
</style>
```

Попробуйте сами »

HTML элемент <link>

HTML элемент `<link>` используется для ссылки на внешние таблицы стилей - **CSS** файлы:

Пример:

```
<link rel="stylesheet" href="mystyle.css">
```

Попробуйте сами »

Примечание: Чтобы узнать больше о **CSS**, посетите Учебник по **CSS**.

HTML элемент <meta>

Элемент `<meta>` используется для определения того, какой набор символов используется (кодировка), описания страницы, ключевые слова, автора и другие метаданные. Метаданные используются браузерами (как отображать содержание), поисковыми системами (ключевые слова) и другими веб-службами.

Определите набор символов (кодировку), что используется на веб-странице:

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

Определите описание веб-страницы:

```
<meta name="description" content="Free Web tutorials">
```

Определите ключевые слова для поисковых систем:

```
<meta name="keywords" content="HTML, CSS, XML, JavaScript">
```

Определите автора страницы:

```
<meta name="author" content="John Doe">
```

Обновить документ каждые 30 секунд:

```
<meta http-equiv="refresh" content="30">
```

Пример `<meta>` тегов :

Пример:

```
<meta charset="UTF-8">
<meta name="description" content="Free Web tutorials">
<meta name="keywords" content="HTML,CSS,XML,JavaScript">
<meta name="author" content="John Doe">
```

Попробуйте сами »

Установка окна просмотра - Viewport

HTML5 представил метод, который позволяет веб-дизайнерам управлять экраном через тег `<meta>`. Окно просмотра (viewport) - видимая область пользователя веб-страницы. Она меняется в зависимости от устройства и будет меньше на мобильном телефоне, чем на экране компьютера.

Для этого необходимо включить следующий элемент `<meta> viewport` на всех веб-страницах:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Элемент `<meta> viewport` предоставляет инструкциям браузера, как контролировать размеры и масштабирование страницы. Часть `width=device-width` устанавливает ширину страницы, чтобы она отвечала ширине экрана устройства (которая зависит от устройства). Часть `initial-scale=1.0` устанавливает начальный уровень масштабирования, когда страница загружается браузером.

Ниже приведён пример веб-страницы без метатега `viewport` и той же веб-страницы с метатегом `viewport`:

Примечание: Если вы просматриваете эту страницу с помощью телефона или планшета, можно нажать две ссылки ниже, чтобы увидеть разницу (в другом окне).

Без viewport метатега

Из viewport метатегом

HTML элемент <script>

Элемент `<script>` используется для определения **JavaScript** на стороне клиента.

Этот **JavaScript** пишет "Hello JavaScript!" в **HTML** элементе с id = "demo":

Пример:

```
<script>
function myFunction {
  document.getElementById("demo").innerHTML = "Hello JavaScript!";
}
</script>
```

Попробуйте сами »

Примечание: Чтобы узнать больше про **JavaScript**, перейдите к Учебнику по JavaScript.

HTML элемент <base>

Элемент `<base>` указывает базовый URL-адрес и базовую цель для всех относительных URL-адресов страницы:

Пример:

```
<base href="https://www.w3schools.com/images/" target="_blank">
```

Попробуйте сами »

Опускать ли <html>, <head> и <body>?

В соответствие стандарту **HTML5** теги `<html>`, `<body>` и `<head>` можно опустить.

Следующий код пройдёт валидацию как **HTML5**:

Пример:

```
<!DOCTYPE html>
<title>Page Title</title>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

Попробуйте сами »

Примечание: W3Schools **не рекомендует** опускать теги `<html>` и `<body>`. Пренебрежение этими тегами может привести к сбою программного обеспечения DOM или XML и возникновению ошибок в старых браузерах (IE9). Впрочем, опускать тег `<head>` - это была широкая практика в течение довольно длительного времени.

HTML элементы головы веб-страницы - Head

Тег	Описание
`<head>`	Определяет информацию про документ

<title>	Определяет название документа
<base>	Определяет адрес по умолчанию или типичную цель для всех ссылок на странице
<link>	Определяет взаимосвязь между документом и внешним ресурсом
<meta>	Определяет метаданные (служебные данные) про документ HTML
<script>	Определяет скрипт на стороне клиента
<style>	Определяет информацию про стиль для документа

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

□ Prev Next □

HTML Заголовки

□ Prev Next □

HTML заголовки. Как создавать заголовки на веб-страницах?

Заголовки

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Попробуйте сами »

Заголовки HTML

Заголовки определяются тегами <h1> до <h6>.

<h1> определяет самый важный заголовок, <h6> определяет наименее важный заголовок.

Пример:

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Попробуйте сами »

Примечание: Браузеры автоматически добавляют отступ (margin) до и после заголовка.

Заголовки важны

Поисковые системы (например, *Google*, *Bing*, *Baidu*, *Yahoo!*, *Yandex*, *DuckDuckGo*, *Meta* и др.) используют заголовки для индексирования структуры и содержания веб-страниц. Пользователи просматривают веб-страницы по заголовкам. Важно использовать заголовки, чтобы показать структуру документа.

<h1> заголовки следует использовать для основных заголовков, а потом заголовки <h2>, потом менее важные <h3> и так

далее.

Примечание: Используйте **HTML** заголовки только для заголовков. Не используйте заголовки, чтобы сделать текст просто большим или полужирным. Для стилизации элементов на веб-странице используйте лишь **CSS**.

Размеры заголовков

Каждый заголовок **HTML** имеет размер по умолчанию. Тем не менее, вы можете указать размер для любого заголовка с атрибутом `style`, используя свойство `font-size` в **CSS**:

Пример:

```
<h1 style="font-size:60px;">Heading 1</h1>
```

Попробуйте сами »

Горизонтальная разделительная линия <hr> в HTML

Тег `<hr>` определяет тематический перерыв в HTML-странице и обычно отображается как горизонтальная линия. Элемент `<hr>` используется для разделения содержания (или определения изменения) на HTML-странице:

Пример:

```
<h1>This is heading 1</h1>
<p>This is some text.</p>
<hr>
<h2>This is heading 2</h2>
<p>This is some other text.</p>
<hr>
```

Попробуйте сами »

Элемент <head> в HTML

Элемент **HTML** `<head>` (*head* - голова) не имеет ничего общего с заголовками **HTML**. Элемент `<head>` является контейнером для метаданных. Метаданные **HTML** - это данные о документе **HTML**. Метаданные не отображаются на веб-странице и необходимы лишь для служебных целей. Элемент `<head>` размещён между тегом `<html>` и тегом `<body>`:

Пример:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First HTML</title>
  <meta charset="UTF-8">
</head>

<body>
...
```

Попробуйте сами »

Примечание: Метаданные обычно определяют название документа, набор символов, стили, ссылки, скрипты и прочую мета-информацию.

Как просмотреть HTML код веб-страницы?

Вы когда-нибудь видели веб-страницу и задавались вопросом: "Как они это сделали?"

Просмотреть исходный код HTML-страницы:

Кликните правой кнопкой мыши на странице **HTML** и выберите "Просмотреть источник страницы" (в *Chrome*) или "Просмотреть источник" (в *Internet Explorer*), или "Программный код страницы" (в *Firefox*), или "Источник страницы" (в *Opera*) или подобные в других веб-браузерах. Это откроет окно, которое содержит исходный код **HTML** страницы.

Как проверить HTML код элемента на веб-странице:

Кликните правой кнопкой мыши по выделенному элементу (или пустому месту) и выберите "Проверить" или "Проверить элемент", чтобы увидеть, с каких элементов он состоит (вы увидите **HTML** и **CSS**). Вы также можете редактировать **HTML** или **CSS** на лету на панели "Элементы" или "Стили", которая открывается в браузере.

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 »

Справочник HTML тегов

Справочник HTML тегов от *W3Schools* содержит дополнительную информацию об **HTML тегах** и их атрибуты. Вы узнаете больше про **HTML-теги** и атрибуты в следующих разделах этого учебника.

Тег	Описание
<html>	Определяет начало загрузки **HTML** документа
<body>	Определяет тело (body) **HTML** документа
<head>	Контейнер для всех элементов головы (head) веб-страницы (название документа, скрипты, стили, мета-информация и много другого)
<h1> - <h6>	Определяет заголовки **HTML** документа
<hr>	Определяет тематическое изменение содержания **HTML** документа

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

□ Prev Next □

HTML Идентификаторы. Атрибут Id

□ Prev Next □

HTML. Для чего необходим атрибут идентификатора - Id в HTML-коде? Использование атрибута Id

Атрибут `id` указывает уникальный идентификатор для HTML элемента (значение должно быть уникальным в HTML документе). Значение `id` может использоваться **CSS** и **JavaScript** для выполнения определённых задач для уникального элемента с заданным значением `id`. В **CSS**, чтобы выбрать элемент с определённым идентификатором, напишите символ hash (#) (решётка), а потом идентификатор (id) элемента:

Пример:

Используйте **CSS** для стилизации элемента с идентификатором "myHeader":

```
<style>
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}
</style>

<h2 id="myHeader">My Header</h2>
```

Результат:

Попробуйте сами »

Совет: Атрибут `id` можно использовать на любом **HTML** элементе.

Примечание: Значение `id` является чувствительным к регистру.

Примечание: Значение `id` должно содержать хотя бы один символ и не должно содержать пустого места (пробелы, вкладки).

Разница между классом (class) и идентификатором (id)

HTML элемент может иметь только один уникальный идентификатор, который принадлежит этому отдельному элементу, а имя класса может использоваться несколькими элементами:

Пример:

```
<style>
/* Стиль элемента с id "myHeader" */
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}

/* Стиль всех элементов с именем класса "city" */
.city {
  background-color: tomato;
  color: white;
  padding: 10px;
}
</style>

<!-- Уникальный элемент -->
<h1 id="myHeader">My Cities</h1>

<!-- Несколько похожих элементов -->
<h2 class="city">London</h2>
<p>London is the capital of England.</p>

<h2 class="city">Paris</h2>
<p>Paris is the capital of France.</p>

<h2 class="city">Tokyo</h2>
<p>Tokyo is the capital of Japan.</p>
```

Попробуйте сами »

Примечание: Вы можете узнать больше о **CSS** в Учебнике по CSS.

HTML закладки с идентификатором (id) и ссылкой

HTML закладки используются для того, чтобы читатели могли переходить к необходимым частям веб-страницы. Закладки могут быть полезными, если ваша веб-страница очень длинная (прокручивается намного вниз). Чтобы создать закладку, сначала необходимо создать саму закладку, а потом добавить ссылку на неё. Когда ссылка будет нажата, страница прокрутится к месту размещения с помощью закладки.

Пример:

Сначала создайте закладку с атрибутом `id`:

```
<h2 id="C4">Глава 4</h2>
```

Потом добавьте ссылку на закладку ("Перейти к главе 4") с одной страницы:

```
<a href="#C4">Перейти к главе 4</a>
```

Или добавьте ссылку на закладку ("Перейти к главе 4") с другой страницы:

Пример:

```
<a href="html_demo.html#C4">Перейти до глави 4</a>
```

Попробуйте сами »

Использование атрибута id в JavaScript

JavaScript может получить доступ к элементу с указанным идентификатором, используя метод `getElementById()`:

Пример:

Используйте атрибут `id` для управления текстом с помощью **JavaScript**:

```
<script>
function displayResult() {
  document.getElementById("myHeader").innerHTML = "Have a nice day!";
}
</script>
```

Попробуйте сами »

Примечание: Узнать больше про JavaScript можно в разделе HTML JavaScript или в Учебнике по JavaScript.

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 »

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

Prev Next

HTML. Iframe. Фреймы

Prev Next

HTML. Как добавить фрейм на веб-странице? Использование фреймов - iframe

Фреймы используются для отображения веб-страницы внутри другой веб-страницы.

Синтаксис фреймов

HTML фрейм определяется тегом `<iframe>`:

```
<iframe src="URL"></iframe>
```

Атрибут `src` указывает URL-адрес (веб-адрес) страницы встроенного фрейма.

Iframe - установка высоты (height) и ширины (width)

Используйте атрибуты `height` (высота) и `width` (ширина), чтобы указать размеры фрейма. Значение атрибутов по умолчанию задаются в пикселях, но они также могут быть в процентах (например, "80%"). При этом для соответствия спецификации **HTML5** в процентах можно указывать лишь через задание стилей для фрейма, то есть через **CSS** или атрибут `style`.

Если указать высоту и ширину фрейма в процентах непосредственно с помощью атрибутов `height` и `width`, то фрейм также будет работать, но при этом код не будет проходить валидацию и соответствовать спецификации **HTML5**.

Пример:

```
<iframe src="demo_iframe.htm" height="200" width="300"></iframe>
```
Попробуйте сами »

Или вы можете использовать **CSS** для установки высоты и ширины фрейма (в пикселях или процентах):

Пример:

```
<iframe src="demo_iframe.htm" style="height:200px;width:300px;"></iframe>
```
Попробуйте сами »

Iframe - удаление границы

По умолчанию фрейм имеет границу вокруг себя. Чтобы удалить границу, добавьте атрибут `style` и используйте **CSS** свойство `border`:

Пример:

```
<iframe src="demo_iframe.htm" style="border:none;"></iframe>
```
Попробуйте сами »

Примечание. Для того, чтобы убрать видимую границу фрейма, можно использовать два значения установки свойства **border**: `border:0;` или `border:none;`.

С помощью **CSS** можно также изменить размер, стиль и цвет границы фрейма:

Пример:

```
<iframe src="demo_iframe.htm" style="border:2px solid red;"></iframe>
```
Попробуйте сами »

Iframe - цель для ссылки - target

В качестве целевого фрейма для ссылки можно использовать iframe. Атрибут `target` ссылки должен ссылаться на атрибут `name` фрейма:

Пример:

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>

<p><a href="https://www.w3schools.com" target="iframe_a">W3Schools.com</a></p>
```
Попробуйте сами »

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 »

HTML тег iframe

Тег	Описание
<iframe>	Определяет встроенный фрейм

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

□ Prev Next □

HTML Изображения

□ Prev Next □

Как вставить изображение в HTML-код на веб-сайте?

Изображения могут значительно улучшить дизайн и внешний вид веб-страниц.

Пример:

```
<img src="pic_trulli.jpg" alt="Italian Trulli">
```
Попробуйте сами »

Пример:

```
<img src="img_girl.jpg" alt="Girl in a jacket">
```
Попробуйте сами »

Пример:

```
<img src="img_chania.jpg" alt="Flowers in Chania">
```
Попробуйте сами »

HTML-изображения. Синтаксис

HTML изображения определяются тегом `<img>`. Тег `<img>` пустой, он содержит лишь атрибуты и не имеет закрывающего тега.

Атрибут `src` указывает URL-адрес (веб-адрес) изображения:

Пример:

```
<img src="url">
```

Атрибут alt

Атрибут `alt` предоставляет альтернативный текст для изображения, если пользователь по какой-то причине не может его просмотреть (например, из-за медленного соединения, ошибку в атрибуте `src` или если пользователь использует экранный ридер).

Значение атрибута `alt` должно описывать изображение:

Пример:

```
<img src="img_chania.jpg" alt="Flowers in Chania">
```
Спробуйте самі! »

Если браузер не может найти изображение, он отобразит значение атрибута `alt`:

Пример:

```
<img src="wrongname.gif" alt="Flowers in Chania">
```
Попробуйте сами »

Примечание: Атрибут `alt` обязателен. Без него веб-страница не будет считаться валидной (соответствовать спецификации **HTML5**).

Размер изображения - width (ширина) и height (высота)

Атрибут `style` можно использовать для определения ширины и высоты изображения.

Пример:

```
<img src="img_girl.jpg" alt="Girl in a jacket" style="width:500px;height:600px;">
```
Попробуйте сами »

Кроме того, можно использовать атрибуты `width` и `height` непосредственно в теге `<img>`:

Пример:

```
<img src="img_girl.jpg" alt="Girl in a jacket" width="500" height="600">
```
Попробуйте сами »

Атрибуты `width` и `height` всегда определяют ширину и высоту изображения в пикселях.

Примечание: Всегда указывайте ширину - width и высоту - height изображения. Если ширина и высота не указаны, страница может мерцать во время загрузки изображения.

Ширина и высота или стиль?

Атрибуты `width`, `height` и `style` действительны в **HTML5**. Между тем, **W3C** предлагает использовать атрибут `style`. Это не позволяет таблицам стилей изменять размер изображений:

Пример:

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  width: 100%;
}
</style>
</head>
<body>

<img src="html5.gif" alt="HTML5 Icon" width="128" height="128">
<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">

</body>
</html>
```

Попробуйте сами »

Изображения в другой папке

Если не указано, браузер ожидает найти изображение в той же папке, что и веб-страница. Тем не менее, обычное сохранение изображений в подпапке, например, с названием img (или image). Затем необходимо указать имя папки в атрибуте `src`:

Пример:

```
<img src="/images/html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">
```

Попробуйте сами »

Изображения на другом сервере

Некоторые веб-сайты сохраняют свои изображения на специальных серверах для хранения изображений. На самом деле, вы можете получить доступ к изображениям с любого веб-адреса в мире:

Пример:

```
<img src="https://www.w3schools.com/images/w3schools_green.jpg" alt="W3Schools.com">
```

Попробуйте сами »

Примечание: Подробнее про пути к файлам вы можете прочитать в разделе HTML Пути к файлам.

Анимированные изображения

HTML позволяет использовать анимированные GIF-файлы:

Пример:

```
<img src="programming.gif" alt="Computer Man" style="width:48px;height:48px;">
```

Попробуйте сами »

Изображения как ссылки

Чтобы использовать изображения как ссылки, поместите тег `<img>` в тег `<a>`:

Пример:

```
<a href="default.asp">
  <img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;border:0;">
```

```
</a>
```
Попробуйте сами »

Примечание: border: 0; добавлен, чтобы предотвратить отображение рамки вокруг изображения в IE9 (и более ранних версиях) (когда изображение является ссылкой).

Свойство Float - Плавающее изображение

Используйте **CSS** свойство `float`, чтобы позволить изображению плавать справа или слева от текста, т.е. быть обтекаемым текстом:

Пример:

```
<p><img src="smiley.gif" alt="Smiley face" style="float:right;width:42px;height:42px;">
The image will float to the right of the text.</p>

<p><img src="smiley.gif" alt="Smiley face" style="float:left;width:42px;height:42px;">
The image will float to the left of the text.</p>
```
Попробуйте сами »

Примечание: Чтобы узнать больше про CSS Float, ознакомтесь с пособием по CSS Float.

Тег <map> - Карта изображений

Тег `<map>` определяет карту изображения. Карта изображения - это изображение с интерактивными (кликабельными) областями.

На изображении ниже кликните изображения компьютера, телефона или чашки кофе:

Пример:

```
<img src="workplace.jpg" alt="Workplace" usemap="#workmap">

<map name="workmap">
  <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">
  <area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm">
  <area shape="circle" coords="337,300,44" alt="Coffee" href="coffee.htm">
</map>
```
Попробуйте сами »

где:

- map - карта;
- name - имя;
- usemap - предназначенная для пользователя карта;
- area - площадь, область;
- shape - форма;
- rect (от rectangle) - прямоугольник;
- circle - круг;
- coords (от coordinates) - координаты.

В данном примере при клике на каждом из объектов на изображении (на ноутбуке, мобильном телефоне и чашке кофе) мы будем переходить на новую страницу с описанием каждого с объектов. Если же мы попробуем кликнуть где-нибудь в другом месте на изображении вне данных объектов, то ничего не произойдёт.

Атрибут `name` тега `<map>` связан с атрибутом `usemap` тега `<img>` и создаёт связь между изображением и картой. Элемент `<map>` содержит несколько тегов `<area>`, которые определяют области изображения, на которые можно кликать мышкой и переходить по ссылке.

Фоновое изображение - Background Image

Чтобы добавить фоновое изображение к **HTML**, воспользуйтесь **CSS** свойством `background-image`:

Пример:

Чтобы добавить фоновое изображение на веб-странице, укажите свойство `background-image` для елемента BODY:

```
<body style="background-image:url('clouds.jpg');">

<h2>Background Image</h2>

</body>
```

Попробуйте сами »

Пример:

Чтобы добавить фоновое изображение к параграфу, укажите свойство `background-image` для элемента P:

```
<body>

<p style="background-image:url('clouds.jpg');">
...
</p>

</body>
```

Попробуйте сами »

Примечание: Чтобы узнать больше о фоновых изображениях, ознакомтесь с Пособием CSS Background.

Элемент <picture>

HTML5 представил элемент `<picture>`, чтобы добавить больше гибкости во время определения ресурсов изображения. Элемент `<picture>` содержит ряд элементов `<source>`, каждый из которых ссылается на разные источники изображения. Таким образом браузер может выбрать изображение, которое наиболее лучше соответствует текущему просмотру и / или устройству. Каждый элемент `<source>` имеет атрибуты, которые описывают, когда их изображение наиболее подходит. Браузер будет использовать первый элемент `<source>` из соответствующими значениями атрибутов и игнорировать любые другие следующие `<source>` элементы.

Пример:

Отображается одно изображение, если окно веб-браузера (окно просмотра) составляет минимум 650 пикселей, а другое изображение - если нет, но больше 465 пикселей.

```
<picture>
  <source media="(min-width: 650px)" srcset="img_pink_flowers.jpg">
  <source media="(min-width: 465px)" srcset="img_white_flower.jpg">
  <img src="img_orange_flowers.jpg" alt="Flowers" style="width:auto;">
</picture>
```

Попробуйте сами »

Примечание: Всегда указывайте элемент `<img>` как последний дочерний элемент элемента `<picture>`. Элемент `<img>` используется браузерами, которые не поддерживают элемент `<picture>`, или если ни один из тегов `<source>` не был сопоставлен.

HTML считыватели экрана (скринридеры)

Считыватель экрана (скринридер) - это программа, которая читает HTML-код, конвертирует текст и позволяет пользователю "слушать" содержание. Считыватели экрана полезны для людей с недостатками зрения или для обучения людей с ограниченными физическими возможностями.

Резюме раздела

- Используйте **HTML** элемент `<img>`, чтобы определить изображение
- Используйте **HTML** атрибут `src` для определения URL-адреса изображения
- Используйте **HTML** атрибут `alt` для определения альтернативного текста для изображения, если оно не может быть отображено
- Используйте **HTML** атрибуты `width` и `height`, чтобы определить размер изображения
- Используйте **CSS** свойства `width` и `height` для определения размера изображения (альтернативно)
- Используйте **CSS** свойство `float`, чтобы позволить изображению быть обтекаемым
- Используйте **HTML** элемент `<map>`, чтобы определить карту изображения

- Используйте **HTML** элемент `<area>`, чтобы определить области, которые можно нажимать на карте изображения
- Используйте **HTML** атрибут `usemap` элемента `<img>`, чтобы указать на карте изображение
- Используйте **HTML** элемент `<picture>` для показа разных изображений для разных устройств

Примечание: Загрузка изображений занимает много времени. Большие изображения могут значительно замедлить работу вашей страницы. Используйте изображения, учитывая их размеры и вес.

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 » Упражнение 5 » Упражнение 6 »

HTML теги изображения

Тег	Описание
`<img>`	Определяет изображение
`<map>`	Определяет карту изображения
`<area>`	Определяет область, которую можно нажимать на карте изображения
`<picture>`	Определяет контейнер для нескольких ресурсов изображения

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

□ Prev Next □

HTML Основы. Введение

□ Prev Next □

Что такое HTML?

HTML является стандартным языком разметки для создания веб-страниц.

- **HTML** означает *Hyper Text Markup Language* (язык гипертекстовой разметки)
- **HTML** описывает структуру веб-страниц с помощью разметки
- **HTML-елементы** являются строительными блоками страниц **HTML**
- Элементы **HTML** представлены *тегами*
- *Теги* **HTML** обозначают фрагменты содержания, такие как **header** (заголовок), **paragraph** (параграф), **table** (таблица) и др.
- Браузеры не показывают *теги* **HTML**, но используют их для отображения содержания web-страницы

Простой HTML документ

Пример:

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

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Попробуй сам »

Объяснение примера:

- Объявление `<!DOCTYPE html>` определяет этот документ как **HTML5**
- Элемент `<html>` является корневым элементом **HTML** страницы
- Элемент `<head>` содержит мета-информацию об **HTML** документе
- Элемент `<title>` указывает название **HTML** документа
- Элемент `<body>` отображает видимое содержание **HTML** страницы
- Элемент `<h1>` определяет большой заголовок на **HTML** странице
- Элемент `<p>` определяет абзац (параграф) в **HTML** документе

Теги HTML

HTML-теги - это названия элементов, окружённые угловыми скобками:

<название тега>Здесь идёт содержание...</название тега>

- Теги **HTML** обычно идут **парами**, например, `<p>` и `</p>`
- Первым тегом в паре есть **начальный тег**, второй - **конечный тег**
- Конечный тег записывается как начальный тег, но перед названием тега вставляется косая черта (/ слэш)

Примечание: Начальный тег также называется - **открывающим тегом**, а конечный тег - **закрывающим тегом**.

Веб-браузеры

Целью веб-браузера (Chrome, Firefox, Opera, Internet Explorer, Safari) является чтение документов *HTML* и их отображение. Браузер не отображает теги *HTML*, но использует их, чтобы определить, как отобразить веб-страницу:

Отображение HTML-страницы в браузере

Структура HTML страницы

Ниже приведена визуализация структуры *HTML-страницы*:

Примечание: В браузере отображается только содержание раздела <body>.

Оъявление <!DOCTYPE>

Объявление `<!DOCTYPE>` представляет собой определение типа документа и помогает браузерам правильно отображать веб-страницы. Оно должно появляться только один раз в верхней части страницы (перед любыми тегами *HTML*).

Объявление `<!DOCTYPE>` не чувствительно к регистру.

Объявление `<!DOCTYPE>` для **HTML5** выглядит так:

<!DOCTYPE html>

Версии HTML

С первых дней существования Интернета было много версий *HTML*:

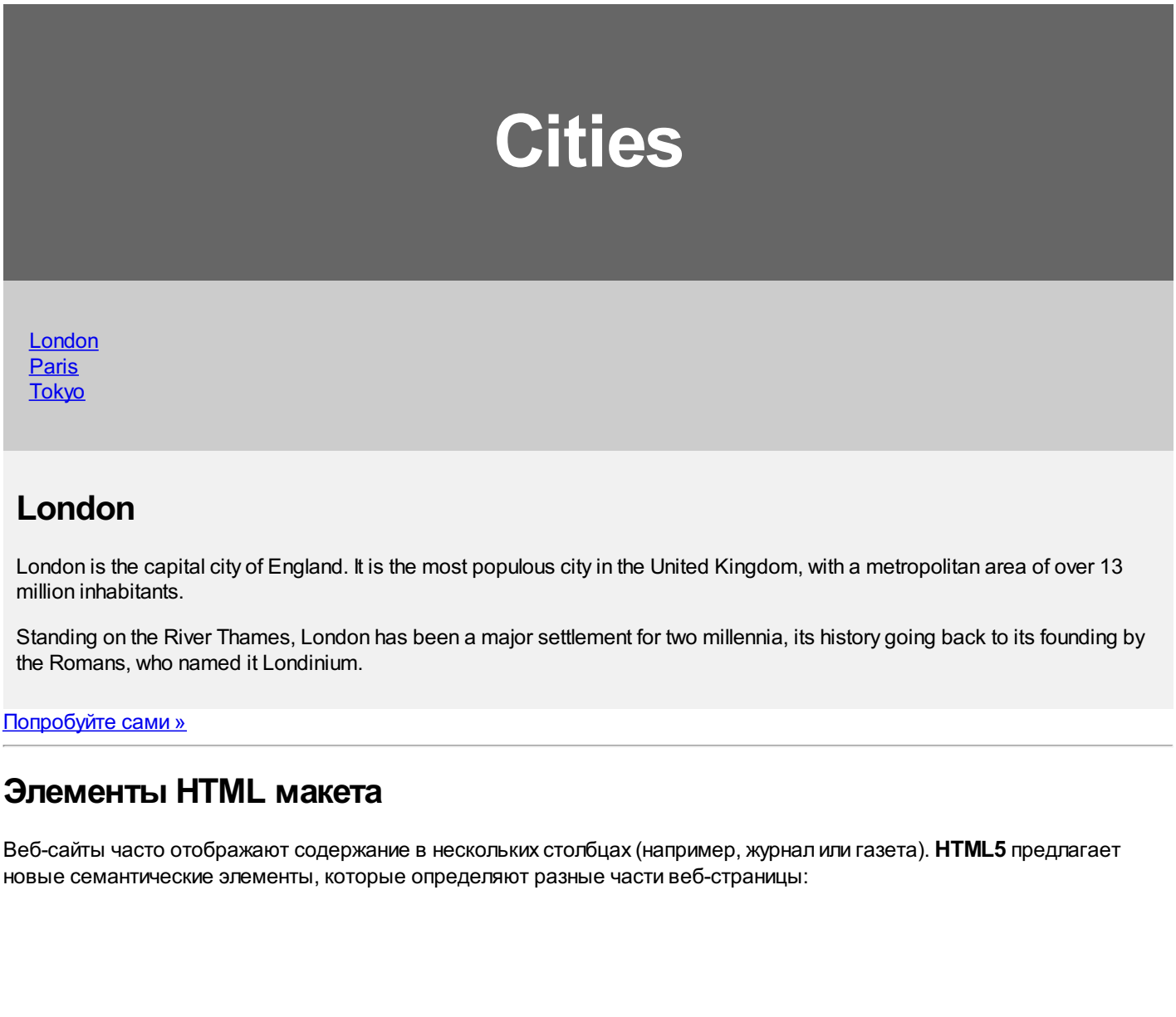
Версия	Год появления
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2014

❮ Prev Next ❯

HTML Макеты веб-страниц

❮ Prev Next ❯

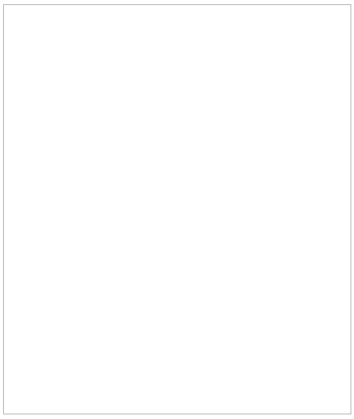
Пример HTML макета



Попробуйте сами »

Элементы HTML макета

Веб-сайты часто отображают содержание в нескольких столбцах (например, журнал или газета). **HTML5** предлагает новые семантические элементы, которые определяют разные части веб-страницы:



- `<header>` - Определяет заголовок для документа или раздела
- `<nav>` - Определяет контейнер для навигационных ссылок
- `<section>` - Определяет раздел (секцию) в документе
- `<article>` - Определяет независимую автономную статью
- `<aside>` - Определяет содержание, кроме основного содержания (например, боковую вставку)
- `<footer>` - Определяет нижний колонтитул (футер) для документа или раздела
- `<details>` - Определяет дополнительные детали
- `<summary>` - Определяет заголовок для элемента `<details>`

Методика HTML разметки

Существует пять разных способов создания многоколоновых макетов. Каждый способ имеет свои плюсы и минусы:

- HTML tables (табличный - **не рекомендуется!**)
- CSS float (обтекание)
- CSS flexbox (гибкая коробка)
- CSS framework (фреймворк)
- CSS grid (сетка)

Какой способ вёрстки выбрать?

HTML таблицы

Элемент `<table>` не был разработан как инструмент компоновки веб-страниц! Целью элемента `<table>` является отображение табличных данных. Значит, не используйте таблицы для создания макета страницы! Они принесут беспорядок в ваш код. И представьте, как тяжело будет сделать редизайн вашего сайта (т.е. изменить дизайн) через какое-то время, т.к. придётся практически полностью переписывать **HTML** код.

Совет: НЕ используйте таблицы для создания макета веб-страницы!

CSS фреймворки (frameworks)

Если вы хотите быстро создать макет, вы можете использовать фреймворк, например W3.CSS, Bootstrap или любой другой (Semantic UI, Materialize, Pure, Foundation, Milligram, Bulma, Ulkit, Skeleton, Base, Spectre, Picnic, Tailwind CSS, Material-UI, Leaf, YAML CSS, Gumby и т.д.).

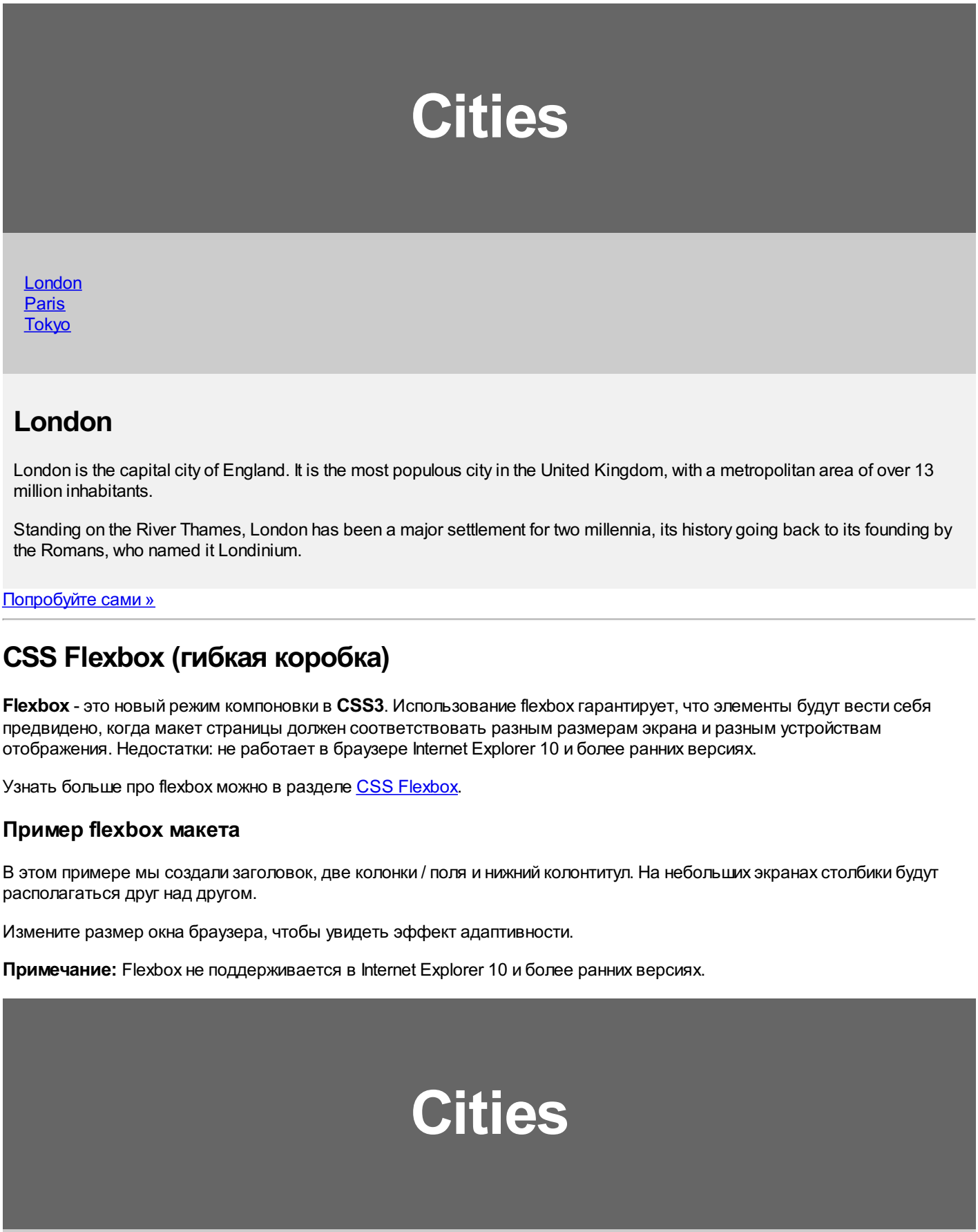
CSS Float (обтекание)

Это обычное использование целых веб-макетов с помощью свойства **CSS float**. Float легко выучить - необходимо просто запомнить, как работают float и чистые свойства. Недостатки: плавающие элементы привязаны к потоку документов, что может мешать гибкости. Подробнее о float можно узнать в разделе CSS Float и Clear.

Пример float макета

В этом примере мы создали заголовок, две колонки / поля и нижний колонтитул. На небольших экранах столбики будут располагаться друг над другом.

Измените размер окна веб-браузера, чтобы увидеть эффект реагирования (об этом вы узнаете в следующим разделе - HTML Адаптивность.)



Попробуйте сами »

CSS Flexbox (гибкая коробка)

Flexbox - это новый режим компоновки в **CSS3**. Использование flexbox гарантирует, что элементы будут вести себя предвидено, когда макет страницы должен соответствовать разным размерам экрана и разным устройствам отображения. Недостатки: не работает в браузере Internet Explorer 10 и более ранних версиях.

Узнать больше про flexbox можно в разделе CSS Flexbox.

Пример flexbox макета

В этом примере мы создали заголовок, две колонки / поля и нижний колонтитул. На небольших экранах столбики будут располагаться друг над другом.

Измените размер окна браузера, чтобы увидеть эффект адаптивности.

Примечание: Flexbox не поддерживается в Internet Explorer 10 и более ранних версиях.

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.

Попробуйте сами! »

CSS Grid (сетка)

Модуль **CSS Grid Layout** предлагает систему компоновки на основе сетки, со строками и столбцами, что облегчает разработку веб-страниц без использования float и позиционирования.

Недостатки: не работает ни в Internet Explorer, ни в Edge 15, ни в более ранних версиях.

Подробнее про **CSS grid** смотрите в разделе CSS Grid.

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

□ Prev Next □

HTML Ссылки

□ Prev Next □

HTML Как создать ссылки на веб-сайте?

Ссылки находятся почти на всех веб-страницах. Ссылки позволяют пользователям нажимать их и переходить от страницы к странице на сайте, а иногда между сайтами.

HTML-ссылки - гиперссылки

HTML-ссылки - это гиперссылки. Вы можете нажать на ссылку и перейти на другую веб-страницу. Когда вы передвигаете курсор мыши через ссылку, стрелка мыши преобразуется в маленькую руку.

Примечание: Ссылка не обязательно должна быть текстом. Это может быть также изображение или любой другой элемент HTML.

HTML-ссылка - синтаксис

HTML ссылки определяются тегом `<a>`:

```
<a href="url">link text</a>
```

Пример:

```
<a href="https://www.w3schools.com/html/">Visit our HTML tutorial</a>
```

Попробуйте сами »

Атрибут `href` определяет адрес назначения (https://www.w3schools.com/html/) ссылки. Текст ссылки является видимой частью (Visit our HTML tutorial). Нажав на текст ссылки, вас отправит на указанный веб-адрес.

Локальные ссылки

В приведенном выше примере использован абсолютный URL-адрес (полный веб-адрес). Локальная ссылка (ссылка на тот самый веб-сайт) указана с относительным URL (без https://www....).

Пример:

```
<a href="html_images.asp">HTML Images</a>
```

Попробуйте сами »

Цвета HTML ссылки

По умолчанию ссылка будет отображаться так (у всех веб-браузерах):

- Непосещенная ссылка - подчеркнута и синим цветом
- Посещенная ссылка - подчёркнута и фиолетовым цветом
- Активная ссылка - подчёркнута и красным цветом

С помощью **CSS** можно изменить цвета ссылок по умолчанию:

Пример:

```
<style>
a:link {
  color: green;
  background-color: transparent;
  text-decoration: none;
}

a:visited {
  color: pink;
  background-color: transparent;
  text-decoration: none;
}

a:hover {
  color: red;
  background-color: transparent;
  text-decoration: underline;
}

a:active {
  color: yellow;
  background-color: transparent;
  text-decoration: underline;
}
</style>
```

Попробуйте сами »

Ссылки часто используются как кнопки, которые делаются с помощью CSS:

Это ссылка

Приклад:

```
<style>
a:link, a:visited {
  background-color: #f44336;
  color: white;
  padding: 15px 25px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
}

a:hover, a:active {
  background-color: red;
}
</style>
```

Попробуйте сами »

Примечание: Чтобы узнать больше про **CSS**, перейдите к Учебнику CSS.

HTML-ссылки - атрибут target

Атрибут `target` (пер. - цель) определяет, где открывать связанный документ.

Атрибут `target` может иметь одно из таких значений:

- `_blank` - открывает связанный документ в новом окне или вкладке
- `_self` - открывает связанный документ в том же окне / вкладке, в котором была нажата ссылка
- `_parent` - открывает связанный документ в родительском фрейме
- `_top` - открывает связанный документ в полном окне
- *framename* - открывает связанный документ в указанном фрейме

Этот пример откроет связанный документ в новом окне / вкладке веб-браузера:

Пример:

```
<a href="https://www.w3schools.com/" target="_blank">Visit W3Schools!</a>
```
Попробуйте сами »

Совет. Если веб-страница заблокирована во фрейме, вы можете воспользоваться `target="_top"`, чтобы выйти с фрейма:

Пример:

```
<a href="https://www.w3schools.com/html/" target="_top">HTML5 tutorial!</a>
```
Попробуйте сами »

HTML-ссылка - изображение как ссылка

Обычно изображения используются как ссылки:

Пример:

```
<a href="default.asp">
  <img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;border:0;">
</a>
```
Попробуйте сами »

Примечание: `border: 0;` добавляется, чтобы предотвратить отображение рамки вокруг изображения в IE9 (и более ранних версиях) (когда изображение является ссылкой). Хотя для других современных браузеров данный параметр не нужен.

HTML-ссылки - атрибут title

Атрибут `title` определяет дополнительную информацию про элемент. Эта информация обычно отображается как текст-подсказка, когда мышка перемещается над элементом (например, над ссылкой).

Пример:

```
<a href="https://www.w3schools.com/html/" title="Go to W3Schools HTML section">Visit our HTML Tutorial</a>
```
Попробуйте сами »

HTML-ссылки - bookmark (закладки)

HTML закладки (bookmark) используются для того, чтобы читатели могли переходить к определённым частям веб-страницы. Закладки могут быть полезными, если ваша веб-страница очень длинная. Чтобы создать закладку, сначала необходимо создать закладку, а потом добавить ссылку на неё. Когда ссылка будет нажата, страница прокрутиться до необходимого места с помощью закладки.

Пример

Сначала создайте закладку с атрибутом `id`:

```
<h2 id="C4">Глава 4</h2>
```

Потом добавьте ссылку на закладку ("Перейти до главы 4") с одной страницы:

```
<a href="#C4">Перейти до глави 4</a>
```

или добавьте ссылку на закладку ("Перейти до главы 4") с другой страницы:

```
<a href="html_demo.html#C4">Перейти до главы 4</a>
```
Попробуйте сами »

Внешние пути

На внешние страницы можно ссылаться с полным URL-адресом или относительно текущей веб-страницы.

Этот пример использует полный URL-адрес для ссылки на веб-страницу:

```
<a href="https://www.w3schools.com/html/default.asp">HTML tutorial</a>
```
Попробуйте сами »

Этот пример ссылается на страницу, размещенную в папке html на текущем веб-сайте:

```
<a href="/html/default.asp">HTML tutorial</a>
```
Попробуйте сами »

Этот пример ссылается на страницу, расположенную в той же папке, что и текущая страница:

```
<a href="default.asp">HTML tutorial</a>
```
Попробуйте сами »

Примечание: Подробнее про пути к файлам вы можете прочитать в разделе HTML Пути к файлам.

Резюме раздела

- Используйте элемент `<a>` чтобы определить ссылку
- Используйте атрибут `href` для определения адреса ссылки
- Используйте атрибут `target` чтобы определить, где открыть связанный документ
- Используйте елемент `<img>` (внтури <a>) чтобы использовать изображение как ссылку
- Используйте атрибут `id` (id="значение"), чтобы определить закладки на странице
- Используйте атрибут `href` (href="#значение") для ссылки на закладку

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 » Упражнение 5 »

HTML теги ссылок

Тег	Описание
<a>	Определяет гиперссылку

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

« Prev Next »

HTML Списки

« Prev Next »

Как создать списки на веб-странице?

HTML список. Пример

Неупорядоченный (ненумерованный) список:

- Item
- Item
- Item
- Item

Упорядоченный (нумерованный) список:

1. First item
2. Second item
3. Third item
4. Fourth item

Попробуйте сами »

Неупорядоченный список HTML

Неупорядоченный (ненумерованный) список начинается с тега `<ul>`. Каждый элемент списка начинается с тега `<li>`. Элементы списка будут обозначены метками (маленькие черные кружочки) по умолчанию:

Пример:

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Попробуйте сами »

Неупорядоченный список HTML - выберите маркер пункта списка

CSS свойство `list-style-type` используется для определения стиля маркера элемента списка:

Значение	Описание
disc	Устанавливает маркер элемента списка в виде закрашенного кружочка (как и по умолчанию)
circle	Устанавливает маркер элемента списка в виде пустого кружочка
square	Устанавливает маркер элемента списка в виде квадрата
none	Элементы списка не будет помечено (то есть маркеры не будут отображаться)

Пример - **disc**:

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Попробуйте сами »

Пример - **circle**:

```
<ul style="list-style-type:circle;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Попробуйте сами »

Пример - **square**:

```
<ul style="list-style-type:square;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Попробуйте сами »

Пример - **none**:

```
<ul style="list-style-type:none;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Попробуйте сами »

Упорядоченный HTML список

Упорядоченный (нумерованный) список начинается с тега `<ol>`. Каждый элемент списка начинается с тега `<li>`. Элементы списка будут обозначены номерами по умолчанию:

Пример:

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Попробуйте сами »

Упорядоченный HTML список - атрибут type

Атрибут `type` тега `<ol>` определяет тип маркера элемента списка:

Значение	Описание
type="1"	Элементы списка будут пронумерованы числами (по умолчанию)
type="A"	Элементы списка будут пронумерованы большими буквами
type="a"	Элементы списка будут пронумерованы маленькими буквами
type="I"	Элементы списка будут пронумерованы большими римскими числами
type="i"	Элементы списка будут пронумерован маленькими римскими числами

Числа:

```
<ol type="1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Попробуйте сами »

Большие буквы:

```
<ol type="A">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Попробуйте сами »

Маленькие буквы:

```
<ol type="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Попробуйте сами »

Большие римские цифры:

```
<ol type="I">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Попробуйте сами »

Маленькие римские цифры:

```
<ol type="i">
  <li>Coffee</li>
```

```
  <li>Tea</li>
  <li>Milk</li>
</ol>
```
Попробуйте сами »

HTML списки описания

HTML также поддерживает списки описания.

Список описания - это список терминов с описанием каждого термина. Тег `<dl>` определяет список описания, тег `<dt>` определяет термин (название), а тег `<dd>` описывает каждый термин:

Пример:

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```
Попробуйте сами »

Вложенные HTML списки

Список может быть вложенным (списки внутри списков):

Пример:

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```
Попробуйте сами »

Примечание: Элементы списка могут содержать новый список и другие элементы **HTML**, например изображения, ссылки и др.

Контроль подсчёта списка

По умолчанию, упорядоченный список начнёт отсчёт с единицы - 1. Если вы хотите начать отсчёт с какого-то другого определённого числа, вы можете использовать атрибут `start` (начало):

Пример:

```
<ol start="50">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```
Попробуйте сами »

Горизонтальный список с помощью CSS

HTML списки можно создавать разными способами с помощью **CSS**. Одним из популярных способов является формирование списка по горизонтали, чтобы создать **навигационное меню**:

Пример:

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

```
<head>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333333;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 16px;
  text-decoration: none;
}

li a:hover {
  background-color: #111111;
}
</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

</body>
</html>
```

Попробуйте сами »

Примечание: Вы можете узнать больше о **CSS** в Учебнике по **CSS**.

Резюме раздела

- Используйте **HTML** элемент `<ul>` для определения неупорядоченного списка
- Используйте **CSS** свойство `list-style-type` для определения вида маркера элемента списка
- Используйте **HTML** элемент `<ol>` для определения упорядоченного (нумерованного) списка
- Используйте **HTML** атрибут `type` чтобы определить тип нумерации
- Используйте **HTML** элемент `<li>` для определения элемента списка
- Используйте **HTML** элемент `<dl>` для определения списка описания
- Используйте **HTML** элемент `<dt>` чтобы определить термин описания
- Используйте **HTML** элемент `<dd>` чтобы описать термин в списке описания
- Списки могут быть вложенными внутри списков
- Элементы списка могут содержать другие элементы **HTML**
- Используйте **CSS** свойство `float: left` или `display: inline` для отображения списка горизонтально

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 » Упражнение 5 » Упражнение 6 »

HTML теги списка

Тег	Описание
`<ul>`	Определяет неупорядоченный (ненумерованный) список

 Определяет упорядоченный (нумерованный) список
 Определяет элемент списка
<dl> Определяет список описания
<dt> Определяет термин в списке описания
<dd> Описывает термин в списке описания

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

Prev Next

HTML Мультимедиа

Prev Next

Мультимедиа в Интернете - это звук, музыка, видео, фильмы и анимация.

Что такое мультимедиа?

Мультимедиа поставляется в различных форматах. Это может быть почти всё, что можно услышать или увидеть.

Примеры: изображение, музыка, звук, видео, записи, фильмы, анимация и тому подобное.

Интернет-страницы часто содержат мультимедийные элементы различных типов и форматов.

В этом разделе вы узнаете о различных мультимедийных форматах.

Поддержка браузерами

Первые веб-браузеры поддерживали только текст, ограничиваясь одним шрифтом одного цвета.

Позже появились браузеры с поддержкой различных цветов и шрифтов, и изображений!

Основные браузеры обрабатывают аудио, видео и анимацию по-разному. Пддерживаются различные форматы, а для некоторых форматов нужны дополнительные вспомогательные программы (плагины) для работы.

Надеемся, что это всё станет историей. Мультимедийные технологии **HTML5** обещают более легкое будущее для мультимедиа.

Форматы мультимедиа

Мультимедийные элементы (например, аудио или видео) хранятся в медиафайлах.

Самым распространенным способом выявления типа файла является просмотр расширение файла.

Мультимедийные файлы имеют такие форматы и различные расширения: .swf, .wav, .mp3, .mp4, .mpg, .wmv, .avi.

Общие форматы видео

Videoformats

MP4 является новым и становящимся основным форматом для Интернет-видео.

MP4 рекомендуется сайтом YouTube.

MP4 поддерживается Flash плеерами.

MP4 поддерживается HTML5.

Формат	Файл	Описание
MPEG	.mpg .mpeg	MPEG. Разработан Moving Pictures Expert Group. Первый популярный видеоформат в Интернете. Использовался для поддержки всех веб-браузеров, но не поддерживается в HTML5 (см. MP4).

AVI	.avi	AVI (Audio Video Interleave). Разработан компанией Microsoft. Обычно используется в видеокамерах и телевизионном оборудовании. Отлично воспроизводится на компьютерах Windows, но не в веб-браузерах.
WMV	.wmv	WMV (Windows Media Video). Разработан компанией Microsoft. Обычно используется в видеокамерах и телевизионном оборудовании. Отлично воспроизводится на компьютерах Windows, но не в веб-браузерах.
QuickTime	.mov	QuickTime. Разработан Apple. Обычно используется в видеокамерах и телевизионном оборудовании. Отлично воспроизводится на компьютерах Apple, но не в веб-браузерах. (см. MP4)
RealVideo	.rm .ram	RealVideo. Разработан Real Media, что позволяет осуществлять потоковое видео с низкой пропускной способностью. Он по-прежнему используется для онлайн-видео и интернет-телевидения, но не воспроизводится в веб-браузерах.
Flash	.swf .flv	Flash. Разработан компанией Macromedia. Часто нужно иметь дополнительный компонент (плагин) для воспроизведения в веб-браузерах.
Ogg	.ogg	Theora Ogg. Разработан Xiph.Org Foundation. Поддерживается HTML5.
WebM	.webm	WebM. Разработан веб-гигантами Mozilla, Opera, Adobe и Google. Поддерживается HTML5.
MPEG-4 или MP4	.mp4	MP4. Разработан Moving Pictures Expert Group. На основе QuickTime. Обычно используется в новых видеокамерах и телевизионном оборудовании. Поддерживается всеми браузерами HTML5. Рекомендуется YouTube.

Только видео в форматах MP4, WebM и Ogg поддерживается стандартом HTML5.

Аудио форматы

MP3 - самый новый формат для записанной музыки. Термин MP3 стал синонимом цифровой музыки.

Если на вашем веб-сайте размещается записанная музыка, самый лучший выбор - это MP3.

Формат	Файл	Описание
MIDI	.mid .midi	MIDI (от англ. Musical Instrument Digital Interface - цифровой интерфейс музыкальных инструментов). Главный формат для всех электронных музыкальных устройств, таких как синтезаторы и звуковые карты ПК. Файлы MIDI не содержат звука, а цифровые заметки, которые могут воспроизводиться электроникой. Отлично воспроизводится на всех компьютерах и музыкальных устройствах, но не в веб-браузерах.
RealAudio	.rm .ram	RealAudio. Разработан Real Media, позволяет передавать аудио с низкой пропускной способностью. Не работает в веб-браузерах.
WMA	.wma	WMA (Windows Media Audio). Разработан компанией Microsoft. Обычно используется в музыкальных плеерах. Отлично воспроизводится на компьютерах Windows, но не в веб-браузерах.
AAC	.aac	AAC (Advanced Audio Coding). Разработан компанией Apple как стандартный формат для iTunes. Отлично воспроизводится на компьютерах Apple, но не в веб-браузерах.
WAV	.wav	WAV. Разработан IBM и Microsoft. Отлично воспроизводится в операционных системах Windows, Macintosh и Linux. Поддерживается HTML5.
Ogg	.ogg	Ogg. Разработан Xiph.Org Foundation. Поддерживается HTML5.
MP3	.mp3	Файлы MP3 - это звуковая часть файлов MPEG. MP3 - самый популярный формат для музыкальных плееров. Сочетает в себе хорошее сжатие (малые файлы) с высоким качеством. Поддерживаются всеми браузерами.
MP4	.mp4	MP4 является видеоформатом, но также может использоваться для аудио. MP4 видео - это новый формат видео в Интернете, который уже становится основным. Это приводит к автоматической поддержке MP4 аудио во всех браузерах.

Стандарт HTML5 поддерживает лишь аудиофайлы MP3, WAV и Ogg.

□ Prev Next □

HTML Plug-ins. Плагины (дополнительные модули)

□ Prev Next □

Для чего нужны плагины?

Задача плагина - это расширение функциональности веб-браузера.

HTML Помощники (плагины)

Вспомогательные приложения (плагины) - это компьютерные программы, которые расширяют стандартную функциональность веб-браузера.

Примерами известных плагинов являются Java-апплеты.

Плагины можно добавлять к веб-страницам с помощью тега `<object>` или тега `<embed>`.

Плагины можно использовать для многих целей: отображения карт, поиска вирусов, проверки вашего банковского идентификатора и т.д.

Чтобы отобразить видео и аудио: используйте `<video>` и `<audio>` теги.

Элемент <object> (объект)

Элемент `<object>` поддерживается всеми браузерами.

Элемент `<object>` определяет встроенный объект в документе HTML.

Он используется для встраивания плагинов (например, Java-апплетов, PDF-ридеров, Flash-плееров) в веб-страницы.

Пример:

```
<object width="400" height="50" data="bookmark.swf"></object>
```
Попробуйте сами »

Элемент `<object>` также можно использовать для включения HTML в HTML:

Пример:

```
<object width="100%" height="500px" data="snippet.html"></object>
```
Попробуйте сами »

Или изображение, если хотите:

Пример:

```
<object data="audi.jpeg"></object>
```
Попробуйте сами »

Элемент <embed>

Элемент `<embed>` поддерживается во всех основных браузерах.

Элемент `<embed>` также определяет встроенный объект в HTML документе.

Веб-браузеры поддерживали элемент <embed> длительное время. Хотя он не был частью спецификации HTML до спецификации HTML5.

Пример:

```
<embed width="400" height="50" src="bookmark.swf">
```
Попробуйте сами »

Обратите внимание, что элемент `<embed>` не имеет закрывающего тега. Он не может содержать альтернативного текста.

Элемент `<embed>` также можно использовать для включения HTML в HTML:

Пример:

```
<embed width="100%" height="500px" src="snippet.html">
```
Попробуйте сами »

Или изображение, если хотите:

Пример:

```
<embed src="audi.jpeg">
```
Попробуйте сами »

❮ Prev Next ❯

HTML. Параграфы (абзацы)

❮ Prev Next ❯

Параграфы в HTML. Как сделать параграфы на веб-страницах?

HTML элемент `<p>` определяет параграф (абзац):

Примечание: Слово *paragraph* обычно переводится, как *абзац*. Но в среде веб-разработчиков чаще используют слово *параграф*, т.е. отступ текста с новой строки. Здесь и далее будет использоваться именно слово *параграф*.

Пример:

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```
Попробуйте сами »

Примечание: Браузеры автоматически добавляют пробел (margin) до и после параграфа.

Отображение HTML

Вы не можете знать наверняка, где именно будет отображаться ваша веб-страница, на каких дисплеях или устройствах. Большие или маленькие экраны и размеры окон показывают разные результаты. С помощью HTML-кода вы не можете изменить исходное отображение, добавляя дополнительные пробелы или дополнительные строки в вашем HTML-коде. Во время отображения страницы веб-браузер удаляет любые дополнительные пробелы и дополнительные строки:

Пример:

```
<p>
This paragraph
contains a lot of lines
in the source code,
but the browser
ignores it.
</p>

<p>
This paragraph
contains        a lot of spaces
in the source        code,
but the        browser
ignores it.
</p>
```
Попробуйте сами »

Не забывайте про конечный тег

Большинство браузеров будет отображать HTML правильно, даже если вы забудете конечный тег:

Пример:

```
<p>This is a paragraph.
<p>This is another paragraph.
```
Попробуйте сами »

Пример выше будет работать в большинстве браузеров, но не надейтесь на это.

Примечание: Удаление конечного тега может привести до неожиданных результатов или ошибок в отображении веб-страницы.

Разрывы HTML-строк

HTML элемент
 определяет **разрыв строки**. Используйте функцию <br>, если необходимо прервать строку (создать новую строку), не начиная новый параграф:

Пример:

```
<p>This is<br>a paragraph<br>with line breaks.</p>
```
Попробуйте сами »

Тег
 - это пустой тег, что означает, что он не имеет конечного тега.

Проблема отображения поэзии

Этот стих будет отображаться в одну строку:

Пример:

```
<p>
 Зродились ми великої години,

 З пожеж війни і з полум'я вогнів,

 Плекав нас біль по втраті України,

 Кормив нас гнів і злість на ворогів.
</p>
```
Попробуйте сами »

HTML элемент <pre>

HTML элемент <pre> определяет предварительно отформатированный текст. Текст внутри элемента <pre> отображается шрифтом фиксированной ширини (обычно Courier), и он сохраняет пробелы и разрывы строк так, как написано:

Пример:

```
<pre>
 І ось ідем у бою життєвому

 Міцні, тверді, незломні мов граніт,

 Бо плач не дав свободи ще нікому,

 А хто борець, той здобуває світ.
</pre>
```
Попробуйте сами »

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 »

Справочник HTML тегов

Справочник HTML тегов *W3Schools* содержит дополнительную информацию про **HTML** элементы и их атрибуты.

Тег	Описание
<p>	Определяет параграф (абзац)
 	Вставляет один разрыв строки
<pre>	Определяет предварительно отформатированный текст

HTML Викторина. Проверочные вопросы

❮ Prev Next ❯

Вы можете проверить свои навыки HTML с викториной от W3Schools.

Тест

Викторина (тест) содержит 40 вопросов и не имеет ограничений по времени.

Тест не является официальным, это просто хороший способ узнать, насколько хорошо вы знаете или не знаете HTML.

Посчитайте свои баллы

Вы получаете 1 балл за каждый правильный ответ. В конце викторины отобразится ваш общий балл. Максимальная сумма баллов - 40.

Начало Викторины

Удачи Вам!

Начать HTML Викторину ❯

Если вы ещё не знаете HTML, предлагаем вам прочитать HTML Учебник из самого начала.

❮ Prev Next ❯

HTML Цитаты

❮ Prev Next ❯

HTML цитаты и элементы цитирования. Как добавлять цитаты на веб-

страницах?

Цитата:

> Цитата бывшего чемпиона мира по боксу и мэра Киева Виталия Кличко:
>
> А сегодня в завтрашний день не все могут смотреть. Вернее смотреть могут не только лишь все, мало кто может это делать...

Попробуйте сами »

HTML элемент <q> для коротких цитат

Элемент **HTML** `<q>` определяет короткую цитату. Браузеры обычно вставляют кавычки вокруг элемента `<q>`.

Пример:

```
<p>Мета WWF - це: <q>Створити майбутнє, де люди житимуть у гармонії з природою. </q></p>
```

Попробуйте сами »

HTML элемент <blockquote> для цитирования

HTML элемент `<blockquote>` определяет раздел, который цитируется с другого источника. Браузеры обычно отображают элементы `<blockquote>` с отступом.

Пример:

```
<p>Here is a quote from WWF's website:</p>
<blockquote cite="http://www.worldwildlife.org/who/index.html">
For 50 years, WWF has been protecting the future of nature.
The world's leading conservation organization,
WWF works in 100 countries and is supported by
1.2 million members in the United States and
close to 5 million globally.
</blockquote>
```

Попробуйте сами »

HTML элемент<abbr> для аббревиатур

Элемент HTML `<abbr>` определяет аббревиатуру (сокращение) или акроним. Пометка аббревиатур может дать полезную информацию для браузеров, систем перевода и поисковых систем.

Пример:

```
<p>The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.</p>
```

Попробуйте сами »

HTML элемент <address> для контактной информации

HTML элемент `<address>` определяет контактную информацию (автор / собственник) документа или статьи. Элемент `<address>` по умолчанию отображается курсивом. Большинство браузеров добавляют разрыв строк до и после элемента.

Пример:

```
<address>
Written by John Doe.<br>
Visit us at:<br>
Example.com<br>
Box 564, Disneyland<br>
USA
</address>
```

Попробуйте сами »

HTML элемент <cite> для названия работы

HTML элемент `<cite>` определяет название произведения. Браузеры обычно показывают `<cite>` элементы курсивом.

Пример:

```
<p><cite>The Scream</cite> by Edvard Munch. Painted in 1893.</p>
```

Попробуйте сами »

HTML элемент <bdo> для двунаправленного переопределения

HTML элемент `<bdo>` используется для переопределения текущего направления текста.

Пример:

```
<bdo dir="rtl">This text will be written from right to left</bdo>
```

Попробуйте сами »

Проверьте себя с помощью упражнений

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 »

HTML цитаты и элементы цитирования

Тег	Описание
<abbr>	Определяет аббревиатуру или акроним
<address>	Определяет контактную информацию автора / владельца документа
<bdo>	Определяет направление текста
<blockquote>	Определяет раздел, который цитируется с другого источника
<cite>	Определяет название произведения
<q>	Определяет короткую встроенную цитату

Справочник HTML тегов

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

Prev Next

HTML. Адаптивный веб-дизайн

Prev Next

Responsive Web Design

Что такое адаптивный веб-дизайн?

Адаптивный веб-дизайн - это использование **HTML** и **CSS** для возможности автоматического изменения размера, скрытия, уменьшения или увеличения веб-страницы, чтобы она выглядела хорошо (читабельно) на всех устройствах (настольных компьютерах, планшетах и смартфонах):

Попробуйте сами »

Примечание: Веб-страница должна хорошо выглядеть на любом устройстве с любым разрешением экрана!

Установка окна просмотра - значение viewport

Создавая адаптивные веб-страницы, добавляйте следующий элемент `<meta>` на все веб-страницы:

Пример:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Попробуйте сами »

Это позволит установить окно просмотра страницы, что укажет инструкциям браузера, как управлять размерами и масштабированием страницы.

Ниже приведён пример веб-страницы без метатега viewport и той же веб-страницы с метатегом viewport:

Без viewport метатега

Из viewport метатегом

Примечание: Если вы просматриваете эту страницу с помощью телефона или планшета, можно нажать две ссылки ниже, чтобы увидеть разницу (в другом окне).

Адаптивные изображения

Адаптивными изображениями являются изображения, которые прекрасно размещаются на веб-странице при любом размере окна веб-браузера.

Использование свойства width

Если CSS свойство `width` (ширина) установлено на 100%, изображение будет реагировать и масштабироваться, уменьшаясь или увеличиваясь, заполняя всю ширину страницы:

Girl

Пример:

```
<img src="img_girl.jpg" style="width:100%;">
```

Попробуйте сами »

Обратите внимание, что в приведённом выше примере изображение может быть расширено, чтобы быть больше, чем его начальный размер. Лучшее решение обычно будет - это использовать свойство max-width вместо этого, то есть установка максимальной ширины.

Использование свойства max-width

Если свойство `max-width` установлено на 100%, изображение уменьшится, если необходимо, но никогда не будет больше, чем его начальный размер.

Пример:

```
<img src="img_girl.jpg" style="max-width:100%;height:auto;">
```

Попробуйте сами »

Отображение разных изображений в зависимости от ширины окна браузера

HTML элемент `<picture>` позволяет определять разные изображения для разных размеров окна браузера.

Измените размер окна веб-браузера, чтобы увидеть, как изменяется изображение в зависимости от размера окна браузера:

Пример:

```
<picture>
  <source srcset="img_smallflower.jpg" media="(max-width: 600px)">
  <source srcset="img_flowers.jpg" media="(max-width: 1500px)">
  <source srcset="flowers.jpg">
  <img src="img_smallflower.jpg" alt="Flowers">
</picture>
```

Попробуйте сами »

Адаптивный размер текста

Размер текста можно установить с помощью модуля "vw", что означает "viewport width" (ширина окна просмотра). Таким образом, размер текста будет соответствовать размеру окна веб-браузера.

Hello World

Измените размер окна браузера, чтобы увидеть размер текста.

Пример:

```
<h2 style="font-size:10vw">Hello World</h2>
```

Попробуйте сами »

Примечание: Viewport - это размер окна браузера. 1vw = 1% от ширины окна просмотра. Если область просмотра составляет 50 см, 1vw - 0,5 см.

Медиа запросы

Кроме изменения размера текста и изображений, также часто используются медиазапросы на веб-страницах, которые соответствуют требованиям. С помощью медиа-запросов вы можете определить разные стили для разных размеров браузера.

Пример: измените размер окна веб-браузера, чтобы увидеть, что три элемента **div** ниже будут отображаться горизонтально на больших экранах и будут располагаться вертикально на небольших экранах:

Left Menu

Main Content

Right Content

Пример:

```
<style>
.left, .right {
  float: left;
  width: 20%; /* Ширина 20% по умолчанию */
}

.main {
  float: left;
  width: 60%; /* Ширина 60% по умолчанию */
}

/* Используйте медиа-запрос, чтобы добавить точку изменения на 800px: */
@media screen and (max-width: 800px) {
  .left, .main, .right {
    width: 100%; /* Ширина будет 100%, когда область просмотра составляет 800 пикселей или меньше */
  }
}
</style>
```

Попробуйте сами »

Примечание: Чтобы узнать больше про медиа запросы и адаптивный веб-дизайн, прочитайте Учебник по RWD (Responsive Web Design/Адаптивный веб-дизайн).

Адаптивная веб-страница - полный пример

Адаптивная веб-страница должна хорошо выглядеть как на больших экранах стационарных компьютеров, так и на небольших мобильных телефонах.

Hello World

Link 1

Link 2

Link 3

Link 4

Lorem Ipsum

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

About

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

Попробуйте сами »

Адаптивный веб-дизайн - фреймворки

Существует много готовых CSS-фреймворков, которые предлагают адаптивный дизайн. Они бесплатны и просты в использовании.

Использование W3.CSS

Отличный способ создать адаптивный дизайн - это использование соответствующей таблицы стилей, например W3.CSS. Фреймворк W3.CSS позволяет легко разрабатывать сайты, которые хорошо выглядят на любом экране: на десктопе, на ноутбуке, планшете или телефоне:

W3.CSS Demo

Измените размер окна страницы, чтобы увидеть адаптивность!

London

London is the capital city of England.

It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

Paris

Paris is the capital of France.

The Paris area is one of the largest population centers in Europe, with more than 12 million inhabitants.

Tokyo

Tokyo is the capital of Japan.

It is the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.

Пример:

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

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<body>

<div class="w3-container w3-green">
  <h1>W3Schools Demo</h1>
  <p>Измените размер окна страницы, чтобы увидеть адаптивность!</p>
</div>

<div class="w3-row-padding">
  <div class="w3-third">
    <h2>London</h2>
    <p>London is the capital city of England.</p>
    <p>It is the most populous city in the United Kingdom,
    with a metropolitan area of over 13 million inhabitants.</p>
  </div>

  <div class="w3-third">
    <h2>Paris</h2>
    <p>Paris is the capital of France.</p>
    <p>The Paris area is one of the largest population centers in Europe,
    with more than 12 million inhabitants.</p>
  </div>

  <div class="w3-third">
    <h2>Tokyo</h2>
    <p>Tokyo is the capital of Japan.</p>
    <p>It is the center of the Greater Tokyo Area,
    and the most populous metropolitan area in the world.</p>
  </div>
</div>

</body>
</html>
```

Попробуйте сами »

Примечание: Узнать больше про фреймворк **W3.CSS** можно в Учебнике W3.CSS.

Bootstrap

Другой популярный фреймворк - **Bootstrap**, он использует **HTML**, **CSS** и **jQuery** для создания адаптивных веб-страниц.

Пример:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Bootstrap Example</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <div class="jumbotron">
    <h1>My First Bootstrap Page</h1>
 </div>
  <div class="row">
    <div class="col-sm-4">
      ...
    </div>
    <div class="col-sm-4">
     ...
    </div>
    <div class="col-sm-4">
```

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

</body>
</html>
```

Попробуйте сами »

Примечание: Узнать больше про фреймворк **Bootstrap** можно в Учебнике Bootstrap 4 на нашем сайте.

❮ Prev Next ❯

HTML JavaScript

❮ Prev Next ❯

Как добавить JavaScript в HTML-код на веб-странице? Использование скриптов

JavaScript делает **HTML** страницы более динамичными и интерактивными.

Пример:

Мой первый JavaScript

Нажмите на меня, чтобы отобразить дату и время

Попробуйте сами »

HTML тег <script>

Тег `<script>` используется для определения скрипта на стороне клиента (**JavaScript**). Элемент `<script>` или содержит операторы сценариев (скрипта), или указывает на внешний файл скрипта через атрибут `src`. Обычно **JavaScript** используется для манипуляции изображениями, проверки форм и динамичных изменений содержания. Чтобы выбрать **HTML** элемент, **JavaScript** очень часто использует метод `document.getElementById()`.

Этот пример **JavaScript** пишет "Hello JavaScript!" в **HTML** элементе с id="demo":

Пример:

```
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
```

Попробуйте сами »

Примечание: Вы можете узнать больше о **JavaScript** в Учебнику по JavaScript.

Наслаждайтесь JavaScript!

Вот несколько примеров того, что **JavaScript** может сделать:

JavaScript может менять содержание **HTML**:

```
document.getElementById("demo").innerHTML = "Hello JavaScript!";
```

Попробуйте сами »

JavaScript может менять **HTML** стили:

```
document.getElementById("demo").style.fontSize = "25px";
document.getElementById("demo").style.color = "red";
document.getElementById("demo").style.backgroundColor = "yellow";
```

Попробуйте сами »

JavaScript может менять атрибуты **HTML**:

document.getElementById("image").src = "picture.gif";

Результат:

Свет включить Свет выключить

Попробуйте сами »

HTML тег <noscript>

Тег `<noscript>` используется для предоставления альтернативного содержания для пользователей, которые выключили скрипты в своём веб-браузере или имеют веб-браузер, который не поддерживает скрипты на стороне клиента:

Пример:

```
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>

<noscript>Sorry, your browser does not support JavaScript!</noscript>
```

Попробуйте сами »

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 »

HTML теги скрипта

Тег	Описание
`<script>`	Определяет скрипт на стороне клиента
`<noscript>`	Определяет альтернативное содержание для пользователей, которые не поддерживают скрипты на стороне клиента

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

Prev Next

HTML. Стили

Prev Next

Стили в HTML. Как создать каскадную таблицу стилей (CSS) на веб-сайте?

Пример:

I am Red

I am Blue

I am Big

Попробуйте сами »

HTML Атрибут style

Установка стиля **HTML** элемента может быть выполнена с помощью атрибута `style`. Атрибут `style` в **HTML** имеет такой **синтаксис**:

Пример:

```
<tagname style="property:value;">
```

где *tagname* - название тега, *property* - это свойство, *value* - это значение.

Примечание: Подробнее о **CSS** вы узнаете далее в этом Учебнике по CSS.

HTML Background Color - Цвет фона

Свойство `background-color` определяет цвет фона для элемента **HTML**. В этом примере для страницы установлен фон (background color): powderblue (голубой цвет):

Пример:

```
<body style="background-color:powderblue;">

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
```

Попробуйте сами »

HTML Color - Цвет текста

Свойство `color` определяет цвет текста для элемента **HTML**:

Пример:

```
<h1 style="color:blue;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>
```

Попробуйте сами »

HTML Fonts - Шрифты

Свойство `font-family` определяет семейство шрифта, который будет использоваться для элемента **HTML**:

Пример:

```
<h1 style="font-family:verdana;">This is a heading</h1>
<p style="font-family:courier;">This is a paragraph.</p>
```

Попробуйте сам »

HTML Font-size - Размер шрифта

Свойство `font-size` определяет размер текста для элемента **HTML**:

Пример:

```
<h1 style="font-size:300%;">This is a heading</h1>
<p style="font-size:160%;">This is a paragraph.</p>
```

Попробуйте сами »

HTML Text-align - Выравнивание текста

Свойство `text-align` определяет горизонтальное выравнивание текста для элемента **HTML**:

Пример:

```
<h1 style="text-align:center;">Centered Heading</h1>
<p style="text-align:center;">Centered paragraph.</p>
```

Попробуйте сами »

Резюме раздела

- Используйте атрибут `style` для стилизации **HTML-элементов**
- Используйте `background-color` для присвоения цвета фону
- Используйте `color` для присвоения цвета тексту
- Используйте `font-family` для присвоения семейства шрифтов тексту
- Используйте `font-size` для присвоения размера тексту
- Используйте `text-align` для горизонтального выравнивания текста

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 » Упражнение 5 » Упражнение 6 »

Справочник HTML тегов

Справочник HTML тегов *W3Schools* содержит дополнительную информацию про элементы **HTML** и их атрибуты.

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

□ Prev Next □

Вы изучили HTML. Теперь что изучать?

□ Prev Next □

HTML Резюме

Этот учебник научил вас использовать **HTML** для создания собственного веб-сайта.

HTML является универсальным языком разметки для Web. **HTML** позволяет форматировать текст, добавлять графику, создавать ссылки, формы ввода, фреймы, таблицы и др., а также сохранять их в текстовом файле, который может читать и отображать любой веб-браузер.

Для получения более подробной и дополнительной информации об **HTML**, пожалуйста, обратитесь к HTML примерам и HTML справочнику.

Теперь вы знаете HTML, что дальше?

Что необходимо изучать после изучения **HTML**?

Как правило, после изучения **HTML** изучают также такие технологии, как **CSS** и **JavaScript** (для фронтенд-разработчиков).

Изучение CSS

CSS позволяет стилизировать HTML-страницы.

CSS предоставляет полный контроль над макетом, не изменяя содержания документа.

Чтобы узнать больше о **CSS**, посетите CSS учебник на нашем сайте.

Изучение JavaScript

JavaScript делает сайт более динамичным. Динамичный веб-сайт может реагировать на события и позволяет взаимодействовать с пользователем.

JavaScript является на сегодняшний день самым популярным скриптовым языком в Интернете (на стороне клиента), и он работает во всех основных браузерах.

Если вы хотите узнать больше про **JavaScript**, пожалуйста, посетите JavaScript учебник на нашем сайте.

Публикация веб-сайта

Чтобы сделать свой веб-сайт доступным для каждого в мире пользователя Интернет, его необходимо опубликовать в сети.

Для этого у вас есть два варианта:

- Использование поставщика услуг Интернета (сервис-провайдера);
- Собственный хостинг веб-сайта.

Использование услуг сервис-провайдера Интернет

Интернет-провайдер (ISP - Internet Service Provider) - это компания, которая предоставляет услуги доступа к Интернету и использования Интернета.

Интернет-услуги, которые обычно предоставляются Интернет-провайдерами, включают доступ к Интернету, Интернет-транзит, регистрацию доменного имени, веб-хостинг, услугу Usenet и колокацию.

Использование почтавщика услуг Интернета (ISP - Интернет сервис-провайдера) является самым распространённым вариантом.

Преимущества:

- **Скорость соединения** - Провайдеры имеют очень скоростное соединение с Интернетом;
- **Мощное оборудование** - Провайдеры имеют мощные веб-серверы, которые могут совместно использовать несколько клиентов. Можно также ожидать эффективной сбалансированной нагрузки и необходимых серверов резервного копирования;
- **Безопасность и стабильность** - Провайдеры - это специалисты веб-хостинга. Вы получите намного больше свободного времени, самые новые исправления программного обеспечения и лучшую защиту от вирусов.

Что необходимо учитывать?

- **24-часовая поддержка** - Провайдер должен предложить 24-часовую поддержку. Бесплатный телефон также может быть жизненно важен;
- **Ежедневный бекап (резервное копирование)** - Провайдер должен выполнять ежедневную процедуру резервного копирования;
- **Объём трафика** - Проверьте ограничения объёма трафика поставщика услуг Интернета (не прекращайте платить деньги за неожиданно высокий трафик);
- **Ограничения пропускной способности или содержания** - Проверьте пропускную способность провайдера и ограничения содержания (можно ли публиковать фотографии, видео или аудио?);
- **Возможности электронной почты** - Убедитесь, что провайдер поддерживает необходимые возможности электронной почты;
- **Доступ к базе данных** - Убедитесь, что провайдер поддерживает необходимый доступ к базе данных.

Хостинг собственного сайта на своём сервере

Хостинг собственного веб-сайта на вашем собственном сервере также является возможным.

Что необходимо учитывать?

- **Затраты на оборудование** - Чтобы запустить "реальный" веб-сайт, вы должны приобрести мощное серверное оборудование (обычный домашний ПК не будет выполнять необходимую работу). Вам также понадобится

постоянное (24/7) высокоскоростное Интернет-соединение;

- **Затраты на программное обеспечение** - Серверные лицензии часто превышают стоимость клиентских лицензий. Серверные лицензии также могут иметь ограничения на количество пользователей;
- **Затраты на оплату труда** - Не ожидайте низких трудовых затрат. Вы должны установить собственное аппаратное и программное обеспечение. Вы также будете иметь дело с ошибками и вирусами, и должны содержать ваш сервер постоянно работающим.

□ Prev Next □

HTML Символы

□ Prev Next □

HTML Символьные элементы

Некоторые символьные **HTML элементы** были описаны в предыдущем разделе. Много математических, технических и валютных символов не существуют на обычной клавиатуре. Чтобы добавить такие символы на HTML-странице, можно использовать название HTML-объекта. Если названия объекта не существует, можно использовать номер объекта, его десятичное или шестнадцатеричное отображение.

Пример:

```
<p>I will display &euro;</p>
<p>I will display &#8364;</p>
<p>I will display &#x20AC;</p>
```

Отобразится как:

I will display €
I will display €
I will display €
Попробуйте сами »

Некоторые математические символы, которые поддерживаются HTML

Символ	Номер	Объект	Описание
∀	∀	∀	ДЛЯ ВСЕХ
∂	∂	∂	ЧАСТИЧНЫЙ ДИФФЕРЕНЦИАЛ
∃	∃	∃	СУЩЕСТВУЮЩЕЕ
∅	∅	∅	ПУСТЫЕ НАБОРЫ
∇	∇	∇	NABLA
∈	∈	∈	ЭЛЕМЕНТ
∉	∉	∉	НЕ ЭЛЕМЕНТ
∋	∋	∋	СОДЕРЖИТ В КАЧЕСТВЕ ЧЛЕНА
∏	∏	∏	N-ARY ПРОДУКТ
∑	∑	∑	N-ARY СУММА

Полный математический справочник

Некоторые греческие буквы, которые поддерживаются в HTML

Символ	Номер	Объект	Описание
Α	Α	Α	GREEK CAPITAL LETTER ALPHA
Β	Β	Β	GREEK CAPITAL LETTER BETA
Γ	Γ	Γ	GREEK CAPITAL LETTER GAMMA
Δ	Δ	Δ	GREEK CAPITAL LETTER DELTA
Ε	Ε	Ε	GREEK CAPITAL LETTER EPSILON
Ζ	Ζ	Ζ	GREEK CAPITAL LETTER ZETA

Полный справочник греческих символов и букв

Некоторые другие объекты, которые поддерживаются в HTML

Символ	Номер	Объект	Описание
©	©	©	ЗНАК АВТОРСКОГО ПРАВА
®	®	®	ЗАРЕГИСТРИРОВАННЫЙ ЗНАК
€	€	€	ЗНАК ЕВРО
™	™	™	ТОРГОВАЯ МАРКА
←	←	←	СТРЕЛКА ВЛЕВО
↑	↑	↑	СТРЕЛКА ВВЕРХ
→	→	→	СТРЕЛКА ВПРАВО
↓	↓	↓	СТРЕЛКА ВНИЗ
♠	♠	♠	ЧЁРНАЯ ПИКА
♣	♣	♣	ЧЁРНАЯ КРЕСТЯ
♥	♥	♥	ЧЁРНАЯ ЧИРВА
♦	♦	♦	ЧЁРНАЯ БУБНА

Полный справочник валют

Полный справочник стрелок

Полный справочник символов

□ Prev Next □

HTML Таблицы

□ Prev Next □

Что такое таблица?

Таблица - это структурированный набор данных, состоящий из строк и столбцов (табличных данных). Таблицы позволяют быстро и легко посмотреть значения, показывающие некоторую взаимосвязь между различными типами данных, например - человек и его возраст, или расписание уроков в школе и т.д.

Как добавить таблицу на веб-странице?

HTML Таблица. Пример

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

Попробуйте сами »

Определение таблицы в HTML

Таблица в **HTML** определяется тегом `<table>`. Каждый рядок таблицы определяется тегом `<tr>`. Заголовок таблицы определяется тегом `<th>`. По умолчанию заголовки таблиц выделены жирным шрифтом и центрированы. Ячейка с данными в таблице (data/cell) определяется тегом `<td>`.

Пример:

```
<table style="width:100%">
  <tr>
```

```
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Попробуйте сами »

Примечание: Элементы `<td>` являются контейнерами данных таблицы. Они могут содержать всякие элементы **HTML**; текст, изображения, списки, другие таблицы и пр.

HTML таблица - добавление границы таблицы (border)

Если вы не укажете границу для таблицы, она будет отображаться без границ. Граница устанавливается с помощью **CSS** свойства `border`.

Пример:

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

Попробуйте сами »

Примечание: Не забывайте определить границы как для таблицы, так и для ячеек таблицы.

HTML таблица - свёрнутые границы (border-collapse)

Если вы хотите, чтобы рамки таблицы отображались в одну границу (одинарной линией), добавьте **CSS**-свойство `border-collapse`.

Пример:

```
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
```

Попробуйте сами »

HTML таблица - добавление внутреннего отступа в ячейках (padding)

Внутренний отступ ячейки определяет пространство между содержанием ячейки и её границами. Если вы не указываете внутренний отступ, ячейки таблицы будут отображаться без отступа.

Чтобы установить внутренний отступ, используйте **CSS** свойство `padding`:

Пример:

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

Попробуйте сам »

HTML таблица - выравнивание заголовков слева (text-align)

По умолчанию заголовки таблиц выделены жирным шрифтом и центрованы. Для выравнивания заголовков таблиц используйте **CSS** свойство `text-align`:

Пример:

```
th {
  text-align: left;
}
```

Попробуйте сами »

HTML таблица - добавление интервала между границами (border-spacing)

Интервал между границами означает пространство между ячейками. Чтобы установить интервал между ячейками, используйте **CSS** свойство `border-spacing`:

Пример:

```
table {
  border-spacing: 5px;
}
```

Попробуйте сами »

Примечание: Если таблица имеет `border-collapse`, то установка `border-spacing` никак не влияет.

HTML таблица - ячейки, которые охватывают много столбцов (colspan)

Чтобы сделать ячейку, которая охватывает больше чем один столбец, используйте атрибут `colspan`:

Пример:

```
<table style="width:100%">
  <tr>
    <th>Name</th>
    <th colspan="2">Telephone</th>
  </tr>
  <tr>
    <td>Bill Gates</td>
    <td>55577854</td>
    <td>55577855</td>
  </tr>
</table>
```

Попробуйте сами »

HTML таблица - ячейки, которые охватывают много строк (rowspan)

Чтобы сделать ячейку, которая охватывает больше чем одну строку, используйте атрибут `rowspan`:

Пример:

```
<table style="width:100%">
  <tr>
    <th>Name:</th>
    <td>Bill Gates</td>
  </tr>
  <tr>
    <th rowspan="2">Telephone:</th>
    <td>55577854</td>
  </tr>
  <tr>
    <td>55577855</td>
  </tr>
</table>
```

Попробуйте сами »

HTML таблица - добавление подписи (caption)

Чтобы добавить подпись к таблице, используйте тег `<caption>`:

Пример:

```
<table style="width:100%">
  <caption>Monthly savings</caption>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$50</td>
  </tr>
</table>
```

Попробуйте сами »

Примечание: Тег `<caption>` должен быть вставлен сразу после тега `<table>`

Особый стиль для одной таблицы (id)

Чтобы определить особый стиль для отдельной таблицы, добавьте атрибут `id` (идентификатор) к таблице:

Пример:

```
<table id="t01">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Теперь вы можете определить особый стиль для этой таблицы:

```
table#t01 {
  width: 100%;
  background-color: #f1f1c1;
}
```

Попробуйте сами »

И добавить новые стили для этой таблицы:

```
table#t01 tr:nth-child(even) {
  background-color: #eee;
}
table#t01 tr:nth-child(odd) {
  background-color: #fff;
}
table#t01 th {
  color: white;
  background-color: black;
}
```

Попробуйте сами »

Резюме раздела

- Используйте HTML элемент `<table>` для определения таблицы
- Используйте HTML элемент `<tr>` для определения строки таблицы
- Используйте HTML элемент `<td>` для определения ячейки с данными таблицы
- Используйте HTML элемент `<th>` чтобы определить заголовок таблицы

- Используйте HTML элемент `<caption>` чтобы определить подпись к таблице
- Используйте CSS свойство `border` для определения границ таблицы
- Используйте CSS свойство `border-collapse` для сворачивания границ таблицы
- Используйте CSS свойство `padding` для добавления внутреннего отступа в таблице
- Используйте CSS свойство `text-align` для выравнивания текста в ячейке
- Используйте CSS свойство `border-spacing` для установки интервала между ячейками
- Используйте атрибут `colspan`, чтобы сделать ячейки, которые охватывают несколько столбцов
- Используйте атрибут `rowspan`, чтобы сделать ячейки, которые охватывают несколько строк
- Используйте атрибут `id` для определения особого стиля для одной таблицы

Проверьте себя с помощью упражнений!

Упражнение 1 » Упражнение 2 » Упражнение 3 » Упражнение 4 » Упражнение 5 » Упражнение 6 »

HTML теги таблицы

Тег	Описание
`<table>`	Определяет таблицу
`<th>`	Определяет ячейку заголовка таблицы
`<tr>`	Определяет строку в таблице
`<td>`	Определяет ячейку в таблице
`<caption>`	Определяет подпись к таблице
`<colgroup>`	Определяет группу с одного или нескольких столбцов в таблице для форматирования
`<col>`	Указывает свойства столбцов для каждого столбца в элементе `<colgroup>`
`<thead>`	Группирует содержание заголовка (head) в таблице
`<tbody>`	Группирует содержание тела (body) в таблице
`<tfoot>`	Группирует содержание нижнего колонтитула (footer) в таблице

Примечание: Для получения полного списка всех доступных **HTML** тегов, посетите Справочник **HTML** тегов.

Prev Next

HTML Старт

Перед вами сайт, на котором можно абсолютно бесплатно пройти курс обучения языку *HTML*. Основная часть содержания на сайте - это перевод на русский язык материалов сайта W3Schools.com - самого большого и самого известного ресурса для веб-разработчиков, который адаптирован для русскоязычного пользователя. Также на сайте вы можете найти дополнительные материалы данной тематики.

Из самого начала весь материал разделён на небольшие уроки, в каждом из которых максимально подробно изложен материал по каждой теме. Для новичков рекомендуется проходить уроки в том порядке, в котором они изложены. Но если вы уже раньше сталкивались с языком *HTML* (например, изучали его в школе) или даже пытались его выучить самостоятельно с помощью Интернета - смотрели видеоуроки или читали учебники - вам также будут полезны данные уроки. Здесь вы сможете найти именно те материалы, которые вам необходимы.

Уроки изложены максимально коротко, просто и понятно даже для новичков, которые раньше никогда не изучали язык *HTML* и не занимались написанием сайтов с помощью программ. Язык *HTML* не является языком программирования, на котором можно было бы написать какие-то программы. *HTML* является языком гипертекстовой разметки (*HTML - HyperText Markup Language*), с помощью которого пишется код, который считывается специальными программами - браузерами. И уже программы-браузеры отображают веб-страницы так, как это более привычно обычным пользователям Интернета.

На сайте представлены уроки как для новичков, так и для более опытных пользователей, которые уже раньше, возможно, изучали язык *HTML* или сталкивались с ним в процессе изучения веб-технологий и языков веб-программирования.

В основном курсе по *HTML* вы сможете получить знания, необходимые для освоения и понимания написания кода *HTML* и сможете писать самые простые веб-страницы и статические сайты. Но если вы хотите научиться создавать полноценные динамические веб-сайты, этих знаний, конечно же, будет недостаточно. Но в любом случае, фундамент уже будет заложен и базовые знания будут у вас в голове.

Чем эти уроки по изучению языка HTML отличаются от других?

В сети Интернет есть огромное количество сайтов, книг и видеоуроков, по которым можно изучать язык разметки *HTML* вместе со стилями - *CSS*. Но большинство из них в русскоязычном сегменте Интернета написаны теми, кто просто банально пытается заработать деньги - на книгах, сайтах или видео на Youtube, часто просто впихивая на свои сайты или видео абсолютно никому ненужную рекламу. Кроме того, сайты или книги, которые были созданы ещё каких-то 2-3 года назад, и видеоуроки на YouTube, очень быстро устаревают и редко поддерживаются своими авторами в течение длительного времени. Соответственно, обучающие материалы по веб-технологиям, в т.ч. даже по *HTML / CSS* также очень быстро теряют свою актуальность. То, что было стандартом ещё каких-то 3-5 лет назад, сейчас уже просто становится неактуальным и не востребованным. Всё меняется, особенно в сфере web-технологий.

На данном же сайте, который, кстати говоря, сделан фактически с помощью лишь *HTML/CSS*, материалы взяты непосредственно из первоисточников - сайтов, которые являются основными по изучению данных технологий на просторах Интернета - онлайн школы w3schools.com, **W3C** - (*Консорциума Всемирной паутины*) - www.w3.org, **WHATWG** *Живого стандарта* - https://whatwg.org/, MDN.Mozilla. Именно с этих сайтов и берётся вся информация по официально утверждённым стандартам и веб-технологиям.

На данном адаптированном для русскоязычного пользователя сайте размещён материал, который является наиболее актуальным на момент создания сайта, при этом материалы выкладываются как в оригинальном оформлении (как на сайтах - первоисточниках, которые являются англоязычными, но с русским переводом), так и с дополнительными пояснениями и примерами от автора данного сайта. Материалы в уроках выкладываются по принципу от простого к сложному, что позволяет изучать *HTML/CSS* даже новичкам без начальных знаний веб-технологий или веб-программирования. Данные материалы, если их изучать постепенно и согласно изложенному порядку, являются доступными для понимания любому новичку, даже детям младшего школьного возраста, которые умеют лишь более-менее нормально читать, пользоваться компьютером и Интернетом и при этом имеют желание учиться.

Для того, чтобы начать изучать основы веб-вёрстки, в частности, технологию **HTML5**, не нужно иметь высшее образование или какие-то специальные знания по программированию или знать высшую математику. Начинать учиться создавать сайты можно как в младшем школьном возрасте, так и наоборот, в достаточно зрелом возрасте, когда уже перевалило за 40. Конечно же, для того, чтобы лучше усваивать данный материал, желательно ещё знать и английский язык - хотя бы на уровне средней школы, - это очень пригодится и поможет в запоминании и освоении материалов.

Начинать изучать веб-технологии необходимо именно с **HTML/CSS**, т.к. именно они являются основой всех сайтов Интернета. Далее, когда вы уже будете хорошо знать и уметь использовать **HTML/CSS**, стоит начинать изучать язык программирования, который используется на стороне клиента (браузером) - **JavaScript**. Знания и владение этими тремя веб-технологиями составляют фронтэнд (front end) - часть веб-разработки, которая служит для взаимодействия между пользователем и серверной частью, которая, в свою очередь, обслуживается веб-разработчиками по бэкэнду (back end). Для бэкэнда необходимы знания таких языков программирования и технологий, как *PHP, SQL, Node.js, Python, Ruby* и др. (в зависимости от проектов).

В некоторых разделах на сайте изложены материалы для более продвинутых пользователей, которые уже освоили основы *HTML5/CSS3* и желают продолжить учиться веб-технологиям, веб-дизайну и веб-программированию. В т.ч., это основы языков программирования *JavaScript*, *PHP*, *SQL*, *Python*, а также фреймворки *jQuery*, *Bootstrap* и др.

Также на данном сайте вы сможете найти примеры готовых решений для создания веб-сайтов, которые в дальнейшем можно будет использовать в своих проектах.

□ Home Next □

С помощью языка *HTML* вы можете создать свой собственный веб-сайт.

Этот учебник от школы *w3schools.com* поможет вам научиться основам языка разметки веб-страниц *HTML*.

HTML легко выучить - вы будете просто наслаждаться обучением, с каждым уроком получая новый результат!

Примеры в каждой главе

Этот учебник от *w3schools.com* содержит сотни примеров **HTML**.

С онлайн-редактором **HTML** на сайте *w3schools.com* вы можете редактировать **HTML код** и, нажав кнопку, сразу же посмотреть результат.

Пример:

```
<!DOCTYPE html>
<html>
<title>Посібник з HTML</title>
<body>
```

```
<h1>Це заголовок</h1>
<p>Це параграф.</p>
<p>Це параграф.</p>

</body>
</html>
```

Попробуй сам »

Нажмите кнопку "Попробуй сам", чтобы узнать, как она работает.

Начните изучать HTML сейчас!

Примеры HTML

В конце **HTML** учебника можно найти более 200 примеров.

С онлайн-редактором *w3schools.com* вы можете самостоятельно редактировать и проверять каждый пример.

Перейдите к HTML-примерам!

HTML-упражнения и викторина

Проверьте свои навыки **HTML** в *W3Schools*!

Запустить HTML-упражнения!

Запустить HTML-викторину!

HTML Упражнения

Этот учебник HTML содержит около 100 упражнений HTML.

Проверьте себя с помощью упражнений

Упражнение:

Добавьте «всплывающую подсказку» к пункту ниже с текстом "About W3Schools".

<p [ ]="About W3Schools">W3Schools is a web developer's site.</p>

Отправить ответ »

Начать упражнение

HTML Викторина

Проверьте свои HTML навыки с HTML-викториной на сайте W3Schools!

Начать HTML Викторину!

HTML справочники

В *W3Schools* вы найдёте полные справочники по тегам, атрибутам, событиям, названиям цветов, сущностям, наборам символов, кодировке URL, кодам языков, HTTP и др.

Справочник тегов HTML

Сдать экзамен по знанию HTML - получить диплом!

□ Home Next □

HTML URL (унифицированный локатор ресурсов)

□ Prev Next □

HTML URL или веб-адрес

URL-адрес - это другое название для веб-адреса. URL-адрес может состоять из слов (w3schools.com) или адреса Интернет-протокола (IP-адреса) (192.68.20.50). Большинство людей вводят имя во время сёрфинга, потому что имена легче запоминать, чем числа.

URL - унифицированный локатор ресурсов

Веб-браузеры запрашивают страницы с веб-серверов с помощью URL-адреса. **Унифицированный информационный ресурс** (URL) используется для адреса документа (или других данных) в Интернете. Веб-адрес, например, https://www.w3schools.com/html/default.asp, придерживается этих правил:

scheme://prefix.domain:port/path/filename

Пояснения:

- **scheme** (схема) - определяет **тип** Интернет-службы (наиболее распространённым является **http** или **https**)
- **prefix** (префикс) - определяет **префикс** домена (по умолчанию для http является **www**)
- **domain** (домен) - определяет **название домена** Интернета (например, w3schools.com)
- **port** (порт) - определяет **номер порта** на хосте (по умолчанию для http - **80**)
- **path** (путь) - определяет **путь** на сервере (если пропущено: корневой каталог сайта)
- **filename** (название файла) - определяет **название документа** или ресурса

Общие схемы URL-адресов

В таблице ниже приведены некоторые общие схемы:

Схема	Коротко	Используется
http	Протокол передачи гипертекста	Общие веб-страницы. Не зашифровано
https	Безопасный протокол передачи гипертекста	Защищает веб-страницы. Зашифровано

ftp	Протокол передачи файлов	Загрузка или выгрузка файлов
file		Файл на вашем компьютере

URL кодировка

URL-адреса можно отправлять только лишь через Интернет с помощью набора символов ASCII. Если URL-адрес содержит символы вне набора ASCII, URL-адрес необходимо конвертировать.

Кодирование URL преобразует не-ASCII символы в формат, который может быть передан через Интернет.

Кодирование URL заменяет символы не-ASCII на "%", за которыми идуть шестнадцатеричные цифры.

URL-адреса не могут содержать пробелы. Кодирование URL обычно заменяет пробел знаком плюс (+) или %20.

Попробуйте сами

Hello Günter Отправить

Если нажать кнопку "Отправить", браузер будет кодировать URL-адрес во входной сигнал, прежде чем он будет отправлен на сервер. Страница на сервере отобразит полученный вход.

Попробуйте ввести другие данные и ещё раз нажмите "Отправить".

Примеры ASCII кодирования

Ваш браузер будет кодировать входные данные в соответствие набору символов, который используется на вашей странице. Типичный набор символов (по умолчанию) в **HTML5** - это **UTF-8**.

Символ	Из Windows-1252	Из UTF-8
€	%80	%E2%82%AC
£	%A3	%C2%A3
©	%A9	%C2%A9
®	%AE	%C2%AE
À	%C0	%C3%80
Á	%C1	%C3%81
Â	%C2	%C3%82
Ã	%C3	%C3%83
Ä	%C4	%C3%84
Å	%C5	%C3%85

Для просмотра более полной информации про все кодировки URL-адресов, посетите Справочник кодирования URL-адресов.

□ Prev Next □

HTML и XHTML

□ Prev Next □

HTML и XHTML

XHTML - это **HTML**, написанный как **XML**.

Что такое XHTML?

- **XHTML** означает "Расширенный язык гипертекстовой разметки" (E**X**tensible **H**yper**T**ext **M**arkup **L**anguage)

- **XHTML** почти идентичный **HTML**
- **XHTML** является более строгим, чем **HTML**
- **XHTML** - это **HTML**, определённый как программа **XML**
- **XHTML** поддерживается всеми основными браузерами

Почему XHTML?

Много страниц в Интернете содержат "плохой" **HTML** код. Этот **HTML** код работает в большинстве браузеров (даже если он не соответствует правилам **HTML**):

Пример:

```
<html>
<head>
  <title>This is bad HTML</title>

<body>
  <h1>Bad HTML
  <p>This is a paragraph
</body>
```

Сегодняшний рынок состоит из разных технологий браузера. Некоторые браузеры работают на компьютерах, а некоторые веб-браузеры работают на мобильных телефонах или других небольших устройствах. Меньшие устройства часто не имеют ресурсов или мощности для интерпретации "плохой" разметки.

XML - это язык разметки, где документы должны быть правильно обозначены (быть "хорошо сформированными"). **XHTML** разрабатывался путём объединения сильных сторон **HTML** и **XML**. **XHTML** является **HTML** переработанный как **XML**. Если вы хотите изучить **XML**, пожалуйста, посетите и почитайте **XML-учебник** на нашем сайте.

Наиболее важные отличия XHTML от HTML:

Структура XHTML документа

- XHTML DOCTYPE **является обязательным**
- Атрибут xmlns в `<html>` **является обязательным**
- `<html>`, `<head>`, `<title>` и `<body>` **являются обязательными**

XHTML элементы

- Элементы XHTML должны быть правильно вложены
- Элементи XHTML всегда должны быть закрыты
- Элементы XHTML должны быть в нижнем регистре
- Документы XHTML должны иметь один корневой элемент

XHTML атрибуты

- Имена атрибутов должны иметь нижний регистр
- Необходимо указать значения атрибутов
- Минимизация атрибутов запрещена

<!DOCTYPE ....> обязателен

XHTML документ должен иметь объявление XHTML DOCTYPE.

Элементы `<html>`, `<head>`, `<title>` і `<body>` также должны присутствовать, а атрибут xmlns в `<html>` должен указывать пространство имён xml для документа.

Полный список всех **XHTML Doctypes** находится в Справочнике HTML тегов.

В этом примере показан документ **XHTML** с минимально необходимыми тегами:

Пример:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>Title of document</title>
</head>

<body>
  some content
</body>

</html>
```

XHTML элементы должны быть правильно вложены

В **HTML** некоторые элементы могут быть **неправильно** вложены друг в друга, например:

```
<b><i>This text is bold and italic</b></i>
```

В **XHTML** все элементы должны быть **правильно** вложены друг в друга, например:

```
<b><i>This text is bold and italic</i></b>
```

XHTML элементы должны быть всегда закрыты

Это неправильно:

```
<p>This is a paragraph
<p>This is another paragraph
```

Это правильно:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

Пустые элементы также должны быть закрыты

Это неправильно:

```
A break: <br>
A horizontal rule: <hr>
An image: <img src="happy.gif" alt="Happy face">
```

Это правильно:

```
A break: <br />
A horizontal rule: <hr />
An image: <img src="happy.gif" alt="Happy face" />
```

XHTML элементы должны быть в нижнем регистре

Это неправильно:

```
<BODY>
<P>This is a paragraph</P>
</BODY>
```

Это правильно:

```
<body>
<p>This is a paragraph</p>
</body>
```

Названия XHTML атрибутов должны быть в нижнем регистре

Это неправильно:

```
<table WIDTH="100%">
```

Это правильно:

```
<table width="100%">
```

Значения атрибутов должны быть взяты в кавычки

Это неправильно:

```
<table width=100%>
```

Это правильно:

```
<table width="100%">
```

Минимизация атрибутов запрещена

Это неправильно:

```
<input type="checkbox" name="vehicle" value="car" checked />
```

Это правильно:

```
<input type="checkbox" name="vehicle" value="car" checked="checked" />
```

Это неправильно:

```
<input type="text" name="lastname" disabled />
```

Это правильно:

```
<input type="text" name="lastname" disabled="disabled" />
```

Как конвертировать с HTML в XHTML

1. Добавьте XHTML `<!DOCTYPE>` к первой строке каждой страницы
2. Добавьте атрибут `xmlns` к html элементу каждой страницы
3. Сделайте все названия элементов буквами нижнего регистра
4. Закройте все пустые элементы
5. Сделайте все названия атрибутов буквами нижнего регистра
6. Возьмите в кавычки все значения атрибутов

Проверить HTML с помощью валидатора W3C

Введите веб-адрес в поле ниже:

https://html5beginner.github.io/html/html_validate.html

Validate the page

□ Prev Next □

HTML5 YouTube видео

□ Prev Next □

Самым простым способом воспроизведения видео в HTML является использование сайта YouTube.

Что такое YouTube и для чего он нужен?

YouTube (от англ. you «ты, вы» + tube «труба» = «телик» жарг. «телевизор») - самый большой и самый популярный мировой видеохостинг, который предоставляет услуги размещения видеоматериалов. Основан 14 февраля 2005 тремя работниками PayPal: Чадом Герли, Стивеном Чени и Джаведом Каримом. Является подразделением компании Google. По состоянию на август 2019 года YouTube является вторым по посещаемости сайтом в Интернете (по данным компании Alexa Internet).

Пользователи могут добавлять, просматривать и комментировать те или иные видеозаписи. Благодаря простоте и удобству использования YouTube стал одним из самых популярных видеохостингов. Служба содержит как профессиональные, так и любительские видеозаписи, включая видеоблоги.

Любой человек может бесплатно зарегистрироваться на сайте Google (создать собственный email), а потом перейти на сайт YouTube и открыть собственный видео-канал. Затем все видео с YouTube можно встраивать на любые другие веб-страницы, используя обычный html-код.

Борьба с видеоформатами?

Ранее в этом учебном пособии вы видели, что вам придется конвертировать видео в различные форматы, чтобы сделать их воспроизводимым во всех браузерах.

Но конвертирование видеофайлов в различные форматы может быть довольно сложным и занимать много времени.

Более простое решение - это встроить видео с YouTube для воспроизведения на вашей веб-странице.

YouTube видео идентификатор - Id

YouTube будет отображать идентификатор (например, tgbNymZ7vqY) при сохранении (или воспроизведение) видео.

Вы можете использовать этот идентификатор и ссылаться на свое видео в HTML-коде.

Воспроизведение видео с YouTube на HTML-странице

Чтобы воспроизвести видео с YouTube на веб-странице, выполните приведенные ниже действия:

- Загрузите видео на YouTube
- Обратите внимание на идентификатор видео - Id
- Определите элемент `<iframe>` на вашей веб-странице
- Атрибут `src` указывает URL-адрес видео
- Используйте атрибуты `width` (ширина) и `height` (высота), чтобы определить размеры плеера
- Добавьте к URL другие необходимые параметры (см. ниже)

Пример - использование iFrame (рекомендуется)

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>
```

Попробуйте сами »

YouTube Autoplay - автоматическое воспроизведение

Вы можете сделать автоматическое воспроизведение видео, когда пользователь посещает страницу, добавив простой параметр к URL-адресу YouTube.

Примечание: Будьте внимательны при принятии решения об автоматическом воспроизведении видео. Автоматический запуск видео может раздражать вашего посетителя и в конечном итоге причинить больше вреда, чем пользы, так как пользователи могут сразу покинуть сайт.

Значение 0 (по умолчанию): видео не будет воспроизводиться автоматически во время загрузки плеера.

Значение 1: видео будет воспроизводиться автоматически во время загрузки плеера.

YouTube - Autoplay

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1">
</iframe>
```

Попробуйте сами »

YouTube Playlist - плейлист

Разделённый запятыми список видео для воспроизведения (кроме исходного URL-адреса).

YouTube Loop - повторение воспроизведения видео

Значение 0 (по умолчанию): видео будет воспроизводиться лишь один раз.

Значение 1: воспроизведение видео будет приостановлено после окончания.

YouTube - Loop

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7vqY&loop=1">
</iframe>
```

Попробуйте сами »

YouTube Controls - панель управления

Значение 0: элементы управления плеера не отображаются.

Значение 1 (по умолчанию): элементы управления плеера отображаются.

YouTube - Controls

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">
</iframe>
```

Попробуйте сами »

YouTube - использование <object> или <embed>

Примечание: YouTube `<object>` и `<embed>` считаются устаревшими с января 2015 года. Вы должны изменить способ добавления вашего видео, используя `<iframe>` вместо этого.

Пример - использование <object> (устаревшее)

```
<object width="420" height="315"
data="https://www.youtube.com/embed/tgbNymZ7vqY">
</object>
```

Попробуйте сами »

Пример - использование <embed> (устаревшее)

```
<embed width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
```

Попробуйте сами »

□ Prev Next □

HTML атрибуты. Полный справочник

□ Prev Next □

HTML Все Атрибуты

В данной таблице собраны **все атрибуты**, используемые в **HTML5** на данный момент.

Атрибуты	Принадлежность	Описание
accept	<input>	Указывает типы файлов, которые принимает сервер (только для type="file")
accept-charset	<form>	Указывает кодировки символов, которые будут использоваться для отправки формы
accesskey	Глобальные Атрибуты	Указывает сочетание клавиш для активации / фокусировки элемента
action	<form>	Указывает, куда отправлять данные формы при отправке формы
align	Не поддерживается в HTML5.	Задает выравнивание в соответствии с окружающими элементами. Используйте CSS
alt	<area>, <img>, <input>	Задает альтернативный текст, если исходный элемент не отображается
async	<script>	Указывает, что скрипт выполняется асинхронно (только для внешних скриптов)
autocomplete	<form>, <input>	Указывает, будет ли в элементе <form> или <input>, должен иметь автозаполнения enabled
autofocus	<button>, <input>, <select>, <textarea>	Указывает, что элемент должен автоматически получать фокус при загрузке страницы
autoplay	<audio>, <video>	Указывает, что воспроизведение аудио/видео начнется, как только оно будет готово
bgcolor	Не поддерживается в HTML 5.	Задает цвет фона элемента. Используйте CSS
border	Не поддерживается в HTML 5.	Задает ширину границы элемента. Используйте CSS
charset	<meta>, <script>	Указывает кодировку символов
checked	<input>	Указывает, что элемент <input> должен быть предварительно выбран при загрузке страницы (для type="checkbox" или type="radio")
cite	<blockquote>, <del>, <ins>, <q>	Задает URL, который объясняет цитату / удаленный / вставленный текст
class	Глобальные Атрибуты	Задает одно или несколько имен классов для элемента (ссылается на класс в таблице стилей)
color	Не поддерживается в HTML 5.	Задает цвет текста элемента. Используйте CSS
cols	<textarea>	Задает видимую ширину текстовой области
colspan	<td>, <th>	Указывает количество столбцов, которое должна охватывать ячейка таблицы
content	<meta>	Дает значение, связанное с http-equiv или атрибутом name
contenteditable	Глобальные Атрибуты	Указывает, является ли содержимое элемента редактируемым или нет
contextmenu	Глобальные Атрибуты	Задает контекстное меню для элемента. Контекстное меню появляется, когда пользователь щелкает правой кнопкой мыши на элементе
controls	<audio>, <video>	Указывает, что должны отображаться элементы управления аудио/видео (например, кнопка воспроизведения / паузы и т. д)
coords	<area>	Задает координаты области
data	<object>	Задает URL ресурса, используемого объектом
data-*	Глобальные Атрибуты	Используется для хранения личных данных пользователя на странице или в приложении
datetime	<del>, <ins>, <time>	Указывает дату и время
default	<track>	Указывает, что трек должен быть включен, если предпочтения пользователя не указывают, что другой трек будет более подходящим
defer	<script>	Указывает, что сценарий выполняется после завершения синтаксического анализа страницы (только для внешних сценариев)
dir	Глобальные Атрибуты	Задает направление текста для содержимого элемента

dirname	<input>, <textarea>	Указывает, что направление текста будет отправлено
disabled	<button>, <fieldset>, <input>, <optgroup>, <option>, <select>, <textarea>	Указывает, что указанный элементы элемент/группа должны быть отключены
download	<a>, <area>	Указывает, что целевой объект будет загружен при нажатии пользователем гиперссылки
draggable	Глобальные Атрибуты	Указывает, является ли элемент перетаскиваемым или нет
dropzone	Глобальные Атрибуты	Указывает, копируются, перемещаются или связываются перетаскиваемые данные
enctype	<form>	Указывает, как данные формы должны быть закодированы при отправке на сервер (только для method="post")
for	<label>, <output>	Определяет форму элемента(ов) метки/расчет обязан
form	<button>, <fieldset>, <input>, <label>, <meter>, <object>, <output>, <select>, <textarea>	Задает имя формы элемент принадлежит
formaction	<button>, <input>	Указывает, куда отправлять данные формы при отправке. Только для type="submit"
headers	<td>, <th>	Задает одну или несколько ячеек заголовков, с которыми связана ячейка
height	<canvas>, <embed>, <iframe>, <img>, <input>, <object>, <video>	Задает высоту элемента
hidden	Глобальные Атрибуты	Указывает, что элемент еще не релевантен или больше не имеет значения
high	<meter>	Задает диапазон, который считается высоким значением
href	<a>, <area>, <base>, <link>	Указывает URL страницы, на которую переходит ссылка
hreflang	<a>, <area>, <link>	Указывает язык связанного документа
http-equiv	<meta>	Предоставляет заголовок HTTP для информации / значения атрибута содержания
id	Глобальные Атрибуты	Задает уникальный идентификатор элемента
ismap	<img>	Задает изображение в качестве серверной карты изображений
kind	<track>	Указывает тип текстовой дорожки
label	<track>, <option>, <optgroup>	Задает заголовок текстовой дорожки
lang	Глобальные Атрибуты	Задает язык содержимого элемента
list	<input>	Относится к элементу <datalist>, который содержит предопределенные параметры для элемента <input>
loop	<audio>, <video>	Указывает, что аудио / видео будет начинаться снова, каждый раз, когда он будет завершено
low	<meter>	Определяет диапазон, который считается низким значением
manifest	<html>	Указывает расположение манифеста кэша документа
max	<input>, <meter>, <progress>	Задает максимальное значение
maxlength	<input>, <textarea>	Задает максимальное количество символов, разрешенных в элементе
media	<a>, <area>, <link>, <source>, <style>	Указывает, для какого носителя / устройства оптимизирован связанный документ
method	<form>	Задает метод HTTP, используемый при отправке данных формы
min	<input>, <meter>	Задает минимальное значение
multiple	<input>, <select>	Указывает, что пользователь может ввести несколько значений
muted	<video>, <audio>	Указывает, что аудиовыход видео должен быть отключен

name	<button>, <fieldset>, <form>, <iframe>, <input>, <map>, <meta>, <object>, <output>, <param>, <select>, <textarea>	Задает имя элемента
novalidate	<form>	Указывает, что форма не должна проверяться при отправке
onabort	<audio>, <embed>, <img>, <object>, <video>	Сценарий для запуска при прерывании
onafterprint	<body>	Сценарий, запускаемый после печати документа
onbeforeprint	<body>	Сценарий, выполняемый перед печатью документа
onbeforeunload	<body>	Скрипт будет выполняться, когда документ будет выгружен
onblur	Все видимые элементы.	Сценарий для запуска, когда элемент теряет фокус
oncanplay	<audio>, <embed>, <object>, <video>	Сценарий для запуска, когда файл готов к запуску воспроизведения (когда он достаточно буферизован, чтобы начать)
oncanplaythrough	<audio>, <video>	Скрипт должен быть запущен, когда файл может быть воспроизведен до конца без паузы для буферизации
onchange	Все видимые элементы.	Скрипт, запускаемый при изменении значения элемента
onclick	Все видимые элементы.	Скрипт, запускаемый при щелчке по элементу
oncontextmenu	Все видимые элементы.	Скрипт, запускаемый при вызове контекстного меню
oncopy	Все видимые элементы.	Скрипт, запускаемый при копировании содержимого элемента
oncuechange	<track>	Сценарий, который будет выполняться при изменении ключа в элементе <track>
oncut	Все видимые элементы.	Скрипт, запускаемый при вырезании содержимого элемента
ondblclick	Все видимые элементы.	Скрипт, запускаемый при двойном щелчке по элементу
ondrag	Все видимые элементы.	Скрипт, запускаемый при перетаскивании элемента
ondragend	Все видимые элементы.	Скрипт, запускаемый в конце операции перетаскивания
ondragenter	Все видимые элементы.	Скрипт, запускаемый при перетаскивании элемента в допустимый целевой объект
ondragleave	Все видимые элементы.	Сценарий для запуска, когда элемент оставляет допустимый целевой объект отбрасывания
ondragover	Все видимые элементы.	Скрипт, запускаемый при перетаскивании элемента по допустимому целевому объекту
ondragstart	Все видимые элементы.	Сценарий для запуска в начале операции перетаскивания
ondrop	Все видимые элементы.	Скрипт, запускаемый при перетаскивании элемента
ondurationchange	<audio>, <video>	Сценарий для запуска при изменении длины носителя
onemptied	<audio>, <video>	Сценарий, который нужно запустить когда что-то плохое случается и архив внезапно недоступен (как непредвиденно разъединения)
onended	<audio>, <video>	Скрипт должен быть запущен, когда в СМИ уже дойдет до конца (полезное мероприятие для сообщения "спасибо за прослушивание")
onerror	<audio>, <body>, <embed>, <img>, <object>, <script>, <style>, <video>	Сценарий для запуска при возникновении ошибки
onfocus	Все видимые элементы.	Сценарий для запуска, когда элемент получает фокус
onhashchange	<body>	Скрипт, запускаемый при изменении привязки URL
oninput	Все видимые элементы.	Сценарий для запуска, когда элемент получает пользовательский ввод
oninvalid	Все видимые элементы.	Сценарий для запуска при недопустимом элементе
onkeydown	Все видимые элементы.	Скрипт, запускаемый при нажатии пользователем клавиши
onkeypress	Все видимые элементы.	Скрипт, запускаемый при нажатии пользователем клавиши
onkeyup	Все видимые элементы.	Скрипт, запускаемый при нажатии пользователем клавиши

onload	<body>, <iframe>, <img>, <input>, <link>, <script>, <style>	Скрипт, запускаемый после завершения загрузки элемента
onloadeddata	<audio>, <video>	Сценарий для запуска при загрузке данных мультимедиа
onloadedmetadata	<audio>, <video>	Скрипт, запускаемый при загрузке метаданных (например, измерений и длительности)
onloadstart	<audio>, <video>	Скрипт должен быть запущен и файл начинает загружаться до загружаемого
onmousedown	Все видимые элементы.	Скрипт, запускаемый при нажатии кнопки мыши на элементе
onmousemove	Все видимые элементы.	Скрипт должен выполняться до тех пор, пока указатель мыши перемещается по элементу
onmouseout	Все видимые элементы.	Скрипт, запускаемый при перемещении указателя мыши из элемента
onmouseover	Все видимые элементы.	Скрипт, запускаемый при наведении указателя мыши на элемент
onmouseup	Все видимые элементы.	Скрипт, запускаемый при отпускании кнопки мыши над элементом
onmousewheel	Все видимые элементы.	Скрипт, запускаемый при прокрутке колесика мыши по элементу
ononline	<body>	Скрипт, запускаемый при запуске браузера в автономном режиме
onpageshow	<body>	Скрипт, запускаемый при запуске браузера в оперативном режиме
onpagehide	<body>	Скрипт, запускаемый при переходе пользователя со страницы
onpaste	Все видимые элементы.	Сценарий, выполняемый при переходе пользователя на страницу
onpause	<audio>, <video>	Сценарий, выполняемый при вставке пользователем некоторого содержимого в элемент
onplay	<audio>, <video>	Сценарий, запускаемый при приостановке носителя пользователем или программным способом
onplaying	<audio>, <video>	Скрипт должен быть запущен, когда СМИ готовы начать играть
onpopstate	<body>	Сценарий для запуска, когда носитель фактически начал играть.
onprogress	<audio>, <video>	Скрипт, запускаемый при изменении истории окна.
onratechange	<audio>, <video>	Сценарий для запуска, когда браузер находится в процессе получения данных мультимедиа
onreset	<form>	Сценарий для запуска при каждом изменении скорости воспроизведения (например, когда пользователь переключается в режим замедленной или быстрой перемотки вперед).
onresize	<body>	Сценарий для запуска при нажатии кнопки сброса в форме.
onscroll	Все видимые элементы.	Скрипт, запускаемый при изменении размера окна браузера.
onsearch	<input>	Скрипт, запускаемый при прокрутке полосы прокрутки элемента
onseeked	<audio>, <video>	Сценарий, выполняемый при записи пользователем чего-либо в поле поиска (для <input="search">)
onseeking	<audio>, <video>	Сценарий, который будет выполняться, когда атрибут поиска имеет значение false, указывающее, что поиск закончился
onselect	Все видимые элементы.	Сценарий, который будет выполняться, если атрибут поиска имеет значение true, указывающее, что поиск активен
onshow	<menu>	Скрипт, запускаемый при выборе элемента
onstalled	<audio>, <video>	Скрипт, запускаемый при отображении элемента <menu> в виде контекстного меню
		Скрипт должен быть запущен, когда браузер не может получить данные по какой-либо причине

onstorage	<body>	Сценарий для запуска при обновлении области веб-хранилища
onsubmit	<form>	Сценарий для запуска при отправке формы
onsuspend	<audio>, <video>	Сценарий, который будет выполняться при извлечении данных мультимедиа останавливается перед полной загрузкой по какой-либо причине
ontimeupdate	<audio>, <video>	Скрипт для запуска при изменении игровой позиции (например, когда пользователь быстро переходит к другой точке в СМИ)
ontoggle	<details>	Сценарий, запускаемый при открытии или закрытии пользователем элемента <details>
onunload	<body>	Скрипт, запускаемый при выгрузке страницы (или закрытии окна браузера)
onvolumechange	<audio>, <video>	Сценарий, который будет выполняться каждый раз, когда видео / аудио громкость была изменена
onwaiting	<audio>, <video>	Скрипт должен быть запущен, когда СМИ приостановлена, но возобновится (например, когда СМИ останавливается для буферизации данных)
onwheel	Все видимые элементы.	Скрипт, запускаемый при накатывании колесика мыши вверх или вниз по элементу
open	<details>	Указывает, что сведения должны быть видны (открыты) пользователю
optimum	<meter>	Указывает, какое значение является оптимальным для датчика
pattern	<input>	Задает регулярное выражение, по которому проверяется значение элемента <input>
placeholder	<input>, <textarea>	Задает короткую подсказку, описывающую ожидаемое значение элемента
poster	<video>	Указывает изображение, которое будет отображаться во время загрузки видео или до тех пор, пока пользователь не нажмет кнопку воспроизведения
preload	<audio>, <video>	Указывает, если автор считает, что аудио / видео должно быть загружено при загрузке страницы
readonly	<input>, <textarea>	Указывает, что элемент доступен только для чтения
rel	<a>, <area>, <link>	Задает связь между текущим документом и связанным документом
required	<input>, <select>, <textarea>	Указывает, что элемент должен быть заполнен перед отправкой формы
reversed		Указывает, что порядок списка должен быть по убыванию (9,8,7...)
rows	<textarea>	Указывает видимое количество строк в текстовой области
rowspan	<td>, <th>	Задает количество строк, которое должна занимать ячейка таблицы
sandbox	<iframe>	Включает дополнительный набор ограничений для содержимого в <iframe>
scope	<th>	Указывает, является ли ячейка заголовка заголовком столбца, строки или группы столбцов или строк
scoped	<style>	Указывает, что стили применяются только к родительскому элементу этого элемента и дочерним элементам этого элемента
selected	<option>	Указывает, что параметр должен быть предварительно выбран при загрузке страницы
shape	<area>	Определяет форму области
size	<input>, <select>	Задает ширину в символах (для <input>) или указывает количество видимых параметров (для <select>)
sizes	<img>, <link>, <source>	Задает размер связанного ресурса
span	<col>, <colgroup>	Задает количество столбцов для интервала
spellcheck	Глобальные Атрибуты	Указывает, следует ли проверять орфографию и грамматику элемента

src	<audio>, <embed>, <iframe>, <img>, <input>, <script>, <source>, <track>, <video>	Задает URL файла мультимедиа
srcdoc	<iframe>	Задает содержимое HTML страницы для отображения в <iframe>
srclang	<track>	Задает язык текстовых данных дорожки (требуется, если kind="subtitles")
srcset	<img>, <source>	Задает URL изображения для использования в различных ситуациях
start		Задает начальное значение упорядоченного списка
step	<input>	Задает допустимые интервалы номеров для поля ввода
style	Глобальные Атрибуты	Задает встроенный стиль CSS для элемента
tabindex	Глобальные Атрибуты	Задает порядок табуляции элемента
target	<a>, <area>, <base>, <form>	Указывает целевой объект для открытия связанного документа или отправки формы
title	Глобальные Атрибуты	Задает дополнительные сведения об элементе
translate	Глобальные Атрибуты	Указывает, следует ли переводить содержимое элемента
type	<button>, <embed>, <input>, <link>, <menu>, <object>, <script>, <source>, <style>	Указывает тип элемента
usemap	<img>, <object>	Задает изображение в качестве клиентской карты изображений
value	<button>, <input>, , <option>, <meter>, <progress>, <param>	Задает значение элемента
width	<canvas>, <embed>, <iframe>, <img>, <input>, <object>, <video>	Задает ширину элемента
wrap	<textarea>	Указывает, как текст в текстовой области должен быть обернут при отправке в форме

□ Prev Next □

HTML Аудио/Видео DOM. Справочник

□ Prev Next □

HTML Аудио и Видео DOM Справочник

HTML5 DOM имеет методы, свойства и события для `<audio>` и `<video>` элементов.

Эти методы, свойства и события позволяют вам манипулировать `<audio>` и `<video>` элементами, используя JavaScript.

HTML Аудио/Видео Методы

Метод	Описание
addTextTrack()	Добавляет новый текстовый трек к аудио/видео
canPlayType()	Проверяет, браузер может воспроизводить указанный тип аудио/видео
load()	Перегружает аудио/видео элемент
play()	Начинает воспроизведение аудио/видео
pause()	Приостанавливает текущее воспроизведение аудио/видео

HTML Аудио/Видео Свойства

Свойство	Описание
audioTracks	Возвращает объект AudioTrackList, представляющий доступны звуковые дорожки
autoplay	Устанавливает или возвращает, следует ли начать воспроизведение аудио/видео, как только оно будет загружено

buffered	Возвращает объект TimeRanges, представляющий буферизированные части аудио/видео
controller	Возвращает объект MediaController, представляющий текущий контроллер медиа аудио/видео
controls	Устанавливает или возвращает, должно ли аудио/видео отражать элементы управления (например, воспроизведение/пауза пр.)
crossOrigin	Устанавливает или возвращает настройки CORS аудио/видео
currentSrc	Возвращает URL текущего аудио/видео
currentTime	Устанавливает или возвращает текущую позицию воспроизведения в аудио/видео (в секундах)
defaultMuted	Устанавливает или возвращает аудио/видео по умолчанию
defaultPlaybackRate	Устанавливает или возвращает стандартную скорость воспроизведения аудио/видео
duration	Возвращает продолжительность текущего аудио/видео (в секундах)
ended	Возвращает закончилось ли воспроизведения аудио/видео
error	Возвращает объект MediaError, представляющий состояние ошибки аудио/видео
loop	Устанавливает или возвращает, следует ли повторять аудио/видео после завершения
mediaGroup	Устанавливает или возвращает группу, к которой относится аудио/видео (используется для соединения нескольких аудио/видео элементов)
muted	Устанавливает или возвращает, выключено аудио/видео или нет
networkState	Возвращает текущее состояние сети аудио/видео
paused	Возвращает если аудио/видео приостановлено либо нет
playbackRate	Устанавливает или возвращает скорость воспроизведения аудио/видео
played	Возвращает объект TimeRanges, представляющий воспроизводимые части аудио/видео
preload	Устанавливает или возвращает, нужно ли загружать аудио/видео при загрузке страницы
readyState	Возвращает текущее состояние готовности аудио/видео
seekable	Возвращает объект TimeRanges, представляющий доступ для поиска части аудио/видео
seeking	Возвращает, ищет ли пользователь в данный момент аудио/видео
src	Устанавливает или возвращает текущее источник аудио/видео элемента
startDate	Возвращает объект Date, представляющий текущий сдвиг времени
textTracks	Возвращает объект TextTrackList, представляющий доступные текстовые дорожки
videoTracks	Возвращает объект VideoTrackList, представляющий доступные видеотреки
volume	Устанавливает или возвращает громкость аудио/видео

HTML Аудио/Видео События

Событие	Описание
abort	Срабатывает, когда загрузка аудио/видео прервана
canplay	Срабатывает, когда браузер может запустить воспроизведение аудио/видео
canplaythrough	Срабатывает, когда браузер может воспроизводить аудио/видео без остановки для буферизации
durationchange	Срабатывает, когда меняется продолжительность аудио/видео
emptied	Срабатывает, когда текущий список воспроизведения пуст
ended	Срабатывает после завершения текущего списка воспроизведения
error	Срабатывает, когда произошла ошибка при загрузке аудио/видео
loadeddata	Срабатывает, когда браузер загружает текущий фрейм (кадр) аудио/видео
loadedmetadata	Срабатывает, когда веб-браузер загрузил метаданные для аудио/видео
loadstart	Срабатывает, когда браузер начинает искать аудио/видео
pause	Срабатывает, когда аудио/видео было приостановлено
play	Срабатывает, когда аудио/видео было запущено или больше не приостановлено
playing	Срабатывает, когда аудио/видео воспроизводится после паузы или остановки для буферизации
progress	Срабатывает, когда браузер загружает аудио/видео
ratechange	Срабатывает при изменении скорости воспроизведения аудио/видео
seeked	Срабатывает, когда пользователь заканчивает перемещение/пропуск к новой позиции в аудио/видео
seeking	Срабатывает, когда пользователь начинает перемещение/пропуск к новой позиции в аудио/видео
stalled	Срабатывает, когда веб-браузер пытается получить медиа-данные, но данные недоступны

suspend	Срабатывает, когда веб-браузер намеренно не получает медиаданных
timeupdate	Срабатывает, когда меняется текущая позиция воспроизведения
volumechange	Срабатывает при изменении громкости
waiting	Срабатывает, когда видео останавливается, поскольку нужно сделать буферизацию следующего кадра

HTML элементы. Полный справочник тегов

Prev Next

HTML теги упорядоченные по категории

= Новые в HTML5.

Базовый HTML

Тег	Описание
<!DOCTYPE>	Определяет тип документа
<html>	Определяет документ HTML
<head>	Определяет информацию о документе
<title>	Определяет заголовок документа
<body>	Определяет тело документа
<h1> to <h6>	Определяет заголовки HTML
<p>	Определяет параграф (абзац)
 	Вставляет один разрыв строки
<hr>	Определяет тематическое изменение содержания
<!--...-->	Определяет комментарий

Форматирование

Тег	Описание
<acronym>	Не поддерживается в HTML5. Используйте <abbr> вместо этого. Определяет акроним
<abbr>	Определяет аббревиатуру или акроним
<address>	Определяет контактную информацию для автора / владельца документа / статьи
<b>	Определяет жирный текст
<bdi>	Изолирует часть текста, который может быть отформатирован в другом направлении от прочего текста за его пределами
<bdo>	Переопределяет текущее направление текста
<big>	Не поддерживается в HTML5. Используйте CSS вместо этого. Определяет большой текст
<blockquote>	Определяет раздел, что цитируется с другого источника
	Не поддерживается в HTML5. Используйте CSS вместо этого. Определяет центрирование текста
<cite>	Определяет название произведения
<code>	Определяет фрагмент компьютерного кода
<del>	Определяет удалённый с документа текст
<dfn>	Представляет определяющий экземпляр термина
	Определяет подчёркнутый текст
<font>	Не поддерживается в HTML5. Используйте CSS вместо этого. Определяет шрифт, цвет и размер для текста
<i>	Определяет часть текста альтернативным голосом или настроением
<ins>	Определяет текст, который был вставлен в документ

<kbd>	Определяет ввод с клавиатуры
<mark>	Определяет отмеченный / выделенный текст
<meter>	Определяет скалярное измерение в границах известного диапазона (датчик)
<pre>	Определяет предварительно отформатированный текст
<progress>	Представляет ход выполнения задания
<q>	Определяет короткую цитату
<rp>	Определяет, что отображать в браузерах, которые не поддерживают ruby аннотации
<rt>	Определяет объяснение / произношение символов (для восточноазиатской типографики)
<ruby>	Определяет аннотацию ruby (для восточноазиатской типографики)
<s>	Определяет текст, которій больше не является правильным
<samp>	Определяет исходные данные с компьютерной программы
<small>	Определяет меньший текст
<strike>	Не поддерживается в HTML5. Используйте <del> или <s> вместо этого. Определяет перечёркнутый текст
	Определяет важный текст
<sub>	Определяет подстрочный текст (нижний индекс)
<sup>	Определяет надстрочный текст (верхний индекс)
<template>	Определяет шаблон
<time>	Определяет дату / время
<tt>	Не поддерживается в HTML5. Используйте CSS вместо этого. Определяет текст телетайпа
<u>	Определяет текст, который должен быть стилистически отличным от обычного текста
<var>	Определяет переменную
<wbr>	Определяет возможный разрыв строки

Формы и входящие данные

Тег	Описание
<form>	Определяет форму HTML для ввода пользователем
<input>	Определяет элемент управления вводом
<textarea>	Определяет многострочный элемент управления вводом (текстовая область)
<button>	Определяет кнопку, которую можно нажимать
<select>	Определяет выпадающий список
<optgroup>	Определяет группу соответствующих параметров в выпадающем списке
<option>	Определяет параметр в выпадающем списке
<label>	Определяет метку для <input> элемента
<fieldset>	Группы связанных элементов в форме
<legend>	Определяет заголовок для <fieldset> элемента
<datalist>	Определяет список предварительно определённых параметров управления вводом
<output>	Определяет результат расчёта

Фреймы

Тег	Описание
<frame>	Не поддерживается в HTML5. Определяет окно (фрейм) в наборе фреймов
<frameset>	Не поддерживается в HTML5. Определяет набор фреймов
<noframes>	Не поддерживается в HTML5. Определяет альтернативное содержание для пользователей, которые не поддерживают фреймы
<iframe>	Определяет встроенный фрейм

Изображения

Тег	Изображение
<img>	Определяет изображение

<map>	Определяет карту изображения на стороне клиента
<area>	Определяет область внутри карты изображения
<canvas>	Используется для рисования на лету с помощью сценариев (обычно на JavaScript)
<figcaption>	Определяет заголовок для элемента <figure>
<figure>	Определяет автономное содержание
<picture>	Определяет контейнер для нескольких ресурсов изображения
<svg>	Определяет контейнер для графики SVG

Аудио / Видео

Тег	Описание
<audio>	Определяет звуковой контент
<source>	Определяет несколько медиа-ресурсов для медиа-элементов (<video>, <audio>, <picture>)
<track>	Определяет текстовые дорожки для медиа-элементов (<video> и <audio>)
<video>	Определяет видео или фильм

Ссылки

Тег	Описание
<a>	Определяет гиперссылку
<link>	Определяет взаимосвязь между документом и внешним ресурсом (наиболее часто используется для ссылки на внешние таблицы стилей)
<nav>	Определяет навигационные ссылки (навигация по сайту)

Списки

Тег	Описание
	Определяет неупорядоченный (ненумерованный) список
	Определяет упорядоченный (нумерованный) список
	Определяет элемент списка
<dir>	Не поддерживается в HTML5. Используйте вместо этого. Определяет список каталогов
<dl>	Определяет список описаний
<dt>	Определяет термин / имя в списке описания
<dd>	Определяет описание / значение термина в списке описания

Таблицы

Тег	Описание
<table>	Определяет таблицу
<caption>	Определяет подпись таблицы
<th>	Определяет ячейку заголовка в таблице
<tr>	Определяет строку в таблице
<td>	Определяет ячейку в таблице
<thead>	Группирует содержание заголовка в таблице
<tbody>	Группирует содержание тела в таблице
<tfoot>	Группирует содержание нижнего колонтитула в таблице
<col>	Указывает свойства столбцов для каждого столбца в элементе <colgroup>
<colgroup>	Определяет группу с одного или нескольких столбцов в таблице для форматирования

Стили и семантика

Тег	Описание
<style>	Определяет информацию о стиле для документа
	Определяет блочный раздел в документе
	Определяет строчный раздел в документе
<header>	Определяет заголовок для документа или раздела

<footer>	Определяет нижний колонтитул (футер) для документа или раздела
<main>	Определяет основное содержание документа
<section>	Определяет раздел (секцию) в документе
<article>	Определяет статью
<aside>	Определяет содержание, кроме содержимого страницы (в стороне)
<details>	Определяет дополнительные детали, которые пользователь может просматривать или прятать
<dialog>	Определяет диалоговiй бокс или окно
<summary>	Определяет видимый заголовок для элемента <details>
<data>	Связывает заданный контент с машиночитаемым переводом

Мета Информация

Тег	Описание
<head>	Определяет информацию о документе
<meta>	Определяет метаданные HTML документа
<base>	Указывает базовый URL-адрес / цель для всех относительных URL-адресов документа
<basefont>	Не поддерживается в HTML5. Используйте CSS вместо этого. Определяет цвет, размер и шрифт по умолчанию для всего текста документа

Программирование

Тег	Описание
<script>	Определяет скрипт на стороне клиента
<noscript>	Определяет альтернативное содержание для пользователей, которые не поддерживают сценари на стороне клиента
<applet>	Не поддерживается в HTML5. Используйте <embed> или <object> вместо этого. Определяет встроенный апплет
<embed>	Определяет контейнер для внешнего (не HTML) приложения
<object>	Определяет встроенный объект
<param>	Определяет параметр для объекта

Prev Next

HTML Canvas. Справочник

Prev Next

Описание

Canvas - переводится с англ. *Холст*.

HTML5 тег `<canvas>` используется для рисования графики на лету с помощью скриптов (обычно с помощью JavaScript).

Однако элемент `<canvas>` не имеет собственных возможностей для рисования (он является лишь контейнером для графики) - вы должны использовать скрипты, чтобы рисовать графику.

Метод `getContext()` возвращает объект, который предоставляет методы и свойства для рисования в **canvas**.

Эта ссылка будет охватывать свойства и методы объекта `getContext("2d")` который можно использовать для рисования текста, линий, кубов, кругов и многого другого - в **canvas**.

Поддержка браузерами

Цифры в таблице определяют первую версию браузера, которая полностью поддерживает этот элемент.

Элемент					
<canvas>	4.0	9.0	2.0	3.1	9.0

Internet Explorer 9, Firefox, Opera, Chrome и Safari поддерживают `<canvas>` и его свойства и методы.

Примечание: Internet Explorer 8 и более ранние версии не поддерживают элемент `<canvas>`.

Цвета, стили и тени

Свойства	Описание
fillStyle	Устанаввливает или возвращает цвет, градиент или шаблон, используемый для заполнения рисунка
strokeStyle	Устанавливает или возвращает цвет, градиент или шаблон для штрихов
shadowColor	Устанавливает или возвращает цвет для использования при создании теней
shadowBlur	Устанавливает или возвращает уровень размытия для теней
shadowOffsetX	Устанавливает или возвращает горизонтальное расстояние тени от формы
shadowOffsetY	Устанавливает или возвращает вертикальное расстояние тени от формы

Метод	Описание
createLinearGradient()	Создает линейный градиент (для использования на содержимом canvas)
createPattern()	Повторяет указанный элемент в указанном направлении
createRadialGradient()	Создает радиальный / круговой градиент (для использования на содержимом canvas)
addColorStop()	Определяет цвета и позиции остановки в объекте градиента

Стили строк

Свойства	Описание
lineCap	Устанавливает или возвращает стиль заглавных букв для строки
lineJoin	Устанавливает или возвращает тип созданного угла, когда встречаются две линии
lineWidth	Устанавливает или возвращает текущую ширину строки
miterLimit	Устанавливает или возвращает максимальную длину митры (скоса)

Прямоугольники

Метод	Описание
rect()	Создаёт прямоугольник
fillRect()	Рисует "заполненный" прямоугольник
strokeRect()	Рисует прямоугольник (без заливки)
clearRect()	Очищает указанные пиксели в пределах данного прямоугольника

Контуры

Метод	Описание
fill()	Заполняет текущий рисунок (контур)
stroke()	Фактически рисует определённый вами контур
beginPath()	Начинается контур или сбрасывается текущий контур
moveTo()	Перемещает контур к указанной точке на холсте, не создавая строки
closePath()	Создаёт контур от текущей точки до начальной точки
lineTo()	Добавляет новую точку и создаёт линию к этой точке с последней указаной точки на холсте
clip()	Закрепляет область любой формы и размера с оригинального холста
quadraticCurveTo()	Создаёт квадратичную кривую Безье
bezierCurveTo()	Создаёт кубическую кривую Безье
arc()	Создаёт дугу/кривую (используется для создания кругов или частей кругов)
arcTo()	Создаёт дугу/кривую между двумя касаемыми
isPointInPath()	Возвращает true (истина), если указанная точка находится в текущем контуре, иначе false (ложь)

Трансформации

Метод	Описание
scale()	Масштабирует текущий рисунок, делая большим или меньшим

rotate()	Возвращает текущий рисунок
translate()	Изменяет позицию (0,0) на холсте
transform()	Заменяет текущую матрицу преобразования для чертежа (рисунка)
setTransform()	Сбрасывает текущее преобразование на идентичную матрицу. Затем запускается transform()

Текст

Свойства	Описание
font	Устанавливает или возвращает текущие свойства шрифта для текстового содержания
textAlign	Устанавливает или возвращает текущее выравнивание для текстового содержимого
textBaseline	Устанавливает или возвращает текущую текстовую базовую линию, которая используется при рисовании текста

Метод	Описание
fillText()	Рисует "заполненный" текст (из заливкой) на холсте
strokeText()	Рисует текст на холсте (без заливки)
measureText()	Возвращает объект, который содержит ширину указанного текста

Рисование изображения

Метод	Описание
drawImage()	Рисует изображение, холст или видео на холсте

Пиксельная манипуляция

Свойства	Описание
width	Возвращает ширину объекта ImageData
height	Возвращает высоту объекта ImageData
data	Возвращает объект, который содержит данные изображения указанного объекта ImageData

Метод	Описание
createImageData()	Создаёт новый, пустой объект ImageData
getImageData()	Возвращает объект ImageData, который копирует пиксельные даные для заданного прямоугольника на холсте
putImageData()	Возвращает данные изображения (из указанного объекта ImageData) назад на холст

Композиция

Свойства	Описание
globalAlpha	Устанавливает или возвращает текущее значение альфа или прозрачности рисунка
globalCompositeOperation	Устанавливает или возвращает способ рисования нового изображения на существующем изображении.

Прочее

Метод	Описание
save()	Сохраняет состояние текущего контекста
restore()	Возвращает ранее сохранённое состояние контура и атрибуты
createEvent()	
getContext()	
toDataURL()	

□ Prev Next □

HTML Наборы символов (коды). Справочник

□ Prev Next □

Общие наборы символов HTML

ANSI был первым официальным набором символов в Windows.

Стандартный набор символов в HTML 4 був 8859-1.

Стандартный набор символов в HTML 5 - UTF-8.

Для более детального ознакомления посетите Полный справочник наборов символов HTML.

Число	ASCII	ANSI	8859-1	UTF-8	Описание
32					space
33	!	!	!	!	exclamation mark
34	"	"	"	"	quotation mark
35	#	#	#	#	number sign
36	$	$	$	$	dollar sign
37	%	%	%	%	percent sign
38	&	&	&	&	ampersand
39	'	'	'	'	apostrophe
40	(	(	(	(	left parenthesis
41	)	)	)	)	right parenthesis
42	*	*	*	*	asterisk
43	+	+	+	+	plus sign
44	,	,	,	,	comma
45	-	-	-	-	hyphen-minus
46	.	.	.	.	full stop
47	/	/	/	/	solidus
48	0	0	0	0	digit zero
49	1	1	1	1	digit one
50	2	2	2	2	digit two
51	3	3	3	3	digit three
52	4	4	4	4	digit four
53	5	5	5	5	digit five
54	6	6	6	6	digit six
55	7	7	7	7	digit seven
56	8	8	8	8	digit eight
57	9	9	9	9	digit nine
58	:	:	:	:	colon
59	;	;	;	;	semicolon
60	<	<	<	<	less-than sign
61	=	=	=	=	equals sign
62	>	>	>	>	greater-than sign
63	?	?	?	?	question mark
64	@	@	@	@	commercial at
65	A	A	A	A	Latin capital letter A
66	B	B	B	B	Latin capital letter B
67	C	C	C	C	Latin capital letter C
68	D	D	D	D	Latin capital letter D
69	E	E	E	E	Latin capital letter E
70	F	F	F	F	Latin capital letter F
71	G	G	G	G	Latin capital letter G
72	H	H	H	H	Latin capital letter H
73	I	I	I	I	Latin capital letter I
74	J	J	J	J	Latin capital letter J
75	K	K	K	K	Latin capital letter K
76	L	L	L	L	Latin capital letter L
77	M	M	M	M	Latin capital letter M

78	N	N	N	N	Latin capital letter N
79	O	O	O	O	Latin capital letter O
80	P	P	P	P	Latin capital letter P
81	Q	Q	Q	Q	Latin capital letter Q
82	R	R	R	R	Latin capital letter R
83	S	S	S	S	Latin capital letter S
84	T	T	T	T	Latin capital letter T
85	U	U	U	U	Latin capital letter U
86	V	V	V	V	Latin capital letter V
87	W	W	W	W	Latin capital letter W
88	X	X	X	X	Latin capital letter X
89	Y	Y	Y	Y	Latin capital letter Y
90	Z	Z	Z	Z	Latin capital letter Z
91	[	[	[	[	left square bracket
92	\	\	\	\	reverse solidus
93	]	]	]	]	right square bracket
94	^	^	^	^	circumflex accent
95	_	_	_	_	low line
96	`	`	`	`	grave accent
97	a	a	a	a	Latin small letter a
98	b	b	b	b	Latin small letter b
99	c	c	c	c	Latin small letter c
100	d	d	d	d	Latin small letter d
101	e	e	e	e	Latin small letter e
102	f	f	f	f	Latin small letter f
103	g	g	g	g	Latin small letter g
104	h	h	h	h	Latin small letter h
105	i	i	i	i	Latin small letter i
106	j	j	j	j	Latin small letter j
107	k	k	k	k	Latin small letter k
108	l	l	l	l	Latin small letter l
109	m	m	m	m	Latin small letter m
110	n	n	n	n	Latin small letter n
111	o	o	o	o	Latin small letter o
112	p	p	p	p	Latin small letter p
113	q	q	q	q	Latin small letter q
114	r	r	r	r	Latin small letter r
115	s	s	s	s	Latin small letter s
116	t	t	t	t	Latin small letter t
117	u	u	u	u	Latin small letter u
118	v	v	v	v	Latin small letter v
119	w	w	w	w	Latin small letter w
120	x	x	x	x	Latin small letter x
121	y	y	y	y	Latin small letter y
122	z	z	z	z	Latin small letter z
123	{	{	{	{	left curly bracket
124	|	|	|	|	vertical line
125	}	}	}	}	right curly bracket
126	~	~	~	~	tilde
127	DEL				
128		€			euro sign
129					NOT USED
130		‚			single low-9 quotation mark
131		*ƒ*			Latin small letter f with hook
132		„			double low-9 quotation mark
133		…			horizontal ellipsis

134	†			dagger
135	‡			double dagger
136	ˆ			modifier letter circumflex accent
137	‰			per mille sign
138	Š			Latin capital letter S with caron
139	‹			single left-pointing angle quotation mark
140	Œ			Latin capital ligature OE
141				NOT USED
142	Ž			Latin capital letter Z with caron
143				NOT USED
144				NOT USED
145	‘			left single quotation mark
146	’			right single quotation mark
147	“			left double quotation mark
148	”			right double quotation mark
149	•			bullet
150	–			en dash
151	—			em dash
152	˜			small tilde
153	™			trade mark sign
154	š			Latin small letter s with caron
155	›			single right-pointing angle quotation mark
156	œ			Latin small ligature oe
157				NOT USED
158	ž			Latin small letter z with caron
159	Ÿ			Latin capital letter Y with diaeresis
160				no-break space
161	¡	¡	¡	inverted exclamation mark
162	¢	¢	¢	cent sign
163	£	£	£	pound sign
164	¤	¤	¤	currency sign
165	¥	¥	¥	yen sign
166	¦	¦	¦	broken bar
167	§	§	§	section sign
168	¨	¨	¨	diaeresis
169	©	©	©	copyright sign
170	ª	ª	ª	feminine ordinal indicator
171	«	«	«	left-pointing double angle quotation mark
172	¬	¬	¬	not sign
173				soft hyphen
174	®	®	®	registered sign
175	¯	¯	¯	macron
176	°	°	°	degree sign
177	±	±	±	plus-minus sign
178	²	²	²	superscript two
179	³	³	³	superscript three
180	´	´	´	acute accent
181	µ	µ	µ	micro sign
182	¶	¶	¶	pilcrow sign
183	·	·	·	middle dot
184	¸	¸	¸	cedilla
185	¹	¹	¹	superscript one
186	º	º	º	masculine ordinal indicator
187	»	»	»	right-pointing double angle quotation mark
188	¼	¼	¼	vulgar fraction one quarter
189	½	½	½	vulgar fraction one half

190	¾	¾	¾	vulgar fraction three quarters
191	¿	¿	¿	inverted question mark
192	À	À	À	Latin capital letter A with grave
193	Á	Á	Á	Latin capital letter A with acute
194	Â	Â	Â	Latin capital letter A with circumflex
195	Ã	Ã	Ã	Latin capital letter A with tilde
196	Ä	Ä	Ä	Latin capital letter A with diaeresis
197	Å	Å	Å	Latin capital letter A with ring above
198	Æ	Æ	Æ	Latin capital letter AE
199	Ç	Ç	Ç	Latin capital letter C with cedilla
200	È	È	È	Latin capital letter E with grave
201	É	É	É	Latin capital letter E with acute
202	Ê	Ê	Ê	Latin capital letter E with circumflex
203	Ë	Ë	Ë	Latin capital letter E with diaeresis
204	Ì	Ì	Ì	Latin capital letter I with grave
205	Í	Í	Í	Latin capital letter I with acute
206	Î	Î	Î	Latin capital letter I with circumflex
207	Ï	Ï	Ï	Latin capital letter I with diaeresis
208	Ð	Ð	Ð	Latin capital letter Eth
209	Ñ	Ñ	Ñ	Latin capital letter N with tilde
210	Ò	Ò	Ò	Latin capital letter O with grave
211	Ó	Ó	Ó	Latin capital letter O with acute
212	Ô	Ô	Ô	Latin capital letter O with circumflex
213	Õ	Õ	Õ	Latin capital letter O with tilde
214	Ö	Ö	Ö	Latin capital letter O with diaeresis
215	×	×	×	multiplication sign
216	Ø	Ø	Ø	Latin capital letter O with stroke
217	Ù	Ù	Ù	Latin capital letter U with grave
218	Ú	Ú	Ú	Latin capital letter U with acute
219	Û	Û	Û	Latin capital letter U with circumflex
220	Ü	Ü	Ü	Latin capital letter U with diaeresis
221	Ý	Ý	Ý	Latin capital letter Y with acute
222	Þ	Þ	Þ	Latin capital letter Thorn
223	ß	ß	ß	Latin small letter sharp s
224	à	à	à	Latin small letter a with grave
225	á	á	á	Latin small letter a with acute
226	â	â	â	Latin small letter a with circumflex
227	ã	ã	ã	Latin small letter a with tilde
228	ä	ä	ä	Latin small letter a with diaeresis
229	å	å	å	Latin small letter a with ring above
230	æ	æ	æ	Latin small letter ae
231	ç	ç	ç	Latin small letter c with cedilla
232	è	è	è	Latin small letter e with grave
233	é	é	é	Latin small letter e with acute
234	ê	ê	ê	Latin small letter e with circumflex
235	ë	ë	ë	Latin small letter e with diaeresis
236	ì	ì	ì	Latin small letter i with grave
237	í	í	í	Latin small letter i with acute
238	î	î	î	Latin small letter i with circumflex
239	ï	ï	ï	Latin small letter i with diaeresis
240	ð	ð	ð	Latin small letter eth
241	ñ	ñ	ñ	Latin small letter n with tilde
242	ò	ò	ò	Latin small letter o with grave
243	ó	ó	ó	Latin small letter o with acute
244	ô	ô	ô	Latin small letter o with circumflex
245	õ	õ	õ	Latin small letter o with tilde
246	ö	ö	ö	Latin small letter o with diaeresis

247	÷	÷	÷	division sign
248	ø	ø	ø	Latin small letter o with stroke
249	ù	ù	ù	Latin small letter u with grave
250	ú	ú	ú	Latin small letter u with acute
251	û	û	û	Latin small letter with circumflex
252	ü	ü	ü	Latin small letter u with diaeresis
253	ý	ý	ý	Latin small letter y with acute
254	þ	þ	þ	Latin small letter thorn
255	ÿ	ÿ	ÿ	Latin small letter y with diaeresis

□ Prev Next □

HTML Названия всех цветов. Справочник

□ Prev Next □

Названия цветов, которые поддерживаются всеми браузерами

Все современные веб-браузеры поддерживают следующие 140 названий цветов (нажмите на название цвета или шестнадцатеричное значение, чтобы просмотреть цвет как фоновый вместе с разными цветами текста):

Чтобы узнать подробнее о цветах HTML, посетите учебник по цветам.

Название цвета	HEX	Цвет	Оттенки	Смешивание
AliceBlue	#F0F8FF		Shades	Mix
AntiqueWhite	#FAEBD7		Shades	Mix
Aqua	#00FFFF		Shades	Mix
Aquamarine	#7FFFD4		Shades	Mix
Azure	#F0FFFF		Shades	Mix
Beige	#F5F5DC		Shades	Mix
Bisque	#FFE4C4		Shades	Mix
Black	#000000		Shades	Mix
BlanchedAlmond	#FFEBCD		Shades	Mix
Blue	#0000FF		Shades	Mix
BlueViolet	#8A2BE2		Shades	Mix
Brown	#A52A2A		Shades	Mix
BurlyWood	#DEB887		Shades	Mix
CadetBlue	#5F9EA0		Shades	Mix
Chartreuse	#7FFF00		Shades	Mix
Chocolate	#D2691E		Shades	Mix
Coral	#FF7F50		Shades	Mix
CornflowerBlue	#6495ED		Shades	Mix
Cornsilk	#FFF8DC		Shades	Mix
Crimson	#DC143C		Shades	Mix
Cyan	#00FFFF		Shades	Mix
DarkBlue	#00008B		Shades	Mix
DarkCyan	#008B8B		Shades	Mix
DarkGoldenRod	#B8860B		Shades	Mix
DarkGray	#A9A9A9		Shades	Mix
DarkGrey	#A9A9A9		Shades	Mix
DarkGreen	#006400		Shades	Mix
DarkKhaki	#BDB76B		Shades	Mix
DarkMagenta	#8B008B		Shades	Mix
DarkOliveGreen	#556B2F		Shades	Mix
DarkOrange	#FF8C00		Shades	Mix
DarkOrchid	#9932CC		Shades	Mix
DarkRed	#8B0000		Shades	Mix

DarkSalmon	#E9967A		Shades	Mix
DarkSeaGreen	#8FBC8F		Shades	Mix
DarkSlateBlue	#483D8B		Shades	Mix
DarkSlateGray	#2F4F4F		Shades	Mix
DarkSlateGrey	#2F4F4F		Shades	Mix
DarkTurquoise	#00CED1		Shades	Mix
DarkViolet	#9400D3		Shades	Mix
DeepPink	#FF1493		Shades	Mix
DeepSkyBlue	#00BFFF		Shades	Mix
DimGray	#696969		Shades	Mix
DimGrey	#696969		Shades	Mix
DodgerBlue	#1E90FF		Shades	Mix
FireBrick	#B22222		Shades	Mix
FloralWhite	#FFFAF0		Shades	Mix
ForestGreen	#228B22		Shades	Mix
Fuchsia	#FF00FF		Shades	Mix
Gainsboro	#DCDCDC		Shades	Mix
GhostWhite	#F8F8FF		Shades	Mix
Gold	#FFD700		Shades	Mix
GoldenRod	#DAA520		Shades	Mix
Gray	#808080		Shades	Mix
Grey	#808080		Shades	Mix
Green	#008000		Shades	Mix
GreenYellow	#ADFF2F		Shades	Mix
HoneyDew	#F0FFF0		Shades	Mix
HotPink	#FF69B4		Shades	Mix
IndianRed	#CD5C5C		Shades	Mix
Indigo	#4B0082		Shades	Mix
Ivory	#FFFFF0		Shades	Mix
Khaki	#F0E68C		Shades	Mix
Lavender	#E6E6FA		Shades	Mix
LavenderBlush	#FFF0F5		Shades	Mix
LawnGreen	#7CFC00		Shades	Mix
LemonChiffon	#FFFACD		Shades	Mix
LightBlue	#ADD8E6		Shades	Mix
LightCoral	#F08080		Shades	Mix
LightCyan	#E0FFFF		Shades	Mix
LightGoldenRodYellow	#FAFAD2		Shades	Mix
LightGray	#D3D3D3		Shades	Mix
LightGrey	#D3D3D3		Shades	Mix
LightGreen	#90EE90		Shades	Mix
LightPink	#FFB6C1		Shades	Mix
LightSalmon	#FFA07A		Shades	Mix
LightSeaGreen	#20B2AA		Shades	Mix
LightSkyBlue	#87CEFA		Shades	Mix
LightSlateGray	#778899		Shades	Mix
LightSlateGrey	#778899		Shades	Mix
LightSteelBlue	#B0C4DE		Shades	Mix
LightYellow	#FFFFE0		Shades	Mix
Lime	#00FF00		Shades	Mix
LimeGreen	#32CD32		Shades	Mix
Linen	#FAF0E6		Shades	Mix
Magenta	#FF00FF		Shades	Mix
Maroon	#800000		Shades	Mix
MediumAquaMarine	#66CDAA		Shades	Mix
MediumBlue	#0000CD		Shades	Mix

MediumOrchid	#BA55D3	Shades	Mix
MediumPurple	#9370DB	Shades	Mix
MediumSeaGreen	#3CB371	Shades	Mix
MediumSlateBlue	#7B68EE	Shades	Mix
MediumSpringGreen	#00FA9A	Shades	Mix
MediumTurquoise	#48D1CC	Shades	Mix
MediumVioletRed	#C71585	Shades	Mix
MidnightBlue	#191970	Shades	Mix
MintCream	#F5FFFA	Shades	Mix
MistyRose	#FFE4E1	Shades	Mix
Moccasin	#FFE4B5	Shades	Mix
NavajoWhite	#FFDEAD	Shades	Mix
Navy	#000080	Shades	Mix
OldLace	#FDF5E6	Shades	Mix
Olive	#808000	Shades	Mix
OliveDrab	#6B8E23	Shades	Mix
Orange	#FFA500	Shades	Mix
OrangeRed	#FF4500	Shades	Mix
Orchid	#DA70D6	Shades	Mix
PaleGoldenRod	#EEE8AA	Shades	Mix
PaleGreen	#98FB98	Shades	Mix
PaleTurquoise	#AFEEEE	Shades	Mix
PaleVioletRed	#DB7093	Shades	Mix
PapayaWhip	#FFEFD5	Shades	Mix
PeachPuff	#FFDAB9	Shades	Mix
Peru	#CD853F	Shades	Mix
Pink	#FFC0CB	Shades	Mix
Plum	#DDA0DD	Shades	Mix
PowderBlue	#B0E0E6	Shades	Mix
Purple	#800080	Shades	Mix
RebeccaPurple	#663399	Shades	Mix
Red	#FF0000	Shades	Mix
RosyBrown	#BC8F8F	Shades	Mix
RoyalBlue	#4169E1	Shades	Mix
SaddleBrown	#8B4513	Shades	Mix
Salmon	#FA8072	Shades	Mix
SandyBrown	#F4A460	Shades	Mix
SeaGreen	#2E8B57	Shades	Mix
SeaShell	#FFF5EE	Shades	Mix
Sienna	#A0522D	Shades	Mix
Silver	#C0C0C0	Shades	Mix
SkyBlue	#87CEEB	Shades	Mix
SlateBlue	#6A5ACD	Shades	Mix
SlateGray	#708090	Shades	Mix
SlateGrey	#708090	Shades	Mix
Snow	#FFFAFA	Shades	Mix
SpringGreen	#00FF7F	Shades	Mix
SteelBlue	#4682B4	Shades	Mix
Tan	#D2B48C	Shades	Mix
Teal	#008080	Shades	Mix
Thistle	#D8BFD8	Shades	Mix
Tomato	#FF6347	Shades	Mix
Turquoise	#40E0D0	Shades	Mix
Violet	#EE82EE	Shades	Mix
Wheat	#F5DEB3	Shades	Mix
White	#FFFFFF	Shades	Mix
WhiteSmoke	#F5F5F5	Shades	Mix

Yellow	#FFFF00		Shades	Mix
YellowGreen	#9ACD32		Shades	Mix

Prev Next

HTML Кодировка страны веб-страницы. Справочник

Prev Next

ISO Коды стран

ISO коды стран определяют сокращения (аббревиатуру) для стран.

В HTML они могут использоваться в качестве дополнения к значению языка в атрибуте `lang`.

Первые два символа языкового кода определяют **язык** (смотрите предыдущий справочник).

Последние два символа определяют **страну**.

```
<html lang="en-US">
...
</html>
```

В данном примере язык - английский, страна - США.

ISO Коды стран

Страна	ISO Код
AFGHANISTAN	AF
ALBANIA	AL
ALGERIA	DZ
AMERICAN SAMOA	AS
ANDORRA	AD
ANGOLA	AO
ANTARCTICA	AQ
ANTIGUA AND BARBUDA	AG
ARGENTINA	AR
ARMENIA	AM
ARUBA	AW
AUSTRALIA	AU
AUSTRIA	AT
AZERBAIJAN	AZ
BAHAMAS	BS
BAHRAIN	BH
BANGLADESH	BD
BARBADOS	BB
BELARUS	BY
BELGIUM	BE
BELIZE	BZ
BENIN	BJ
BERMUDA	BM
BHUTAN	BT
BOLIVIA	BO
BOSNIA AND HERZEGOVINA	BA
BOTSWANA	BW
BOUVET ISLAND	BV
BRAZIL	BR
BRITISH INDIAN OCEAN TERRITORY	IO

BRUNEI DARUSSALAM	BN
BULGARIA	BG
BURKINA FASO	BF
BURUNDI	BI
CAMBODIA	KH
CAMEROON	CM
CANADA	CA
CAPE VERDE	CV
CAYMAN ISLANDS	KY
CENTRAL AFRICAN REPUBLIC	CF
CHAD	TD
CHILE	CL
CHINA	CN
CHRISTMAS ISLAND	CX
COCOS (KEELING) ISLANDS	CC
COLOMBIA	CO
COMOROS	KM
CONGO	CG
CONGO, THE DEMOCRATIC REPUBLIC OF THE	CD
COOK ISLANDS	CK
COSTA RICA	CR
CÔTE D'IVOIRE	CI
CROATIA	HR
CUBA	CU
CYPRUS	CY
CZECH REPUBLIC	CZ
DENMARK	DK
DJIBOUTI	DJ
DOMINICA	DM
DOMINICAN REPUBLIC	DO
ECUADOR	EC
EGYPT	EG
EL SALVADOR	SV
EQUATORIAL GUINEA	GQ
ERITREA	ER
ESTONIA	EE
ETHIOPIA	ET
FALKLAND ISLANDS (MALVINAS)	FK
FAROE ISLANDS	FO
FIJI	FJ
FINLAND	FI
FRANCE	FR
FRENCH GUIANA	GF
FRENCH POLYNESIA	PF
FRENCH SOUTHERN TERRITORIES	TF
GABON	GA
GAMBIA	GM
GEORGIA	GE
GERMANY	DE
GHANA	GH
GIBRALTAR	GI
GREECE	GR
GREENLAND	GL
GRENADA	GD
GUADELOUPE	GP
GUAM	GU
GUATEMALA	GT

GUINEA	GN
GUINEA-BISSAU	GW
GUYANA	GY
HAITI	HT
HEARD ISLAND AND MCDONALD ISLANDS	HM
HONDURAS	HN
HONG KONG	HK
HUNGARY	HU
ICELAND	IS
INDIA	IN
INDONESIA	ID
IRAN, ISLAMIC REPUBLIC OF	IR
IRAQ	IQ
IRELAND	IE
ISRAEL	IL
ITALY	IT
JAMAICA	JM
JAPAN	JP
JORDAN	JO
KAZAKHSTAN	KZ
KENYA	KE
KIRIBATI	KI
KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF	KP
KOREA, REPUBLIC OF	KR
KUWAIT	KW
KYRGYZSTAN	KG
LAO PEOPLE'S DEMOCRATIC REPUBLIC (LAOS)	LA
LATVIA	LV
LEBANON	LB
LESOTHO	LS
LIBERIA	LR
LIBYAN ARAB JAMAHIRIYA	LY
LIECHTENSTEIN	LI
LITHUANIA	LT
LUXEMBOURG	LU
MACAO	MO
MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF	MK
MADAGASCAR	MG
MALAWI	MW
MALAYSIA	MY
MALDIVES	MV
MALI	ML
MALTA	MT
MARSHALL ISLANDS	MH
MARTINIQUE	MQ
MAURITANIA	MR
MAURITIUS	MU
MAYOTTE	YT
MEXICO	MX
MICRONESIA, FEDERATED STATES OF	FM
MOLDOVA, REPUBLIC OF	MD
MONACO	MC
MONGOLIA	MN
MONTENEGRO	ME
MONTSERRAT	MS
MOROCCO	MA
MOZAMBIQUE	MZ

MYANMAR	MM
NAMIBIA	NA
NAURU	NR
NEPAL	NP
NETHERLANDS	NL
NETHERLANDS ANTILLES	AN
NEW CALEDONIA	NC
NEW ZEALAND	NZ
NICARAGUA	NI
NIGER	NE
NIGERIA	NG
NIUE	NU
NORFOLK ISLAND	NF
NORTHERN MARIANA ISLANDS	MP
NORWAY	NO
OMAN	OM
PAKISTAN	PK
PALAU	PW
PALESTINIAN TERRITORY, OCCUPIED	PS
PANAMA	PA
PAPUA NEW GUINEA	PG
PARAGUAY	PY
PERU	PE
PHILIPPINES	PH
PITCAIRN	PN
POLAND	PL
PORTUGAL	PT
PUERTO RICO	PR
QATAR	QA
RÉUNION	RE
ROMANIA	RO
RUSSIAN FEDERATION	RU
RWANDA	RW
SAINT HELENA	SH
SAINT KITTS AND NEVIS	KN
SAINT LUCIA	LC
SAINT PIERRE AND MIQUELON	PM
SAINT VINCENT AND THE GRENADINES	VC
SAMOA	WS
SAN MARINO	SM
SAO TOME AND PRINCIPE	ST
SAUDI ARABIA	SA
SENEGAL	SN
SERBIA	RS
SEYCHELLES	SC
SIERRA LEONE	SL
SINGAPORE	SG
SLOVAKIA	SK
SLOVENIA	SI
SOLOMON ISLANDS	SB
SOMALIA	SO
SOUTH AFRICA	ZA
SOUTH GEORGIA AND THE SOUTH SANDWICH ISLANDS	GS
SPAIN	ES
SRI LANKA	LK
SUDAN	SD
SURINAME	SR

SVALBARD AND JAN MAYEN	SJ
SWAZILAND	SZ
SWEDEN	SE
SWITZERLAND	CH
SYRIAN ARAB REPUBLIC	SY
TAIWAN	TW
TAJIKISTAN	TJ
TANZANIA, UNITED REPUBLIC OF	TZ
THAILAND	TH
TIMOR-LESTE	TL
TOGO	TG
TOKELAU	TK
TONGA	TO
TRINIDAD AND TOBAGO	TT
TUNISIA	TN
TURKEY	TR
TURKMENISTAN	TM
TURKS AND CAICOS ISLANDS	TC
TUVALU	TV
UGANDA	UG
UKRAINE	UA
UNITED ARAB EMIRATES	AE
UNITED KINGDOM	GB
UNITED STATES	US
UNITED STATES MINOR OUTLYING ISLANDS	UM
URUGUAY	UY
UZBEKISTAN	UZ
VANUATU	VU
VENEZUELA	VE
VIET NAM	VN
VIRGIN ISLANDS, BRITISH	VG
VIRGIN ISLANDS, U.S.	VI
WALLIS AND FUTUNA	WF
WESTERN SAHARA	EH
YEMEN	YE
ZAMBIA	ZM
ZIMBABWE	ZW

□ Prev Next □

HTML Атрибуты событий. Справочник

□ Prev Next □

Глобальные атрибуты событий

HTML 4 добавил возможность позволить событиям запускать действия в браузере, например, запуск **JavaScript**, когда пользователь нажимает на элемент.

Чтобы узнать больше о программировании событий, посетите JavaScript Учебник.

Ниже приведены глобальные атрибуты событий, которые можно добавить к **HTML** элементам, чтобы определить действия событий.

= Новые атрибуты событий в **HTML5**.

Окно атрибутов событий

События, что инициируются для объекта окна (применяются к тегу <body>):

Атрибут	Значение	Описание
onafterprint	*script*	Скрипт будет запущен после печати документа
onbeforeprint	*script*	Скрипт будет запущен перед печатью документа
onbeforeunload	*script*	Скрипт будет запущен когда документ будет выгружен
onerror	*script*	Скрипт будет запущен при возникновении ошибки
onhashchange	*script*	Скрипт будет запущен когда произошло изменение привязки части URL-адреса
onload	*script*	Запускается после завершения загрузки страницы
onmessage	*script*	Скрипт будет запущен когда сообщение срабатывает
onoffline	*script*	Скрипт будет запущен когда браузер начинает работать в автономном режиме
ononline	*script*	Скрипт будет запущен когда браузер начинает работать в Интернете
onpagehide	*script*	Скрипт будет запущен когда пользователь покидает страницу
onpageshow	*script*	Скрипт будет запущен когда пользователь переходит на страницу
onpopstate	*script*	Скрипт будет запущен когда история окна меняется
onresize	*script*	Запускается, когда меняется размер окна веб-браузера
onstorage	*script*	Скрипт будет запущен когда область веб-хранилища обновляется
onunload	*script*	Запускается после выгрузки страницы (или закрытия окна веб-браузера)

События формы

События, вызванные действиями в форме HTML (применяется к почти всем элементам HTML, но наиболее часто используется в элементах формы):

Атрибут	Значение	Описание
onblur	*script*	Запускает момент, когда элемент теряет фокус
onchange	*script*	Запускает момент, когда меняется значение элемента
oncontextmenu	*script*	Скрипт запускается, когда срабатывает контекстное меню
onfocus	*script*	Запускает момент, когда элемент получает фокус
oninput	*script*	Скрипт будет запущен, когда элемент получает ввод пользователя
oninvalid	*script*	Скрипт будет запущен когда элемент не действителен
onreset	*script*	Срабатывает при нажимании кнопки "Сброс" в форме
onsearch	*script*	Запускается, когда пользователь записывает что-то в поле поиска (для <input="search">)
onselect	*script*	Запускается после выбора текста в элементе
onsubmit	*script*	Запускается во время отправки формы

События с помощью клавиатуры

Атрибут	Значение	Описание
onkeydown	*script*	Запускается, когда пользователь нажимает клавишу
onkeypress	*script*	Запускается, когда пользователь нажимает клавишу
onkeyup	*script*	Запускается, когда пользователь отпускает клавишу

События с помощью мыши

Атрибут	Значение	Описание
onclick	*script*	Запускается при нажимании мыши на элементе
ondblclick	*script*	Запускается при двойном клике мыши на элементе
onmousedown	*script*	Запускается, когда кнопка мыши нажата на элементе
onmousemove	*script*	Запускается, когда указатель мыши перемещается над элементом
onmouseout	*script*	Запускается, когда указатель мыши выходит за пределы элемента
onmouseover	*script*	Запускается, когда указатель мыши перемещается над элементом
onmouseup	*script*	Запускается, когда кнопка мишы отпускается над элементом
onmousewheel	*script*	Устаревший. Используйте атрибут onwheel вместо этого

onwheel	*script*	Запускается, когда колесо мыши прокручивается вверх или вниз над элементом

События перетягивания

Атрибут	Значение	Описание
ondrag	*script*	Скрипт будет запущен когда элемент перетягивается
ondragend	*script*	Скрипт будет запущен после завершения операции перетягивания
ondragenter	*script*	Скрипт буде запущен когда элемент перетягивается к действительной цели скидывания
ondragleave	*script*	Скрипт будет запущен когда элемент оставляет действительную цель скидывания
ondragover	*script*	Скрипт будет запущен когда элемент перетягивается через действительную цель скидывания
ondragstart	*script*	Скрипт будет запущен в начале перетягивания
ondrop	*script*	Скрипт будет запущен когда перетягивается элемент
onscroll	*script*	Скрипт будет запущен когда полоса прокрутки элемента прокручивается

События буфера обмена

Атрибут	Значение	Описание
oncopy	*script*	Запускается, когда пользователь копирует содержимое элемента
oncut	*script*	Запускается, когда пользователь сокращает содержимое элемента
onpaste	*script*	Запускается, когда пользователь вставляет определённое содержание в элемент

События Медиа

События, которые запускаются средствами медиа, такими как видео, изображения и аудио (применяются ко всем элементам HTML, но наиболее часто встречаются в медиа-элементах, например <audio>, <embed>, <img>, <object> и <video>).

Совет: Посетите HTML Аудио и Видео DOM Справочник для получения более подробной информации.

Атрибут	Значение	Описание
onabort	*script*	Скрипт будет запущен во время отмены
oncanplay	*script*	Скрипт будет запущен когда файл готов к началу воспроизведения (когда он буферизирован для начала)
oncanplaythrough	*script*	Скрипт будет запущен когда файл можно воспроизвести до конца, не останавливаясь для буферизации
oncuechange	*script*	Скрипт будет запущен когда меняется сигнал в элементе <track>
ondurationchange	*script*	Скрипт будет запущен когда меняется продолжительность медиа
onemptied	*script*	Скрипт будет запущен когда случится что-то плохое и файл неожиданно становится недоступным (например, неожиданно отключается)
onended	*script*	Скрипт будет запущен когда медиа достигли конца (полезное событие для сообщений типа "спасибо за прослушивание/просмотр")
onerror	*script*	Скрипт будет запущен когда возникает ошибка во время загрузки файла
onloadeddata	*script*	Скрипт будет запущен когда медиа-данные загружены
onloadedmetadata	*script*	Скрипт будет запущен когда загружаются метаданные (например, размеры и продолжительность)
onloadstart	*script*	Скрипт будет запущен когда файл начинает загружаться до того, как что-то действительно загружено
onpause	*script*	Скрипт будет запущен когда медиа приостановлено или пользователем, или программно
onplay	*script*	Скрипт будет запущен когда медиа готово к началу воспроизведения
onplaying	*script*	Скрипт будет запущен когда медиа фактически начало воспроизводиться
onprogress	*script*	Скрипт будет запущен когда браузер находится в процессе получения медиаданных
onratechange	*script*	Скрипт будет запущен каждый раз, когда скорость воспроизведения меняется (например, когда пользователь переключается на режим медленной или быстрой перемотки вперёд)
onseeked	*script*	Скрипт будет запущен когда атрибут ищет значение false, что указывает, что поиск закончился

onseeking	*script*	Скрипт будет запущен когда атрибут ищет значение true, что указывает на то, что поиск является активным
onstalled	*script*	Скрипт будет запущен когда браузер не может получить данные медиа по какой-либо причине
onsuspend	*script*	Скрипт будет запущен при получении медиаданных, останавливается перед полной загрузкой по какой-либо причине
ontimeupdate	*script*	Скрипт будет запущен когда меняется позиция воспроизведения (например, когда пользователь быстро переходит в другую точку на медиа)
onvolumechange	*script*	Скрипт будет запущен каждый раз, когда меняется громкость (включая настройки громкости на "выключение звука")
onwaiting	*script*	Скрипт будет запущен когда медиа приостановлено, но ожидается, что его будет возобновлено (например, когда медиа приостанавливается для буферизации большого объёма данных)

Другие События

Атрибут	Значение	Описание
ontoggle	*script*	Запустится когда пользователь откроет или закроет элемент <details>

Prev Next

HTML Элементы и валидные DOCTYPES. Справочник

Prev Next

HTML Элементы - Валидные DOCTYPE

В приведённой ниже таблице приведены все HTML элементы и показано, что каждый элемент отображается в !DOCTYPE.

		HTML 4.01 / XHTML 1.0			
Тег	HTML5	Transitional	Strict	Frameset	XHTML 1.1
<a>	Yes	Yes	Yes	Yes	Yes
<abbr>	Yes	Yes	Yes	Yes	Yes
<acronym>	No	Yes	Yes	Yes	Yes
<address>	Yes	Yes	Yes	Yes	Yes
<applet>	No	Yes	No	Yes	No
<area>	Yes	Yes	Yes	Yes	No
<article>	Yes	No	No	No	No
<aside>	Yes	No	No	No	No
<audio>	Yes	No	No	No	No
<b>	Yes	Yes	Yes	Yes	Yes
<base>	Yes	Yes	Yes	Yes	Yes
<basefont>	No	Yes	No	Yes	No
<bdi>	Yes	No	No	No	No
<bdo>	Yes	Yes	Yes	Yes	No
<big>	No	Yes	Yes	Yes	Yes
<blockquote>	Yes	Yes	Yes	Yes	Yes
<body>	Yes	Yes	Yes	Yes	Yes
 	Yes	Yes	Yes	Yes	Yes
<button>	Yes	Yes	Yes	Yes	Yes
<canvas>	Yes	No	No	No	No
<caption>	Yes	Yes	Yes	Yes	Yes
	No	Yes	No	Yes	No
<cite>	Yes	Yes	Yes	Yes	Yes
<code>	Yes	Yes	Yes	Yes	Yes
<col>	Yes	Yes	Yes	Yes	No

<colgroup>	Yes	Yes	Yes	Yes	No
<datalist>	Yes	No	No	No	No
<dd>	Yes	Yes	Yes	Yes	Yes
<del>	Yes	Yes	Yes	Yes	No
<details>	Yes	No	No	No	No
<dfn>	Yes	Yes	Yes	Yes	Yes
<dialog>	Yes	No	No	No	No
<dir>	No	Yes	No	Yes	No
	Yes	Yes	Yes	Yes	Yes
<dl>	Yes	Yes	Yes	Yes	Yes
<dt>	Yes	Yes	Yes	Yes	Yes
	Yes	Yes	Yes	Yes	Yes
<embed>	Yes	No	No	No	No
<fieldset>	Yes	Yes	Yes	Yes	Yes
<figcaption>	Yes	No	No	No	No
<figure>	Yes	No	No	No	No
<font>	No	Yes	No	Yes	No
<footer>	Yes	No	No	No	No
<form>	Yes	Yes	Yes	Yes	Yes
<frame>	No	No	No	Yes	No
<frameset>	No	No	No	Yes	No
<h1> to <h6>	Yes	Yes	Yes	Yes	Yes
<head>	Yes	Yes	Yes	Yes	Yes
<header>	Yes	No	No	No	No
<hr>	Yes	Yes	Yes	Yes	Yes
<html>	Yes	Yes	Yes	Yes	Yes
<i>	Yes	Yes	Yes	Yes	Yes
<iframe>	Yes	Yes	No	Yes	No
<img>	Yes	Yes	Yes	Yes	Yes
<input>	Yes	Yes	Yes	Yes	Yes
<ins>	Yes	Yes	Yes	Yes	No
<kbd>	Yes	Yes	Yes	Yes	Yes
<label>	Yes	Yes	Yes	Yes	Yes
<legend>	Yes	Yes	Yes	Yes	Yes
	Yes	Yes	Yes	Yes	Yes
<link>	Yes	Yes	Yes	Yes	Yes
<main>	Yes	No	No	No	No
<map>	Yes	Yes	Yes	Yes	No
<mark>	Yes	No	No	No	No
<meta>	Yes	Yes	Yes	Yes	Yes
<meter>	Yes	No	No	No	No
<nav>	Yes	No	No	No	No
<noframes>	No	Yes	No	Yes	No
<noscript>	Yes	Yes	Yes	Yes	Yes
<object>	Yes	Yes	Yes	Yes	Yes
	Yes	Yes	Yes	Yes	Yes
<optgroup>	Yes	Yes	Yes	Yes	Yes
<option>	Yes	Yes	Yes	Yes	Yes
<output>	Yes	No	No	No	No
<p>	Yes	Yes	Yes	Yes	Yes
<param>	Yes	Yes	Yes	Yes	Yes
<pre>	Yes	Yes	Yes	Yes	Yes
<progress>	Yes	No	No	No	No
<q>	Yes	Yes	Yes	Yes	Yes
<rp>	Yes	No	No	No	No
<rt>	Yes	No	No	No	No

<ruby>	Yes	No	No	No	No
<s>	Yes	Yes	No	Yes	No
<samp>	Yes	Yes	Yes	Yes	Yes
<script>	Yes	Yes	Yes	Yes	Yes
<section>	Yes	No	No	No	No
<select>	Yes	Yes	Yes	Yes	Yes
<small>	Yes	Yes	Yes	Yes	Yes
<source>	Yes	No	No	No	No
	Yes	Yes	Yes	Yes	Yes
<strike>	No	Yes	No	Yes	No
	Yes	Yes	Yes	Yes	Yes
<style>	Yes	Yes	Yes	Yes	Yes
<sub>	Yes	Yes	Yes	Yes	Yes
<summary>	Yes	No	No	No	No
<sup>	Yes	Yes	Yes	Yes	Yes
<table>	Yes	Yes	Yes	Yes	Yes
<tbody>	Yes	Yes	Yes	Yes	No
<td>	Yes	Yes	Yes	Yes	Yes
<textarea>	Yes	Yes	Yes	Yes	Yes
<tfoot>	Yes	Yes	Yes	Yes	No
<th>	Yes	Yes	Yes	Yes	Yes
<thead>	Yes	Yes	Yes	Yes	No
<time>	Yes	No	No	No	No
<title>	Yes	Yes	Yes	Yes	Yes
<tr>	Yes	Yes	Yes	Yes	Yes
<track>	Yes	No	No	No	No
<tt>	No	Yes	Yes	Yes	Yes
<u>	Yes	Yes	No	Yes	No
	Yes	Yes	Yes	Yes	Yes
<var>	Yes	Yes	Yes	Yes	Yes
<video>	Yes	No	No	No	No
<wbr>	Yes	No	No	No	No

□ Prev Next □

Http сообщения на веб-странице. Справочник

□ Prev Next □

Когда браузер запрашивает услугу с веб-сервера, может произойти ошибка.

Это список сообщений о состоянии HTTP, которые могут быть возвращены:

1xx: Информация

Сообщение:	Описание:
100 Продолжить	Сервер получил заголовки запроса, и клиент должен приступить к отправке тела запроса
101 Протоколы переключения	Заказчик попросил сервер переключить протоколы
103 Контрольная точка	Используется в предложении о возобновляемых запросах для возобновления прерванных запросов PUT или POST

2xx: Успешный

Сообщение:	Описание:

200 OK	Запрос в порядке (это стандартный ответ для успешных запросов HTTP)
201 Создан	Запрос был выполнен, и создан новый ресурс
202 Принято	Запрос принят к обработке, но обработка не завершена
203 Неавторизованная информация	Запрос был успешно обработан, но возвращает информацию, которая может быть из другого источника
204 Без содержания	Запрос был успешно обработан, но не возвращает никакого содержания
205 Сбросить содержимое	Запрос был успешно обработан, но не возвращает никакого содержимого и требует, чтобы запрашивающая сторона сбросила представление документа
206 Частичное содержание	Сервер доставляет только часть ресурса из-за заголовка диапазона, отправленного клиентом

3xx: Перенаправление

Сообщение:	Описание:
300 Множественный выбор	Список ссылок. Пользователь может выбрать ссылку и перейти в это место. Максимум пять адресов
301 Перемещено постоянно	Запрашиваемая страница перемещена на новый URL
302 Найдено	Запрашиваемая страница временно перемещена на новый URL
303 Смотрите Другое	Запрошенную страницу можно найти по другому URL
304 Не модифицировано	Указывает, что запрошенная страница не была изменена с момента последнего запроса
306 Переключить прокси	*Больше не используется*
307 Временный редирект	Запрашиваемая страница временно перемещена на новый URL
308 Резюме неполное	Используется в предложении о возобновляемых запросах для возобновления прерванных запросов PUT или POST

4xx: Ошибка клиента

Сообщение:	Описание:
400 Неверный запрос	Запрос не может быть выполнен из-за неправильного синтаксиса
401 Неразрешенный	Запрос был законным, но сервер отказывается отвечать на него. Используется, когда аутентификация возможна, но не прошла или ещё не была предоставлена
402 Требуется оплата	*Зарезервировано для будущего использования*
403 Запрещено	Запрос был законным, но сервер отказывается отвечать на него.
404 Не обнаружено	Запрашиваемая страница не может быть найдена, но может быть снова доступна в будущем
405 Метод не разрешен	Был сделан запрос на странице с использованием метода запроса, не поддерживаемого этой страницей
406 Неприемлимо	Сервер может генерировать только тот ответ, который не принят клиентом
407 Требуется проверка подлинности прокси	Клиент должен сначала аутентифицировать себя с прокси
408 Тайм-аут запроса	Тайм-аут сервера в ожидании запроса
409 Конфликт	Запрос не может быть выполнен из-за конфликта в запросе
410 Потеряно	Запрашиваемая страница больше не доступна
411 Требуемая длина	«Длина содержимого» не определена. Сервер не примет запрос без него
412 Предварительное условие не выполнено	Предварительное условие, указанное в запросе, оценивается сервером как ложное
413 Слишком большой объект запроса	Сервер не примет запрос, так как объект запроса слишком большой
414 Слишком длинный запрос URI	Сервер не примет запрос, потому что URL слишком длинный. Происходит при преобразовании запроса POST в запрос GET с длинной информацией о запросе
415 Неподдерживаемый тип носителя	Сервер не примет запрос, потому что тип носителя не поддерживается
416 Запрошенный диапазон не удовлетворяется	Клиент запросил часть файла, но сервер не может предоставить эту часть

417 Ожидание не удалось	Сервер не может удовлетворить требования поля ожидать заголовок запроса

5xx: Ошибка сервера

Сообщение:	Описание:
500 Внутренняя ошибка сервера	Общее сообщение об ошибке, которое появляется, когда более подходящее сообщение не подходит
501 Не реализовано	Сервер либо не распознает метод запроса, либо ему не хватает возможности выполнить запрос
502 Плохой шлюз	Сервер действовал как шлюз или прокси и получил неверный ответ от вышестоящего сервера
503 Сервис недоступен	Сервер в данный момент недоступен (перегружен или выключен)
504 Время ожидания шлюза	Сервер действовал как шлюз или прокси и не получил своевременного ответа от вышестоящего сервера.
505 Версия HTTP не поддерживается	Сервер не поддерживает версию протокола HTTP, используемую в запросе
511 Требуется сетевая аутентификация	Клиент должен пройти аутентификацию, чтобы получить доступ к сети

□ Prev Next □

Http методы запроса на веб-странице.

□ Prev Next □

Два наиболее распространенных метода HTTP: GET и POST.

Что такое HTTP?

Hypertext Transfer Protocol - Протокол передачи гипертекста (HTTP) предназначен для обеспечения связи между клиентами и серверами.

HTTP работает как протокол запроса-ответа между клиентом и сервером.

Веб-браузер может быть клиентом, а приложение на компьютере, на котором размещен веб-сайт, может быть сервером.

Пример: клиент (браузер) отправляет HTTP-запрос на сервер; затем сервер возвращает ответ клиенту. Ответ содержит информацию о состоянии запроса и может также содержать запрошенный контент.

HTTP Методы

- **GET**
- **POST**
- **PUT**
- **HEAD**
- **DELETE**
- **PATCH**
- **OPTIONS**

Метод GET

GET используется для запроса данных от указанного ресурса.

GET - один из самых распространенных методов HTTP.

Обратите внимание, что строка запроса (пары имя/значение) отправляется в URL-адресе запроса GET:

/test/demo_form.php?name1=value1&name2=value2

Некоторые другие заметки о запросах GET:

- GET-запросы могут быть кэшированы
- GET запросы остаются в истории браузера
- GET запросы могут быть добавлены в закладки
- Запросы GET никогда не должны использоваться при работе с конфиденциальными данными.
- GET-запросы имеют ограничения по длине
- GET-запросы используются только для запроса данных (не изменяются)

Метод POST

POST используется для отправки данных на сервер для создания/обновления ресурса.

Данные, отправленные на сервер с помощью POST, хранятся в теле запроса HTTP.:

```
POST /test/demo_form.php HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

POST является одним из самых распространенных методов HTTP.

Некоторые другие заметки о запросах POST:

- POST-запросы никогда не кэшируются
- POST-запросы не сохраняются в истории браузера
- POST-запросы не могут быть добавлены в закладки
- POST-запросы не имеют ограничений по длине данных

Метод PUT

PUT используется для отправки данных на сервер для создания / обновления ресурса.

Разница между POST и PUT заключается в том, что PUT-запросы являются идентичными. То есть, вызов одного и того же запроса PUT несколько раз всегда будет приводить к одному и тому же результату. Напротив, вызов POST-запроса неоднократно имеет побочные эффекты от создания одного и того же ресурса несколько раз.

Метод HEAD

HEAD почти идентичен GET, но без тела ответа.

Другими словами, если GET/users возвращает список пользователей, то HEAD/users сделает такой же запрос, но не вернет список пользователей.

Запросы HEAD полезны для проверки того, что будет возвращен запрос GET, перед тем, как фактически выполнить запрос GET, например, перед загрузкой большого файла или тела ответа.

Метод DELETE

Метод DELETE удаляет указанный ресурс.

Метод OPTIONS

Метод OPTIONS описывает параметры связи для целевого ресурса.

Сравнение GET и POST

В следующей таблице сравниваются два метода HTTP: GET и POST.

	GET	**POST**

Кнопка НАЗАД/Перезагрузить	Безвредный	Данные будут повторно отправлены (браузер должен предупредить пользователя о том, что данные должны быть повторно отправлены)
Закладки	Может быть в закладках	Не может быть в закладках
Кэширование (сохранённая копия)	Может быть кэширование	Не может быть кэширования
Тип кодирования	application/x-www-form-urlencoded	application/x-www-form-urlencoded или multipart/form-data. Используйте многочастное кодирование для двоичных данных
История	Параметры остаются в истории браузера	Параметры не сохраняются в истории браузера
Ограничения на длину данных	Да, при отправке данных метод GET добавляет данные в URL; и длина URL-адреса ограничена (максимальная длина URL-адреса составляет 2048 символов)	Нет ограничений
Ограничения на тип данных	Разрешены только символы ASCII	Нет ограничений. Двоичные данные также разрешены
Безопасность	GET менее безопасен по сравнению с POST, потому что отправленные данные являются частью URL Никогда не используйте GET при отправке паролей или другой конфиденциальной информации!	POST немного безопаснее, чем GET, поскольку параметры не сохраняются в истории браузера или в журналах веб-сервера (в логах).
Видимость	Данные видны всем в URL	Данные не отображаются в URL

□ Prev Next □

Горячие клавиши

□ Prev Next □

Экономьте время, используя сочетания клавиш.

Сочетания клавиш для Windows и Mac

Сочетания клавиш часто используются в современных операционных системах и компьютерных программах.

Использование сочетаний клавиш может сэкономить вам много времени.

Основные сочетания

Описание	Windows	Mac OS
меню Edit	Alt + E	Ctrl + F2 + F
меню File	Alt + F	Ctrl + F2 + E
меню View	Alt + V	Ctrl + F2 + V
Выделить весь текст	Ctrl + A	Cmd + A
Копировать текст	Ctrl + C	Cmd + C
Найти текст	Ctrl + F	Cmd + F
Найти и заменить текст	Ctrl + H	Cmd + F
Новый документ	Ctrl + N	Cmd + N
Открыть файл	Ctrl + O	Cmd + O
Параметры печати	Ctrl + P	Cmd + P
Сохранить файл	Ctrl + S	Cmd + S
Вставить текст	Ctrl + V	Cmd + V
Вырезать текст	Ctrl + X	Cmd + X

Повторить текст	Ctrl + Y	Shift + Cmd + Z
Отменить текст	Ctrl + Z	Cmd + Z

Редактирование текста

Описание	Windows	Mac OS
Движение курсора		
Переход направо или к началу следующего перевода строки	Стрелка вправо	Стрелка вправо
Переход влево или в конец предыдущего разрыва строки	Стрелка влево	Стрелка влево
Подняться на один ряд	Стрелка вверх	Стрелка вверх
Опуститься на один ряд	Стрелка вниз	Стрелка вниз
Перейти к началу текущей строки	Home	Cmd + Стрелка влево
Перейти в конец текущей строки	End	Cmd + Стрелка вправо
Перейти к началу документа	Ctrl + Home	Cmd + Стрелка вверх
Перейти в конец документа	Ctrl + End	Cmd + Стрелка вниз
Переместиться на один фрейм вверх	Page Up	Fn + Стрелка вверх
Переместиться на один фрейм вниз	Page Down	Fn + Стрелка вниз
Перейти к началу предыдущего слова	Ctrl + Стрелка влево	Option + Стрелка влево
Перейти к началу следующего слова	Ctrl + Стрелка вправо	Option + Стрелка вправо
Перейти к началу перевода строки	Ctrl + Стрелка вверх	Cmd + Стрелка влево
Перейти к концу строки	Ctrl + Стрелка вниз	Cmd + Стрелка вправо
Выбор текста		
Выберите символы слева	Shift + Стрелка влево	Shift + Стрелка влево
Выберите символы справа	Shift + Стрелка вправо	Shift + Стрелка вправо
Выберите линии вверх	Shift + Стрелка вверх	Shift + Стрелка вверх
Выберите линии вниз	Shift + Стрелка вниз	Shift + Стрелка вниз
Выберите слова слева	Shift + Ctrl + Left	Shift + Opt + Left
Выберите слова справа	Shift + Ctrl + Right	Shift + Opt + Right
Выберите абзацы слева	Shift + Ctrl + Up	Shift + Opt + Up
Выберите абзацы справа	Shift + Ctrl + Down	Shift + Opt + Down
Выделите текст между курсором и началом текущей строки	Shift + Home	Cmd + Shift + Стрелка влево
Выделите текст между курсором и концом текущей строки	Shift + End	Cmd + Shift + Стрелка вправо
Выделите текст между курсором и началом документа	Shift + Ctrl + Home	Cmd + Shift + Стрелка вверх или Cmd + Shift + Fn + Стрелка влево
Выделите текст между курсором и концом документа	Shift + Ctrl + End	Cmd + Shift + Стрелка вниз или Cmd + Shift + Fn + Стрелка вправо
Выберите один фрейм за раз текста над курсором	Shift + Page Up	Shift + Fn + Стрелка вверх
Выберите один фрейм за раз текста под курсором	Shift + Page Down	Shift + Fn + Стрелка вниз
Выделить весь текст	Ctrl + A	Cmd + A
Найти текст	Ctrl + F	Cmd + F
Форматирование текста		
Сделать выделенный текст жирным	Ctrl + B	Cmd + B
Сделать выделенный текст курсивом	Ctrl + I	Cmd + I
Подчеркнуть выделенный текст	Ctrl + U	Cmd + U
Сделать выделенный текст надстрочным (в верхнем индексе)	Ctrl + Shift + =	Cmd + Shift + =
Сделать выделенный текст подстрочным (в нижнем индексе)	Ctrl + =	Cmd + =
Редактирование текста		
Удалить символы слева	Backspace	Backspace
Удалить символы справа	Delete	Fn + Backspace
Удалить слова справа	Ctrl + Del	Cmd + Backspace

Удалить слова слева	Ctrl + Backspace	Cmd + Fn + Backspace
Отступ	Tab	Tab
Выступ	Shift + Tab	Shift + Tab
Копировать текст	Ctrl + C	Cmd + C
Найти и заменить текст	Ctrl + H	Cmd + F
Вставить текст	Ctrl + V	Cmd + V
Вырезать текст	Ctrl + X	Cmd + X
Повторить текст	Ctrl + Y	Shift + Cmd + Z
Отменить текст	Ctrl + Z	Cmd + Z

Веб браузеры

Описание	Windows	Mac OS
Навигация		
Прокрутите фрейм вниз	Space или Page Down	Space или Fn + Стрелка вниз
Прокрутите фрейм вверх	Shift + Space или Page Up	Shift + Space или Fn + Стрелка вверх
Перейти в конец страницы	End	Cmd + Стрелка вниз
Перейти в начало страницы	Home	Cmd + Стрелка вверх
Вернуться	Alt + Стрелка влево или Backspace	Cmd + Стрелка влево
Идти вперёд	Alt + Стрелка вправо или Shift + Backspace	Cmd + Стрелка вправо
Обновить веб-страницу	F5	Cmd + R
Обновить веб-страницу (без кэша)	Ctrl + F5	Cmd + Shift + R
Остановить	Esc	Esc
Включить полноэкранный режим	F11	Cmd + Shift + F
Приблизить/Увеличить	Ctrl + +	Cmd + +
Уменьшить	Ctrl + -	Cmd + -
Zoom 100% (по умолчанию)	Ctrl + 0	Cmd + 0
Открыть домашнюю страницу	Alt + Home	Option + Home или Option + Fn + Стрелка влево
Найти текст	Ctrl + F	Cmd + F
Tab / Управление окнами		
Откройте новую вкладку	Ctrl + T	Cmd + T
Закрыть текущую вкладку	Ctrl + W	Cmd + W
Закрыть все вкладки	Ctrl + Shift + W	Cmd + Q
Закрыть все вкладки, кроме текущей	Ctrl + Alt + F4	Cmd + Opt + W
Перейти в следующую вкладку	Ctrl + Tab	Control + Tab или Cmd + Shift + Стрелка вправо
Перейти к предыдущей вкладке	Ctrl + Shift + Tab	Shift + Control + Tab или Cmd + Shift + Стрелка влево
Перейти на конкретный номер вкладки	Ctrl + 1-8	Cmd + 1-8
Перейти к последней вкладке	Ctrl + 9	Cmd + 9
Открыть последнюю закрытую вкладку	Ctrl + Shift + T	Cmd + Shift + T
Открыть новое окно	Ctrl + N	Cmd + N
Закрыть текущее окно	Alt + F4	Cmd + W
Перейти к следующему окну	Alt + Tab	Cmd + Tab
Перейти к предыдущему окну	Alt + Shift + Tab	Cmd + Shift + Tab
Открыть последнее закрытое окно	Ctrl + Shift + N	
Открыть ссылки в новой вкладке в фоновом режиме	Ctrl + Click	Cmd + Click
Открывать ссылки в новой вкладке на переднем плане	Ctrl + Shift + Click	Cmd + Shift + Click
Распечатать текущую веб-страницу	Ctrl + P	Cmd + P
Сохранить текущую веб-страницу	Ctrl + S	Cmd + S

Адресная строка		
Цикл между панелью инструментов, панелью поиска и элементами страницы	Tab	Tab
Перейти в адресную строку браузера	Ctrl + L или Alt + D	Cmd + L
Сфокусируйтесь и выберите панель поиска браузера	Ctrl + E	Cmd + E / Cmd + K
Откройте адресную строку в новой вкладке	Alt + Enter	Opt + Enter
Показать список ранее введенных адресов	F4	
Добавить "www." в начало и ".com" в конец текста, набранного в адресной строке (например, введите «w3schools» и нажмите Ctrl + Enter, чтобы открыть «www.w3schools.com»)	Ctrl + Enter	Cmd + Enter или Control + Enter

Закладки		
Открыть меню закладок	Ctrl + B	Cmd + B
Добавить закладку для текущей страницы	Ctrl + D	Cmd + Opt + B или Cmd + Shift + B
Открыть историю просмотра	Ctrl + H	Cmd + Shift + H или Cmd + Y
Открыть историю загрузок	Ctrl + J	Cmd + J или Cmd + Shift + J

Скриншоты

Описание	Windows	Mac OS
Сохранить скриншот всего экрана в файл		Cmd + Shift + 3
Скопировать скриншот всего экрана в буфер обмена	PrtScr (Print Screen) или Ctrl + PrtScr	Cmd + Ctrl + Shift + 3
Сохранить скриншот окна как файл		Cmd + Shift + 4, затем Space
Скопировать скриншот окна в буфер обмена	Alt + PrtScr	Cmd + Ctrl + Shift + 4, затем Space
Скопировать скриншот нужной области в буфер обмена		Cmd + Ctrl + Shift + 4
Сохранить скриншот нужной области в файл		Cmd + Shift + 4

Заметка: Из-за различных настроек клавиатуры некоторые сочетания клавиш могут быть несовместимы с некоторыми пользователями.

□ Prev Next □

HTML Кодирование языка веб-страницы. Справочник

□ Prev Next □

ISO Коды языков

HTML атрибут `lang` может использоваться для объявления языка веб-страницы или части веб-страницы. Это должно помочь поисковым системам и браузерам.

В соответствии с рекомендацией W3C вы должны объявить основной язык для каждой веб-страницы с атрибутом `lang` внутри тега `<html>`, как здесь:

```
<html lang="en">
...
</html>
```

В XHTML язык объявляется внутри тега `<html>` следующим образом:

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
...
</html>
```

ISO 639-1 Коды языка

ISO 639-1 определяет сокращения для языков.

В HTML и XHTML они могут использоваться в атрибутах `lang` и `xml:lang`.

Смотрите также следующий справочник кодов стран.

Язык	ISO Код
Abkhazian	ab
Afar	aa
Afrikaans	af
Akan	ak
Albanian	sq
Amharic	am
Arabic	ar
Aragonese	an
Armenian	hy
Assamese	as
Avaric	av
Avestan	ae
Aymara	ay
Azerbaijani	az
Bambara	bm
Bashkir	ba
Basque	eu
Belarusian	be
Bengali (Bangla)	bn
Bihari	bh
Bislama	bi
Bosnian	bs
Breton	br
Bulgarian	bg
Burmese	my
Catalan	ca
Chamorro	ch
Chechen	ce
Chichewa, Chewa, Nyanja	ny
Chinese	zh
Chinese (Simplified)	zh-Hans
Chinese (Traditional)	zh-Hant
Chuvash	cv
Cornish	kw
Corsican	co
Cree	cr
Croatian	hr
Czech	cs
Danish	da
Divehi, Dhivehi, Maldivian	dv
Dutch	nl
Dzongkha	dz
English	en
Esperanto	eo
Estonian	et
Ewe	ee
Faroese	fo
Fijian	fj
Finnish	fi
French	fr
Fula, Fulah, Pulaar, Pular	ff
Galician	gl

Gaelic (Scottish)	gd
Gaelic (Manx)	gv
Georgian	ka
German	de
Greek	el
Greenlandic	kl
Guarani	gn
Gujarati	gu
Haitian Creole	ht
Hausa	ha
Hebrew	he
Herero	hz
Hindi	hi
Hiri Motu	ho
Hungarian	hu
Icelandic	is
Ido	io
Igbo	ig
Indonesian	id, in
Interlingua	ia
Interlingue	ie
Inuktitut	iu
Inupiak	ik
Irish	ga
Italian	it
Japanese	ja
Javanese	jv
Kalaallisut, Greenlandic	kl
Kannada	kn
Kanuri	kr
Kashmiri	ks
Kazakh	kk
Khmer	km
Kikuyu	ki
Kinyarwanda (Rwanda)	rw
Kirundi	rn
Kyrgyz	ky
Komi	kv
Kongo	kg
Korean	ko
Kurdish	ku
Kwanyama	kj
Lao	lo
Latin	la
Latvian (Lettish)	lv
Limburgish (Limburger)	li
Lingala	ln
Lithuanian	lt
Luga-Katanga	lu
Luganda, Ganda	lg
Luxembourgish	lb
Manx	gv
Macedonian	mk
Malagasy	mg
Malay	ms
Malayalam	ml

Maltese	mt
Maori	mi
Marathi	mr
Marshallese	mh
Moldavian	mo
Mongolian	mn
Nauru	na
Navajo	nv
Ndonga	ng
Northern Ndebele	nd
Nepali	ne
Norwegian	no
Norwegian bokmål	nb
Norwegian nynorsk	nn
Nuosu	ii
Occitan	oc
Ojibwe	oj
Old Church Slavonic, Old Bulgarian	cu
Oriya	or
Oromo (Afaan Oromo)	om
Ossetian	os
Pāli	pi
Pashto, Pushto	ps
Persian (Farsi)	fa
Polish	pl
Portuguese	pt
Punjabi (Eastern)	pa
Quechua	qu
Romansh	rm
Romanian	ro
Russian	ru
Sami	se
Samoan	sm
Sango	sg
Sanskrit	sa
Serbian	sr
Serbo-Croatian	sh
Sesotho	st
Setswana	tn
Shona	sn
Sichuan Yi	ii
Sindhi	sd
Sinhalese	si
Siswati	ss
Slovak	sk
Slovenian	sl
Somali	so
Southern Ndebele	nr
Spanish	es
Sundanese	su
Swahili (Kiswahili)	sw
Swati	ss
Swedish	sv
Tagalog	tl
Tahitian	ty
Tajik	tg

Tamil	ta
Tatar	tt
Telugu	te
Thai	th
Tibetan	bo
Tigrinya	ti
Tonga	to
Tsonga	ts
Turkish	tr
Turkmen	tk
Twi	tw
Uyghur	ug
Ukrainian	uk
Urdu	ur
Uzbek	uz
Venda	ve
Vietnamese	vi
Volapük	vo
Wallon	wa
Welsh	cy
Wolof	wo
Western Frisian	fy
Xhosa	xh
Yiddish	yi, ji
Yoruba	yo
Zhuang, Chuang	za
Zulu	zu

□ Prev Next □

Конвертер пикселей в единицу EM

□ Prev Next □

Инструмент ниже позволяет вам определить размеры em от пикселей (или наоборот).

Конвертер Pixel в EM

- Установить размер пикселя по умолчанию для body (обычно 16 пикселей)
- Затем преобразуйте значение пикселя в em в зависимости от размера пикселя по умолчанию
- Или преобразуйте значение em в пиксели на основе размера по умолчанию

Установить размер пикселя по умолчанию:

16

px

Конвертировать PX в EM:

px

Конвертировать EM в PX:

em

Конвертировать

Результат:

Размер шрифта body

В приведенной ниже таблице выберите размер основного шрифта в пикселях (px), чтобы отобразить полную таблицу преобразования px в em и в процент.

Совет: Размер шрифта по умолчанию обычно составляет 16 пикселей.

px	em	процент
5px	0.3125em	31.25%
6px	0.3750em	37.50%
7px	0.4375em	43.75%
8px	0.5000em	50.00%
9px	0.5625em	56.25%
10px	0.6250em	62.50%
11px	0.6875em	68.75%
12px	0.7500em	75.00%
13px	0.8125em	81.25%
14px	0.8750em	87.50%
15px	0.9375em	93.75%
16px	1.0000em	100.00%
17px	1.0625em	106.25%
18px	1.1250em	112.50%
19px	1.1875em	118.75%
20px	1.2500em	125.00%
21px	1.3125em	131.25%
22px	1.3750em	137.50%
23px	1.4375em	143.75%
24px	1.5000em	150.00%
25px	1.5625em	156.25%

В чем разница между PX, EM и Процентами?

Пиксель - это статическое измерение, а проценты и EM - относительные измерения. Размер EM или процентов зависит от его родителя. Если размер текста body составляет 16 пикселей, тогда 150% или 1,5 EM будут 24 пикселя (1,5*16). Смотрите на CSS Единицы для получения большего количества единиц измерения.

□ Prev Next □

HTML Глобальные атрибуты. Справочник

□ Prev Next □

HTML атрибуты предоставляют элементам смысл и контекст.

Глобальные атрибуты ниже можно использовать на **любом** HTML элементе.

HTML Глобальные Атрибуты

= Атрибут добавлен в HTML5.

Атрибут	Описание
accesskey	Указывает сочетание клавиш для активации/фокусировки элемента
class	Задает одно или несколько имен классов для элемента (ссылается на класс в таблице стилей)
contenteditable	Указывает, является ли содержимое элемента редактируемым или нет
contextmenu	Задает контекстное меню для элемента. Контекстное меню появляется, когда пользователь щелкает правой кнопкой мыши на элементе
data-*	Используется для хранения личных данных пользователя на странице или в приложении
dir	Задает направление текста для содержимого элемента
draggable	Указывает, является ли элемент перетаскиваемым или нет
dropzone	Указывает, копируются, перемещаются или связываются перетаскиваемые данные
hidden	Указывает, что элемент еще не релевантен или больше не имеет значения
id	Задает уникальный идентификатор элемента
lang	Задает язык содержимого элемента
spellcheck	Указывает, следует ли проверять орфографию и грамматику элемента
style	Задает встроенный стиль CSS для элемента
tabindex	Задает порядок табуляции элемента
title	Задает дополнительные сведения об элементе
translate	Указывает, следует ли переводить содержимое элемента

□ Prev Next □

HTML Кодирование URL адресов. Справочник

□ Prev Next □

Кодирование URL преобразует символы в формат, который может быть передан через Интернет.

URL (Uniform Resource Locator) - Унифицированный указатель информационного ресурса

Веб-браузеры запрашивают страницы с веб-серверов с помощью URL-адреса.

URL-адрес - это адрес веб-страницы, например: **https://www.w3schools.com.**

Кодирование URL (кодирование с процентами)

URL-адреса можно отправлять только через Интернет с помощью набора символов ASCII.

Поскольку URL-адреса часто содержат символы вне набора ASCII, URL-адрес должен быть преобразован в правильный формат ASCII.

Кодирование URL заменяет опасные символы ASCII на "%", за которыми следуют две шестнадцатеричные цифры.

URL-адреса не могут содержать пробелы. Кодирование URL обычно заменяет пробел знаком плюс (+) или %20.

Попробуйте сами

Если нажать кнопку Submit ("Отправить") ниже, браузер закодирует входной URL-адрес прежде, чем он будет отправлен на сервер. Страница на сервере отобразит полученные данные.

Hello Günter Submit

Попробуйте ввести другие данные и опять нажмите Submit ("Отправить") (*работает только на странице сайта*

https://www.w3schools.com/).

Функции кодирования URL-адресов

В JavaScript, PHP и ASP существуют функции, которые можно использовать для URL-кодирования строки.

PHP имеет функцию rawurlencode(), а ASP имеет функцию Server.URLEncode().

В JavaScript можно использовать функцию **encodeURIComponent()**.

Нажмите кнопку "URL Encode", чтобы увидеть, как функция JavaScript кодирует текст (работает только на сайте w3schools.com).

Hello Günter URL Encode

Примечание: Функция JavaScript кодирует пробел как %20.

ASCII Кодировки. Справочник

Ваш браузер будет кодировать входящие данные в соответствие набору символов, который используется на веб-странице.

Типичный (стандартный) набор символов (кодировка) в HTML5 - это UTF-8.

Символ	Из Windows-1252	Из UTF-8
space	%20	%20
!	%21	%21
"	%22	%22
#	%23	%23
$	%24	%24
%	%25	%25
&	%26	%26
'	%27	%27
(	%28	%28
)	%29	%29
*	%2A	%2A
+	%2B	%2B
,	%2C	%2C
-	%2D	%2D
.	%2E	%2E
/	%2F	%2F
0	%30	%30
1	%31	%31
2	%32	%32
3	%33	%33
4	%34	%34
5	%35	%35
6	%36	%36
7	%37	%37
8	%38	%38
9	%39	%39
:	%3A	%3A
;	%3B	%3B
<	%3C	%3C
=	%3D	%3D
>	%3E	%3E
?	%3F	%3F
@	%40	%40

A	%41	%41
B	%42	%42
C	%43	%43
D	%44	%44
E	%45	%45
F	%46	%46
G	%47	%47
H	%48	%48
I	%49	%49
J	%4A	%4A
K	%4B	%4B
L	%4C	%4C
M	%4D	%4D
N	%4E	%4E
O	%4F	%4F
P	%50	%50
Q	%51	%51
R	%52	%52
S	%53	%53
T	%54	%54
U	%55	%55
V	%56	%56
W	%57	%57
X	%58	%58
Y	%59	%59
Z	%5A	%5A
[	%5B	%5B
\	%5C	%5C
]	%5D	%5D
^	%5E	%5E
_	%5F	%5F
`	%60	%60
a	%61	%61
b	%62	%62
c	%63	%63
d	%64	%64
e	%65	%65
f	%66	%66
g	%67	%67
h	%68	%68
i	%69	%69
j	%6A	%6A
k	%6B	%6B
l	%6C	%6C
m	%6D	%6D
n	%6E	%6E
o	%6F	%6F
p	%70	%70
q	%71	%71
r	%72	%72
s	%73	%73
t	%74	%74
u	%75	%75
v	%76	%76
w	%77	%77
x	%78	%78
y	%79	%79

z	%7A	%7A
{	%7B	%7B
|	%7C	%7C
}	%7D	%7D
~	%7E	%7E
	%7F	%7F
`	%80	%E2%82%AC
	%81	%81
‚	%82	%E2%80%9A
ƒ	%83	%C6%92
„	%84	%E2%80%9E
…	%85	%E2%80%A6
†	%86	%E2%80%A0
‡	%87	%E2%80%A1
ˆ	%88	%CB%86
‰	%89	%E2%80%B0
Š	%8A	%C5%A0
‹	%8B	%E2%80%B9
Œ	%8C	%C5%92
	%8D	%C5%8D
Ž	%8E	%C5%BD
	%8F	%8F
	%90	%C2%90
‘	%91	%E2%80%98
’	%92	%E2%80%99
“	%93	%E2%80%9C
”	%94	%E2%80%9D
•	%95	%E2%80%A2
–	%96	%E2%80%93
—	%97	%E2%80%94
˜	%98	%CB%9C
™	%99	%E2%84
š	%9A	%C5%A1
›	%9B	%E2%80
œ	%9C	%C5%93
	%9D	%9D
ž	%9E	%C5%BE
Ÿ	%9F	%C5%B8
	%A0	%C2%A0
¡	%A1	%C2%A1
¢	%A2	%C2%A2
£	%A3	%C2%A3
¤	%A4	%C2%A4
¥	%A5	%C2%A5
¦	%A6	%C2%A6
§	%A7	%C2%A7
¨	%A8	%C2%A8
©	%A9	%C2%A9
ª	%AA	%C2%AA
«	%AB	%C2%AB
¬	%AC	%C2%AC
	%AD	%C2%AD
®	%AE	%C2%AE
¯	%AF	%C2%AF
°	%B0	%C2%B0
±	%B1	%C2%B1
²	%B2	%C2%B2

³	%B3	%C2%B3
´	%B4	%C2%B4
µ	%B5	%C2%B5
¶	%B6	%C2%B6
·	%B7	%C2%B7
¸	%B8	%C2%B8
¹	%B9	%C2%B9
º	%BA	%C2%BA
»	%BB	%C2%BB
¼	%BC	%C2%BC
½	%BD	%C2%BD
¾	%BE	%C2%BE
¿	%BF	%C2%BF
À	%C0	%C3%80
Á	%C1	%C3%81
Â	%C2	%C3%82
Ã	%C3	%C3%83
Ä	%C4	%C3%84
Å	%C5	%C3%85
Æ	%C6	%C3%86
Ç	%C7	%C3%87
È	%C8	%C3%88
É	%C9	%C3%89
Ê	%CA	%C3%8A
Ë	%CB	%C3%8B
Ì	%CC	%C3%8C
Í	%CD	%C3%8D
Î	%CE	%C3%8E
Ï	%CF	%C3%8F
Ð	%D0	%C3%90
Ñ	%D1	%C3%91
Ò	%D2	%C3%92
Ó	%D3	%C3%93
Ô	%D4	%C3%94
Õ	%D5	%C3%95
Ö	%D6	%C3%96
×	%D7	%C3%97
Ø	%D8	%C3%98
Ù	%D9	%C3%99
Ú	%DA	%C3%9A
Û	%DB	%C3%9B
Ü	%DC	%C3%9C
Ý	%DD	%C3%9D
Þ	%DE	%C3%9E
ß	%DF	%C3%9F
à	%E0	%C3%A0
á	%E1	%C3%A1
â	%E2	%C3%A2
ã	%E3	%C3%A3
ä	%E4	%C3%A4
å	%E5	%C3%A5
æ	%E6	%C3%A6
ç	%E7	%C3%A7
è	%E8	%C3%A8
é	%E9	%C3%A9
ê	%EA	%C3%AA
ë	%EB	%C3%AB

ì	%EC	%C3%AC
í	%ED	%C3%AD
î	%EE	%C3%AE
ï	%EF	%C3%AF
ð	%F0	%C3%B0
ñ	%F1	%C3%B1
ò	%F2	%C3%B2
ó	%F3	%C3%B3
ô	%F4	%C3%B4
õ	%F5	%C3%B5
ö	%F6	%C3%B6
÷	%F7	%C3%B7
ø	%F8	%C3%B8
ù	%F9	%C3%B9
ú	%FA	%C3%BA
û	%FB	%C3%BB
ü	%FC	%C3%BC
ý	%FD	%C3%BD
þ	%FE	%C3%BE
ÿ	%FF	%C3%BF

Справочник URL кодировок

Набор символов ASCII **%00-%1F** сначала разрабатывался для управления аппаратными устройствами.

Наборы символов не имеют никакого отношения к URL-адресам.

ASCII символы	Описание	URL-кодировка
NUL	null character	%00
SOH	start of header	%01
STX	start of text	%02
ETX	end of text	%03
EOT	end of transmission	%04
ENQ	enquiry	%05
ACK	acknowledge	%06
BEL	bell (ring)	%07
BS	backspace	%08
HT	horizontal tab	%09
LF	line feed	%0A
VT	vertical tab	%0B
FF	form feed	%0C
CR	carriage return	%0D
SO	shift out	%0E
SI	shift in	%0F
DLE	data link escape	%10
DC1	device control 1	%11
DC2	device control 2	%12
DC3	device control 3	%13
DC4	device control 4	%14
NAK	negative acknowledge	%15
SYN	synchronize	%16
ETB	end transmission block	%17
CAN	cancel	%18
EM	end of medium	%19
SUB	substitute	%1A
ESC	escape	%1B
FS	file separator	%1C

GS	group separator	%1D
RS	record separator	%1E
US	unit separator	%1F

□ Prev Next □

SEO. Поисковая оптимизация сайта. Краткие рекомендации

Поисковая оптимизация сайта или **SEO** (англ. search engine optimization) - процесс корректировки **HTML**-кода, текстового наполнения (контента), структуры сайта, контроль внешних факторов для соответствия требованиям алгоритма поисковых систем, с целью поднятия позиции сайта в результатах поиска в этих системах по определенным запросам пользователей. Чем выше позиция сайта в результатах поиска, тем больше вероятность, что посетитель перейдет на него с поисковых систем, поскольку люди обычно идут по первым ссылкам, которые находятся в **ТОП-10** поисковых выдач, например, в сервисах Google, Yandex, Bing, Yahoo и др.

□ Home Next □

SEO optimization. Продвижение и раскрутка. ТОП-10 в результатах поиска

Поисковые системы ранжируют сайты в поисковой выдаче по определенному коэффициенту релевантности (то есть, соответствия) ключевому запросу. Существует более 200 факторов ранжирования, используемых **SEO**-оптимизаторами для продвижения сайтов. Используя и максимально учитывая все эти 200 факторов, можно получить высококачественный сайт, который будет отвечать требованиям поисковых систем.

Методы SEO оптимизации

Методы оптимизации можно условно разделить на три класса:

- **Белая оптимизация** - работа над ресурсом без применения официально запрещенных каждой поисковой системой методов раскрутки ресурса - без влияния на поисковые алгоритмы сайтов. Обо всех разрешенных методах поисковой оптимизации можно узнать непосредственно на поисковых сервисах;
- **Серая оптимизация** - сюда можно отнести добавление большого количества ключевых слов в текст страницы, часто с потерей читабельности для человека, неоднократное повторение ключевых запросов определенное количество раз в разных падежах, единственном и множественном числе, а также различных формах глаголов и т.п.;
- **Чёрная оптимизация** - это все методы, которые противоречат правилам поисковых систем. Среди них можно выделить следующие: использование дорвеев (страниц и ресурсов, созданных специально для роботов поисковых систем, зачастую с большим количеством ключевых слов на странице), приём под названием клоакинг (пользователю отдается одна страница, легко читается, а поисковому роботу - другая, оптимизированная под какие-либо запросы), использование скрытого текста на страницах сайта, использование "однопиксельных ссылок" и т.п.

Кратко про SEO оптимизацию сайтов

Вообще, если с самого начала делать сайт по общепринятым рекомендациям и в соответствии с ныне действующей спецификацией **HTML5** , который проходит валидацию без ошибок и предупреждений при проверке валидатором, то сайт уже будет прекрасно находиться различными поисковыми сервисами, в частности, такими как *Google* или *Yandex*. И хотя правильно сверстанный и валидный сайт ещё не означает присутствие в **ТОП-10** поисковых выдач по соответствующим запросам и согласно определенной на сайте тематике, всё равно это уже гарантирует его возможный успех в индексации и появлению в поисковой выдаче, пусть даже и не в первой десятке.

А для того, чтобы сайт появился в **ТОП-10** результатов поиска, конечно, необходимо уделить должное внимание его **SEO оптимизации** с принятием всех необходимых мер для его продвижения и раскрутки. Особенно это касается коммерческих сайтов, владельцы которых имеют целью зарабатывание с помощью сайта денег и получение прибыли за счёт интернет-пользователей.

Некоторые **СЕО-оптимизаторы** используют методы поисковой оптимизации сайтов, которые не всегда являются "честными", когда благодаря искусственно добавленным ключевым словам или фразам на страницах такой сайт попадает в **ТОП-10** поисковой выдачи, но сам сайт не содержит полезной для пользователей информации. Сейчас такие "чёрные" методы поисковой оптимизации прекрасно распознаются поисковыми сервисами, которые даже блокируют такие сайты, выбрасывая их из своих результатов поиска вообще.

Но любой веб-мастер, веб-разработчик или веб-дизайнер, который работает непосредственно с **HTML** кодом, может провести **SEO оптимизацию сайта** для поисковых сервисов самостоятельно, используя очень простые, но достаточно эффективные методы для попадания в **ТОП-10** поисковой выдачи. Для этого необходимо лишь соблюдать определенные правила и рекомендации, которые сообщают сами поисковые сервисы, такие как *Google*, *Yandex*, *Yahoo*, *Bing* и другие. Ниже приводятся лишь некоторые, но наиболее важные рекомендации и требования по **SEO оптимизации** сайтов и, в результате, возможного попадания их в первую десятку поисковой выдачи.

Некоторые рекомендации по SEO оптимизации:

- Используйте оригинальный текст на сайте, которого нет больше нигде в Интернете;
- Длина тега заголовка `<title>` должна быть от 10 до 70 символов (вместе с пробелами);
- Длина метатега описания *description* должна быть в пределах 70 - 320 символов (вместе с пробелами);
- Используйте обязательный атрибут `alt` для изображений;
- Используйте атрибут `title` для изображений;
- Используйте тольки один тег `<h1>` на странице;
- Используйте длину текста между тегами `<h1>-</h1>` не более 70 символов (с пробелами);
- Используйте ограниченное количество заголовков с тегами `<h2>-<h4>` на странице (рекомендуется в пределах 2-5);
- Используйте соотношение код/текст не более 20% (рекомендуется не более 10%);
- Используйте ключевые (тематические) слова равномерно по содержанию страницы;
- Используйте выделение ключевых слов в содержании тегами `<strong>` и `<em>` умеренно и равномерно на странице;
- Используйте количество выделенных ключевых слов не более 3-4% от общего количества текста;
- Используйте на странице с основным контентом не меньше 500 символов;
- В тексте должны встречаться ключевые слова, которые есть в `<h1>`, `<title>`, *description*;
- Установите правильный язык вашей страницы (сайта), на котором больше всего создано контента в теге `<html lang="ru">` (например, русский);
- Обязательно указывайте кодировку страницы **UTF-8** (это оптимальный вариант);
- URL-адрес сделайте более понятным людям и поисковым системам (согласно содержанию сайта или страницы);
- Используйте карту сайта **XML Sitemaps**;
- Используйте файл **robots.txt**;
- Используйте **Favicon** (иконку) для сайта;
- Используйте микроразметку - словари **schema.org** - которые читаются машинами;
- Используйте теги разметки **Open Graph Protocol** (для соц. сетей);
- Разместите в корне сайта страницу **404 error page**;
- Используйте на странице не более 3-х внешних ссылок;
- Используйте атрибут со значением `rel="nofollow"` для внешних ссылок, которые не должны индексироваться;
- Используйте перелинковку внутри сайта (ссылки с одной страницы на другую, но не более одной-двух ссылок на странице);
- Используйте на каждой второстепенной странице одну ссылку на главную страницу сайта;
- Используйте сжимание кода (компрессия HTML/CSS/JS файлов);
- Оптимизируйте изображения (уменьшите размер насколько это возможно);
- Используйте **Mobile Viewports** (оптимизация для мобильных устройств);
- Используйте инструмент отслеживания аналитики (**Google Analytics**, **Yandex-метрика**);
- Сделайте оптимальным размер активных элементов (размер ссылок для малых экранов);
- Вес любой веб-страницы со всеми элементами не должен превышать 100Кбайт (лучше до 50Кбайт);
- С одной страницы должно быть не более 50 ссылок на другие страницы сайта;
- При клике на логотип сайта должен быть переход на главную страницу сайта с любой другой страницы;
- При перелинковке анкором ссылки необходимо делать с помощью слова или словосочетания, которое будет соответствовать тематике.

Дополнительная SEO оптимизация

- Зарегистрируйте сайт в поисковых сервисах (Google, Yandex, Bing, Meta и др.) и в разных каталогах сайтов;
- Максимально увеличте количество активных ссылок на ваш сайт (или на отдельные его страницы) из других сайтов;
- Поощряйте пользователей посетить ваш сайт (используя ссылку на сайт в соцсетях);
- Используйте основной запрос (или ключевые слова) в первом и последнем абзаце текста. Помните, абзацы должны быть не более 4 строк;
- Используйте маркеры и списки, выделяя важные части нумерацией или жирным шрифтом (тег `<strong>`);
- Добавьте на сайт кнопки лайков, репоста в любую из соц.сетей, что будет привлекать внимание новых пользователей.

ЧТО НЕЛЬЗЯ ДЕЛАТЬ ПРИ SEO ОПТИМИЗАЦИИ:

- Не злоупотребляйте количеством ключевых слов на странице;
- Не злоупотребляйте количеством слов, выделенных тегами `<strong>` или `<em>`;

- Количество **HTML** кода не должно превышать 20% от общего веса страницы;
- Не ставьте на странице циклические ссылки (которые ссылаются на эту же страницу);
- Не используйте подчеркивание в url-адресах (адрес_сайта_с_подчёркиванием) без крайней необходимости;
- Не допускайте неработающие (пустые, битые) ссылки на странице;
- Не устанавливайте на странице в качестве обычного текста email адрес;
- Не используйте или старайтесь использовать минимально встроенные стили (CSS inline);
- Не используйте или старайтесь использовать минимально скрипты JavaScript на странице;
- Не используйте Flash без крайней необходимости;
- Не используйте iFrames без крайней необходимости;
- Не скрывайте текст из ключевых слов с помощью очень мелкого шрифта или цвета фона страницы. Поисковые сервисы это быстро распознают и обходят такие сайты.

□ Home Next □

Какие бывают сайты? Разновидности сайтов

> **Сайт**, или **веб-сайт** (от англ. *website*, место, площадка в Интернете) — совокупность веб-страниц, доступных в сети Интернет, которые объединены как по содержанию, так и навигацией под единым доменным именем. Физически сайт может размещаться как на одном, так и на нескольких серверах.
>
> **Сайтом** также называют узел сети Интернет, компьютер, за которым закреплен уникальный IP-адрес, и вообще - какой-то объект в Интернете, за которым закреплен адрес, идентифицирующий его в сети (FTP-site, WWW-site и т.д.).
>
> (*из Википедии*)

□ Home Next □

Какие бывают сайты? Виды сайтов

В данной статье вы узнаете, какие бывают сайты.

Какие бывают виды сайтов?

Сейчас, с развитием новых технологий, видов сайтов достаточно много. Как правило, вид сайта практически всегда определяется его целями и задачами. Например, вы хотите сделать сайт о себе любимом. Такой сайт будет иметь вид **Персонального (личного) сайта** . Или же - сайт-визитка, который может быть создан не только об одном человеке, но и о небольшой компании, организации или предприятии. Такие сайты, как правило, состоят всего лишь из одной или нескольких страниц (обычно до 10 страниц), которые могут быть написаны только с использованием **HTML/CSS**. Это самые простые сайты, состоящие из статических (не меняющихся) страниц. Сайты такого вида как правило включают в себя общую информацию о владельце сайта и его контактные данные. Такие сайты обычно самые дешёвые для заказчика и создаются достаточно быстро.

Корпоративные сайты - это более полнофункциональные представительства компаний в Интернете. Этот тип сайта подходит уже для средних и крупных компаний. Корпоративные сайты содержат полную информацию о компании и её деятельности, и соответственно, предназначены для продвижения какого-то бизнеса и зарабатывания денег.

Интернет-витрины , **интернет-магазины** , **промо-сайты** - это виды сайтов, основная задача которых - продавать (товары, услуги, информацию), популяризировать некую торговую марку, товар, услугу или даже человека (например, "звезду" кино или шоу-бизнеса).

Тематические сайты обычно содержат некоторую информацию любой конкретной тематики. Сюда же можно отнести и интернет-энциклопедии (например, *Википедию*).

Интернет-порталы - это тип сайтов, содержащих большое количество разнообразной информации. Как правило, порталы похожи по структуре с тематическими сайтами, но имеют более развитый функционал и большее количество сервисов и разделов. Также на порталах часто бывают разделы для общения пользователей: чаты, блоги и форумы.

Блоги - это тип сайтов, на которых владелец (обычно он же - администратор сайта) или редактор блога пишет посты со своими новостями, идеями, мыслями или другой информацией, которая постоянно обновляется. Отличительной особенностью блогов является актуальность публикуемой информации, так как устаревшая информация становится неактуальной и неинтересной посетителям. Блоги обычно делаются на какой-то **CMS** (от английского *Content Management System*) - системе управления содержимым (контентом). Это компьютерная программа или информационная система, используемая для организации и обеспечения процесса относительно общего создания, управления и редактирования содержимого сайта. Такие системы еще называют "движками" сайтов. Самая популярная **CMS** в мире - это *Wordpress*. Есть и другие известные движки сайтов, такие как *Joomla*, *Drupal*, *DataLife Engine*, *MODX* и

многие другие.

Каталоги сайтов - это вид сайтов, основным содержанием которых являются структурированные ссылки на другие сайты, а также их краткие описания.

Поисковые системы - вид сайтов, придназначенных для поиска страниц (сайтов) в Интернете по определённым запросам. Самые популярные поисковые системы: *Google.com*, *Yahoo.com*, *Bing.com*, *AOL* (американские), *Yandex*, *Rambler* (российские), *Baidu.com*, *360 Seach* (китайские), *Meta*, *Ukr.net* (украинские) и др.

Почтовые сервисы - этот тип сайтов предоставляет интерфейс для работы с электронной почтой.

Интернет-форумы - на сайтах этого вида пользователи могут создавать темы, а также комментировать их. Как правило, форумы ограничены одной специфической тематикой, хотя встречаются и форумы «обо всем».

Сайты-хостинги - на сайтах этого типа реализована функция хранения любых файлов. Также часто встречаются сайты-хостинги с возможностью просмотра загруженных файлов прямо через браузер. В зависимости от типа файлов отображаются соответствующие значки. Если это видео-файлы, то иногда можно просматривать и само видео. Самый большой и самый популярный сайт-хостинг видео-файлов - *YouTube.com*. В Рунете также популярными являются такие видеохостинги, как *Rutube.ru*, *Myvi.ru*, *Smotri.com* и др. Также к сайтам-хостингам относятся сайты для создания и хранения самих сайтов. Некоторые из них также являются и конструкторами сайтов. Например, один из популярных сайтов в Рунете - бесплатный хостинг и конструктор *Ucoz.com*. Некоторые как платные, так и бесплатные хостинг-сервисы также предоставляют готовые движки сайтов (**CMS**) для сайтов-блогов, например, популярные *Blogger.com*, *LiveJournal.com* (Живой Журнал), *Wordpress.com* и другие.

Доска объявлений - на таких сайтах пользователи могут размещать или искать информацию в виде каких-либо объявлений, например - о купле-продаже товаров или услуг. Также к этому типу относятся сайты о поиске и предложения работы или сайты знакомств.

Социальные сети - тип сайтов, созданных для общения пользователей между собой. Это интерактивные сайты, на которых можно просматривать страницы других пользователей, общаться, комментировать, создавать группы по интересам и множество других сервисов. Также некоторые соц.сети ещё являются и большими файлообменными сервисами, на которых можно загружать, хранить и обмениваться различными типами файлов. Крупнейшие соцсети: *Facebook.com*, *Twitter.com*, *Instagram.com*, *VK.com*, *Одноклассники.ru*, *Qzone.com* (самая большая китайская соц.сеть) и др.

Некоторые сайты являются также и приложениями (обычно для мобильных устройств). Такие сайты уже создаются с помощью языков программирования (например, на языке *PHP*, *Python* (Питон), *Ruby* и др.).

□ Home Next □