**Frontend questions**

**HTML**

[**Хороший сайт с вопросами и ответами**](https://www.frontendinterviewhandbook.com/ru/html-questions/)

[W3SCHOOL](https://www.w3schools.com/html/default.asp)

HTML (Hypertext Markup Language) - это код, который используется для структурирования и отображения веб-страницы и её контента. Например, контент может быть структурирован внутри множества параграфов, маркированных списков или с использованием изображений и таблиц данных. Как видно из названия, эта статья даст вам базовое понимание HTML и его функций.

Что такое HTML на самом деле?

HTML не является языком программирования; это язык разметки, и используется, чтобы сообщать вашему браузеру, как отображать веб-страницы, которые вы посещаете. Он может быть сложным или простым, в зависимости от того, как хочет веб-дизайнер. HTML состоит из ряда элементов, которые вы используете, чтобы вкладывать или оборачивать различные части контента, чтобы заставить контент отображаться или действовать определённым образом. Ограждающие теги могут сделать слово или изображение ссылкой на что-то ещё, могут сделать слова курсивом, сделать шрифт больше или меньше и так далее.

### [Анатомия HTML элемента](https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics" \l "анатомия_html_элемента)

Давайте рассмотрим элемент абзаца более подробно.



Главными частями нашего элемента являются:

1. Открывающий тег (Opening tag): Состоит из имени элемента (в данном случае, "p"), заключённого в открывающие и закрывающие угловые скобки. Открывающий тег указывает, где элемент начинается или начинает действовать, в данном случае — где начинается абзац.
2. Закрывающий тег (Closing tag): Это то же самое, что и открывающий тег, за исключением того, что он включает в себя косую черту перед именем элемента. Закрывающий элемент указывает, где элемент заканчивается, в данном случае — где заканчивается абзац. Отсутствие закрывающего тега является одной из наиболее распространённых ошибок начинающих и может приводить к странным результатам.
3. Контент (Content): Это контент элемента, который в данном случае является просто текстом.
4. Элемент(Element): Открывающий тег, закрывающий тег и контент вместе составляют элемент.

Элементы также могут иметь атрибуты, которые выглядят так:



Атрибуты содержат дополнительную информацию об элементе, которую вы не хотите показывать в фактическом контенте. В данном случае, class это имя атрибута, а editor-note это значение атрибута. Класс позволяет дать элементу идентификационное имя, которое может позже использоваться, чтобы обращаться к элементу с информацией о стиле и прочих вещах.

Атрибут всегда должен иметь:

1. Пробел между ним и именем элемента (или предыдущим атрибутом, если элемент уже имеет один или несколько атрибутов).
2. Имя атрибута, за которым следует знак равенства.
3. Значение атрибута, заключённое с двух сторон в кавычки.

Пустые элементы

Некоторые элементы не имеют контента, и называются пустыми элементами. Возьмём элемент <img>, который уже имеется в нашем HTML:

*<img src="images/firefox-icon.png" alt="Моё тестовое изображение">*

Он содержит два атрибута, но не имеет закрывающего тега </img>, и никакого внутреннего контента. Это потому, что элемент изображения не оборачивает контент для влияния на него. Его целью является вставка изображения в HTML страницу в нужном месте.

### [Анатомия HTML документа](https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics" \l "анатомия_html_документа)

Мы завершили изучение основ отдельных HTML элементов, но они не очень полезны сами по себе. Теперь мы посмотрим, как отдельные элементы объединяются в целую HTML страницу. Давайте вернёмся к коду, который мы записывали в наш index.html (с которым мы впервые встретились в статье [Работа с файлами](https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/Dealing_with_files)):

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Моя тестовая страница</title>

</head>

<body>

<img src="images/firefox-icon.png" alt="Моё тестовое изображение">

</body>

</html>

Здесь мы имеем:

Copy to Clipboard

Copy to Clipboard

Copy to Clipboard

* <!DOCTYPE html> — доктайп. В прошлом, когда HTML был молод (около 1991/1992), доктайпы должны были выступать в качестве ссылки на набор правил, которым HTML страница должна была следовать, чтобы считаться хорошим HTML, что могло означать автоматическую проверку ошибок и другие полезные вещи. Однако в наши дни, никто не заботится об этом, и они на самом деле просто исторический артефакт, который должен быть включён для того, что бы все работало правильно. На данный момент это все, что вам нужно знать.
* <html></html> — элемент [<html>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/html). Этот элемент оборачивает весь контент на всей странице, и иногда известен как корневой элемент.
* <head></head> — элемент [<head>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/head). Этот элемент выступает в качестве контейнера для всего, что вы пожелаете включить на HTML страницу, но не являющегося контентом, который вы показываете пользователям вашей страницы. К ним относятся такие вещи, как ключевые слова и описание страницы, которые будут появляться в результатах поиска, CSS стили нашего контента, кодировка и многое другое.
* <body></body> — элемент [<body>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/body). В нем содержится весь контент, который вы хотите показывать пользователям, когда они посещают вашу страницу, будь то текст, изображения, видео, игры, проигрываемые аудиодорожки или что-то ещё.
* <meta charset="utf-8"> — этот элемент устанавливает UTF-8 кодировку вашего документа, которая включает в себя большинство символов из всех известных человечеству языков. По сути, теперь документ может обрабатывать любой текстовый контент, который вы в него вложите. Нет причин не устанавливать её, так как это может помочь избежать некоторых проблем в дальнейшем.
* <title></title> — элемент [<title>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/title). Этот элемент устанавливает заголовок для вашей страницы, который является названием, появляющимся на вкладке браузера загружаемой страницы, и используется для описания страницы, когда вы добавляете её в закладки/избранное.

## [Разметка текста](https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics" \l "разметка_текста)

В этом разделе рассмотрим некоторые из основных HTML элементов, которые вы будете использовать для разметки текста.

### [Заголовки](https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics" \l "заголовки)

Элементы заголовка позволяют вам указывать определённые части вашего контента в качестве заголовков или подзаголовков. Точно так же, как книга имеет название, названия глав и подзаголовков, HTML документ может содержать то же самое. HTML включает шесть уровней заголовков [<h1> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements)–[<h6> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), хотя обычно вы будете использовать не более 3-4 :

<h1>Мой главный заголовок</h1>

<h2>Мой заголовок верхнего уровня</h2>

<h3>Мой подзаголовок</h3>

<h4>Мой под-подзаголовок</h4>

Теперь попробуйте добавить подходящее название для вашей HTML страницы, чуть выше элемента [<img>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/img).

Copy to Clipboard

### [Абзацы](https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics" \l "абзацы)

Как было сказано раньше, элемент [<p>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/p) предназначен для абзацев текста; вы будете использовать их регулярно при разметке текстового контента:

<p>Это одиночный абзац</p>

Добавьте свой образец текста (вы создавали его в статье [*Каким должен быть ваш веб-сайт?*](https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/What_should_your_web_site_be_like)) в один или несколько абзацев, расположенных прямо под элементом [<img>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/img).

Copy to Clipboard

### [Списки](https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics" \l "списки)

Большая часть веб-контента является списками и HTML имеет специальные элементы для них. Разметка списка всегда состоит по меньшей мере из двух элементов. Наиболее распространёнными типами списков являются нумерованные и ненумерованные списки:

1. Ненумерованные списки - это списки, где порядок пунктов не имеет значения, как в списке покупок. Они оборачиваются в элемент [<ul>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/ul).
2. Нумерованные списки - это списки, где порядок пунктов имеет значение, как в рецепте. Они оборачиваются в элемент [<ol>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/ol).

Каждый пункт внутри списков располагается внутри элемента [<li>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/li) (list item, элемент списка).

Например, если мы хотим включить часть следующего фрагмента абзаца в список:

<p>Mozilla, мы являемся мировым сообществом технологов, мыслителей и строителей, работающих вместе ... </p>

Мы могли бы изменить разметку на эту:

Copy to Clipboard

<p>Mozilla, мы являемся мировым сообществом</p>

<ul>

<li>технологов</li>

<li>мыслителей</li>

<li>строителей</li>

</ul>

<p>работающих вместе ... </p>

Попробуйте добавить упорядоченный или неупорядоченный список на свою страницу.

Copy to Clipboard

## [Ссылки](https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics" \l "ссылки)

Ссылки очень важны — это то, что делает Интернет Интернетом. Чтобы добавить ссылку, нам нужно использовать простой элемент — [<a>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/a) — a это сокращение от "anchor" ("якорь"). Чтобы текст в вашем абзаце стал ссылкой, выполните следующие действия:

1. Выберите некоторый текст. Мы выбрали текст "Манифест Mozilla".
2. Оберните текст в элемент [<a>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/a), например так:

<a>Манифест Mozilla</a>

1. Задайте элементу [<a>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/a) атрибут href, например так:

Copy to Clipboard

<a href="">Манифест Mozilla</a>

1. Заполните значение этого атрибута веб-адресом, на который вы хотите указать ссылку:

Copy to Clipboard

<a href="https://www.mozilla.org/ru/about/manifesto/details/">Манифест Mozilla</a>

Copy to Clipboard

Вы можете получить неожиданные результаты, если в самом начале веб-адреса вы опустите https:// или http:// часть, называемую протоколом. После создания ссылки, кликните по ней, чтобы убедиться, что она направляет вас туда, куда вы хотели.

Примечание: href сначала может выглядеть довольно непонятым выбором для имени атрибута. Если у вас возникли проблемы с тем, чтобы запомнить его, можете запомнить, что атрибут href образуется как hypertext reference ("гипертекстовая ссылка").

Теперь добавьте ссылку на вашу страницу, если вы ещё не сделали этого.

# **Глобальные атрибуты**

Глобальные атрибуты — это атрибуты общие для всех HTML-элементов; они могут использоваться всеми элементами, хотя на некоторые элементы могут не оказывать влияния.

Глобальные атрибуты могут быть указаны для любых [элементов HTML](https://developer.mozilla.org/ru/docs/Web/HTML/Element), даже для тех, которые не указаны в стандарте. Это значит, что все нестандартные элементы должны допускать эти атрибуты, даже если использование этих элементов означает, что документ больше не является HTML5-совместимым. Например, HTML5-совместимые браузеры скрывают содержимое, помеченное как <foo hidden>...<foo> , хотя <foo> не является верным HTML-элементом.

В дополнение к основным для HTML глобальным атрибутам, также существуют следующие глобальные атрибуты:

* xml:lang и xml:base — унаследованы от спецификации XHTML, и хотя являются устаревшими, сохранены в целях совместимости.
* Многочисленные [aria-\*](https://developer.mozilla.org/ru/docs/Web/Accessibility/ARIA) атрибуты, используемые для улучшения доступности.
* Атрибуты обработчиков событий: onabort, onautocomplete, onautocompleteerror, onblur, oncancel, oncanplay, oncanplaythrough, onchange, onclick, onclose, oncontextmenu, oncuechange, ondblclick, ondrag, ondragend, ondragenter, ondragexit, ondragleave, ondragover, ondragstart, ondrop, ondurationchange, onemptied, onended, onerror, onfocus, oninput, oninvalid, onkeydown, onkeypress, onkeyup, onload, onloadeddata, onloadedmetadata, onloadstart, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onmousewheel, onpause, onplay, onplaying, onprogress, onratechange, onreset, onresize, onscroll, onseeked, onseeking, onselect, onshow, onsort, onstalled, onsubmit, onsuspend, ontimeupdate, ontoggle, onvolumechange, onwaiting.

**[Description](https://developer.mozilla.org/ru/docs/Web/HTML/Global_attributes" \l "описание)**

# Категории контента

Каждый элемент [HTML](https://developer.mozilla.org/ru/docs/Web/HTML) принадлежит некоторому количеству категорий контента, которые объединяют элементы с общим набором характеристик. Такая группировка является свободной, то есть не задаёт на самом деле отношение между элементами таких типов, но помогает определить и описать их поведение и правила, которым они должны следовать, особенно когда дело доходит до сложных деталей. Также возможна ситуация, когда элемент не входит ни в одну из этих категорий.

Существует три типа категорий контента:

* Основные категории контента, описывающие общие для многих элементов правила;
* Категории контента для элементов форм, описывающие общие правила для элементов форм;
* Особые категории контента, описывающие редкие категории, актуальные только для нескольких элементов, возможно, лишь в особом контексте.

## [Основные категории контента](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "основные_категории_контента)

### [Метаданные](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "метаданные)

Элементы, принадлежащие к категории метаданных, изменяют отображение или поведение HTML-документа, связывают его с другими документами и предоставляют другую дополнительную информацию о документе.

Элементами метаданных являются: [<base>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/base), <command>, [<link>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/link), [<meta>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/meta), [<noscript>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/noscript), [<script>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/script), [<style>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/style) и [<title>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/title).

### [Основной поток](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "основной_поток)

Элементы основного потока обычно содержат текст и встроенный контент.

[<a>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/a), [<abbr>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/abbr), [<address>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/address), [<article>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/article), [<aside>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/aside), [<audio>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/audio), [<b>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/b),[<bdo>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/bdo), [<bdi>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/bdi), [<blockquote>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/blockquote), [<br>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/br), [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button), [<canvas>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/canvas), [<cite>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/cite), [<code>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/code), <command>, [<data>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/data), [<datalist>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/datalist), [<del>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/del), [<details>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/details), [<dfn>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/dfn), [<div>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/div), [<dl>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/dl), [<em>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/em), [<embed>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/embed), [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset), [<figure>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/figure), [<footer>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/footer), [<form>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/form), [<h1> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), [<h2> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), [<h3> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), [<h4> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), [<h5> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), [<h6> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), [<header>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/header), [<hgroup>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/hgroup), [<hr>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/hr), [<i> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/i), [<iframe> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe), [<img>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/img), [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input), [<ins>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/ins), [<kbd>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/kbd), [<keygen> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/keygen), [<label>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/label), [<main>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/main), [<map>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/map), [<mark>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/mark), [<math>](https://developer.mozilla.org/ru/docs/Web/MathML/Element/math), [<menu>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/menu), [<meter>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/meter), [<nav>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/nav), [<noscript>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/noscript), [<object> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/object), [<ol>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/ol), [<output>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/output), [<p>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/p), [<pre>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/pre), [<progress>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/progress), [<q> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/q), [<ruby>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/ruby), [<s> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/s), [<samp> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/samp), [<script>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/script), [<section>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/section), [<select>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/select), [<small> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/small), [<span>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/span), [<strong>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/strong), [<sub> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sub), [<sup> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sup), [<svg>](https://developer.mozilla.org/ru/docs/Web/SVG/Element/svg), [<table> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/table), [<template>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/template), [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea), [<time>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/time), [<ul>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/ul), [<var> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/var), [<video>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/video), [<wbr>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/wbr) и Текст.

Устарело

Несколько других элементов, которые тоже принадлежат к этому типу, но требуют определённых условий:

* [<area>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/area), только внутри элемента [<map>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/map)
* [<link>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/link), при наличии атрибута [itemprop](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes" \l "attr-itemprop) (en-US)
* [<meta>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/meta), при наличии атрибута [itemprop](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes" \l "attr-itemprop) (en-US)
* [<style>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/style), при наличии атрибута [scoped](https://developer.mozilla.org/ru/docs/Web/HTML/Element/style" \l "attr-scoped)

### [Секционный контент](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "секционный_контент)

Подобные элементы создают [секции (блоки) в текущей структуре документа](https://developer.mozilla.org/ru/docs/Sections_and_Outlines_of_an_HTML5_document), определяющие область действия [заголовочного контента](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "Заголовочный_контент) и элементов [<header>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/header) и [<footer>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/footer)

К этой категории принадлежат элементы [<article>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/article), [<aside>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/aside), [<nav>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/nav) и [<section>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/section).

Примечание: Не стоит путать данную модель контента с категорией [корня задания разделов](https://developer.mozilla.org/ru/docs/Web/Guide/HTML/Sections_and_Outlines_of_an_HTML5_document" \l "Корни_задания_разделов), которая изолирует своё содержимое от обычной структуры.

### [Заголовочный контент](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "заголовочный_контент)

Заголовочный контент задаёт заголовок секции, явно отмеченной структурным элементом или неявно – самим заголовочным.

Данной категории принадлежат такие элементы, как [<h1> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), [<h2> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), [<h3> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), [<h4> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), [<h5> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements), [<h6> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements) и [<hgroup>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/hgroup).

Примечание: Элемент [<header>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/header) сам по себе не является заголовочным, хотя с большой вероятностью содержит контент такого типа.

Примечание: Стоит заметить, что элемент [<hgroup>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/hgroup) был удалён из спецификации W3C HTML до приведения HTML 5 к окончательному варианту, но до сих пор является частью спецификации WHATWG и по крайней мере частично поддерживается большинством браузеров.

### [Фразовый контент](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "фразовый_контент)

Фразовый контент определяет текст и его формат. Серии фразового контента образуют параграфы.

К данной категории принадлежат следующие элементы: [<abbr>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/abbr), [<audio>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/audio), [<b>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/b), [<bdo>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/bdo), [<br>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/br), [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button), [<canvas>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/canvas), [<cite>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/cite), [<code>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/code), <command>, [<data>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/data), [<datalist>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/datalist), [<dfn>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/dfn), [<em>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/em), [<embed>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/embed), [<i> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/i), [<iframe> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe), [<img>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/img), [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input), [<kbd>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/kbd), [<keygen> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/keygen), [<label>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/label), [<mark>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/mark), [<math>](https://developer.mozilla.org/ru/docs/Web/MathML/Element/math), [<meter>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/meter), [<noscript>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/noscript), [<object> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/object), [<output>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/output), [<progress>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/progress), [<q> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/q), [<ruby>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/ruby), [<samp> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/samp), [<script>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/script), [<select>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/select), [<small> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/small), [<span>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/span), [<strong>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/strong), [<sub> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sub), [<sup> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sup), [<svg>](https://developer.mozilla.org/ru/docs/Web/SVG/Element/svg), [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea), [<time>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/time), [<var> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/var), [<video>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/video), [<wbr>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/wbr) и обычный текст (не только состоящий из символов пробелов).

Устарело

Ещё несколько элементов входят в данную категорию при соблюдении особых условий:

* [<a>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/a), если содержит в себе только фразовый контент
* [<area>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/area), только внутри элемента [<map>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/map)
* [<del>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/del), если содержит в себе только фразовый контент
* [<ins>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/ins), если содержит в себе только фразовый контент
* [<link>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/link), при наличии атрибута [itemprop](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes" \l "attr-itemprop) (en-US)
* [<map>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/map), если содержит в себе только фразовый контент
* [<meta>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/meta), при наличии атрибута [itemprop](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes" \l "attr-itemprop) (en-US)

### [Встроенный контент](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "встроенный_контент)

Встроенный контент импортирует в документ другой ресурс или вставляет содержимое на другом языке разметки или принадлежащее другому пространству имён. Элементами данной категории являются: [<audio>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/audio), [<canvas>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/canvas), [<embed>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/embed), [<iframe> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe), [<img>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/img), [<math>](https://developer.mozilla.org/ru/docs/Web/MathML/Element/math), [<object> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/object), [<svg>](https://developer.mozilla.org/ru/docs/Web/SVG/Element/svg), [<video>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/video).

### [Интерактивный контент](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "интерактивный_контент)

К интерактивному контенту относятся элементы, который специально разработаны для взаимодействия с пользователем. В данную категорию входят [<a>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/a), [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button), [<details>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/details), [<embed>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/embed), [<iframe> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe), [<keygen> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/keygen), [<label>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/label), [<select>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/select) и [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea). Некоторые элементы считаются интерактивным контентом только при соблюдении определённых условий:

Устарело

* [<audio>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/audio), если указан атрибут [controls](https://developer.mozilla.org/ru/docs/Web/HTML/Element/audio" \l "attr-controls)
* [<img>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/img), если указан атрибут [usemap](https://developer.mozilla.org/ru/docs/Web/HTML/Element/img" \l "attr-usemap)
* [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input), если атрибут [type](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input" \l "attr-type) не скрыт
* [<menu>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/menu), если атрибут [type](https://developer.mozilla.org/ru/docs/Web/HTML/Element/menu" \l "attr-type) имеет значение toolbar
* [<object> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/object), если указан атрибут [usemap (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/object)
* [<video>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/video), если указан атрибут [controls](https://developer.mozilla.org/ru/docs/Web/HTML/Element/video" \l "attr-controls)

### [Явный контент](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "явный_контент)

Контент считается явным, когда он не скрыт и не является пустым, это контент, который отображён и предметен. У элементов потокового или фразового контента должен быть хотя бы один явный узел.

### [Контент форм](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "контент_форм)

Контент форм включает в себя элементы, у которых есть владелец формы, обозначенный атрибутом form. Владельцем формы является либо элемент [<form>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/form), либо элемент, id которого указан в атрибуте form.

* [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button)
* [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset)
* [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input)
* [<keygen> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/keygen)

Устарело

* [<label>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/label)
* [<meter>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/meter)
* [<object> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/object)
* [<output>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/output)
* [<progress>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/progress)
* [<select>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/select)
* [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea)

Данная категория включает несколько подкатегорий:

listed

Элементы, перечисленные в IDL коллекциях [form.elements (en-US)](https://developer.mozilla.org/en-US/docs/Web/API/HTMLFormElement/elements) и fieldset.elements. Включают [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button), [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset), [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input), [<keygen> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/keygen), [<object> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/object), [<output>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/output), [<select>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/select), и [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea).

Устарело

labelable

Элементы, которые могут ассоциироваться с элементами [<label>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/label). Включают [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button), [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input), [<keygen> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/keygen), [<meter>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/meter), [<output>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/output), [<progress>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/progress), [<select>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/select) и [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea).

Устарело

submittable

Элементы, которые могут использоваться для построения набора данных формы при отправке на сервер. Включают [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button), [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input), [<keygen> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/keygen), [<object> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/object), [<select>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/select) и [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea).

Устарело

resettable

Элементы, которые могут быть затронуты при сбросе данных формы. Включают [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input), [<keygen> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/keygen), [<output>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/output),[<select>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/select) и [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea).

Устарело

## [Вторичные категории](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "вторичные_категории)

Существуют некоторые вторичные классификации элементов, о которых тоже полезно знать.

### [Элементы поддержки скриптов](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "элементы_поддержки_скриптов)

Элементы поддержки скриптов – это элементы, которые напрямую не влияют на отрисовку документа. Вместо этого они служат для внедрения скриптов, путём либо содержания кода скрипта напрямую, либо указания данных, которые будут использованы скриптами.

Элементами поддержки скриптов являются:

* [<script>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/script)
* [<template>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/template)

## [Прозрачная модель контента](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "прозрачная_модель_контента)

Если элемент обладает прозрачной моделью контента, его содержимое должно быть структурировано таким образом, чтобы оно оставалось валидным HTML5, даже при удалении прозрачного элемента или заменой его на дочерние элементы.

Например, элементы [<del>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/del) и [<ins>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/ins) являются прозрачными:

<p>Мы почитаем следующие истины <del><em>священными &amp; неоспоримыми</em></del> <ins>само собой разумеющимися</ins>.</p>

Если удалить эти элементы, данный фрагмент всё равно будет представлять собой валидный HTML.

<p>Мы почитаем следующие истины <em>священными &amp; неоспоримыми</em> само собой разумеющимися.</p>

## [Прочие модели контента](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "прочие_модели_контента)

Корень задания разделов.

[More Info](http://learning.dman54.tmweb.ru/index_2.html)

**Что такое doctype? И для чего он используется?**

Элемент **<!DOCTYPE>** предназначен для указания типа текущего документа — DTD (document type definition, описание типа документа). Это необходимо, чтобы браузер понимал, как следует интерпретировать текущую веб-страницу, поскольку HTML существует в нескольких версиях, кроме того, имеется XHTML (EXtensible HyperText Markup Language, расширенный язык разметки гипертекста), похожий на HTML, но различающийся с ним по синтаксису. Чтобы браузер «не путался» и понимал, согласно какому стандарту отображать веб-страницу и необходимо в первой строке кода задавать **<!DOCTYPE>**. Также браузер понимает, какие теги являются валидными, а какие нет.

Существует несколько видов **<!DOCTYPE>**, они различаются в зависимости от версии языка, на которого ориентированы. В табл. 1. приведены основные типы документов с их описанием.

Табл. 1. Допустимые DTD

|  |  |
| --- | --- |
| DOCTYPE | Описание |
| HTML 4.01 | |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"> | Строгий синтаксис HTML. |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"> | Переходный синтаксис HTML. |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd"> | В HTML-документе применяются фреймы. |
| HTML 5 | |
| <!DOCTYPE html> | Для всех документов. |
| XHTML 1.0 | |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> | Строгий синтаксис XHTML. |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> | Переходный синтаксис XHTML. |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd"> | Документ написан на XHTML и содержит фреймы. |
| XHTML 1.1 | |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> | Разработчики XHTML 1.1 предполагают, что он постепенно вытеснит HTML. Никакого деления на виды это определение не имеет, синтаксис один и подчиняется четким правилам. |

### Синтаксис

<!DOCTYPE [Элемент верхнего уровня] [Публичность] "[Регистрация]//[Организация]//[Тип] [Имя]//[Язык]" "[URL]">

### Параметры

Элемент верхнего уровня — указывает элемент верхнего уровня в документе, для HTML это тег **<html>**.

Публичность — объект является публичным (значение PUBLIC) или системным ресурсом (значение SYSTEM), например, таким как локальный файл. Для HTML/XHTML указывается значение PUBLIC.

Регистрация — сообщает, что разработчик DTD зарегистрирован в международной организации по стандартизации (International Organization for Standardization, ISO). Принимает одно из двух значений: плюс (+) — разработчик зарегистрирован в ISO и - (минус) — разработчик не зарегистрирован. Для W3C значение ставится «-».

Организация — уникальное название организации, разработавшей DTD. Официально HTML/XHTML публикует W3C, это название и пишется в **<!DOCTYPE>**.

Тип — тип описываемого документа. Для HTML/XHTML значение указывается DTD.

Имя — уникальное имя документа для описания DTD.

Язык — язык, на котором написан текст для описания объекта. Содержит две буквы, пишется в верхнем регистре. Для документа HTML/XHTML указывается английский язык (EN).

URL — адрес документа с DTD.

### Закрывающий тег

Не требуется.

Пример 1. HTML 4.01

HTML 4.01IECrOpSaFx

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html4/strict.dtd">

**<html>**

**<head>**

**<title>**!DOCTYPE**</title>**

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

**</head>**

**<body>**

**<p>**Разум — это Будда, а прекращение умозрительного мышления — это путь.

Перестав мыслить понятиями и размышлять о путях существования и небытия,

о душе и плоти, о пассивном и активном и о других подобных вещах,

начинаешь осознавать, что разум — это Будда,

что Будда — это сущность разума,

и что разум подобен бесконечности.**</p>**

**</body>**

**</html>**

Пример 2. HTML 5

HTML5IECrOpSaFx

<!DOCTYPE html>

**<html>**

**<head>**

**<title>**!DOCTYPE**</title>**

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

**</head>**

**<body>**

**<p>**Разум — это Будда, а прекращение умозрительного мышления — это путь.

Перестав мыслить понятиями и размышлять о путях существования и небытия,

о душе и плоти, о пассивном и активном и о других подобных вещах,

начинаешь осознавать, что разум — это Будда,

что Будда — это сущность разума,

и что разум подобен бесконечности.**</p>**

**</body>**

**</html>**

### Браузеры

Internet Explorer до версии 6.0 требует, чтобы **<!DOCTYPE>** стоял обязательно в первой строке кода. В противном случае браузер переходит в режим совместимости (quirk mode).

Хотя значение URL является не обязательным, браузеры при его отсутствии могут перейти в режим совместимости, поэтому всегда указывайте полный путь к DTD-файлу, как показано в табл. 1.

**Что такое валидация? И какие типы проверок HTML документа вы знаете?**

**Валидация html-документа** – это проверка html-документа на соответствие существующим веб-стандартам.

Проверить html-документ на наличие ошибок можно на сайте <http://validator.w3.org/>. Открываем вкладку Validate by File Upload, нажимаем кнопку Выбрать файл, выбираем html-документ, затем нажимаем кнопку Check:

При написании html кода соблюдайте ряд правил:

1. Обязательно указывайте тип документа <!doctype>.
2. Не используйте атрибуты тегов для оформления элементов веб-страницы, они уже давно устарели, и вместо них нужно использовать каскадные таблицы стилей.
3. Соблюдайте правила вложенности тегов, например, такая запись будет считаться неправильной:

<strong>текст <em>текст</strong> текст</em>

Нужно сначала закрыть вложенный тег <em>, затем его родитель <strong>:

<strong>текст <em>текст</em> текст</strong>

1. Не забывайте об обязательных атрибутах.
2. Строчные элементы не могут содержать в себе блочные. Такая запись будет считаться ошибочной:

<strong>текст <p>текст</p> текст</strong>

Правильной будет такая запись:

<p>текст <strong>текст</strong> текст</p>

1. Не используйте устаревшие теги. Со всеми тегами, входящими в спецификацию пятой версии HTML можно ознакомиться на сайте [htmlbook.ru](http://htmlbook.ru/) или [www.w3schools.com](http://www.w3schools.com/)

Типы проверок: проверка синтаксиса, проверка вложенности тегов, проверка DTD, проверка на наличие посторонних элементов или тегов.

[HTML-валидатор](http://validator.w3.org/) производит несколько проверок Вашего кода. Основные из них:

1. **Валидация синтаксиса** — проверка на наличие синтаксических ошибок. <foo bar="baz"> является корректным синтаксисом, несмотря на то, что <foo> не является допустимым HTML-тэгом, так что проверка синтаксиса является минимально полезной для написания хорошего HTML.
2. **Проверка вложенности тэгов** — тэги должны быть закрыты в обратном порядке относительно их открытия. Например, эта проверка отлавливает ошибки с неправильно закрытыми <div>.
3. **Валидация DTD** — проверка соответствия Вашего кода указанному Document Type Definition. Она включает проверку названий тэгов, атрибутов, и «встраивания» тэгов (тэги одного типа внутри тэгов другого типа)
4. **Проверка на посторонние элементы** — проверка выявляет все, что есть в коде, но отсутствует в DTD. Например, пользовательские тэги и атрибуты.

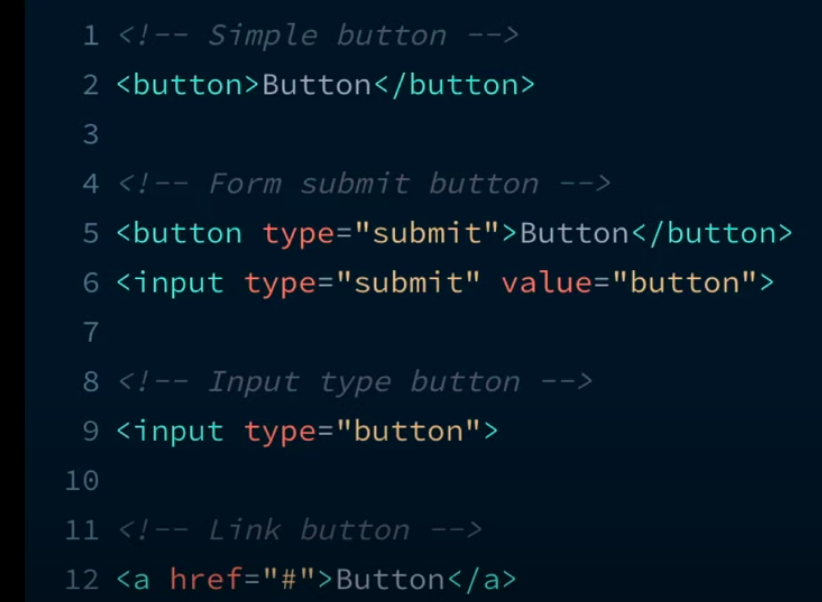
### **Представьте HTML5 как открытую веб-платформу. Из каких блоков состоит HTML5?[​](https://www.frontendinterviewhandbook.com/ru/html-questions" \l "представьте-html5-как-открытую-веб-платформу-из-каких-блоков-состоит-html5)**

* Семантика. Позволяет более точно описать из чего состоит контент.
* Связанность. Позволяет общаться с сервером новыми и инновационными способами.
* Офлайн и хранилище. Позволяют страницам хранить данные локально на клиентской стороне и более эффективно работать в офлайне.
* Мультимедиа. Ставит создание видео и аудио на первое место в вебе.
* 2D/3D-графика и эффекты. Позволяет расширить возможности презентации.
* Производительность и интеграция. Обеспечивает большую скорость оптимизации и лучшее использование аппаратных средств.
* Доступ к устройствам. Позволяет взаимодействовать с различными устройствами ввода и вывода.
* Стилизация. Позволяет создавать более сложные темы оформления.

###### Ссылки[​](https://www.frontendinterviewhandbook.com/ru/html-questions" \l "ссылки-4)

* <https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5>

[Какой тэг использовать для того, что бы сверстать кнопку?](https://youtu.be/G7hLwudGWL4?t=96)





[Что такое инлайновый стиль? Можно ли его переопределить?](https://youtu.be/G7hLwudGWL4?t=155)

[Есть ли у HTML элементов свои дефолтные специфичные стили?](https://youtu.be/G7hLwudGWL4?t=181)

Да, есть, у всех свои специфические дефолтные стили, зависят от браузера. Можно привести к единому стилю во всех браузерах. Для этого используется либо normalize, либо reset.

[Что такое семантика? Какие семантичные тэги вы знаете?](https://youtu.be/ycYp7CYOnO0?t=149)

<https://www.w3schools.com/html/html5_semantic_elements.asp>

Давным-давно (лет пятнадцать назад) почти все делали сайты и не переживали о том, что под капотом. Верстали таблицами, использовали всё, что попадётся под руку (а попадались в основном div и span) и не особо заморачивались о доступности. А потом случился HTML5 и понеслось.

**Семантическая вёрстка** — подход к разметке, который опирается не на содержание сайта, а на смысловое предназначение каждого блока и логическую структуру документа. Даже в этой статье есть заголовки разных уровней — это помогает читателю выстроить в голове структуру документа. Так и на странице сайта — только читатели будут немного другими.

**Почему семантика важна**

Чтобы сделать сайт доступным. Зрячие пользователи могут без проблем с первого взгляда понять, где какая часть страницы находится — где заголовок, списки или изображения. Для незрячих или частично незрячих всё сложнее. Основной инструмент для просмотра сайтов не браузер, который отрисовывает страницу, а скринридер, который читает текст со страницы вслух.

Этот инструмент «зачитывает» содержимое страницы, и семантическая структура помогает ему лучше определять, какой сейчас блок, а пользователю понимать, о чём идёт речь. Таким образом семантическая разметка помогает большему количеству пользователей работать с вашим сайтом. Например, наличие заголовков помогает незрячим в навигации по странице. У скринридеров есть функция навигации по заголовкам, что ускоряет знакомство с информацией на сайте.

Чтобы сайт был выше в поисковиках. Поисковики не разглашают правила ранжирования, но известно, что наличие семантической разметки страниц помогает поисковым ботам лучше понимать, что находится на странице, и в зависимости от этого ранжировать сайты в поисковой выдаче.

Разработчики tutu.ru сверстали таблицу тегом table вместо div и их сниппет оказался в выдаче Google по важному коммерческому запросу.

Семантика прописана в стандартах. Многие разработчики по старинке пользуются конструкциями типа <div id="nav"> для обозначения навигации или других структурных элементов страницы. Тем временем в стандарте HTML есть несколько семантических тегов, которые рекомендуется использовать для разметки страниц вместо <div> и span. В спецификации для каждого семантического элемента описана его роль.

Ну и представьте, насколько проще читать <nav></nav> вместо <div class="nav"></div>. Или вот такой код. Смотрите и сразу понятно, что тут и зачем.

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="utf-8">

<title>Заголовок страницы</title>

</head>

<body>

<header class="main-header">

<!— Шапка сайта —>

</header>

<main>

<!— Основное содержимое страницы —>

</main>

<footer class="main-footer">

<!— Подвал сайта —>

</footer>

</body>

</html>

**Основные семантические теги HTML**

Среди «старых» тегов из ранних версий HTML тоже есть семантические — например, тег <p>, который обозначает параграф. При этом теги <i> или <b> не семантические, потому что они не добавляют смысла выделенному тексту, а просто определяют его внешний вид.

Но в актуальной версии стандарта HTML Living Standard есть семантические теги почти для всех основных частей сайта, и лучше пользоваться ими. Вот несколько примеров семантических тегов.

<article>

Значение: независимая, отделяемая смысловая единица, например комментарий, твит, статья, виджет ВК и так далее.

Особенности: желателен заголовок внутри.

Типовые ошибки: путают с тегами <section> и <div>.

<section>

Значение: смысловой раздел документа. Неотделяемый, в отличие от <article>.

Особенности: желателен заголовок внутри.

Типовые ошибки: путают с тегами <article> и <div>.

<aside>

Значение: побочный, косвенный для страницы контент.

Особенности: может иметь свой заголовок. Может встречаться несколько раз на странице.

Типовые ошибки: считать <aside> тегом для «боковой панели» и размечать этим тегом основной контент, который связан с окружающими его элементами.

<nav>

Значение: навигационный раздел со ссылками на другие страницы или другие части страниц.

Особенности: используется для основной навигации, а не для всех групп ссылок. Основной является навигация или нет — на усмотрение верстальщика. Например, меню в подвале сайта можно не оборачивать в <nav>. В подвале обычно появляется краткий список ссылок (например, ссылка на главную, копирайт и условия) — это не является основной навигацией, семантически для такой информации предназначен <footer> сам по себе.

Типовые ошибки: многие считают, что в <nav> может быть только список навигационных ссылок, но согласно спецификации там может быть навигация в любой форме.

<header>

Значение: вводная часть смыслового раздела или всего сайта, обычно содержит подсказки и навигацию. Чаще всего повторяется на всех страницах сайта.

Особенности: этих элементов может быть несколько на странице.

Типовые ошибки: использовать только как шапку сайта.

<main>

Значение: основное, не повторяющееся на других страницах, содержание страницы.

Особенности: должен быть один на странице, исходя из определения.

Типовые ошибки: включать в этот тег то, что повторяется на других страницах (навигацию, копирайты и так далее).

<footer>

Значение: заключительная часть смыслового раздела или всего сайта, обычно содержит информацию об авторах, список литературы, копирайт и так далее. Чаще всего повторяется на всех страницах сайта.

Особенности: этих элементов может быть несколько на странице. Тег <footer> не обязан находиться в конце раздела.

Типовые ошибки: использовать только как подвал сайта.

**Как разметить страницу с точки зрения семантики**

Процесс разметки можно разделить на несколько шагов с разной степенью детализации.

Крупные смысловые блоки на каждой странице сайта. Теги: <header>, <main>, <footer>.

Крупные смысловые разделы в блоках. Теги: <nav>, <section>, <article>, <aside>.

Заголовок всего документа и заголовки смысловых разделов. Теги: <h1>-<h6>.

Мелкие элементы в смысловых разделах. Списки, таблицы, демо-материалы, параграфы и переносы, формы, цитаты, контактная информация и прогресс.

Фразовые элементы. Изображения, ссылки, кнопки, видео, время и мелкие текстовые элементы.

Более подробно методика создания семантической разметки описана в навыке «Создание семантической разметки по макету» и профессиональных курсах HTML Academy.

**Сомневаюсь, какие теги использовать**

Есть простые правила для выбора нужных тегов.

Получилось найти самый подходящий смысловой тег — использовать его.

Для потоковых контейнеров — <div>.

Для мелких фразовых элементов (слово или фраза) — <span>.

**Правило для определения <article>, <section> и <div>:**

Можете дать имя разделу и вынести этот раздел на другой сайт? — <article>

Можете дать имя разделу, но вынести на другой сайт не можете? — <section>

Не можете дать имя? Получается что-то наподобие «новости и фотогалерея» или «правая колонка»? — <div>

Как точно не нужно делать

Не используйте семантические теги для красоты. Для этого есть CSS.

Может показаться, что некоторые теги подходят для того, чтобы сделать страницу покрасивее, подвигать текст или добавить ему интервалов. Но то, что браузер по умолчанию отображает теги как-то, как вам нужно, не значит, что это нужно использовать. Посмотрим на пример.

Тег <blockquote> должен использоваться для выделения в тексте цитат, а не просто случайного выделения текста. Так совпало, что в браузерах этот блок по умолчанию выделен, но это не значит, что нужно его использовать таким образом.

Тег <ul> тоже использован для визуального «сдвига» текста. Это неверно, потому что этот тег должен быть использован только для обозначения списков, а во-вторых, в тег <ul> можно вкладывать только теги <li> и ничего больше.

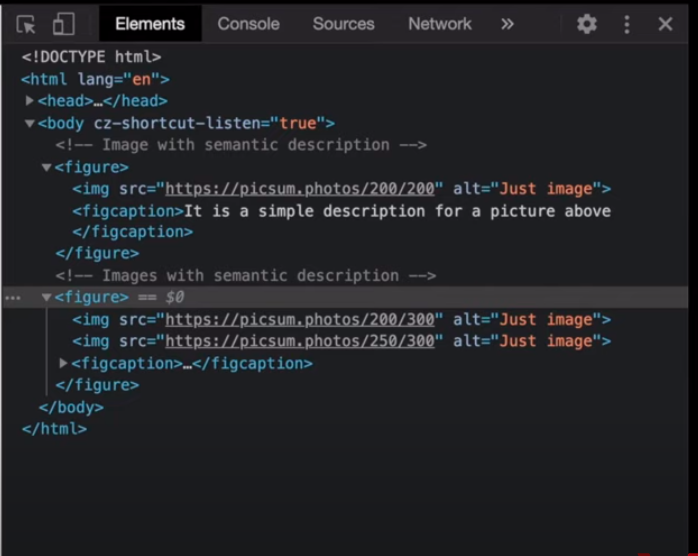
Тег <p> использован, чтобы визуально раздвинуть текст. На самом деле этот тег используется для выделения параграфов.

А любое выделение, сдвиг или иные превращения текста можно выполнить с помощью CSS.

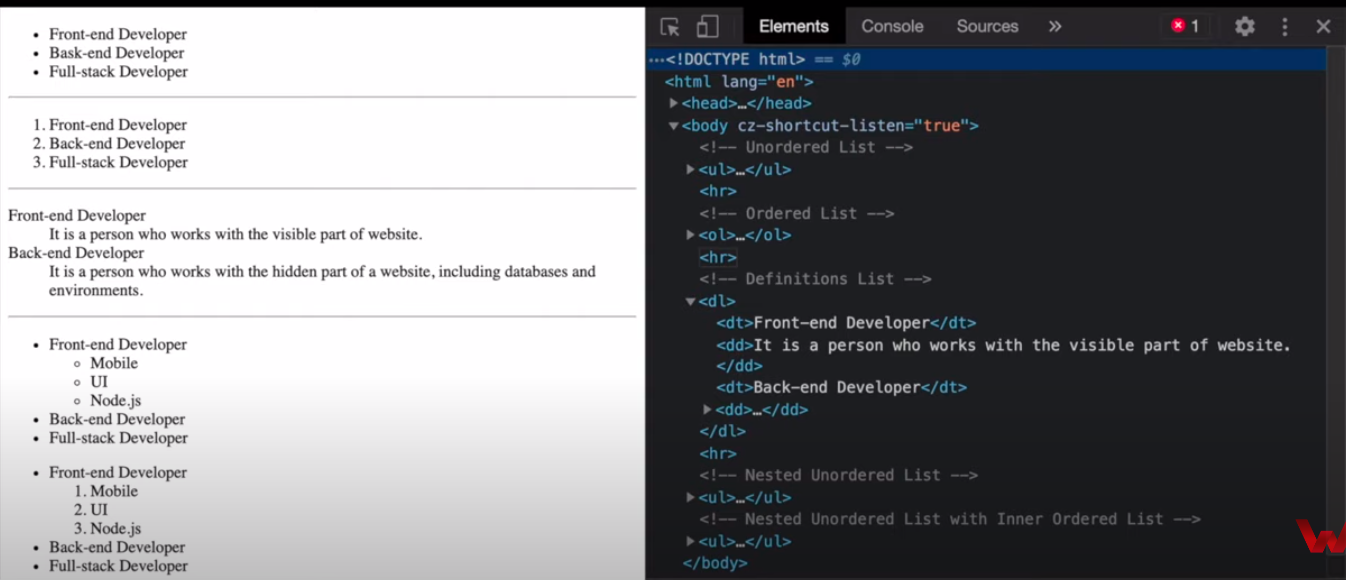
Поэтому используйте семантические теги по назначению.

[Also useful source](https://html5css.ru/html/html5_semantic_elements.php)

[Как семантически правильно сверстать картинку с подписью?](https://youtu.be/1eIRTdgzHtw?t=67)

[](https://youtu.be/1eIRTdgzHtw?t=29)

[Типы списков в HTML?](https://youtu.be/1eIRTdgzHtw?t=29)



[Для какого тэга используется атрибут alt и зачем он нужен?](https://youtu.be/G7hLwudGWL4?t=230)

Aтрибут alt устанавливает альтернативный текст для изображений. Такой текст позволяет получить текстовую информацию о рисунке при отключенной в браузере загрузке изображений. Поскольку загрузка изображений происходит после получения браузером информации о нем, то замещающий рисунок текст появляется раньше. А уже по мере загрузки текст будет сменяться изображением.

### Синтаксис

|  |  |
| --- | --- |
| HTML | **<img**alt="текст"**>** |
| XHTML | **<img**alt="текст" **/>** |

### Значения

Любая подходящяя текстовая строка. Ее обязательно надо брать в двойные или одинарные кавычки.

### Обязательный атрибут

Да.

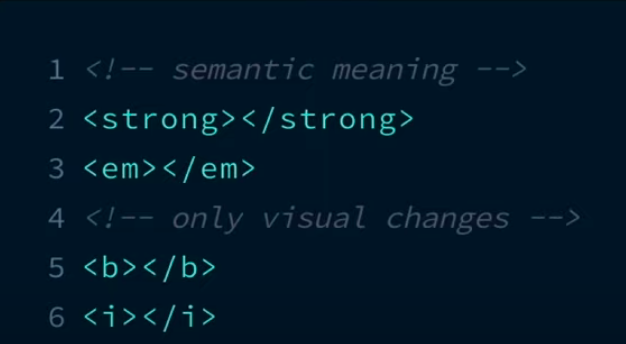
### Значение по умолчанию

Нет.

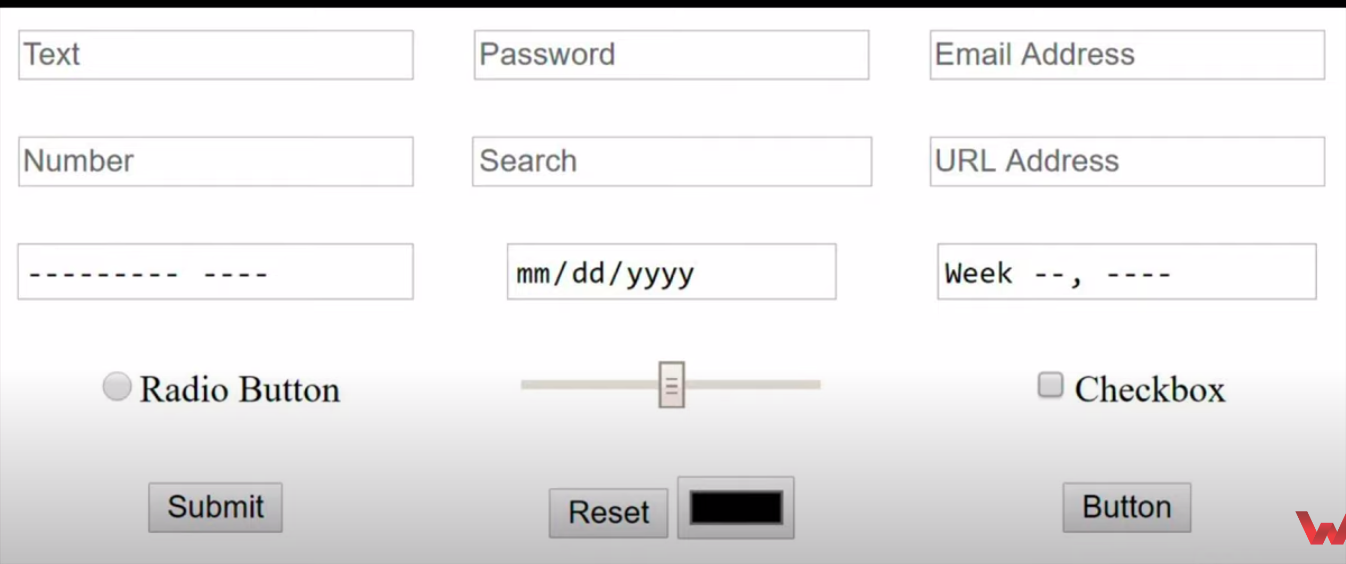
Нужен для доступности или для отображения, если картинка не загрузилась.

[Какая разница между тэгами <strong><em> и <b><i>?](https://youtu.be/ycYp7CYOnO0?t=202)

strong, b – bold text, em, I – italic text in the browser, but:

[](https://youtu.be/CjdCxxqObaM?t=87)

[Типы <input> элементов в HTML?](https://youtu.be/CjdCxxqObaM?t=87)



Элемент HTML <input> используется для создания интерактивных элементов управления в веб-формах для получения данных от пользователя; в зависимости от устройства и [user agent](https://developer.mozilla.org/ru/docs/Glossary/User_agent), доступен широкий выбор типов входных данных и виджетов управления. Из-за огромного количества возможных сочетаний типов ввода и атрибутов это один из самых мощных и сложных элементов HTML.

* [*Content categories (en-US)*](https://developer.mozilla.org/en-US/docs/Web/HTML/Content_categories) [Flow content](https://developer.mozilla.org/en-US/docs/Web/HTML/Content_categories" \l "flow_content)(en-US), listed, submittable, resettable, form-associated element, [phrasing content](https://developer.mozilla.org/en-US/docs/Web/HTML/Content_categories" \l "phrasing_content)(en-US). If the [type](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input" \l "attr-type) has not the hidden value, labellable element, palpable content.
* Permitted content None, it is an [empty element](https://developer.mozilla.org/ru/docs/Glossary/Void_element).
* Tag omission Must have a start tag and must not have an end tag.
* Permitted parent elements Any element that accepts [phrasing content](https://developer.mozilla.org/en-US/docs/Web/HTML/Content_categories" \l "phrasing_content)(en-US).
* DOM interface [HTMLInputElement](https://developer.mozilla.org/ru/docs/Web/API/HTMLInputElement)

## [Атрибуты](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input" \l "атрибуты)

Этот элемент содержит [глобальные атрибуты (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes).

* Расширение файла, начинающееся с символа точки (U+002E). Наприм., '.jpg, .png, .doc)
* Валидный тип MIME без расширения
* audio/\* для аудиофайлов
* video/\* для видеофайлов
* image/\* для файлов с изображениями

**[type](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input" \l "attr-type)**

Тип элемента для отображения. Если этот атрибут не указан, по умолчанию используется text. Возможными значениями являются:

* button: Кнопка без предопределённого поведения.
* checkbox: Флажок («чекбокс»). Следует использовать атрибут value для определения значения, которое будет отдано этим элементом. Используйте атрибут checked, чтобы указать, должен ли флажок быть выставлен. Можно также использовать атрибут indeterminate, чтобы указать, что флажок находится в неопределённом состоянии (на большинстве платформ при этом рисуется горизонтальная линия поперёк флажка).
* color: Элемент управления цветом. Пользовательский интерфейс выбора цвета не имеет никаких других функций, кроме принятия простых цветов в виде текста ([больше информации](https://www.w3.org/TR/html5/forms.html" \l "color-state-(type=color))).
* date: Элемент управления для ввода даты (год, месяц и день, без времени).
* datetime: Элемент управления для ввода даты и времени (час, минута, секунда и доля секунды) в соответствии с часовым поясом UTC.
* datetime-local: Элемент управления для ввода даты и времени без часового пояса.
* email: Поле для редактирования адреса электронной почты. Перед отправкой проверяется, что входное значение содержит либо пустую строку, либо один действительный адрес электронной почты. Соответствуют CSS псевдоклассам [:valid](https://developer.mozilla.org/ru/docs/Web/CSS/:valid) and [:invalid](https://developer.mozilla.org/ru/docs/Web/CSS/:invalid).
* file: Элемент управления, который позволяет пользователю выбрать файл. Используйте атрибут accept, чтобы определить типы файлов, которые могут быть выбраны.
* hidden: Элемент управления, которые не отображается, но чьё значение отправлено на сервер.
* image: Кнопка вставки изображения. Вы должны использовать атрибут src, чтобы определить путь к изображению и атрибут alt - для определения альтернативного текста. Вы можете использовать атрибуты height и width, чтобы определить размер вставки изображения в пикселях.
* month: Элемент управления для ввода месяца и года без часового пояса.
* number: Элемент управления ввода числа(тип float).
* password: Однострочное текстовое поле, чьё значение скрыто символом "звёздочка". Используйте атрибуты minlength и maxlength, чтобы указать минимальную и максимальную длину значения, которое может быть введено.

Примечание: Любые формы, в которых присутствует важная информация(например, пароль), должны быть обработаны через HTTPS; в настоящий момент Firefox реализует составной механизм предупреждения, направленные против небезопасных форм для входа в систему - смотрите [Небезопасные пароли (en-US)](https://developer.mozilla.org/en-US/docs/Web/Security/Insecure_passwords).

* radio: Кнопка-переключатель, позволяет выбрать одно значение из множественного выбора.
* range: Элемент управления для ввода числа, точное значение которого не имеет значения. Этот тип управления использует следующие значения по умолчанию, если соответствующие атрибуты не указаны:
  + min: 0
  + max: 100
  + value: min + (max-min)/2, or min if max is less than min
  + step: 1
* reset: Кнопка сброса содержимого формы в состояние по умолчанию.
* search: Однострочное текстовое поле для ввода строк поиска; разрывы строк автоматически удаляются из входного значения.
* submit: Кнопка для отправления формы.
* tel: Элемент управления для ввода номера телефона; разрывы строк автоматически удаляются из входного значения, но никакой другой синтаксис не применяется. Можно использовать такие атрибуты как pattern и maxlength, чтобы ограничить вводимое значение. Псевдоклассы CSS [:valid](https://developer.mozilla.org/ru/docs/Web/CSS/:valid) and [:invalid](https://developer.mozilla.org/ru/docs/Web/CSS/:invalid) применяются при необходимости..
* text: Однострочное текстовое поле. Переносы строк автоматически удаляются из входного значения.
* time: Элемент управления для ввода значения времени без часового пояса.
* url: Поле для редактирования URI. Введённое значение должно содержать либо пустую строку, либо допустимый абсолютный URL. В противном случае значение не будет принято. Переводы строк, лидирующие и завершающие пробельные символы будут автоматически удалены из введённого значения. Можно использовать такие атрибуты как pattern или maxlength, чтобы ограничить вводимые значения. Псевдоклассы CSS [:valid](https://developer.mozilla.org/ru/docs/Web/CSS/:valid) and [:invalid](https://developer.mozilla.org/ru/docs/Web/CSS/:invalid) применяются при необходимости.
* week: Элемент управления для ввода даты, содержащей число неделя-год и номер недели без часового пояса.

[Для чего используют data-атрибуты?](https://youtu.be/rlWgI7AvV18?t=34)

До того, как JavaScript-фреймворки стали популярны, фронтенд-разработчики использовали data- атрибуты чтобы хранить дополнительные данные прямо в DOM без хаков вроде нестандартных атрибутов или дополнительных свойств в DOM. Атрибуты этого семейства предназначены для хранения частных данных пользователя, для которых не существует более подходящих атрибутов или элементов на странице или в приложении.

На сегодняшний день использование data-атрибутов не поощряется. Одной из причин является то, что пользователь может модифицировать данные в атрибуте, используя инспектор кода в браузере. Данные лучше хранить в самом JavaScript и обновлять DOM при помощи связывания данных через библиотеку или фреймворк.

###### Ссылки[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "ссылки-3)

* <http://html5doctor.com/html5-custom-data-attributes/>
* [https://www.w3.org/TR/html5/dom.html#embedding-custom-non-visible-data-with-the-data-\*-attributes](https://www.w3.org/TR/html5/dom.html" \l "embedding-custom-non-visible-data-with-the-data-*-attributes)

Минусы: безопасность, загромождение. Сейчас заменили фреймворки.

[Разница между <script>, <script async> и <script defer>?](https://youtu.be/rlWgI7AvV18?t=77)

* <script> - отрисовка HTML блокируется, скрипт извлекается и выполняется немедленно, отрисовка HTML возобновляется после окончания выполнения скрипта.
* <script async> - скрипт будет получен и обработан параллельно с отрисовкой HTML, его выполнение закончится как только это будет возможно (обычно до того, как закончилась отрисовка HTML). Используйте async тогда, когда скрипт не зависит от других скриптов на странице, например для аналитики.
* <script defer> - скрипт будет получен параллельно с отрисовкой HTML, его выполнение произойдет после того, как вся страница будет загружена. Если таких скриптов несколько, то каждый из них будет исполнятся в том порядке, в котором они расположены в документе. Если скрипту нужен полностью распарсеный DOM, то атрибут defer обеспечит уверенность в том, что на момент отработки скрипта весь HTML отрисован. Нет особой разницы со скриптами, расположенными перед тегом <body>. Отложенный скрипт не должен содержать document.write.

Примечание: Атрибуты async и defer игнорируются, если у тега <script> нет атрибута src.

###### Ссылки[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "ссылки-6)

* <http://www.growingwiththeweb.com/2014/02/async-vs-defer-attributes.html>
* <https://stackoverflow.com/questions/10808109/script-tag-async-defer>
* https://bitsofco.de/async-vs-defer/

Для чего используется элемент <datalist>?

### Описание

Создает список вариантов, которые можно выбирать при наборе в текстовом поле. Изначально этот список скрыт и становится доступным при получении полем фокуса или наборе текста.

### Синтаксис

**<input**list="<идентификатор>"**>**

**<datalist**id="<идентификатор>"**>**

**<option**value="Текст1"**>**

**<option**value="Текст2"**>**

**</datalist>**

### Атрибуты

Список, создаваемый тегом **<datalist>**, связывается с текстовым полем посредством атрибута id. Его значение должно совпадать со значением атрибута list тега **<input>**.

### Закрывающий тег

Обязателен.

Пример

HTML 5IE 10CrOpSaFx

<!DOCTYPE html>

**<html>**

**<head>**

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

**<title>**datalist**</title>**

**</head>**

**<body>**

**<p>**Выберите любимого персонажа:**</p>**

**<p><input**list="character"**>**

**<datalist**id="character"**>**

**<option**value="Чебурашка"**></option>**

**<option**value="Крокодил Гена"**></option>**

**<option**value="Шапокляк"**></option>**

**</datalist></p>**

**</body>**

**</html>**

Используется для создания выпадающего списка с возможностью нативной для html реализации автозаполнения для того, чтобы добавить эту возможность на обычный импут элемент. Когда пользователь начинает что-то печатать в инпут поле, ему предлагается список с подходящими вариантами, и он может либо выбрать соответствующий, либо внести свое значение.

### **Почему хорошей практикой считается располагать <link> для подключения CSS между <head></head>, а <script> для подключения JS ставить перед </body>? Знаете ли вы исключения?[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "почему-хорошей-практикой-считается-располагать-link-для-подключения-css-между-headhead-а-script-для-подключения-js-ставить-перед-body-знаете-ли-вы-исключения)**

Размещение <link> внутри <head>

Размещение <link> внутри тега <head> необходимо при создании оптимизированного веб-сайта. Когда страница загружается впервые, HTML и CSS анализируются одновременно; HTML создает DOM (объектную модель документа), а CSS создает CSSOM (объектную модель CSS). И то, и другое необходимо для создания визуальных элементов на веб-сайте, что позволяет быстро определить время «первого значимого рисования». Этот прогрессивный рендеринг является категорией оптимизации сайтов, которые измеряются в их показателях эффективности. Размещение таблиц стилей в нижней части документа - это то, что препятствует прогрессивной загрузке страницы во многих браузерах. Некоторые браузеры блокируют рендеринг, чтобы избежать перерисовки элементов страницы, если ее стили изменятся. Все это время пользователь будет пялиться на пустую белую страницу. В других случаях может возникать мерцание нестилизованного содержимого (FOUC), на котором может отображаться веб-страница без применения стилей.

Размещение <script> прямо перед </body>

Теги <script> блокируют отрисовку HTML на то время, пока они скачиваются и исполняются. Размещение скриптов внизу позволяет сперва распарсить и показать пользователю весь HTML.

Исключением является случай, когда в вашем скрипте содержится document.write(). Но на сегодняшний день его использование не считается хорошей практикой. К тому же, расположение скриптов внизу разметки означает, что браузер не может начать их скачивать до тех пор, пока не отрисован весь документ. Единственным рабочим способом, при котором <script> будет расположен внутри <head>, является добавление атрибута defer.

###### Ссылки[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "ссылки-7)

* [https://developer.yahoo.com/performance/rules.html#css\_top](https://developer.yahoo.com/performance/rules.html" \l "css_top)
* <https://www.techrepublic.com/blog/web-designer/how-to-prevent-flash-of-unstyled-content-on-your-websites/>
* <https://developers.google.com/web/fundamentals/performance/critical-rendering-path/>

[Что такое мета-тэги?](https://youtu.be/ngyOYuTrUk8?t=205)

Метатеги — служебные теги в разметке HTML. Они предназначены для указания сведений поисковым роботам и браузерам. Взгляните на вкладку в браузере с этим уроком и вы увидите текст, с помощью которого всегда сможете найти нужную вкладку, открытую в браузере.

При выводе результатов поискового запроса (например в Google или Yandex) также показывается этот заголовок и описание.

При добавлении статьи с сайта на страницу в социальных сетях появляется картинка и описание.

Всё это — работа метатегов. Именно в них можно определить заголовок и описание для текущей HTML-страницы, а также картинку, которая добавится при копировании ссылки в социальных сетях.

## Тег title

За заголовок во вкладке браузера отвечает парный тег <title>, внутри которого содержится название страницы.

Обычно советуют использовать заголовок от 50 до 80 символов. Такое ограничение существует из-за того, что поисковые системы не могут показать больше символов в качестве заголовка страницы в результатах поисковой выдачи. Заголовок внутри тега <title> отобразится во вкладке в браузере.

<title>Code Basics: основы программирования</title>

## Тег meta

Большинство метатегов записываются с использованием непарного тега <meta>. Он принимает несколько основных атрибутов, которые помогают установить метаинформацию

### Описание

Описание часто используется при добавлении ссылки на страницу в социальных сетях. Эта информация используется поисковыми системами при выводе страницы в результатах поиска.

Чтобы установить описание, используется тег <meta> с двумя атрибутами:

* name="description". Атрибут name указывает на то, какой тип метаинформации описывается в теге
* content. Атрибут content указывает саму информацию

<meta name="description" content="Бесплатные практические уроки по программированию для тех, кто начинает с нуля.">

### Ключевые слова

В разработке для страниц часто указывают ключевые слова. Они помогают роботам корректнее собирать информацию о странице, что хорошо сказывается на продвижении сайта в интернете. Как и с описанием, для установки ключевых слов используется два атрибута. В качестве значения атрибута name используется keywords. Сами ключевые слова перечисляются через запятую

<meta name="keywords" content="программирование, курсы, HTML, CSS">

### Кодировка страницы

Существует большое количество кодировок — таблиц символов, по которым происходит поиск буквы или символа. Раньше проблема кодировок стояла очень остро — почти для каждого языка использовалась уникальная кодировка, из-за чего открыть документ на другом языке было невозможно.

В текущее время есть универсальные кодировки, которые поддерживают символы самых разных языков, но считается хорошим тоном, в качестве метатега, указывать кодировку страницы.

Для этого указывается только один атрибут — charset, значением которого выступает название кодировки. В 99% случаев это UTF-8. Если вы встретитесь с другой кодировкой, то, скорее всего, об этом вам сообщат заранее

<meta charset="UTF-8">

[**Что описывается в тэге <head>?**](https://youtu.be/ngyOYuTrUk8?t=252)

Тег **<head>** предназначен для хранения других элементов, цель которых — помочь браузеру в работе с данными. Также внутри контейнера **<head>** находятся метатеги, которые используются для хранения информации предназначенной для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных.

Содержимое тега **<head>** не отображается напрямую на веб-странице, за исключением тега **<title>** устанавливающего заголовок окна веб-страницы.

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

### Синтаксис

**<head>**

...

**</head>**

### Закрывающий тег

Не обязателен.

### Атрибуты

[profile](http://htmlbook.ru/html/head/profile)

Указывает адрес профиля метаданных.

Также для этого тега доступны [глобальные атрибуты](http://htmlbook.ru/html/attr/common).

[More info](https://developer.mozilla.org/ru/docs/Learn/HTML/Introduction_to_HTML/The_head_metadata_in_HTML)

[Для чего используются тэги <tr>, <th>, <td>?](https://youtu.be/ngyOYuTrUk8?t=306)

<table>  
  <tr>  
    <th>Person 1</th>  
    <th>Person 2</th>  
    <th>Person 3</th>  
  </tr>  
  <tr>  
    <td>Emil</td>  
    <td>Tobias</td>  
    <td>Linus</td>  
  </tr>  
  <tr>  
    <td>16</td>  
    <td>14</td>  
    <td>10</td>  
  </tr>  
</table>

<https://www.w3schools.com/html/html_tables.asp>

[Расскажите о meta-теге с name="viewport"?](https://youtu.be/rWEsjNWBoIE?t=36)

Viewport - это видимая пользователю область веб-страницы, то, что может увидеть пользователь, не прибегая к прокрутке.

Окно просмотра представляет собой многоугольную (обычно прямоугольную) область в компьютерной графике, которая в настоящее время просматривается. В терминах веб-браузера это относится к той части просматриваемого документа, которая в данный момент отображается в её окне (или на экране, если документ просматривается в полноэкранном режиме). Содержимое вне окна просмотра не отображается на экране до тех пор, пока оно не будет просмотрено.

Meta-тег viewport сообщает браузеру о том, как именно обрабатывать размеры страницы, и изменять её масштаб. Этот тег необходимо добавлять в секцию HEAD.

Пример:

<meta name="viewport" content="width=device-width, initial-scale=1">

### [А](https://developer.mozilla.org/ru/docs/Glossary/Viewport" \l "атрибуты_метатега_viewport)трибуты метатега viewport

Copy to Clipboard

Meta-тег viewport может иметь следующие атрибуты, указанные через запятую (,):

width – ширина области просмотра.

Значением атрибута является целое неотрицательное число от 200 до 10000 пикселей или константа device-width, которая задаёт ширину страницы в соответствии с размером экрана.

Если значение не задано, по умолчанию устанавливается – в мобильном Safari = 980px, Opera = 850px, Android WebKit = 800px, IE = 974px.

Примечание: для сайтов с адаптивным дизайном рекомендуется использовать: width=device-width.

height – высота области просмотра.

Значением атрибута является целое неотрицательное число от 233 до 10000 пикселей или константа device-height, которая задаёт высоту страницы в соответствии с размером экрана.

Примечание: если указан атрибут width, указывать атрибут height не обязательно.

initial-scale – начальный масштаб страницы.

Значением атрибута является вещественное число от 0.1 до 1.0. Значение 1.0 определяет масштаб 1:1, т.е. «не масштабировать».

Примечание: в некоторых операционных системах (iOS, Windows Phone и т.д.) ширина страницы, при повороте, остаётся неизменной. Вместо перераспределения контента выполняется его масштабирование. Поэтому рекомендуется использовать: initial-scale=1.0.

user-scalable – доступность масштабирования страницы пользователем.

Значение атрибута является логическое «yes» (1) – можно масштабировать или «no» (0)– нельзя масштабировать.

Примечание: рекомендуется использовать значение «yes» , а т.к. оно установлено по умолчанию, то user-scalable можно и не указывать.

minimum-scale – минимальный масштаб области просмотра.

Значением атрибута является вещественное число от 0.1 до 1.0. В мобильном браузере Safari по умолчанию 0.25. Значение 1.0 определяет масштаб 1:1, т.е. «не масштабировать».

maximum-scale – максимальный масштаб области просмотра.

Значением атрибута является вещественное число от 0.1 до 1.0 . В мобильном браузере Safari по умолчанию 1.6. Значение 1.0 определяет масштаб 1:1, т.е. «не масштабировать».

Примечание: избегайте атрибутов user-scalable, minimum-scale и maximum-scale, т.к. они отрицательно сказываются на доступности содержания.

## [Дополнительные и полезные метатеги](https://developer.mozilla.org/ru/docs/Glossary/Viewport" \l "дополнительные_и_полезные_метатеги)

Meta-тег HandheldFriendly определяет оптимизирована ли страница сайта под мобильные устройства на Palm и Blackberry, в таком браузере как AvantGo. Сейчас распознаётся и многими другими мобильными браузерами.

Пример:

<meta name="HandheldFriendly" content="true">

Meta-тег MobileOptimized (<http://goo.gl/ZpLjZz>) задаёт ширину области просмотра в мобильных браузеров IE Mobile или Pocket IE. Является аналогом атрибута width в meta-теге viewport.

Copy to Clipboard

Пример:

<!-- фиксированная ширина в 320 пикселей -->

<meta name="MobileOptimized" content="320">

<!-- ширина страницы в соответствии с размером экрана, аналог device-width -->

<meta name="MobileOptimized" content="width">

Meta-тег apple-mobile-web-app-capable (<http://goo.gl/VGDYQC>) позволяет странице работать в полноэкранном режиме, актуален для мобильных устройств Apple.

Copy to Clipboard

Пример:

<meta name="apple-mobile-web-app-capable" content="yes">

## [Р](https://developer.mozilla.org/ru/docs/Glossary/Viewport" \l "рекомендованный_набор_метатегов)екомендованный набор метатегов

Copy to Clipboard

Используемый мной набор meta-тегов для сайтов с адаптивным дизайном, заточенным под мобильные устройства:

<meta name='viewport' content='width=device-width,initial-scale=1'/>

<meta content='true' name='HandheldFriendly'/>

<meta content='width' name='MobileOptimized'/>

<meta content='yes' name='apple-mobile-web-app-capable'/>

[Что такое элемент <canvas>? И для чего он используется?](https://youtu.be/CjdCxxqObaM?t=200)

[<canvas>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/canvas) — это [HTML](https://developer.mozilla.org/ru/docs/Web/HTML) элемент, использующийся для рисования графики средствами языков программирования (обычно это [JavaScript](https://developer.mozilla.org/ru/docs/Glossary/JavaScript)). Он может, к примеру, использоваться для рисования графов, создания коллажей или простой ([и не очень](https://developer.mozilla.org/ru/docs/orphaned/Web/API/Canvas_API/A_basic_ray-caster)) анимации.

Canvas - что это такое? На первый взгляд, элемент <canvas> - это элемент HTML5, который можно использовать для рисования. Он представляет собой лишь контейнер, в котором можно рисовать что-либо с помощью других методов. HTML canvas используется для отображения графиков, диаграмм, анимации и чтобы обойти проблемы с поддержкой CSS в некоторых браузерах.

Что можно рисовать с помощью <canvas>: арки, кривые (их различные типы), линии и прямоугольники. Конечно же, их можно комбинировать, и за счет этого создавать практически любые формы или объекты с различными эффектами (прозрачность, смешивание, градиенты, тени, заливки и контуры).

[Что такое <svg> и <canvas>?](https://youtu.be/rWEsjNWBoIE?t=287)

SVG и canvas — это технологии, которые можно использовать для рисования чего-либо на веб-страницах. Поэтому их стоит сравнить и разобраться в том, когда стоит применять SVG, а когда — canvas. Даже весьма поверхностное понимание сути этих технологий позволяет сделать вполне осознанный выбор. Собственно говоря, вот — две типичных ситуации, в одной из которых стоит предпочесть SVG, а в другой — canvas:

Нужно нарисовать небольшую иконку? Это, безусловно, территория SVG.

Нужно создать интерактивную браузерную игру? Тут, определённо, нужна технология canvas.

**SVG — векторная графика, описываемая декларативно**

Если вы знаете о том, что вам нужно разместить на веб-странице векторное изображение, это значит, что вам подходит SVG. Векторные изображения обычно выглядят лучше аналогичных растровых, хранящихся, например, в JPG-файлах. Кроме того, файлы векторных изображений обычно меньше файлов растровых изображений.

В результате оказывается, что SVG очень часто используется для рисования логотипов. Код, описывающий такие изображения, можно встроить прямо в HTML. Он выглядит как набор декларативных инструкций:

Если вам нужно, чтобы изображения были бы гибкими и отзывчивыми, значит SVG — это ваш выбор.

**Canvas — JavaScript-API для рисования**

Для формирования canvas-изображений в HTML-коде страницы размещают элемент <canvas>, после чего, средствами JavaScript, «рисуют» на этом элементе. Другими словами, программист даёт браузеру команды, касающиеся того, как ему нужно что-то нарисовать (такой подход ближе к императивному, чем к декларативному). Вот как это выглядит:

**SVG-изображения хранятся в DOM**

Если вы знакомы с событиями DOM, наподобие click и mouseDown, то знайте о том, что этими событиями можно пользоваться и при работе с SVG. В этом плане SVG-элемент <circle> не особенно сильно отличается от HTML-элемента <div>.

SVG лучше подходит для обеспечения доступности содержимого страниц

Элементу canvas вполне можно назначит его текстовую альтернативу:

То же самое можно сделать и для SVG-элемента. Но, так как SVG-изображения и их внутренние механизмы хранятся прямо в DOM, SVG обычно рассматривают как технологию, используемую в тех случаях, когда создают проекты, отличающиеся доступностью для людей с ограниченными возможностями.

Другими словами — можно создать SVG-изображение, с элементами которого смогут работать ассистивные технологии, помогающие своим пользователям ориентироваться на веб-страницах.

Текстовые данные, являющиеся частью изображений, это тоже тот случай, когда предпочтительнее использовать SVG. В SVG даже имеется специальный элемент — <text>. Он позволяет выводить чёткие символы текстов, его воспринимают ассистивные технологии. В случае же использования для визуализации текстов элемента canvas, такие тексты нередко выводятся некачественно, выглядят размытыми.

**Canvas лучше подходит для работы с изображениями на уровне отдельных пикселей**

**Комбинация SVG и canvas**

С технической точки зрения технологии canvas и SVG нельзя назвать взаимоисключающими. Например, SVG-изображения можно выводить в элементы <canvas>. При этом, как можно видеть в данном примере, SVG-изображения, выводимые в элемент <canvas>, вполне могут оставаться весьма чёткими и качественными.

<https://habr.com/ru/companies/ruvds/articles/476292/>

http://html5ru.com/canvas-vs-svg.html

Итак, вопрос, который по определению провокационный и флеймообразующий: что использовать — canvas или SVG, — на первый взгляд, он не очень правомерен: SVG — это векторная графика, Canvas представляет растовые изображения. Но тем не менее области применения обеих технологий на веб-страницах сильно пересекаются. Какими преимуществами обладает каждая из них и на какой все-таки остановиться?

Объекты в svg — это часть DOM-дерева документа, они доступны из JavaScrip-сценариев, к ним можно привязать события (такие как click или mouseover) и организовать интерактивное взаимодействие пользователя с контентом. К этим элементам можно непосредственно применять CSS-стили.

Кроме этого, у данного формата есть еще два, не столь очевидных преимущества. Первое — это возможность масштабирования: картинку в SVG можно увеличивать в размерах без видимой потери качества, чего нельзя сказать про битмапные изображения canvas.

Второе преимущество заключается в возможности индексации csg-контента поисковыми системами. И это не просто теоретическая возможность — поисковые механизмы Google индексируют SVG уже почти два года, а это серьезно!

Следует ли из этого, что SVG однозначно удобнее и лучше по всем параметрам? Естественно, нет. Применение SVG нам дает невероятную простоту при рисовании… простых вещей.

В Canvas необходимо будет реализовывать то, что уже реализовано в SVG в виде XML-объектов. С другой стороны, тут без всяких примитивов можно отрисовывать произвольные формы, на реализацию которых с помощью SVG уйдет довольно много кода, требующего, в свою очередь, больших процессорных ресурсов для интерпретации.

Такие же вещи, как попиксельная работа с изображениями, манипуляции с видео и многие другие, для SVG просто невозможны.

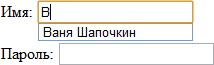
Еще одно преимущество Canvas — возможности оптимизации и кэширования графики.

Подытоживая, можно сказать, что сфера применения SVG — это ифографика, диаграммы, схемы, графики, иллюстрации, в то время как сфера действия canvas — видеоигры, обработка фотографий, «веб-живопись». Впрочем, это всего лишь мое мнение, которое я никому не навязываю.

[Для чего нужен атрибут autocomplete?](https://youtu.be/rWEsjNWBoIE?t=612)

Этот атрибут помогает заполнять поля форм текстом, который был введён в них ранее. Значения сохраняет и подставляет браузер, при этом автозаполнение по соображениям безопасности может отключаться пользователем в настройках и не может в таком случае управляться атрибутом autocomplete. К примеру, включение автозаполнения в браузере Chrome показано на рис. 1.

При вводе первых букв текста выводится список сохранённых ранее значений, из которого можно выбрать необходимое (рис. 2).



*Рис. 2. Список для подстановки*

### Синтаксис

**<input**type="text"autocomplete="on | off"**>**

**<input**type="password"autocomplete="on | off"**>**

**<input**type="email"autocomplete="on | off"**>**

**<input**type="search"autocomplete="on | off"**>**

**<input**type="url"autocomplete="on | off"**>**

**<input**type="tel"autocomplete="on | off"**>**

### Значения

on

Включает автозаполнение текста.

off

Отключает автозаполнение. Это значение обычно используется для отмены сохранения в браузере важных данных (паролей, номеров банковских карт), а также редко вводимых или уникальных данных (капча).

### Обязательный атрибут

Нет.

### Значение по умолчанию

Зависит от настроек браузера.

Пример

HTML5IECrOpSaFx

<!DOCTYPE html>

**<html>**

**<head>**

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

**<title>**Тег input, атрибут autocomplete**</title>**

**</head>**

**<body>**

**<form>**

**<p>**Имя: **<input**name="user"autocomplete="on"**></p>**

**<p>**Пароль: **<input**name="pass"type="password"**></p>**

**<p><input**type="submit"value="Отправить"**></p>**

**</form>**

**</body>**

**</html>**

[Что такое элемент <output> в HTML5?](https://youtu.be/rWEsjNWBoIE?t=669)

HTML-элемент вывода (<output>) является контейнерным элементом, в котором сайт или приложение могут выводить результаты вычислений или действий пользователя.

| [Категории контента](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories) | [Потоковый контент](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "flow_content), [фразовый контент](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "phrasing_content), [перечисляемые](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "form_listed), [лэйбловые](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "form_labelable), [сбрасываемые форма-ассоциированные элементы](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "form_resettable), явный контент. |
| --- | --- |
| Допустимый контент | [Фразовый контент](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "phrasing_content). |
| Пропуск тегов | Нет, открывающий и закрывающий теги обязательны. |
| Допустимые родители | Все элементы, которые принимают [фразовый контент](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "phrasing_content). |
| Допустимые ARIA-роли | Любые |
| DOM-интерфейс | [HTMLOutputElement (en-US)](https://developer.mozilla.org/en-US/docs/Web/API/HTMLOutputElement) |

## [Атрибуты](https://developer.mozilla.org/ru/docs/Web/HTML/Element/output" \l "атрибуты)

Этот элемент включает [глобальные атрибуты](https://developer.mozilla.org/ru/docs/Web/HTML/Global_attributes).

**[for](https://developer.mozilla.org/ru/docs/Web/HTML/Element/output" \l "attr-for)**

Пробело-разделяемый список [id](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/id) других элементов, указывающий, что эти элементы предоставили входные значения для (или иным образом повлияли) вычисления.

**[form](https://developer.mozilla.org/ru/docs/Web/HTML/Element/output" \l "attr-form)**

[Элемент формы](https://developer.mozilla.org/ru/docs/Web/HTML/Element/form), с которым связан этот элемент ("владелец формы"). Значением атрибута должен быть id элемента [<form>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/form) в том же документе. Этот атрибут не нужен, если элемент <output> является потомком элемента <form> (в этом случае эта форма является владельцем формы), или, если элемент <output> вообще не связан с формой.

**[name](https://developer.mozilla.org/ru/docs/Web/HTML/Element/output" \l "attr-name)**

Имя элемента; используется для идентификации этого <output> при отправке формы.

## [Пример](https://developer.mozilla.org/ru/docs/Web/HTML/Element/output" \l "пример)

Эта форма содержит слайдер, значение которого находится в пределах от 0 до 100, и элемент [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/input), в который можно ввести второе значение. Два числа складываются и результат отображается в элементе <output> каждый раз, когда значение любого управляющего элемента меняется.

<form oninput="result.value=parseInt(a.value)+parseInt(b.value)">

<input type="range" name="b" value="50" /> +

<input type="number" name="a" value="10" /> =

<output name="result">60</output>

</form>

[Что такое свойство valueAsNumber?](https://youtu.be/rWEsjNWBoIE?t=717)

В HTML5 представлено свойство JavaScript valueAsNumber для полей формы (в частности: number, date, range). Оно возвращает значение в виде числа, а не строки, то есть нам больше не нужно использовать parseInt или parseFloat, и оператор + складывает, а не склеивает.

Пример 3. Использование свойства valueAsNumber для получения числового значения из полей

**<form**onsubmit="return false"oninput="o.value = a.valueAsNumber + b.valueAsNumber"**>**

**<input**name="a"id="a"type="number"step="any"**>** +

**<input**name="b"id="b"type="number"step="any"**>** =

**<output**name="o"for="a b"**></output>**

[Что такое атрибут target? Какие значения он принимает?](https://youtu.be/ovV8GhIkzBE?t=30)

Атрибут **target** определяет, где открывать документ, на который ведет ссылка.

Допустим только, если определен атрибут [href](https://msiter.ru/node/4287).

В HTML5 фреймы больше не поддерживаются. Значения \_top, \_parent и имя\_фрейма теперь используются в основном с элементами **<iframe>**.

#### Синтаксис атрибута

<a target="\_blank | \_self | \_parent | \_top | имя\_фрейма">

#### Значения атрибута

| Значение | Описание |
| --- | --- |
| \_blank | Открывает ссылку в новом окне браузера. |
| \_self | Значение по умолчанию. Открывает ссылку в том же окне браузера. |
| \_parent | Открывает ссылку в родительском фрейме, если фреймов нет, то работает как значение \_self |
| \_top | Открывает ссылку в полном окне браузера, если фреймов нет, то работает как значение \_self |
| имя\_фрейма | Открывает ссылку в фрейме с указанным именем. |

[Что такое ApplicationCache в HTML5?](https://youtu.be/ovV8GhIkzBE?t=92)

Application Cache API — совокупность функций, обеспечивающих продвинутое кэширование ресурсов web-приложения, и с помощью которых можно просматривать загруженные ранее сайты без подключения к сети Интернет. Особое внимание я уделил практическому использованию и проблемам Application Cache.

Copy to Clipboard

Далее по статье под словами «кэш» и его производными имеется в виду Application Cache и работа с ним. Стандартный кэш браузера обозначается как «стандартный кэш».

Механизм работы Application Cache на первый взгляд прост. Когда пользователь первый раз пользуется сайтом, страницы запоминаются в кэше браузера. Затем при последующих посещениях, а также когда соединение с Интернет теряется, используются заранее сохраненные данные в специальном «хранилище». Контроль над кэшированием сайта производится с помощью специального файла, названного manifest. В некоторых статьях используется .apppcache вместо .manifest. В данной статье используется .manifest. Хотя можно использовать и свое название.

«Хранилище» — условное название, используемое в браузерах для обозначения места хранения Application Cache. Механизмы работы Application Cache и стандартного кэша браузера различны, поэтому требуют отдельного размещения. Отличия механизмов работы между двумя видами кэширования следующие:

Данные помещённые в стандартный кэш могут быть автоматически, без команды со стороны пользователя или сервера, удаленны при его заполнении или истечения стока действия, указанного в заголовках файлов. Данные помещённые в хранилище Application Cache могут быть удалены только по команде пользователя или сервера.

В стандартный кэш попадают только файлы, загруженные в процессе просмотра страницы. В Application Cache можно поместить любые файлы, загружаемые с сервера согласно инструкции manifest.

[More info](https://habr.com/ru/articles/151815/) [More](http://html5ru.com/appcache-upravlyaem-keshirovaniem-vplot-do-polnogo-offline.html" \l ":~:text=HTML5 предполагает технологию кэширования ресурсов,в отсутствие связи с сервером.)

[Для чего используется элемент <picture>?](https://youtu.be/70VnuTXi4Wk?t=189)

\*\*HTML-элемент <picture> \*\*служит контейнером для одного или более элементов [<source>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/source) и одного элемента [<img>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/img) для обеспечения оптимальной версии изображения для различных размеров экрана. Браузер рассмотрит каждый из дочерних элементов <source> и выберет один, соответствующий лучшему совпадению; если совпадений среди элементов <source> найдено не будет, то будет выбран файл, указанный атрибутом [src](https://developer.mozilla.org/ru/docs/Web/HTML/Element/img" \l "attr-src) элемента <img> . Затем выбранное изображение отображается в пространстве, занятом элементом <img>.

Чтобы выбрать оптимальное изображение, [user agent](https://developer.mozilla.org/ru/docs/Glossary/User_agent) анализирует атрибуты [srcset](https://developer.mozilla.org/ru/docs/Web/HTML/Element/source" \l "attr-srcset), [media](https://developer.mozilla.org/ru/docs/Web/HTML/Element/source" \l "attr-media), и [type](https://developer.mozilla.org/ru/docs/Web/HTML/Element/source" \l "attr-type) элемента <source> и выбирает совместимое изображение, которое наилучшим образом соответствует текущему макету страницы, характеристикам устройства отображения и т. д.

Элемент <picture> часто используют для того, чтобы обеспечить HiDPI (Retina)-версию изображений для дисплеев с высоким разрешением, а также для добавления других форматов изображений, не поддерживаемых всеми браузерами.

| [Категории контента](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories) | [Потоковый контент](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories" \l "flow_content), фразовый контент, встроенный контент |
| --- | --- |
| Допустимый контент | Ноль или более элементов [<source>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/source), затем один элемент [<img>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/img), при необходимости смешанный с элементами поддержки сценариев. |
| Пропуск тегов | Нет, открывающий и закрывающий теги обязательны. |
| Допустимые родители | Любой элемент, который разрешает встроенный контент |
| Допустимые ARIA-роли | Нет |
| DOM-интерфейс | [HTMLPictureElement (en-US)](https://developer.mozilla.org/en-US/docs/Web/API/HTMLPictureElement) |

## [Атрибуты](https://developer.mozilla.org/ru/docs/Web/HTML/Element/picture" \l "атрибуты)

Этот элемент содержит только [универсальные атрибуты](https://developer.mozilla.org/ru/docs/Web/HTML/Global_attributes).

## [Примеры](https://developer.mozilla.org/ru/docs/Web/HTML/Element/picture" \l "примеры)

Эти примеры демонстрируют влияние различных атрибутов элемента [<source>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/source) на выбор изображения, которое будет отображаться при их использовании внутри тега <picture>.

### [Атрибут](https://developer.mozilla.org/ru/docs/Web/HTML/Element/picture" \l "атрибут_media)media

Атрибут media позволяет определить медиавыражение, которое веб-браузер будет анализировать для выбора элемента [<source>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/source). Если медиавыражение определяется как ложное (false), то элемент [<source>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/source) пропускается.

<picture>

<source srcset="mdn-logo-wide.png" media="(min-width: 600px)">

<img src="mdn-logo-narrow.png" alt="MDN">

</picture>

### [А](https://developer.mozilla.org/ru/docs/Web/HTML/Element/picture" \l "атрибут_type)трибут type

Copy to Clipboard

Атрибут type позволяет вам обозначить MIME-тип данных, указанных в атрибуте srcset элемента [<source>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/source). Если браузер не поддерживает данный тип, то элемент [<source>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/source) опускается.

<picture>

<source srcset="mdn-logo.svg" type="image/svg+xml">

<img src="mdn-logo.png" alt="MDN">

</picture>

[Что такое srcset? Как работает srcset?](https://youtu.be/70VnuTXi4Wk?t=254)

Copy to Clipboard

Атрибут srcset предназначен для объявления адаптивных изображений. Браузер самостоятельно анализирует размер экрана пользователя и решает, какую из объявленных картинок (т.е. какой размер картинки) выбрать для демонстрации. Далее вы видите один из стандартных примеров объявления srcset в коде, который позволит разобраться с синтаксисом данного атрибута:

<img src="small.png" srcset="medium.png 500w, large.png 1000w" alt="img">

Указанный выше код отвечает только за объявление изображений разных размеров, а всю остальную работу, которая заключается в подборе оптимального размера для устройства пользователя, выполняет браузер.

В качестве вспомогательного инструмента вы можете воспользоваться онлайн-сервисом [MyDevice.io](https://timeweb.com/go?url=https%3A%2F%2Fwww.mydevice.io%2F&hash=03a1d03010efb358c78e3b008e9a39defcc0788d). С помощью него вы узнаете ширину области просмотра текущего дисплея и другую информацию об устройстве.

Вычисления зависят не только от размера экрана, но и от плотности пикселей, поэтому алгоритм подбора оптимального изображения всегда будет разным.

Комьюнити теперь в Телеграм

Подпишитесь и будьте в курсе последних IT-новостей

[Подписаться](https://t.me/twcommunity)

## Зачем нужны адаптивные изображения

У некоторых пользователей при прочтении данной статьи может возникнуть вопрос по поводу того, зачем вообще использовать отзывчивые изображения. Разве не проще загрузить одну картинку и сжимать ее, если это потребуется? Дело в том, что при изначально большом размере картинки (например, 4516px/3Мб) средство оптимизации сайта не сработает так быстро, как этого хотелось бы. Соответственно, скорость загрузки контента несколько замедлится.

Если же вместо этого загрузить уменьшенную копию, обладатели больших экранов увидят ее в сжатом качестве, что тоже плохо. Оптимальным вариантом будет использование рассматриваемого атрибута и загрузка сразу нескольких изображений, чтобы браузер самостоятельно принимал решение о том, какое из них выбрать. Этот фрагмент кода обретает примерно следующий вид:

<img srcset="picture-640.jpg 640w,

             picture-920.jpg 920w,

             picture-1280.jpg 1280w,

             picture-1600.jpg 1600w"

     src="picture.jpg" alt="Картинка">

При таком раскладе браузер не будет загружать полноформатное изображение для тех пользователей, которые заходят на сайт с мобильного устройства, что благоприятно скажется на скорости загрузки страницы.

## Принцип обработки адаптивных изображений

Вкратце разберу то, как браузер подбирает оптимальное изображение для загрузки. Весь этап можно разделить на несколько последовательных шагов:

1. Определение ширины картинки в верстке, то есть ее CSS-пикселей.
2. Умножение полученного значения на соотношение сторон устройства (pixel ratio).
3. Определение реальной ширины и подбор картинки.
4. Алгоритм браузера переходит к массиву с изображениями, в котором ищет максимально близкий по значению файл.

[Как семантически верно сверстать навигационное меню?](https://youtu.be/VYQl2GhbCUs?t=376)

HTML-элемент <nav> определяет отдельную секцию документа, назначение которой обозначение ссылок навигации (как внутри текущего документа, так и ведущих на другую страницу). В качестве примера такой секции можно привести меню, якорные ссылки.

| [Категории контента](https://developer.mozilla.org/ru/docs/Web/HTML/Content_categories) | Потоковый контент, секционный контент, явный контент. |
| --- | --- |
| Допустимое содержимое | Потоковый контент. |
| Пропуск тегов | Нет, открывающий и закрывающий теги обязательны. |
| Допустимые родители | Любой элемент, содержащий потоковый контент. |
| Допустимые ARIA-роли | Нет |
| DOM-интерфейс | [HTMLElement](https://developer.mozilla.org/ru/docs/Web/API/HTMLElement) |

## [Атрибуты](https://developer.mozilla.org/ru/docs/Web/HTML/Element/nav" \l "атрибуты)

Этот элемент поддерживает [глобальные атрибуты](https://developer.mozilla.org/ru/docs/Web/HTML/Global_attributes).

## [Примечание](https://developer.mozilla.org/ru/docs/Web/HTML/Element/nav" \l "примечание)

* Не обязательно все ссылки должны быть обёрнуты в <nav>. <nav> следует использовать лишь для главных навигационных блоков. Например, [<footer>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/footer) часто содержит список ссылок, которые не стоит оборачивать в [<nav>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/nav) .
* Документ может содержать несколько [<nav>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/nav) элементов. Например, один для навигации по сайту, второй для навигации по странице.
* Пользовательские агенты, такие как устройства чтения с экрана, предназначенные для людей с плохим зрением, могут использовать этот элемент, чтобы определить следует ли пускать рендеринг содержимого навигации.

## [Примеры](https://developer.mozilla.org/ru/docs/Web/HTML/Element/nav" \l "примеры)

В данном примере, блок <nav> содержит ненумерованный список ([<ul>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/ul)) ссылок. С помощью CSS данный блок можно использовать как сайдбар, навигационную колонку или выпадающее меню.

<nav class="menu">

<ul>

<li><a href="#">Главная</a></li>

<li><a href="#">О нас</a></li>

<li><a href="#">Контакты</a></li>

</ul>

</nav>

[Что такое <iframe>?](https://youtu.be/VYQl2GhbCUs?t=415)

iFrame или плавающий фрейм — отдельное окно, HTML-документ, который отображается вместе с другим содержимым страницы в окне браузера. Располагаться он может в любом месте, без привязки к тегу <iframe>.

Использование данного тега зачастую рискованно получением фильтра: сайт перестает полностью контролировать содержимое документа, так как его часть подгружается с другого сайта через iframe.

Поскольку мы перестаем контролировать содержимое, то через iframe возможна подгрузка нежелательного контента (18+, вирусы и прочее), что, в свою очередь, может привести к санкциям. А это отразится на продвижении в выдаче.

https://serpstat.com/ru/blog/chto-takoe-iframe/

[Для чего используются тэги <sub> и <sup>?](https://youtu.be/VYQl2GhbCUs?t=491)

Tег <sup> (сокращение от «superscript») отображает текст в виде верхнего индекса, а тег <sub> (сокращение от «subscript») в виде нижнего индекса.

Эти теги чаще используются для выделения отдельных символов, а не слов. Их используют для указания единиц измерения или написания простых формул. Например:

20м<sup>2</sup>

H<sub>2</sub>O

X<sup>3</sup>+X<sup>2</sup>=1

[Как можно скрыть элемент разметки не используя CSS и JS?](https://youtu.be/VYQl2GhbCUs?t=539)

Hidden — это логический HTML-атрибут, скрывающий элементы, которым он назначен. Когда браузер загружает страницу, он не выведет элементы с атрибутом hidden, за исключением тех случаев, когда видимость элементов будет включена вручную средствами CSS. Действие этого атрибута похоже на применение к элементу CSS-правила display: none.  
  
Рассмотрим следующий пример:

<**h1**>Spring is on the way</**h1**>

<**img** hidden src="landscape.jpg" alt="">

<**p**><!-- Description content --></**p**>

Тут имеется разметка, задающая заголовок, изображение и описание. Изображение должно выводиться только в том случае, если ширина области просмотра превышает 400px. К элементу <img> я добавил атрибут hidden.  
  
В CSS я воспользовался атрибутом hidden для вывода элемента только в том случае, если область просмотра страницы имеет необходимый размер.

Тут у вас может появиться вопрос о том, почему бы просто не использовать display: none. Хороший вопрос. Когда селектор изображения вызывается через его атрибут hidden, мы можем быть уверены в том, что даже если CSS-код по какой-то причине не загрузился, элемент будет скрыт.

### ▍Атрибут hidden и доступность контента

Если рассмотреть атрибут hidden с точки зрения доступности контента, то окажется, что этот атрибут полностью скрывает элемент. В результате с этим элементом не смогут работать средства для чтения с экрана. Не используйте этот атрибут в тех случаях, когда некие элементы страниц нужно делать невидимыми для человека, но не для программ для чтения с экрана.

https://habr.com/ru/companies/ruvds/articles/485640/

[Разница между <meter> и <progress>?](https://youtu.be/VYQl2GhbCUs?t=608)

[Элемент meter](http://www.w3.org/TR/html5/forms.html" \l "the-meter-element) представляет собой числовое (возможно дробное) значение в **известном диапазоне**. Этот элемент не подходит для измерения чего-то вроде температуры, поскольку она не имеет фиксированного диапазона. Однако meter может, к примеру, использоваться для отражения занятого места на жестком диске.

[Элемент progress](http://www.w3.org/TR/html5/forms.html" \l "the-progress-element) используется для отображения хода выполнения задачи. В отличие от элемента meter, прогресс, описанный в progress, может быть неокончательным. Например, можно написать, что задание выполняется, но неизвестно, когда задача будет выполнена до конца.

https://www.geeksforgeeks.org/what-are-progress-and-meter-tags-in-html5/

[Как можно сгруппировать опции внутри тэга <select>?](https://youtu.be/trriSYNrHw4?t=304)

Тег <optgroup> позволяет группировать опции внутри элемента <select>. Внутри тега <optgroup> стандартно используется один или несколько тегов <option>.

Как пишется

Нельзя вкладывать один <optgroup> в другой.

Возможные атрибуты:

disabled — находящиеся внутри группы опции станут недоступны для выбора.

<select>

<optgroup label="Необычные цветы" disabled>

<option>Ангулоя одноцветковая</option>

<option>Обезьяний дракула</option>

<option>Пассифлора инкарнатная</option>

</optgroup>

</select>

label — обязательный атрибут. Имя группы будет отображено в выпадающем списке. Теоретически мы можем оставить атрибут пустым или не использовать его вообще, но в таком случае над списком будет пустое пространство.

Кроме того, можно применить любые глобальные атрибуты.

[Как можно изменить форму картинки или HTML элемента?](https://youtu.be/xIGp2FCxqj0?t=163)

Border-radius?

[Чем отличается <article> от <section>?](https://youtu.be/xIGp2FCxqj0?t=232)

аньше почти все разделы верстались на дивах. Но в HTML5 добавили сразу два новых тега для разметки разделов:  
<section> — смысловой или логический раздел документа;  
<article> — самостоятельный и независимый раздел документа.  
Чтобы не было путаницы, разберём где и когда использовать разные контейнеры:  
<div> — контейнер общего назначения, не обязательно смысловой. Дивы используются для разметки мелких блоков, создания сетки и декоративных эффектов.  
<section> — более крупный логический контейнер, объединяющий содержание по смыслу. Например, блок «О компании», список товаров, раздел личной информации в профиле и так далее.  
<article> — самостоятельный, цельный и независимый раздел документа. Этот раздел можно в неизменном виде использовать в различных местах, в том числе и на других сайтах. Примеры: статья, пост в блоге, сообщение на форуме и так далее.

section — «документ или раздел приложения». Обычно содержит верхний (header) и нижний (footer) колонтитулы. Это несколько схожих по смыслу статей, подраздел одной большой статьи, главная часть страницы (например, раздел новостей) или страница интерфейса с закладками. article — «независимая часть документа или сайта».

[Расскажите об особенностях стилизации <svg>?](https://youtu.be/xIGp2FCxqj0?t=298)

https://habr.com/ru/articles/157965/

[Разница между кнопкой и ссылкой в HTML?](https://youtu.be/J6CgOSKFOlw?t=30)

Кнопка (html тег <button> ) — это функциональный элемент, то есть элемент, отвечающий за выполнение определённой функции.

Ссылка (html тег <a> ) — это элемент навигации, то есть элемент, отвечающий за взаимосвязь разделов веб-ресурса (или разных ресурсов) между собой.

Если клик по ней должен открывать окно для ввода регистрационных данных на этой же странице, это кнопка. Если при нажатии должна открываться отдельная страница для входа, это ссылка.

[Для чего используется атрибут decoding?](https://youtu.be/J6CgOSKFOlw?t=150)

Свойство decoding интерфейса [HTMLImageElement](https://developer.mozilla.org/ru/docs/Web/API/HTMLImageElement) предназначено для указания браузеру, как он должен декодировать изображение.

## [Синтаксис](https://developer.mozilla.org/ru/docs/Web/API/HTMLImageElement/decoding" \l "синтаксис)

refStr = imgElem.decoding;

imgElem.decoding = refStr;

### [Значения](https://developer.mozilla.org/ru/docs/Web/API/HTMLImageElement/decoding" \l "значения)

[DOMString](https://developer.mozilla.org/ru/docs/conflicting/Web/JavaScript/Reference/Global_Objects/String_6fa58bba0570d663099f0ae7ae8883ab) предоставляет указания по декодированию. Возможные значения:

* sync: Декодировать изображение синхронно для отображения одновременно с другим содержимым.
* async: Декодировать содержимое асинхронно для уменьшения задержки в отображении другого содержимого.
* auto: Режим по умолчанию, который не отдаёт предпочтения ни одному режиму декодирования, предоставляя браузеру решать, какой режим для пользователя оптимальное.

[Для чего используется атрибут enterkeyhint?](https://youtu.be/J6CgOSKFOlw?t=256)

Атрибут enterkeyhint — это глобальный атрибут, который можно применять к элементам form control или элементам, атрибут contenteditable которых имеет значение true . Этот атрибут помогает пользователям мобильных устройств, пользующимся экранными клавиатурами.

nterkeyhint может принимать семь возможных значений, определяющих, что видит пользователь на своей клавише «enter»:

enter, done, go, next, previous, search, send.

Понятно, как эти «подсказки» могут быть полезными для пользователя. Пользователь выполняет последовательность действий? Отправляет информацию? Сохраняет параметр? В зависимости от его задачи вы можете настраивать подсказку, чтобы она соответствовала потребностям вашего приложения.

[Для чего используют атрибут novalidate?](https://youtu.be/trriSYNrHw4?t=451)

Отменяет встроенную проверку данных введенных пользователем в форме на корректность. Такая проверка осуществляется браузером автоматически при отправке формы на сервер и происходит для полей <input type="email">, <input type="url">, а также при наличии атрибута pattern или required.

### Синтаксис

**<form**novalidate**>**...**</form>**

[Для чего используют атрибут inputmode?](https://youtu.be/trriSYNrHw4?t=512)

Атрибут inputmode говорит браузеру на устройствах с экранной клавиатурой какой набор символов показать при вводе данных в конкретное поле.

[Для чего используется атрибут pattern?](https://youtu.be/xIGp2FCxqj0?t=111)

Атрибут pattern определяет регулярное выражение, по которому проверяются вводимые данные. Атрибут pattern работает со следующими типами элемента <input>: text, search, url, tel, email, password. Атрибут pattern для тега <input> был добавлен в HTML5.

[Для чего используется тэг <label>?](https://youtu.be/3NGkctg4lsE?t=338)

Тег **<label>** устанавливает связь между определенной меткой, в качестве которой обычно выступает текст, и элементом формы (**<input>**, **<select>**, **<textarea>**). Такая связь необходима, чтобы изменять значения элементов формы при нажатии курсором мыши на текст. Кроме того, с помощью **<label>** можно устанавливать горячие клавиши на клавиатуре и переходить на активный элемент подобно ссылкам.

Существует два способа связывания объекта и метки. Первый заключается в использовании идентификатора id внутри элемента формы и указании его имени в качестве атрибута for тега **<label>**. При втором способе элемент формы помещается внутрь контейнера **<label>**.

### Синтаксис

**<input**id="идентификатор"**><label**for="идентификатор"**>**Текст**</label>**

**<label><input**type="..."**>** Текст**</label>**

### Атрибуты

[accesskey](http://htmlbook.ru/html/label/accesskey)

Доступ к элементам формы с помощью горячих клавиш.

[for](http://htmlbook.ru/html/label/for)

Идентификатор элемента, с которым следует установить связь.

### Закрывающий тег

Обязателен.

Пример

HTML5IECrOpSaFx

<!DOCTYPE HTML>

**<html>**

**<head>**

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

**<title>**Тег LABEL**</title>**

**</head>**

**<body>**

**<form**action="handler.php"**>**

**<p><b>**Lorem ipsum dolor sit amet...**</b></p>**

**<p><input**type="checkbox"id="check1"**><label**for="check1"**>**Lorem**</label><Br>**

**<input**type="checkbox"id="check2"**><label**for="check2"**>**Ipsum**</label><Br>**

**<input**type="checkbox"id="check3"**><label**for="check3"**>**Dolor**</label><Br>**

**<input**type="checkbox"id="check4"**><label**for="check4"**>**Sit amet**</label></p>**

**</form>**

**</body>**

**</html>**

HTML тег <label> (в переводе с англ. - метка) определяет текстовую метку для элемента <input>. По своему виду метка ничем не отличается от обычного текста, однако она позволяет пользователю выбрать элемент формы простым кликом по тексту, расположенному внутри элемента <label>, а не по самому элементу <input>. Чтобы определить к какому элементу формы относится текущая метка, необходимо использовать атрибут for тега <label>. Значение атрибута for должно быть эквивалентно значению атрибута id того элемента формы, к которому будет относится метка.

Атрибутом "for" можно не пользоваться, если элемент <input> будет находиться внутри элемента <label>.

### **Как следует оформлять страницу, содержимое которой может быть на разных языках?[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "как-следует-оформлять-страницу-содержимое-которой-может-быть-на-разных-языках)**

Вопрос немного расплывчатый. Полагаю, что речь о наиболее частом случае: как показывать страницу, где содержимое доступно на нескольких языках, но отображается на одном определенном.

Когда к серверу делается HTTP-запрос, то браузер пользователя обычно отсылает информацию о предпочитаемом языке в заголовке Accept-Language. Сервер может использовать эту информацию, чтобы вернуть версию документа на подходящем языке, если такая возможность есть. В возвращённом HTML-документе обязательно должен быть указан атрибут lang у тега <html>, к примеру <html lang="en">...</html>.

На бэкенде HTML-разметка будет содержать плейсхолдер i18n, а контент для конкретного языка будет храниться в YAML- или JSON-формате. Сервер динамически формирует HTML-страницу с контентом на конкретном языке, чаще всего при помощи бэкенд-фреймворка.

###### Ссылки[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "ссылки-1)

* <https://www.w3.org/International/getting-started/language>

### **На что необходимо обратить внимание при разработке мультиязычных сайтов?[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "на-что-необходимо-обратить-внимание-при-разработке-мультиязычных-сайтов)**

* Используйте атрибут lang в HTML.
* Перенаправляйте пользователей на версию сайта на их языке. Позволяйте быстро и без проблем изменить страну и язык.
* Текст на картинках плохо поддается адаптации. Многие до сих пор помещают текст на картинки чтобы получить хорошо выглядящий несистемный шрифт на любом компьютере. Однако чтобы перевести текст картинкой, нужно иметь подготовленную картинку с каждой строкой текста для каждого языка. При большом количестве текста это быстро выйдет из-под контроля.
* Ограничение длины слов и предложений. Некоторый текст может быть длиннее при написании на другом языке. Будьте внимательны к макету и проверяйте поведение блоков при переполнении. Количество символов важно в таких элементах, как заголовки, лейблы и кнопки. Но не так важно в основном тексте или в блоке комментария.
* Помните о восприятии цветов. В разных языках и культурах цвета имеют разное значение. Дизайн должен учитывать эти особенности.
* Форматируйте даты и валюты. Календарные даты иногда пишутся по-разному. Например, «Май 31, 2012» в США или «31 мая 2012» в странах Европы.
* Не склеивайте переведённые строки. Не пишите что-то вроде "Сегодняшняя дата " + date. Эта фраза будет выглядеть некорректно на языках с другим порядком слов. Вместо этого используйте шаблонную строку с подстановкой параметров для каждого языка. Например, посмотрите на следующие два предложения на русском и китайском соответственно: Я буду путешествовать {% date %} и {% date %} 我会出发. Обратите внимание, что положение переменной отличается из-за грамматических правил языка.
* Разные направления чтения. В русском мы читаем слева направо, сверху вниз. В традиционном японском языке текст читается сверху вниз, справа налево.

###### Ссылки[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "ссылки-2)

* <https://www.quora.com/What-kind-of-things-one-should-be-wary-of-when-designing-or-developing-for-multilingual-sites>

### **Объясните разницу между cookie, sessionStorage и localStorage.[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "объясните-разницу-между-cookie-sessionstorage-и-localstorage)**

Все вышеупомянутые технологии являются механизмами хранения типа ключ-значение на клиентской стороне. Они могут хранить данные только как строки.

|  | cookie | localStorage | sessionStorage |
| --- | --- | --- | --- |
| Инициатор | Клиент или сервер. Сервер может использовать заголовок Set-Cookie | Клиент | Клиент |
| Срок хранения | Устанавливается вручную | Всегда | До закрытия вкладки |
| Хранение между сессиями | Зависит от установки срока хранения | Да | Нет |
| Отправка на сервер с каждым HTTP-запросом | автоматически, с помощью заголовка Cookie | Нет | Нет |
| Емкость (на один домен) | 4 КБ | 5 МБ | 5 МБ |
| Доступность | В любом окне | В любом окне | В той же вкладке |

###### Ссылки[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "ссылки-5)

* <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies>
* <http://tutorial.techaltum.com/local-and-session-storage.html>

### **Что такое прогрессивный рендеринг[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "что-такое-прогрессивный-рендеринг)**

Прогрессивный рендеринг — это название технологий, используемых для ускорения отрисовки страниц (в частности, для уменьшения времени загрузки), чтобы показать пользователю контент как можно скорее.

До того, как широкополосный интернет распространился повсеместно, прогрессивный рендеринг встречался довольно часто. Но этот подход по прежнему полезен в современной разработке, поскольку все более популярным (и ненадежным) становится мобильный доступ в интернет.

Примеры подобных технологий:

* Ленивая загрузка изображений. Изображения на странице не загружаются все разом. JavaScript подгрузит изображения тогда, когда пользователь доскроллит до той части страницы, на которой они расположены.
* Приоритизация видимого контента. Только минимум CSS, контента, скриптов, необходимых для отрисовки той части страницы, которую пользователь увидит первой. Вы можете использовать отложенные скрипты или слушать события DOMContentLoaded или load, чтобы загрузить остальные ресурсы и контент.
* Асинхронные фрагменты HTML. Отправка в браузер частей HTML-страницы, созданной на бэкенде. Более подробно про эту технологию можно почитать [в этой статье](http://www.ebaytechblog.com/2014/12/08/async-fragments-rediscovering-progressive-html-rendering-with-marko/).

###### Ссылки[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "ссылки-8)

* <https://stackoverflow.com/questions/33651166/what-is-progressive-rendering>
* <http://www.ebaytechblog.com/2014/12/08/async-fragments-rediscovering-progressive-html-rendering-with-marko/>

### **Для чего используется атрибут srcset в теге изображения? Опишите процесс, который браузер использует при обработке содержимого этого атрибута.[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "для-чего-используется-атрибут-srcset-в-теге-изображения-опишите-процесс-который-браузер-использует-при-обработке-содержимого-этого-атрибута)**

Вы должны использовать атрибут srcset, когда хотите показывать пользователям разные изображения в зависимости от их ширины дисплея их устройств - предоставление изображений более высокого качества устройствам с retina-дисплеями улучшает работу пользователя, в то время как предоставление изображений с низким разрешением на устройства с низким разрешением повышает производительность и уменьшает объём передаваемых данных (поскольку нет видимой разницы с большими изображениями). Например: <img srcset="small.jpg 500w, medium.jpg 1000w, large.jpg 2000w" src="..." alt=""> указывает браузеру отображать малое, среднее или большое изображение в зависимости от разрешения дисплея пользователя. Первое значение - это имя изображения, а второе - ширина изображения в пикселях. Для ширины устройства 320px выполняются следующие расчеты:

* 500 / 320 = 1.5625
* 1000 / 320 = 3.125
* 2000 / 320 = 6.25

Если у клиента стандартное разрешение 1x, то 1.5625 является ближайшим коэффициентом, и 500w соответствующее small.jpg будет отображено в браузере.

Если же разрешение retina (2x), браузер будет использовать ближайшее разрешение выше минимального. Это означает, что он не выберет 500w (1.5625), потому что оно больше 1 и изображение может выглядеть плохо. Браузер затем выберет изображение с результирующим соотношением ближе к 2, то есть 1000w (3.125).

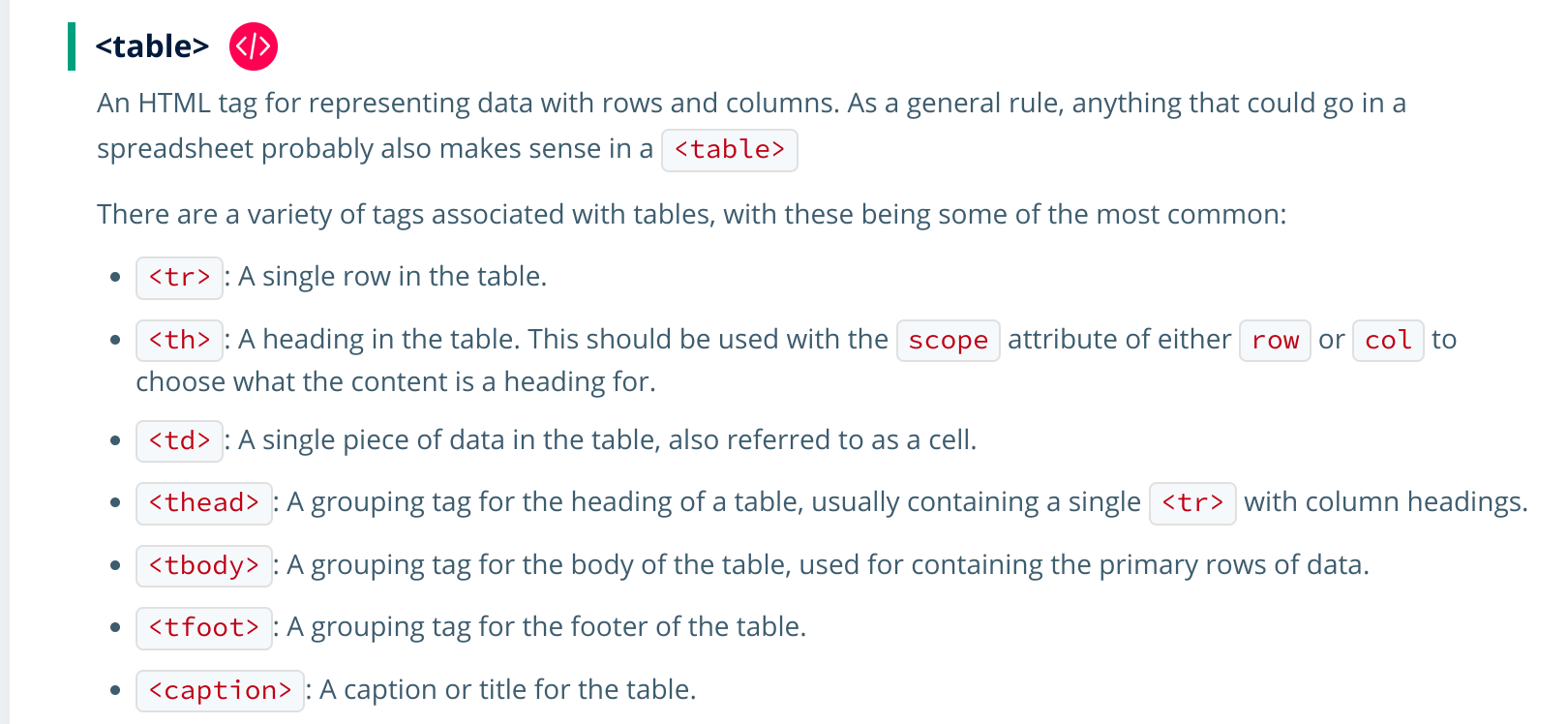
srcset решает проблему, когда вы хотите показывать файлы изображений меньшего размера для устройств с узким экраном, поскольку они не нуждаются в больших изображениях, как на настольных дисплеях, - а также, при желании, вы можете показывать изображения с различным разрешением для экранов с высокой/низкой плотностью пикселей.

###### Ссылки[​](https://www.frontendinterviewhandbook.com/ru/html-questions/" \l "ссылки-9)

* <https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Responsive_images>
* <https://css-tricks.com/responsive-images-youre-just-changing-resolutions-use-srcset/>

**PRE** - The preformatted text tag. This tag preserves whitespace, which can be useful when indentation and

newlines need to be preserved.

[DOM](https://www.w3schools.com/whatis/whatis_htmldom.asp)

