# Конспект «Основы HTML и CSS»

## HTML

HTML расшифровывается как «Hypertext Markup Language», то есть «язык гипертекстовой разметки».

Язык HTML отвечает за структуру и содержание страницы. HTML состоит из тегов, а тег состоит из имени, заключённого между знаками «меньше» и «больше». Примеры тегов: <h1>, <p>, <ul>.

### Парные теги

Теги бывают парными и одиночными. Парные теги могут содержать текст и другие теги. У парных тегов, в отличие от одиночных, есть вторая половинка — закрывающий тег:

<h1>Текст заголовка</h1>

В закрывающей части парных тегов перед именем ставится символ / («слэш»).

В парные теги можно вкладывать другие теги. Например, как в списках:

<ul>

<li>Элемент списка</li>

</ul>

У вложенных тегов всегда нужно следить за правильным порядком закрытия. Вложенный тег не может закрываться позже родительского:

<ul><li>Элемент списка</ul></li> <!-- Плохо -->

<ul><li>Элемент списка</li></ul> <!-- Хорошо -->

Не все теги можно вкладывать в другие теги, например, тег <h1> нельзя вкладывать в <p>.

### Одиночные теги

Существуют не только парные, но и одиночные теги. Например, тег <img> позволяет добавить картинку в разметку.

Сам по себе <img> не имеет смысла. Чтобы этот тег был действительно полезен, необходимо написать внутри него адрес, ведущий к картинке. Делается это с помощью атрибута src:

<img src="keks.png">

У тега может быть несколько атрибутов. В этом случае они пишутся через пробел:

<тег атрибут1="значение1" атрибут2="значение2">

Например, картинке при желании можно задать размеры:

<img src="keks.png" width="200" height="100">

### Комментарии

Код, заключённый между символами <!-- и -->, работать не будет. Если эти символы удалить, то код заработает, это называется «раскомментировать». С помощью комментариев обычно временно отключают какой-то код или оставляют подсказки и разъяснения.

<!-- Это комментарий в HTML -->

## CSS

CSS расшифровывается как «Cascading Style Sheets», то есть «каскадные таблицы стилей».

Язык CSS отвечает за внешний вид страницы.

С помощью CSS можно задавать параметры для любого тега: ширину и высоту, отступы, цвет и размер шрифта, фон и так далее. Все эти параметры задаются с помощью свойств в следующем формате:

свойство: значение;

Например:

color: red;

padding: 10px;

Стили к тегам добавляются чаще всего при помощи атрибута class.

Например, если мы хотим, чтобы определённые стили, описанные, допустим, в классе feature-kitten, применились к тегу <p>, то в разметке напишем так:

<p class="feature-kitten">...</p>

### CSS-правила

Можно сказать, что CSS-правило — это группа свойств и их значений, которая целиком применяется к тем тегам, на которые указывает селектор.

И выглядит это так:

селектор {

свойство1: значение;

свойство2: значение;

}

Задавать стили можно не только с помощью атрибута class, но и по тегам. Селектор указывает, к каким тегам применятся свойства из CSS-правила. Селекторы по тегам работают проще всего: они выбирают все теги с подходящим именем.

p { color: red; }

В примере селектором является p, и он выбирает все теги с именем p (то есть теги <p>), а теги с другим именем, например h1, не выбирает.

Когда же стилизация задаётся по классам, то стили применяются только к тегам с такими классами.

.название\_класса {

свойство: значение;

}

### Миксование классов

У HTML-элемента может быть сколько угодно классов, в этом случае они перечисляются в атрибуте class через пробел, например:

<li class="product">Товар</li>

<li class="product hit">Товар, а ещё хит продаж</li>

<li class="product hit sale">Товар, хит продаж и со ски-и-идкой!</li>

Обычно миксование используют так: в один класс выносят общее оформление, а в дополнительных классах описывают его модификации.

### Комментарии

В CSS тоже существуют комментарии, их отличие от HTML-комментариев в том, что код или подсказки пишутся между символами /\* и \*/.

**Конспект «Структура HTML-документа»**

Каждый HTML-документ начинается с декларации типа документа или «доктайпа». Тип документа необходим, чтобы браузер мог определить версию HTML и правильно отобразить страницу.

<!DOCTYPE html>

Простейшая HTML-страница состоит как минимум из трёх тегов: <html>, <head> и <body>. Тег <head> обычно содержит заголовок, ключевые слова, описание страницы и другие служебные данные. Также внутри него подключаются внешние ресурсы, например, стили. Содержимое этого тега не отображается на странице напрямую. А в теге <body> хранится содержание страницы, которое отображается в окне браузера.

Для подключения стилей к странице существует тег <link>. Для этого у него есть атрибут href в котором задаётся адрес стилевого файла, а значение stylesheet атрибута rel говорит браузеру, что мы подключаем именно стили, а не что-то другое.

<head>

<link href="адрес\_файла\_стилей.css" rel="stylesheet">

</head>

Ещё один элемент, который располагается в <head> — это тег <title>. В нём задаётся заголовок страницы, который отображается во вкладках браузера. По заголовку должно быть понятно, о чём эта страница, даже когда она не открыта в браузере, а отображается в результатах поиска или в браузерных закладках.

<head>

<title>Тренажёры — HTML Academy</title>

</head>

Ещё один важный тег, располагающийся внутри <head> это тег <meta>. Он одиночный, то есть не требует парный закрывающий тег в конце. С помощью <meta> можно сообщать браузеру, поисковому роботу или другому устройству различную служебную информацию (или метаинформацию) о вашем сайте: кодировку текста, описание контента и так далее. Для этого используются теги <meta> с разными атрибутами и их значениями.

Кодировка текста HTML-страницы указывается с помощью атрибута charset:

<meta charset="название кодировки">

Самая распространённая современная кодировка — utf-8.

Перечень ключевых слов задаётся тегом <meta>, у которого атрибут name имеет значение keywords. Ключевые слова (самые важные слова из содержания страницы) перечисляются в атрибуте content через запятую:

<meta name="keywords" content="важные, ключевые, слова">

Краткое описание (или аннотация) страницы задаётся похожим образом, только значение атрибута name меняется на description:

<meta name="description" content="краткое описание">

Внутри <body> находятся те теги, которые отображаются на странице. Например, тег <main> выделяет основное содержание страницы, которое не повторяется на других страницах. И обычно на странице используется один <main>.

Тег <header> содержит вводную часть страницы, которую чаще называют «шапкой», а тег <footer> описывает заключительную часть страницы, или «подвал». Существует тег <section>, который обозначает крупный смысловой (или «логический») раздел.

Тег <article>, обозначает цельный, законченный и самостоятельный фрагмент информации.

Для создания логического раздела с основной навигацией предназначен тег <nav> (сокращение от английского «navigation»). Обычно в <nav> включают ссылки на другие страницы или навигацию по текущей странице.

Тег <aside> включает в себя дополнительное содержание, не связанное напрямую с основным. Такие блоки ещё часто называют «сайдбарами» или боковыми панелями.

Для создания основной структуры текста используют заголовки. В HTML существует целое семейство заголовочных тегов: от <h1> до <h6>. Тег <h1> обозначает самый важный заголовок (заголовок верхнего уровня), а тег <h6> обозначает подзаголовок самого нижнего уровня.

<h1>Спецификация HTML</h1>

<h2>Раздел 1 Введение</h2>

<h3>Раздел 1.1 Происхождение языка</h3>

Основную структуру текста создают с помощью заголовков, а более мелкую выстраивают с помощью параграфов (или абзацев). Для разметки параграфов предназначен тег <p>. По умолчанию абзацы начинаются с новой строки и отделяются от остального контента отступами сверху и снизу.

# Конспект «Разметка текста»

## Списки

### Неупорядоченный список

Тег <ul> (сокращение от «unordered list»). Используется, когда порядок элементов не важен. Например, для разметки перечня ссылок в меню, преимуществ товара, ингредиентов в составе продукта.

Непосредственно в теге <ul> могут находиться только теги <li> (сокращение от «list item»), которые обозначают элементы или пункты списка:

<ul>

<li>Я пункт списка, могу быть на любом месте</li>

<li>И я пункт списка, и мне тоже не важен порядок</li>

</ul>

По умолчанию элементы <ul> отмечаются маркерами такого же цвета, как цвет текста.

* Я пункт списка, могу быть на любом месте
* И я пункт списка, и мне тоже не важен порядок

### Упорядоченный список

Тег <ol> (сокращение от «ordered list»). В этом списке действительно важно, в каком порядке идут элементы. Упорядоченные списки подходят для разметки алгоритмов, инструкций, рецептов, результатов соревнований и так далее.

Пункты упорядоченного списка тоже размечаются с помощью тега <li>. Пример кода:

<ol>

<li>Я первый и только первый пункт</li>

<li>Я не я, если я не второй пункт</li>

<li>Третий после стольких лет? Всегда!</li>

</ol>

По умолчанию перед элементами <ol> ставится их порядковый номер.

1. Я первый и только первый пункт
2. Я не я, если я не второй пункт
3. Третий после стольких лет? Всегда!

У <ol> может быть несколько атрибутов: start, reversed и type.

Атрибут start меняет стартовое число нумерации пунктов. Может быть отрицательным.

Атрибут reversed меняет направление нумерации на противоположный. Этот атрибут не требует значения.

С помощью атрибута type можно задавать различные типы маркеров: строчные и заглавные латинские буквы или римские цифры.

### Список описаний

Тег <dl> (сокращение от «description list»). Список описаний используется для разметки вопросов-ответов, наименований и определений, категорий и тем. Он создаётся с помощью трёх тегов:

* <dl> обозначает сам список описаний;
* <dt> (сокращение от «description term») обозначает термин;
* <dd> (сокращение от «description definition») обозначает описание или определение.

Теги <dt> и <dd> пишутся внутри <dl>. Каждый список <dl> может содержать один или несколько терминов и одно или несколько описаний для каждого термина. Пример кода:

<dl>

<dt>HTML</dt>

<dd>Язык гипертекстовой разметки</dd>

<dt>CSS</dt>

<dd>Каскадные таблицы стилей</dd>

<dd>Язык для оформления HTML-документов</dd>

</dl>

По умолчанию браузер добавляет небольшой отступ слева от определений.

HTML

Язык гипертекстовой разметки

CSS

Каскадные таблицы стилей

Язык для оформления HTML-документов

## Преформатированный текст и код

Тег <pre> (сокращение от «preformatted text»). Используется для отображения примеров кода, также применяется для отображения картинок ASCII Art. Браузер сохраняет и отображает все пробелы и переносы, которые есть внутри тега <pre>.

<pre>Пример

преформатированного

текста с сохранёнными пробелами

и переносами строк</pre>

Пример

преформатированного

текста с сохранёнными пробелами

и переносами строк

Тег <code>. Используется для обозначения фрагментов кода.

С его помощью размечается любой фрагмент текста, который распознается компьютером: код программы, разметки, название файла и так далее. Обычно браузеры отображают текст в теге <code> моноширинным шрифтом.

Тег <code>ul</code> — это неупорядоченный список.

Тег ul — это неупорядоченный список.

Тег <code> можно вкладывать внутрь тега <pre>.

## Цитаты

### Небольшие цитаты

Тег <q> (сокращение от «quote»). Предназначен для выделения цитат внутри предложения. Текст внутри тега браузер автоматически обрамляет кавычками, поэтому добавлять кавычки вручную не нужно.

### Источник цитат

Тег <cite>. В нём можно указывать помимо адреса источника цитаты ещё и название произведения, откуда цитируется текст, а также имя автора или организации, чей текст цитируется. Содержимое <cite> в браузере выделяется курсивом.

<p>По словам <cite>Чарльза Буковски</cite> — <q>Интеллектуал о простой вещи говорит сложно — художник сложную вещь описывает простыми словами.</q></p>

По словам Чарльза Буковски — Интеллектуал о простой вещи говорит сложно — художник сложную вещь описывает простыми словами.

Тег <cite> может быть самостоятельным и не привязываться к цитате:

<p>Какой доктор ваш любимый (в сериале <cite>Доктор Кто</cite>)?</p>

### Длинные цитаты

Тег <blockquote>. Предназначен для выделения длинных цитат, которые могут состоять из нескольких абзацев. Тег выделяет цитату не как фрагмент текста в предложении, а как отдельный блок текста с отступами.

<blockquote>

<p>Ум ценится дорого, когда дешевеет сила.</p>

<cite>Джейсон Стэтхэм</cite>

</blockquote>

В браузере контенту тега <blockquote> обычно добавляется дополнительный отступ слева и справа.

Обычный текст.

Ум ценится дорого, когда дешевеет сила.

Джейсон Стэтхэм

## Разметка фрагментов текста

### Символы-мнемоники

Это особые строки, которые начинаются с амперсанда (&) и заканчиваются точкой с запятой (;). Например, знак *меньше* на страницу можно вставить мнемоникой &lt; (less than), а знак *больше* мнемоникой &gt; (greater than):

Некоторые символы в HTML зарезервированы, то есть браузер считает их HTML-кодом. Например, любой текст после знака *меньше* (<) браузер будет пытаться интерпретировать как тег и на странице не отобразит. Чтобы использовать специальные символы в тексте страницы как обычные символы их нужно заменить на символы-мнемоники.

&lt;**ul**&gt;

&lt;/**ul**&gt;

<ul>  
</ul>

### Перенос строк

Тег <br> (сокращение от «line break»). Применяется, чтобы вставить в текст перенос строки, не создавая при этом абзац. Например, при разметке стихов или текстов песен.

### Верхний и нижний индексы

Теги <sup> и <sub>. Названия образованы от слов «superscript» и «subscript».

Тег <sup> отображает текст в виде верхнего индекса, а <sub> отображает текст в виде нижнего индекса.

Их используют для указания единиц измерения или для написания простых формул:

20м<sup>2</sup>

H<sub>2</sub>O

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

20м2  
H2O  
X3+X2=1

Для создания более сложных формул, эти теги можно использовать внутри друг друга.

### Дата и время

Тег <time>. С помощью него можно описывать даты одновременно и для человека, и для машины. Для указания даты в машиночитаемом формате ISO 8601 существует атрибут datetime и выглядит так:

<time datetime="2016-11-18T09:54">09:54 утра</time>

<time datetime="2015-11-18">18 ноября 2015</time>

<time datetime="2018-09-23">в прошлую субботу</time>

<time datetime="2017-04-20">вчера</time>

Браузер отображает только содержимое тега, а содержимое datetime не отображается.

### Акцентирование внимания

Теги <em> и <i>. Названия образованы от слов «emphasis» и «italic». Предназначены для акцентирования внимания на слово или фразу. Визуально оба тега одинаковы, они выделяют текст курсивом.

Тег <em> определяет текст, на который сделан *особый акцент*, меняющий смысл предложения.

Я <em>просто обожаю</em> холодные зимние дни!

Тег <i> применяется для обозначения текста, который отличается от окружающего текста, но не является более важным. Например, в <i> можно заключать *названия*, *термины*, *иностранные слова*. Также в этот тег можно обернуть *мысли* героя. В речи такой текст обычно выделяется интонационно:

Он взглянул в окно и подумал — <i>такого просто не может быть</i>!

### Выделение и придание важности

Теги <strong> и <b>. Название <b> образовано от слова «bold». Отображаются оба тега одинаково, они выделяют текст жирным.

Тег <strong> указывает на **важность** отмеченного текста. Он может использоваться для выделения предупреждений или части документа, которую пользователь должен увидеть раньше остального. При этом обозначение части текста тегом <strong> не должно изменять смысла предложения.

<strong>Внимание!</strong> Это место опасно. <strong>Вы можете упасть в пропасть</strong>, если подойдёте близко к краю.

Тег <b> предназначен для выделения текста с целью привлечения к нему внимания, но без придания ему особой важности. Использовать его нужно только в случае, когда остальные теги выделения не подходят. Типичный пример — выделение вводного предложения статьи.

Вы входите в небольшую комнату. Ваш <b>меч</b> загорается ярче. <b>Крыса</b> стремительно пробегает вдоль стены.

### Описание изменений

Теги <del> и <ins>. Названия тегов образованы от слов «delete» и «insert». Предназначены для описания изменений в документе.

Тег <del> выделяет текст, который был удалён в новой версии документа. В браузере этот текст перечёркивается.

Тег <ins> выделяет текст, который был добавлен в новой версии документа. В браузере этот текст подчёркивается.

<ul>

<li>Почистить посудомоечную машину</li>

<li><del datetime="2009-10-11T01:25-07:00">Погулять</del></li>

<li><del datetime="2009-10-10T23:38-07:00">Поспать</del></li>

<li><ins>Купить принтер</ins></li>

</ul>

* Почистить посудомоечную машину
* Купить принтер

## Разделение контента

Теги <div> и <span>. Это «чистые» элементы, и обычно они отлично подходят в качестве обёртки для стилизации или группировки других элементов. Использовать эти теги рекомендуется в тех случаях, если более подходящих семантических тегов не нашлось.

Тег <div> используется для группировки структурных элементов или в качестве вспомогательных контейнеров для создания нужной раскладки.

Тег <span> используется для группировки текстовых элементов, выделения отдельных слов или фраз внутри абзацев, пунктов списка и так далее.

<article>

<div class="highlight">

<p>…</p>

<p>…</p>

</div>

<p>Текст, в котором <span>выделена фраза</span></p>

</article>

# Конспект «Ссылки и изображения»

## Ссылки

### Что такое ссылка

Типичная ссылка представляет собой участок текста, щёлкая на который, вы переходите на другую страницу, открываете изображение, начинаете скачивать файл или перемещаетесь к какому-то месту на текущей странице.

Ссылки создаются с помощью тега <a>. Например:

<a href="https://htmlacademy.ru">HTML Academy</a>

Тег <a> можно использовать вообще без адреса, то есть без атрибута href. Такой тег обозначает «ссылку-заглушку», которая в других условиях может стать обычной ссылкой. Часто ссылки-заглушки используют, чтобы показать, что мы находимся на текущей странице:

<ul>

<li><a>1 страница</a></li>

<li><a href="2">2 страница</a></li>

<li><a href="3">3 страница</a></li>

</ul>

Когда мы удаляем атрибут href у ссылки, то лучше оставить подсказку о том, почему мы это сделали. Подсказку можно добавить с помощью атрибута title. Подсказка появится, когда курсор задержится над ссылкой некоторое время.

* 1 страница
* [2 страница](https://htmlacademy.ru/courses/305/run/16)
* [3 страница](https://htmlacademy.ru/courses/305/run/16)

### Абсолютные ссылки

Если нужно сделать ссылку на другой сайт в интернете, то необходимо использовать «обычный» адрес. Этот «обычный» или полный адрес называется абсолютным. Выглядит он, например, так:

https://site.ru/blog/index.html

Абсолютные адреса содержат минимум три части: протокол, имя сервера и путь.

### Относительные ссылки

Когда файл по ссылке должен открываться локально на компьютере, используются относительные адреса. В отличие от «обычных» адресов, в нём нет адреса сайта. Например:

day-2.html

Относительные адреса работают не только для файлов на компьютере, но и для страниц в сети. Если выложить два каких-то файла в интернет (не меняя их взаимное расположение), то их ссылка друг на друга всё равно будет работать.

Если в адресе нет имени сервера или протокола, то это относительный адрес.

### Ссылки на файл

По ссылкам можно не только переходить, но и скачивать файлы. Для этого необходимо просто в атрибуте href прописать ссылку на этот файл. А для того чтобы предотвратить открытие файлов прямо в браузере, у тега <a> существует атрибут download.

<a href="file.pdf" download>Браузер скачает меня, а не будет читать</a>

### Ссылки-якоря

Ссылка-якорь — это обычная ссылка, в адресе которой используется символ #, после которого следует идентификатор элемента. Идентификатор создаётся с помощью атрибута id у того тега, к которому надо перейти при щелчке по ссылке.

<a href="#part1">Глава 1</a>

Ссылка-якорь используется для прокрутки к заданной части страницы, в том числе используется и в абсолютных адресах.

## Изображения

### Подключение изображений

Для подключения изображений существует тег <img>, для него не требуется закрывающего тега, а путь к картинке задаётся в атрибуте src. Например:

<img src="logo.png">

### Форматы изображений

Существует несколько основных форматов изображений: JPEG, PNG, SVG и GIF.

Формат SVG переводится как масштабируемая векторная графика. Качество таких изображений не меняется при изменении размеров, да и вес у них небольшой. Отлично подходит для малоцветных схем, логотипов и иконок. Чаще всего используется в случаях, когда необходимо масштабировать изображение без потерь, изменять цвет элементов изображения, анимировать части изображения.

Формат JPEG подходит для фотографий, рисунков с большим количеством разноцветных деталей, изображений с плавным переходом яркости и контраста. При сжатии изображения ухудшается его качество.

Формат PNG позволяет сохранять изображения, в которых требуется особенная чёткость. Главная особенность этого формата — поддержка прозрачности. Подходит для изображений с прозрачностью и полупрозрачностью, когда необходима повышенная точность полноцветных изображений и для изображений с резкими переходами цветов.

Формат GIF используется для простейших анимаций. В последнее время GIF-изображения становятся всё менее используемыми и заменяются на другие, более оптимальные форматы.

### Размеры изображения

Для управления шириной или высотой изображения, используются атрибуты width и height. Размеры в этих атрибутах задаются без единиц измерения.

<img src="logo.png" width="200" height="100">

Если задать только один из размеров, ширину или высоту, то вторую размерность браузер вычислит самостоятельно исходя из пропорций изображения. Если задать картинке одновременно и высоту, и ширину, то браузер может нарушить пропорции исходного изображения.

### Атрибут alt

В случаях, когда картинка не может отобразиться, для того, чтобы было понятно, что на ней изображено, существует альтернативный текст. Также альтернативный текст помогает сайтам оставаться доступными, например, для категории пользователей, которая не имеет возможности видеть картинки.

Альтернативный текст изображения задаётся с помощью атрибута alt. Например:

<img src="cat.png" alt="Кот, который гуляет сам по себе">

### Figure и figcaption

Тег <figure> подходит для любого иллюстративного или демонстрационного материала, которым можно дополнить содержание документа: схемы, графики, примеры кода, таблицы и так далее. При удалении такого материала основное содержание не должно пострадать, иначе это не дополнительный материал и тег <figure> не подходит для его разметки.

Для разъясняющего комментария к такому иллюстративному материалу существует тег <figcaption>, который размещается первым или последним элементом внутри <figure>. Например:

<figure>

<img src="image.jpg" alt="Альтернативный текст">

<figcaption>Подпись к содержимому</figcaption>

</figure>

Подпись к содержимому

### Изображение-ссылка

Ссылки можно делать не только с помощью текста, но и с помощью изображений. Для этого нужно обернуть тег <img> в тег <a>. Например:

<a href="http://keksby.ru">

<img src="cat.png" alt="Кекс">

</a>

# Конспект: основы CSS

## CSS-правила

CSS — это язык для оформления структурированных документов, например, HTML- документов. Синтаксис — это плоский список CSS-правил. CSS-правило состоит из селектора и перечня свойств и их значений:

селектор {

свойство: значение;

свойство: значение;

}

Для комментариев в CSS используются символы /\* и \*/.

## Селекторы

Селектор находится в начале CSS-правила, до фигурных скобок, и определяет, к каким HTML-элементам применятся свойства и значения из правила.

.feature-kitten {

padding-top: 60px;

}

Простейшие (и самые популярные) селекторы — это селекторы по тегам и по классам. Селекторы по тегам содержат имя тега без символов < и > и применяются ко всем подходящим тегам. Селекторы по классам начинаются с точки, за которой идёт имя класса, и применяются ко всем тегам с подходящим атрибутом class.

h1 { color: red; }

.info { color: blue; }

На странице может быть несколько списков, и стили применятся ко всем спискам, даже к тем, которые вы менять не хотели. Чтобы избежать таких ситуаций, лучше не использовать селекторы по тегам или использовать их как можно реже.

Если у CSS-правил отличаются только селекторы, а свойства и значения одинаковые, то их можно сгруппировать через запятую.

Также можно комбинировать любые типы селекторов через пробел. Такие селекторы называются вложенными или контекстными и читаются справа налево. Например:

nav a {…}

.menu ul {…}

.post .title {…}

## Свойства и значения

Список свойств и значений находится внутри фигурных скобок CSS-правила. Свойство определяет, какую характеристику внешнего вида мы хотим изменить, а значение — как именно.

.feature-kitten {

padding-top: 60px;

}

Каждый раз, когда мы добавляем новое свойство или изменяем его значение, мы меняем что-то на странице.

## Наследование

Наследование в CSS — это механизм, с помощью которого значения свойств элемента-родителя передаются его элементам-потомкам. Стили, присвоенные одному элементу, наследуются всеми потомками (вложенными элементами), но только в том случае, если они где-то явно не переопределены.

## Составные свойства

В CSS есть обычные свойства, управляющие одним параметром отображения, и есть составные свойства, управляющие одновременно несколькими параметрами. Например, свойство font. Оно задаёт сразу шесть параметров: размер и название шрифта, высоту строки и некоторые другие.

font: 16px/26px "Arial", sans-serif;

Если значение обычного свойства не было задано в составном, то браузер при «расшифровке» использует исходное значение этого свойства.

## Типы значений: абсолютные и относительные

Абсолютные единицы измерения привязаны к настоящим физическим размерам и связаны между собой жёсткими пропорциями. Пиксели, px, используют чаще всего, остальные абсолютные единицы почти не применяют. Примеры абсолютных единиц измерения:

font-size: 1cm;

font-size: 10mm;

font-size: 38px;

Относительные единицы измерения описывают значения, которые зависят от других значений. Например, ширина элемента в процентах зависит от ширины родительского элемента, а ширина элемента в em зависит от размера шрифта самого элемента. К относительным единицам относятся em, rem, vh, vw и некоторые другие, ну и, конечно же, проценты.

## Стили по умолчанию

Некоторым элементам можно не задавать никаких стилей, но у них всё равно будет какое-то оформление. Например, у списка <ul> есть отступы и маркеры. Такие стили называются стилями по умолчанию и задаются внутри браузерных стилей изначально. Их можно переопределить или сбросить, задав другие значения свойств элементу.

## Каскадирование

Когда браузер отрисовывает страницу, он должен определить итоговый вид каждого HTML-элемента. Для этого он собирает все CSS-правила, которые относятся к каждому элементу, ведь на элемент могут влиять сразу несколько CSS-правил. Механизм комбинирования стилей из разных источников в итоговый набор свойств и значений для каждого тега называется каскадностью. Например, есть такой элемент в разметке:

<p class="beloved-color">Зелёный - мой любимый цвет</p>

Заданные стили:

.beloved-color { color: green; }

Браузерные стили:

margin: 1em 0;

Итоговые стили:

color: green;

margin: 1em 0;

## Конфликт свойств

На один элемент могут действовать несколько CSS-правил. Если в этих правилах есть одинаковые свойства с разными значениями, то возникает конфликт. Например:

ul { list-style: disc; }

.blog-navigation ul { list-style: none; }

Браузеру нужно как-то решать, какими будут итоговые значения конфликтующих свойств. Конфликт разрешается максимум за три шага. Если на текущем шаге определиться не удалось, то выполняется следующий шаг. Вот эти шаги:

1. Сравниваются приоритеты стилевых файлов, в которых находятся конфликтующие свойства. Например, авторские (то есть наши) стили приоритетнее браузерных.
2. Сравнивается специфичность селекторов у CSS-правил с конфликтующими свойствами. Например, селектор по классу более специфичен, чем селектор по тегу.
3. Побеждает то свойство, которое находится ниже в коде.

Каскад работает и внутри CSS-правил.

## Встраивание и подключение внешних стилей

Внешние стили подключаются через тег <link>

<link rel="stylesheet" href="style.css">

Встраивание стилей в тег <style>. Его обычно размещают внутри <head>:

<head>

<style>

CSS-код

</style>

</head>

Такой способ используется для оптимизации загрузки страницы, ведь в таком случае браузер не будет отправлять дополнительных запросов на сервер.

Встраивание в атрибут style:

<div style="width: 50%;"></div>

Свойства и значения, прописанные таким образом, применятся точечно к одному элементу.

Обычно использование этого способа считается плохой практикой. Но иногда в виде исключения бывает удобнее воспользоваться встраиванием стилей в атрибут style, чем писать отдельные CSS-правила. Например, когда нужно управлять стилями именно из разметки, и создавать отдельные классы при этом будет излишне. Так бывает, когда какие-то стилевые параметры устанавливаются с помощью сторонних программ или другими людьми, например, через CMS.

# Конспект «Оформление текста»

## Свойство font-size

Управляет размером шрифта. Значение свойства задаёт желаемую высоту символа шрифта. Причём единицы измерения могут быть [абсолютными или относительными](https://htmlacademy.ru/courses/307/run/8).

Самая часто используемая единица измерения размера шрифта — пиксели px:

p {

font-size: 20px;

}

Но, чтобы при изменении основного размера шрифта родителя его дочерние элементы пропорционально меняли свои размеры шрифта, есть специальная единица измерения — em.

Величина 1em — это *такой же* размер шрифта, что и у родителя. Соответственно, если нужно, чтобы шрифт дочернего элемента был всегда в 2 раза больше родительского, то надо задать значение 2em:

h1 {

font-size: 2em;

}

## Свойство line-height

Свойство управляет высотой строки или межстрочным интервалом.

По умолчанию это свойство имеет значение normal. Оно указывает браузеру, что межстрочный интервал нужно подобрать автоматически, исходя из размера шрифта. Спецификация рекомендует устанавливать его в пределах 100-120% от размера шрифта. То есть:

p {

font-size: 10px;

line-height: normal; /\* значение будет примерно 12px \*/

}

Значение normal позволяет всем нестилизованным текстам выглядеть удобочитаемо. Однако, если есть необходимость отойти от стилизации по умолчанию, line-height можно задать фиксированное абсолютное значение в px.

p {

font-size: 16px;

line-height: 26px;

}

Если нужно задать line-height относительное значение, но не такое, как normal, то значение задаётся в процентах или в виде множителя. В таком случае браузер вычисляет значение динамически в зависимости от font-size:

p {

font-size: 10px;

line-height: 150%; /\* вычисленное значение: 10px \* 150% = 15px \*/

line-height: 2; /\* вычисленное значение: 10px \* 2 = 20px \*/

}

Относительные значения более гибкие, чем абсолютные. Но для простых сайтов фиксированных font-size и line-height будет вполне достаточно.

## Свойство font-family

В значении этого свойства указывается список названий шрифтов, перечисленных через запятую. В начале списка располагают самый желаемый шрифт, затем менее желаемый, а в самом конце списка — общий тип шрифта. Браузер проходит по списку слева направо и использует первый найденный в системе или на сайте шрифт. Если название шрифта состоит из нескольких слов, то его нужно заключать в кавычки.

body {

font-family: "PT Sans", "Arial", sans-serif;

}

## Свойство font-weight

Свойство задаёт насыщенность или толщину шрифта. Шрифт может быть жирнее или тоньше обычного начертания. В качестве значения можно использовать ключевое слово или число. Самые часто встречающиеся значения:

* 400 или normal — обычный шрифт, значение по умолчанию;
* 700 или bold — жирный шрифт.

Например:

h1 {

font-weight: 400; /\* то же самое что и normal \*/

}

p {

font-weight: bold; /\* то же самое что и 700 \*/

}

## Свойство text-align

Описывает, как выравнивается текст и другие инлайновые элементы (изображения, инлайн-блоки, инлайн-таблицы и другие) внутри блока по горизонтали.

Свойство может принимать следующие значения:

1. left — выравнивание по левому краю блока, это значение по умолчанию;
2. right — по правому краю блока;
3. center — по центру блока;
4. justify — по ширине блока, при этом слова в строке будут размещаться так, чтобы занять равномерно всё пространство строки (пробелы между словами в таком случае становятся неравномерными, так как браузер «растягивает» слова в строке).

Важно помнить, что свойство text-align применяется именно к самому блоку-контейнеру, внутри которого находится текстовый контент:

HTML:

<p>

Я текст внутри абзаца

</p>

CSS:

p {

text-align: center;

}

## Свойство vertical-align

Свойством можно выравнивать *инлайновые* элементы относительно содержащей его строки. Самый простой пример — выровнять картинку <img> по вертикали в текстовой строке.

У свойства vertical-align много значений, но самые часто используемые:

1. top — выравнивание по верхнему краю строки;
2. middle — по середине строки;
3. bottom — по нижнему краю строки;
4. baseline — по базовой линии строки (значение по умолчанию).

В отличие от text-align свойство vertical-align задаётся самому элементу, а не содержащему его контейнеру:

HTML:

<p>

<img src="picture.png" alt="Я картинка">

Я текст внутри абзаца

</p>

CSS:

img {

vertical-align: middle;

}

## Свойство color

Цветом текста можно управлять свойством color.

Цвет может быть задан в виде ключевого слова (полный список ключевых слов приводится в [спецификации](https://www.w3.org/TR/css-color-3/#svg-color)). Например:

color: black; /\* чёрный цвет \*/

color: red; /\* красный цвет \*/

color: white; /\* белый цвет \*/

Ещё один вариант указания цвета — в виде [шестнадцатеричного значения](https://ru.wikipedia.org/wiki/%D0%A8%D0%B5%D1%81%D1%82%D0%BD%D0%B0%D0%B4%D1%86%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F). В этом случае цвет формируется из *красной*, *зелёной* и *синей* составляющих, заданных в виде шестнадцатеричного числа от 00 до ff. Помимо шести, цветовой код может содержать три знака, в этом случае второй символ в цветовых составляющих дублируется первым:

color: #000000; /\* чёрный цвет \*/

color: #f00; /\* красный цвет, то же что #ff0000 \*/

color: #fff; /\* белый цвет, то же что #ffffff \*/

Если не хочется иметь дело с шестнадцатеричными значениями, можно воспользоваться специальной функцией rgb, в которой указывается цвет в более привычном десятичном виде в диапазоне от 0 до 255 также в виде трёх цветовых составляющих, перечисленных через запятую:

color: rgb(0, 0, 0) /\* чёрный, то же что #000000 \*/

color: rgb(255, 0, 0) /\* красный, то же что #ff0000 \*/

color: rgb(255, 255, 255) /\* белый, то же что #ffffff \*/

У функции rgb есть расширенная версия — rgba. В этом случае помимо указания цвета последним значением указывается степень непрозрачности цвета — alpha. Значение может быть от 0 (полностью прозрачный) до 1 (полностью непрозрачный):

color: rgba(0, 0, 0, 0.5) /\* чёрный, непрозрачный на 50% \*/

color: rgba(255, 0, 0, 0.3) /\* красный, непрозрачный на 30% \*/

color: rgba(255, 255, 255, 0.9) /\* белый, непрозрачный на 90% \*/

## Контраст цвета текста и фона

Фоновое изображение и фоновый цвет блока всегда должен достаточно сильно контрастировать с цветом текста в этом блоке. Чем больше контраст, тем удобнее этот текст читать в разных условиях освещённости и на разных устройствах. Поэтому если вы задаёте блоку фоновое изображение, нужно обязательно дополнительно задавать подходящий фоновый цвет. В этом случае, пока изображение загружается, или в случае, если оно совсем не загрузится, текст всё равно можно будет прочитать:

p {

/\* идеальный контраст: цвет текста белый, цвет фона — чёрный \*/

background-color: #000000;

color: #ffffff;

}

span {

/\* плохой контраст: цвет текста и фона — серые \*/

background-color: #cccccc;

color: #dddddd;

}

## Свойство white-space, управление пробелами

Браузер игнорирует множественные пробелы и переносы строк в HTML-коде. С помощью свойства white-space можно управлять пробелами и переносами строк. Свойство принимает значения:

* nowrap — схлопывает лишние пробелы и отображает весь текст одной строкой без переносов;
* pre — сохраняет пробелы и переносы как в исходном коде аналогично тегу <pre>;
* pre-wrap — работает как значение pre, но добавляет автоматические переносы, если текст не помещается в контейнер;
* normal — режим по умолчанию: лишние пробелы и переносы строк схлопываются, текст переносится, пробелы в конце строк удаляются.

## Свойство text-decoration

Задаёт дополнительное оформление текста. Значения свойства:

1. underline — подчёркивание;
2. line-through — зачёркивание;
3. overline — надчёркивание;
4. none — убирает вышеперечисленные эффекты.

К тексту можно одновременно применить несколько эффектов, если перечислить значения через пробел:

p {

text-decoration: underline; /\* подчёркнутый текст \*/

}

span {

/\* подчёркнутый и зачёркнутый текст \*/

text-decoration: underline line-through;

}

Свойство text-decoration — [составное](https://htmlacademy.ru/courses/307/run/7). Оно раскладывается на отдельные свойства:

* text-decoration-line — вид линии: зачёркивание, подчёркивание или надчёркивание;
* text-decoration-style — стиль линии, может принимать значения:
  + solid — сплошная линия;
  + double — двойная линия;
  + dotted — точечная линия;
  + dashed — пунктирная линия;
  + wavy — волнистая линия.
* text-decoration-color — цвет линии.

## Свойство font-style

Свойством можно задать начертание текста. Его основные значения:

1. normal — обычное начертание;
2. italic — курсивное начертание;
3. oblique — наклонное начертание.

Если задано значение italic, браузер будет пытаться найти в заданном шрифте отдельное курсивное начертание символов. В некоторых шрифтах отдельный курсив предусмотрен.

Если отдельного курсивного начертания в шрифте не предусмотрено, то браузер сделает текст наклонным, то есть сымитирует курсив. Что равноценно заданию тексту значения font-style: oblique.

## Свойство text-transform

С его помощью можно управлять регистром символов: делать буквы строчными (маленькими) или заглавными (большими). Значения свойства:

1. lowercase — все строчные;
2. uppercase — все заглавные;
3. capitalize — каждое слово начинается с большой буквы;
4. none — отменяет изменение регистра.

## Отступы

Важный фактор того, что текст в блоке будет удобочитаемым, это наличие свободного пространства в блоке для этого текста. Вокруг текста должно быть достаточно «воздуха», он не должен «прилипать» к краям, ему не должно быть «тесно».

За отступы в CSS отвечают два свойства: padding задаёт внутренние отступы в блоке, а margin задаёт внешние отступы.

# Конспект «Основы JavaScript»

## JavaScript: что это такое и как подключить его на страницу

Язык программирования JavaScript придумали специально для того, чтобы создавать интерактивные сайты.

Код на языке JavaScript называют скриптом. Его сохраняют в отдельный файл с расширением js, а чтобы запустить, подключают этот файл на страницу. В HTML для добавления JavaScript есть специальный тег:

<script src="адрес\_файла"></script>

Подключают скрипт обычно в самом конце страницы, перед закрывающим тегом </body>.

Программа на JavaScript — это последовательность инструкций, то есть указаний браузеру выполнить какие-то действия. Инструкции выполняются последовательно, сверху вниз.

Чтобы сказать JavaScript, что инструкция закончена, нужно поставить точку с запятой или перейти на новую строку. Новая строка правильно работает *в большинстве случаев*, а точка с запятой — **всегда**. Поэтому лучше ставить точку с запятой в конце каждой инструкции.

JavaScript не меняет исходный файл с разметкой, но, выполняя инструкции, меняет страницу прямо в браузере пользователя.

## Комментарии

Комментарий — это текст, поясняющий код. Он не выводится в браузер и никак не влияет на работу программы. Инструкции внутри комментария не выполняются, поэтому комментарии часто используют, если нужно временно отключить часть кода.

В JavaScript есть два вида комментариев:

// Однострочные комментарии.

/\*

И многострочные.

Они могут отключить сразу несколько строк кода.

\*/

## Метод querySelector

Чтобы найти на странице элемент, нужно использовать метод querySelector, он ищет по селектору:

document.querySelector('селектор');

Эта инструкция состоит из двух частей. Первая часть — элемент, внутри которого будет искать JavaScript. Словом document обозначается веб-страница, к которой скрипт подключён. Неважно, как называется файл на самом деле, в JavaScript это всегда «документ». Он является элементом-родителем для любого другого элемента на странице.

Вторая часть инструкции — это то, что нужно сделать. Её называют методом.

## Консоль

Консоль — инструмент разработчика, который помогает тестировать код. Если во время выполнения скрипта возникнет ошибка, в консоли появится сообщение о ней. А ещё в консоль можно выводить текстовые подсказки. Чтобы вывести сообщение в консоль, нужно использовать console.log:

console.log('Привет от JavaScript!');

// Выведет: Привет от JavaScript!

console.log(document.querySelector('.page'));

// Выведет в консоль найденный элемент

## Переменная

Переменная — способ сохранить данные, дав им понятное название.

Переменную можно создать, или **объявить**, с помощью ключевого слова let. За ним следует имя переменной. После объявления в переменную нужно записать, или **присвоить**, какое-то значение:

let header = document.querySelector('header');

Имя переменной может быть почти любым, но не должно начинаться с цифры, а из спецсимволов разрешены только '\_' и '$'. Ещё для именования переменных нельзя использовать [зарезервированные слова](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Lexical_grammar#%D0%9A%D0%BB%D1%8E%D1%87%D0%B5%D0%B2%D1%8B%D0%B5_%D1%81%D0%BB%D0%BE%D0%B2%D0%B0). Имена переменных чувствительны к регистру: header, Header и HEADER — это разные переменные. Имя переменной должно описывать то, что в ней хранится.

Когда в коде встречается переменная, браузер вместо её имени подставляет присвоенное ей значение. Когда мы используем переменную, снова писать let не нужно:

console.log(header);

Ключевое слово let появилось в JavaScript в 2015 году, до этого для объявления переменных использовалось слово var.

## Методы для изменения классов

Чтобы убрать у элемента класс, нужно использовать метод classList.remove. Он убирает с элемента тот класс, который указан в скобках:

элемент.classList.remove('класс');

Чтобы добавить элементу класс, нужно использовать метод classList.add:

элемент.classList.add('класс');

Метод-переключатель classList.toggle убирает у элемента указанный класс, если он есть, и добавляет, если этого класса нет:

элемент.classList.toggle('класс');

## Свойство textContent

У каждого элемента имеется множество свойств: его размеры, цвет и так далее. Свойство textContent хранит в себе текстовое содержимое элемента. Свойствам можно присваивать новые значения:

let paragraph = document.querySelector('p');

paragraph.textContent = 'Здесь был Кекс. Мяу!';

## Свойство value

У полей ввода есть особое свойство — value. Оно хранит данные, введённые в поле. Мы можем вывести их прямо на страницу:

let input = document.querySelector('input');

paragraph.textContent = input.value;

## Конкатенация

Операция, когда мы «склеиваем» несколько значений, называется **конкатенацией** и в JavaScript выполняется с помощью знака плюс.

let name = 'Кекс';

paragraph.textContent = 'Вас зовут ' + name + '. Хорошего дня!';

console.log(paragraph.textContent);

// Выведет: Вас зовут Кекс. Хорошего дня!

## Обработчики событий onclick и onsubmit

JavaScript следит за всем, что происходит на странице. Клик по кнопке или отправка формы — это *событие*. Мы можем сказать JavaScript, что сделать, когда некое событие произойдёт. Для этого используют обработчики событий. Инструкции, которые должны будут выполниться, когда событие произойдёт, располагают между фигурных скобок.

Свойство onclick означает «по клику»:

let button = document.querySelector('button');

button.onclick = function() {

console.log('Кнопка нажата!');

};

При каждом клике по кнопке в консоли будет появляться новое сообщение Кнопка нажата!.

За обработку отправки формы отвечает свойство onsubmit:

let form = document.querySelector('form');

form.onsubmit = function() {

console.log('Форма отправлена!');

};

После отправки формы в консоли появится сообщение Форма отправлена!.

// Короткая запись

<p><?= $name ?></p>

Работают эти два варианта одинаково. Разница только в количестве символов.

## База данных

База данных — сложная система, которая хранит информацию сайта в организованном виде.

## Работа с адресом

Адресная строка — специальное поле в браузере, в котором написаны адреса страниц в интернете.

В адресе есть специальная часть, которая называется «параметры запроса». Параметры запроса располагаются после знака вопроса.

https://www.gloevk.ru/product.php?product\_id=1

С помощью команды $\_GET можно получить информацию из адреса.

Чтобы получить значение product\_id с помощью $\_GET, нужно написать $\_GET['product\_id'].