Оглавление

[**1 Знакомство с HTML и CSS** 5](#_Toc20151943)

[**1.1 HTML** 5](#_Toc20151944)

[1.1.1 Парные теги 5](#_Toc20151945)

[1.1.2 Одиночные теги 5](#_Toc20151946)

[1.1.3 Комментарии 6](#_Toc20151947)

[**1.2 CSS** 6](#_Toc20151948)

[1.2.1 CSS-правила 7](#_Toc20151949)

[1.2.2 Миксование классов 8](#_Toc20151950)

[1.2.3 Комментарии 8](#_Toc20151951)

[**2 Структура HTML-документа** 9](#_Toc20151952)

[**3 Разметка текста** 12](#_Toc20151953)

[**3.1 Списки** 12](#_Toc20151954)

[3.1.1 Неупорядоченный список 12](#_Toc20151955)

[3.1.2 Упорядоченный список 12](#_Toc20151956)

[3.1.3 Список описаний 13](#_Toc20151957)

[**3.2 Преформатированный текст и код** 13](#_Toc20151958)

[**3.3 Цитаты** 14](#_Toc20151959)

[3.3.1 Небольшие цитаты 14](#_Toc20151960)

[3.3.2 Источник цитат 14](#_Toc20151961)

[3.3.3 Длинные цитаты 15](#_Toc20151962)

[**3.4 Разметка фрагментов текста** 15](#_Toc20151963)

[3.4.1 Символы-мнемоники 15](#_Toc20151964)

[3.4.2 Перенос строк 15](#_Toc20151965)

[3.4.3 Верхний и нижний индексы 16](#_Toc20151966)

[3.4.3 Дата и время 16](#_Toc20151967)

[3.4.4 Акцентирование внимания 17](#_Toc20151968)

[3.4.5 Выделение и придание важности 17](#_Toc20151969)

[3.4.6 Описание изменений 18](#_Toc20151970)

[**3.5 Разделение контента** 18](#_Toc20151971)

[**4 Ссылки и изображения** 20](#_Toc20151972)

[**4.1 Ссылки** 20](#_Toc20151973)

[4.1.1 Что такое ссылка 20](#_Toc20151974)

[4.1.2 Абсолютные ссылки 20](#_Toc20151975)

[4.1.3 Относительные ссылки 21](#_Toc20151976)

[4.1.4 Ссылки на файл 21](#_Toc20151977)

[4.1.5 Ссылки-якоря 21](#_Toc20151978)

[**4.2 Изображения** 21](#_Toc20151979)

[4.2.1 Подключение изображений 21](#_Toc20151980)

[4.2.2 Форматы изображений 22](#_Toc20151981)

[4.2.3 Размеры изображения 22](#_Toc20151982)

[4.2.4 Атрибут alt 23](#_Toc20151983)

[4.2.5 Figure и figcaption 23](#_Toc20151984)

[4.2.6 Изображение-ссылка 23](#_Toc20151985)

[**5 Основы CSS** 24](#_Toc20151986)

[**5.1 CSS-правила** 24](#_Toc20151987)

[**5.2 Селекторы** 24](#_Toc20151988)

[**5.3 Свойства и значения** 25](#_Toc20151989)

[**5.5 Наследование** 25](#_Toc20151990)

[**5.6 Составные свойства** 25](#_Toc20151991)

[**5.7 Типы значений: абсолютные и относительные** 26](#_Toc20151992)

[**5.8 Стили по умолчанию** 26](#_Toc20151993)

[**5.9 Каскадирование** 26](#_Toc20151994)

[**5.10 Конфликт свойств** 27](#_Toc20151995)

[**5.11 Встраивание и подключение внешних стилей** 27](#_Toc20151996)

[**6 Оформление текста** 29](#_Toc20151997)

[**6.1 Свойство font-size** 29](#_Toc20151998)

[**6.2 Свойство line-height** 29](#_Toc20151999)

[**6.3 Свойство font-weight** 30](#_Toc20152000)

[**6.4 Свойство text-align** 31](#_Toc20152001)

[**6.5 Свойство vertical-align** 31](#_Toc20152002)

[**6.6 Свойство color** 32](#_Toc20152003)

[**6.7 Контраст цвета текста и фона** 33](#_Toc20152004)

[**6.7 Свойство white-space, управление пробелами** 34](#_Toc20152005)

[**6.8 Свойство text-decoration** 34](#_Toc20152006)

[**6.9 Свойство font-style** 35](#_Toc20152007)

[**6.10 Свойство text-transform** 36](#_Toc20152008)

[**6.11 Отступы** 36](#_Toc20152009)

[7 Таблицы 36](#_Toc20152010)

[**7.1 Построение таблицы** 36](#_Toc20152011)

[7.2 Задание рамок с помощью CSS 37](#_Toc20152012)

[**7.3 Отступы внутри ячеек** 37](#_Toc20152013)

[7.4 Отступы между ячейками 38](#_Toc20152014)

[**7.5 Ячейки-заголовки** 38](#_Toc20152015)

[**7.6 Заголовок таблицы** 39](#_Toc20152016)

[**7.7 Объединение ячеек в строках** 39](#_Toc20152017)

[**7.8 Объединение ячеек в столбцах** 40](#_Toc20152018)

[**7.9 Комбо: colspan + rowspan** 40](#_Toc20152019)

[**7.10 Выравнивание содержимого в ячейках** 40](#_Toc20152020)

[**7.11 Добавим цвета** 41](#_Toc20152021)

[**7.12 Задание размера таблицы** 42](#_Toc20152022)

[**7.13 Задание размера отдельных ячеек и столбцов** 43](#_Toc20152023)

[8 Формы 44](#_Toc20152024)

[**8.1 Формы. Основные понятия.** 44](#_Toc20152025)

[**8.2 Поле ввода** 45](#_Toc20152026)

[**8.3 Идентификатор и значение по умолчанию** 45](#_Toc20152027)

[**8.4 Подпись для поля ввода** 46](#_Toc20152028)

[**8.5 Связывание подписи и поля по id** 46](#_Toc20152029)

[8.6 Кнопка отправки формы 46](#_Toc20152030)

[**8.7 Многострочное поле ввода** 47](#_Toc20152031)

[**8.8 Чекбокс или «галочка»** 47](#_Toc20152032)

[**8.9 Переключатель или «радиобаттон»** 48](#_Toc20152033)

[**8.10 Группа переключателей** 48](#_Toc20152034)

[**8.11 Раскрывающийся список или «селект»** 48](#_Toc20152035)

[**8.12 «Мультиселект»** 49](#_Toc20152036)

[**8.13 Поле для загрузки файлов** 49](#_Toc20152037)

[**8.14 Скрытое поле** 50](#_Toc20152038)

[**9 Селекторы в CSS** 51](#_Toc20152039)

[9.1 Селекторы по тегам 51](#_Toc20152040)

[**9.2 Селекторы по классам** 51](#_Toc20152041)

[**9.3 Контекстные селекторы** 52](#_Toc20152042)

[**9.4 Соседние селекторы** 53](#_Toc20152043)

[9.4.1 Контекстные и соседние селекторы 54](#_Toc20152044)

[**9.5 Дочерние селекторы** 54](#_Toc20152045)

[**9.6 Псевдоклассы** 55](#_Toc20152046)

[9.6.1 Псевдокласс :nth-child 55](#_Toc20152047)

[9.6.2 :nth-child и контекстные селекторы 56](#_Toc20152048)

[9.6.3 Псевдокласс :hover 56](#_Toc20152049)

[9.6.4 Динамические эффекты с помощью :hover 57](#_Toc20152050)

[9.6.5 Псевдоклассы :link, :visited и :active 57](#_Toc20152051)

[9.6.6 Псевдокласс :focus 58](#_Toc20152052)

[9.7 Селекторы атрибутов 58](#_Toc20152053)

[9.7.1 Селектор по id 59](#_Toc20152054)

[**10 Наследование и каскадирование в CSS** 60](#_Toc20152055)

[10.1 Иерархическое дерево 60](#_Toc20152056)

[10.2 Наследование 60](#_Toc20152057)

[10.2.1 Наследование «на пальцах» 61](#_Toc20152058)

[10.2.2 Ещё немного про наследование 61](#_Toc20152059)

[10.2.3 Наследуемые свойства 62](#_Toc20152060)

[10.2.4 Ненаследуемые свойства 63](#_Toc20152061)

[10.2.5 Принудительное наследование 63](#_Toc20152062)

[**10.3 Каскадирование** 63](#_Toc20152063)

[10.3.1 Расчёт значения специфичности 67](#_Toc20152064)

[10.3.2 Перекрёстное наследование 68](#_Toc20152065)

**1 Знакомство с HTML и CSS**

**1.1 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>.

1.1.1 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>.

1.1.2 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">

1.1.3 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.

**1.2 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>

1.2.1 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;

}

1.2.2 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.

1.2.3 Комментарии

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

**2 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>Курсы — 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.

**3 Разметка текста**

**3.1 Списки**

3.1.1 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.

3.1.2 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. 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.

3.1.3 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.

**3.2 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>

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 <code> можно вкладывать внутрь тега <pre>.

**3.3 Цитаты**

3.3.1 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.

3.3.2 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>

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

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

3.3.3 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.

**3.4 Разметка фрагментов текста**

3.4.1 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;

3.4.2 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.

3.4.3 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

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

3.4.3 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.

3.4.4 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>!

3.4.5 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.

3.4.6 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>

**3.5 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>

**4 Ссылки и изображения**

**4.1 Ссылки**

4.1.1 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.

4.1.2 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.

4.1.3 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.

4.1.4 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>

4.1.5 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.

**4.2 Изображения**

4.2.1 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">

4.2.2 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.

4.2.3 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.

4.2.4 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.

Альтернативный текст изображения задаётся с помощью атрибута alt. Например:

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

4.2.5 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>.

<figure>

Схема, график, диаграмма или код

<figcaption>Подпись к содержимому</figcaption>

</figure>

4.2.6 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>

**5 Основы CSS**

**5.1 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 \*/.

**5.2 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 {…}

**5.3 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.

**5.5 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.

**5.6 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.

**5.7 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.

**5.8 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.

**5.9 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;

**5.10 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.

**5.11 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.

**6 Оформление текста**

**6.1 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;}

**6.2 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.

**6.3 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 \*/

}

**6.4 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;}

**6.5 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;

}

**6.6 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/" \l "svg-color" \t "_blank)). 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% \*/

**6.7 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: #ddddddd;

}

**6.7 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.

**6.8 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 через пробел можно одновременно применить несколько эффектов, если перечислить значения:

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.

**6.9 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.

**6.10 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.

**6.11 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.

# 7 Таблицы

**7.1 Построение таблицы**

Простейшая таблица описывается с помощью трёх тегов:

1. <table> обозначает таблицу.
2. <tr> расшифровывается как *«table row»*, обозначает строку таблицы.
3. <td> расшифровывается как *«table data»*, обозначает ячейку внутри строки таблицы.

Теги <td> располагаются внутри тегов <tr>, а те, в свою очередь, внутри <table>. Почти всё текстовое содержимое таблицы размещается внутри тегов <td>.

В простейшей таблице в каждой строке должно быть одинаковое количество ячеек, то есть внутри всех <tr> должно быть одинаковое количество <td>.

Для того чтобы добавить столбец в таблицу, надо в каждую строку <tr> добавить по ячейке <td>.

# 7.2 Задание рамок с помощью CSS

Рамки по умолчанию отображаются, если у тега <table> задан атрибут border с ненулевым значением. С помощью атрибута border  можно только изменять их толщину.

С помощью CSS-свойства border можно задавать как внешние рамки таблицы, так и рамки каждой ячейки.

По умолчанию браузер рисует рамки таблицы и рамки отдельных ячеек раздельно, это отлично видно на примере.

Чтобы избавиться от таких двойных рамок, используется CSS-свойство таблицы border-collapse. Вот так:

table {border-collapse: collapse;}

Значение collapse убирает двойные рамки: cхлопываются рамки соседних ячеек, а также рамки ячеек и внешняя рамка таблицы. При этом внешняя рамка таблицы может исчезнуть, и, чтобы её вернуть, нужно увеличить её ширину.

Для отображения конкретных рамок ячейки необходимо использовать одно из свойств:

* border-top,
* border-right,
* border-bottom,
* border-left.

Эти свойства задают отображение только одной рамки ячейки: верхней, правой, нижней или левой соответственно.

**7.3 Отступы внутри ячеек**

Отступы внутри ячеек можно добавлять с помощью атрибута cellpadding тега <table>. Но лучше его не использовать, а задавать отступы с помощью CSS.

CSS-свойство padding задаёт «внутренние отступы элемента» со всех сторон. Можно задавать отступы для каждой из сторон отдельно, используя свойства:

* padding-top,
* padding-right,
* padding-bottom,
* padding-left.

Например, чтобы задать у ячеек все отступы в 10 пикселей, а отступ слева в 20 пикселей, нужно написать такой CSS-код:

td {

padding: 10px;

padding-left: 20px;

}

# 7.4 Отступы между ячейками

Большинство задач по оформлению таблиц решаются с помощью работы с рамками, отступами внутри ячеек, изменения цвета фона ячеек.

Помимо внутренних отступов можно задавать отступы между ячейками таблицы.

Отступы между ячейками не работают с border-collapse: collapse. Атрибут border-collapse: separate, это значение по умолчанию.

Отступы между ячейками можно задать:

* с помощью атрибута cellspacing тега <table>
* или c помощью CSS-свойства border-spacing.

Отметим, что свойство border-spacing задаётся для таблицы, в отличие от padding, которое задаётся для ячеек.

**7.5 Ячейки-заголовки**

Обычно в таблицах выделяют названия столбцов или строк. В HTML для этого предусмотрен специальный тег <th>, который расшифровывается как *«table header»*и обозначает ячейку-заголовок. Тег <th> аналогичен <td>, он так же должен располагаться внутри <tr>, для него стилями можно задавать все те же свойства.

По умолчанию текст внутри <th> выделяется жирным и выравнивается по центру ячейки.

**7.6 Заголовок таблицы**

Подпись (название) таблицы, делают с помощью тега <caption>.

Тег <caption> должен размещаться внутри тега <table>, причём непосредственно внутри него и первым, до остальных вложенных тегов. Вот так:

<table>

<caption>Текст</caption>

...

</table>

Тег заголовка идёт первым внутри таблицы, но с помощью CSS можно переместить заголовок таблицы в любое место: сверху или снизу таблицы, по центру, справа или слева.

По вертикали заголовок таблицы перемещается CSS-свойством caption-sideсо значениями top и bottom, которые обозначают *до* и *после* таблицы соответственно.

По горизонтали заголовок таблицы выравнивается CSS-свойством text-alignсо значениями left, right и center.

**7.7 Объединение ячеек в строках**



Чтобы объединить ячейки по горизонтали, необходимо использовать атрибут colspan у тегов <th> или <td>.

Когда вы задаёте ячейке атрибут colspan со значением 2, то ячейка «растягивается» на ячейку справа, но та ячейка не исчезает, а отодвигается и в таблице появляется новый столбец. Чтобы удалить его, нужно удалить ячейку, которая находится справа от «растянутой».

**7.8 Объединение ячеек в столбцах**

Объединение ячеек по вертикали осуществляется с помощью атрибута rowspan у тега <td> или <th>.

Когда вы задаёте ячейке атрибут rowspan со значением 2, то ячейка «растягивается» на следующую строку. При этом ячейка, которая была под «растянутой» отодвигается в своей же строке вправо, что добавляет в таблицу лишний столбец. Удалив ячейку, которая была под «растянутой», мы избавимся от этого столбца.

**7.9 Комбо: colspan + rowspan**

Можно растягивать ячейку одновременно и по вертикали, и по горизонтали. Для этого нужно задать ячейке два атрибута: rowspan и colspan.

Вытеснение соседних ячеек будет происходить так же, как и при обычном объединении ячеек. Правда, вытесненных ячеек окажется больше. Например, ячейка с rowspan="2" и colspan="2" вытеснит три соседних ячейки.

**7.10 Выравнивание содержимого в ячейках**

Содержимое ячеек можно выравнивать по горизонтали и по вертикали с помощью CSS.

За выравнивание по горизонтали отвечает CSS-свойство text-align. Чаще всего используются значения left, center и right.

За выравнивание по вертикали отвечает CSS-свойство vertical-align. Чаще всего используются значения top, middle и bottom.

Чтобы задать выравнивание содержимого ячеек, надо в стилях указать:

{

vertical-align: *значение*;

text-align: *значение*;

}

Эти стили повлияют на все ячейки. Чтобы задать выравнивание только в определённых ячейках, нужно назначить им классы и определить стили для классов. В этом примере для вас уже подготовлены ячейки с классами.

**7.11 Добавим цвета**

Таблицы можно раскрашивать, задавая цвет фона ячеек, цвет текста в ячейках, а также цвет рамок. Можно задавать и фоновые изображения.

Цветами можно управлять с помощью этих свойств:

* background-color — задаёт цвет фона,
* color — цвет текста,
* border-color — цвет рамок.

Значения цветов в CSS задаются [разными способами](https://developer.mozilla.org/en/docs/Web/CSS/color_value). Таким образом, чтобы задать цвета для ячейки в CSS, нужен такой код:

td {

color: *цвет*;

background-color: *цвет*;

border: 1px solid *цвет*;

}

Раскрашивать можно и td, и th, и даже table.

Если стили нужно задать для какой-то определённой ячейки, группы ячеек, или строки, можно использовать классы и применять стили для этих классов. Например, вот так:

.my-class {*стили*}

**7.12 Задание размера таблицы**

По умолчанию ширина и высота таблицы зависит от содержимого и отступов внутри ячеек.

С помощью CSS можно управлять размерами таблицы, задавать желаемую ширину и высоту. Также размерами можно управлять с помощью атрибутов таблицы.

Стоит отметить, что у таблицы есть минимальные размеры, которые зависят от содержания, меньше которых она не сожмётся, какое бы значение ширины или высоты ни задавалось.

Ширина таблицы задаётся с помощью CSS-свойства width, а высота с помощью свойства height, например:

table {

width: 100px;

height: 100px;

}

Размеры таблицы можно задавать как в абсолютных единицах, например, в пикселях — 20px, так и в относительных, в процентах — 20%.

При использовании процентов размеры таблицы будут вычисляться с учётом размеров родительского элемента, в нашем случае окна мини-браузера.

Особое значение auto включает расчёт размеров по умолчанию. Например, width: auto;или height: auto;.

Важное замечание. Проценты при задании высоты *обычно не работают*.

**7.13 Задание размера отдельных ячеек и столбцов**

Размеры ячеек и столбцов можно задавать вручную. Размеры ячеек задаются точно так же, как и размеры таблицы: с помощью CSS-свойств width и height.

Есть два варианта добавления стилей ячейкам:

1. Назначать ячейкам уникальные имена классов, например, class="cell-11", и применять стили для этих классов.
2. Использовать атрибут style, внутри которого можно писать CSS-код.

Пример второго варианта:

<td style="width: 100px;">

...

</td>

# 8 Формы

**8.1 Формы. Основные понятия.**

Формы нужны для того, чтобы отправлять данные с веб-страницы на веб-сервер, который сможет эти данные обработать: зарегистрировать пользователя, создать сообщение на форуме, отправить письмо и так далее.

Чтобы создать форму, нужно использовать парный тег <form>, внутри которого размещаются поля формы. У тега <form> есть два важных атрибута:

* action задаёт адрес, URL, отправки формы;
* method задаёт метод отправки формы.

Пример:

<form action="https://echo.htmlacademy.ru" method="get">

*поля формы*

</form>

Для отправки формы обычно используют методы get или post. Если не указать атрибут method, то будет использован get.

Метод get посылает данные формы в строке запроса, то есть они видны в адресной строке браузера и следуют после знака вопроса. Например:

https://www.google.com/search?*q=htmlacademy*

Метод get лучше использовать в поисковых формах, потому что он позволяет получить ссылку на результаты поиска и передать её кому-то.

Метод post посылает данные в теле HTTP-запроса и используется, когда нужно отправить много данных и ссылка на результат обработки этих данных не нужна. Например, при редактировании личного профиля.

**8.2 Поле ввода**

Большинство полей форм создаётся с помощью одиночного тега <input>. У этого тега два обязательных атрибута:

* type задаёт тип поля;
* name задаёт имя поля.

Тип поля влияет на то, как оно будет отображаться и вести себя. Самый распространённый тип — это text, который обозначает *текстовое поле*. Он же используется по умолчанию. Пример:

<form action="https://echo.htmlacademy.ru" method="get">

<input type="text" name="search">

</form>

Имя поля нужно, чтобы правильно обработать данные на сервере. Обычно, имя поля должно быть уникальным в пределах формы, хотя есть исключения. Для задания имени поля используют латинские буквы и цифры.

Поле предназначенное для ввода пароля, в котором текст будет отображаться «звёздочками», имеет атрибут type – password.

**8.3 Идентификатор и значение по умолчанию**

Атрибут id поля ввода обозначает *идентификатор*. Он должен быть уникальным не только в пределах формы, но и на всей странице.

Обычно идентификаторы используют для повышения удобства работы с формой, например, создают подписи, связанные с мелкими полями. При нажатии на такие подписи активируется связанное поле. И это удобно, так как по большой подписи попасть легче, чем по маленькому полю. Также идентификаторы используют в JavaScript для работы с полями.

Идентификатор в отличие от имени поля не передаётся на сервер. Лучше использовать идентификаторы, отличающиеся от имени поля.

Атрибут value задаёт значение поля ввода по умолчанию. Это тоже повышает удобство.

**8.4 Подпись для поля ввода**

Есть специальный тег, который позволяет логически связать подпись и поле - парный тег <label>. Если нажать на текст в такой подписи, то курсор переместится в соответствующее поле.

Первый способ создать подпись — просто обернуть текст подписи и тег поля в тег <label:

<label>

Подпись <input type="text" name="username">

</label>

**8.5 Связывание подписи и поля по id**

Иногда обернуть поле и текст подписи в тег <label> нельзя. Например, когда они размещены в разных ячейках таблицы.

В этом случае можно связать подпись с полем с помощью атрибута id. Алгоритм такой:

1. Добавляем к полю ввода идентификатор с помощью атрибута id.
2. Оборачиваем текст подписи в тег <label>.
3. Добавляем тегу <label> атрибут for.
4. В атрибут for записываем такое же значение, что и в атрибуте id у поля.

Например:

<label for="*user-field-id*">Имя пользователя</label>

...

*много-много других тегов*

...

<input id="*user-field-id*" type="text" name="username">

# 8.6 Кнопка отправки формы

Кнопка для отправки формы создаётся с помощью тега <input> c типом submit.

Надпись на кнопке можно задать с помощью атрибута value. Для кнопки отправки формы задавать имя необязательно. Но если имя задано, то на сервер будут отправляться имя и значение кнопки.

Обычно имя для кнопки отправки задают, когда в форме несколько кнопок, отвечающих за разные действия. Браузер отправляет на сервер имя и значение только той из них, на которую нажал пользователь. Таким образом, сервер может понять, какую кнопку нажали и что нужно сделать.

**8.7 Многострочное поле ввода**

Многострочное текстовое поле создаётся с помощью парного тега <textarea>. У него есть атрибуты name и id, которые аналогичны атрибутам текстового поля.

Атрибут rows принимает целочисленное значение и задаёт высоту многострочного поля в строках. Атрибут cols задаёт ширину поля в символах. В качестве ширины символа берётся некоторая «усреднённая ширина».

Атрибут value у многострочного поля отсутствует, а значение по умолчанию задаётся текстом, расположенным между открывающим и закрывающим тегом <textarea>

<textarea>Значение по умолчанию</textarea>

**8.8 Чекбокс или «галочка»**

Поле-галочка — это тег <input> с типом checkbox.

Галочка работает по принципу «либо да, либо нет». Если галочка стоит, то браузер посылает переменную с именем поля на сервер, если галочки нет, то не посылается ничего. Таким образом, атрибут value не является обязательным.

Чтобы галочка стояла по умолчанию, нужно добавить к тегу атрибут checked. Вот так:

<input type="checkbox" checked>

Чекбокс не подразумевает выбор одного элемента из нескольких. Поэтому если в одной форме есть несколько чекбоксов, то имена у них должны быть *разными*.

**8.9 Переключатель или «радиобаттон»**

Поле-переключатель — это тег <input> с типом radio.

Обычно переключатели размещают группами по несколько штук. Причём у переключателей из одной группы должно быть *одинаковое имя* и разные значения, которые задаются c помощью value.

Таким образом, атрибут value является для переключателей обязательным. Браузер отправляет на сервер значение value выбранного переключателя.

**8.10 Группа переключателей**

Теперь добавим ещё один вариант ответа в наш переключатель. Для этого нужно добавить ещё один <input> с таким же именем, но другим значением value. Подобным образом можно создавать группы переключателей с любым количеством вариантов.

Чтобы сделать какой-либо вариант в переключателе выбранным по умолчанию, нужно добавить к соответствующему тегу <input> атрибут checked, как у поля-галочки.

Кстати, имя поля у переключателей одной группы должно быть одинаковым, но идентификаторы всегда должны быть уникальными.

**8.11 Раскрывающийся список или «селект»**

Раскрывающийся список так же, как и переключатель, позволяет выбрать один вариант ответа из нескольких.

Раскрывающийся список создаётся с помощью парного тега <select>, у которого есть знакомые атрибуты name и id.

Варианты ответов задаются с помощью парных тегов <option>, которые должны располагаться внутри тега <select>. Например:

<select name="theme">

<option value="light">Светлая тема</option>

<option value="dark">Тёмная тема</option>

...

</select>

В атрибуте value тега <option> задаётся значение варианта ответа, а внутри этого тега располагается подпись варианта ответа.

Если при отправке формы у выбранного варианта задан value, то на сервер отправится значение этого атрибута. В противном случае будет отправлен текст подписи.

**8.12 «Мультиселект»**

Раскрывающийся список можно превратить в так называемый «мультиселект», то есть список, в котором можно выбрать не один, а несколько вариантов.

Чтобы сделать это, нужно добавить к тегу <select> атрибут multiple.

Выбрать несколько вариантов можно, щёлкая по ним с зажатой клавишей Ctrl на Windows или Command на MacOS.

Высоту мультиселекта можно изменять с помощью атрибута size тега <select>.

Чтобы отметить как выбранные по умолчанию одно или несколько значений, нужно к соответствующим тегам <option> добавить атрибут selected.

При отправке данных мультиселекта на сервер с PHP после имени в значении атрибута name ставятся символы квадратных скобок []. Например, <select name="days[]">. Это необязательное требование для имени мультиселекта, а нужно только для корректной обработки данных в PHP.

**8.13 Поле для загрузки файлов**

Поле для загрузки файлов — это тег <input> с типом file. Для этого поля обязательным атрибутом является имя.

Чтобы поле заработало и браузер смог передать выбранный файл на сервер, необходимо добавить *форме* атрибут enctype со значением multipart/form-data. Не полю, а форме.

Внешний вид таких полей очень сильно отличается в зависимости от операционной системы и очень плохо изменяется с помощью стилей.

**8.14 Скрытое поле**

Его используют, когда в форме нужно отправить какие-то дополнительные служебные данные, которые не вводятся пользователем.

Скрытое поле — это тег <input> с типом hidden.

**9 Селекторы в CSS**

Селекторы позволяют очень точно указывать к каким элементам применять CSS-свойства.

Но для начала представим, что механизма селекторов не существует. Как в этом случае задавать CSS-свойства элементам? С помощью атрибута style. Например:

<p style="color: red;">...</p>

# 9.1 Селекторы по тегам

Задавать атрибут style для каждого тега неудобно и долго. Особенно, если тот же результат можно получить с помощью единственного CSS-правила, в котором используется селектор для тега li.

С помощью селекторов по именам тегов можно задать стили для всех элементов списка, изображений, абзацев и так далее. Эти селекторы содержат имя тега без символов < и >. Например:

*li* {/\* стили для элементов списка \*/}

**9.2 Селекторы по классам**

Класс — это один из атрибутов тегов. Выглядит он вот так:

<li class="first"></li>

Этот атрибут особенный, так как в CSS существует возможность выбирать элементы по классу. Делается это с помощью такого селектора: .*имя\_класса*. Например:

.first {/\* стили для класса *first* \*/}

Имена классов могут состоять из латинских символов, цифр и знаков - и \_. Имя класса должно начинаться с латинской буквы.

Синтаксис CSS позволяет выбирать элементы не только по одному классу или тегу. Можновыбрать элемент одновременно по тегу и по классу или же элемент с двумя классами сразу. Для этого селектор составляется просто одной строкой из всех желаемых «частей» без пробелов. Давайте рассмотрим примеры.

В селекторе по тегу и классу первым пишется название тега, а потом идёт класс:

/\* выбор всех тегов ul с классом target \*/

ul.target {...}

<ul class="target"></ul>

Если у элемента задано несколько классов, в HTML и в CSS-селекторе они могут идти в разном порядке — это не будет влиять на выборку элементов:

/\* выбор элементов в двумя классами: text и green \*/

.text.green {...}

<span class="text green"></span>

<p class="green text"></p>

Количество классов в селекторе может быть любым:

/\* выбор тегов span в четырьмя классами: underlined, red, big и text \*/

span.underlined.red.big.text {...}

<span class="underlined red big text"></span>

**9.3 Контекстные селекторы**

Селектор может состоять из нескольких частей, разделённых пробелом, например:

p strong { ... }

ul .hit { ... }

.footer .menu a { ... }

Такие селекторы называют *контекстными* или *вложенными*. Их используют для того, чтобы применить стили к элементу, только если он вложен в нужный элемент.

Например, селектор .menu a сработает для ссылки a только в том случае, если она расположена внутри элемента с классом .menu.

Читать их проще всего справа налево:

/\* выбрать все теги strong внутри тегов p \*/

p strong { ... }

/\* выбрать все элементы с классом .hit внутри тегов ul \*/

ul .hit { ... }

/\* выбрать все ссылки внутри элементов с классом .menu,

которые лежат внутри элементов с классом .footer \*/

.footer .menu a { ... }

Таким образом, можно задавать элементам различные стили в зависимости от их контекста. Если ссылка расположена внутри меню, сделать её крупнее, а если внутри основного текста, то задать ей нужный цвет.

**9.4 Соседние селекторы**

Контекстные селекторы используются для вложенных друг в друга элементов, а соседние — для расположенных рядом.

Например, теги <li> в списке являются соседними по отношению друг к другу и вложенными в тег <ul>.

Соседние селекторы записываются с помощью знака +, например, селектор1 + селектор2. Стили применятся к элементу, подходящему под селектор2, только если сразу перед ним расположен элемент, подходящий под селектор1.

Пример. Есть два элемента списка:

<ul>

<li class="hit"></li>

<li class="miss"></li>

</ul>

Селектор .hit + .miss применит стили к элементу с классом miss, так как перед ним есть элемент с классом hit.

Селектор .hit + li, а также селектор li + .miss, или даже li + li тоже применит стили к элементу с классом miss, то есть ко второму элементу списка.

А вот селектор .miss + .hit не сработает, так как элемент с классом miss находится после элемента с классом hit в разметке.

# 9.4.1 Контекстные и соседние селекторы

Селекторы в CSS можно очень гибко комбинировать. В частности, можно комбинировать контекстные и соседние селекторы.

Например, селектор .player-1 .hit + .miss сработает для тега с классом miss, если сразу перед ним расположен тег с классом hit и оба тега расположены внутри тега с классом player-1.

**9.5 Дочерние селекторы**

Потомком называются любые элементы, расположенные внутри родительского элемента. А дочерними элементами называются ближайшие потомки. Взгляните на пример:

<ul>

<li><span>...</span></li>

<li><span>...</span></li>

</ul>

По отношению к списку <ul> элементы <li> являются дочерними элементами и потомками, а <span> — потомки, но не дочерние элементы.

Контекстные селекторы влияют на всех потомков, что не всегда удобно. Иногда необходимо задать стили только для дочерних элементов. Особенно это полезно при работе с [многоуровневыми списками](https://htmlacademy.ru/courses/38/run/5).

Для этого существует дочерний селектор, в котором используется символ >. Например: ul > li или ul > li > span.

**9.6 Псевдоклассы**

Псевдоклассы — это дополнения к обычным селекторам, которые делают их ещё точнее и мощнее. Обычный селектор — это снайперский прицел, а с псевдоклассом он становится прибором ночного видения.

Псевдокласс добавляется к селектору c помощью символа :, вот так селектор:псевдокласс. Например:

a:visited { ... }

li:last-child { ... }

.alert:hover { ... }

Знакомство с псевдоклассами мы начнём с first-child и last-child.

Псевдокласс first-child позволяет выбрать первый дочерний элемент родителя, а last-child — последний дочерний элемент. Например:

li:last-child { ... }

Этот селектор выберет последний элемент списка.

9.6.1 Псевдокласс :nth-child

Псевдоклассы из предыдущего примера относятся к семейству псевдоклассов, помогающих выбирать элементы по их расположению.

Вспомним [задание 4](https://htmlacademy.ru/courses/42/run/4). В нём каждому тегу <li> был задан собственный класс. Используя классы, мы могли выбрать любой из пяти тегов. Если бы тегов было десять, то пришлось бы использовать десять разных классов.

С помощью псевдокласса nth-child можно выбирать теги по порядковому номеру, *не используя классы*. Синтаксис псевдокласса: селектор:nth-child(*выражение*). *Выражением*может быть число или формула. Например:

1. li:nth-child(2) { ... }

2. li:nth-child(4) { ... }

3. li:nth-child(2n) { ... }

Первый селектор выберет второй элемент списка, второй селектор — четвёртый элемент списка, третий селектор — все чётные элементы.

9.6.2 :nth-child и контекстные селекторы

Селекторы с псевдоклассами хорошо сочетаются с контекстными селекторами.

Например, следующий селектор выберет третий тег <li> внутри блока с классом shooter-2.

.shooter-2 li:nth-child(3) { ... }

Селектор не обязательно должен заканчивается псевдоклассом. Например, следующий селектор выберет элемент с классом .text в четвёртом элементе списка >li<.

li:nth-child(4) .text { ... }

9.6.3 Псевдокласс :hover

Некоторые псевдоклассы позволяют выбирать элементы, с которыми взаимодействует пользователь. Сначала познакомимся с псевдоклассом :hover.

Этот псевдокласс позволяет выбрать элемент, когда на него наведён курсор мыши и кнопка мыши не нажата. Примеры:

1. a:hover { ... }

2. tr:hover { ... }

3 .menu-item:hover { ... }

Первый селектор выбирает ссылку, второй строку таблицы, третий элемент с классом menu-item, но только в том случае, если на них наведён курсор мыши.

Благодаря этому псевдоклассу можно добавлять в интерфейс динамику и интерактивность, так как элементы начинают реагировать на действия пользователя, изменяя свой внешний вид.

9.6.4 Динамические эффекты с помощью :hover

Интересовались ли вы, как с помощью CSS создаются выпадающие меню?

Львиная доля динамических эффектов, создаваемых с помощью CSS, опираются на несколько псевдоклассов, главный из которых, конечно же, :hover. Весь секрет заключается в сочетании контекстных селекторов и псевдоклассов. Посмотрите на пример:

li.top ul.submenu {

display: none;

}

li.top:hover ul.submenu {

display: block;

}

Первое правило прячет список-подменю. Второе правило гласит: «если на верхний пункт меню, в котором находится подменю, наведут курсор, то надо показать подменю». Вот так всё просто.

Общий принцип такой: родительский элемент реагирует на наведение мыши и изменяет свойства элементов-потомков. То есть всё работает на контекстных селекторах вида селектор1:hover селектор2.

9.6.5 Псевдоклассы :link, :visited и :active

Отвлечёмся от биатлона и познакомимся с псевдоклассами для ссылок.

* :link выбирает ещё не посещённые ссылки.
* :visited выбирает посещённые ссылки.
* :active выбирает активные ссылки (кнопка мыши зажата на ссылке).

Пример задания CSS-правил для ссылок:

a:link { ... }

a:visited { ... }

a:hover { ... }

a:active { ... }

Обратите внимание на порядок правил. Если их расположить по-другому, то некоторые могут не сработать.

# 9.6.6 Псевдокласс :focus

Псевдокласс :focus позволяет выбрать элемент, который в данный момент в фокусе. Например, текстовое поле, в которое установлен курсор, находится в фокусе.

В фокусе могут быть не только текстовые поля. Если вы переключаетесь между элементами веб-страницы с помощью клавиши **tab**, то в фокус будут попадать ссылки.

Пример использования псевдокласса:

input:focus { /\* стили для поля в фокусе \*/}

# 9.7 Селекторы атрибутов

Ранее мы познакомились с атрибутом class и специальными селекторами по классу. Существуют селекторы, которые позволяют выбирать элементы по любым атрибутам.

Чаще всего такие селекторы используются [при работе с формами](https://htmlacademy.ru/courses/50), так как поля форм имеют атрибут type с разными значениями.

Селекторы атрибутов записываются с использованием квадратных скобок: элемент[атрибут]. Примеры селекторов:

1. input[checked] { ... }

2. input[type="text"] { ... }

Первый селектор выберет поля формы, у которых есть атрибут checked, второй селектор выберет поля формы, у которых атрибут type имеет значение text.

9.7.1 Селектор по id

Существует ещё один HTML-атрибут, для которого существует специальный селектор. Этот атрибут id (идентификатор), а селектор записывается с помощью символа #, например, #some-id.

На значение id распространяются те же ограничения, что и на имя класса. Также id должен быть уникальным на странице.

Пример:

HTML<p id="greeting">Приветствие!</p>

CSS#greeting {...}

Использование селекторов по id при оформлении считается плохой практикой. Существуют редкие исключения из этого правила, например, при оживлении слайдера на чистом CSS.

**10 Наследование и каскадирование в CSS**

# 10.1 Иерархическое дерево

HTML-документ представляет собой иерархическое дерево. Это означает, что у каждого элемента (кроме корневого) есть только один родитель, то есть элемент, внутри которого он располагается. У корневого раздела родитель отсутствует. Рассмотрим простейшую страницу:

<html>

<head></head>

<body>

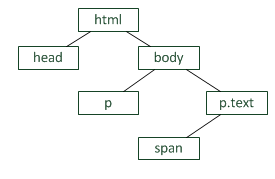
<p>Текст документа</p>

<p class="text">Выделенная <span>строка</span></p>

</body>

</html>

Для этой страницы можно нарисовать такое иерархическое дерево:



Оно схематически отображает структуру вложенности элементов. В данном примере видно, что у элемента span родителем является p.text, а у p.text родитель — body.

Иерархическая структура документа определяет основы концепции наследования.

# 10.2 Наследование

Наследование в CSS — механизм, с помощью которого значения свойств элемента-родителя передаются его элементам-потомкам.

Стили, присвоенные некоторому элементу, наследуются всеми потомками (вложенными элементами), если они не переопределены явно. Например, размер шрифта и его цвет достаточно применить к body, чтобы все элементы внутри имели те же свойства.

Наследование позволяет сократить размер таблицы стилей, но если стилей много, то отследить какой родительский элемент установил некоторое свойство, становится сложнее.

# 10.2.1 Наследование «на пальцах»

Давайте на простых примерах подробнее разберёмся, в чём же преимущество наследования. Рассмотрим пример:

<p class="text">Cтрока c выделенным <span>словом</span></p>

Представим, что нам нужно установить красный цвет текста для всего текста. Зададим CSS-свойства следующим образом:

.text {color: red;}

Благодаря наследованию цвет текста в теге span автоматически станет красным:

Cтрока c выделенным словом

А так бы выглядел результат, если бы наследование не работало:

Cтрока c выделенным словом

Нам пришлось бы отдельно прописывать цвет текста для тега span. И тогда установка таких простых свойств как стиль шрифта стала бы большой проблемой: нужно было бы задавать свойства для всех возможных вложенных тегов.

# 10.2.2 Ещё немного про наследование

Наверняка вы обращали внимание, что не все свойства наследуются тегами-потомками от их родителей.

Действительно, было бы странно, если бы свойство border автоматически устанавливалось для всех вложенных элементов.

Например, для этого куска кода:

<p class="bordered">Cтрока c выделенным <span>словом</span></p>

Установим CSS-свойство:

.bordered {border: 1px solid green;}

Если бы наследовались все свойства, то результат бы выглядел так:

Cтрока c выделенным словом

На самом деле граница будет нарисована только у тега p.

О том, какие именно свойства наследуются, мы расскажем в следующих заданиях.

10.2.3 Наследуемые свойства

К наследуемым свойствам относятся в первую очередь свойства, определяющие параметры отображения текста:

font-size, font-family, font-style, font-weight, color, text-align, text-transform, text-indent, line-height, letter-spacing, word-spacing, white-space, direction и т. д.

Также к наследуемым свойствам относятся list-style, cursor, visibility, border-collapseи некоторые другие. Но они используются значительно реже.

Весь список наследуемых свойств смотрите в [стандарте CSS](https://www.w3.org/TR/CSS21/propidx.html" \t "_blank). Значение yes в колонке Inherited?.

Эти свойства можно и нужно задавать через предков, следуя семантике документа.

Например, параметры текста зачастую не меняются в пределах отдельных блоков страницы: меню, основного содержания, информационных панелей. Поэтому общие параметры текста (цвет, размер, гарнитура) обычно указывают в стилях самих блоков.

# 10.2.4 Ненаследуемые свойства

В предыдущем задании мы перечислили основные наследуемые свойства. Все остальные относятся к ненаследуемым. Это параметры позиционирования, размеров, отступов, фона, рамок и т. д.

А именно: background, border, padding, margin, width, height, position и др.

Весь список ненаследуемых свойств смотрите в [стандарте CSS](https://www.w3.org/TR/CSS21/propidx.html" \t "_blank). Значение no в колонке Inherited?.

Не наследуются они из соображений здравого смысла. Например: если для какого-либо блока установлен внутренний отступ, автоматически выставлять такой же отступ каждому вложенному элементу нет никакой надобности. Эти параметры чаще всего уникальны для каждого отдельного блока.

# 10.2.5 Принудительное наследование

Для каждого свойства может быть задано значение inherit.

Оно означает, что данное свойство принимает такое же значение, как и у родительского элемента. Значение inherit может быть использовано для усиления наследуемых значений, а также в свойствах, которые обычно не наследуются.

Запись выглядит следующим образом:

p {background: inherit;}

В данном случае у тегов p свойство background будет таким же, как и у их родительских тегов.

**10.3 Каскадирование**

CSS расшифровывается как «*Cascading* Style Sheets» или «*каскадные* таблицы стилей».

Каскадность обозначает, что к одному и тому же элементу может применяться несколько CSS-правил (наборов CSS-свойств). Среди этих свойств могут быть и конфликтующие между собой. Поэтому существуют инструкции, которые определяют, каким будет финальный набор свойств элемента.

Например, для элемента:

<p class="text" style="color: red;"></p>

CSS-правила существуют как минимум в трёх разных местах:

1. в подключаемом файле style.css для селекторов p или .text;
2. в атрибуте style;
3. в стандартных стилях отображения, встроенных в браузер.

Каскадирование как раз и определяет, какие именно свойства из этих источников применятся к данному абзацу.

Имеется три основные концепции, управляющие порядком, в котором применяются CSS-свойства:

1. важность;
2. специфичность;
3. порядок исходного кода.

Самыми важными для нас являются последние две и подробнее о них мы поговорим отдельно.

Разберём пример HTML-кода, в котором есть абзац с двумя классами:

<p class="red blue">Синий или красный?</p>

И CSS-код c двумя правилами для этих классов:

.blue {color: blue;}

.red {color: red;}

Вопрос: какого цвета будет текст абзаца? Какое из CSS-правил приоритетнее?

Ответ: красного цвета, второе правило приоритетнее.

Это происходит потому, что селекторы у правил одинакового типа, и соответственно обладают одинаковой специфичностью. В таком случае более высоким приоритетом обладает то правило, которое расположено в CSS-коде ниже. В нашем случае это правило для класса red.

Чуть более сложный пример. Тот же HTML:

<p class="red blue">Синий или красный?</p>

Немного другой CSS:

p.blue {color: blue;}

.red {color: red;}

В этом случае текст абзаца будет синим. Происходит это потому, что селектор p.blue *более специфичный*, чем селектор .red.

Простое объяснение специфичности звучит так:

Чем меньшее количество элементов потенциально может выбрать селектор, тем он специфичнее.

В нашем примере селектор .red выберет *все теги* с нужным классом, а селектор p.blueвыберет *только абзацы* с нужным классом.

А теперь посмотрим, как ведут себя контекстные селекторы.

<div class="experiment">

<p class="red blue">Синий или красный?</p>

</div>

И в CSS используем контекстный селектор для второго правила:

p.blue {color: blue;}

.experiment .red {color: red;}

Какой теперь будет цвет текста?

Как вы уже знаете, существуют селекторы не только по классам, но и по id. Они начинаются с решётки #.

HTML:

<div id="experiment-1" class="experiment">

<p class="red blue">Синий или красный?</p>

</div>

CSS:

#experiment-1 .blue {color: blue;}

.experiment .red {color: red;}

Особенность атрибута id заключается в том, что его значение должно быть уникальным в пределах страницы. То есть, может существовать только один тег с определённым значением id.

Получается, что селектор по id может выбрать только один элемент. И поэтому он на порядок специфичнее селекторов по тегам, классам, а также комбинаций этих селекторов.

/\*Битва подходит к концу, и чтобы удержать победу, Рудольф решается на запрещённый приём: он прописывает стили с помощью атрибута style прямо в теге курочки.\*/

CSS-правила, которые прописаны в style обладают наивысшим приоритетом. Такой способ задания стилей не приветствуется в профессиональной вёрстке сайтов и годится только для создания быстрых прототипов. Поэтому мы и назвали этот приём запрещённым.

Однако существует способ переопределить из подключаемых CSS-файлов даже стили, заданные в атрибуте style. Для этого нужно использовать ключевое слово !important. Оно задаёт CSS-свойству усиленный приоритет. Вот пример:

HTML:

<p style="color: red;" class="blue">Синий или красный?</p>

CSS:

.blue {color: blue !important;}

Цвет текста в этом примере будет синим.

При вёрстке не рекомендуется часто использовать !important. По возможности старайтесь обходиться без него.

Как вы уже догадались, самым главным механизмом для определения приоритетов стилей является специфичность. Поэтому давайте ещё немного потренируемся работать с ней.

Рассмотрим пример: на полу в коробке сидит кот

<div id="floor">

<span class="cat-in-box">Кексик</span>

</div>

Допустим в стилях существуют следующие определения:

span {background-color: #27ae60; /\* Зелёный \*/}

div span {background-color: #2980b9; /\* Синий \*/}

#floor .cat-in-box {background-color: #34495e; /\* Мокрый асфальт \*/}

.cat-in-box {background-color: #8e44ad; /\* Фиолетовый \*/}

#floor span {background-color: #c0392b; /\* Красный \*/}

div .cat-in-box {background-color: #e67e22; /\* Оранжевый \*/}

А теперь вопрос на засыпку: какого цвета будет коробка? Сначала сделайте предположение, а затем проверьте.

10.3.1 Расчёт значения специфичности

Вы вдоволь наигрались со специфичностью, а теперь пришло время изучить полные правила её вычисления.

Специфичность селектора разбивается на 4 группы — a, b, c, d:

* если стиль встроенный, то есть определён как style="...", то а=1, иначе a=0;
* значение b равно количеству идентификаторов (тех, которые начинаются с #) в селекторе;
* значение c равно количеству классов, псевдоклассов и селекторов атрибутов;
* значение d равно количеству селекторов типов элементов и псевдо-элементов.

После этого полученное значение приводится к числу (обычно в десятичной системе счисления). Селектор, обладающий большим значением специфичности, обладает и большим приоритетом.

Посчитаем специфичность в нашем примере:

|  |  |  |
| --- | --- | --- |
| Селектор | a, b, c, d | Число |
| span | 0, 0, 0, 1 | 1 |
| div.cat-in-box | 0, 0, 1, 1 | 11 |
| **#floor.cat-in-box** | **0, 1, 1, 0** | **110** |
| div span | 0, 0, 0, 2 | 2 |
| .cat-in-box | 0, 0, 1, 0 | 10 |
| #floor span | 0, 1, 0, 1 | 101 |

Отсюда сразу видно, что в нашем примере самым приоритетным является селектор #floor .cat-in-box.

Если два CSS-правила применяются к одному и тому же элементу и имеют одинаковую специфичность, то более приоритетным будет то правило, которое появится в коде позже другого.

10.3.2 Перекрёстное наследование

При создании стилей для сходных по внешнему виду или функциональности элементов, которые могут использоваться на странице неоднократно, очень удобно пользоваться перекрёстным наследованием.

Приём этот заключается в следующем:

1. создаётся базовый стиль для таких элементов;
2. определяются вспомогательные стили, которые применяются к элементам по мере надобности;
3. элемент наследует базовый стиль и один или несколько вспомогательных.

**Пример:**

На странице используются кнопки разного назначения: для отправки форм, для сброса информации в полях формы, как элементы навигации и т. д.

Можно вынести общее оформление (размеры, отступы и так далее) для всех кнопок в отдельное CSS-правило для класса, например, .button.

А затем создать дополнительные CSS-правила, в которых будут определены только различающиеся свойства этих кнопок, например, цвет фона. Для этих правил можно использовать такие названия классов: .button-send, .button-clear, .button-navigation.

Каждая кнопка в HTML-коде будет иметь два класса: общий и дополнительный.

<a class="button button-send">Отправить</a>