# Возможности HTML и CSS

## Структура HTML-страницы

### Теги header и footer, хедер и футер

<header> — хедер сайта или раздела;

<footer> — футер сайта или раздела.

Да-да, это те самые хедер и футер, которые раньше верстались дивами с классами header или footer.

Футеров и хедеров может быть несколько на странице при различных ситуациях. Хедер может быть шапкой какой-нибудь статьи. Для Хедера-шапки страницы поэтому нужно применять соответствующие классы. С футером также.

### Тег main, основное содержание

Для разметки основного содержимого используем тег <main>. Для его стилизации класс использовать не будем, так как на сайте будет только одно единственное основное содержимое.

### Теги article и section, разделы страницы

<section> — смысловой или логический раздел документа;

<article> — самостоятельный и независимый раздел документа.

<div> — контейнер общего назначения, не обязательно смысловой. Дивы используются для разметки мелких блоков, создания сетки и декоративных эффектов.

<section> — более крупный логический контейнер, объединяющий содержание по смыслу. Например, блок «О компании», список товаров, раздел личной информации в профиле и так далее.

<article> — самостоятельный, цельный и независимый раздел документа. Этот раздел можно в неизменном виде использовать в различных местах, в том числе и на других сайтах. Примеры: статья, пост в блоге, сообщение на форуме и так далее.

### Теги h1-h6, заголовки разделов страницы

Как правило, в разделы <section> и <article> включают заголовки, резюмирующие содержимое разделов.

### Тег nav, основная навигация

Есть специальный элемент для навигационных блоков — тег <nav>. Так что меню и блоки ссылок лучше оборачивать не в дивы, а в навы. В случае, когда навигация шапки дублируется в подвале сайта, не рекомендуется использовать тег <nav> повторно.

## Приемы стилизации

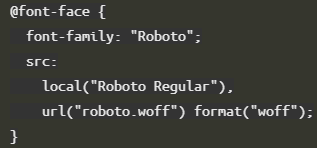
### Использование нестандартных шрифтов

Простейший способ использовать такой шрифт — найти его в специальном сервисе по названию, получить там код подключения шрифта, вставить этот код в свою вёрстку и использовать шрифт, как обычно, с помощью свойства font-family. При использовании веб-шрифтов не забывайте указывать так называемые «фоллбэк»-шрифты — стандартные шрифты, которые будут отображаться, если веб-шрифт либо недоступен, либо не поддерживается старым браузером:

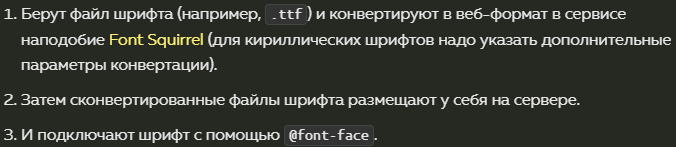


### Подробнее о шрифтах: правило @font-face (эт-правило)

Технически подключение веб-шрифтов производится с помощью CSS-правила @font-face



Обычно сначала указывают название шрифта в системе, чтобы браузер попытался найти его локально на компьютере пользователя, а затем указывают адрес файла шрифта в интернете.



### Тег time, дата и время для людей и машин (есть в прошлом курсе)

С помощью <time> можно описывать даты одновременно и для человека, и для машины, например:



### Теги figure и figcaption, картинки с подписями (есть в прошлом курсе)

(есть в прошлом курсе)

### Тег video, видео

* width и height задают ширину и высоту видео
* controls пустой атрибут, при наличии которого отображается панель управления видео
* preload задаёт режим предзагрузки видео, имеет 3 возможных значения:

none — не загружать ничего;

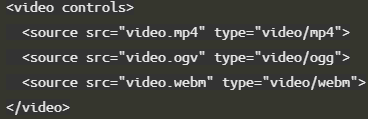
metadata — загрузить служебную мета-информацию (длительность, первый кадр и так далее);

auto — можно загрузить всё видео.

значение по умолчанию зависит от браузера

* src задаёт адрес видеофайла
* autoplay пустой атрибут, при наличии которого воспроизведение видео начинается автоматически
* poster задаёт адрес картинки-обложки, которая отображается, когда видео ещё не загрузилось или не воспроизводится

В текущий момент существует несколько форматов видео, каждый из которых хорошо поддерживается лишь некоторыми браузерами. оэтому мы должны в видео указывать адреса файлов во всех этих форматах (и конвертировать исходное видео в эти форматы, конечно). Делается это с помощью тегов <source>:



### Тег audio, аудио

* controls пустой атрибут, при наличии которого отображается панель управления проигрывателем
* preload задаёт режим предзагрузки аудио, имеет 3 возможных значения:

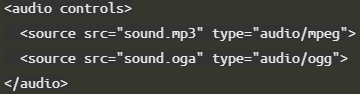
none — не загружать ничего;

metadata — загрузить служебную мета-информацию;

auto — можно загрузить весь файл.

* src задаёт адрес аудиофайла
* autoplay пустой атрибут, при наличии которого воспроизведение звука начинается автоматически

Мы должны так же, как и в случае с видео, перечислить адреса звуковых файлов в разных форматах с помощью тегов <source>:



# Знакомство с таблицами

### Добавляем строки

* <table> обозначает таблицу.
* <tr> расшифровывается как «table row», обозначает строку таблицы.
* <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>.

### Добавляем столбцы

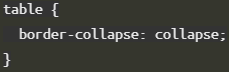
Для того чтобы добавить столбец в таблицу, надо в каждую строку <tr> добавить по ячейке <td>

### Задаём рамки с помощью CSS

У таблицы рамки отображаются, если у тега <table> задан атрибут border с ненулевым значением. Но более гибко управлять рамками лучше из CSS.

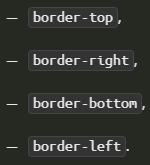
### Улучшаем отображение рамок

По умолчанию браузер рисует рамки таблицы и рамки отдельных ячеек раздельно. Чтобы избавиться от таких двойных рамок, используется CSS-свойство таблицы border-collapse:



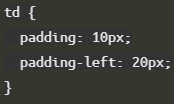
### Горизонтальные и вертикальные рамки

Для этого необходимо использовать не свойство border, которое задаёт рамки для всех сторон ячейки, а одно из свойств:



### Отступы внутри ячеек

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



<table cellpadding="5"> - так лучше не делать

### Отступы между ячейками

Отступы между ячейками не работают с border-collapse: collapse, что достаточно логично, ведь рамки ячеек в этом режиме «склеены».

Отступы между ячейками можно задать:

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

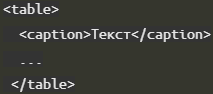
Отметим, что свойство border-spacing задаётся для таблицы, в отличие от padding, которое задаётся для ячеек.

### Ячейки-заголовки

Предусмотрен специальный тег <th>, который расшифровывается как «table header» и обозначает ячейку-заголовок. Тег <th> аналогичен <td>.

### Заголовок таблицы

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



Тег заголовка идёт первым внутри таблицы, но с помощью CSS можно переместить заголовок таблицы в любое место:

* По вертикали заголовок таблицы перемещается CSS-свойством caption-side со значениями top и bottom
* По горизонтали заголовок таблицы выравнивается CSS-свойством text-align со значениями left, right и center

### Объединяем ячейки в строках colspan

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 как бы «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».

### Объединяем ячейки в столбцах rowspan

Oсуществляется с помощью атрибута rowspan у тега <td> или <th>. Когда вы задаёте ячейке атрибут rowspan со значением 2, то ячейка как бы «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.

### Комбо: colspan + rowspan

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

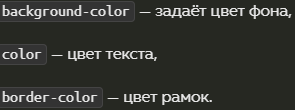
### Выравнивание содержимого в ячейках

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.

### Добавим цвета в таблицу

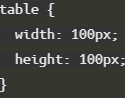
Таблицы можно раскрашивать, задавая цвет фона ячеек, цвет текста в ячейках, а также цвет рамок.



### Задаём размеры таблицы

По умолчанию ширина и высота таблицы зависит от содержимого и отступов внутри ячеек. Чем меньше содержимого, тем меньше размеры таблицы.

С помощью CSS можно управлять размерами таблицы. Также размерами можно управлять с помощью атрибутов таблицы, но нежелательно.



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;.

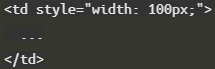
Важное замечание. Проценты при задании высоты обычно не работают.

### Задаём размеры отдельных ячеек и столбцов

Есть два варианта добавления стилей ячейкам:

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

Пример второго варианта:



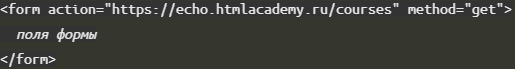
Обычно размеры ячеек прописывают, когда надо вручную установить ширину столбцов таблицы: для этого достаточно задать ширину для каждой ячейки из первой строки.

# Формы. Часть 1

### Понятие формы

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.



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:



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

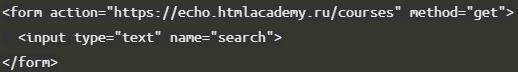
* Метод post посылает данные в теле HTTP-запроса и используется, когда нужно отправить много данных и ссылка на результат обработки этих данных не нужна. Например, при редактировании личного профиля.

### Текстовое 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:



Имя поля нужно, чтобы правильно обработать данные на **сервере**. Имя должно быть уникально в пределах формы.

### Идентификатор и значение по умолчанию

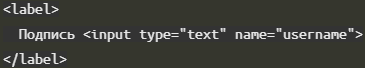
Атрибут id **поля ввода** обозначает идентификатор. Он должен быть уникальным на всей странице. Идентификаторы используют для повышения удобства работы с формой, например, создают подписи, связанные с мелкими полями. При нажатии на такие подписи активируется связанное поле. И это удобно, так как по большой подписи попасть легче, чем по маленькому полю. Также идентификаторы используют в JavaScript.

Идентификатор в отличие от имени поля не передаётся на сервер. Атрибут **value** задаёт значение поля ввода по умолчанию.

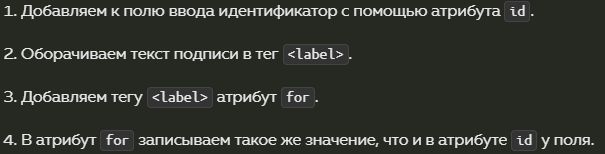
### Подпись для поля ввода

<label> связывает текст и поле ввода логически. А ещё если нажать на текст в такой подписи, то курсор переместится в соответствующее поле.

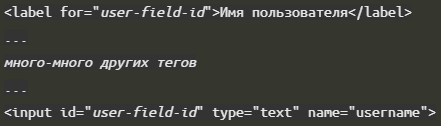
Первый способ создать подпись — просто обернуть текст подписи и тег поля в тег <label>, вот так:



### Связываем подпись и поле по id



Пример:



### Поле для ввода пароля

Нужно просто изменить значение атрибута type на password.

### Кнопка отправки формы

Кнопка 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.

### 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 по-другому. Текст, расположенный между открывающим и закрывающим тегом <textarea> и является значением по умолчанию.

### Чекбокс или «галочка»

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

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

### Переключатель или «радиобаттон»

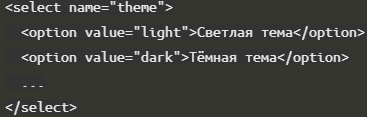
Поле-переключатель — это тег <input> с типом radio.

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, id тоже разыне.

Таким образом, атрибут value является для переключателей обязательным. Браузер отправляет на сервер значение value выбранного переключателя. Checked ставится у выбранного по умолчанию

### Раскрывающийся список или «селект»

Позволяет выбрать один вариант ответа из нескольких. Создаётся с помощью парного тега <select>, у которого есть знакомые атрибуты name и id. Варианты ответов задаются с помощью парных тегов <option>:



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

### «Мультиселект»

Селект, в котором можно выбрать не один, а несколько вариантов. Нужно добавить к тегу <select> атрибут multiple. 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.

### 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. Если вы хотите отправить на сервер файл, данные из формы эффективнее всего передавать по частям. За это и отвечает значение multipart/form-data.

### Скрытое поле

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

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

# Формы. Часть 2

### Сброс введенных значений

Кнопка, которая сбрасывает введённые значения и возвращает изначально установленные по умолчанию. Это поле ввода с типом reset.



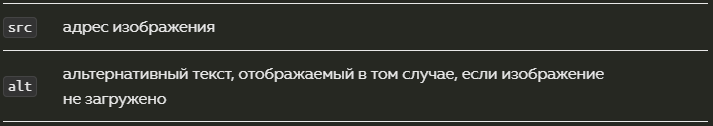
### Простая кнопка

Существуют и «просто кнопки». При нажатии на такую кнопку никаких действий не происходит, а все необходимые действия обычно задаются при помощи JavaScript.



### Кнопка-изображение

В качестве кнопки отправки формы можно использовать изображение. Для этого у тега input нужно указать тип image.



Кнопка-изображение работает аналогично кнопке submit, но на сервер дополнительно передаются координаты точки, по которой был произведен щелчок.

### Альтернативный способ задания кнопок

Помимо тега <input> для добавления кнопок можно использовать тег <button>. Он расширяет возможности создания кнопок.

Внутри тега <button> можно размещать любые HTML-элементы, в том числе изображения. Например:



Если в атрибуте type тега <button> указать значение **submit** (по умолчанию) или **reset**, то кнопка будет отправлять данные на сервер или сбрасывать введенные значения. Значение type="**button**" избавит кнопку от всей изначальной функциональности.

### Обязательные поля

Чтобы указать, что поле обязательно для заполнения, нужно добавить ему пустой атрибут required:



### Поле выбора даты

Для выбора даты из календаря существует новый тип поля ввода — date. При клике на данное поле в форме всплывает календарик.

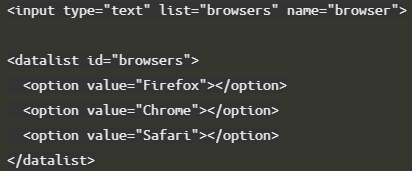
### Поле выбора времени

иногда нужно указывать дату в других форматах, и для этого существуют дополнительные «временные» типы полей, например, time для выбора времени.



### Список возможных значений

Для текстовых полей можно заранее определить список возможных значений, которые отображаются, когда вы начинаете вводить текст в поле. Для этого существует специальный тег <datalist>.



Связывание текстового поля и списка осуществляется при помощи атрибута list у тега input — значение list должно быть таким же, как значение атрибута id у списка.

### Поле ввода числового значения

Для ввода числовых значений существует специальный тип поля ввода number. Рядом с полем браузер автоматически подставляет две стрелочки для увеличения и уменьшения числового значения.



При помощи вспомогательных атрибутов min и max можно установить верхнюю и нижнюю границу допустимых значений. А атрибут step устанавливает величину шага изменения значения.

### Поле поиска



В некоторых браузерах внутри него появляется крестик для сброса введённого значения.

Закрыть

### Автофокус

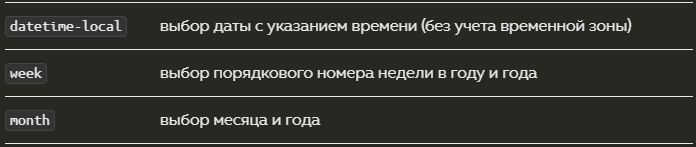
При загрузке страницы можно сообщить браузеру в какое поле установить курсор по умолчанию. Для этого используется пустой атрибут autofocus.



Такой атрибут должен быть только один на странице.

### Другие поля для ввода дат

Существуют и другие типы (type) полей для ввода дат:



### Выбор из диапазона

Такое поле выглядит как шкала с ползунком и позволяет выбрать число из некоторого интервала значений.



Атрибуты min и max устанавливают нижнюю и верхнюю границу допустимых значений. А атрибут step устанавливает величину шага изменения значения.

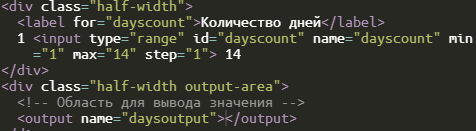
### Область для вывода результата

Тег <output> представляет собой область, куда выводятся какие-либо результаты вычислений, обычно полученные при помощи JavaScript.



Значение по умолчанию при этом можно не задавать, тогда область вывода будет пустой.



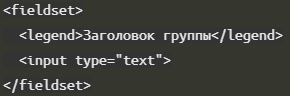


### Группировка полей формы

Когда формы становятся очень большими, возникает потребность зрительно отделить одни поля от других. Для группировки полей используется тег fieldset.



Также для каждой группы можно добавить её заголовок. Для этого внутрь тега fieldset надо поместить тег legend:



### Паттерны значений полей

Чтобы добавить автоматическую проверку формата номера в поле, используем атрибут pattern, в котором с помощью регулярного выражения опишем требуемый формат.





### Поле ввода телефона

Поле с типом tel. Так же можно воспользоваться pattern в этом поле.

### Подсказка при заполнении полей

У полей, в которые вводятся текстовые значения (textarea, разные типы input и так далее) есть возможность вывести подсказку. Для этого используется специальный атрибут placeholder:



Текст подсказки выводится внутри текстового поля, а при вводе значения — автоматически убирается.

### Поля ввода адресов сайтов и email

Есть два типа полей email и url, предназначенные для ввода электронной почты и адреса сайта. Особенностью этих полей является то, что они автоматически проверяют формат введённых данных.



Когда вы начинаете заполнять такое поле на мобильнике, там автоматически переключается раскладка клавиатуры.

### Поле выбора цвета

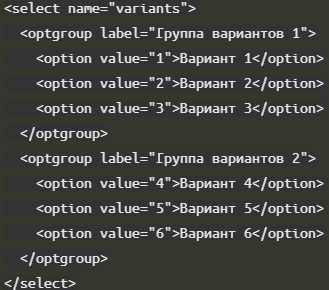
В HTML5 добавили новый тип color, предназначенный для полей выбора цвета. При клике на такое поле появляется окно с возможностью выбрать цвет из палитры.



### Группировка элементов списка

Так как список выбора может быть довольно большой, в теге select используем возможность объединять option в группы. Обычно это используется для большей наглядности и удобства поиска нужного варианта.

Для формирования группы используется тег optgroup. Атрибут label этого тега определяет заголовок группы.



Вложенность групп не ограничена

### Запрет редактирования полей

Есть два способа: использование атрибута readonly и использование атрибута disabled.



* Атрибут readonly не дает пользователю изменять поле (вводить новый текст, модифицировать существующий). Введенное значение можно выделить и скопировать. Данные из этого поля отправляются на сервер.
* Атрибут disabled не дает пользователю изменять поле (вводить новый текст, модифицировать существующий). Нельзя поставить фокус в это поле, введенное значение нельзя выделять и копировать. Данные из этого поля НЕ отправляются на сервер.

### Управление автозаполнением полей

Браузер может запоминать значения, вводимые в текстовые поля. При вводе первых букв текста выводится список сохранённых ранее значений, из которого можно выбрать подходящее. Параметрами автозаполнения можно управлять используя атрибут autocomplete.

Он может принимать два значения on и off. Первое включает автозаполнение, второе — отключает.

### Переключение между полями

При нажатии клавиши Tab браузер передает управление (фокус) от одного элемента к другому в том порядке, в котором они были объявлены на странице. Этим порядком можно управлять при помощи атрибута tabindex.



Закрыть

### localStorage

Эта технология относится к JavaScript. Суть localStorage или «локального хранилища» заключается в том, что в него можно записывать данные, которые будут сохраняться в браузере. Эти данные не исчезнут даже если вы закроете браузер и откроете его снова или уйдете со страницы и потом вернётесь на неё.

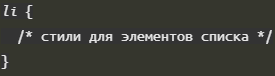
# Селекторы, часть 1

### Жизнь без селекторов, атрибут style

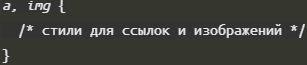


### Селекторы по тегам

С помощью селекторов по именам тегов можно задать стили для всех элементов списка, изображений, абзацев и так далее. Например:



**Одно правило может относиться сразу к нескольким селекторам**, в таком случае селекторы перечисляются через запятую:

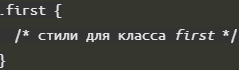


### Селекторы по классам

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



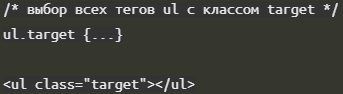
Этот атрибут особенный, так как в CSS существует возможность выбирать элементы по классу. Делается это с помощью такого селектора: .имя\_класса. Например:



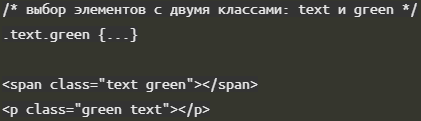
### Выбор определенного тега с определенным классом (базовый прием)

Можно, например, **выбрать 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.

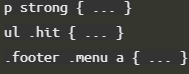
В селекторе по тегу и классу первым пишется название тега, а потом идёт класс:



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



### Контекстные селекторы (вложенность) (базовый прием)



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

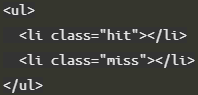
Например, селектор **.menu a** сработает для ссылки a только в том случае, если она расположена внутри элемента с классом .menu.

Читать их проще всего справа налево.



### Соседние селекторы (+) (базовый прием)

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.



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

Закрыть

### 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.

### Дочерние селекторы (базовый прием)

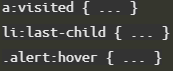
Потомком называются любые элементы, расположенные внутри родительского элемента. А дочерними элементами называются ближайшие потомки. Иногда необходимо задать стили только для дочерних элементов. Для этого существует дочерний селектор:



## Псевдоклассы

Псевдоклассы — это дополнения к обычным селекторам, которые делают их ещё точнее.

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



### first-child и last-child

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

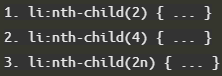


Этот селектор выберет последний элемент списка.

### Псевдокласс :nth-child

С помощью псевдокласса nth-child можно выбирать теги по порядковому номеру.

Синтаксис псевдокласса: селектор:nth-child(выражение). Выражением может быть число или формула. Например:



Полезно знать *Формула общего члена*

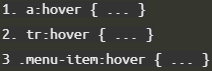
### :nth-child и контекстные селекторы

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



### Псевдокласс :hover

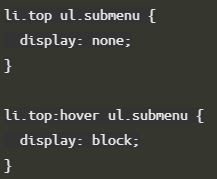
Этот псевдокласс позволяет выбрать элемент, когда на него наведён курсор мыши и кнопка мыши не нажата. Примеры:



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

### Динамические эффекты с помощью :hover

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:

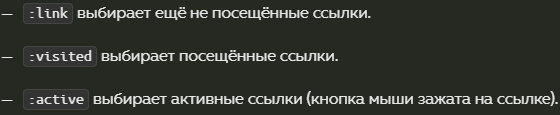


ервое правило прячет список-подменю. Второе правило гласит: «если на верхний пункт меню, в котором находится подменю, наведут курсор, то надо показать подменю». Вот так всё просто.

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.

### Псевдоклассы :link, :visited и :active

Познакомимся с псевдоклассами для ссылок.



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

### 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

### Селекторы атрибутов

Чаще всего такие селекторы используются при работе с формами, так как поля форм имеют атрибут type с разными значениями.

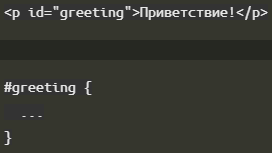
Селекторы атрибутов записываются с использованием квадратных скобок: элемент[атрибут]. Примеры селекторов:



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

### Селектор по id (для атрибута id)

селектор записывается с помощью символа #, например, #some-id.



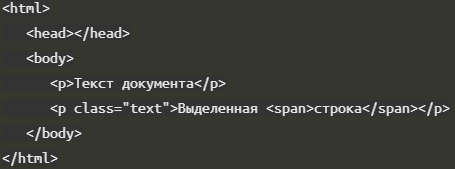
# Наследование и каскадирование

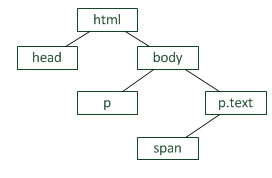
## Наследование

### 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.

Пример:





### 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.

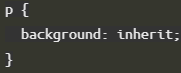
### Не все свойства наследуются

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

### Принудительное наследование inherit

Для каждого свойства может быть задано значение inherit.

Значение inherit может быть использовано в свойствах, которые обычно не наследуются или для усиления наследуемых значений.



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

## Каскадирование

### Что такое каскадирование

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.

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

* важность;
* специфичность;
* порядок исходного кода.

### Специфичность

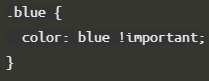
Простое объяснение специфичности звучит так: «*Чем меньшее количество элементов потенциально может выбрать селектор, тем он специфичнее*.»

### Селектор по id

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

### !important

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:

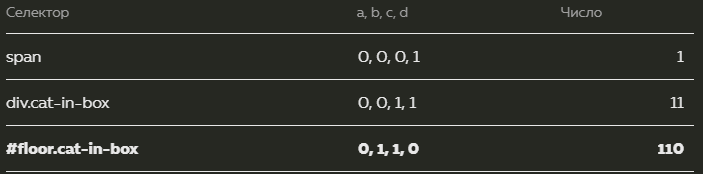


### Расчёт значения специфичности

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.

Пример:



### Перекрёстное наследование

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:



# Фоны, часть 1

### Cвойство background-color

Цвет фона можно задать с помощью CSS-свойства background-color.

### Свойство background-image

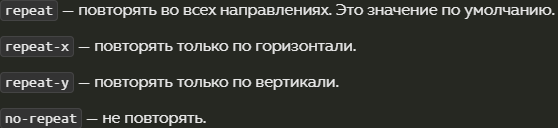
Фоновое изображение можно задать с помощью CSS-свойства background-image.



Элементу можно одновременно задавать и цвет фона, и фоновую картинку. В этом случае картинка будет отображаться поверх фонового цвета.

### Свойство background-repeat

По умолчанию фоновое изображение повторяется. Это хорошо заметно, когда оно меньше блока. Управлять этим поведением можно с помощью CSS-свойства background-repeat. У свойства 4 значения:



### Свойство background-position

Управляет расположением фонового изображения. Значение свойства состоит из двух частей, разделенных пробелом: x y.

* В качестве значения x можно использовать ключевые слова left, center, right, значения в процентах и в пикселях.
* В качестве значения y можно использовать ключевые слова top, center, bottom, значения в процентах и в пикселях.

 значит, отступ 50px от левого края и 100px от верхнего;

Когда фоновое изображение больше блока, то оно обрезается. Управлять тем, какую часть изображения будет видно, также можно с помощью свойства background-position.

Иногда для этого удобно использовать относительные значения (проценты), а иногда абсолютные (пиксели). Кстати, можно использовать не только положительные, но и отрицательные значения. А также комбинировать пиксели и проценты.

### Свойство background-attachment

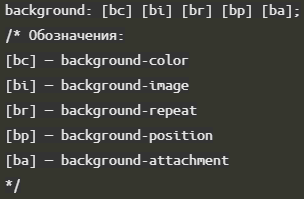
С помощью свойства background-attachment можно зафиксировать фон на месте и он не будет перемещаться при прокручивании.

Значение свойства:

* scroll — фон прокручивается вместе с содержимым. Это значение по умолчанию.
* fixed — фон не прокручивается, зафиксирован на одном месте.

### Свойство background

Также задать фон можно с помощью сокращенного свойства background, в котором через пробел перечисляются его компоненты:



Если какой-то компонент не указан, то берется значение по умолчанию.

### Формат JPEG

Этот формат хорошо подходит для фотографий и картин и плохо подходит для схем, чертежей, текста и графики. JPEG не поддерживает прозрачность, поэтому изображение в формате JPEG — это всегда прямоугольник. Используйте JPEG для размещения фото с наилучшим соотношением "размер файла"/"качество".

### Формат PNG-8

PNG-8 по характеристикам схож с форматом GIF. Он хорошо подходит для схем, чертежей, графиков и текста, а также изображений, где мало цветов. Максимальное количество цветов — 256. PNG-8, как и GIF, поддерживает прозрачность. Это означает, что пиксель либо полностью прозрачный, либо полностью непрозрачный.

Если нужно сделать изображение с прозрачностью, то лучше задавать обводку такую же, как цвет фона. В фотошопе при сохранении обводка называется «Matte». Без обводки картинка будет выглядеть «обкусанной». Если фон неоднородный (градиент, 3 блок в мини-браузере), то PNG-8 с обводкой будет смотреться плохо.

Чем PNG лучше GIF? PNG более современный и свободный формат, а GIF более старый и проприетарный (за его использование могут потребовать деньги). Уровень сжатия, качество и другие характеристики примерно одинаковые, поэтому лучше всегда использовать PNG.

### Формат PNG-24

Хорошо подходит и для схем-чертежей-графиков-текста, и для сложных многоцветных изображений, так как поддерживает практически неограниченное количество цветов.

Самый главный плюс PNG-24 — это полноценная поддержка полупрозрачности, которой нет ни в одном другом формате. Поэтому в веб-разработке при вёрстке сложных фонов, графических элементов со сложными тенями и так далее альтернативы PNG-24 просто нет.

На фотографиях обычно он уступает JPEG по размеру файла, но превосходит по качеству изображения. Но иногда разница по размеру файла настолько некритична, что можно использовать PNG.

### Формат GIF

Наверное, единственная причина использовать GIF заключается в том, что он поддерживает анимированные изображения.

Во всех остальных случаях используйте PNG или JPEG (если речь идёт о фотографиях и PNG даёт слишком тяжёлый файл).

### Несколько фонов

Рассмотрим старую надёжную технику создания нескольких фонов.

Суть техники заключается в том, что мы вкладываем элементы друг в друга и делаем их одинакового размера, а затем каждому элементу задаём свой фон. То есть каждый элемент служит одним слоем фона.

Фоны вложенных элементов перекрывают друг друга: чем глубже элемент, тем выше его фон.

Кстати, для удобства ширину лучше задавать внешнему элементу (так как все вложенные будут той же ширины), а высоту самому глубокому, так как он растянет по высоте всех своих родителей.

### Спрайты

Спрайт — это одно большое изображение, в котором содержится много маленьких, как бы карта изображений.

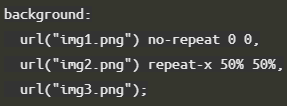
Спрайты используются, чтобы снизить количество запросов на сервер. Каждая маленькая картинка — это отдельный запрос, а чем меньше запросов, тем лучше. Поэтому маленькие картинки «склеивают» в одну большую.

Части спрайта отображают в элементах с небольшими размерами. Такому элементу задают картинку-спрайт в качестве фона и смещают её таким образом, чтобы была видна нужная её часть.

В спрайты обычно объединяют иконки и различные мелкие декоративные изображения.

### Множественные фоновые изображения

Это очень просто. Берёте знакомое свойство background и перечисляете в нем картинки через запятую. Например:



В этом примере у элемента будет 3 фоновых картинки. Причем картинка img3.png будет в нижнем слое, а img1.png в верхнем.

# Селекторы, часть 2

### Объединение селекторов(базовый прием)

Когда нужно выбрать элементы, которые одновременно удовлетворяют сразу нескольким условиям. В CSS есть запись, которая фактически выполняет операцию логического умножения, «И».



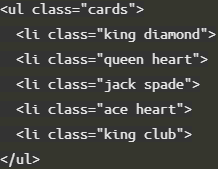
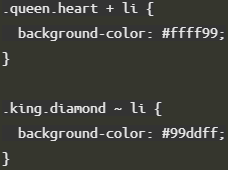
Стили будут применяться ко всем элементам, которые одновременно имеют класс class1 и class2.



### Cелектор последующих элементов (базовый прием)

Стили применятся к элементу, подходящему под селектор2, только если перед ним расположен элемент, подходящий под селектор1. Оба элемента должны принадлежать одному родителю. Отличие от соседнего селектора состоит в том, что между элементами селектор1 и селектор2 могут находиться другие элементы. Поэтому будем называть селектор1 ~ селектор2 селектором следующих элементов.

Пример:

В первом случае выделится одна карта, расположенная сразу за червовой дамой, то есть пиковый валет.

Во втором случае выделятся все карты, которые стоят за бубновым королем, то есть червовая дама, пиковый валет, червовый туз и трефовый король.

## Псевдлклассы

### Псевдокласс :not

С его помощью можно выбрать элементы, которые НЕ содержат указанный селектор:



Этот селектор выберет все теги <li>, НЕ являющиеся последними в их родителе.

В качестве селектора могут указываться псевдоклассы, теги, идентификаторы, классы и селекторы атрибутов.

Не сработает:

* объединение селекторов: например, li:not(.heart.jack) — некорректный селектор;
* псевдоэлементы: li:not(::after) — неправильная запись (подробнее о псевдоэлементах рассказано далее в курсе);
* селекторы-потомки, групповые селекторы или комбинации: например, нельзя писать li:not(a span) или li:not(a + span)

### Комбинируем :not

Отрицающий селектор :not, как и любые другие селекторы, можно комбинировать с другими. Например:



Выберет все теги <li>, которые НЕ являются первыми и последними в их родителе.

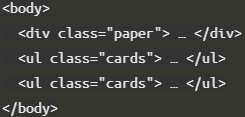
### Псевдокласс :nth-last-child

В отличие от псевдокласса :nth-child, отсчет ведется не от первого элемента, а от последнего. Вот и все различия.

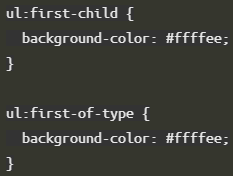
### Псевдокласс :first-of-type (new)

Псевдокласс :first-of-type очень похож на :first-child. Он выбирает первый дочерний элемент родителя, только с учетом типа элементов.

Пример:



Сравним две записи:



Верхний селектор выбирает первый дочерний элемент в родителе, причем этот элемент должен быть ul. В нашем случае не выберется ничего, потому что первым дочерним элементом body является div.

Нижний селектор выбирает первый ul среди всех дочерних ul в своем родителе. В нашем случае будет выбрана первая строка с картами.

### Псевдокласс :last-of-type

Псевдокласс :last-of-type работает аналогично, только выбирает последний дочерний элемент родителя с учетом типа.

### Псевдокласс :nth-of-type

Псевдокласс :nth-of-type работает почти так же, как и :nth-child. Разница заключается в том, что он учитывает тип элемента.

Пример:



* Верхний cелектор выберет второй по счёту дочерний элемент и этот элемент должен быть ul.
* А нижний селектор выберет второй по счету **ИМЕННО ul** среди дочерних ul.

### Псевдокласс :nth-last-of-type

Элементы выбираются по их расположению, отсчет ведется от конца, **учитывается тип элемента**.

Пример записи:



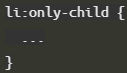
### Псевдокласс :empty

Псевдокласс :empty, выбирает только те теги, у которых нет дочерних элементов (в том числе текстовых узлов). Учтите, что даже переход на новую строку считается текстовым узлом, помните об этом в процессе проектирования структуры страницы.

 В данном случае выберутся все пустые элементы ul.

### Псевдокласс :only-child

Псевдокласс :only-child пригодится, когда нужно прописать индивидуальные стили для элемента, который является единственным дочерним элементом внутри родительского контейнера.

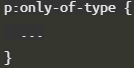
 Сработает, когда этот <li> в списке единственный.

Cелектор :only-child эквивалентен селектору elem:last-child:first-child.

### Псевдокласс :only-of-type

Псевдокласс :only-of-type работает почти так же, как и :only-child. Отличие состоит в том, что он учитывает тип элемента.

Пример:

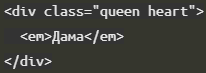
 В данном случае стили будут применены к элементу p, если это единственный p **внутри своего родителя**.

## Псевдоэлементы

### Псевдоэлемент ::before

Псевдоэлемент before позволяет с помощью CSS добавить псевдотег внутрь другого элемента и оформить его. Cодержимое псевдотега задаётся с помощью свойства content.

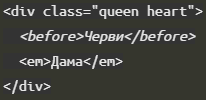
Например, у нас есть такой HTML:



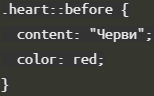
Добавим CSS-правило с необычным селектором с двойным двоеточием:



И HTML изменится вот так:

 Но! Исходный HTML-код не изменится, тег <before> не попадёт в код страницы, а будет «виртуально» существовать где-то в браузере. Поэтому мы и используем приставку псевдо.

Ведёт себя псевдотег так же, как обычный <span> с текстом. Ему можно задавать дополнительные стили, например:

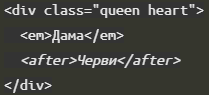
 Чтобы псевдоэлемент появился, ему необходимо задать свойство content. Достаточно даже пустой строки в значении свойства — content: "";.

### Псевдоэлемент ::after

Псевдоэлемент after аналогичен before. Отличие заключается в том, что он добавляет псевдотег не в начало, а в конец элемента. Например:



Даст такой результат:



### Позиционирование псевдоэлементов

Вы можете задавать псевдоэлементам любые CSS-свойства. Можно менять тип элемента, задавать ему позиционирование, отступы, фон и так далее.

Этими возможностями особенно часто пользуются при создании различных декоративных эффектов. Это очень удобно.

Во-первых, не нужно добавлять лишний тег под каждую очередную мелкую иконку.

Во-вторых, можно управлять этими иконками/декоративными элементами только с помощью CSS, что дает возможность создания динамических эффектов.

### Фон для псевдоэлементов

Можно задавать

### Псевдоэлементы ::first-line и ::first-letter

Псевдоэлемент first-line задает стиль первой строки форматированного текста. Длина этой строки зависит от многих факторов, таких как используемый шрифт, размер окна браузера, ширина блока, языка и так далее. В правилах стиля допустимо использовать только свойства, относящиеся к шрифту, изменению цвета текста и фона. Пример:



Аналогично псевдоэлемент **first-letter** определяет стиль первого символа в тексте элемента, к которому добавляется. К этому псевдоэлементу могут применяться только стилевые свойства, связанные со свойствами шрифта, полями, отступами, границами, цветом и фоном.

