# Класс

class — атрибут, который помогает различать элементы. Его можно присвоить нескольким элементам и обращаться к ним по имени класса. Это удобно при работе со стилями и скриптами.

Скопировать кодHTML

<p class="my-paragraph">Параграф с классом my-paragraph</p>

Одному элементу можно присвоить несколько классов. Один класс может задавать параметры всех кнопок, а другой покрасить каждую в свой цвет — в зависимости от контекста использования.

### HTML

Скопировать кодHTML

*<!-- Кнопка отправки формы: -->*

<button class="project-button project-button\_type\_submit">Отправить</button>

### CSS

Скопировать кодCSS

*/\* Все кнопки будут 100 на 30 пикселей, со скруглёнными углами и надписью заглавными буквами \*/*

.project-button {

width: 100px;

height: 30px;

border-radius: 4px;

text-transform: uppercase;

}

*/\* Кнопка отправки формы будет синего цвета стали с белой надписью \*/*

.project-button\_type\_submit {

background-color: steelblue;

color: white;

}

Принято стилизовать элементы по имени класса, а не тега. Так с ними удобнее работать в CSS.

# Селекторы

Чтобы стилизовать элементы, их нужно выбрать. Для этого существуют селекторы. С их помощью можно сразу выбрать целую группу элементов (параграфы, ссылки, классы) и задать всей этой группе определённые параметры: размер, шрифт, цвет и что только не.

Самые распространённые селекторы:

1. По имени тега:

Скопировать кодCSS

*/\* выбираем все элементы с тегом p на странице \*/*

p {

color: red;

}

1. По имени класса:

Скопировать кодCSS

*/\* выбираем все элементы с классом my-class \*/*

.my-class {

color: red;

}

1. Селекторы вложенности. Их используют, когда нужно выбрать все дочерние элементы по отношению к родительскому элементу:

Скопировать кодCSS

*/\* все p внутри div \*/*

div p {

color: red;

}

*/\* все элементы с классом my-class внутри div \*/*

div .my-class {

color: red;

}

1. Комбинированные селекторы. Их используют, когда нужно выбрать специфичный элемент, например, одновременно с двумя классами или ссылку с определенным параметром:

Скопировать кодCSS

*/\* ссылка с классом my-link \*/*

a.my-link {

color: red;

}

*/\* элемент с классами my-button и my-button\_large одновременно \*/*

.my-button.my-button\_large {

color: red;

}

Для повседневной работы этого хватит. Если хотите разобраться во всех тонкостях выбора элементов, рекомендуем игру для изучения селекторов: <https://flukeout.github.io/> (на английском).

}

**Элементы в потоке**

Все HTML-элементы следуют друг за другом: либо находятся на одной строке с соседом, либо начинают новую строку. Это зависит от типа элемента — условно одного из двух:

1. **Строчные элементы.** Занимают ровно столько места, сколько нужно для их контента, располагаются на одной строке с другими строчными элементами и не реагируют на изменение ширины и высоты.

Так ссылка внутри параграфа находится на одной строке с текстом этого параграфа:

Скопировать кодHTML

<p>В этом абзаце есть <a href="http://ya.ru">ссылка</a></p>

1. **Блочные элементы.** По умолчанию занимают всю ширину своего родителя и располагаются на новой строке.

Например, абзацы, контейнеры div и заголовки.

Скопировать кодHTML

*<!-- Все эти элементы займут ширину родителя целиком и будут начинаться с новой строки -->*

<h1>Самый главный заголовок</h1>

<p>Это абзац текста.</p>

<div></div>

Свойство display определяет поведение элементов. Строчные могут стать блочными и наоборот:

Скопировать кодCSS

a {

display: block; */\* теперь ссылка занимает всю ширину родителя и расположена на новой строке \*/*

}

p {

display: inline; */\* абзац займёт ровно те размеры, которые нужны для контента внутри, и окажется на одной строке с другими строчными элементами \*/*

}

Когда блочный элемент становится строчным, ему нельзя задать ширину и высоту. Чтобы он вёл себя как строчный, но не игнорировал свойства width и height, придумали комбинированный тип — блочно-строчный:

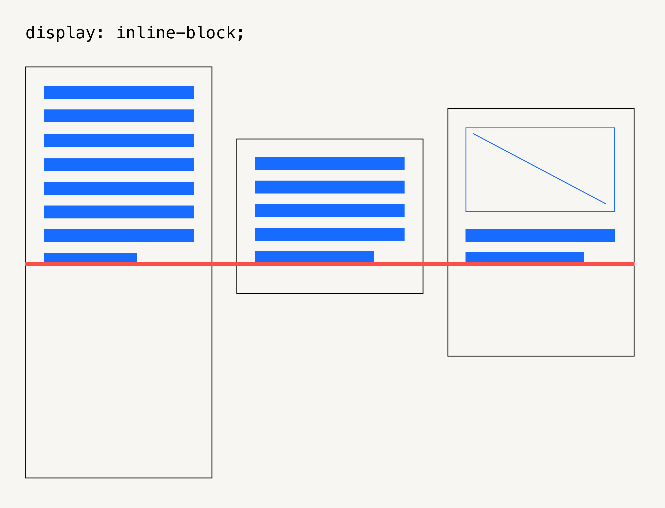
Скопировать кодCSS

p {

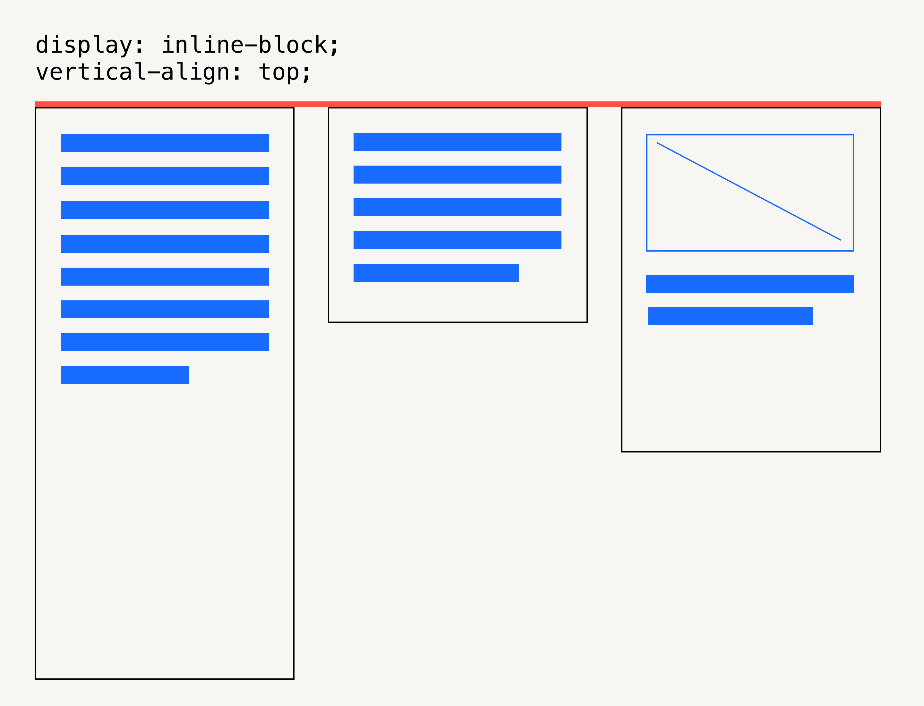
display: inline-block; */\* абзац окажется на одной строке со строчными и блочно-строчными элементами, но ему можно задать ширину и высоту \*/*

}

Блочно-строчные элементы выравниваются по последней строке текста внутри них:



Поэтому вертикальным выравниванием нужно управлять, например, задавать выравнивание по верхней строчке:



Если два блочно-строчных элемента описаны в коде друг под другом, в документе их будет разделять пробел.

Скопировать кодHTML

<p>Первый абзац</p>

<p>Второй абзац</p>

Для этих двух абзацев можно написать такой CSS:

Скопировать кодCSS

p {

display: inline-block;

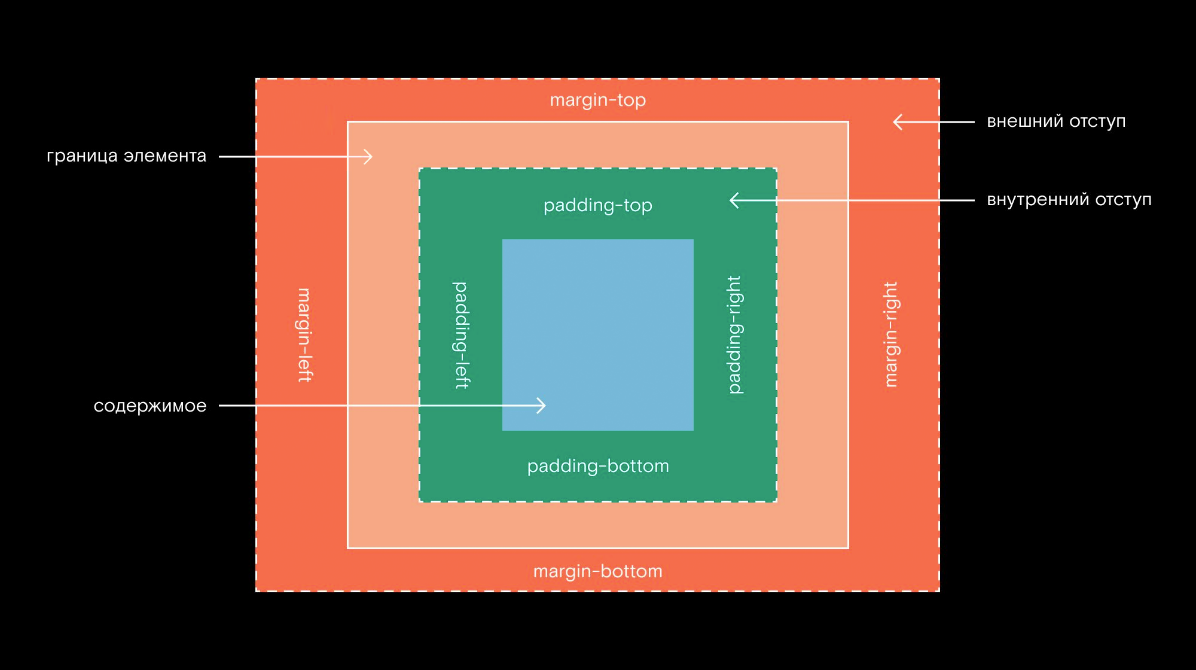
vertical-align: top;

}

Абзацы встанут в одну строку, но между ними появится пробел.

**Блочная модель документа**

Блок в HTML состоит из нескольких частей: размеров, границы, внешних и внутренних отступов. Это сочетание называют блочной моделью документа. Чаще всего её описывают изображением:



**Размеры блока** задают шириной и высотой. Можно использовать абсолютные (пиксели) и относительные (сотые доли) величины:

* % — от размеров родительского блока,
* vh — от высоты окна браузера,
* vw — от ширины окна браузера.

Скопировать кодCSS

div {

width: 300px;

height: 200px;

}

**Граница** — группа свойств со множеством параметров. Основные:

border-width— толщина границы,

border-style— стиль границы,

border-color — её цвет.

Для краткости эти свойства пишут в одну строку — друг за другом, через пробел.

Скопировать кодCSS

div {

border: 1px solid red; */\* сплошная граница в 1 пиксель красного цвета \*/*

}

**Внешние отступы** margin, или поля — пространство между элементом и его соседями:

Скопировать кодCSS

div {

margin-top: 20px; */\* расстояние до верхнего соседа \*/*

margin-right: 20px; */\* расстояние до правого соседа \*/*

margin-bottom: 20px; */\* расстояние до соседа снизу \*/*

margin-left: 20px; */\* расстояние до соседа слева \*/*

}

*/\* Сокращённая запись того же самого \*/*

div {

margin: 20px;

}

Если блочному элементу установить margin-left и margin-right в значении auto, блок центрируется по горизонтали. Для вертикального выравнивания это не сработает.

Скопировать кодCSS

div {

margin-left: auto;

margin-right: auto;

}

*/\* Сокращённая запись \*/*

div {

margin: 0 auto;

}

Внутренний отступ padding отодвигает контент внутри блока от границы. У padding нет специального значения auto.

Скопировать кодCSS

div {

padding-top: 20px; */\* расстояние от верхней границы до содержимого \*/*

padding-right: 20px; */\* расстояние от правой границы до содержимого \*/*

padding-bottom: 20px; */\* расстояние от нижней границы до содержимого \*/*

padding-left: 20px; */\* расстояние от левой границы до содержимого \*/*

}

*/\* Сокращённая запись \*/*

div {

padding: 20px;

}

Когда мы описываем границу или внутренний отступ, размер блока увеличивается. Добавляется толщина границы, а внутренние отступы «раздувают» блок, чтобы создать пространство вокруг контента.

Такое поведение меняют свойством box-sizing (со значением content-box по умолчанию). Если изменить значение на border-box, границы будут отрисованы внутри блока, а внутренние отступы не повлияют на его размеры:

Скопировать кодCSS

div {

box-sizing: border-box; */\* задали box-sizing \*/*

width: 300px; */\* ширина блока \*/*

height: 200px; */\* высота блока \*/*

border: 2px solid red; */\* граница не влияет на размер \*/*

padding: 20px; */\* отступ не растягивает блок \*/*

}

# Свойства текста и шрифта

А теперь давайте повторим базовые CSS-свойства (собрали только самые основные, вообще их гораздо больше). Их важно знать, поскольку они станут основой для более сложных концепций.

## Текст

* color — цвет текста. Можно задать ключевыми словами, HEX-кодом, в RGB- или RGBA-палитре.
* text-align — выравнивание текста внутри блока: по левому краю left, по правому right, по центру center.
* text-decoration — оформление текста. Часто используют подчеркнутый текст underline и неоформленный none.
* text-transform — перевод текста в верхний uppercase или нижний lowercase регистр.

## Шрифт

* font-family — семейство шрифтов. Работает только со шрифтами, которые есть на устройстве пользователя. Можно перечислять шрифты через запятую, чтобы браузер искал среди них установленный.
* font-size — размер шрифта.
* font-weight — начертание: полужирное bold, обычное normal и тонкое lighter.

**Свойства фона**

Эти свойства фона вам обязательно пригодятся, поэтому напомним, что там к чему:

background-color — цвет фона. Можно задать словами, HEX-кодом, в RGB- или RGBA-палитре.

background-image — фоновое изображение. Ссылка на ресурс задаётся внутри параметра url():

Скопировать кодCSS

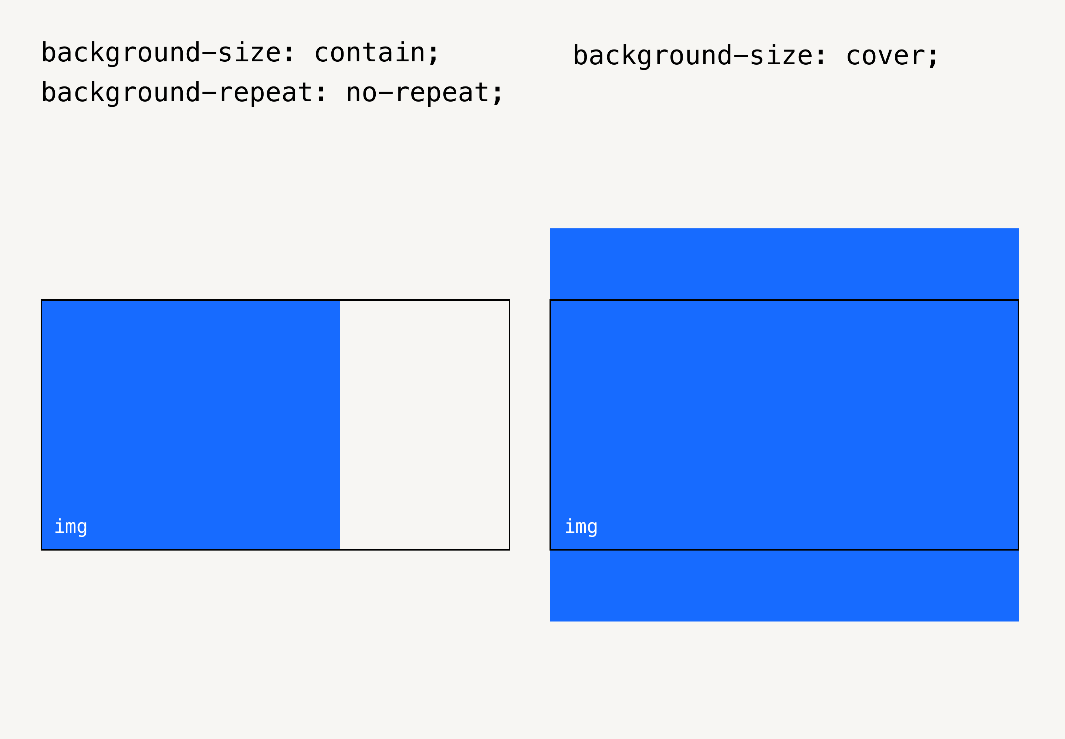
div {

background-image: url(путь\_к\_изображению);

}

background-size — размер фона. Часто используют значения:

* contain — фоновое изображение целиком вписывается в блок;
* cover — фон накрывает блок.



Свойство contain умещает изображение целиком, но оставляет поле. А свойство cover — наоборот, не оставляет полей, но обрезает изображение.

background-position — расположение фона по вертикали и горизонтали. Можно задать его позицию словами:

* left, center и right для горизонтального выравнивания;
* top, center и bottom для вертикального.

Скопировать кодCSS

div {

background-position: center top; */\* изображение центрировано по горизонтали,*

*верхний край фона прижат к верхней границе \*/*

}

### Как писать комментарии

Комментарии заключают в специальные конструкции, поэтому браузер их не видит.

В HTML комментарии заключают между <!-- и -->:

Скопировать кодHTML

*<!-- Блок со слайдером -->*

В CSS и JavaScript — между /\* и \*/:

Скопировать кодCSS

*/\* Общие стили для body \*/*

Комментарий может располагаться на одной строке или на нескольких:

Скопировать кодHTML

*<!-- Когда получим картинки от заказчика,*

*здесь нужно заменить заглушки в img -->*

Скопировать кодCSS

*/\* Временные стили для этого блока,*

*шрифты будут другие,*

*напомнить дизайнеру подобрать подходящие \*/*

В JavaScript есть ещё и однострочные комментарии. Они начинаются с двух слешей, а заканчиваются вместе со строкой.

Скопировать кодJAVASCRIPT

*// Объявляем функцию-обработчик события под кодом, меняющим цвет.*

Комментарии — полезная вещь, но не стоит ими злоупотреблять, иначе код станет трудным для восприятия. Со временем вы научитесь оставлять ровно столько комментариев, сколько необходимо, и именно там, где это нужно.

# О браузерных стилях

Когда вы пишете код страницы, браузер сам назначает элементам стили. Например, делает заголовок первого уровня больше последующих и оставляет отступы вокруг контента. Но как убедиться, что во всех браузерах всё одинаково? Никак — если дополнительно кое-что не поменять.

**Вариант 1**

Вручную переписать все основные стили. Это самый безошибочный, хоть и трудоёмкий подход: в Яндексе делают именно так. Такой способ — часть методологии БЭМ, которую мы начнём изучать уже в этом спринте.

**Вариант 2**

Сбросить всё до отправной точки — сделать элементы такими, будто браузер не задавал им собственные параметры. Свои стили придётся писать с нуля.

Этот подход предложил Эрик Мейер — создатель файла reset.css. Когда мы подключаем этот файл, большинство текстовых элементов страницы становятся одинаковыми, а затем мы сами определяем их стили.

Способ удобный, но его часто критикуют: reset.css очень жёстко обходится с элементами, нивелируя их особенности. Это может вызвать неожиданное поведение.

Всегда подключайте reset.css в коде выше собственных стилей, иначе стили из reset.css окажутся приоритетнее и заменят ваши:

Скопировать кодHTML

<head>

*<!-- метатеги и title -->*

<link rel="stylesheet" href="reset.css"> *<!-- Сначала reset.css -->*

<link rel="stylesheet" href="style.css"> *<!-- Потом свои стили -->*

</head>

Reset.css можно скопировать на компьютер или подключить из интернета: <https://meyerweb.com/eric/tools/css/reset/>

**Вариант 3**

Нормализовать стили — убрать различия отображения в разных браузерах. Для этого Николас Галлахер создал файл normalize.css, который описывает одинаковое поведение для всех браузеров, а не убирает отличия между элементами. Этот подход мы и будем использовать в учёбе.

Подключайте normalize.css выше своих стилей, но будьте внимательны. У некоторых элементов — таких как заголовки, параграфы или пункты списка — сохранятся стили, которые вы не писали. Их придётся вручную переписывать в CSS.

Файл normalize.css удобен тем, что его можно не использовать целиком, — он разделён на блоки. Стили текста собраны в одну группу, стили форм — в другую. Можно удалить стили для элементов, которых нет и не будет в проекте, и сайт будет загружаться быстрее.

Всегда подключайте normalize.css в коде выше собственных стилей, иначе будет как с reset.css: они «перекроют» ваши.

Скопировать кодHTML

<head>

*<!-- метатеги и title -->*

<link rel="stylesheet" href="normalize.css"> *<!-- Сначала normalize.css -->*

<link rel="stylesheet" href="style.css">*<!-- Потом свои стили -->*

</head>

Normalize.css можно сохранить на компьютере или подключить в интернете: <https://necolas.github.io/normalize.css/>