**CSS**

Если HTML сообщает браузеру о содержании веб-страницы, то CSS — о том, как её элементы выглядят и где расположены. Например, вы хотите сделать текст розовым и поместить в центр экрана.

Описание внешнего вида элемента называется стиль. CSS — это язык стилей.

К HTML-коду вашего проекта подключён файл стилей. В нём есть какие-то наработки и набор комментариев к ним. Комментарии в коде выделены серым, обрамлены специальными символами. Браузер не читает их — они для вас.

Разберёмся в этом CSS-коде и поработаем над внешним видом кнопки.

Цвета фона:

<https://www.w3schools.com/Colors/colors_names.asp>

**JavaScript**

CSS позволил создать визуальный образ кнопки, но она не работает. Включить её может JavaScript. Этот язык вдыхает жизнь в элементы веб-страниц, управляя их поведением.

Код на JavaScript называется «скрипт», в переводе «сценарий». Когда скрипт запускается, браузер совершает операции по определённому сценарию.

Загляните в 13 строку index.html, к странице теперь подключен файл script.js. Там уже что-то написано, осталось самое приятное — сделать так, чтобы клик по кнопке заработал.

Кнопка срабатывает, но отменить свой выбор нельзя: лайкнешь — живи с этим. Дадим пользователю возможность снять отметку Like.

Cкажите браузеру, что делать при клике по уже нажатой кнопке. Нужно убрать метку clicked и сказать, что состояние isClicked становится false. Для этого добавьте в файл script.js на строчку 10 такой код:

СкопироватькодJAVASCRIPT

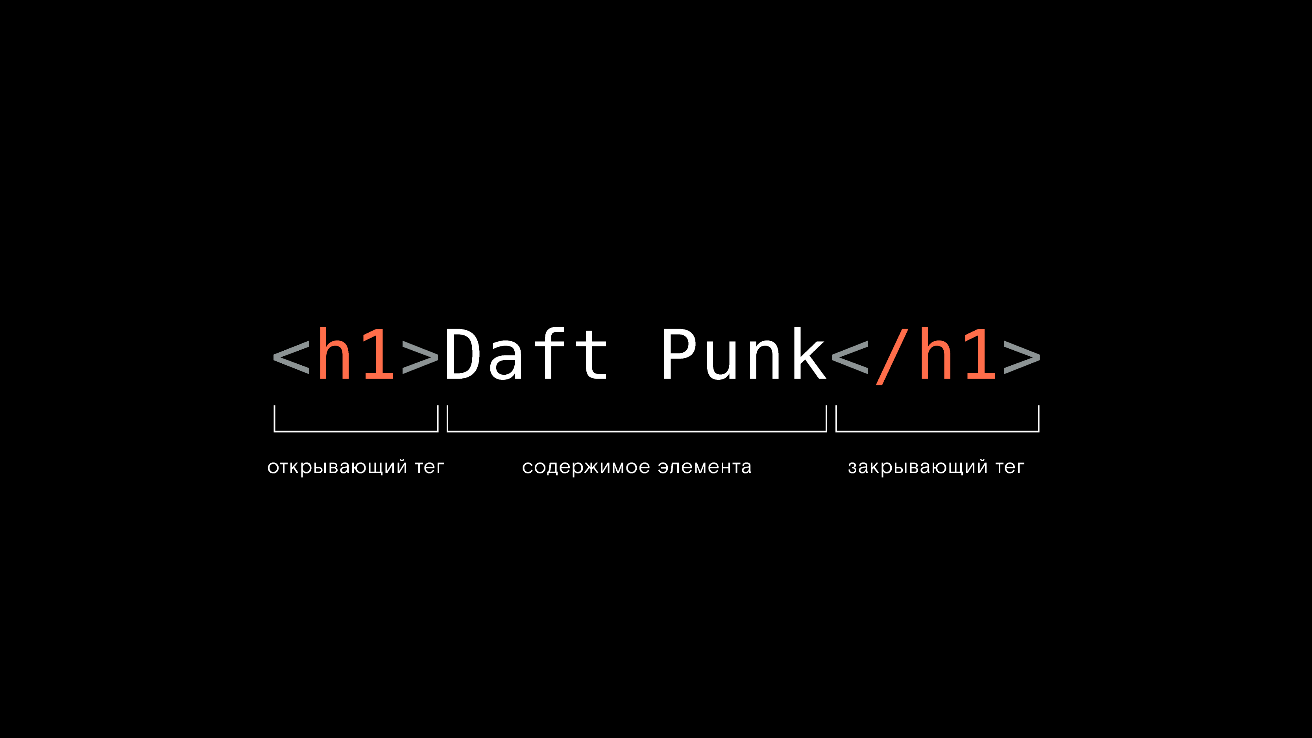
button.classList = '';

isClicked = false;

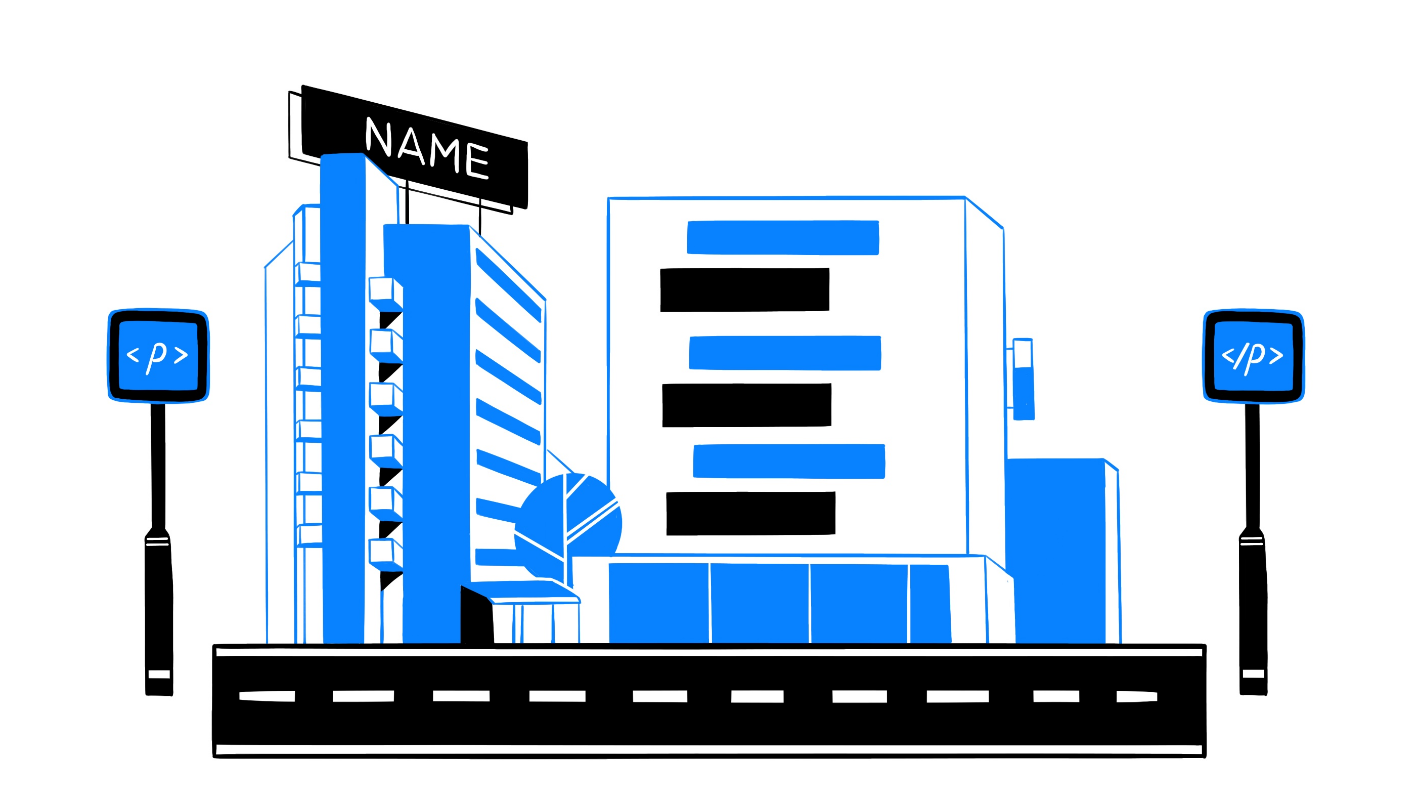
# Теги HTML

Начнём подробнее изучать HTML. Этот язык создаёт каркас сайта. Название расшифровывается как HyperTextMarkupLanguage.

Для размещения на странице видимый контент (текст, изображения, видео) заключают в управляющие конструкции — теги. Такие «бирки» указывают браузеру, как отобразить то, что в них «обёрнуто».



Обычно HTML-элемент состоит из тегов и размещённого между ними содержания. У любого HTML-тега есть имя и угловые скобки. Например, тег самого главного заголовка носит имя h1 (heading 1, «заголовок первого уровня»).



Теги расставляются как дорожные знаки с названием города. На въезде в город знак объявляет его имя, а на выезде перечеркнутое имя сообщает, что город закончился. Так и в начале HTML-элемента ставят открывающий тег с именем тега, а в конце — закрывающий тег, где имя будто перечёркнуто косой чертой. Созданный элемент называют по имени тега.

# Заголовки

В HTML существует шесть тегов заголовков разного размера: h1, h2, h3, h4, h5, h6. Идея заимствована из книг, где есть название на форзаце, заголовки частей (h2), разделов и более мелких глав.

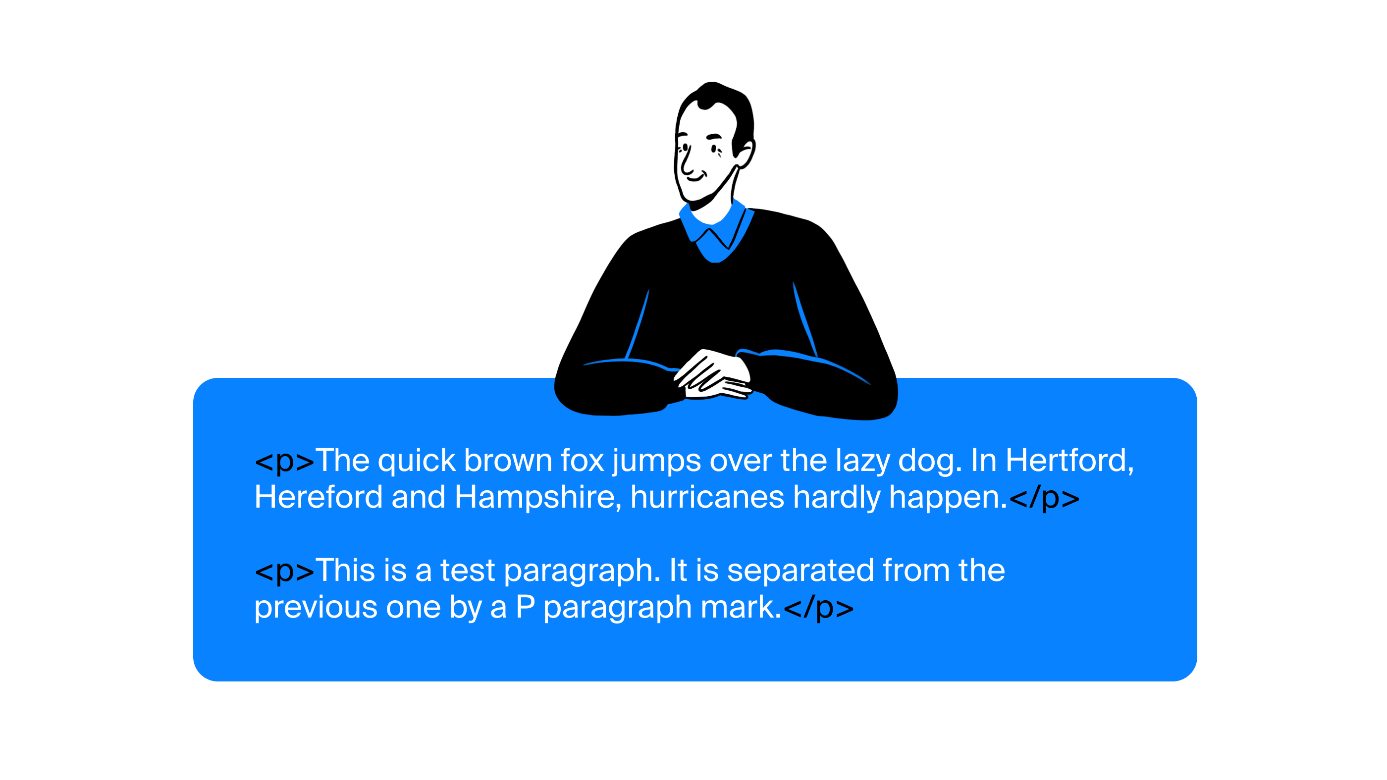
Заголовок первого уровня — самый крупный, заголовок шестого уровня — мельче простого текста. Если они правильно распределены в тексте по важности, поисковым системам легче анализировать сайт, а пользователям проще его читать.



На практике h1 используется всегда, h2 и h3 почти всегда, h4 и h5 реже, экзотический h6 — крайне редко.

# Абзац

Основная часть текста на сайте состоит из абзацев. Их заключают в специальные теги <p></p>.



Вот какой пример создатель языка HTML Тим Бернерс-Ли приводил в первом руководстве для разработчиков.

# Ссылка

Всемирную паутину и профессию веб-разработчика создал именно этот элемент. Название его тега <a></a> происходит от слова anchor, «якорь». Можно представить, что ссылка «бросает якорь» в бездну интернета и крепится к указанному адресу. Так образуется всемирная сеть.

Обычно [текст ссылки](https://takiedela.ru/) — это не адрес сайта, на который она ведёт, а слова, с которых на сайт можно попасть. Где же тогда прячется адрес?

Для дополнительных сведений в тегах прописывают атрибуты (по-латыни «признаки»). Так, адрес ссылки хранится в атрибуте с именем href. Посмотрите, как это выглядит в коде:

Скопировать кодHTML

*<!-- ссылка поведёт на один известный медиасайт -->*

<ahref="https://takiedela.ru/">Перейти на «Такие дела»</a>

# Об атрибутах

Атрибуты сообщают дополнительные сведения об элементе. Для разных сведений — разные атрибуты.

Имя атрибута говорит браузеру, какой признак нужно поменять, а значение — каким этот признак должен стать. Например, значение атрибута с именем href представляет собой адрес ресурса, куда можно перейти:



Атрибут встраивается только в открывающий тег. Значение атрибута приводят в одинарных href='http://info.cern.ch' или двойных href="http://info.cern.ch" кавычках. Выбор кавычек значения не имеет.

Рабочая ссылка без атрибута href не обходится. Бывают и необязательные случаи передачи данных через атрибут. Та же ссылка <a href=""></a> помимо непременного href может иметь атрибут target. Он «нацеливает» браузер, предписывая, где открыть ссылку — например, в этой вкладке или в новой.

Без такого атрибута вполне можно прожить, поэтому он необязательный. Но в ситуации, когда ссылка уводит на другой сайт, лучше открывать его в новой вкладке. Тогда текущая страница не закроется в браузере пользователя.

Несколько атрибутов надо писать через пробел в открывающем теге элемента. Их порядок значения не имеет:

Скопировать кодHTML

<ahref="http://info.cern.ch"target="\_blank">Сайт о первом веб-сайте</a>

*<!-- то же самое -->*

<atarget="\_blank"href="http://info.cern.ch">Сайт о первом веб-сайте</a>

**Изображения**

Элемент для добавления картинки чем-то похож на ссылку, у него есть обязательный атрибут. В атрибуте нужно указать адрес, откуда взять изображение — ведь у любого файла в интернете есть адрес, любой файл в сети где-то хранится.

Теперь найдите два отличия:

Скопировать кодHTML

*<!-- ссылка -->*

<ahref=""></a>

*<!-- изображение -->*

<imgsrc="">

Первое отличие — путь к картинке указывается атрибутом src.

Второе отличие в том, что изображение создаётся всего одним открывающим тегом. У <img> нет закрывающего, потому что нет смысла говорить браузеру, где заканчивается картинка — она и так имеет определённые размеры.



Для тега <img> есть ещё атрибут alt. Его значение — текст, который пользователь видит, когда картинка по разным причинам не загружается:

Скопировать кодHTML

*<!-- Это образец. Не копировать! -->*

<imgsrc="desktop/image.jpg"alt="здесь должна быть картинка, но что-то пошло не так">

# Структура HTML-документа

Вы знаете ряд элементарных тегов. Пора поговорить о структуре всей веб-страницы. Если вспомнить задачу о кнопке Like, то там изначальный HTML-код выглядел вот так:

Скопировать кодHTML

<!DOCTYPEhtml>

<html>

<head>

<metacharset="UTF-8">

<title>КнопкадляКатииСаши</title>

</head>

<body>

<button>Like</button>

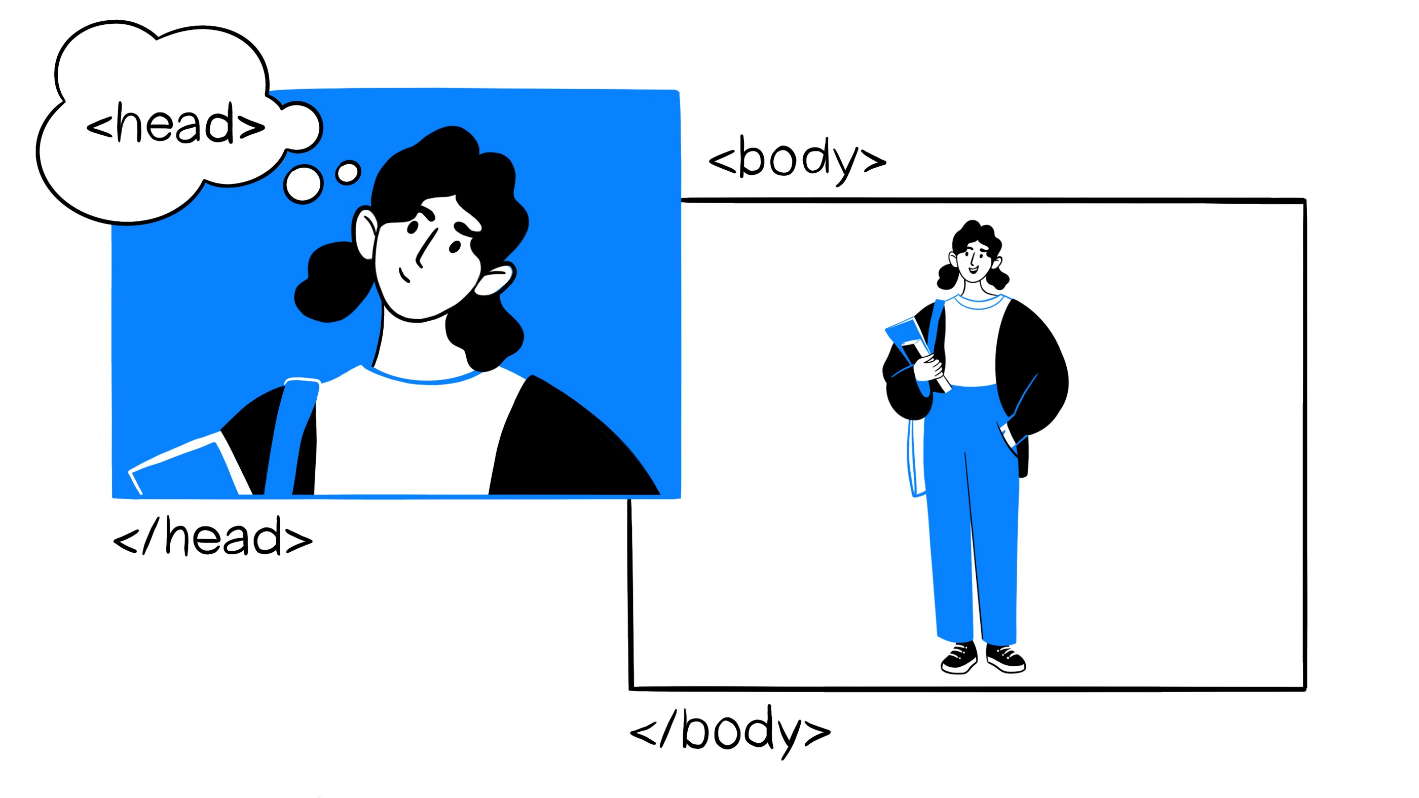
</body>

</html>

Зачем эти DOCTYPE и html, почему заголовок находится в head, а сама кнопка в body — объясним аналогией.

Встречая людей на улице, мы считываем одежду, цвет глаз, улыбку. Но мы не воспринимаем человека только как набор внешних признаков. Мы знаем, что есть мысли и чувства, незаметные со стороны. Скажем, они находятся в голове.

Любая веб-страница устроена так же — есть видимая часть и невидимая. В невидимой содержится скрытая дополнительная информация: заголовок страницы, подключаемые файлы и шрифты, кодировка.



Назовём видимое телом, а скрытое — тем, что в голове. Происходящее в голове определяет внешний вид.

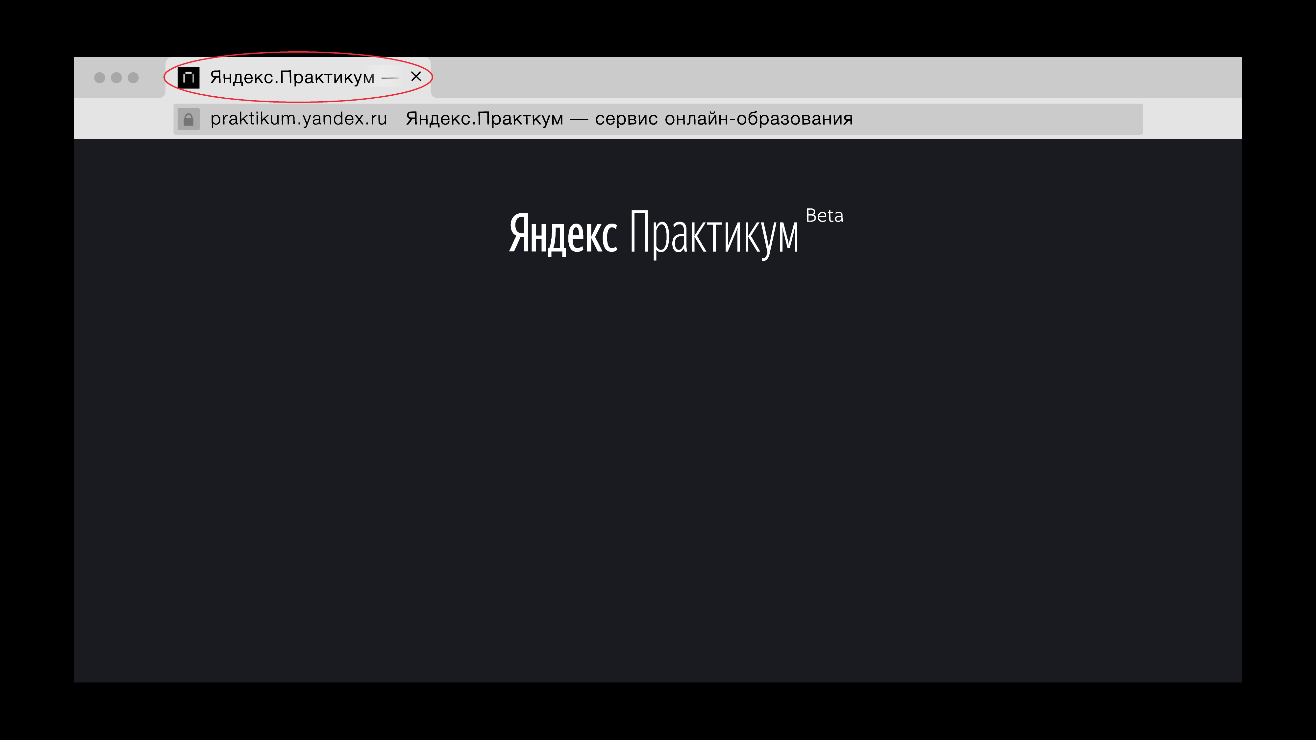
Несколько задач, чтобы запомнить эту несложную идею.

# Страница рассказывает о себе

Веб-страница может сообщать о себе дополнительные данные: описание для поисковиков и социальных сетей, кодировку для правильного отображения, язык, на котором написан текст страницы. Страница описывает информацию о данных, которые она содержит.

Всё это лежит внутри элемента head.

Название веб-страницы заключается в теги <title></title>. Оно отображается на вкладке:



Инструкции, которые страница передаёт браузеру, называются метаданные (то есть «данные о данных»). Они заключены внутри метатегов. Это теги с именем meta, что по-гречески значит «о себе».

Одиночный метатег <metacharset="UTF-8"> подключает к HTML-документу алфавиты всех живых языков Земли. Без него браузер может неверно отобразить иероглифы, буквы кириллицы и вообще символы, не входящие в базовый латинский алфавит.

В head можно задать favicon — иконку для вкладки браузера. Для этого используется тег <link>. Он содержит ссылку на внешний файл, указание типа и предназначения подключаемого файла.

СкопироватькодHTML

<linkrel="icon"href="fav.ico"type="image">

*<!--rel — назначение (от англ. relation, отношение); здесь это иконка -->*

*<!--type — тип файла. здесь это картинка -->*

*<!--href — уже знакомый атрибут адреса -->*

Элемент link дает возможность подключить не только иконку, но и файлы других типов и другого назначения, например — таблицы стилей.

# Введение в стили

HTML-документ с текстом, ссылками и заголовком — уже сайт, хоть и очень простой. [Первый в мире сайт](http://info.cern.ch/hypertext/WWW/TheProject.html), которому около 30 лет, выглядел похоже:



В наше время этого мало. Нужны цвета, шрифты, отступы, а элементы должны быть уложены в какую-то сетку. За внешность веб-страницы отвечают стили на языке CSS. Изучить CSS несложно: практически это американский английский.

Чтобы придать элементу индивидуальный стиль, в тег встраивается атрибут style. Ему присваивается название свойства (что именно менять) и значение этого свойства (как это менять).

Для примера сделаем заголовок h1 ярко-зелёным. Нужное свойство называется color, а значение — springgreen.



Свойства и значения разделены двоеточием. После значения всегда стоит точка с запятой, поскольку элемент может иметь много свойств, и нужен разделитель:

СкопироватькодHTML

<h1style="color: springgreen; font-size: 72px;">ТвинПикс</h1>

Добавьте заголовку первого уровня свойство для управления регистром и наклоните его, задав начертание курсивом. Вампонадобятсясвойства text-transform: uppercase; и font-style: italic;.

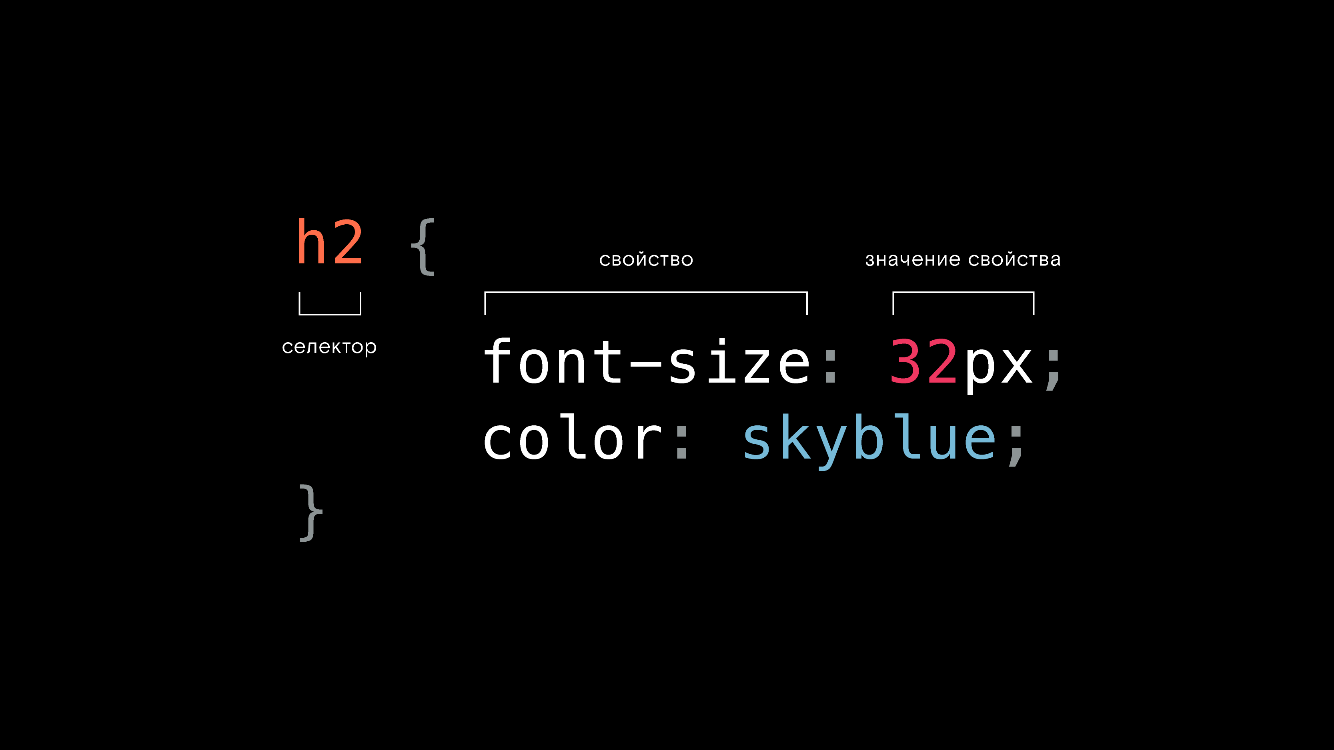
Uppercase – значит «**верхний регистр**», а italic–*курсив.*

**Правила CSS**

Стиль, заданный внутри тега, называется встроенным или инлайновым. Такой стиль подходит для уникального элемента, который должен бросаться в глаза. Но что если на странице 228 абзацев и нужно, чтобы все они выглядели одинаково?

На такой случай в CSS есть правила стиля. Правило состоит из селектора и свойств со значениями. Селектор (по-латыни «тот, кто проводит отбор») — это отличительный признак элементов, которым данное правило установит единый стиль.

Вот код, предписывающий всем заголовкам второго уровня небесно-голубой цвет и размер шрифта 32px. В роли селектора выступает имя тега:



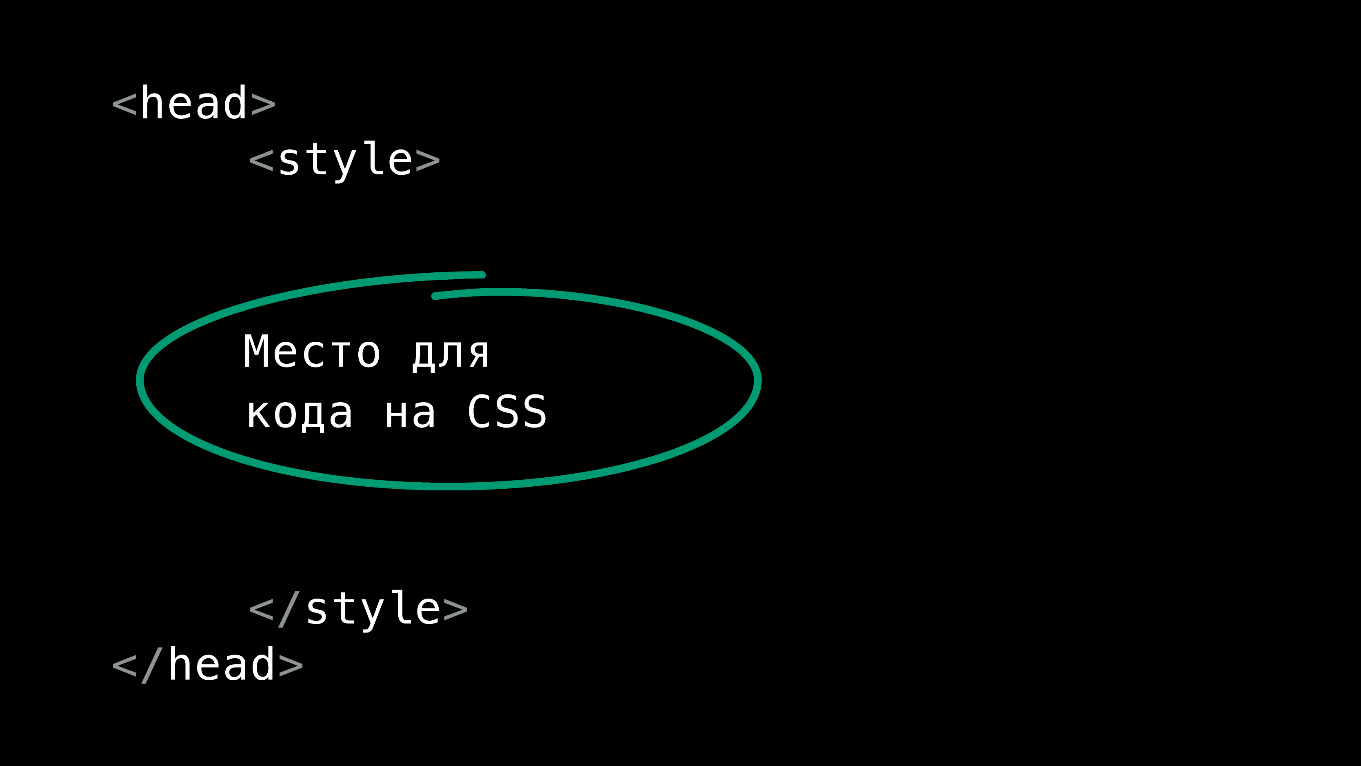
О правилах нужно знать, что:

1. после селектора в фигурных скобках указывают свойства стилей;
2. фигурные скобки вмещают сколько угодно пар «свойство-значение»;
3. каждую новую пару принято писать на следующей строке, ставьте точку с запятой;
4. селектор — не обязательно и далеко не всегда имя тега.

**Связь CSS и HTML. Тег style**

Надо разделять зоны ответственности HTML и CSS. Стили можно задавать как в самом HTML-документе, так и в отдельном файле.

В первом случае внутри элемента <head></head> создаётся элемент style. Пространство между тегами <style></style> — это зона языка CSS.



Не размещайте style внутри элемента body, держите его в head. Стиль — это закон для всей веб-страницы, а законы должны быть в голове.

**Связь CSS и HTML. CSS-файл**

Второй способ подключения стилей — через отдельный файл. Разработчики предпочитают этот способ, когда объём написанного CSS-кода превышает 30-40 строк.

В этом случае файлы с расширением .html хранят HTML-код, а файлы с расширением .css — код на CSS.

Когда стили выносят в отдельный CSS-файл, HTML-страница связывается с ним тегом <link>:

Скопировать кодHTML

<head>

*<!-- здесь что-то про кодировку, заголовок, фавиконку -->*

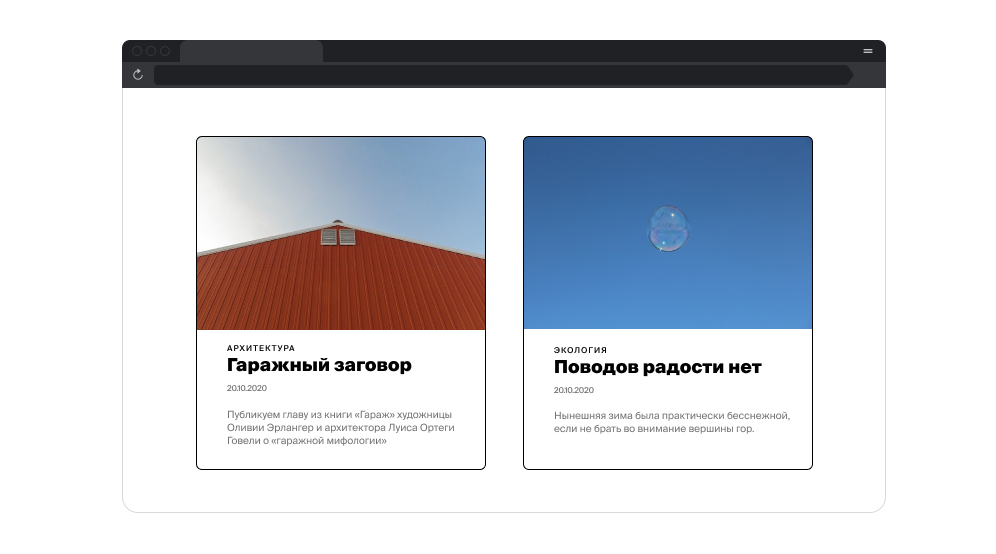
<linkrel="stylesheet"href="style.css">

</head>

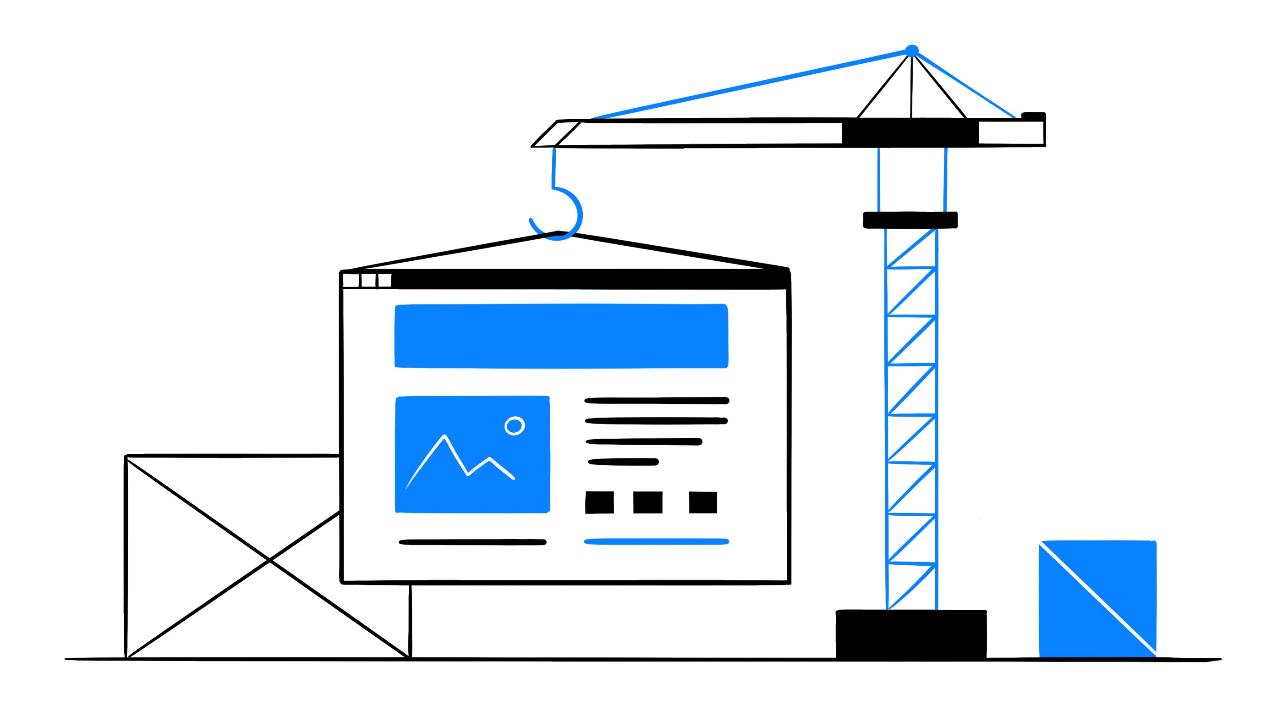
Вспомним: у тега <link> атрибут rel указывает на предназначение подключаемого файла. Значение stylesheet сообщает браузеру, что передаются именно стили. Второй атрибут, href, указывает адрес CSS-файла.

# Блоки

Веб-страница и внешне, и технически в коде, делится на блоки из HTML-элементов. Например, каждая новость на сайте — это единый блок, который состоит из картинки, названия раздела, заголовка, даты и текста.



Чтобы объединить сразу несколько HTML-элементов в один общий блок, в коде их нужно отделить от остального контента. Это делает универсальный строительный элемент div.



Любой элемент с открывающим и закрывающим тегами в HTML устроен как коробка. Особое предназначение div — быть абстрактной пустой коробкой, куда можно положить что-то ещё. Без содержимого и стилей для div пользователь может даже не увидеть его физически.

Скопировать кодHTML

*<!-- родительский элемент / parent -->*

<div>

*<!-- дочерние элементы / children -->*

<h2>Дом стоит, свет горит.</h2>

<p>Из окна видно даль.</p>

</div>

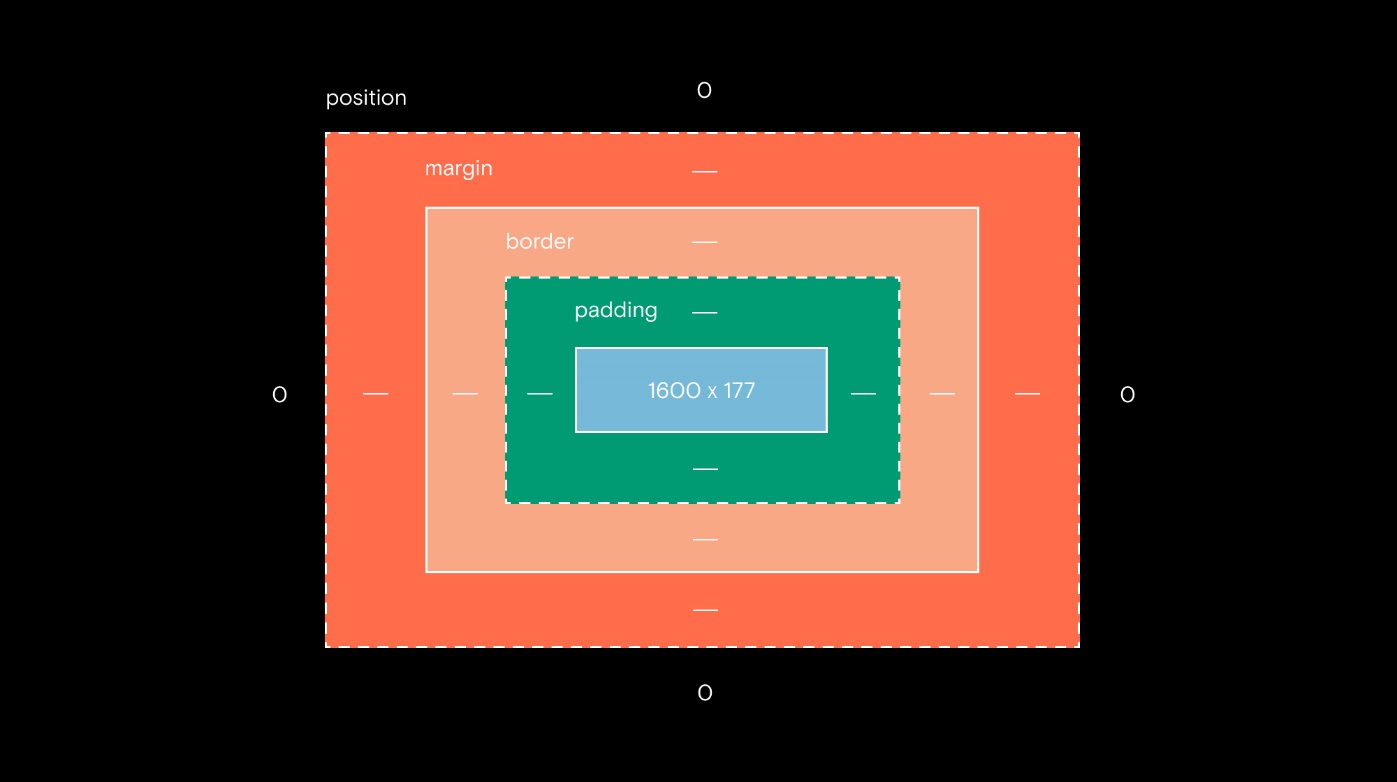
Как дом из кирпичей, страница складывается из блоков. Элемент div эти блоки создаёт и позволяет CSS управлять сразу всем набором элементов, которые оказались между <div> и </div>.

Вкладывая элементы один внутрь другого, при написании кода смещайте вложенные элементы на два пробела вправо (или клавишей Tab). Когда код вырастет в размерах, вы увидите разницу: читать код с такой разметкой намного проще, а исправлять — быстрее.

# Отступы

Почему вокруг текста обложки образовались произвольные пустоты? Браузер назначил элементам отступы — так происходит по умолчанию, пока веб-разработчик не возьмёт управление стилем на себя.

Теорию и принципы работы отступов прочитаем чуть позже, сейчас предлагаем скорее доделать проект. Для этого достаточно знать, что внешний отступ от элемента (поле) в CSS называется margin. Это свойство создаёт пустое пространство между элементом и его соседями.



Отступы можно задавать отдельно для верхней, правой, нижней или левой сторон. Тогда имя свойства обозначается не просто margin, а соответственно margin-top, margin-right, margin-bottom или margin-left.

Если написать margin: 20px, отступы в 20 пикселей появятся со всех четырёх сторон, а если margin-left: 20px — только слева.

# Playground

Игровой площадкой в коде называется место, где можно работать в режиме творчества — без заданий и без последствий. Пока Саша и Катя не прокомментировали сделанную вами обложку — экспериментируйте.



Например, можете свободно поменять фоновый цвет для элемента div. А хотите — поставьте на фон градиент. [Выбирайте градиент](https://webgradients.com/), <https://webgradients.com/>

копируйте кнопкой Copy CSS и вставляйте CSS-строчку среди стилей для div.

Можете поменять шрифт, его настройки и цвет. Можете поставить другую картинку. Только помните об авторском праве. Рекомендуем копировать адреса изображений с <https://unsplash.com/>, где работы фотографов свободны к распространению.

Главная цель урока — удовольствие от происходящего.

<https://webgradients.com/>:

background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);

background-image: linear-gradient(120deg, #d4fc79 0%, #96e6a1 100%);

background-image: linear-gradient(to right, #fa709a 0%, #fee140 100%);

# Ещё одна победа

Вас попросили придумать обложку для плейлиста, но вы заморочились и собрали, по сути, целый сайт. Супер! Теперь Катя сможет тестировать его на пользователях.

Соединив знания о HTML и CSS, вы сделали свой первый веб-проект. Без готового кода, с нуля.



## Следующая тема

Простые стили освоены, теперь вас ждёт CSS посерьёзнее. Кате нужно помочь с поиском новых верстальщиков в команду. Она хочет сделать страницу о правилах студии, чтобы прикрепить к посту с вакансией.

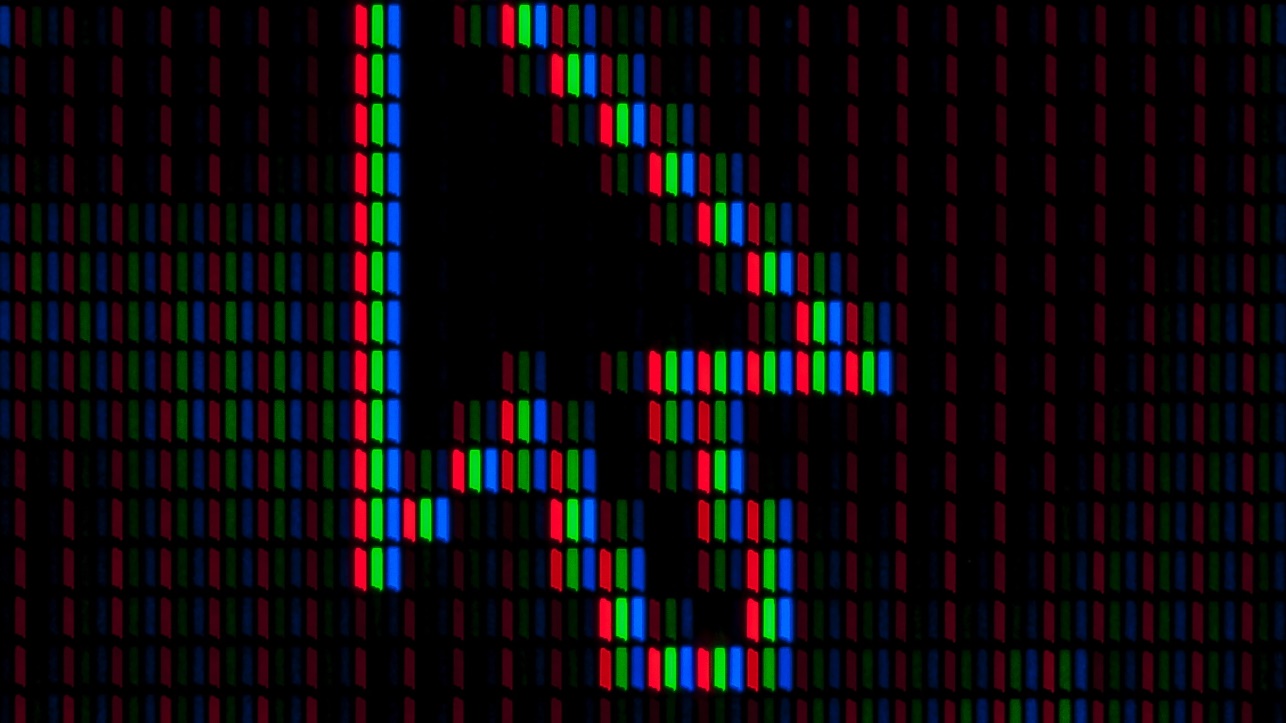
Вы сделаете настоящий лендинг: с входным экраном, вёрсткой в две колонки, фоновыми изображениями. Это непросто, зато прочувствуете, каково быть верстальщиком.

# Размеры в пикселях

Видели макет? По интуиции уже не справиться. Предстоит разобраться в фундаментальных вопросах, первый из которых — размеры.

Вы уже встречались с ними, когда верстали обложку. Чаще всего размеры даются в фиксированном размере, в пикселях. Например, 200px.

Слово «пиксель» (pixel) возникло в 1965 году, когда межпланетные станции начали передавать по одной точке фотографии поверхности Марса и Луны. Самый маленький неделимый элемент изображения надо было как-то назвать. Остановились на комбинации pix и el.



Теперь так называют самые малые точки экрана, для которых можно установить цвет и силу свечения. Сочетание этих точек создает итоговое изображение. Пиксель обозначается буквами px. Между числом и единицами измерения пробел не ставится.

# Размеры в процентах и долях

Элемент выглядит по-разному в зависимости от разрешения дисплея, то есть от количества пикселей на экране. Чем больше пикселей на дисплее, тем меньше выглядит элемент.

Квадрат со стороной 300px на мобильном телефоне покажется большим, но потеряется на мониторе Full HD (1920 на 1080 точек). В таких случаях можно определять размеры элементов относительно того, во что они вложены.

**1. Сравнение с родительским элементом — в процентах.**

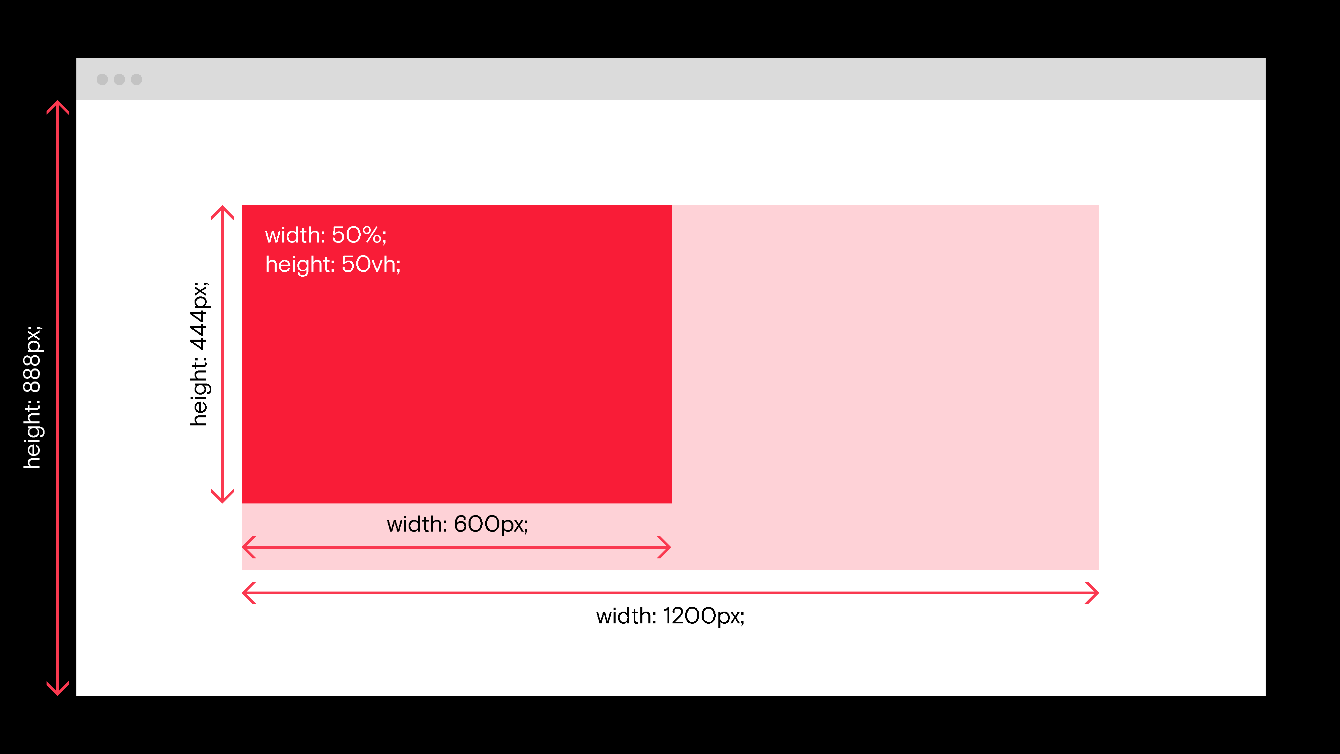
Если задать элементу width: 50%, то он посмотрит на ширину родительского элемента, разделит пополам и применит к себе. То же самое и с высотой.

Элементы h1–h6, p, div и header по умолчанию занимают 100% ширины своего родителя. Ширина элемента body по умолчанию равна видимой ширине окна браузера. Добавим обезьян 🙈🙉🙊 для привлечения особого внимания к этому факту.

**2. Сравнение с окном просмотра — в долях.**

Вы можете встретить сайты, на которых секции точь-в-точь соответствуют размерам браузера. Точнее, окна браузера, в котором вы просматриваете сайт.

По-английски окно просмотра в браузере называется viewport. Его ширина измеряется в единицах vw, а высота в единицах vh. Полная ширина окна равна 100vw, полная высота окна — 100vh.



Например, если установить элементу свойство height: 50vh, его высота станет равна половине высоты окна просмотра; 100vh — всей высоте, 40vh — 40% высоты.

height: 100% для header не поможет — высота родительского body сейчас равна тому, что в него вложено.Укажите для header значение 100vh !!!

# Цвета в HTML

В CSS цвета можно задавать не только словами на английском языке. Браузеры понимают всего 147 названий в духе wheat, seagreen, navy — на все цвета слов не хватит. Другие стандарты определения цвета позволяют передать более тонкие оттенки.

RGB-палитра описывает цвета комбинацией красного (R), зеленого (G) и синего (B). Человеческое зрение устроено так, что любой оттенок складывается из этих трёх цветов. Яркость каждого указывается в диапазоне от 0 до 255 — так можно получить какой угодно видимый цвет.

Например, чистый синий задаётся так: rgb(0, 0, 255). Здесь яркость синего цвета максимальна, а красного и зеленого равна 0.



Скопировать кодCSS

*/\* черный цвет \*/*rgb(0, 0, 0);

*/\* белый цвет \*/*rgb(255, 255, 255);

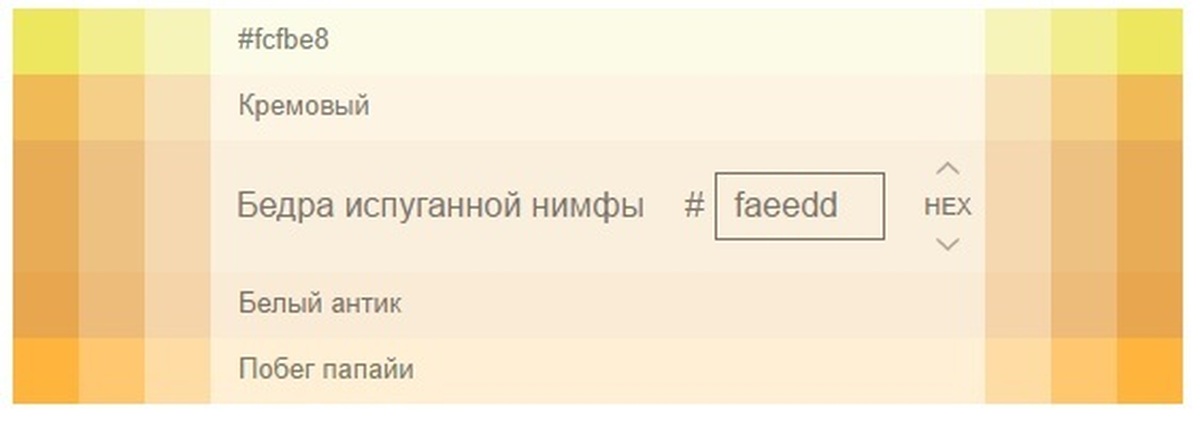
*/\* небесно-голубой цвет \*/*rgb(0, 150, 200);

/\* красный \*/ rgb(255, 0, 0);

HEX-colors — это другой стандарт. Принцип тот же, что в стандарте RGB, только числа от 0 до 255 заменяют на соответствующую им комбинацию двух символов из ряда 0 1 2 3 4 5 6 7 8 9 A B C D E F. По два символа на красный, зелёный, синий — достаточно шести символов.

Браузер узнает запись в стандарте HEX-colors по символу хеш — *#*. Так, жёлтый цвет вместо rgb(255, 255, 0) обозначается #FFFF00.

Буквы могут быть как строчными, так и заглавными, это не влияет на отображение. Например, #faeedd и #FAEEDD дадут один результат. По классификации Яндекса это [цвет бёдер испуганной нимфы](https://yandex.ru/search/?lr=213&text=%D1%86%D0%B2%D0%B5%D1%82%20%D0%B1%D0%B5%D0%B4%D1%80%D0%B0%20%D0%B8%D1%81%D0%BF%D1%83%D0%B3%D0%B0%D0%BD%D0%BD%D0%BE%D0%B9%20%D0%BD%D0%B8%D0%BC%D1%84%D1%8B):

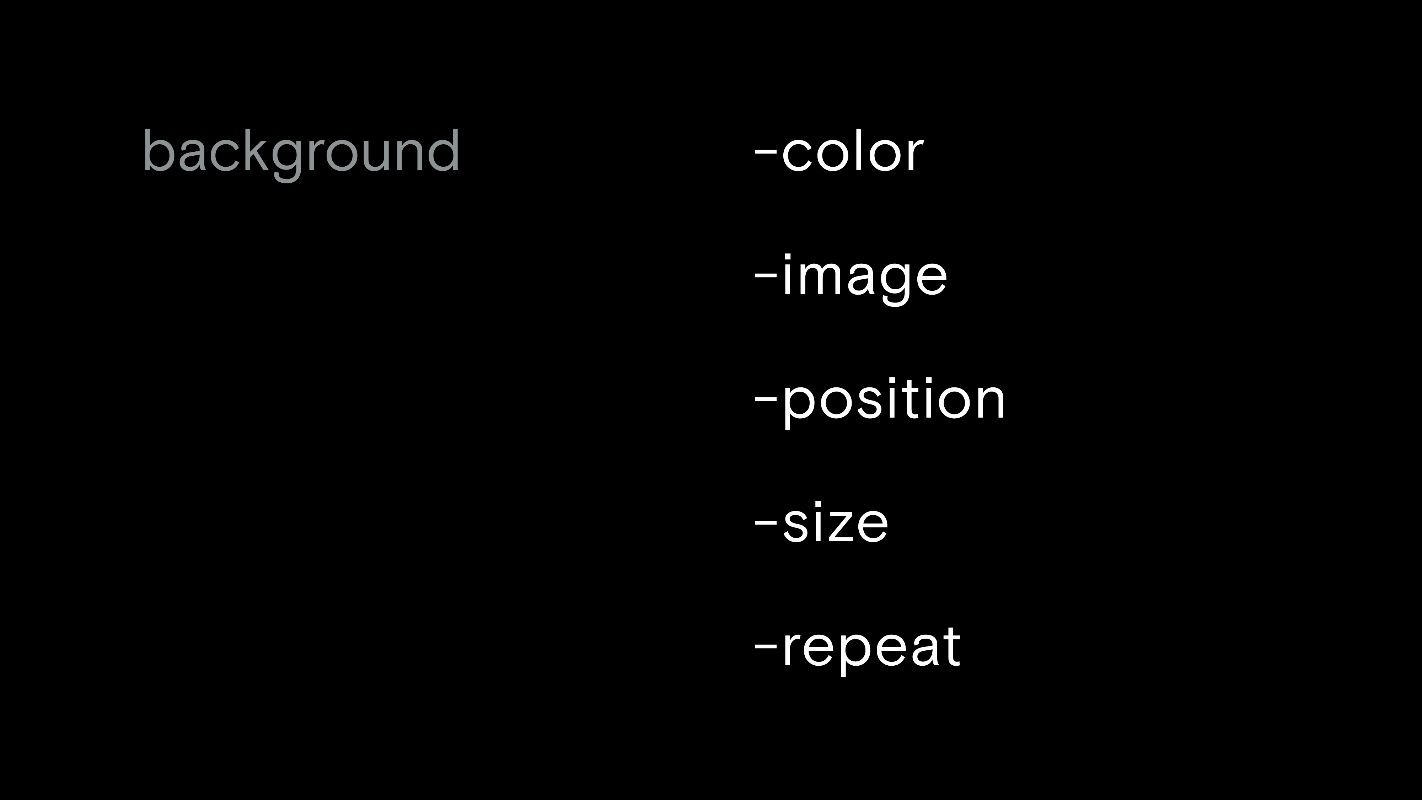


Более простые цвета, где символы в трёх парах повторяются, можно указывать коротким HEX-кодом. Например, вместо #ffffff пишется #fff, вместо #ff0000 можно #f00, вместо #ee77cc — #e7c. Попробуйте. Укажите для фона бледно-серый #eee цвет.

# Фон элемента

Не все названия свойств пишутся в одно слово, как color. Если в названии стоит дефис — например, background-color — вы имеете дело со свойством из определённой группы.

Так, для стиля фона background можно определить несколько дополнительных параметров: цвет, изображение, его размер и расположение, повторяемость. Параметр указывается после дефиса.



Скопировать кодCSS

div {

*/\* цвет фона \*/*

background-color: maroon;

*/\* фоновоеизображение \*/*

background-image: url(https://yastatic.net/s3/auth2/\_/msoffice.d1ada09f.svg);

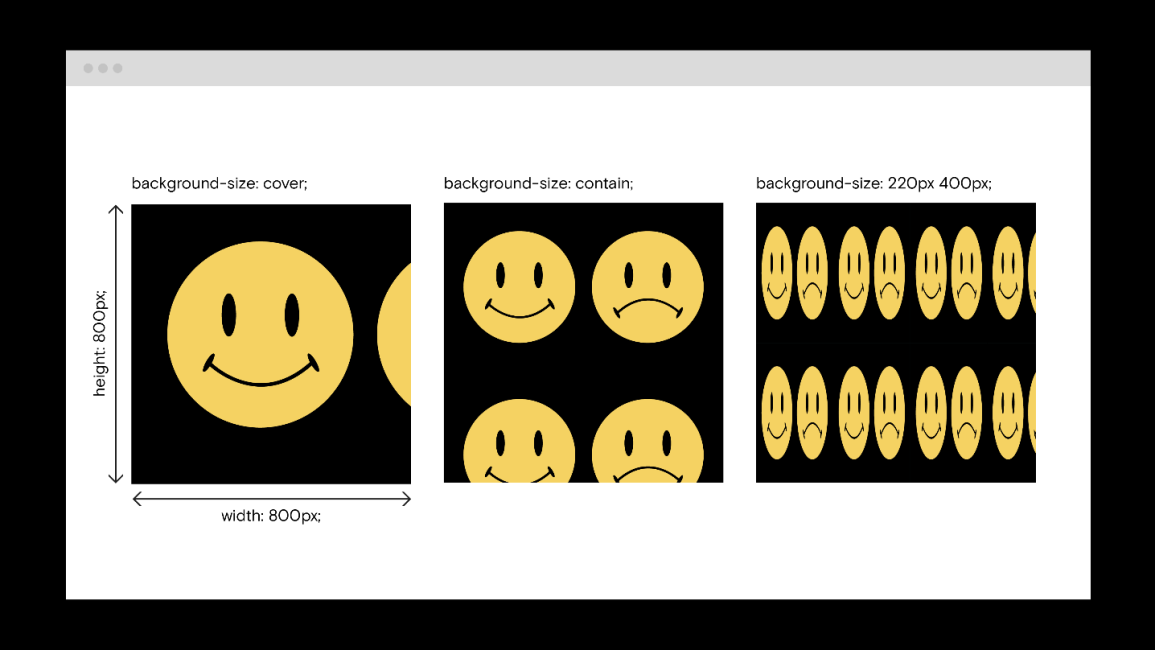
}

Адрес изображения нужно писать в скобках url(). Теперь у вас ещё больше поводов использовать изображения из интернета, только помните об авторском праве.

# Позиция, размер, повтор фона

Чаще всего для управления фоновым изображением хватает трёх свойств: background-size, background-repeat и background-position.

Размеры фонового изображения определяет background-size:



Скопировать кодCSS

*/\* cover (англ. «накрыть») сожмёт или растянет картинку \*/*

*/\* так, чтобы занимала всю поверхность \*/*

*/\* сохранитпропорции \*/*

background-size: cover;

*/\* contain (англ. «содержать») сделает так, чтобы вся картинка уместилась \*/*

*/\* сохранитпропорции \*/*

background-size: contain;

*/\* можно указать размер в пикселях \*/*

*/\* первое значение — ширина, второе — высота \*/*

background-size: 220px 400px;

Если картинка меньше поверхности блока, то по умолчанию она будет повторяться, чтобы заполнить всё пространство. Значение background-repeat может отменить повтор изображения.



Скопировать кодCSS

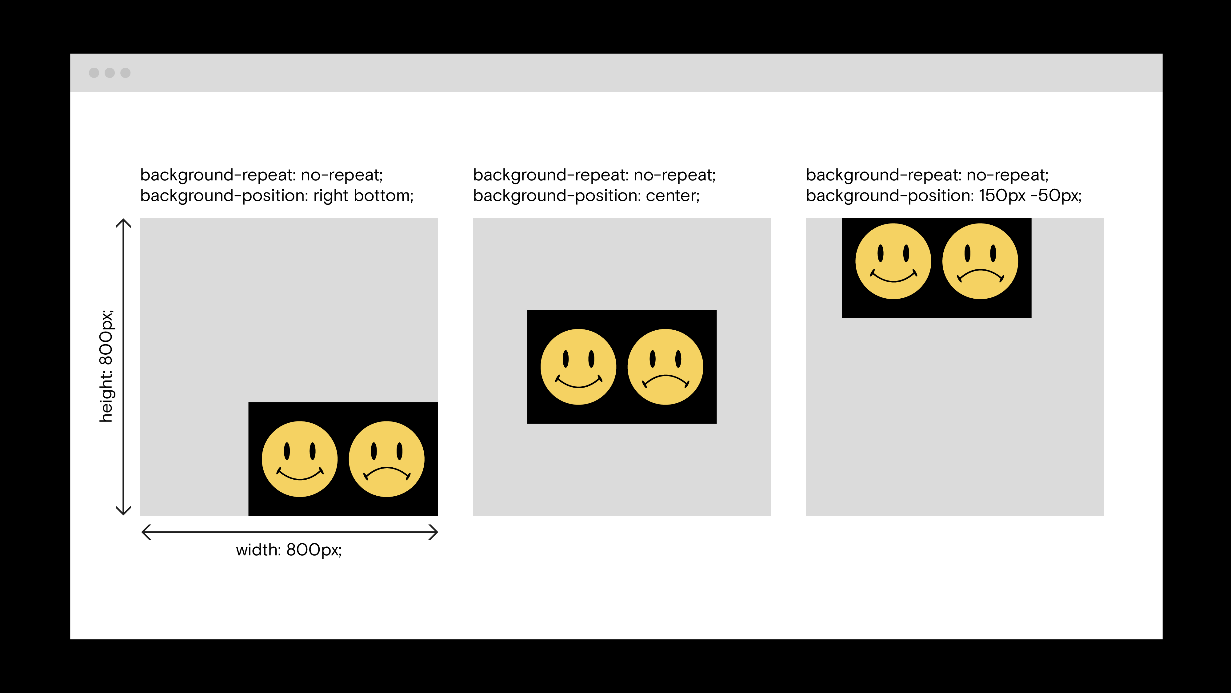
*/\* так отменяется повтор изображения \*/*

background-repeat: no-repeat;

*/\* изображение повторяется только по вертикали \*/*

background-repeat: repeat-y;

Положениемфонауправляет background-position:



Скопировать кодCSS

*/\* по центру — сразу по горизонтали и по вертикали \*/*

background-position: center;

*/\* по горизонтали к правому краю, по вертикали — к нижнему \*/*

background-position: rightbottom;

*/\* можноуказатьсдвигвпикселях \*/*

*/\* по горизонтали сдвиг на 150px вправо от левого верхнего угла блока \*/*

*/\* по вертикали — на 50px вверх \*/*

background-position: 150px-50px;

**Прозрачность**

Представьте, что между фоновым изображением и текстом стоит полупрозрачное стекло тёмного оттенка. Это распространённый приём усиления контраста и увеличения читаемости текста поверх изображения.

Аналог такого стекла в вебе — элемент с частично прозрачной заливкой цветом. Цвет можно сделать прозрачным, дополнив RGB-палитру альфа-каналом, который устанавливает прозрачность.

Скопировать кодCSS

*/\* rgb без альфа-канала, просто цвет \*/*

rgb(115, 170, 200);

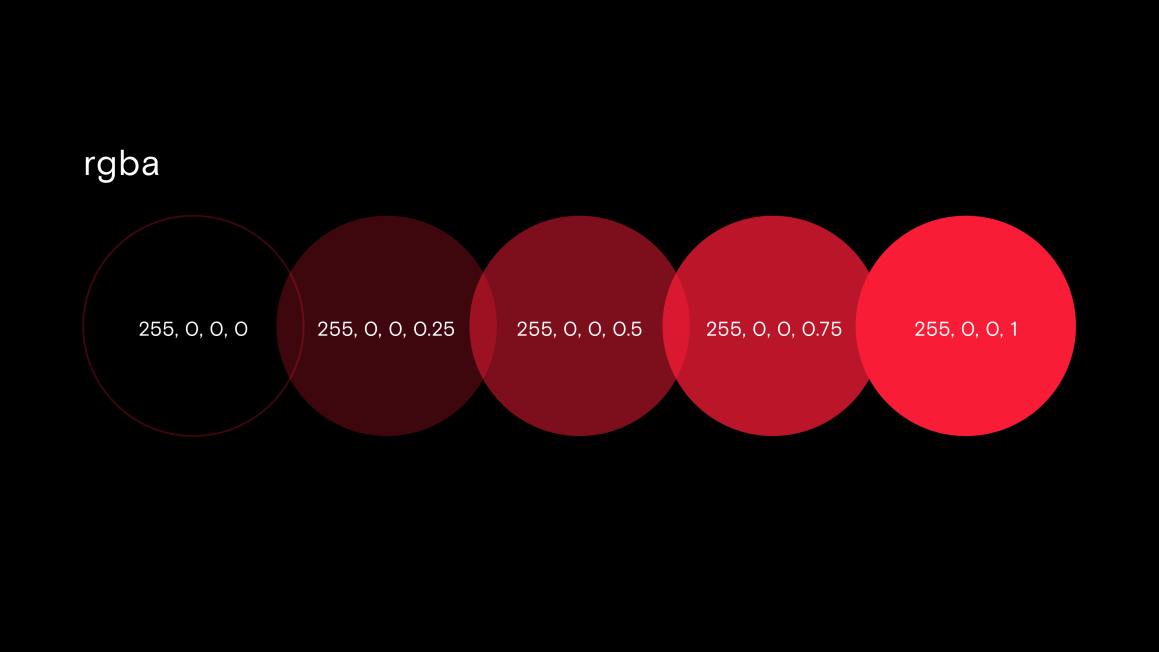
*/\* rgb c альфа-каналом, цвет + прозрачность 50% \*/*

*/\* вместо rgb пишут rgba \*/*

rgba(115, 170, 200, 0.5);

Величина прозрачности меняется от 1 до 0. Единица — это на 100% непрозрачное состояние, как закрашенное плотным однородным цветом оконное стекло. Ноль — это 0% цвета, как будто стекло начисто вымыли и сделали прозрачным.

Если нужно сделать розовый фон цвета rgb(255, 160, 201) прозрачнее в пять раз, то для rgba это выглядит как rgba(255, 160, 201, 0.2). Значение 0.2 предписывает браузеру отрисовать фону 20% цвета и 80% прозрачности.



Десятичная дробь в CSS и многих других языках пишется с точкой — не с запятой, как в школьной тетради. Также рекомендуем писать дробь без нуля.

Скопировать кодCSS

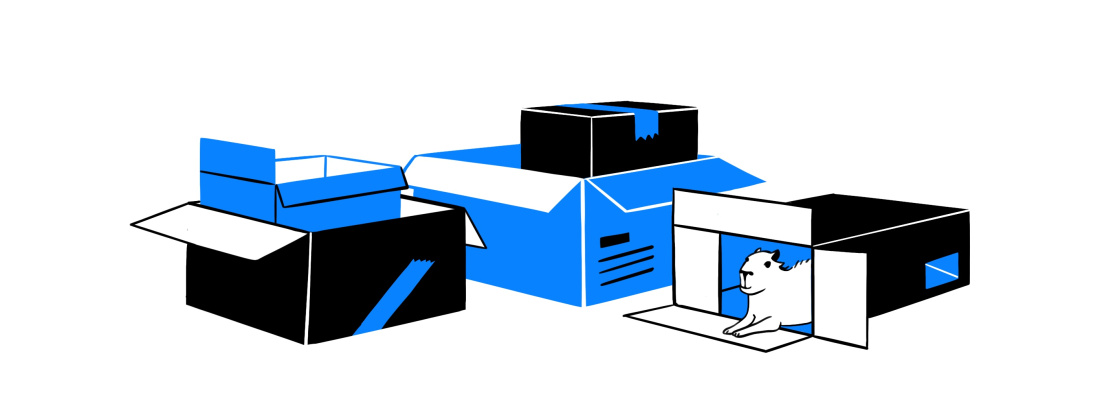
*/\* один и тот же цвет \*/*

rgba(115, 170, 200, 0.5);

rgba(115, 170, 200, .5);

**Коробка в коробке**

На сайтах элементы объединены в блоки по смыслу. Пост в социальной сети состоит из описания, картинки, комментариев — эти фрагменты страницы собраны в единый блок. Например, <div></div> или <header></header> создают абстрактный элемент-коробку, куда можно вложить другие элементы.



В языке HTML одни коробки можно вкладывать в другие без ограничений. Именно эта опция позволит вам проложить прозрачное «стёклышко» между фоновым изображением и текстом.

**Наследование**

Стиль родительских элементов оказывает влияние на стиль дочерних.

Скопировать кодHTML

*<!-- родительский -->*

<div>

*<!-- дочерний -->*

<p>Задача веб-дизайна — сделать не только красиво.</p>

</div>

Например, если абзацу p не задан конкретный размер шрифта, цвет текста или выравнивание, абзац будет «прислушиваться» к родителю, то есть к стилю элемента div.

И если для div определить свойство font-size: 40px, высота символов во вложенном элементе p тоже станет 40px. Установите для div выравнивание по центру text-align: center — и текст абзаца p выровняется по центру.

Это называется наследованием и работает с большинством CSS-свойств. Но если дочернему элементу установить собственное конкретное значение, то свойство наследоваться не будет.

**Типографика**

Текст имеет много настроек, в CSS можно разбить их на две группы. За сам шрифт отвечает группа font, за поведение текстовых элементов — группа text.

Из группы font популярны свойства:

Скопировать кодCSS

font-size */\* размер шрифта \*/*

font-weight */\* «вес» шрифта, толщина (отношение ширины штриха к высоте буквы) \*/*

font-family */\* семейство шрифта \*/*

font-style */\* стиль шрифта. например, italic — курсивный шрифт \*/*

Из группы text вам знакомо свойство text-align — выравнивание. Часто используют правило text-decoration: underline (подчеркнутый текст) или text-decoration: none (убирает подчёркивание; например, заданное по умолчанию для гиперссылок). Регистром букв управляет text-transform.



Ещё для текста можно установить высоту строки свойством line-height — это называется интерлиньяж. Свойство letter-spacing определяет расстояние между символами в тексте.

# Больше вёрстки

Шапку сайта вы сверстали, но это только начало, ведь лендинги обычно состоят из нескольких секций. Дальше — самое интересное: вы сделаете готовый продукт.

С чем вы подходите к этому рубежу? Вы познакомились с размерными величинами и цветами в CSS. Можете делать блоки такими, как захотите; управлять прозрачностью цвета и работать с фоновыми картинками.

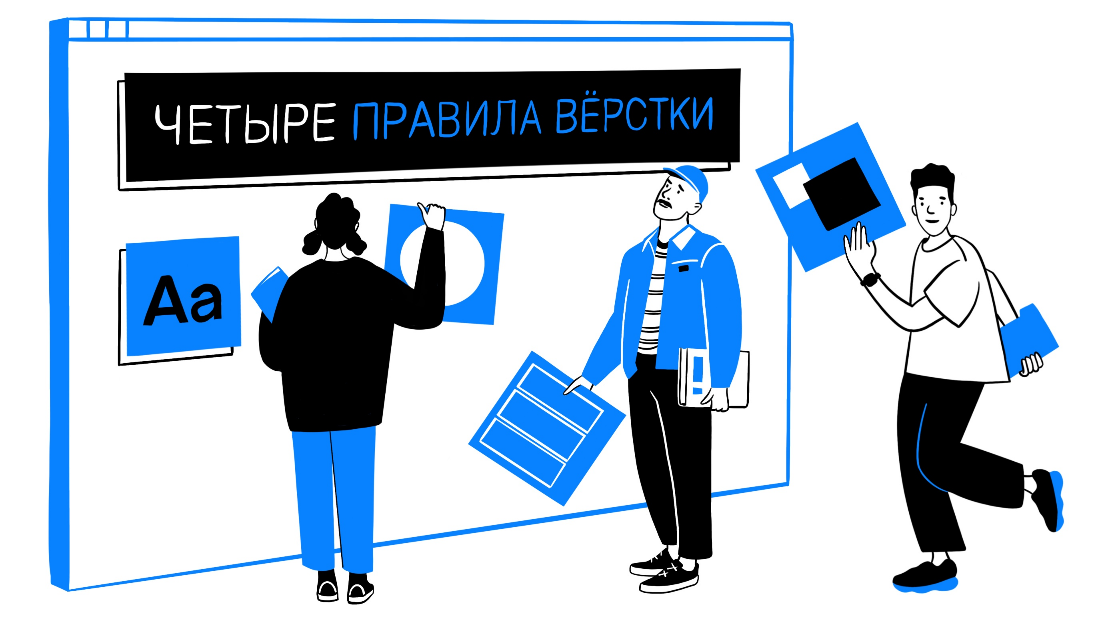
Вторая хорошая новость: по примеру свёрстанной секции с изображением и заголовком вы уже в состоянии сделать небольшой сайт-визитку.

## Следующая тема

Все элементарные принципы и понятия пройдены, впереди основная часть CSS.

Узнаем, какие бывают типы элементов и что такое блочная модель.

Она позволяет сделать то, что хочет Катя — верстать в несколько колонок.





# Новая секция

Вы уже реализовали входную секцию сайта, открыв новые возможности CSS — фон, цвета, размеры, свойства шрифтов и текстов. В этой теме вы доделаете сайт, а правил и трюков узнаете ещё больше.

Ближайший шаг — сразу после входного экрана предстоит создать секцию с основным контентом сайта. Туда попадут карточки с правилами дизайна, актуальными и для разработчиков.

Для ориентира, схема сайта станет такой:

Скопировать код

—> входной экран header

—> начало секции для контента

—>—> карточки внутри секции

—> конец секции для контента

Начать стоит с контейнера для этих карточек: заодно узнаете новый элемент section.

Рекомендуем почаще наблюдать результат в режиме просмотра — кнопка для расширения окна находится в правом верхнем углу тренажёра.

Для создания секций и разделов применяют элемент section. Он ведёт себя как div, но сразу сообщает браузеру о своём предназначении — это позволит поисковикам качественнее прочитать ваш сайт.

Сразу после элемента header создайте пустой элемент section

Откуда там возник тёмно-серый фон и что происходит? Всё логично.

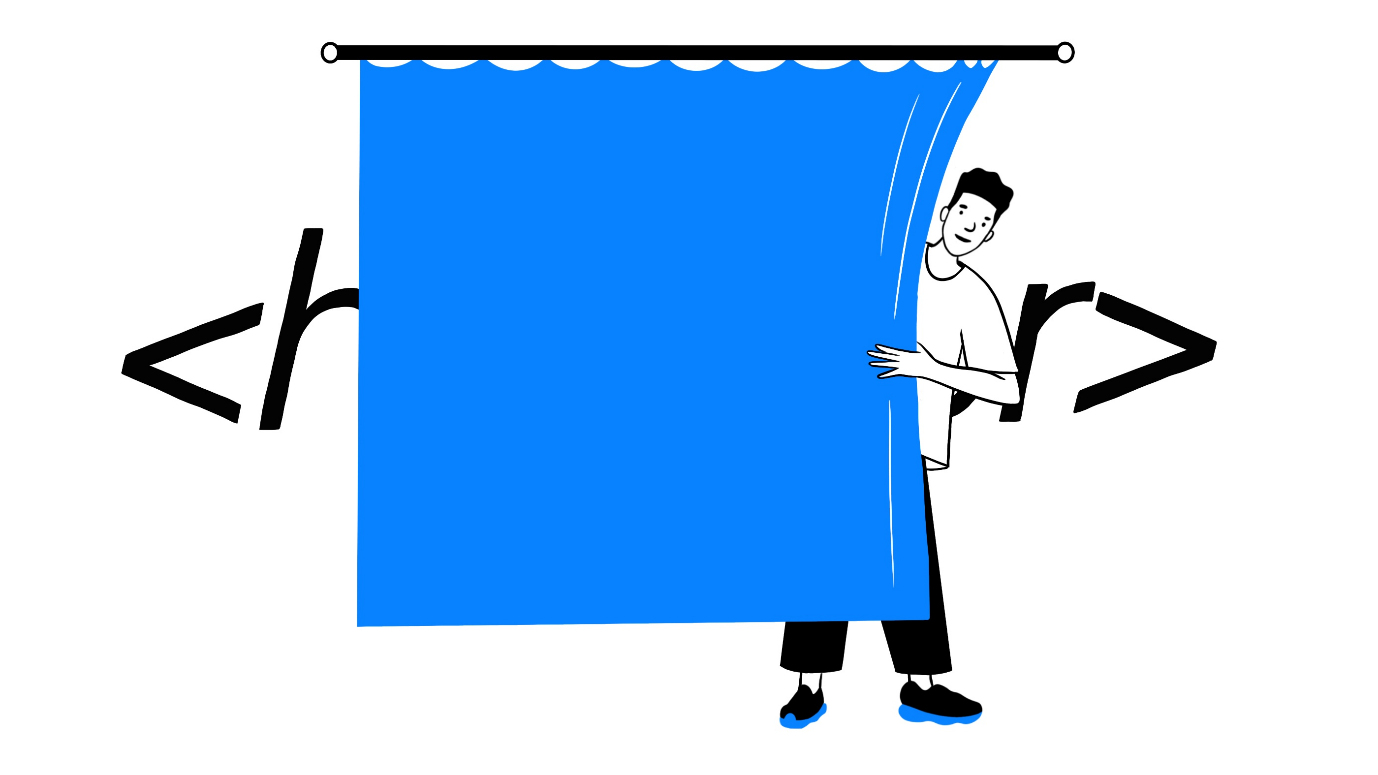
Ранее внутри header вы поставили div, как полупрозрачное стекло между текстом и изображением. Написали для него правила в style.css — они действуют для всех div на странице.

Карточка — это ещё один div. Элемент section получил размеры из-за вложенного текста, div забрал размеры section и получил цвет фона. Это не ошибка, всё в порядке.

Как это исправить, расскажем в следующем уроке. Ничего не меняйте, нажимайте «Проверить» и «Далее».

# Приятное и полезное

В предыдущей версии этого курса студенты жаловались на один нюанс. Сейчас после каждого обновления результата нужно проскролить header, чтобы увидеть изменения. Это отвлекает, раздражает и усложняет восприятие. Вот бы этот header убрать на время, но не удалять. Так можно?



Можно. Комментарии в коде нужны не только для информативных заметок, но и для временного отключения строк кода без их удаления. В каждом языке комментарии оформляются по-своему.

Скопировать кодHTML

*<!-- в HTML-коде комментарий окружён такими символами -->*

*<!-- <p>невидимый пользователю абзац</p> -->*

Скопировать кодCSS

*/\* в CSS — такими \*/*

*/\* p {*

*color: white;*

*} \*/*

Браузер игнорирует и не реализует написанное внутри комментариев.

# Классы

Теперь представьте: нужно не два, а десять разных div. Тогда селектор div {} навяжет полупрозрачный серый фон всем десяти, пресекая любой индивидуализм.

На выручку приходят атрибуты class. Классы задают каждому элементу именной ярлык. Так, <div class="card"></div> позволит обращаться в стилях к элементу card, не затрагивая свойства других div.

Один и тот же класс можно присвоить любому количеству элементов. Например, класс white-text — всем элементам с текстом белого цвета.

Скопировать кодHTML

<h1 class="white-text">Откуда к нам пришла зима</h1>

<p class="white-text">Не знаешь ты, никто не знает</p>

*<!-- в html название класса указывается в кавычках атрибута class -->*

Текст будет белым, если в стилях прописано правило для класса:

Скопировать кодCSS

.white-text {

color: #fff;

}

*/\* в стилях селектор класса начинается с точки \*/*

Только элементы с классом white-text получат белый текст.

Имена классов пишут латиницей, без пробелов, начиная с букв, а не цифр или спецсимволов. Такой способ определения стилей — надёжен и точен. В дальнейшем рекомендуем задавать стили для классов и свести к минимуму использование имён элементов в качестве селекторов.

# Типографика под контролем

Отступы вокруг заголовка карточки вы не заказывали. Браузер установил их без вашего ведома.

Верстальщик берёт контроль за поведением элементов на себя. Предстоит сбросить все «заводские настройки» браузера, а затем задать стили для текстов внутри карточек.

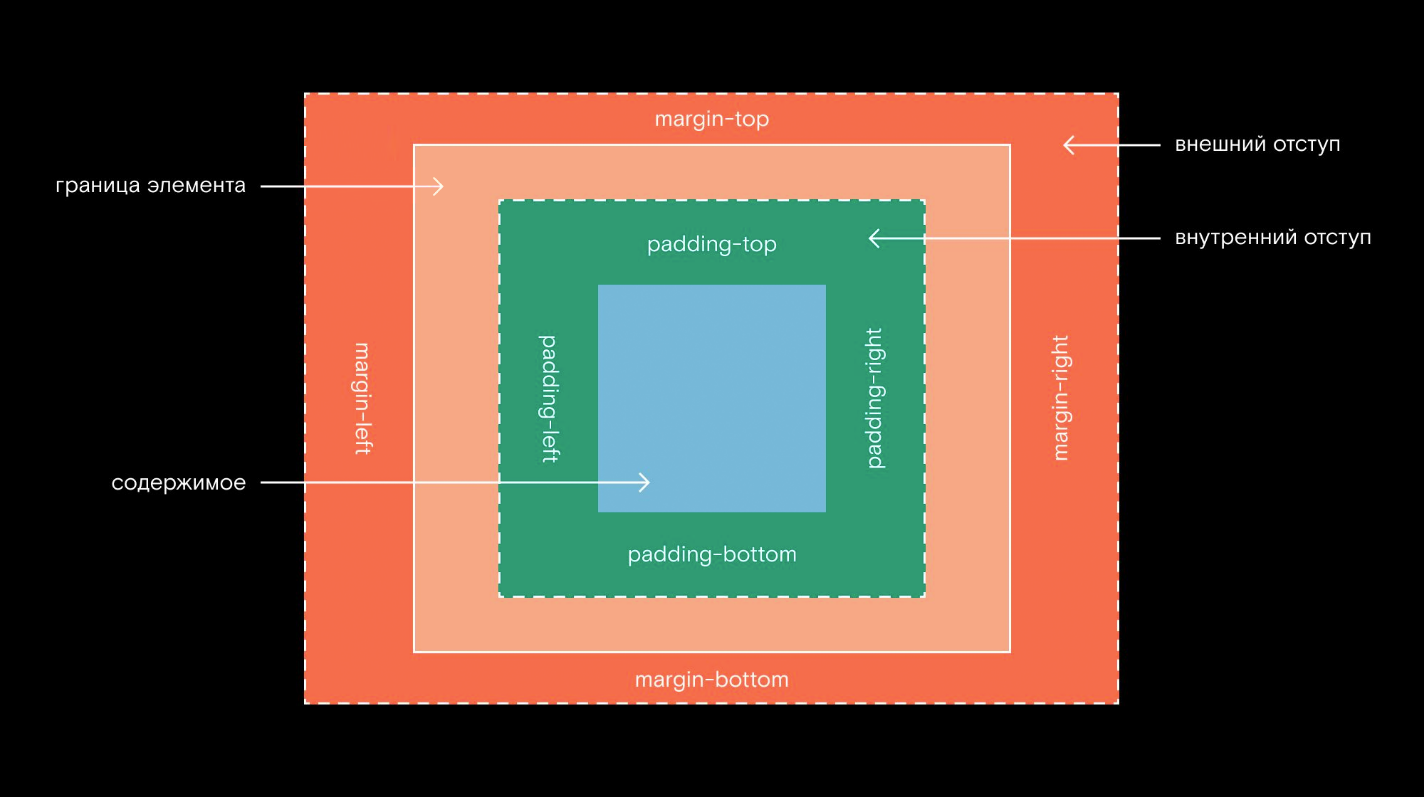
**Поток и блочная модель**

Порядок расположения элементов на странице называют поток. Браузер читает код файла сверху вниз и так же отрисовывает страницу. Поэтому говорят, что элементы следуют друг за другом в потоке.

Знакомые вам div, section, header, h1–h6 и p, располагаясь в потоке, по умолчанию занимают всю ширину своего родителя. Такие элементы называются блочными.

Модель любого блочного элемента:

* содержимое с размерами width и height;
* внешние отступы, они же поля: margin;
* внутренние отступы padding;
* границы, которых в коде вы пока не встречали, но ещё встретите.



Когда говорят «элемент 200 на 300», то имеют в виду размер содержимого до границы включительно, margin сюда не входит.

# Внешние и внутренние отступы

Внешний отступ — это пространство вокруг границы элемента снаружи. Задаётся свойствами группы margin, с ними вы уже имели дело.

В отличие от внешних отступов, внутренние — padding — смотрят внутрь элемента и создают свободное пространство между границей и содержимым.

Слово padding взято у портных и означает подплечник, подушечку между тканью пиджака и плечом. По аналогии: человек в пиджаке — это элемент, сам пиджак — видимая граница, тело под пиджаком — это контент. Чем больше padding, тем крупнее силуэт при том же объёме контента.

Для разных сторон внутренние отступы указываются, как у margin.

# Короткая запись свойств

Устанавливать отступы для одного элемента с разных сторон можно короче. CSS позволяет записывать свойство в одну строку без уточнений стороны.

Скопировать кодCSS

*/\* направление расстановки — по часовой стрелке, начиная сверху \*/*

*/\* 4 значения \*/*

*/\* для каждой стороны – своё значение \*/*

padding: 20px 15px 30px 15px;

*/\* 3 значения \*/*

*/\* верх — 10px, по бокам — 20px, низ — 30px \*/*

padding: 10px 20px 30px;

*/\* 2 значения \*/*

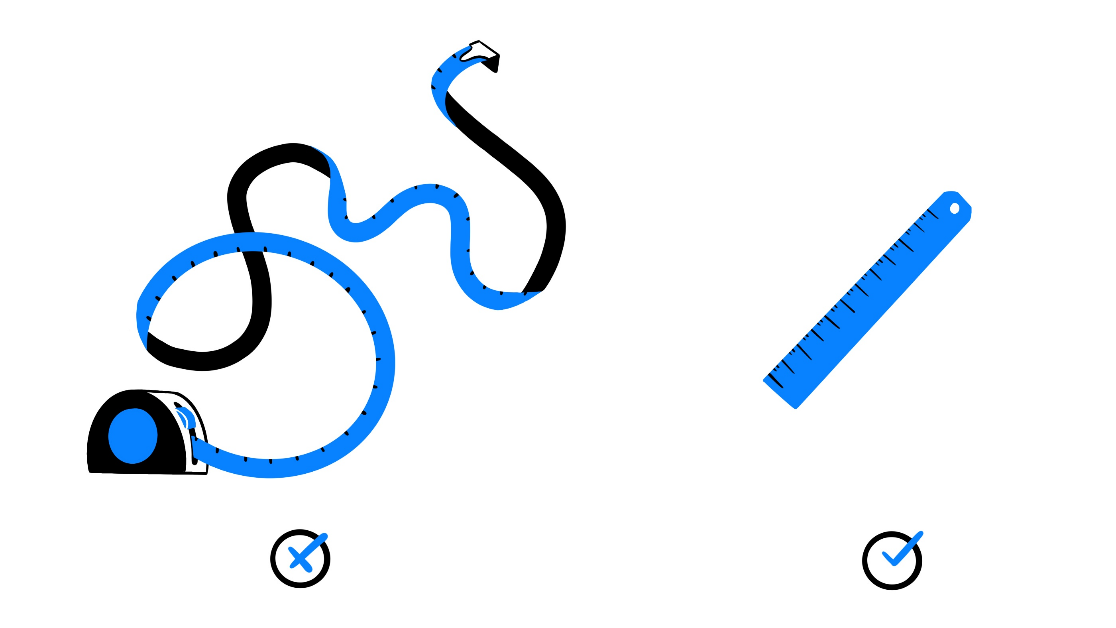
*/\* верх и низ — 10px, по бокам — 20px \*/*

padding: 10px 20px;

*/\* 1 значение \*/*

*/\* 10px со всех сторон \*/*

padding: 10px;



Работает и для margin, и для padding.

# Границы

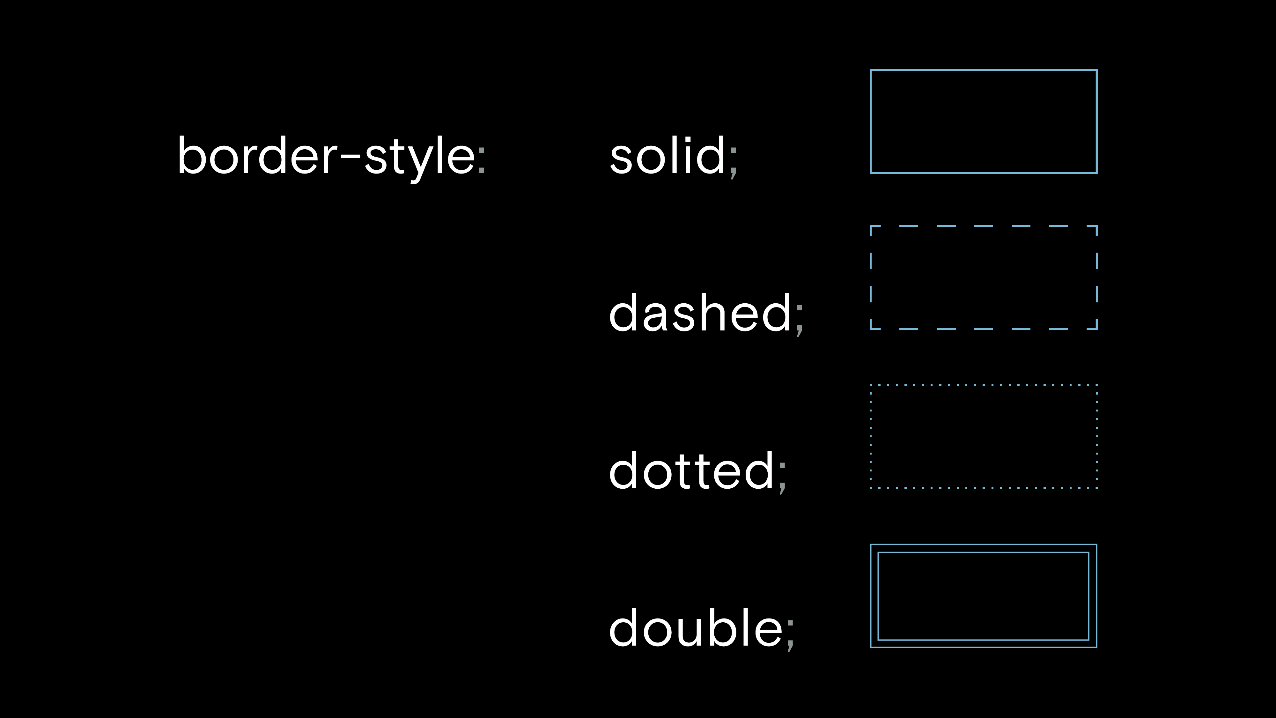
Граница элемента находится между его margin и padding. В стилях границы определяют свойствами группы border:

Скопировать кодCSS

border-color: #000; */\* цвет границы \*/*

border-width: 1px; */\* толщина границы в px \*/*

border-style: solid; */\* начертание границы (см. картинку) \*/*



Опять же — вместо написания трёх свойств по отдельности, разработчики применяют краткую форму. Значения color, width и style указывают внутри одного свойства через пробел:

Скопировать кодCSS

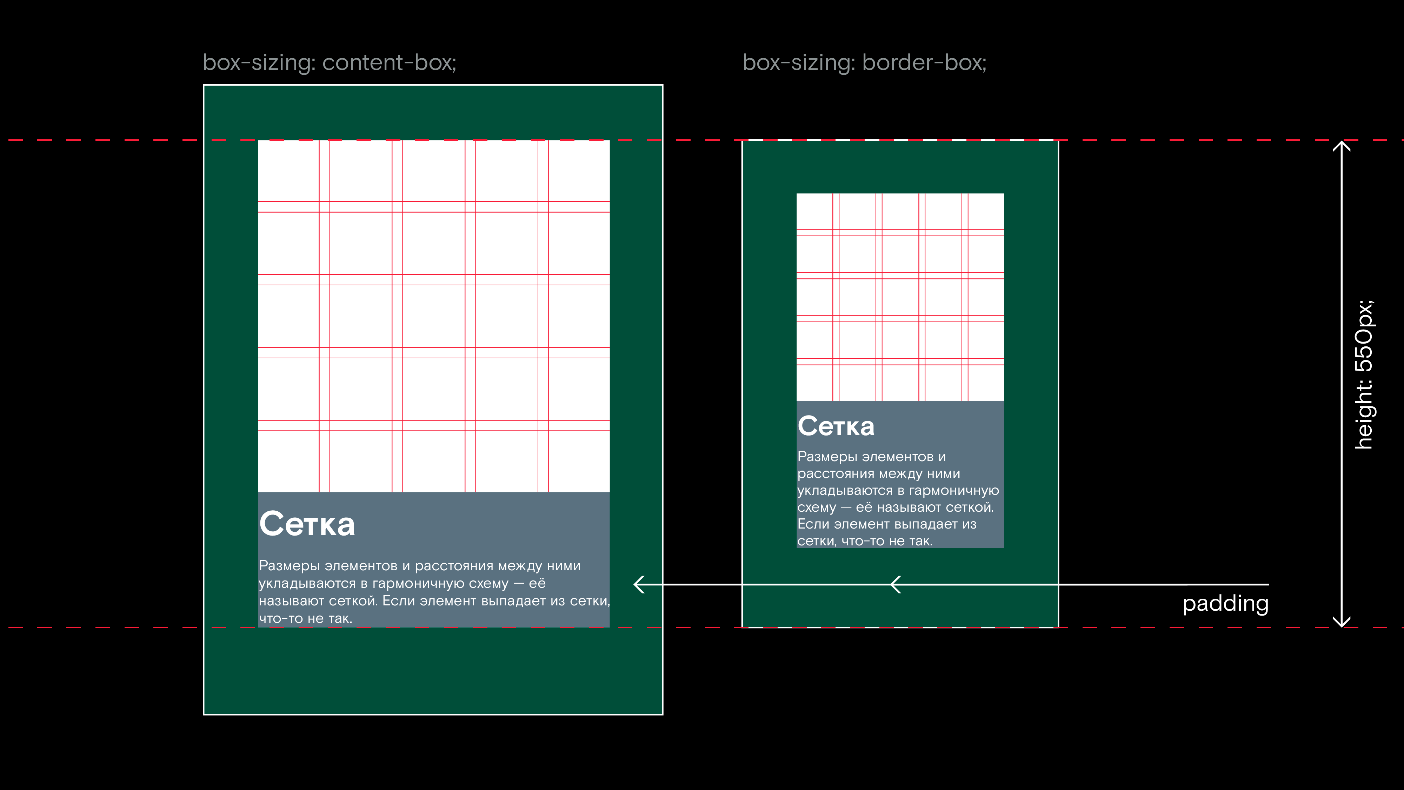
*/\* непрерывная граница черного цвета толщиной 3px \*/*

border: 3px solid #000;

# Внешний и внутренний фокус элемента

Обратите внимание на смысловую нестыковку. Ранее вы указали ширину 350px для карточки — как на макете. Но после манипуляций с внутренними отступами и границей, которые увеличили элемент, его ширина уже не та.

Выход есть. Например, брать калькулятор и высчитывать, насколько уменьшить значение width для класса card, чтобы в сумме с отступами и границами вышло 350.



Долго, сложно и чревато ошибками. Чтобы не считать вручную, есть свойство box-sizing — оно определяет поведение границ и внутренних отступов. По умолчанию для всех элементов действует правило box-sizing: content-box, границы и внутренний отступ расширяют элемент.

Установив для box-sizing значение border-box, вы измените фокус элемента: границы и внутренний отступ отрисуются внутрь. Общая ширина будет равна значению width. Довольно интуитивное поведение, поэтому нормальная практика — задать его сразу для всех элементов на странице.

# Блочные + строчные

Кроме блочных элементов, в HTML есть строчные. Например, ссылки.

Блочный элемент занимает по умолчанию всю ширину родительского элемента.

Строчные элементы занимают ровно столько места, сколько контента в них содержится. Если они идут подряд, то по умолчанию все расположены на одной строке. Им невозможно задать ширину или высоту — они игнорируют указание размеров через стили.

Что делать, когда блоки с определенными размерами должны следовать друг за другом по горизонтали и не занимать всю строку? Можно задать элементам комбинированный тип — блочно-строчный. С одной стороны, они не занимают собой всю горизонталь, с другой, восприимчивы к указанию размеров через CSS. Например, так ведут себя элементы img.

Переопределяется тип свойством display:

Скопировать кодCSS

*/\* display по-английски «отображение» \*/*

display: block; */\* сделает элемент блочным \*/*

display: inline; */\* сделает элемент строчным \*/*

display: inline-block; */\* сделает элемент блочно-строчным \*/*

Благодаря работе с display становится возможной сетка элементов из текста второй карточки

# margin: auto;

Напомним, карточки — блочные элементы, и поэтому занимают всю родительскую ширину даже несмотря на установленное width: 350px. Теперь мысленный эксперимент: как бы вы разместили карточки посередине занятой ими строки?

Свойство text-align не поможет — оно действует только на текст. Можно на глаз задать количество пикселей для margin-left первой карточки, пока правила не окажутся в центре. Но такой способ сработает только на вашем компьютере, а для экрана другого размера потребуется другое число пикселей.

С центрированием блочных элементов работает особое значение auto. Оно автоматически устанавливает максимально возможный отступ по горизонтали. Если задать свойствам margin-left и margin-right значение auto, получатся максимальные отступы с обеих сторон, и элемент встанет по центру своего родителя.

# Несколько классов

Несколько классов пишут внутри значения одного атрибута class через пробел:

Скопировать кодHTML

<div class="first second"></div>

*<!-- у элемента есть и класс first, и класс second -->*

Например, общие свойства можно задать правилом для класса text, а уникальные — индивидуальным классом special-text.

Скопировать кодHTML

<p class="text">Композиция дня:</p>

<p class="text special-text">Bonobo, Rhye — Break Apart</p>

Скопировать кодCSS

.text {

font-size: 20px;

}

.special-text {

font-size: 32px;

color: #FEEB78;

}

Через пробел можно задавать и два, и три, и четыре класса. Можно даже пять. Или шесть. Ну, вы поняли.

**Тени**

Тень — полезный инструмент дизайна. За её создание отвечает свойство box-shadow.

Свойство передаёт браузеру информацию, с какой стороны относительно элемента следует нарисовать тень, сделать её побольше или поменьше, и каким цветом. В коде это выглядит так:

Скопировать кодCSS

div {

box-shadow: -2px 2px 5px #FD6969;

*/\* сдвинута на 2px влево, на 2px вниз \*/*

*/\* радиус размытия — 5px \*/*

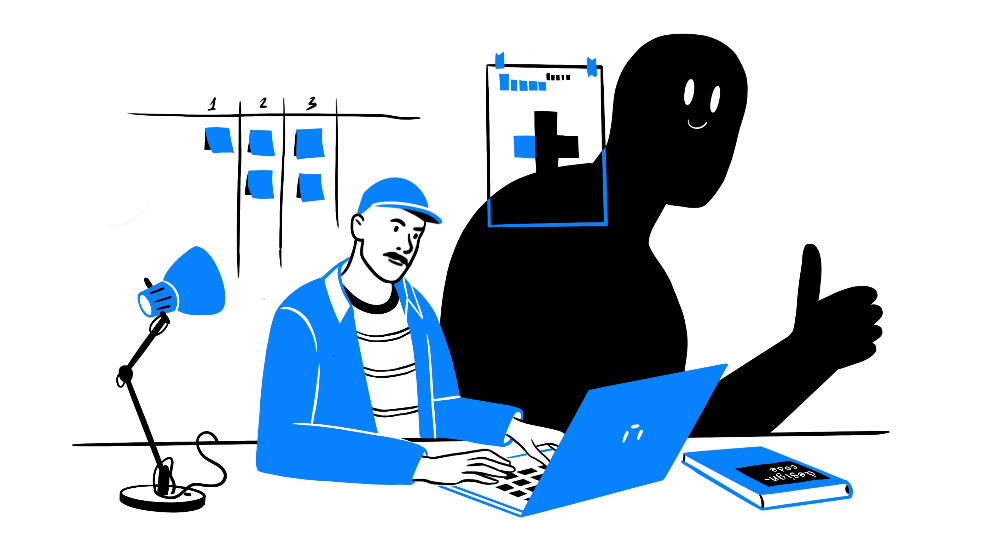
*/\* цвет красноватый, #FD6969 \*/*

}

Последовательность значений, которые описывают тень:

* горизонтальный сдвиг (отрицательное значение помещает тень левее элемента, положительное — правее);
* вертикальный сдвиг (отрицательное значение помещает тень выше элемента, положительное — ниже);
* радиус размытия (чем значение больше, тем тень шире и бледней);
* цвет (задаётся так же, как цвет текста или фона).

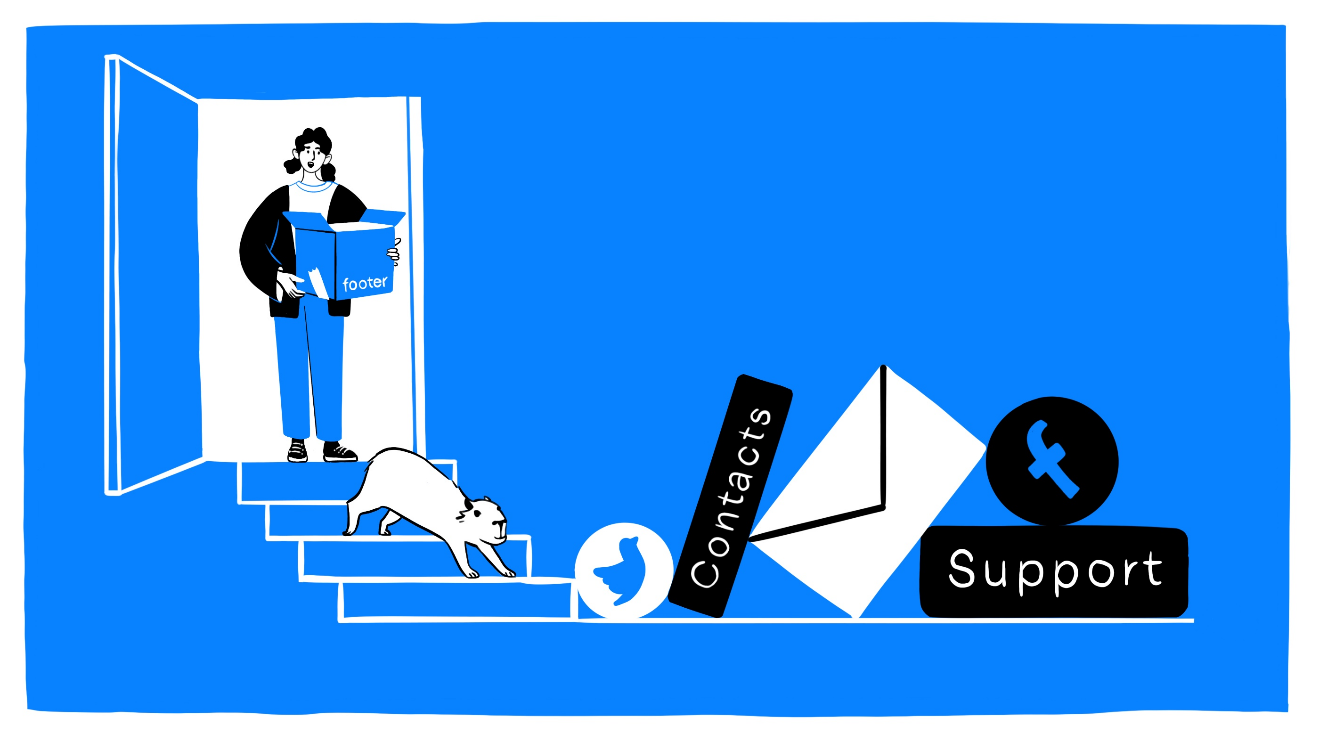
Иногда тень не нужно сдвигать, достаточно только размытия. В этом случае указывают, что первые два значения равны 0.



Тень можно создавать не только для границ элемента, но и для текста. В этом случае используют свойство text-shadow. Оно работает аналогично.

# Подвал сайта

Если верх веб-страницы называют «шапкой», то низ — это «подвал» сайта. Обычно он, как титры, содержит сопутствующую информацию: контакты, адреса, ссылки на социальные сети и другие страницы сайта.



Для создания подвала в HTML существует специальный элемент footer. Он, как div, header или section, создаёт пустую коробку.

# Центрировать по вертикали

Вы уже сталкивались с этим, когда нужно было центрировать надпись «ЧЕТЫРЕ ПРАВИЛА ВЁРСТКИ» по вертикали. Тогда вам пришлось устанавливать внутренний отступ в пикселях, чтобы подвинуть текст в нужное место.

Это простой способ, только он не всегда работает. Но значение 100px на маленьком устройстве — это уже треть экрана, а на большом мониторе — почти незаметно. Поэтому не подгоняйте пиксели только под свой экран, подумайте и о посетителях вашего сайта.

И всё-таки: уже всерьёз планируют полёт на Марс, а в вебе до сих пор нет способа поставить элемент по центру автоматически?

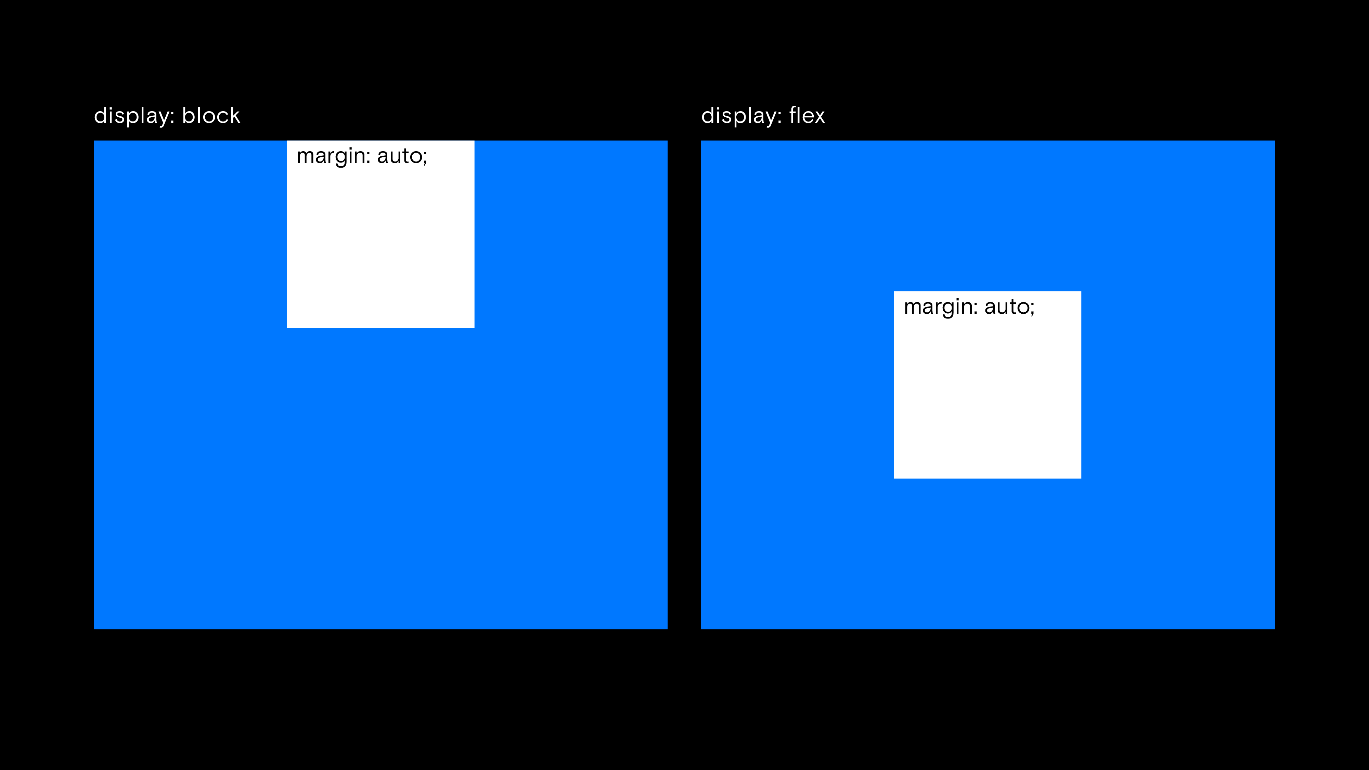
Способ есть. Для уже изученного свойства display помимо block, inline и inline-block есть особое значение flex:

Скопировать кодCSS

display: flex;

*/\* \*flex\* переводится как «гибкий» \*/*

Элемент с таким свойством становится flex-элементом и — как супергерой — получает сверхспособности. Например, внутри него margin: auto работает не только по горизонтали! Невероятно, но это так. Чтобы запомнить, решите задачу центрирования для подвала.



Скопировать кодCSS

*/\* PS >*

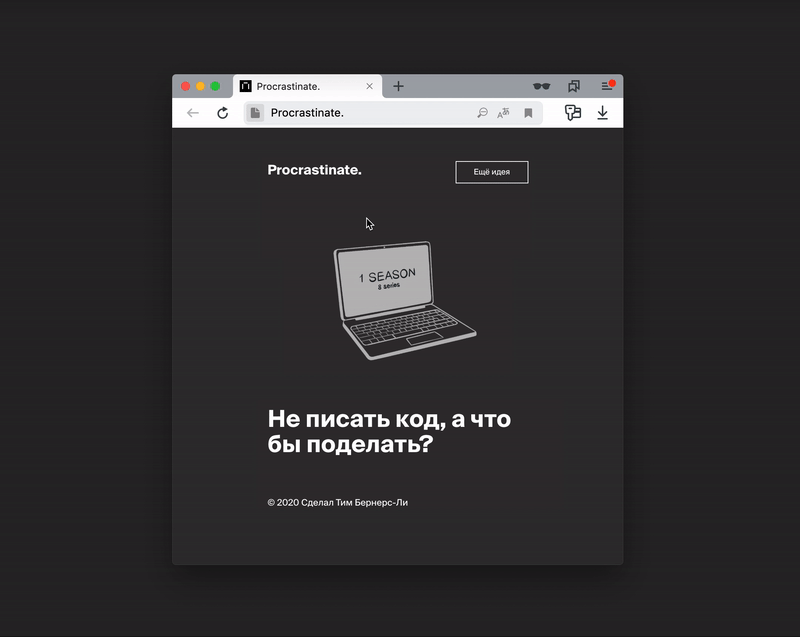
*О полном спектре возможностей \*flex\* мы рассказываем в курсе профессиональной вёрстки. \*/*

**Новая цель**

Самообразование — это круто, но атмосфера душная. Как будто вы попали в спортзал олимпийского резерва в интернете, а соревнования отложили.

Все успевают одновременно учиться и вести блог о невероятной продуктивности. Ежедневно постят сотни статей, о том, как превратиться в сверхчеловека. Как стать лучше? Как работать эффективнее? Как писать код на беговой дорожке? Обычно эти статьи либо не помогают, либо того хуже — вызывают свинцовую досаду. У кого-то получилось, а вам ещё предстоит работать.

Мы не собираемся превращать жизнь в страдание. Конечно, нужно много трудиться, чтобы достичь новых целей, но даже самые продуктивные люди прокрастинируют. Просто они об этом не пишут.



Но уметь отдыхать не менее важно, чем уметь работать. Создайте на JavaScript приложение, которое позволит расслабиться и сделать перерыв. Зайдите в него, нажмите на кнопку и получите совет. Пусть в наших соцсетях продуктивным сверхлюдям станет чуть спокойнее.

Не писать код, а:

* Разобраться, о чём поют рэперы
* Почитать новости и ужаснуться в комментах
* Постоять у окна, подумать о жизни
* Подумать о запуске своего подкаста
* Юрий Дудь
* Отправить другу смешную гифку
* Попасть в поток грустных песен и вспомнить все ошибки молодости
* Почитать о пользе медитаций

**Незаменимый инструмент**

В предыдущем задании вы научились показывать пользователю диалоговое окно:

Скопировать кодJAVASCRIPT

alert('В чём сила?');

Но теперь начистоту: когда вы в последний раз видели такие окна? Мы — году в 1998. Командой alert почти не пользуются — встроенные диалоговые окна неклассно выглядят и их стиль нельзя менять.

Мы долго думали, когда alert может пригодиться:

* в процессе разработки, чтобы вывести промежуточный результат на экран;
* для прикола.

Больше ничего не придумали 🤷‍♀️

Постойте. Зачем вообще выводить на экран какую-то информацию, если она не нужна пользователю?



А просто так!



Чтобы поделиться мыслями с браузером.



Чтобы понять, что творится в программе.



Это своего рода исповедь перед браузером.

Для вывода промежуточного результата есть инструмент лучше, чем alert — браузерная консоль. Информацию туда выводят командой console.log:

Скопировать кодJAVASCRIPT

console.log('Сила — в правде.');

# Числа

Программа на JavaScript имеет дело с данными. Она получает их на вход, что-то делает и отдаёт результат. Например, учебный тренажёр Практикума получает ваш код, анализирует его и сообщает, правильно вы выполнили задание или нет.

Для тренажёра код — данные. Список товаров на Алиэкспрессе — данные. Твиты про Илона Маска и количество просмотров в сториз — данные. Поэтому, чтобы создать советы для прокрастинатора, сначала нужно разобраться, с какими данными JS имеет дело.

Данные бывают разных типов. Первый тип, с которым мы познакомимся — числа. В JavaScript тоже можно складывать, вычитать, умножать, делить и возводить в степень:

Скопировать кодJAVASCRIPT

console.log(2 + 2); *// выведет в консоль 4*

console.log(2 \* 2); *// выведет в консоль 4*

console.log(2 - 2); *// выведет в консоль 0*

console.log(2 / 2); *// выведет в консоль 1*

console.log(2 \*\* 3); *// возведение в степень. выведет в консоль 8*

А можно использовать скобки и составлять целые примеры:

Скопировать кодJAVASCRIPT

console.log((2 + 2) / 2); *// фух!*

Шутки шутками, а управляться с арифметикой и числами в JS надо уметь. И этому не так сложно научиться.

console.log(40000+22000); //62000

# Строки

Другой тип данных — это строки. Они позволяют работать с текстом. Вы уже сталкивались со строками:

Скопировать кодJAVASCRIPT

alert('В чём сила?');

*// 'В чём сила?' в этом коде — строка*

Обратите внимание, что строки нужно заключать в кавычки. Они могут быть одинарными или двойными:

Скопировать кодJAVASCRIPT

'роман' *// строка в одинарных кавычках*

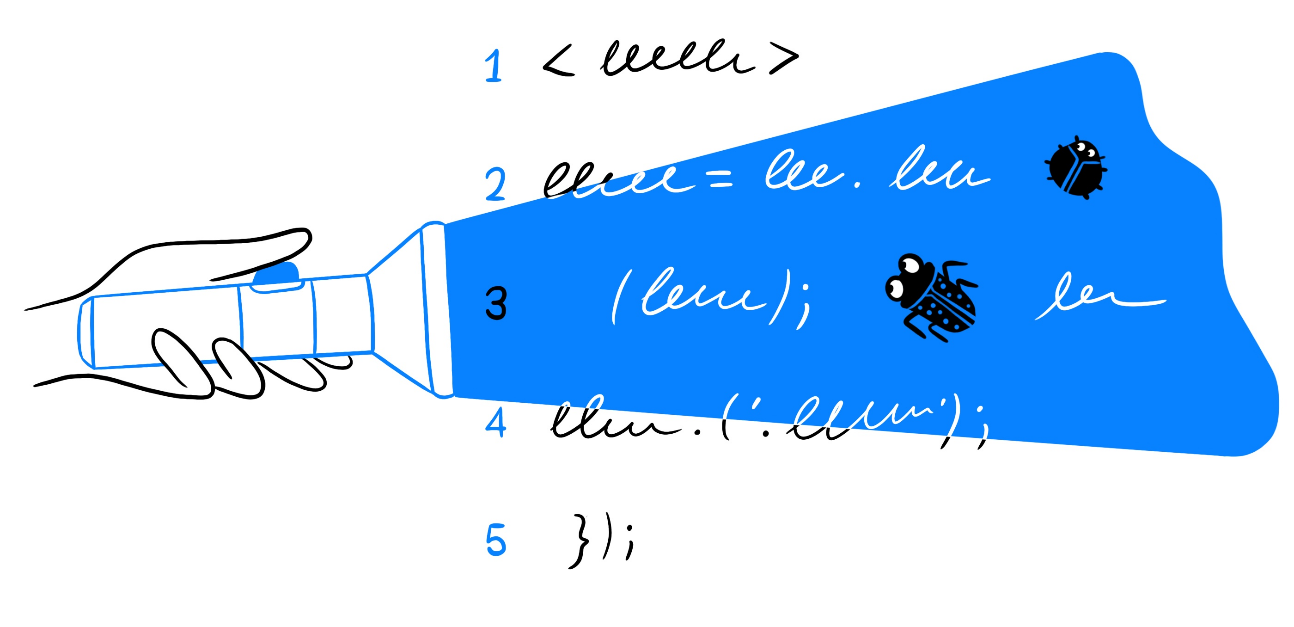
"Джорджа Оруэлла" *// строка в двойных кавычках*

'1984' *// число в кавычках тоже становится строкой*

Можно пользоваться любыми кавычками, но главное — открывающая и закрывающая кавычки должны быть одинаковыми. Разные кавычки приведут к ошибке:

Скопировать кодJAVASCRIPT

"Джордж Оруэлл' // такой код приведёт к ошибке



Кстати, об ошибках. Бóльшую часть времени разработчик ищет ошибки, а не пишет код. Ошибок будет много, и важно относиться к ним правильно. Если код не работает сразу — всё идёт, как надо. Примите это и постарайтесь разобраться в причинах. Если не получится, мы поможем.

# Арифметика строк

Мы освоили два типа данных: числа и строки. Числа мы даже научились складывать. Но что будет, если сложить две строки?

"абыр" + "валг" = ?



"абырвалг"

В JavaScript, если сложить две строки, они склеятся в одну.



Будет ошибка.



"главрыба"

А если так: "3" + "3" = ?



6



"33"

В этом примере "3" и "3" — это строки, так как они обёрнуты в двойные кавычки. Мы уже выяснили, что несколько строк при сложении склеятся в одну. Поэтому "3" плюс "3" даст строку "33", каким бы странным это ни казалось.



Кажется, будет ошибка.

И самый сложный вопрос. Складываем число со строкой: 21 + "2" = ?



"212"

Ошибиться тут несложно. Но дело не в нас, дело в JavaScript. Если сложить строку с числом, JS поступит так, будто складывает только строки — всё склеит в одну. Такие правила.



23



"не212"



Ну здесь-то точно будет ошибка.

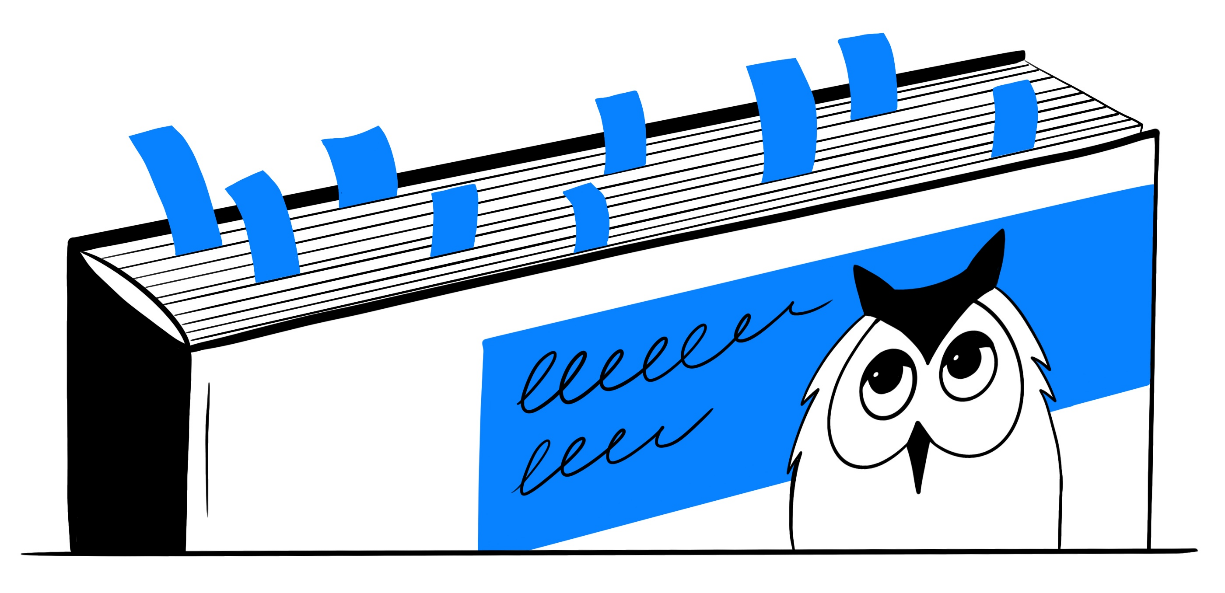
Сложение строк называют «конкатенацией», от какого-то английского слова.

Конкатенацией собирают одну строку из нескольких. Например, через запятую выводят товары, добавленные в корзину интернет-магазина. Или в шапке сайта приветствуют пользователя по имени, объединив строки "Привет, " и "Олег". В общем, довольно часто.

# Переменные

Окей, вы научились работать с числами и строками: проводить подсчёты, даже склеивать строки и выводить результаты в консоль. Но только выводить. Как сохранить этот результат и потом использовать, вам пока не известно.

Представьте: вы читаете книгу, но не можете запомнить, сколько страниц прочли. Умеете читать, но вот запомнить номер страницы не удаётся. Вылетает из головы, где вы остановились, поэтому приходится отсчитывать страницы от начала книги. 10, 24, 140, 250 страниц — каждый раз как в первый.



Так же беспомощны программы, которые не могут сохранять и изменять данные. Именно поэтому настоящее программирование начинается с переменных, запоминающих результаты промежуточных действий.

Переменная работает как подписанная коробка или ячейка, куда можно что-то положить и не потерять. Создают переменную словом let:

Скопировать кодJAVASCRIPT

let pages;

*// создали коробку с именем pages, то есть «страницы»*

После ключевого слова let следует название коробки, то есть имя переменной. Тут есть ограничение: имя нельзя написать на русском или с пробелами — так же как имя пользователя в «Инстаграме».

Вернёмся к чтению. Отложив программирование и ночной сон, вы прочли 210 страниц. Теперь можно сказать:

Скопировать кодJAVASCRIPT

pages = 210;

В созданную ранее коробку pages вы положили значение 210. Писать let не потребовалось, ведь переменная уже была создана раньше.

Теперь пора попробовать переменные в деле.

# Магия присваивания

В предыдущем уроке вы считали число страниц в книге и записывали это число в переменную. Представьте, что взялись за чтение новой книги и уже 45 страниц позади:

Скопировать кодJAVASCRIPT

let pages = 45;

*// создавать переменную и задавать ей значение можно сразу в одну строчку*

Хочется продолжить чтение. Но как быть с изменением количества прочитанных страниц?

Всё хорошо, ведь у переменных есть важное свойство — их значения могут меняться, пока программа работает. JavaScript может найти нужную коробку по её имени, извлечь значение и заменить его другим.

Для наглядности покажем пару базовых ситуаций с изменением переменных.

### I. Прямое переприсваивание

Скопировать кодJAVASCRIPT

*// начали книгу и залпом прочитали 45 страниц*

let pages = 45;

*// прочитали ещё пять страниц. переопределим переменную*

pages = 50;

*// ещё 23 страницы? окей*

pages = 73;

*// надоело читать*

pages = 'ВЖУХ';

Код выполняется сверху вниз. Создав переменную (через let), вы всегда можете обратиться к ней по имени (уже без let) и присвоить новое значение.

### II. Переприсваивание с использованием переменной

Это то самое место, где начинается магия логики. Может показаться странным, но ничего логичнее вы не встречали:

Скопировать кодJAVASCRIPT

*// начали книгу и залпом прочитали 45 страниц*

let pages = 45;

*// прочитали ещё пять страниц. переопределим переменную*

pages = pages + 5;

*// стоп, это как?*

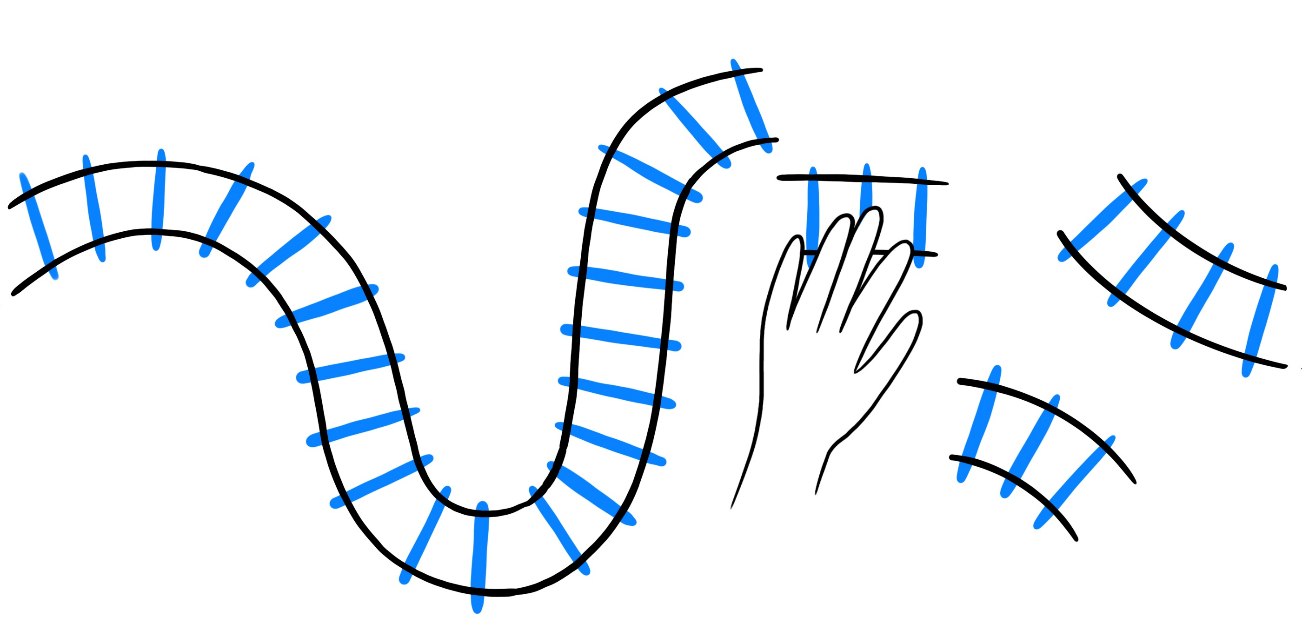
Прочитаем выражение слева направо. Новое значение pages равно старому значению pages плюс 5. Старое значение — 45. Значит теперь pages равно 50.

Используйте это знание, научив навигатор такси изменять оставшееся в пути время.

**Что происходит?**

Первая встреча с JavaScript позади. Можно перевести дух, обдумать проделанное и предстоящее. С одной стороны, вы:

* изучили два типа данных: числа и строки;
* узнали, что представляют собой переменные;
* научились работать с переменными и выводить данные в консоль.



С другой стороны, как всё это применимо в реальной жизни? Сложи два числа, выведи количество страниц в консоль. Хорошо хоть вас не попросили сохранить свои имя и возраст в переменной.

А ведь мы могли бы. Но от таких примеров студенты сбегают со страницы тренажёра в социальные сети.

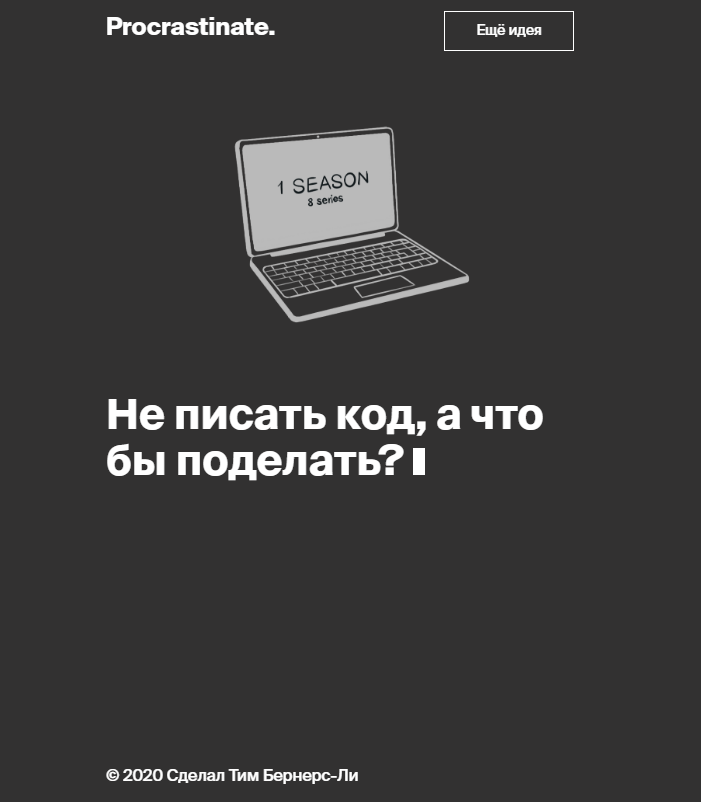
Но первое, что они увидят — имя или никнейм, новости, перепосты — всё это данные, которые оказались на странице благодаря переменным JS.

Переменные преследуют нас даже в социальных сетях, что а) сводит к нулю смысл побега б) подчёркивает особую важность переменных в быту.

Так что подсчёт времени поездки для туристов из Бельгии — дело важное. Можете даже не сомневаться.

Впереди ещё несколько часов обучения. Как итог — генератор идей для прокрастинации.

Не станем кормить вас обещаниями и «завтраками». Вёрстка проекта уже подготовлена, взгляните на то, что получится.



# Мастер прокрастинации

Вроде этот JavaScript вполне посильная штука. Да и конструктор прокрастинации сам себя не сделает.

У вас получится [такой проект](https://code.s3.yandex.net/web-developer/procrastinate/index.html). Настоящее приложение, неприхотливый инструмент для работы и для жизни.

Между этим результатом и вами — всего несколько часов обучения, ваша лень и это видео: <https://www.ted.com/talks/tim_urban_inside_the_mind_of_a_master_procrastinator?language=ru>

Если JavaScript будет одолевать, сделайте глубокий вдох, не менее глубокий выдох — и напишите в поддержку. Поможем.

**Подключите JavaScript к HTML**

Нравится вам JS или нет, на сайтах от HTML не уйти. Помните, как это было?

Скопировать кодHTML

<!DOCTYPE html>

<html>

<head>

<title>Заголовок страницы</title>

</head>

<body>

</body>

</html>

Чтобы соединить скрипт и HTML-страницу, код можно писать:

* внутри тегов <script></script>, которые мы рекомендуем помещать прямо перед закрывающим </body>,
* в отдельном файле, ссылаясь на него в HTML-документе:

Скопировать кодHTML

<script src="путь к файлу"></script>

Очень похоже на подключение CSS внутри <head></head>, но имейте в виду, что:

1. тегу <script> всегда нужен закрывающий </script>, в отличие от <link>;
2. путь к JavaScript-файлу задаётся атрибутом src (как с картинками), а не href (как с гиперссылками).

Теперь подключите файл script.js к вёрстке всего проекта.

**Массивы**

JavaScript-файл подключён. В нём будет код проекта о прокрастинации. Сценарий его работы таков:

1. пользователь гадает, чем бы заняться, и нажимает на кнопку;
2. программа выбирает случайный ответ из базы;
3. подставляет ответ в нужное место на сайте.

Представьте себе с десяток возможных ответов, обычных строк: 'Поваляться', 'Слушать подкаст', 'Медитировать'. Строки могут храниться в разных переменных. Но выбирать удобнее, если они собраны в одном месте под общим именем. Для этого придуман массив — ещё одна структура данных в JavaScript.

В массиве элементы перечисляются через запятую, все они заключены в общие квадратные скобки. Обычно они однотипны, как товары, которые пользователь добавил в корзину на Алиэкспресс:

Скопировать кодJAVASCRIPT

let aliExpress = ['Лазерная указка Xioamo', 'Гарнитура в виде телефонной трубки', 'Форма для льда «Титаник»'];

Каждый элемент массива имеет свой порядковый номер — индекс:

Скопировать кодJAVASCRIPT

console.log(aliExpress[1]); *// 'Гарнитура в виде телефонной трубки'*

Хм, индекс 1 выдал не первый элемент массива, а второй. Ошибки здесь нет — счёт в массивах начинается с нуля. У первого элемента индекс нулевой:

Скопировать кодJAVASCRIPT

console.log(aliExpress[0]); *// 'Лазерная указка Xioamo'*

# Случайные числа

Чтобы получить определённый элемент массива, нужно имя массива и индекс элемента.

Но как выбрать случайный элемент массива? С этим вопросом не мы первые, не мы последние: на JavaScript написано немало игр, приложений и программ, в которых происходит выбор случайных значений.

Для генерации случайных чисел есть команда Math.random(). Она возвращает случайное число между 0 и 1, включая ноль:

Скопировать кодJAVASCRIPT

let randomNumber = Math.random();

console.log(randomNumber); *// например, 0.9752705074780903*

Да, числа в JavaScript необязательно целые. К дробям применимы все те же арифметические операции:

Скопировать кодJAVASCRIPT

console.log(0.12 + 0.34); *// 0.46*

Но вернёмся к случайности. Мы выяснили, что промежуток, в котором работает Math.random() — от 0 до 1. Но что если нужно случайное число от 0 до 10? Придётся вспомнить арифметику.

Как получить случайное число от 0 до 10?



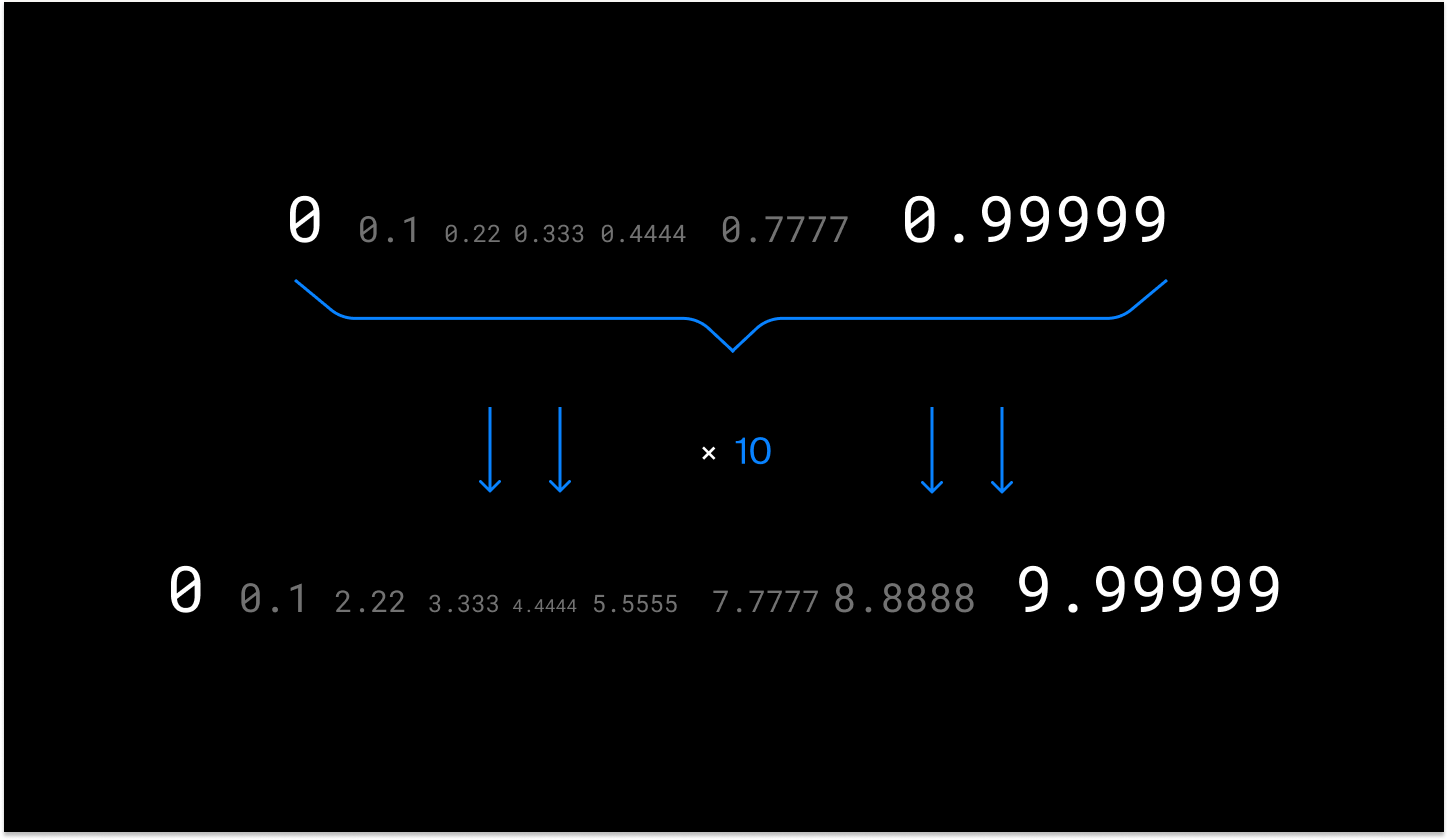
Math.random(0, 10);

Попытка хорошая, но Math.random() так не работает. Эта команда не реагирует на то, что вы пишете в её круглых скобках.



Math.random() \* 10.

Да. Разберёмся, почему это правильный ответ.



Math.random() генерирует число от 0 до 0.99999999999. Получается, что самое маленькое число, которое может быть сгенерировано — ноль, самое большое — почти 1, но не 1. Поэтому Math.random() \* 10 генерирует числа от 0 до 9.9999999999.

Вы можете умножать Math.random() на количество элементов в массиве и получать случайное число. Попробуете?

# Математика ниже плинтуса

Math.random() практически всегда генерирует дробное число. Соответственно, Math.random() \* [длина массива] — тоже дробное число, а вовсе не случайный индекс массива.

Но дробные числа можно округлить до целых. Для этого как раз есть команда Math.floor(). В круглые скобки она принимает нецелое число и округляет его вниз, то есть отбрасывает все цифры после точки:

Скопировать кодJAVASCRIPT

let randomNumber = Math.random() \* 10;

let randomNumberInt = Math.floor(randomNumber);

console.log(randomNumber); *// 9.315448768805199*

console.log(randomNumberInt); *// 9*

Можно записать и в одну строку:

Скопировать кодJAVASCRIPT

let randomNumberInt = Math.floor(Math.random() \* 10);

Здесь в переменной randomNumberInt окажется число от 0 до 9. Почему не до 10? Потому, что Math.random() генерирует число от 0 до 1, включая 0 и не включая 1. Стало быть, самое большое число, которое она может сгенерировать, скажем, 0.99999999999 при умножении на 10 даст 9.99999999999, а при округлении вниз — 9.

Математика — царица наук.

Джаваскрипт может лучше. Здесь у массива есть свойство length, оно хранит его длину. Чтобы её получить, нужно после имени массива поставить точку и написать length:

Скопировать кодJAVASCRIPT

console.log(aliExpress.length) *// покажет длину массива*

Замените цифру 6 в коде на длину массива phrases

# Функции

Вот код, приветствующий пользователя по имени:

Скопировать кодJAVASCRIPT

alert('Привет, Андрей');

Если такое приветствие понадобится один раз, можно поместить этот код там, где он нужен. Но если здороваться нужно неоднократно, лучше создать функцию.

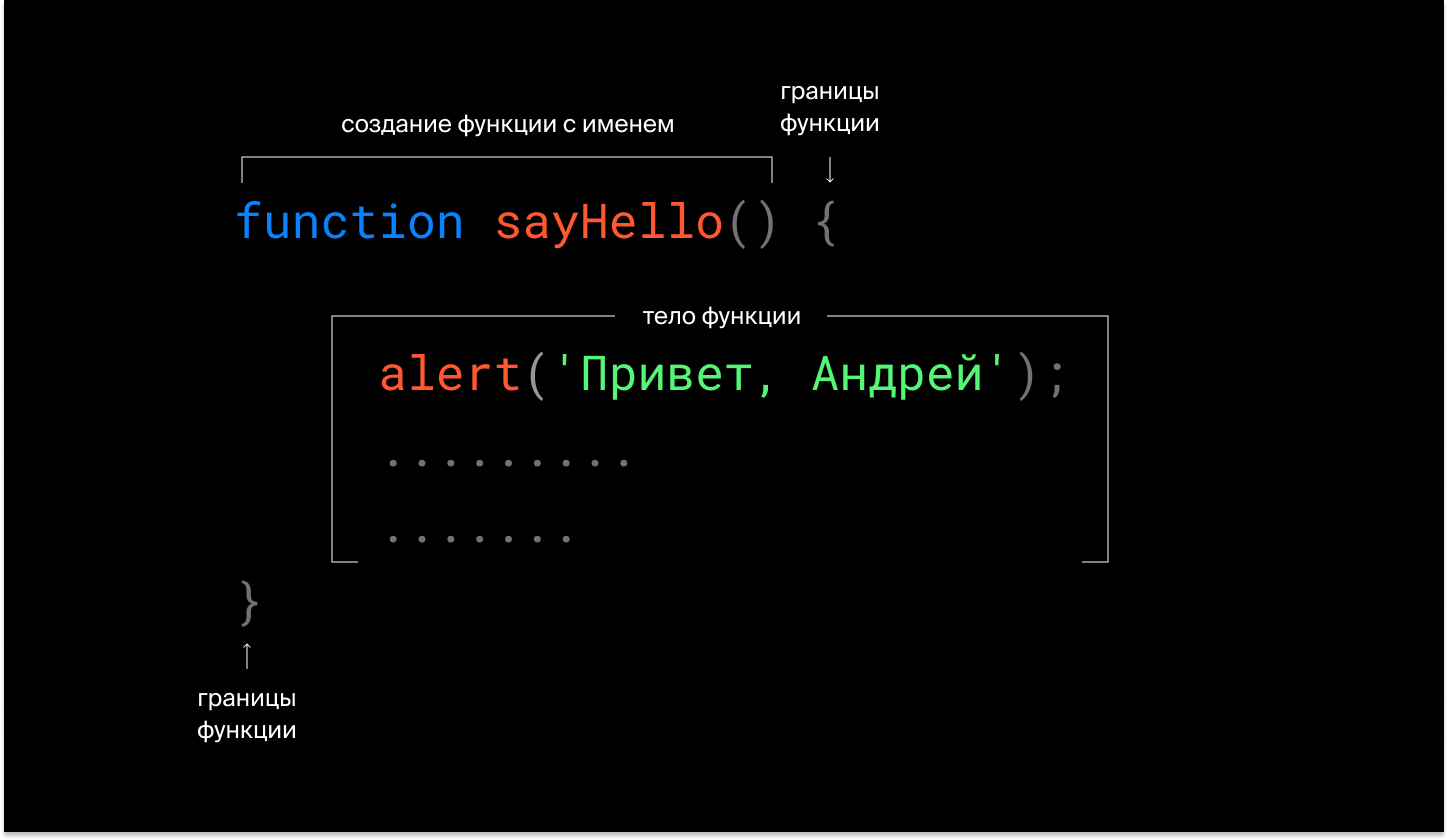
Функция — это блок кода, который можно переиспользовать много раз. Создают функцию ключевым словом function:

Скопировать кодJAVASCRIPT

function sayHello() {

}

sayHello — это имя функции. Его придумываем мы, оно может быть любым. Между фигурными скобками пишут код функции или «тело функции»:



Скопировать кодJAVASCRIPT

function sayHello() {

alert('Привет, Андрей'); *// тело функции*

}

*// обратите внимание, как оформлять код функции — в несколько строк*

Если в городе появился сервис по доставке пиццы — это не означает, что пицца уже на пути к вам. Пиццу нужно заказать, «вызвать».

То же и с функцией. Создавая функцию, вы создаёте лишь инструкцию — что будет сделано, если к функции позже обратятся по имени.

Чтобы использовать функцию, нужно написать её имя и поставить круглые скобки:

Скопировать кодJAVASCRIPT

function sayHello() {

alert('Привет, Андрей');

}

sayHello(); *// Привет, Андрей*

Такое использование называется вызовом, говорят: «вызвать функцию».

В случае с приветствием нужность функции может быть неочевидна. Но если тело функции занимает много строк кода, эти строки не придётся писать раз за разом. Достаточно лишь вызвать функцию одной строкой — в этом вся соль.

Кстати. Вызов функции sayHello ничего вам не напоминает?



Вызов функции sayHello ничего мне не напоминает.

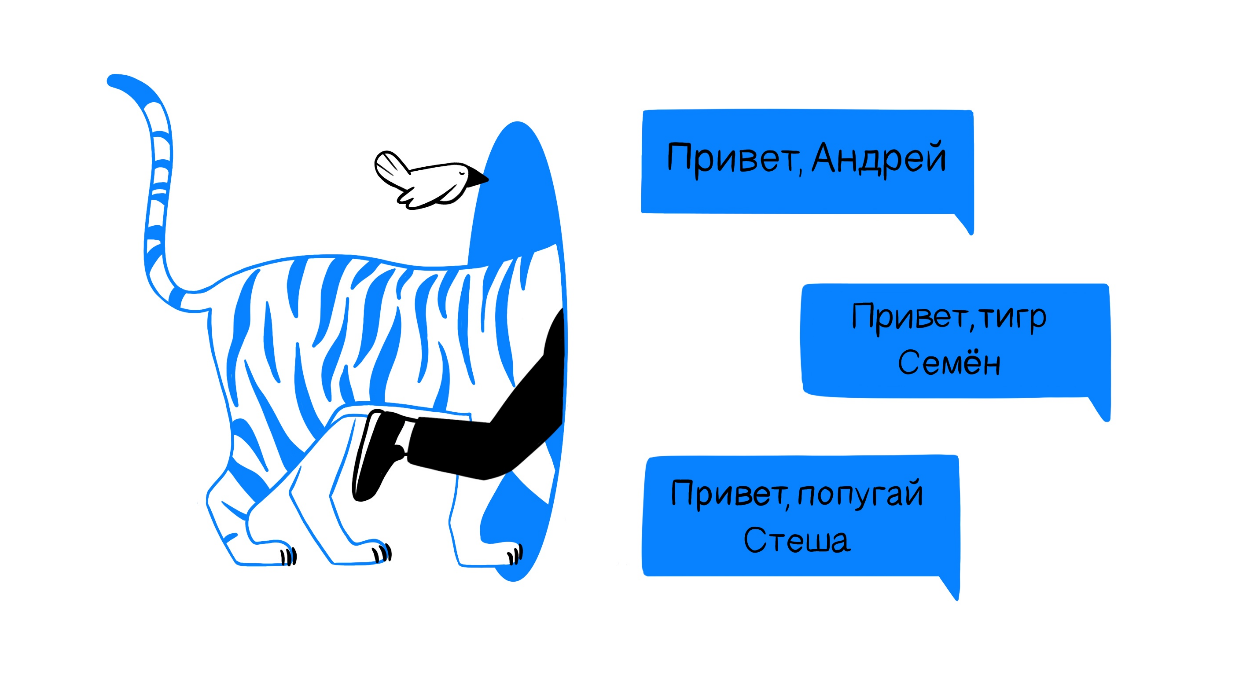
Ничего страшного. Смотрите подпись ко второму ответу — увидите сходство.



Когда мы используем console.log, alert или Math.random, то тоже пишем круглые скобки. Как при вызове sayHello.

alert и console.log — это тоже вызовы функций. Просто эти функции созданы разработчиками браузера и их код спрятан внутри. А sayHello — функция, созданная нами, но принцип один.

# Функции с аргументами



Функция sayHello должна приветствовать пользователя:

Скопировать кодJAVASCRIPT

function sayHello() {

alert('Привет, Андрей');

}

Если всех пользователей сайта зовут Андрей, беспокоиться не о чем. Но если на сайт зайдёт Анна, мы будем близки к провалу.

Этого можно избежать:

Скопировать кодJAVASCRIPT

function sayHello(name) {

alert('Привет, ' + name); *// name — параметр*

}

После функции есть скобки — в них можно перечислять параметры. При объявлении функции параметр внутри неё играет роль переменной, чьё значение пока неизвестно.

При вызове функции ей можно передать конкретное значение — аргумент. Он будет подставлен вместо переменной name:

Скопировать кодJAVASCRIPT

sayHello('Анна'); *// "Привет, Анна"*

sayHello('Михаил'); *// "Привет, Михаил"*

Функция sayHello стала универсальной. Благодаря аргументам можно приветствовать всех.

Более сложные функции зависят не только от одного, но и от двух, трёх, четырёх, и даже пяти, шести, семи (ну вы поняли) параметров

# Возвращаемое значение

Функции могут намного больше, чем выводить сообщения в консоль. Самое главное, что функция превращает входные данные в результаты. Представьте соковыжималку. В принципе она выполняет функцию:

Скопировать кодJAVASCRIPT

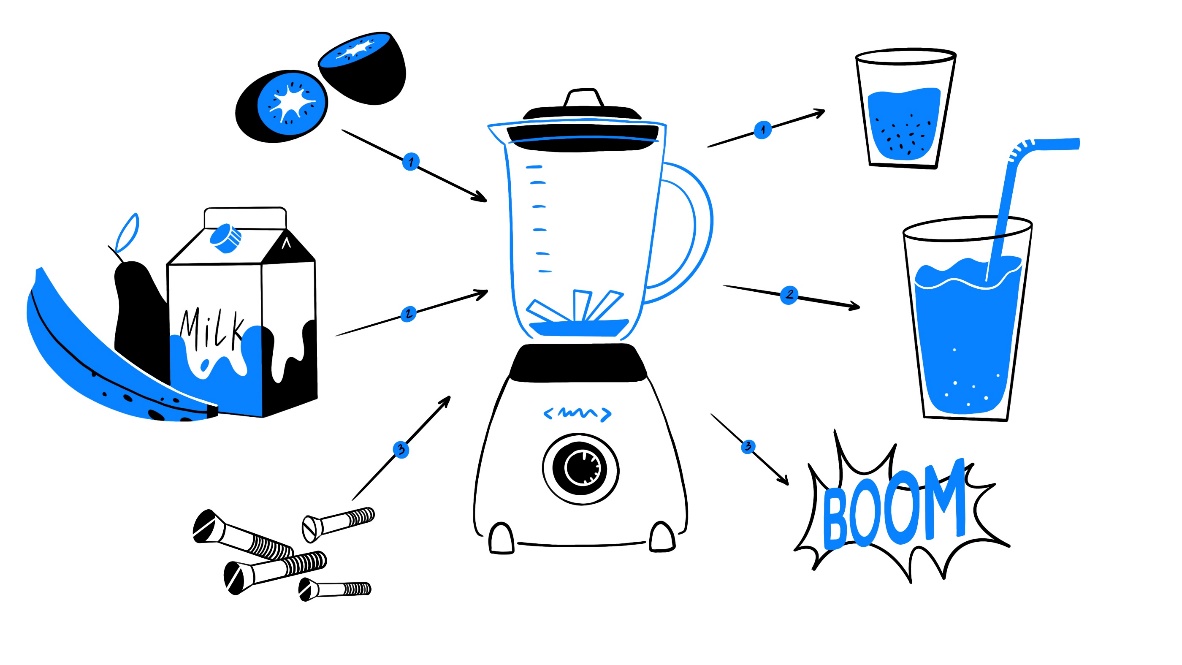
сделатьСок(фрукт) {

вернуть сок;}

Программисты сказали бы: соковыжималка принимает фрукт и возвращает сок. Она не определяет, куда вы этот сок перельёте — в стакан, пластиковую бутылку, неперерабатываемую банку или в карман.

Так и функциям в JavaScript обычно поручают преобразовывать данные, но не определять, где они применяются (в консоли, во всплывающем окне, в других функциях — вариантов слишком много). Нужно просто вернуть эти данные скрипту для дальнейшего использования. Возвращаемые значения указывает оператор return.

Вот так функция sayHello возвращает результат. Его можно присвоить переменной и использовать где угодно:



Скопировать кодJAVASCRIPT

function sayHello(name) {

return 'Привет, ' + name;}

let greeting = sayHello('Венера');

console.log(greeting); *// "Привет, Венера" — вывели в консоль*

alert(greeting); *// "Привет, Венера" — показали в модальном окне*

Если в другом месте программы приветствие понадобится снова — например, не только в консоли, — мы сможем вызвать ту же функцию и не потерять результат.

# Выбор и изменение элементов страницы

Чтобы работать с элементами страницы, в JavaScript есть специальный ящик, он называется document. В этом ящике — вся информация о веб-странице: URL-адрес, таблицы стилей, кодировка, гиперссылки, все тексты. И да, через document можно получить доступ к управлению любым элементом на странице.

Раньше вы умели изменять элементы только средствами HTML и CSS. Но элементами можно управлять и через JavaScript — надо только их найти, например, по названию класса элемента.

Помимо ящиков в JavaScript есть инструменты, которыми можно менять всё, что оказалось в ящиках — не только в document.

Например, инструмент querySelector. Он работает как функция и принимает на вход селектор класса элемента, к которому вы хотите получить доступ:

Скопировать кодJAVASCRIPT

let photoElement = document.querySelector('.photo');

*// доступ к инструментам осуществляется через символ точки*

*// инструмент querySelector нашёл на странице нужный элемент*

Так в переменную photoElement попадёт элемент с классом photo. Теперь JavaScript может изменить содержимое элемента, его стиль и поведение. Как это сделать? Разберёмся на практике.