**HTML**

1. Теги

Одиночные без закрывания

Img

* [meta](https://doka.guide/html/meta/) — инструкции и информация для браузеров, про кодировку символов например;
* [link](https://doka.guide/html/link/) — ссылка на внешний файл, со стилями, например.

Набор тегов, которые обязательно должны быть на любой странице:

Скопировать кодHTML

<!DOCTYPE html>

<html>

<head>

</head>

<body>

</body>

</html>

Внутри тегов <head> описывают правила, по которым браузер читает страницу. Внутри <body> — то, что видит пользователь.

2. Атрибуты

Для некоторых тегов атрибуты обязательны. Чтобы разместить изображение, в теге <img> нужно прописать атрибут src. Он указывает путь до файла с картинкой, без него тег не заработает.

Атрибуты кнопки: <https://doka.guide/html/button/#atributy>

3. Классы

Классы удобны при работе со стилями, потому что можно обращаться сразу к нескольким элементам по одному имени

**CSS**

Чтобы код заработал, надо подружить CSS и HTML. Внутри <head> напишите:

<!DOCTYPE html>

<html lang="ru">

<head>

*<!-- всевозможные теги meta итд -->*

<link rel="stylesheet" href="очень-оригинальное-и-классное-а-главное-ёмкое-название-вашего-файла.css">

</head>

Правила CSS состоят из селекторов, свойств и значений:

* .my-class — это селектор, он выбирает все элементы с классом my-class;
* color — свойство, которое задаёт цвет текста;
* red — значение, которое определяет, что цвет текста будет красный.

.my-class {

color: red;

}

1. Селекторы

Селекторы по тегу: <https://doka.guide/css/tag-selector/>

Селекторы по классу. Пример:

.list {

font-weight: 700;

}

Селекторы вложенности. Их используют, когда нужно выбрать все вложенные элементы:

*/\* все p внутри div \*/*

div p {

margin-left: 20px;

}

*/\* все элементы с классом app внутри div \*/*

div .app {

font-weight: normal;

}

Комбинированные селекторы. Их используют, когда нужно выбрать специфичный элемент, например, блок с двумя классами или ссылку с определённым параметром:

*/\* ссылки с классом source \*/*

a.source {

color: #fd24af;

text-decoration: none;

}

*/\* элементы с классами app и best-choice одновременно \*/*

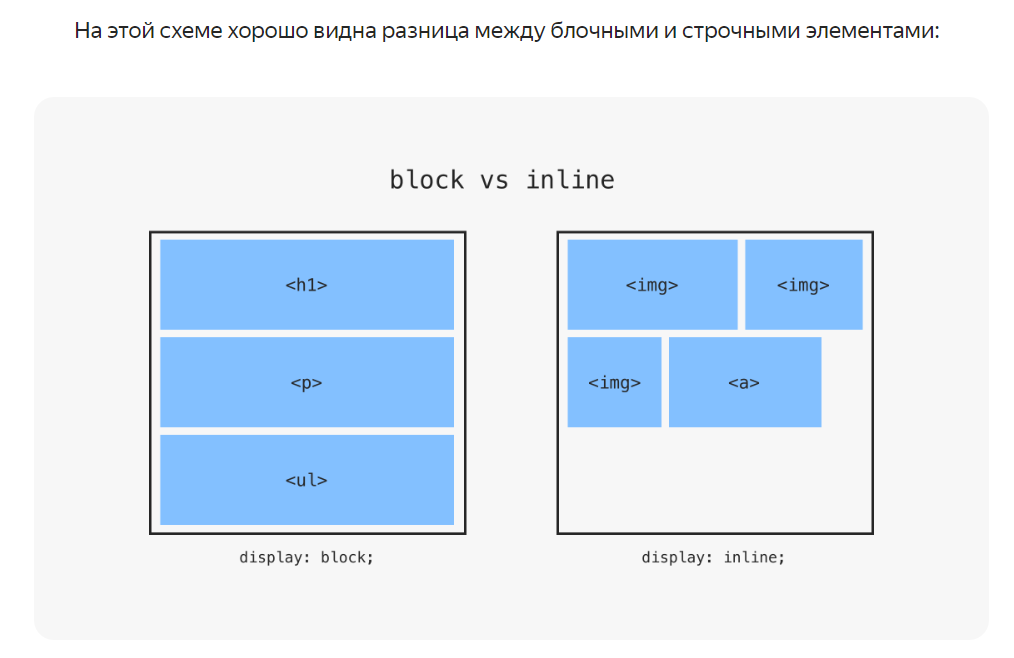
.app.best-choice {

font-style: italic;

}

**Элементы в HTML**

1. **Условно строчные элементы. Например, ссылка внутри параграфа.**
2. **Условно блочные элементы. По умолчанию занимают всю ширину своего родителя и располагаются на новой строке. Например, абзацы, контейнеры <div> и заголовки.**



Однако поведение элементов можно изменить — за это отвечает свойство display. Строчные элементы могут вести себя как блочные, и наоборот

Строчным элементам нельзя задавать ширину и высоту, а блочные не могут стоять на одной строке. Но в вёрстке это часто нужно, когда у нас есть несколько колонок с текстом, например. Для этого есть строчно-блочный тип.

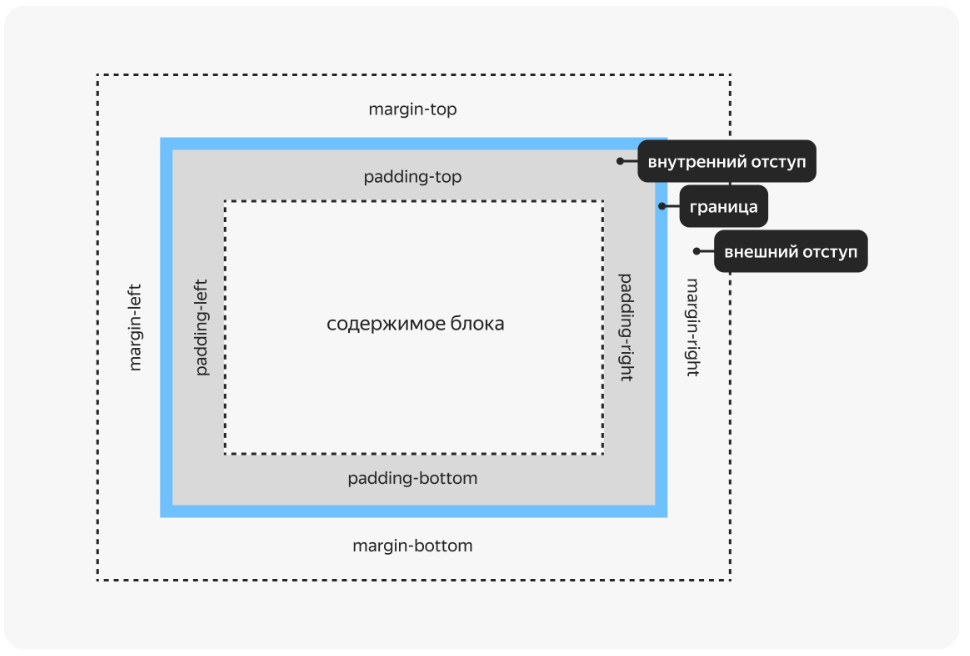
p {

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

}

Вертикальным выравниванием можно управлять. Например, задавать выравнивание по верхней строчке

**Блоки**



Размеры блока задают шириной и высотой.

Можно задать точные размеры в пикселях или использовать относительные величины:

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

1. **Границы** можно задавать отдельными свойствами каждый параметр границы:

* border-width— толщина,
* border-style— стиль,
* border-color — цвет.

Но все пишут в одну строчку через пробел, при этом обязательным является второе свойство — border-style:

div {

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

}

1. **Отступы**

Есть внешние отступы, или поля, — пространство между элементом и его соседями. Они задаются свойством margin

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;

}

Можно установить margin-left и margin-right в значении auto, чтобы центрировать блок по горизонтали.

Когда мы задаём свойству margin значение auto, то отдаём всё свободное пространство.

В блочной модели auto не сработает для вертикального выравнивания, его можно устанавливать только слева и справа.

Внутренний отступ padding отодвигает контент внутри блока от границы. В отличие от margin, у padding нет специального значения auto и отрицательное значение задать нельзя.

(!) *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*

**ТЕКСТ И ШРИФТ**

**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 краю — right, по центру — center.
* text-decoration — оформление текста. Часто используют подчеркнутый текст — underline и неоформленный — none.
* text-transform — перевод текста в другой регистр: верхний — uppercase , нижний — lowercase или каждое слово с заглавной буквы — capitalize.

**Шрифт**

* font-family — семейство шрифтов. Можно перечислять шрифты через запятую, чтобы браузер искал среди них установленный. Этому свойству будет посвящена целая тема в следующем спринте
* font-size — размер шрифта.
* font-weight — начертание: полужирное — bold, обычное — normal и тонкое —lighter.

**ФОН**

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

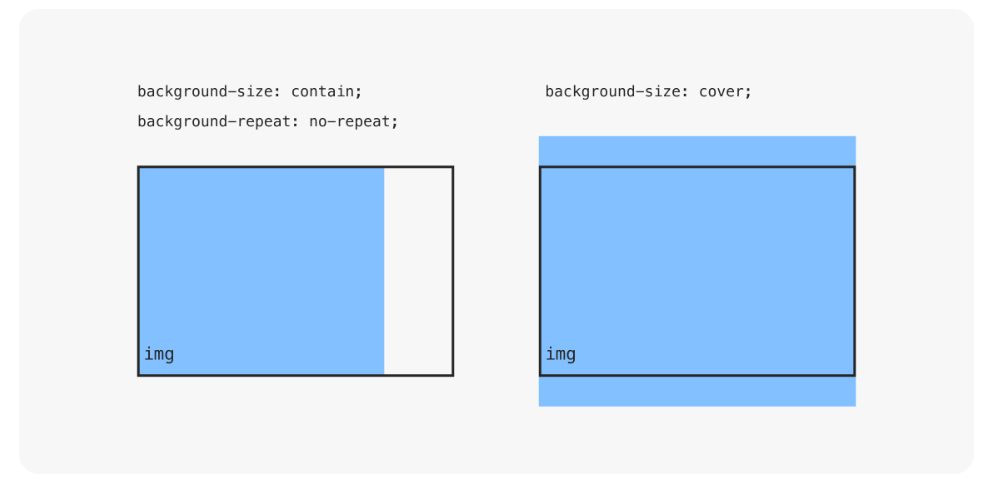
div {

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

}

background-size — размер фона. Часто используют значения:

* contain — фоновое изображение целиком вписывается в блок;
* cover — фоновое изображение занимает всю доступную площадь блока. То, что не влазит в блок, — обрезается, пользователи эти части изображения не увидят.



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 выравнивания.

Скопировать кодCSS

div {

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

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

}

**ФИГМА**

Есть ещё режим редактирования. Чтобы перейти в него, нужно зарегистрироваться в Figma, а потом нажать стрелочку рядом с именем файла и выбрать Duplicate to your drafts.

**1. Отступы**

Потом наведите мышь на соседние блоки. Figma посчитает отступы от выделенного элемента до того, на котором находится курсор, и покажет расстояние в пикселях. Если хотите узнать расстояние до группы элементов, то зажмите **[**Ctrl**]** (Windows) или **[**⌘] (macOS).

В режиме редактирования тоже нужно выделить элемент или группу, а на соседние элементы навести курсор при зажатой клавише **[**Alt**]** (Windows) или **[**Option**]** (macOS).

**2. Текст и цвета**

В Figma можно работать не только с отступами, но также с текстами и цветами.

**В режиме просмотра** кликните по тексту. В выпадающем списке справа выберите “CSS”. Увидите знакомые до боли свойства и их значения.

**В режиме редактирования** кликните по элементу. В столбце справа появятся три вкладки:

* DESIGN открывается рабочая панель отдельного элемента. Отсюда можно управлять его положением и способом выравнивания, настраивать цвета, границы и специальные эффекты. Здесь же в блоке TEXT собрана информация о тексте;
* PROTOTYPE здесь дизайнер может эмулировать события: клики по элементу, наведение мыши, непрерывное нажатие;
* INSPECT происходит то же, что в режиме просмотра — отображаются свойства и значения.

**3. Код**

Figma и подобные инструменты позволяют копировать и использовать готовый CSS-код. Но с ним надо уметь обращаться.

Figma предлагает такой код, который вырывает элемент из потока и делает его «невидимым» для других элементов на странице. Этот подход хорош для статичной графики, но плох для динамичного макета в браузере.

Вдобавок всем элементам задаются точные координаты относительно чётко заданного размера холста. Если экран пользователя будет отличаться по размерам от макета в Figma, вся вёрстка «расползётся».

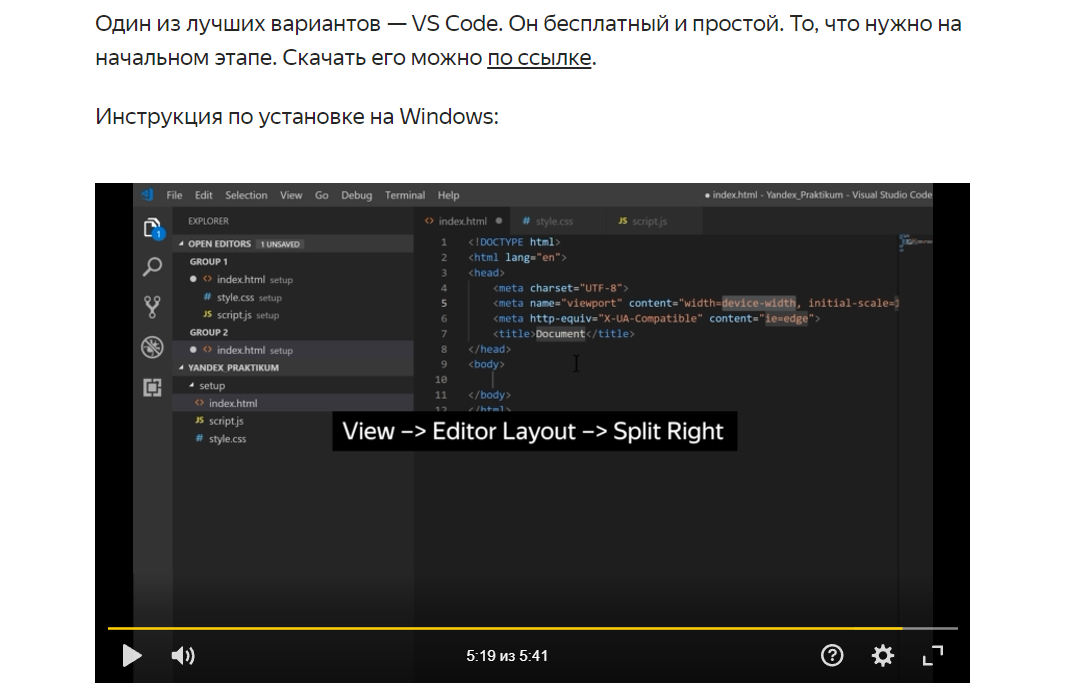
Будьте осторожны и с копированием значения высоты. Блок с фиксированной высотой может не вместить предназначенный для него контент.

В общем, тот самый момент, когда решение всех проблем одним способом вызывает только массу новых проблем.

Лучше копировать из графического редактора **только**:

* всё, что касается шрифтов: размер, межстрочное расстояние (line-height), цвет;
* всё, что касается фона и декорирования: цвет, фильтры, тени, градиенты.

**VC Code**



**Плагины для VC Code**

**LiveServer**

## Auto rename tag

[W3C Web Validator](https://marketplace.visualstudio.com/items?itemName=CelianRiboulet.webvalidator) – что-то не очень работает. Проверить или заменить.

[Prettier](https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode) . Он исправляет все помарки оформления и сэкономит ваше время.

Чтобы отформатировать код, нужно в любом месте документа нажать правой кнопкой мыши, выбрать пункт «Форматировать документ с помощью…» и в выпадающем списке выбрать Prettier. И вуаля, ваш код как с картины сошёл!

# Комментарии в коде

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

В CSS между /\* и \*/

Владимир Рыбин приглашает вас на запланированную конференцию: Zoom.

Тема: Зал персональной конференции Владимир Рыбин

Войти Zoom Конференция

https://us05web.zoom.us/j/9910290923?pwd=SzFqMjBiY1pyZ1FmN3FiYWtqczZsQT09

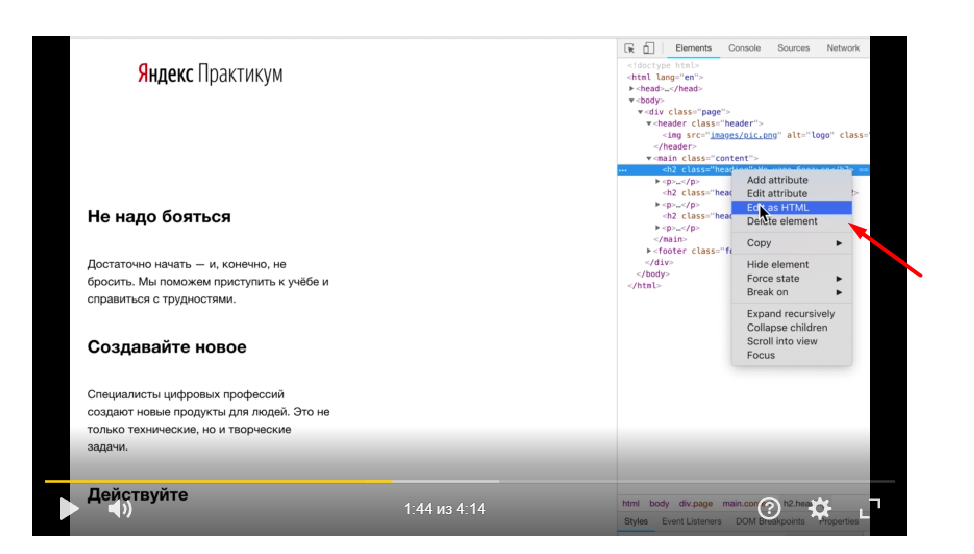
Идентификатор конференции: 991 029 0923

Код доступа: K7UT5e

DevTools

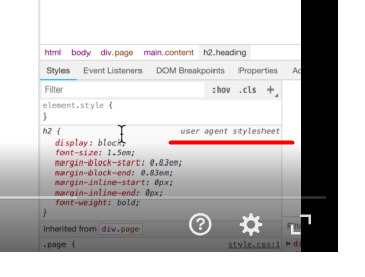
браузере и найдите вкладки с HTML, CSS и консолью.

Редактировать html



Редактировать CSS

Стили, которые браузер применяет по-умолчанию



В инструментах разработчика браузерные стили называются «Таблица стилей агента пользователя» или «User agent styles».

Изменить свойства элемента



Во вкладке Console работа с JS

Можно прям там писать код и команды +смотреть ошибки.

**User agent styles**

Иногда при вёрстке **браузерные** стили мешают и не дают сделать как надо. Существует несколько подходов для решения вопроса с браузерными стилями.

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/>

## Нормализовать стили

## Нормализовать стили — значит сделать так, чтобы элементы вели себя одинаково в разных браузерах.

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 из интернета: <https://necolas.github.io/normalize.css/>

**Использовать грубую конструкцию с универсальным селектором**

В некоторых проектах в начале CSS вы можете встретить блок с таким или бОльшим набором правил:

Скопировать кодCSS

\*,

\*::before,

\*::after {

margin: 0;

padding: 0;

box-sizing: border-box;

}

Да, это может показаться удобным: одним махом сбросили всё, что нам не нравится у всех элементов. Но это как экскаватором копать грядки.

## !!!!

## Пожалуйста, не используйте в учебных проектах reset.css, normalize.css или правила с универсальным селектором. Переопределяйте стили отдельных элементов точечно.

**Подход Pixel Perfect**

А как может оценить работу верстальщика дизайнер, который нарисовал макет? Ведь он наверняка «подгонял» все отступы и размеры с любовью и вниманием. И захочет, чтобы и в вёрстке всё было так же.

Для этого используют подход «Pixel Perfect» — сверку вёрстки с макетом на предмет расхождения размеров, отступов, положений элементов.

.pixel-perfect-image{

position: fixed;

top: 50%;

left: 50%;

opacity: 0.5;

filter: sepia(0.6);

transform: translate(-50%, -50%);

}

**Приём 1: Добавляем макет в разметку**

Можно прямо в разметке разместить тег <img> с картинкой макета и добавить ему следующие стили:

Скопировать кодCSS

.pixel-perfect-image {

position: fixed;

top: 0;

left: 50%;

transform: translateX(-50%);

opacity: 0.5;

filter: sepia(0.6);

}

**Приём 2: Используем плагин PerfectPixel**

Есть специальный плагин, который помогает наложить картинку на страницу без дополнительных элементов и стилей. Он называется **PerfectPixel**. Скачать его можно с официального сайта → <https://www.welldonecode.com/perfectpixel/>

Откройте вёрстку в браузере, в котором установлен плагин. Откройте плагин и загрузите туда наш макет.

Почитать больше про Pixel Perfect можно в Доке → <https://doka.guide/css/pixel-perfect/>

# Абсолютный путь и тег <link>

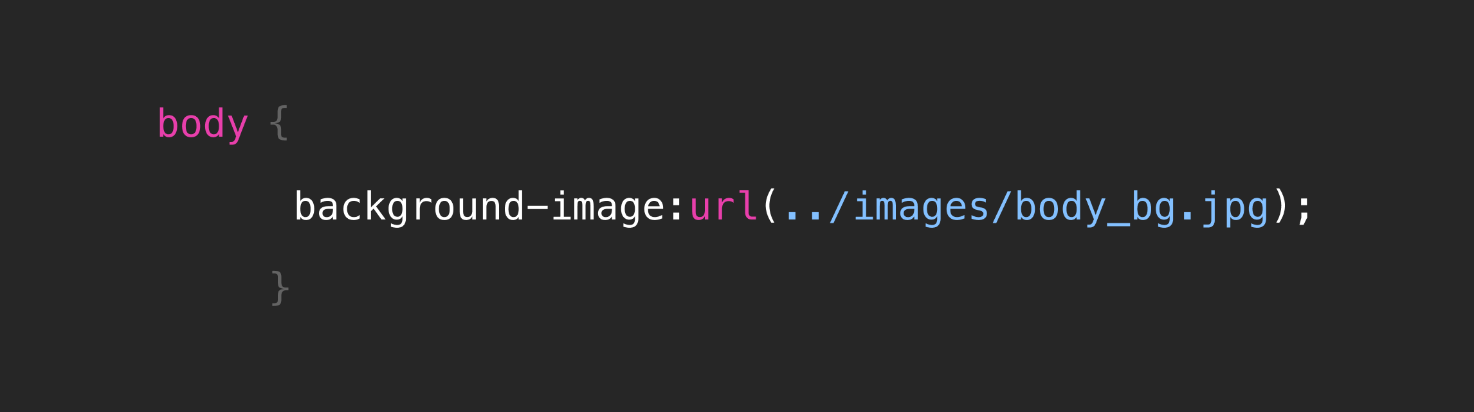
Тег <link> позволяет обращаться к файлам в интернете, указывая ссылку на них в атрибуте href.

Ресурсы с кодом библиотек называют CDN — Content Delivery Network (англ. сеть доставки контента). К файлам на CDN также делают запрос через <link>.

# Относительные пути к файлам

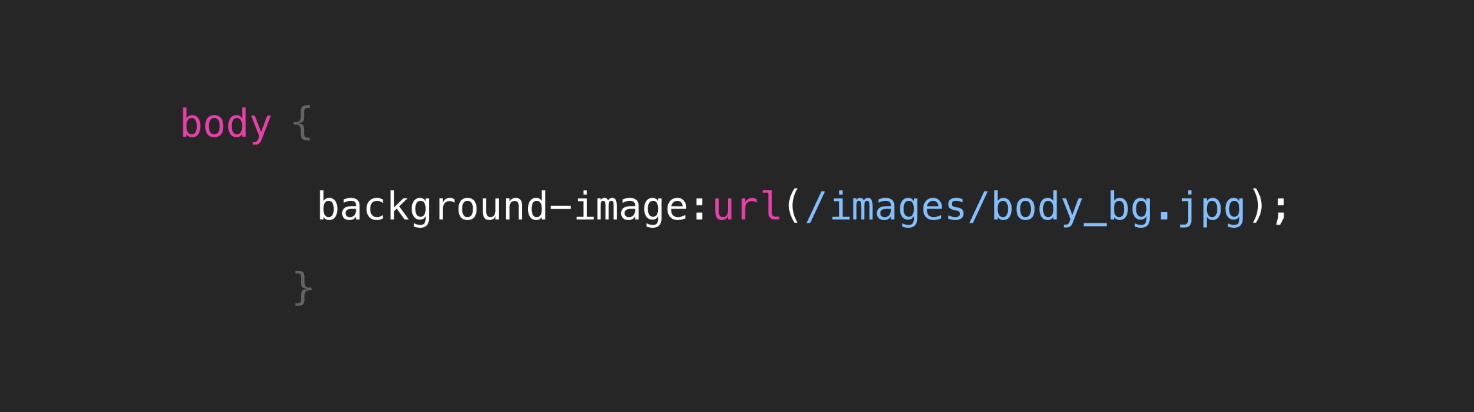
./ — указание на каталог, в файле которого мы находимся;

../ — указание на каталог одним уровнем выше.



Другой вид относительных путей — от корневого каталога проекта. К нему нас возвращает указатель /.

Из любого файла проекта мы можем обратиться к другому файлу, начав свой путь от корневого каталога.



# CSS-директивы. @import

Элемент <link> — не единственный способ подключения файла к проекту. В этом уроке мы разберём подключение файлов директивами.

Директивы — инструкции для CSS, которые меняют отображение или поведение элементов на странице. Ими можно подключать CSS-код и шрифты, создавать анимации. Обязательная часть директивы состоит из «собачки» @ и названия директивы:

Директив в CSS много, у каждой свой синтаксис. Сейчас разберём, как работает директива @import. А другие директивы нам ещё встретятся.

Директива @import позволяет включить внешние файлы, такие как шрифты или CSS, в текущую таблицу стилей.

Кроме того, @import даёт возможность скрепить несколько CSS-файлов в один, как в конструкторе LEGO. Или разделить один большой CSS-файл на несколько обособленных, которые легче поддерживать и повторно использовать. В большом проекте может быть десяток тысяч строк CSS-кода — без разделения CSS никак.

Синтаксис: @import url(main.css);

Объявляя директиву, указывают путь до запрашиваемого файла внутри знакомой конструкции url(). Обратите внимание, что после директивы нет двоеточия, но точка с запятой в конце обязательна.

У @import есть ряд особенностей:

* CSS работает с @import так, что каждая обрабатываемая директива загружается по очереди. Это может плохо отразиться на скорости загрузки сайта. Элемент <link> позволяет загружать файлы параллельно.
* @import будет работать только когда написана в начале файла. Внешние файлы загружаются первыми. Следующие за ними стили могут их перезаписать.