# Структура файлов, пути к файлам. Введение

При вёрстке разработчики создают много файлов — HTML, CSS, JS. Они используют изображения, файлы шрифтов и видео. В этой теме разберём, как файлы организованы внутри компьютера. Когда ваши проекты станут большими, количество файлов и папок резко увеличится. Вам нужно будет устанавливать связи между ними.

Любую информацию нужно как-то организовать. Например, карточки пациентов в поликлинике хранят в коробках — по первой букве фамилии. Коробки на шкафах расположены в алфавитном порядке, шкафы поделены по кварталам.

Такая структура позволяет быстро найти карточку пациента. Сначала определяем шкаф по адресу, затем — коробку, а в ней карточку по фамилии пациента. Это легко, ведь мы понимаем логику, по которой разложены карточки.

Данные внутри компьютера тоже организованы в структуру: файлы хранятся в папках, папки — в других папках. Важно эту структуру понимать, чтобы легко находить файлы и устанавливать между ними связи. Например, определять, как файлу с разметкой найти нужный файл стилей.

В этой теме мы разберём, из чего состоят файлы, и расскажем о типичных файловых структурах. После этого будем связывать различные файлы внутри проекта. Не бойтесь, морские узлы изучать не придётся.

# Файл

У вас в тренажёре вкладки с кодом на разных языках: HTML, CSS и JS. Это достаточно удобно — всего несколько вкладок, между которыми вы переключаетесь. Не нужно тратить время на поиск вкладки, вы сосредоточены на написании кода. В настоящих проектах разработчик сам должен решить, где расположить style.css, а где — index.html. Разобраться поможет организация файлов при вёрстке.

Сначала разберём, что такое файл. Это набор данных. Пользователь или программа могут записать, изменить или считать такие данные. Файлы занимают определённое количество памяти устройства, на котором хранятся. Например, на жёстком диске компьютера.

У каждого человека есть имя — у файлов тоже есть имена. Операционная система обращается к файлам по именам, которые могут повторяться.

Ещё у файлов обычно есть «фамилии» — расширения. Расширение помогает системе определить, как исполнить файл. К примеру, разметка нашего проекта хранится в HTML-файлах, код стилей в CSS-файлах, а скрипты — в JS-файлах. HTML-файлы система откроет в браузере, CSS и JS — в текстовом редакторе.

Продолжая аналогию, наша профессия — это атрибут файла. Атрибуты — набор свойств файла. Список атрибутов доступен в специальной вкладке операционной системы.

Атрибутов бывает много. Например, атрибут readonly запрещает пользователю запись и редактирование файла. Это может стать проблемой. Файлы с атрибутом readonly без дополнительных действий не получится отредактировать на удалённом сервере. А файлы с атрибутом hidden вообще будут скрыты от пользователя.

Файлы можно разделить на две категории:

* Файлы данных — интересуют веб-разработчика в первую очередь. Графические и текстовые файлы, файлы баз данных и сценариев *(скриптов)*.
* Программные файлы. Имеют расширение текущей OC. Выполняют системные функции. Например, файлы с расширением .lnk — ярлыки или ссылки в Windows.

Каталог или директория — место для хранения других директорий и файлов. Наши проекты хранятся в директориях, которые мы для них создаём.

Теперь, когда в терминологии разобрались, можно перейти к построению структур из файлов и директорий.

# Линейная структура и иерархия. Разделение файлов по типам

Файловая структура — это способ организации файлов по каталогам.

Простая файловая структура — линейная, без вложенных директорий внутри родительского каталога.

У простой файловой структуры есть свои недостатки. Проект с большим количеством HTML-, CSS-, JS-файлов и изображений превращается в хаос. Найти нужный файл становится очень сложно. А скорость — ключевой показатель эффективности разработчика.

Удобнее разделять файлы по папкам, отводя для изображений, шрифтов, файлов стилей и скриптов свои каталоги. Это облегчает поиск. Файловая структура со вложенными директориями называется иерархической.

Иерархическая файловая структура — это файловая структура, которая содержит вложенные друг в друга каталоги и подкаталоги с файлами.

Вот файловая структура, которую часто организуют веб-разработчики:

1. HTML-файлы хранятся в корневом каталоге.
2. В этот же «корень» вложены подкаталоги —
3. /images, изображения — иконки, фоновые и контентные изображения различных расширений;
4. /styles, файлы стилей;
5. /scripts, JS-файлы;
6. /fonts, шрифты.

Это самая примитивная файловая структура, которую в основном используют для небольших проектов. Со временем вы будете строить более сложную БЭМ-структуру файлов. А пока остановимся и разберёмся, как связывать файлы друг с другом.

# Абсолютные и относительные пути к файлам

При вёрстке часто приходится связывать файлы друг с другом. Например, в файле стилей указать ссылку на изображение для свойства background-image. Изображение может лежать в другой папке или в интернете. Вам нужно забрать его, как посылку с почты. Чтобы её забрать, нужно проследовать по маршруту до почтового отделения. Чтобы забрать файл — написать до него путь.

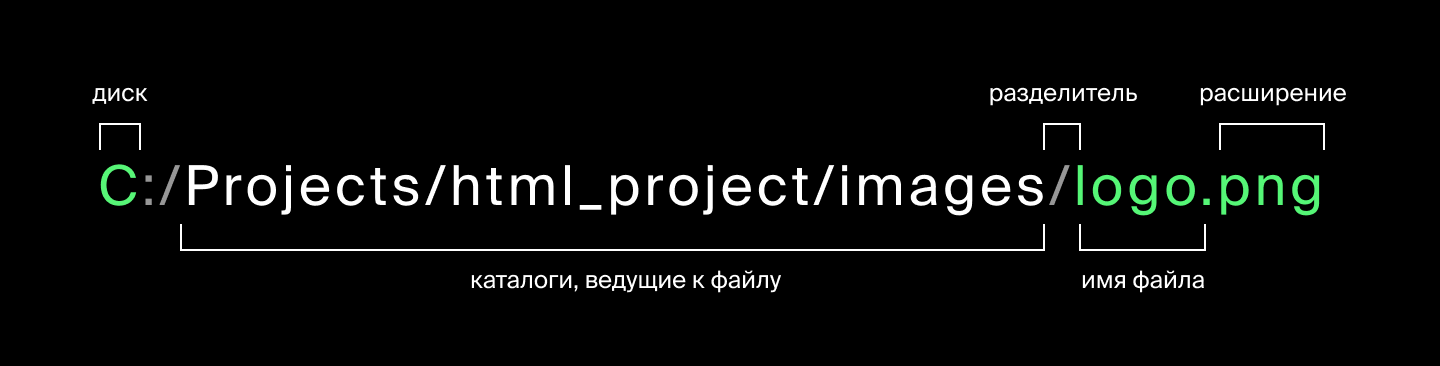
Путь — это адрес файла на вашем компьютере или в интернете. Пути к файлам бывают абсолютные и относительные.

Представим, что жёсткий диск компьютера — это почтовое отделение, а наш текущий каталог — подъезд многоквартирного дома. Абсолютный путь — это путь от почтового отделения до нашей квартиры, а относительный — от дверей нашего подъезда. Для абсолютного пути важно, к какому почтовому отделению относится наш подъезд, а для относительного — нет.

Все файлы вашего компьютера лежат на жестком диске в иерархической структуре.

Абсолютный путь на компьютере — это перечисление всех шагов от самого глубокого домашнего каталога до нужного файла.

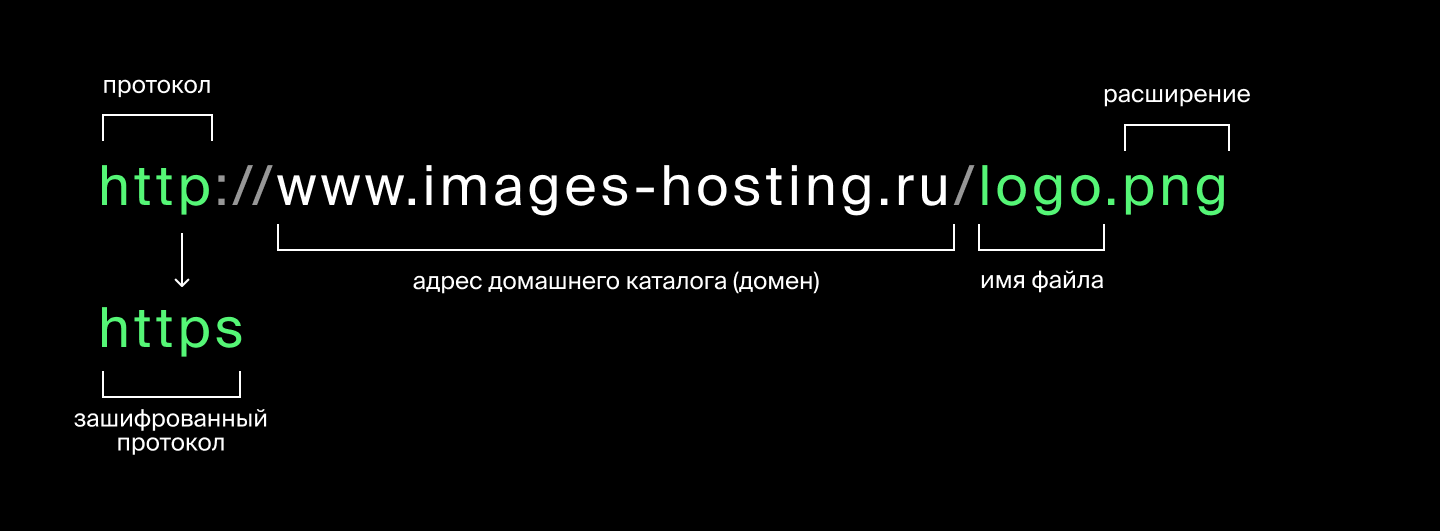
Жесткий диск C:/ — корневой каталог, начало пути до адреса, по которому нужно доставить файл. C:/Projects/html\_project/images/logo.png — полный абсолютный путь, где logo.png — название файла, /Projects/html\_project/images — каталоги на пути, а C: — диск, на котором «лежит» каталог. Символ / разделяет названия каталогов, ведущих к logo.png.



Такая ссылка будет работать в любом месте жёсткого диска. Но если мы переместим файл в другой каталог, по старому пути он станет недоступен.

Абсолютный путь файла в интернете — ссылка на ресурс, к которому обращён запрос. Начинается с указания протокола http:// или https://. Если представить обращение к ресурсу как отправку письма в другой город, то HTTP-протокол — дорога, по которой поедет машина с нашим письмом, а затем и ответом на него. HTTPS-протокол — платная автомагистраль, более защищённая и качественная в сравнении с обычной дорогой.

Домены, как и люди, переезжают. Сменив адрес в интернете, файл перестанет быть доступным по старому пути. Если домен сайта с изображением http://www.images-hosting.ru/logo.png изменится на http://www.logo-hosting.ru/logo.png, изображение по запросу http://www.images-hosting.ru/logo.png не найти.



Попытка скачать что-нибудь из интернета на страницу, размещённую на компьютере, занимает время. Скорость выполнения запроса зависит от качества интернет-соединения и работоспособности сайта, с которого вы собираетесь скачать файлы. Если вы храните файлы сайта в интернете и используете абсолютные ссылки — знайте, что это влияет на производительность вашего сайта.

# Абсолютный путь и тег link

Вы использовали элемент <link> для подключения стилей, но делали это в простой файловой структуре, когда style.css находился на одном уровне с index.html. Вот как это выглядело:

Скопировать кодHTML

<head>

<link rel="stylesheet" href="style.css">

</head>

<link> позволяет обращаться к файлам в интернете, указывая ссылку на них в уже знакомом атрибуте href. Нередко это библиотека — написанный другим разработчиком код, который позволит вам не решать уже решённые кем-то проблемы. Библиотеки удобно использовать для вспомогательных задач, на решение которых уйдёт больше времени, чем на написание основного кода проекта. Ресурсы с кодом библиотек называют CDN — Content Delivery Network (англ. сеть доставки контента). К файлам на CDN также делают запрос через link.

Известный CDN для поиска фронтенд-библиотек:

<https://cdnjs.com/>

Есть ряд минусов:

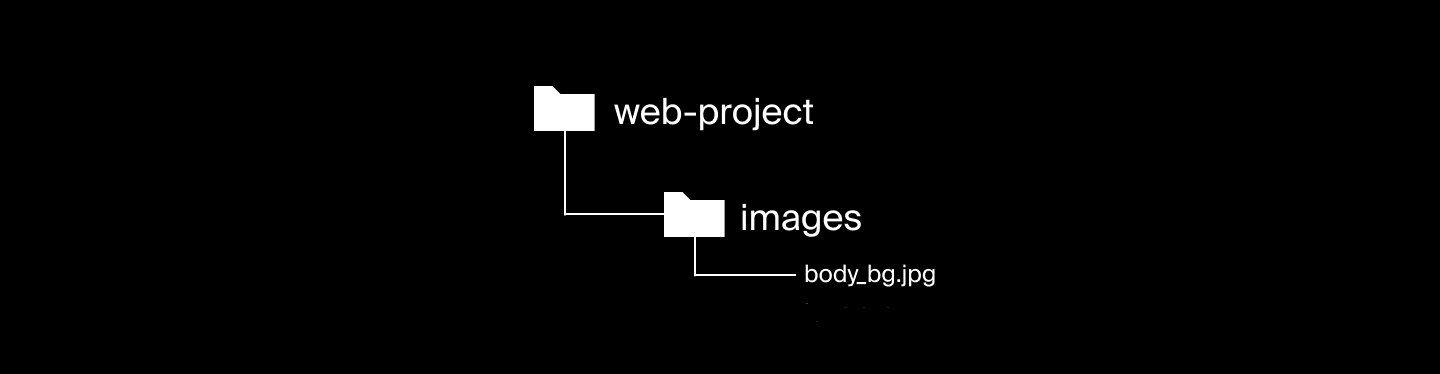
* Вы не можете редактировать получаемые с CDN файлы;
* Сторонние библиотеки поддерживают их владельцы. Они добавляют в них возможности и меняют код. Вы не можете на это влиять.
* Выполняется запрос к стороннему файлу. Сервер может сломаться и перестать отвечать на запросы. Если ваш сайт зависит от файла на этом сервере, он тоже сломается.

Чтобы эти проблемы нас не касались, мы будем скачивать чужие библиотеки разными инструментами. Но об этом позже.

# Относительные пути к файлам

В прошлом уроке вы подключали файлы, указывая абсолютные пути. При таком подходе возникает ряд проблем. Когда вы размещаете вёрстку на удалённом сервере, все абсолютные пути до вашего компьютера перестают работать. Веб-сервер имеет собственный диск, где структура файлов отличается от вашей.

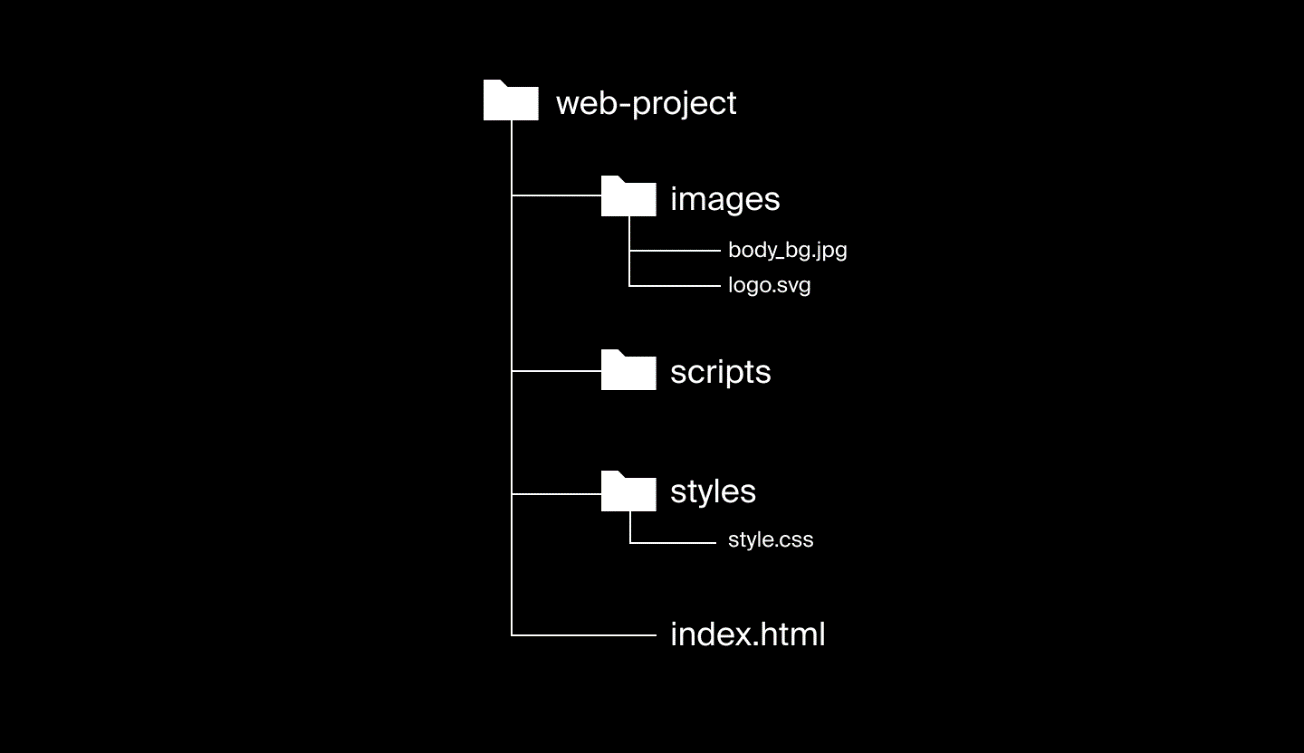
Чтобы избежать такой проблемы, есть относительные пути. Путь к нужному файлу указывают относительно файла, в котором он вызывается. Например, берём изображение из каталога /images для CSS-свойства background-image:



Путь до изображения

Если представить наш проект как многоквартирный дом, при создании относительных путей неважно, в каком городе или стране он находится. Имеет значение, из какой квартиры на каком этаже мы отправимся к соседям за солью.

Так и относительные пути указывают, как пройти до нужного файла из того места, где этот путь прописан.

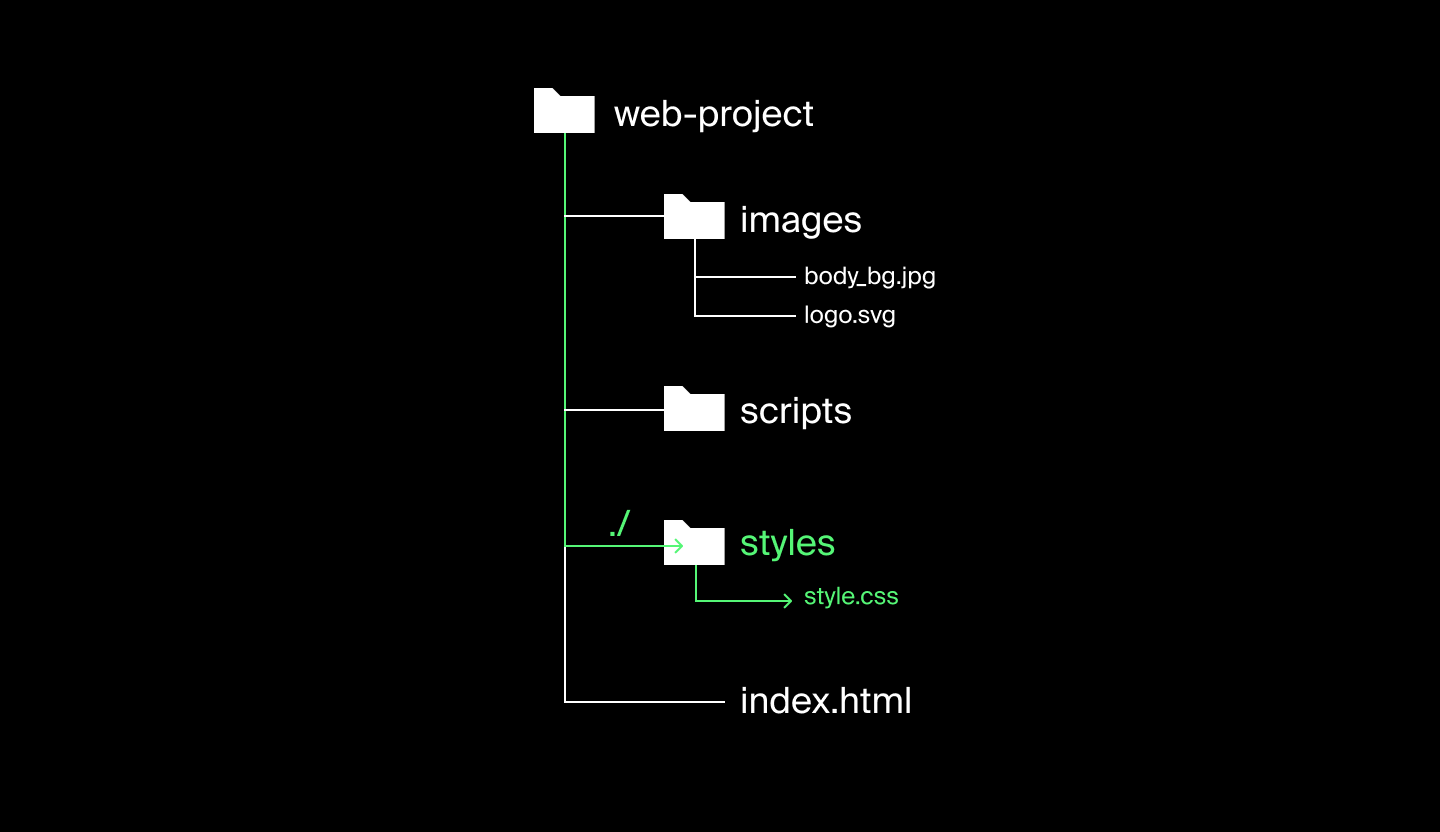


Файловая структура проекта

Многоквартирный дом — это папка проекта:

* его корневой каталог — web-project,
* этажами images/, scripts/, styles/
* и цокольным этажом index.html.

Наша квартира — style.css на этаже styles/. Из неё мы пойдём за солью body\_bg.jpg в квартиру на этаже images/ .



Отображает путь до style.css в директории styles из родительского каталога

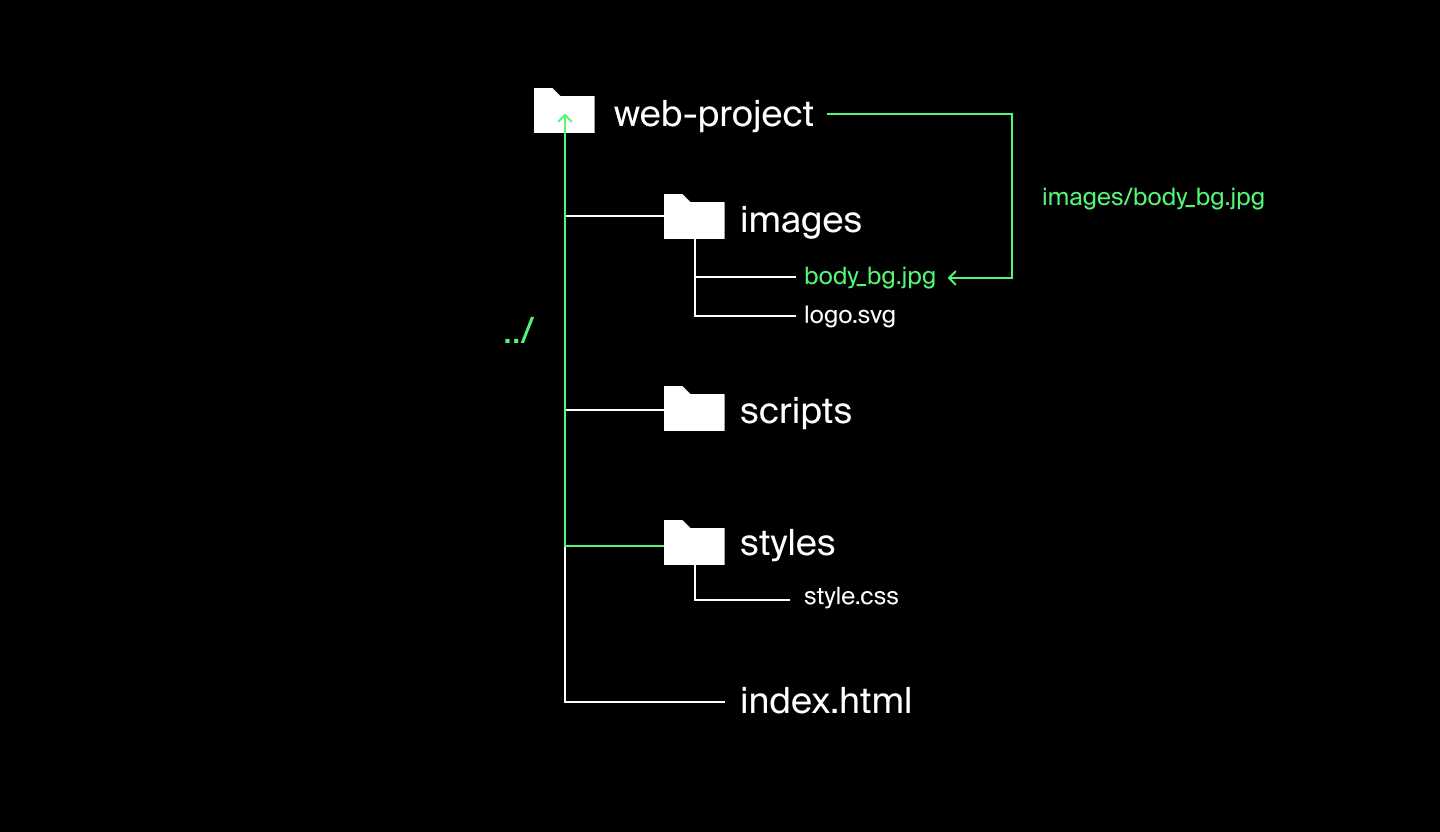
Путь из квартиры на лестничную клетку можно представить символами ./. Указав такой путь в style.css, мы окажемся на этаже styles/ и сможем позвонить в другие квартиры — файлы стилей — если они есть:

Скопировать кодHTML

*<!-- выходим на лестничную клетку, поднимаемся на этаж styles и звоним в квартиру style.css -->*

<link rel="stylesheet" href="./styles/style.css">

Попасть на лестницу можно командой ../. Эта комбинация символов ведёт на уровень выше — в каталог web-project/. Это и есть путь к соседям.

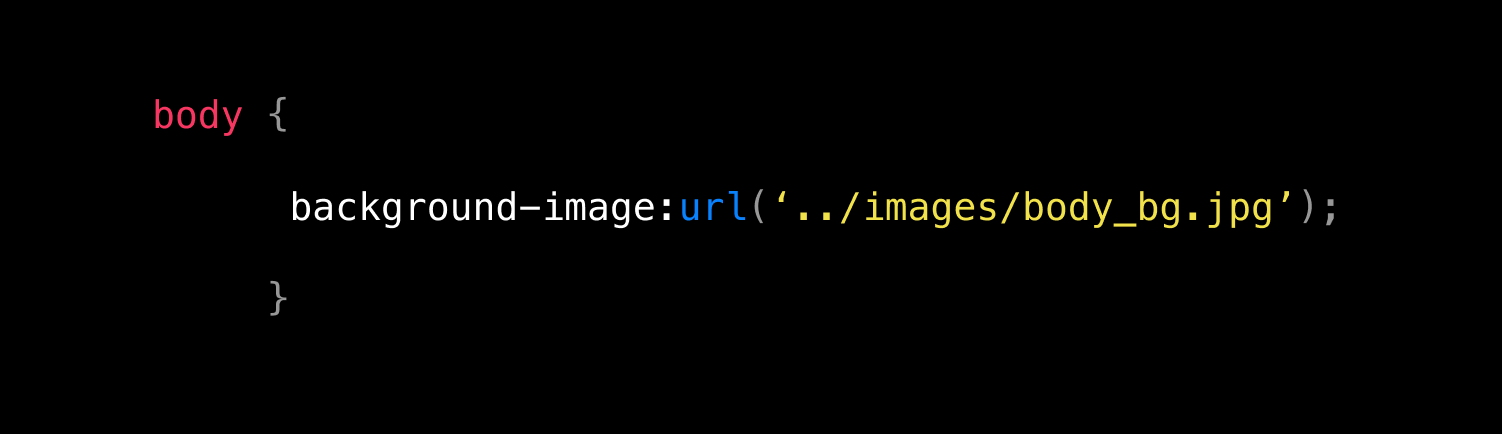


Зеленым показан путь до изображения body\_bg.jpg: сперва попадаем в корневой каталог web-project командой ../ , затем — в каталог изображений images

Выйдя на лестничную клетку, мы спустились на этаж images/ и постучались в квартиру body\_bg.jpg.

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

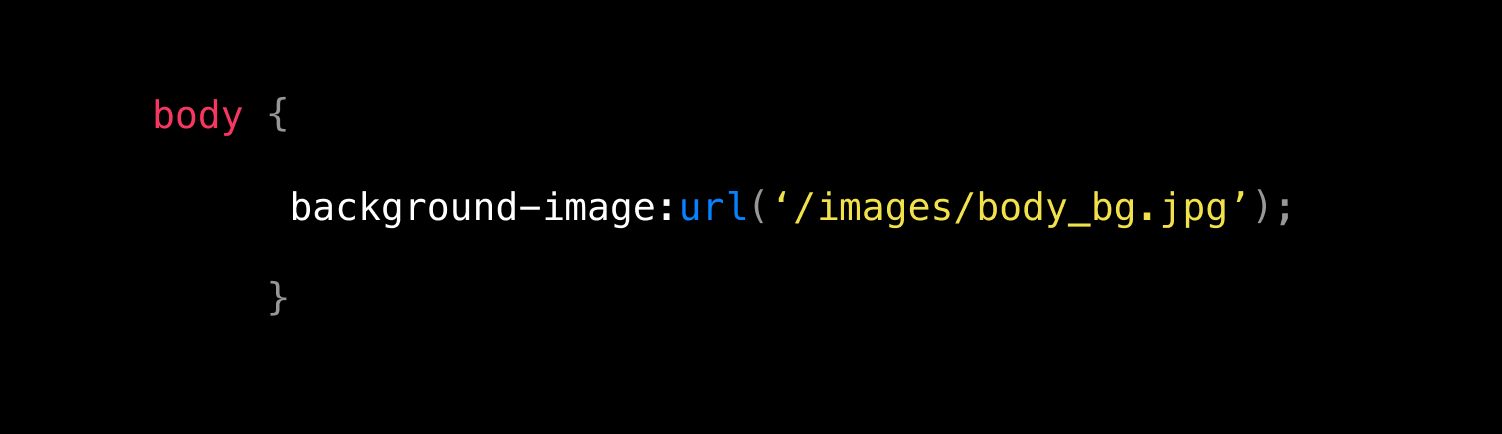
../ — указание на каталог одним уровнем выше.



Жёлтым показан путь до изображения body\_bg.jpg: сперва попадаем в корневой каталог web-project, затем — в каталог изображений images

Другой вид относительных путей — от корневого каталога проекта. К нему нас возвращает указатель /. Это как путь от двери подъезда до любой квартиры.

Путь из примера с соседями и солью можно представить так:



Указываем путь к фоновому изображению от корня проекта

Такая запись позволит, например, перенести проект на другой сервер без разрыва внутренних ссылок. Например, если в каталоге images/ появится ещё одна директория, в которую будет перемещён body\_bg.jpg.

Учтите, что указатель / работает только на веб-сервере. Если вы напишете такой путь в файлах вёрстки на своём компьютере и откроете ссылку в браузере — окажетесь в самом корне операционной системы.

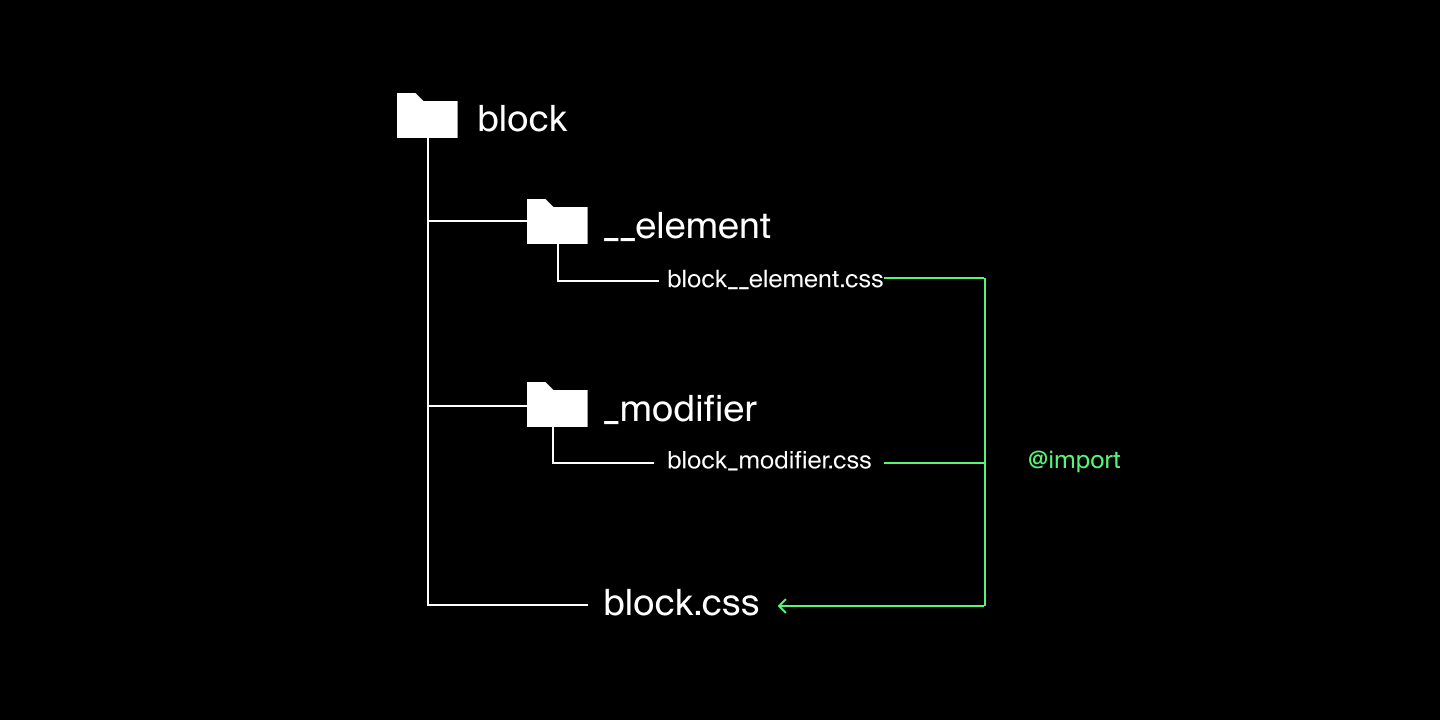
# CSS-директивы. @import

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 коммерческой at @ 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:

Скопировать кодCSS

@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.

Идея импорта раскроет себя при изучении автоматических сборщиков, таких как Webpack. Вы сможете разделять код на небольшие блоки согласно его функциям и особенностям. Это позволит создавать заготовки для быстрой разработки проектов.

Пока что важно запомнить, что такое директива и как импортировать файл в свой CSS-код.