**Пути к файлам и организация файлов по БЭМ. Введение**

Вы уже знакомы с файловой структурой, при которой файлы группируются по своим типам: все изображения хранятся в директории images/, CSS-файлы в styles/, шрифты в fonts/, скрипты в scripts/. У неё есть недостаток: стили всех блоков описаны в одном файле — style.css.

Например, вы подключали файл стилей для этого проекта: <https://praktikum.yandex.ru/trainer/web/lesson/e1e634f9-465d-4d6d-a312-3c7483ed56b2/task/3ca4262f-b431-4a83-bd4b-305162535e60>. Разберём проблемы, связанные с таким подходом:

* Общий CSS-файл для всех страниц. Например, форма есть только на одной странице, а код этой формы подгружается для каждой. Браузер выполняет лишнюю работу, загружая ненужные стили.
* С ростом проекта поддерживать такой код становится всё сложнее. Крупный проект может иметь несколько тысяч строк стилей. Если хранить их в одном месте, то поиск нужного селектора займёт много времени. Представьте, что в библиотеке тысячи томов хранятся на одной огромной полке — вряд ли вы разыщете там нужную книгу за десять минут.
* Неудобно использовать блоки повторно. Чтобы найти блок со всеми его элементами и модификаторами, приходится листать длинный CSS-файл. Чем больше код, тем сложнее это сделать.

Существуют и другие способы организации файлов и кода в проекте.

Методология БЭМ предлагает несколько подходов к организации файловой структуры: Nested, Flat и Flex. Сейчас мы будем изучать Nested — классический вариант. В интернете вы легко найдёте для него множество примеров и гайдов.

Сначала мы расскажем, как организовать файлы в Nested-структуре, затем полностью перенесём на неё уже знакомый проект.

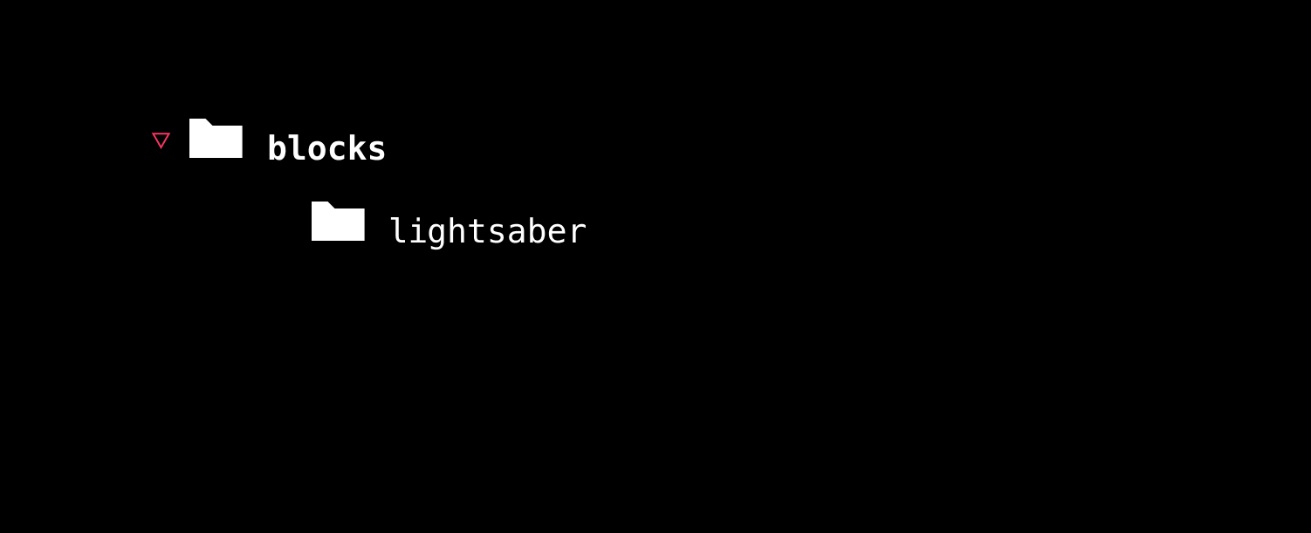
Перейдём к главному: как выглядит файловая структура по БЭМ?

# Объясним всё и сразу аналогией

Группировка файлов по методологии БЭМ — объединение файлов не только по их типам, но и по функциональности.

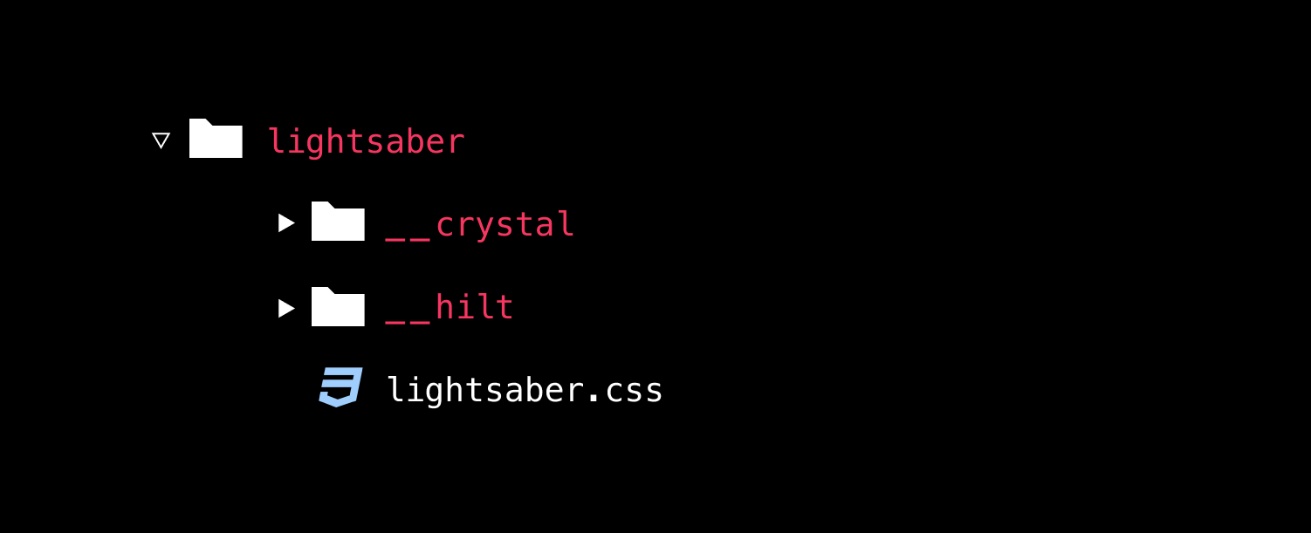
Звучит сложно, поэтому начнём с аналогии. Представим, что где-то в далёкой-далёкой галактике мы — рыцари-джедаи. Лучше всего о нас говорит наше оружие — световой меч. Чаще всего он состоит из двух частей — рукояти и кристалла. Сейчас мы занимаемся написанием кода, поэтому отложим меч подальше — разберём его и положим в ящик. В одной секции ящика лежат рукояти, в другой — кристаллы. Больше в этом ящике ничего не лежит. Так нам будет проще собрать оружие в случае нападения ситхов.

Сразу переведём эту историю на язык БЭМ. Сам меч — это БЭМ-блок lightsaber, а рукоять и кристалл — БЭМ-элементы \_\_hilt и \_\_crystal. Место в файловой структуре, где будет располагаться БЭМ-блок, — каталог blocks.



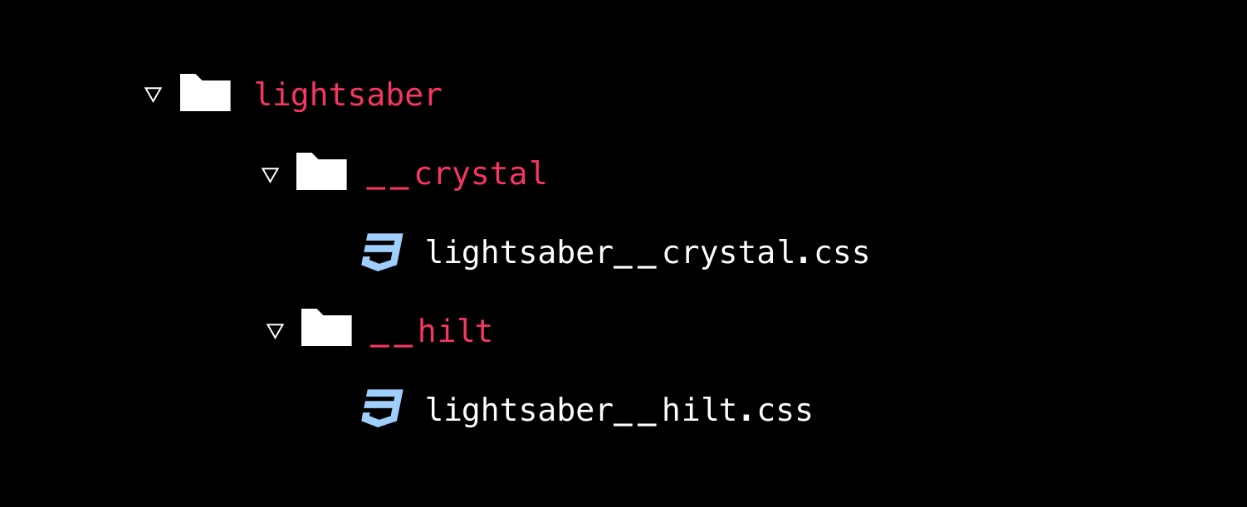
Расположение блока lightsaber в директории blocks

В этом каталоге хранятся рукояти и кристаллы нашего светового меча:



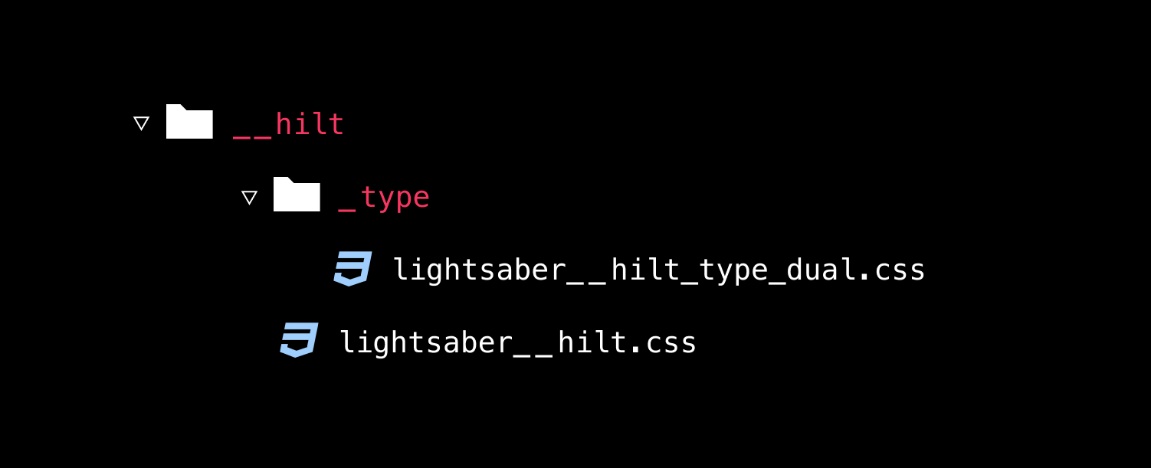
Структура блока lightsaber

Опишем рукоять и кристалл в отдельных CSS-файлах. Включим в названия этих файлов название их блока:



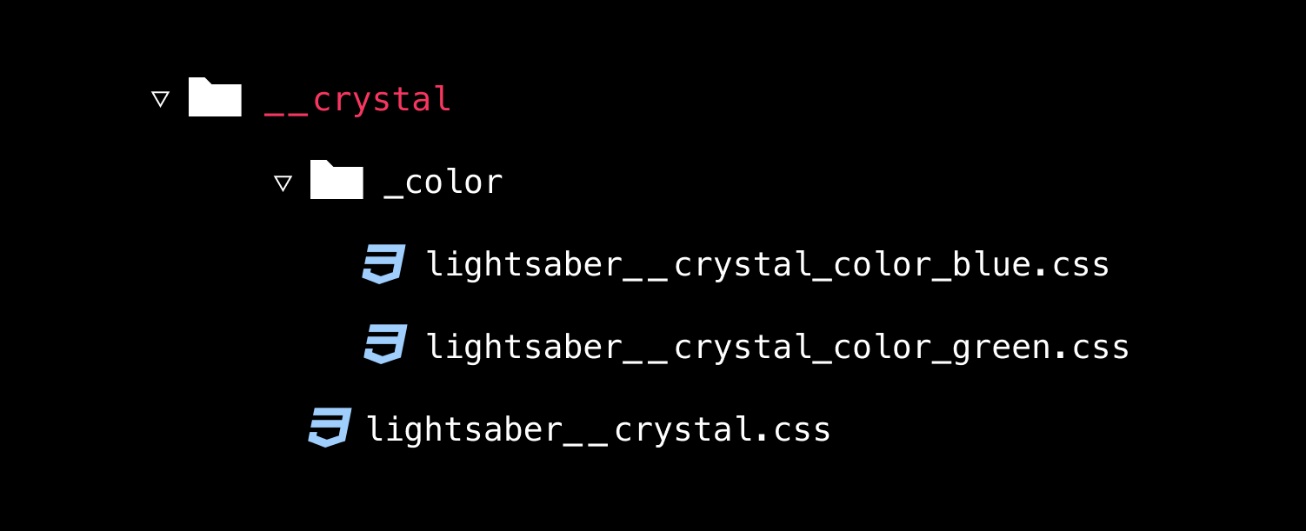
Файловая структура для элементов \_\_hilt и \_\_crystal

Во время путешествия по галактике мы пополнили коллекцию рукоятью для двухклинкового меча и несколькими кристаллами: синего и зелёного цветов. Представим их как БЭМ-модификаторы. Рукоять двухклинкового меча — сложный модификатор элемента \_\_hilt с названием типа — ключом \_type и его значением \_dual. Отразим это в структуре. Новому каталогу дадим ключ модификатора \_type, а CSS-файлу этого модификатора — полное название, вместе с модифицируемым элементом \_\_hilt и значением модификатора \_dual.



Файловая структура для модификатора \_type\_dual

Синий и зелёный цвета кристалла — это значения модификатора для ключа \_color элемента \_\_crystal:



Файловая структура для модификаторов \_color\_blue и \_color\_green

Круто! Все составные части меча лежат в нужных местах, он собирается быстро и легко.

Теперь нужно подключить все блоки, элементы и модификаторы к странице. Научимся это делать в следующем уроке.

**Подключаем нужные блоки, элементы и модификаторы к странице**

Мы собрали все составные части светового меча. Разделили их по соответствующим полкам ящика. Другими словами, мы подготовили файловую структуру для блока lightsaber, его элементов и модификаторов: рукояти и кристалла.

Теперь поговорим о том, как подключить стили блоков, элементов и модификаторов.

Вернёмся к джедайской аналогии. Вы путешествуете по целым солнечным системам и даже галактикам на своём корабле. Когда вы отправляетесь на какую-то планету, логично не брать с собой все кристаллы и рукояти, а сразу собрать меч и отправиться в путешествие.

Представим, что вы отправляетесь на планету Набу и хотите взять с собой клинок с двойной рукоятью и синими кристаллами. Вы соберёте клинок и отправитесь на планету, а все оставшиеся части продолжат лежать в ящике до того момента, когда они вам потребуются.

Переведём эту историю на язык БЭМ. Планета Набу — одна из страниц проекта. В проекте может быть много страниц, поэтому логично использовать на них только те блоки, их элементы и модификаторы, которые в данный момент нужны. Если потребуется что-то новое — достаточно будет просто подключить это к странице. Используя такую логику при подключении стилей, браузер не выполняет лишнюю работу, загружая совершенно ненужные стили.

Если на одной из страниц проекта нам потребуется двойной световой клинок с синим кристаллом, то нужно подключить:

1. Сам блок lightsaber, который содержит стили блока. Мы не можем использовать элементы блока и модификаторы в отрыве от блока.
2. Его элементы — lightsaber\_\_crystal и lightsaber\_\_hilt, содержащие стили элементов блока. \_type\_dual и \_color\_blue — это модификаторы. Мы не можем воспользоваться только модификаторами, без модифицируемого элемента.
3. Необходимые модификаторы lightsaber\_\_crystal\_color\_blue и lightsaber\_\_hilt\_type\_dual.

Рассмотрим подробнее, как именно подключать нужные файлы к странице. У вас есть компонент lightsaber внутри папки blocks. Теперь нужно подключить оформление блока к HTML-странице, причём так, чтобы на страницу не загружался лишний код. Есть два подхода:

**Быстрее загрузится**

Вы можете подключать нужные блоки, элементы и модификаторы через тег link внутри head. Способ хороший, ведь стили начнут загружаться, как только браузер дочитает вашу страницу до блока head.

Но такой код перегружен. Внутри head становится очень много элементов link.

Для финальной версии, которую вы загрузите на сервер, такой подход хорош, но в разработке неудобен.

**Удобный в разработке**

Можно создать отдельную папку в структуре, назвать её pages и разместить в ней файлы для каждой страницы. В эти файлы импортировать только блоки, элементы и модификаторы, которые нужны соответствующей странице. Теперь вы можете подключить в HTML всего один файл со всеми нужными стилями, найдя его по заданному имени.

Этот способ делает код медленнее, ведь каждый импорт — это отдельный запрос к серверу, а их становится больше. И всё же так удобнее работать и запоминать зависимости. В будущем вы изучите специальную программу-сборщик — webpack. Она может автоматически собирать все файлы по цепочке импортов в готовый к загрузке на сервер быстрый проект.

Воспользуемся вторым подходом. К примеру, при подготовке к полёту на планету Набу — создадим директорию pages/, в ней CSS-файл, например, naboo.css. И уже в этот файл директивой @import подключим необходимые файлы:

Скопировать кодCSS

*/\* /pages/naboo.css \*/*

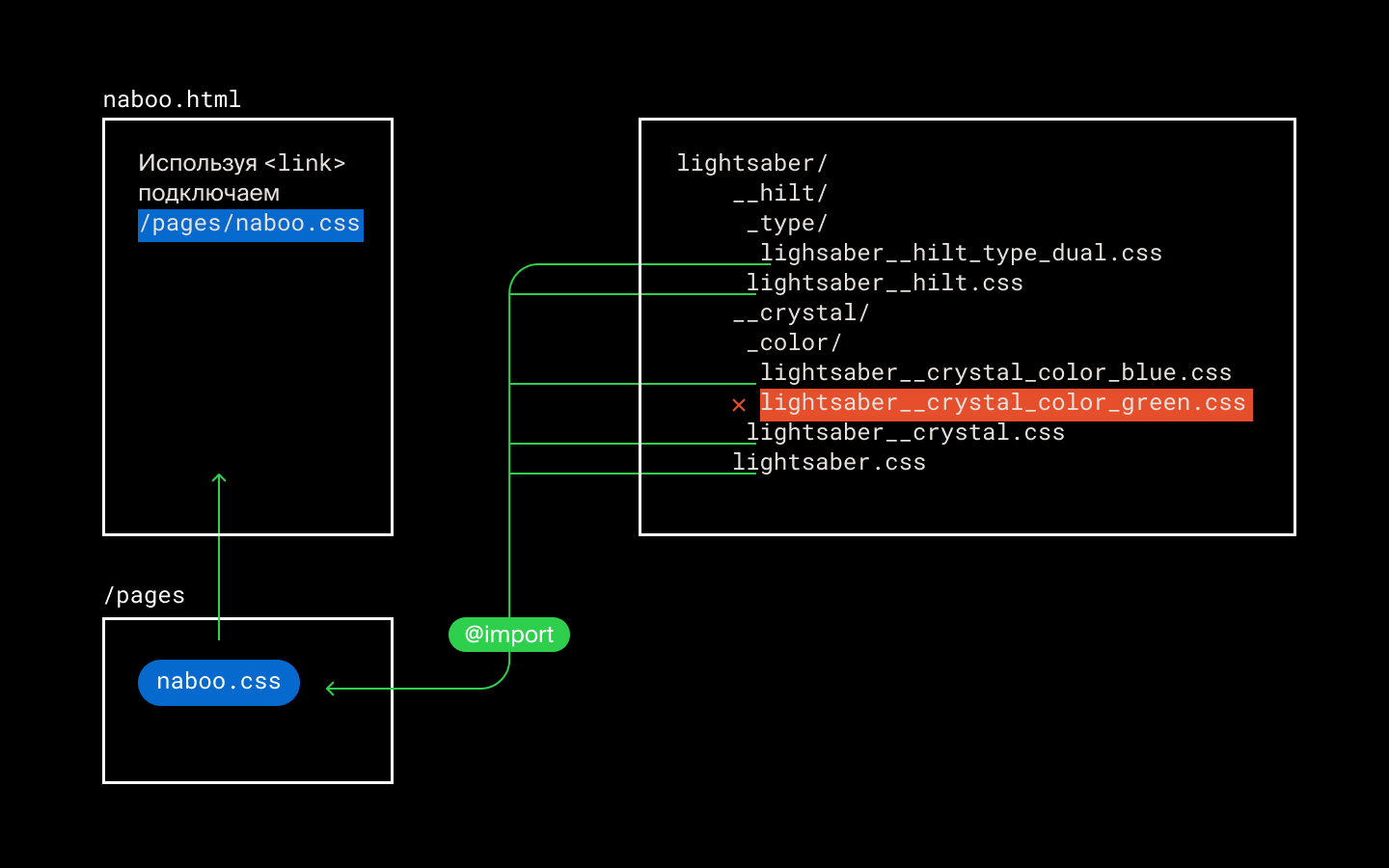
@import url(../blocks/lightsaber/lightsaber.css);

@import url(../blocks/lightsaber/\_\_crystal/lightsaber\_\_crystal.css);

@import url(../blocks/lightsaber/\_\_crystal/\_color/lightsaber\_\_crystal\_color\_blue.css);

@import url(../blocks/lightsaber/\_\_hilt/lightsaber\_\_hilt.css);

@import url(../blocks/lightsaber/\_\_hilt/\_type/lightsaber\_\_hilt\_type\_dual.css)



Подключаем только необходимый на странице блок, его элементы и модификаторы. Модификатор \_color\_green на странице naboo.html не используется.

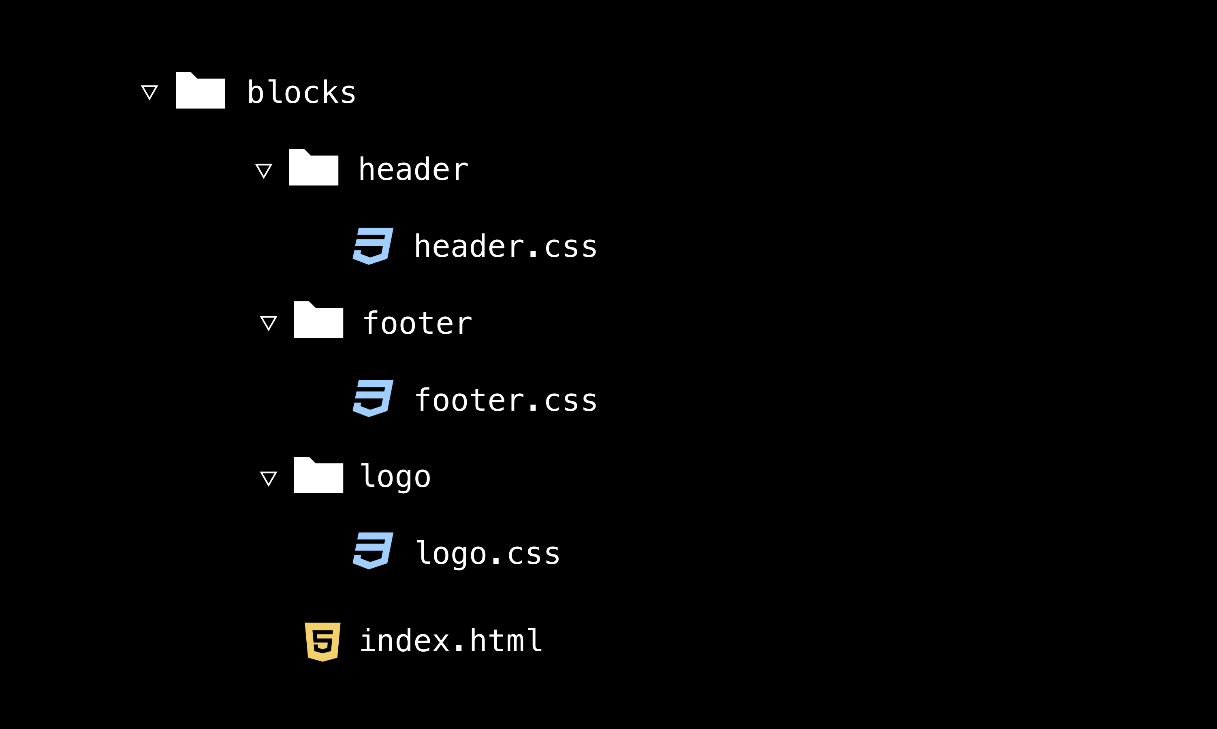
**Структура для блоков**

Блок в методологии БЭМ — основной строительный материал. Как из кирпичей строят здание, так и из блоков, организованных в сетку, делают веб-страницы. Отсюда требования к коду блоков:

* Код должен быть обособленным — не зависеть от своего окружения. Значит, нам не подходят внешние отступы и позиционирование. Мы стремимся к возможности взять блок и без изменений скопировать его код на новую страницу или в новую зону сайта. Блок должен встать «как влитой», чтобы вам приходилось менять его как можно меньше.
* Называйте блоки понятно. Название блока должно сообщать о его функциональности. Идеально, когда для идентификации блока вообще не нужно отвлекаться от кода и смотреть на страницу.
* Все элементы и модификаторы должны быть внутри блока — без него они не имеют смысла.

Когда вы продумали структуру блоков на странице, смело создавайте отдельные директории под каждый блок внутри отдельной папки blocks. А в каждой папке блока — CSS-файл с его именем. Эту папку вы будете переносить из проекта в проект; весь код блока будет уже в ней.

Вот примерная структура, когда на странице есть только блоки:



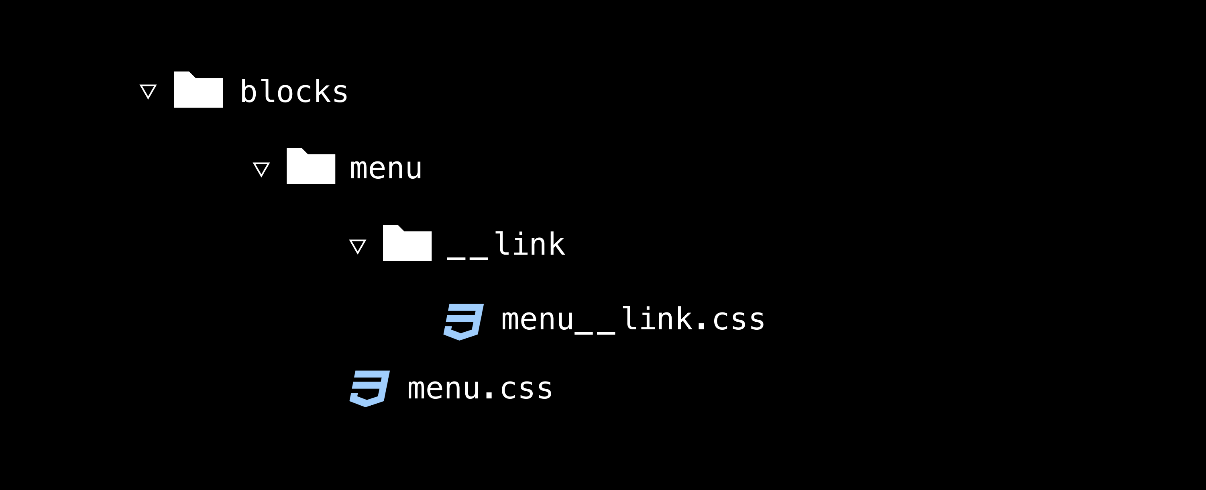
Пример файловой структуры по БЭМ в проекте

# Структура для элементов

Элементы в БЭМ — составляющие блока, они не существуют в отрыве от него. Поэтому в БЭМ-структуре весь код элементов размещают внутри папок блоков. Чтобы экономить время, папки называют сокращенно: \_\_имя-элемента, ведь из структуры следует, к какому блоку относится элемент.

Имя CSS-файла для элемента пишут полностью, чтобы в коде не приходилось догадываться о принадлежности элемента к блоку, анализируя весь путь до него.

Вот пример структуры одного блока с элементом:



Структура блока menu и его элемента \_\_link

Построив такую файловую структуру, нужно сделать, чтобы страница «увидела» использующийся на ней элемент блока. Для этого в CSS-файл страницы импортируют CSS-файл элемента.

Код index.css:

Скопировать кодCSS

@import url(../blocks/menu/menu.css);

*/\* подключаем стили блока \*/*

@import url(../blocks/menu/\_\_link/menu\_\_link.css);

*/\* после блока подключаем стили элемента \*/*

# Структура для модификаторов

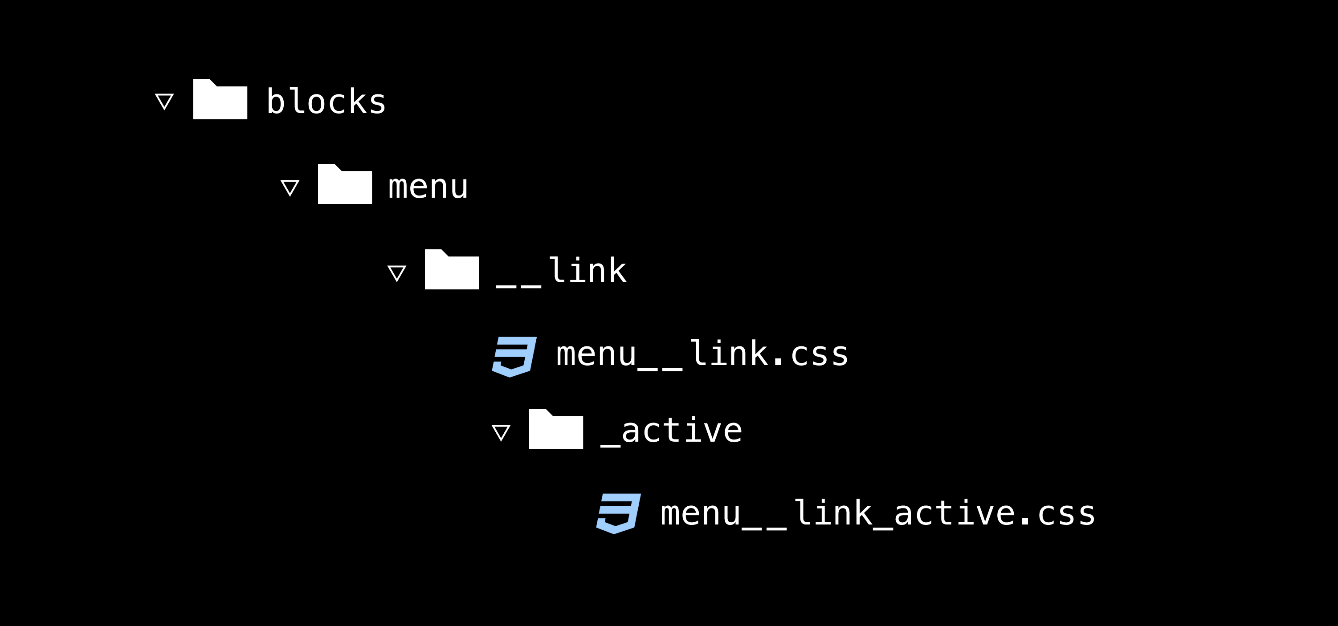
Модификаторы бывают разные, поэтому вариантов их интеграции в структуру несколько. Модификатор может изменять и блок, и элемент. Папку для файла с модификатором создают внутри той директории, на которую он воздействует. Если модифицирован блок, папка лежит в папке блока, если элемент — в папке элемента.

Если модифицируют блок:



Расположение модификатора блока button

Если модифицируют элемент:

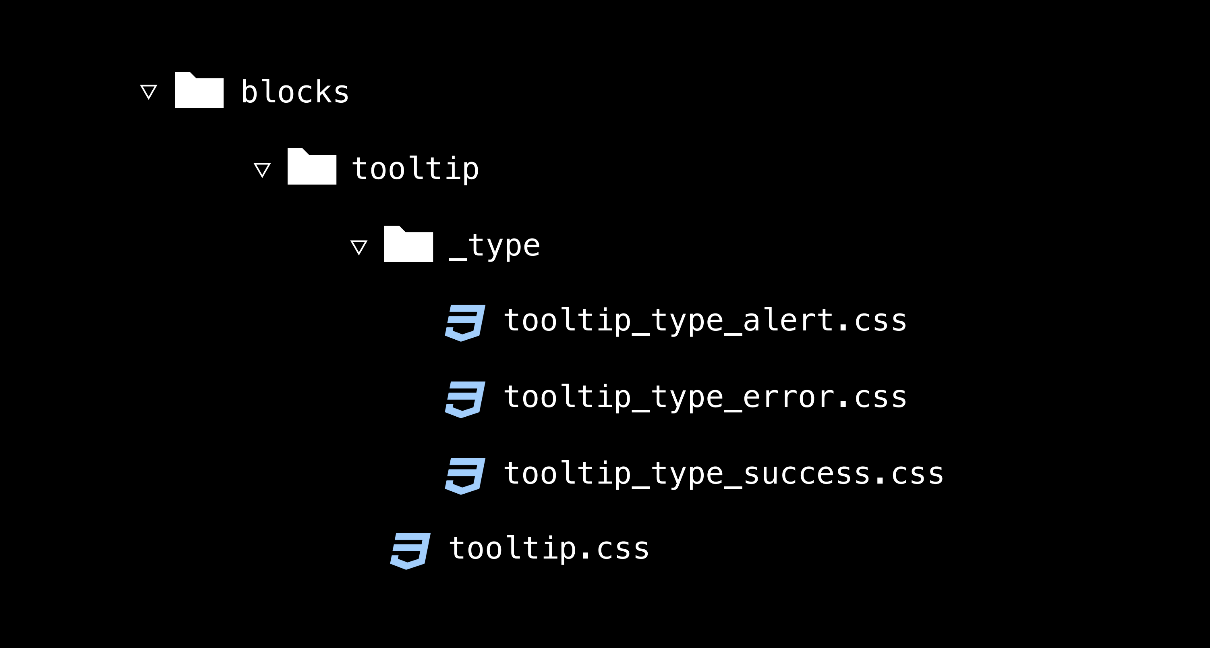


Расположение модификатора элемента \_\_link

Как и в случае элемента, папки для модификаторов записывают короткой записью, не повторяя имя блока, а сам файл называют полным именем класса модификатора. Модификаторы в классическом БЭМ-именовании обозначены одним нижним подчёркиванием.

Мы продемонстрировали запись для простого модификатора — булевого. Его присваивают, когда у элемента всего два значения, например, активный и неактивный. Чаще используют сложные модификаторы вида ключ-значение. Такие модификаторы оформляют в структуре иначе: создают отдельную папку только для ключа, а файлы со значениями вкладывают внутрь.

Например:



Модификаторы ключ-значение располагаются в одной директории с названием ключа

Модификаторы тоже импортируют в CSS-файл страницы. Важно помнить, что модификаторы нужно импортировать ниже модифицируемого элемента, иначе стили могут конкурировать друг с другом.