# Введение

Самое главное, чему надо научиться программисту - это “гуглить”. Поэтому в этой методичке не будет ссылок, зато будут небольшие подсказки как правильно находить информацию в интернете самостоятельно. В процессе реальной работе на производстве, задача “инвестигации” (подробного изучения не решаемой ранее проблемы через официальную документацию) встает довольно регулярно. Тем не менее, если поиски в сети Интернете не увенчались успехом, попробуйте посоветоваться с “коллегами по работе” - вашими одногруппниками. Если и это не помогло - обратитесь к “тимлиду” - тематическому чату в телеграм с работодателем и преподавателем.

# CSS

## Что такое CSS

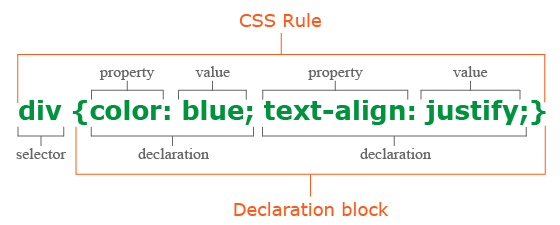
CSS (Cascading Style Sheets) - это язык таблицы стилей, который нужен для описания внешнего вида веб-страницы, чаще всего используется с HTML. Основная задача CSS в том, чтобы все HTML-элементы, находящиеся у вас на страницы, имели какой-то свой вид и каскадную структуру отображения.

В идеале HTML-файл должен состоять только из тегов, отображающих его логику и структуру, а уже вид элементов будет задаваться через стили.

*Исчерпывающую информацию о синтаксисе CSS можно найти по запросу* ***css mdn*** *в поисковых системах.*

## Синтаксис

Такую конструкция используется чаще всего:



Рассмотрим подробнее, что тут происходит:

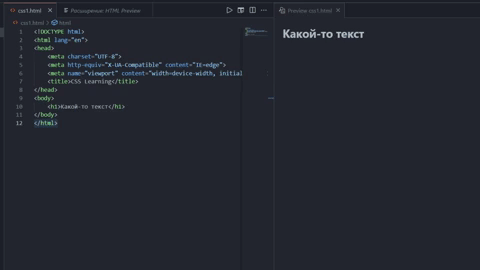
1. **selector** - имя, которое мы будем стилизовать. Здесь вы чаще всего будете использовать имена тегов и имена классов. Также можно перечислять их в коде через запятую, чтобы использовать описанный стиль для нескольких блоков. Более подробные сведения о разных видах селекторах изучим чуть позже.
2. **property** (свойство, параметр) - то, характеристика элемента. В схеме синтаксиса приведен пример с параметром color - цвет элемента.
3. **value** (значение) - явное значение интересующего нас свойства. *С помощью поисковых систем изучите свойство* ***margin*** *и попробуйте понять как меняется стиль элементов при изменении его значения.*
4. **declaration** - описание вашего стиля, которые вы применяете.

## Использование в HTML-коде

Как можно использовать CSS в HTML:

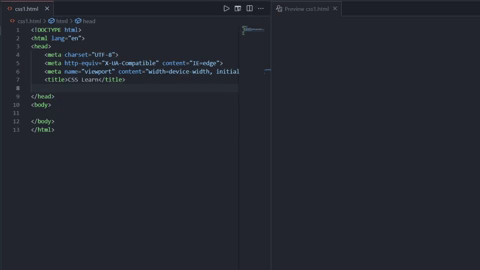
1. Инлайн-стили
2. Подключение со стороннего сервера в head
3. Подключение файла \*.css, в котором будет вся информация в head

Inline - когда стили задаются в самом HTML-файле через атрибут **style**



В таком случае вы описываете стили к конкретному элементу атрибутом **style**

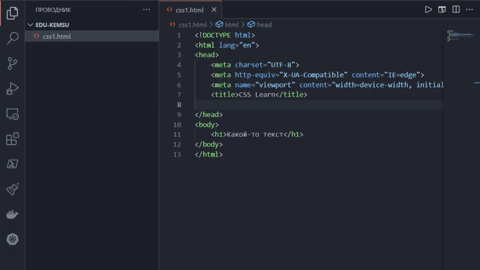
Также “инлайн-стили” могут добавляться в html-код целым блоком кода и являются приоритетными - если вы подключили какой-то css-код, но используете тег **style**:



Как вы можете уже заметить - в блок кода **style** можно записывать неограниченное количество свойств, которые вы хотите применить к какому-либо элементу страницы

В целом, такой подход действителен, но если таких стилей будет много, то это сильно перегрузит документ. Более оптимальный вариант использований собственных стилей - это подключение их из файла, например **style.css**

В таком файле вы будете писать также, как и в теге **style**, только без самого тега:



Как вы здесь можете увидеть - подключается сам файл **style.css**, используя следующую конструкцию:

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

Где в атрибуте href - вы пишите относительный путь до вашего файла

Вызвать такую конструкцию возможно при помощи следующего emmet’а:

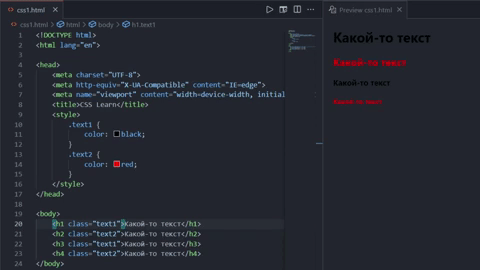
**link:css**

# Атрибут class

**class** задает стилевой класс, который позволяет связать определенный тег со стилевым оформлением. В значении допускается указывать сразу несколько классов, разделяя их между собой пробелом

Сам селектор **class** в **css** в отличие от обычного селектора, когда мы обращаемся к конкретному тегу, записывается как **.класс**

Давайте рассмотрим тег **class** на примере использования следующего стиля и нескольких элементов



Здесь вы можете также заметить, что все элементы мы можем группировать и применять следующим образом стили сразу к нескольким элементам

# Тег <div>

Настало время узнать про контейнеры, благодаря которым вы будете выстраивать определенную логику в структуре тела вашего веб-сайта

Этот тег называют элементом разделения контента.

Сам контейнер является пустым, пока ему не пропишут конкретный класс со стилями. По большому счету, он используется для группировки контента, что позволяет легко стилизовать с помощью того же атрибута **class**

Когда вы поместите в контейнер некоторое количество тегов и определите к нему класс, также описанный в вашем css-файле - стили будут применяться ко всем элементам

# БЭМ

БЭМ расшифровывается как: блок, элемент, модификатор. Это такой набор абстракций, на который можно разбить интерфейс и разрабатывать всё в одних и тех же терминах. Как говорит сайт bem.info, БЭМ предлагает единые правила написания кода, помогает его масштабировать и повторно использовать, а также увеличивает производительность и упрощает командную работу.

Круто, да? Но зачем вам масштабируемость и командная работа, если вы один верстальщик на проекте, который не претендует на популярность Яндекса? Чтобы ответить на этот вопрос, нужно отмотать историю лет на 10 назад, когда это подход только начали формулировать.

В основу того, что мы сейчас называем БЭМом, легла идея независимых блоков, которую Виталий Харисов сформулировал и презентовал в 2007 году на первой российской конференции по фронтенду. Это было настолько давно, что тогда даже слова «фронтенд» ещё не было, тогда это называлось клиент-сайд.

Идея была в том, чтобы ограничить возможности CSS для более предсказуемых результатов. Использовать минимум глобальных стилей и каждый отдельный элемент страницы делать блоком со своим уникальным классом и стилями, которые полностью его описывают. Селекторы по элементам и ID, хрупкие связки вложенности — всё это заменялось на простые селекторы по классам. Каждый класс в стилях — это блок. Благодаря этому блоки можно легко менять местами, вкладывать друг в друга и не бояться конфликтов или влияния.

Потом появились абсолютно независимые блоки (АНБ), где у элементов внутри есть свой префикс с именем родителя, а состояния блоков снова дублируют класс, но уже с постфиксом. Подход обрёл черты современного БЭМа, одна из которых — многословность классов.

.block

.block\_mod

.block\_\_element

.block\_\_element\_mod

Через пару лет в Яндексе окончательно сформулировали нотацию БЭМ. Любой интерфейс предлагалось разделять на блоки. Неотделимые части блоков — элементы. У тех и других есть модификаторы.

<ul class="menu">

<li class="

menu\_\_item">

</li>

<li class="

menu\_\_item

menu\_\_item\_current">

</li>

</ul>

Например, блок поиска по сайту. Он может быть в шапке и в подвале — значит это блок. В нём есть обязательные части: поле поиска и кнопка «найти» — это его элементы. Если поле нужно растянуть на всю ширину, но только в шапке, то блоку или элементу, который отвечает за это растягивание, даётся модификатор.

Для полной независимости блоков мало назвать классы правильно или изолировать стили, нужно собрать всё, что к нему относится. В проекте по БЭМу нет общего script.js или папки images со всеми картинками сайта. В одной папке с каждым блоком лежит всё, что нужно для его работы. Это позволяет удобно добавлять, удалять и переносить блоки между проектами. Потом все ресурсы блоков при сборке объединяются в зависимости от задач проекта.

# Документация

Некоторые ссылки, которые помогут в изучении:

* [Методология БЭМ](https://ru.bem.info/methodology/)
* [Архитектура CSS](https://web-standards.ru/articles/css-architecture/)
* [CSS Modules](https://glenmaddern.com/articles/css-modules)
* [Продвинутый HTML и CSS](https://htmlacademy.ru/intensive/adaptive/)
* [Разделы 4 и 5](https://www.appbrewery.co/p/web-development-course-resources) содержат всю информацию для изучения