**Язык документа и его элементов**

При создании сайта полезно указывать, на каком языке написан его текст. Это поднимет шансы сайта оказаться выше в результатах поиска. А современные браузеры предложат автоматически перевести страницу, если она написана на языке, иностранном для пользователя.

Язык документа указывают в атрибуте lang тега html. Значение этого атрибута — двухбуквенный код языка, например, для русского — ru:

Скопировать кодHTML

<html lang="ru">

Языковые коды определены почти для каждого из языков. Стандарты определяет [Администрация адресного пространства интернет (IANA)](https://www.iana.org/assignments/language-subtag-registry/language-subtag-registry).

Атрибут lang универсален. Им мы можем определить язык отдельных блоков. Если текст одного из HTML-элементов написан на языке, отличающемся от основного языка страницы — следует отметить это в открывающем теге:

Скопировать кодHTML

<a href="#" lang="en">Hello, world!</a>

<a href="#" lang="de">Hallo Welt!</a>

<a href="#" lang="fr">Bonjour le monde!</a>

Поисковые машины благосклонны к ресурсам с такими уточнениями, в этом они видят признак тщательной проработки контента.

# Описание сайта для браузеров и поисковых машин

Мы уже говорили, что в head описывают правила, по которым браузер работает с веб-страницей. Мы разобрали три таких правила:

1. <meta charset="utf-8"> — определяет кодировку страницы;
2. <title></title> — задаёт имя во вкладке браузера;
3. <link rel="shortcut icon" href="favicon.ico"> — задаёт изображение во вкладке браузера.

Расширим этот список и наполним head дополнительной информацией. Но прежде разберёмся, кáк устроены метатеги:

* название элемента сообщает браузеру о смысле тега,
* атрибут name определяет назначение метатега,
* атрибут content содержит саму инструкцию.

Сайт могут открывать с разных устройств: компьютеров, планшетов или телефонов. Все эти устройства обладают разными экранами и разрешением. Такое обилие устройств с различными разрешениями экрана вынуждает нас следить за поведением страницы на разных экранах. Поэтому важно прописывать метатег <meta name="viewport" content="">. Этот метатег определяет поведение страницы в зависимости от размера дисплея.

Значение viewport сообщает браузеру, как выглядит видимая пользователю область страницы. Если этот метатег не указывать, сайт может отображаться по-разному на разных устройствах. Где-то всё будет работать хорошо, а где-то — нет. Поэтому лучше всегда прописывать этот метатег.

Укажем инструкции в атрибуте content:

Скопировать кодHTML

<meta name="viewport" content="width=device-width, initial-scale=1">

Здесь две инструкции:

* width=device-width означает «занимать всю ширину экрана»;
* initial-scale=1 — что страница на любом устройстве должна отображаться в масштабе 1:1, то есть не масштабироваться при загрузке.

### Метатеги для поисковиков

Также есть метатеги для поисковых машин: вроде Гугла и Яндекса. Алгоритмы поиска опираются на значения определённых метатегов. Поэтому, чтобы сайт был показан именно тем пользователям, которые им интересуются, следует эти метатеги прописывать.

Есть три метатега для поисковиков: описание сайта, его авторы и ключевые слова.

Описание контента прописывают в метатеге description. Он содержит небольшой текст — около абзаца — рассказывающий, чему посвящён сайт.

Автора или авторов сайта указывают в метатегах author. Обратите внимание: в один метатег meta name="author" вписывается имя одного автора.

Также есть метатег keywords — в нём прописывают ключевые слова сайта.

Значение атрибута content в этих метатегах указывается в свободной форме на основном языке сайта:

Скопировать кодHTML

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta name="description" content="Страница про различные веб-технологии">

<meta name="keywords" content="HTML5, CSS">

<meta name="author" content="Василий Петров">

<meta name="author" content="Петр Васильев">

Мы указали самые необходимые метатеги, но их ещё много. Для управления отображением ссылки при «шеринге» в соцсетях есть свои метатеги; для настройки систем аналитики — свои. Пока это всё знать необязательно, но по мере роста разработчика, ему нужно всё больше метатегов. Может наступить момент, когда раздел head вашего сайта станет больше, чем раздел body.

**Разнообразие контейнеров HTML5**

До сих пор мы структурировали страницу только контейнером div. Это традиционный и самый популярный блочный тег. Но есть и другие блоки для структурирования.

В стандарте HTML5 есть целое семейство контейнеров — семантических тегов. Визуально они не отличаются от div, но обладают скрытыми преимуществами. Если применять семантические теги:

* поисковым машинам будет проще прочитать и понять содержимое сайта,
* страница станет доступной для слабовидящих,
* код будет понятнее другим разработчикам.

**Семантические теги** размечают страницу с учётом смысла контента. По ним браузер отличает контент, «шапку» и «подвал» (колонтитулы), навигацию.

Самые распространённые семантические контейнеры:

* <header></header> — шапка сайта;
* <footer></footer> — подвал;
* <nav></nav> — блок меню (от англ. navigation, навигация);
* <main></main> — основное содержимое страницы; на многостраничных сайтах этот контейнер содержит всё, чем данная страница отличается от остальных. Имя тега по-английски означает «главный».
* <section></section> — организует несколько блоков в близкие по смыслу секции.

# Семантика и выразительность HTML

В предыдущем уроке мы разобрали контейнеры, которые описывают структуру страницы. Контейнеры описывают, из каких основных блоков состоит страница.

Но кроме контейнеров полезно применять теги, которые уточняют смысл определённых элементов. Цель у этого та же — семантические теги помогают понимать ваш сайт роботам: браузеру и поисковым машинам (Яндекс, Гугл).

Семантических тегов много. Чем больше вы будете писать и читать код, тем шире будет ваш «словарный запас». В этом уроке мы опишем популярные теги, которые позволят писать выразительный HTML-код, а не код, который «просто работает».

## Списки

В HTML есть два вида списков: маркированный и нумерованный.

**Маркированный список** используют, когда порядок элементов не важен: список покупок, чеклист. По умолчанию такой список оформлен буллитами — маркерами в виде точек:

Список документов:

* ксерокопия паспорта,
* фотография 3 на 4,
* заполненная анкета.

Чтобы создать список, используют контейнер ul. Внутрь контейнера помещают элементы li.

Скопировать кодHTML

Список документов:

<ul>

<li>ксерокопия паспорта,</li>

<li>фотография 3 на 4,</li>

<li>заполненная анкета.</li>

</ul>

**Нумерованный список** применяют, когда порядок имеет значение: в инструкции, описании процесса. Такой список маркирован цифрами, буквами или их комбинациями:

Чтобы участвовать в розыгрыше:

1. подпишитесь на канал,
2. поставьте ролику лайк,
3. напишите в комментариях свои контакты.

Такой список создают контейнером ol. Внутри него размещают такие же элементы li, как и в маркированном списке:

Скопировать кодHTML

Чтобы участвовать в розыгрыше:

<ol>

<li>подпишитесь на канал,</li>

<li>поставьте ролику лайк,</li>

<li>напишите в комментариях свои контакты.</li>

</ol>

Маркерами и нумерацией списков можно управлять из CSS. Для этого есть свойство list-style. У него много возможных значений: от квадратных маркеров до маркировки буквами армянского языка. Полный список вариантов есть [в документации CSS](https://html5book.ru/css-spiski/#list-style-type). Этим же свойством задают положение и изображение маркера.

Часто нужно скрыть маркер вовсе. Это делается так:

Скопировать кодCSS

ul {

list-style: none;

}

## Тег статьи

Значительную часть интернета составляют сайты с текстовым контентом: блоги и издания. Основной единицей информации там служит статья.

Для разметки статьи используют контейнер article. Он позволяет поисковым системам распознавать на странице блок, отведённый под статью.

Скопировать кодHTML

<article>

<h2>Заголовок статьи</h2>

<p>Основной текст статьи</p>

</article>

## Цитаты

Цитаты в тексте располагают между тегами <blockquote></blockquote>.

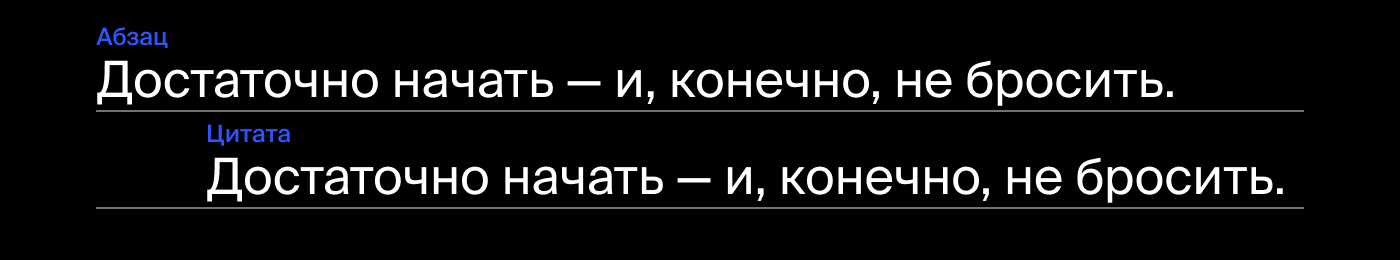
У этого тега есть необязательный, но полезный атрибут cite. Он позволяет указать автора цитаты.

Скопировать кодHTML

<blockquote cite="https://praktikum.yandex.ru">Достаточно начать — и, конечно, не бросить.

Мы поможем приступить к учёбе и справиться с трудностями.</blockquote>

По умолчанию цитата оформлена отступами:



## Таблицы

Иногда нужно оформить таблицу. Для этого есть тег <table></table>.

Таблица состоит из строк. Их задают тегами <tr></tr>. Строки разбивают на ячейки тегами <td></td>. Исключение составляют ячейки первой строки, потому что они содержат заголовки колонок. Поэтому такие ячейки задают тегами <th></th>.

Скопировать кодHTML

<table>

<tr> *<!-- первая строка с заголовками - поэтому разбиваем на ячейки тегами th -->*

<th>строка 1 ячейка 1 - заголовок первой колонки</th>

<th>строка 1 ячейка 2 — заголовок второй колонки</th>

<th>строка 1 ячейка 3 — заголовок третьей колонки</th>

</tr>

<tr> *<!-- вторая и последующие строки — содержимое, поэтому разбиваем их тегами td-->*

<td>строка 2 ячейка 1</td>

<td>строка 2 ячейка 2</td>

<td>строка 2 ячейка 3</td>

</tr>

<tr>

<td>строка 3 ячейка 1</td>

<td>строка 3 ячейка 2</td>

<td>строка 3 ячейка 3</td>

</tr>

<tr>

<td>строка 4 ячейка 1</td>

<td>строка 4 ячейка 2</td>

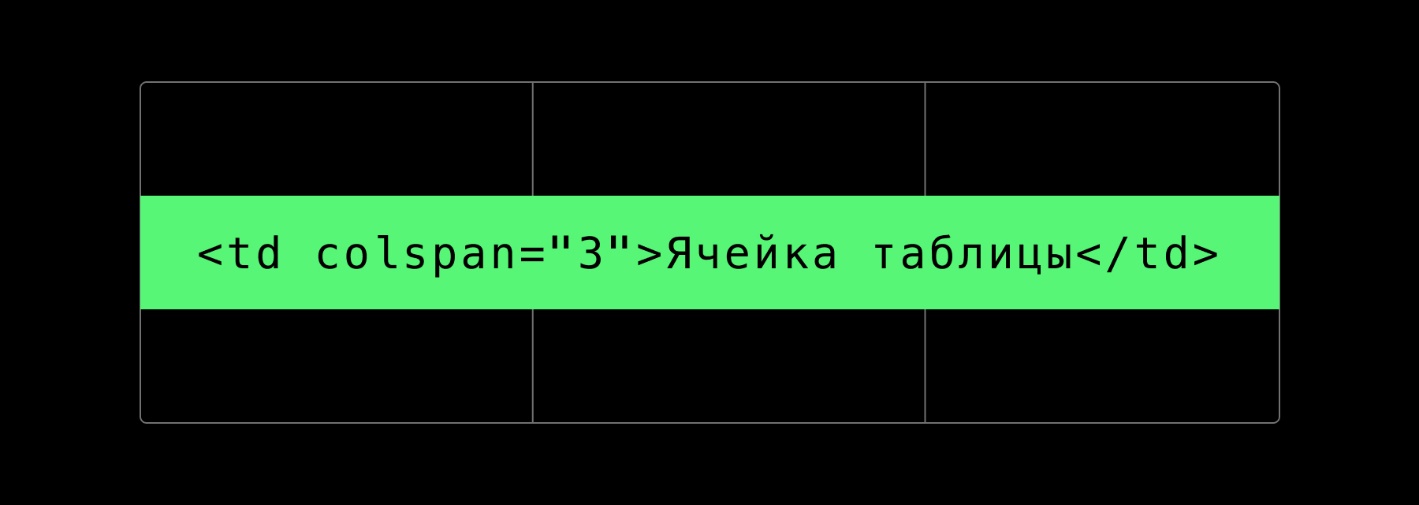
<td>строка 4 ячейка 3</td>

</tr>

</table>

Располагайте изображения и текст только внутри элементов td и th. Остальные табличные теги — table и tr — предназначены исключительно для разметки.

В самой простой ситуации число ячеек в каждой строке одинаковое. Однако ячейка таблицы может занять место нескольких ячеек:



Ячейка второй строки занимает 3 столбца

За это отвечают атрибуты rowspan и colspan. Запись <td colspan="3">Ячейка таблицы</td> означает, что ячейка займёт место трёх столбцов. Следовательно, в строке, где размещена эта ячейка, должно быть на две ячейки меньше, чем в прочих строках — их место «съела» ячейка с colspan="3".

Скопировать кодHTML

<table>

<tr>

<th></th>

<th></th>

<th></th>

</tr>

<tr>

<td colspan="3">Ячейка таблицы</td> *<!-- в этой строке только 1 ячейка -->*

</tr>

<tr>

<td></td>

<td></td>

<td></td>

</tr>

</table>

Верстайте таблицами только настоящие таблицы. Смета, размещённая на сайте, может быть свёрстана в table, а вот секция сайта, выстроенная в сетку — нет. Не используйте тег <table>, чтобы расположить элементы в строку или в колонку — для этого есть флекс.

Ещё раз подчеркнём, что семантическое разнообразие HTML познаётся только опытным путём. Обращайте внимание на специфические элементы и задавайтесь вопросом: «Неужели для этого элемента не придумали специального тега?» Задайте этот же вопрос поисковой системе — и, скорее всего, специализированный семантический тег найдётся.

# Методология БЭМ

В 1926 году был издан роман Франца Кафки «Зáмок». Герой произведения приезжает в деревню, чтобы выполнить землемерные работы. Но в течение всего романа ходит от одного чиновника к другому, пытаясь выяснить, какую работу и где нужно делать. Его лишь отправляют из одной канцелярии в другую, оттуда — в третью. И так до бесконечности по огромному и бессмысленному бюрократическому лабиринту, из которого нет выхода и в котором не ясно, что делать.

Примерно так же могут чувствовать себя разработчики, когда трудятся над проектом достаточно долго.

Сперва код кажется простым и понятным. Но в команду приходят новые люди, дописывают новую функциональность. В какой-то момент кода становится так много, что сложно понять, какая его часть за что отвечает.

Приходится писать документацию к коду и в ней подробно расписывать, что делает каждая часть сайта. Это занимает уйму времени: сначала у того, кто пишет документацию, а затем — у каждого, кто её читает.

Но это не единственная проблема. В больших проектах нередко есть очень похожие друг на друга части. Например, поисковая строка в разных сервисах Яндекса практически не изменяется. Если писать каждый раз одно и то же с нуля, придётся тратить лишнее время и силы, а значит — и деньги.

Чтобы не тратить время и деньги на ветер, следует определиться с правилами написания кода — методологией.

## Что такое методология?

Хорошая методология позволяет писать самодокументируемый код. Это когда, читая код, понятно, чтó он делает, и документация не нужна. Также методология решает проблему повторного использования частей кода: отдельные компоненты становится легко переносить из проекта в проект.

Единого стандарта построения структуры и именования классов не существует. Есть несколько распространённых методологий: SMACSS, ITCSS, БЭМ.

Мы будем работать с БЭМ, потому что это одна из самых популярных в мире методологий.

## Что такое БЭМ?

БЭМ была разработана в Яндексе, а сейчас стала стандартом рынка разработки, в том числе зарубежного. В основе методологии лежат два набора правил: для классов и для файлов.

**Правила именования классов.** Следования этим правилам как раз и делают код самодокументируемым. По коду становится легко понять, какой блок за что отвечает, где находится и какими особенностями обладает.

**Правила организации файлов.** БЭМ определяет, как разбить код на файлы и как эти файлы организовать. Это позволяет использовать код повторно, не переписывая его.

На разных сайтах используются одинаковые компоненты: строка поиска, кнопки, поля форм. Если следовать БЭМ, вы без проблем сможете повторно применять фрагменты одного проекта в другом.

### Блок, элемент, модификатор

БЭМ — аббревиатура от «блок, элемент и модификатор». Именно с этими тремя сущностями оперирует БЭМ.

**Блок** — это независимый фрагмент страницы: шапка сайта, логотип, строка поиска, кнопка. Это не техническое понятие, а логическое. В [официальной спецификации БЭМ](https://ru.bem.info/methodology/key-concepts/) блок описан как «логически и функционально независимый компонент страницы».

Блок можно вырвать из контекста, разместить на другой странице или даже сайте. При этом блок продолжит выглядеть и работать как прежде.

Блоки можно вкладывать один в другой.

**Элемент** — это часть блока, которая не может существовать отдельно от него. Так кнопка «Отправить» — это часть веб-формы, вне формы она не используется. Элемент не может использоваться вне своего блока-родителя.

**Модификатор** — это свойство блока или элемента, которое переопределяет их параметры. Например, существует блок logo. Он используется в двух местах: в шапке сайта и в подвале. В обоих случаях он выглядит и ведёт себя одинаково, но в шапке логотип чёрный, а в подвале — белый. В этой ситуации применяется модификатор, который изменит только цвет логотипа.

### Зачем на это тратить время

Изучение БЭМ потребует много времени и усилий. Как и построение структуры. Но все эти затраты окупаются.

Представьте, что в какой-то компании разрабатывают не по БЭМ. Стили могут выглядеть так:

Скопировать кодCSS

*/\* стиль для третьего по счёту тега <a>, вложенного на первом уровне в <div>,*

*если у этого <a> атрибут href заканчивается на '.pdf': \*/*

div > a[href$='.pdf']:nth-of-type(3) {

*/\* здесь стили \*/*

}

Когда в компанию приходит начинающий разработчик, он потратит больше времени на расшифровку такого синтаксиса и на поиск причин и взаимосвязей, чем на работу над кодом.

Та же проблема возникнет и у вас, когда вы вернётесь к своему коду, написанному пару месяцев назад.

Если же сайт свёрстан по спецификации БЭМ, новичку проще разобраться в коде: каждый селектор — это просто имя класса, по которому понятно, какой блок оно описывает и каких последствий стоит ожидать при изменении кода.

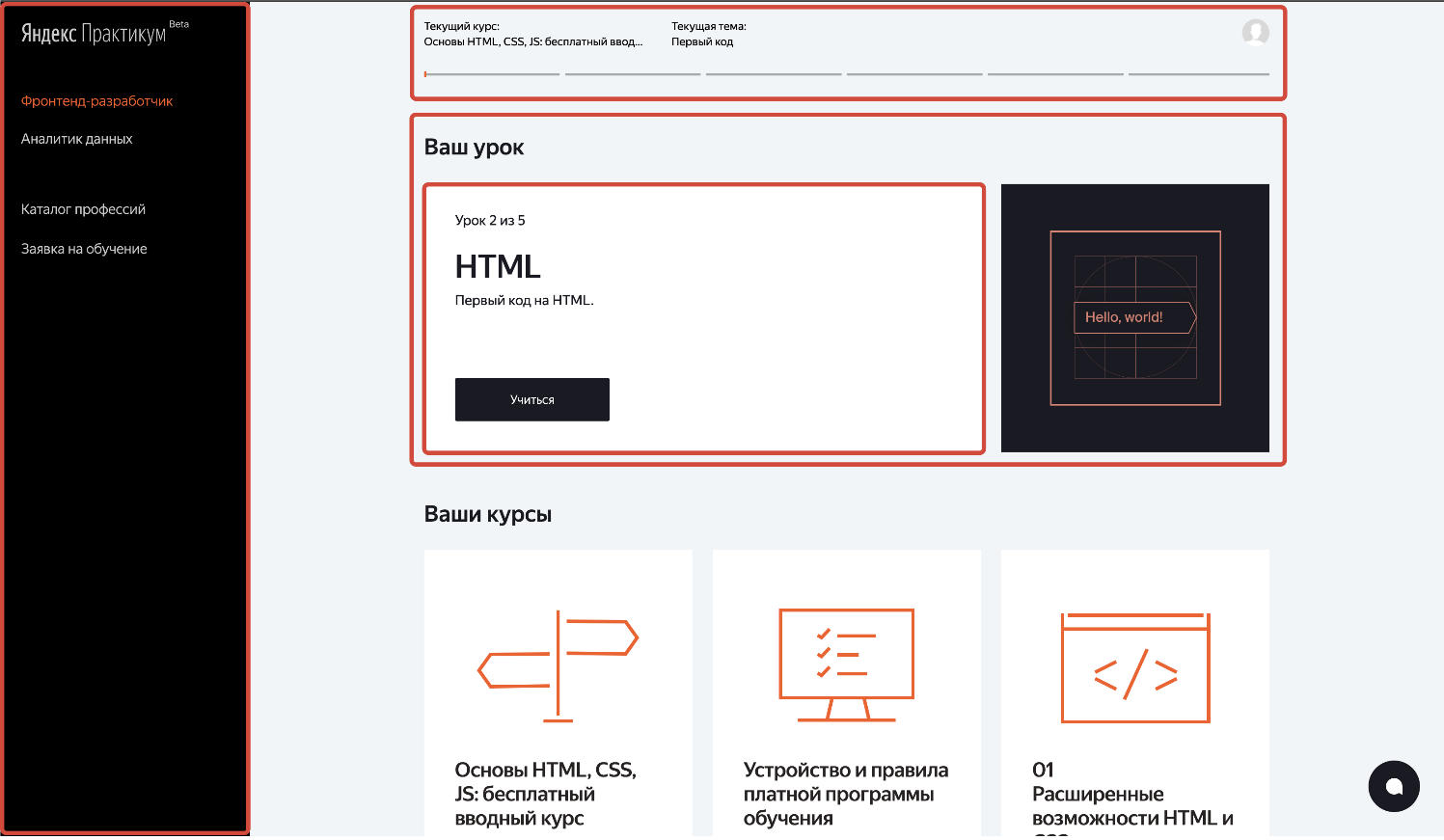
Официальная документация БЭМ: [https://ru.bem.info](https://ru.bem.info/)

# Блоки

В прошлом уроке разобрались, что БЭМ — это методология именования классов и организации файлов. Аббревиатура «БЭМ» расшифровывается как «блок, элемент, модификатор».

Начнём с блоков. Найдите на своих любимых сайтах повторяющиеся фрагменты — это и есть блоки.

Одни блоки можно вкладывать в другие. В блоке header может располагаться логотип — блок logo. Но logo также можно поместить и в другой контекст — например, в подвале. В этом случае нам не придётся писать новых стилей для логотипа. Даже перейдя в подвал, блок logo сохранит все свои CSS-свойства.



На скриншоте красными рамками выделены примеры блоков

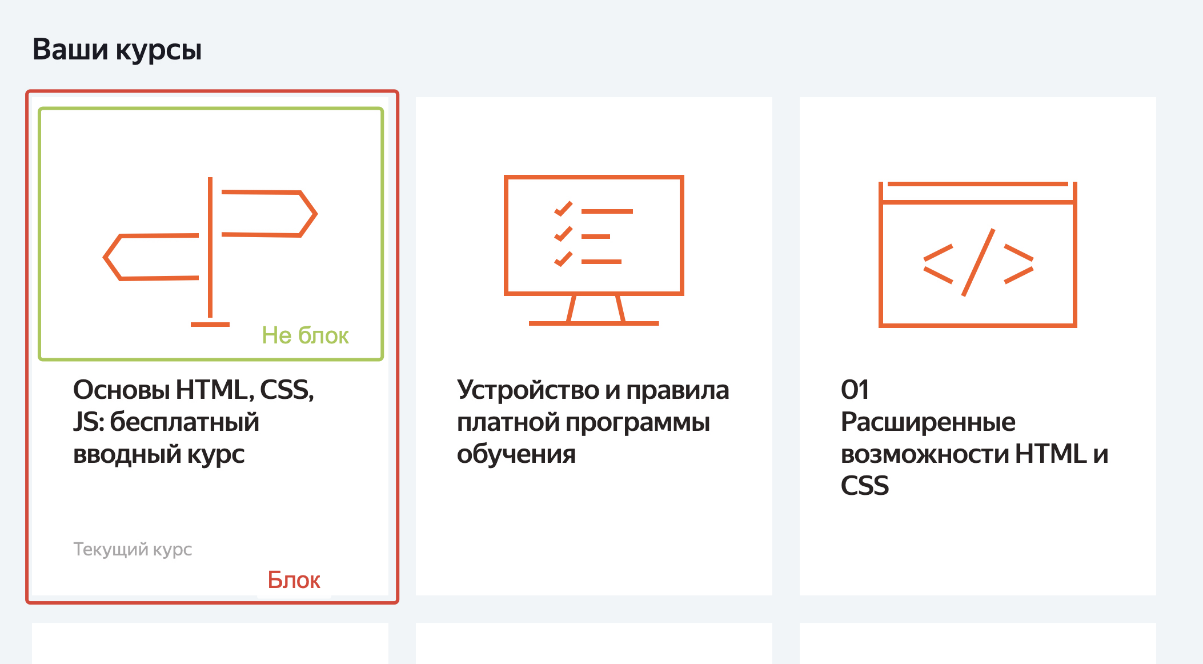
В этом уроке разберёмся, как определять блоки, называть их, верстать и стилизовать.

### Найдите блоки на странице

Это исследовательская часть работы.

У вас будет прототип и макет сайта от дизайнера. По этим двум документам можно найти и выделить БЭМ-блоки во всём проекте. Анализируйте каждый фрагмент страницы и задавайте себе вопросы:

* может ли этот фрагмент быть использован в другом месте страницы, сайта или на другом ресурсе?
* может ли этот фрагмент существовать отдельно от своего родителя, не теряя смысла?



На скриншоте каждая карточка курса — это блок. Карточку можно переместить в другую часть сайта: рядом есть похожие карточки. Это копии — экземпляры одного и того же блока. Такие же блоки есть и на страницах других профессий, они отличаются только текстом и картинками.

Но картинка на карточке — это уже не блок. Картинку нельзя вырвать из контекста и использовать отдельно. Пока изображение находится внутри карточки, мы понимаем, что это иллюстрация ко всей карточке. Но если вырвать картинку из контекста, мы не сможем понять, чтó она значит. В терминологии БЭМ картинка на карточке — это «элемент».

### Дайте блокам имена

В мировой практике есть разные подходы к именованию БЭМ-сущностей. Мы будем использовать те же правила, что и программисты Яндекса, потому что это самый распространённый подход. Про альтернативы вы можете [почитать на сайте БЭМ](https://ru.bem.info/methodology/naming-convention/#%D0%B0%D0%BB%D1%8C%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B5-%D1%81%D1%85%D0%B5%D0%BC%D1%8B-%D0%B8%D0%BC%D0%B5%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F).

Разберёмся с правилами именования.

**Правило 1.** Имена блокам прописывают в атрибуте class:

Скопировать кодHTML

<header class="header"></header>

**Правило 2.** Имя блока должно отражать его содержание и назначение, но не состояние или внешний вид:

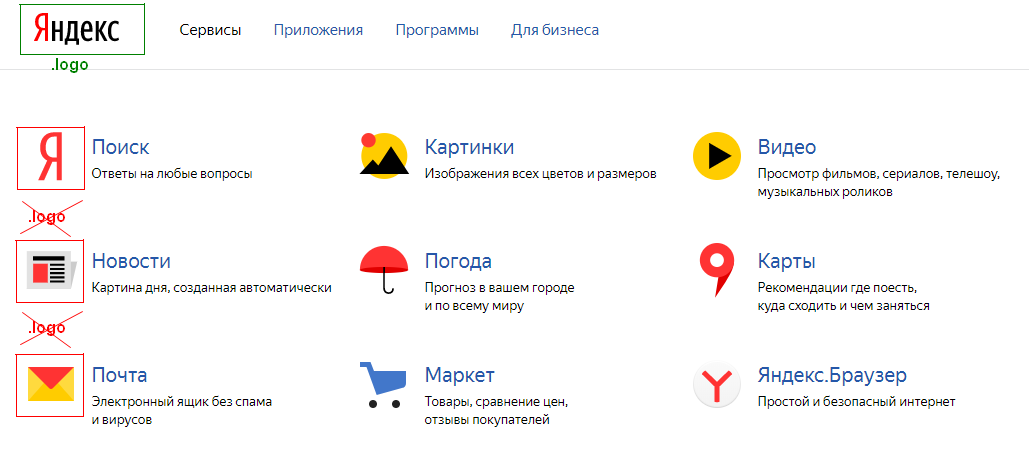
|  | **ПЛОХО** | **ХОРОШО** |
| --- | --- | --- |
| Шапка сайта | top-of-page | header |
| Меню | left-menu | menu |
| Текст | grey-colored-text | text |
| Левая колонка | black-column | sidebar-left |
|  |  |  |

**Правило 3.** Имя блока может состоять из нескольких слов. В этом случае слова разделяют дефисами.

**Правило 4.** Имя блока должно быть уникальным: у двух разных блоков не может быть одного имени.

Например, на странице есть логотип, который находится в шапке сайта и в подвале. Логотип — это типичный блок в терминологии БЭМ. Поэтому мы можем дать ему имя logo.

Но на том же сайте есть список проектов. Описание каждого из них сопровождается логотипом проекта. Эти логотипы имеют иное значение, выглядят и ведут себя иначе, нежели логотип сайта. Потому логотип проекта — это иной блок. Он не должен называться logo.



### Разместите блоки на странице

Располагайте блок так, чтобы он никак не влиял на окружение.

Главная «фишка» блока в том, что его можно использовать повторно. Поэтому не задавайте блоку свойств, влияющих на другие DOM-элементы. Не задавайте блоку внешних отступов и позиционирования — иначе в новом контексте все эти свойства придётся переопределять.

Как правильно позиционировать блок, мы расскажем ниже.

### Задайте стили

Блок везде выглядит одинаково. Поэтому стили блока должны быть независимыми от окружающих и родительских блоков.

CSS-селектор блока всегда пишут по его имени, то есть по названию класса. И поскольку блок обладает уникальным именем, стили, прописанные для него, никак не затронут другие фрагменты страницы.

Скопировать кодCSS

*/\* Не по БЭМ: блоку заданы внешние отступы и позиционирование со смещением.*

*Эти свойства влияют на окружающие блок DOM-элементы. \*/*

.logo {

margin: 20px;

position: absolute;

left: 10px;

}

*/\* Не по БЭМ: использован вложенный селектор. \*/*

.header .logo {

padding: 20px;

}

*/\* А вот так — по БЭМ \*/*

.logo {

padding: 20px;

position: relative; */\* не влияет на позицию блока, если не указано смещение \*/*

}

По БЭМ можно стилизовать фрагменты только по именам классов. Название тегов, идентификаторы или любые другие признаки нельзя использовать для селекторов. Это одна из технических основ стандарта.

Скопировать кодCSS

*/\* Не по БЭМ: не следует использовать имена тегов в таблице стилей \*/*

div.logo {

*/\* styles \*/*

}

p {

*/\* styles \*/*

}

*/\* Не по БЭМ: не следует использовать вложенность \*/*

div .logo {

*/\* styles \*/*

}

*/\* Не по БЭМ: не следует использовать ID элементов в таблице стилей \*/*

#logo {

*/\* styles \*/*

}

*/\* А вот так — по БЭМ \*/*

.logo {

*/\* styles \*/*

}

### Создайте контейнеры и дайте блокам имена

Опишите блоки в вёрстке подходящими элементами: div, nav, section, header, footer.

Блок — это не всегда контейнер. Блоком могут выступать заголовки, абзацы, поля форм, кнопки и вообще любые элементы, если они созданы для многоразового использования.

Дайте каждому блоку имя и запишите его в свойство class. В БЭМ стили прописывают именно селекторам классов, а имена тегов и идентификаторы не используют.

Скопировать кодHTML

<header class="header"> *<!-- блок header: класс совпадает с именем тега -->*

<nav class="menu"> *<!-- в header вложен блок по имени menu -->*

</nav>

</header>

### Старайтесь не пользоваться normalize.css и reset.css

Согласно спецификации БЭМ, нельзя подключать файлы normalize.css и reset.css: они влияют на внешний вид всей страницы, а стили каждого блока должны быть независимы. Тем не менее, разработчики часто подключают эти файлы к своим проектам, если в этом есть практическая польза. В учебных заданиях мы тоже ими пользуемся.

БЭМ — отличная методология, которую нужно знать и пользоваться ей. Но следование правилам не должно быть фанатичным: выбор инструментов зависит от задачи и контекста. Не пренебрегайте независимостью блока от окружения. Задавайте внешний вид блока так, чтобы и без normalize.css всё работало правильно.

# Элементы

Предыдущий урок был посвящён блокам: из них состоит каркас веб-страницы. Теперь перейдём ко второй букве в аббревиатуре БЭМ — к элементам.

Блок — самостоятельная единица. Его можно вырвать из контекста и использовать в другой части сайта. Одни блоки могут быть вложены в другие, но при этом каждый вложенный блок может существовать самостоятельно.

**Элемент** — это составная часть блока, которую нельзя создавать и использовать в отрыве от блока.

Скопировать кодHTML

<header class="header"> *<!-- это блок, «шапка» страницы -->*

<a href="#" class="header\_\_link">[Ваше имя]</a>

</header>

Блок header можно целиком скопировать в код страницы другого сайта, он и там исправно послужит шапкой.

Блок — как готовый пирог: его можно переставлять со стола на стол, можно резать ножом или ломать на куски — вкус пирога от этого не меняется. Элемент же похож на ингредиент пирога (сырое яйцо, мука или масло): он выполняет определённую задачу, но его не подашь как отдельное блюдо.

Компонент с классом header\_\_link — это элемент, ссылка внутри шапки. Такая ссылка должна быть только в шапке и нигде более.

### Именование элементов

БЭМ рекомендует именовать классы так, чтобы сразу было ясно, что это за компонент — элемент или блок. Рекомендации такие:

* Название класса блока — слово или несколько слов, разделённых дефисом. Например, header и header-block — корректные названия.
* Элементы именуют так: сначала пишут название блока, к которому относится элемент, потом два нижних подчёркивания \_\_, и затем — имя элемента. Если название элемента состоит из двух слов, вместо пробелов ставят дефис.

Правильное имя для ссылки в шапке — header\_\_link, то есть  
шапка\_\_ссылка.

Для стилизации элементов используют только селекторы классов. Так же, как и для стилизации блоков.

### Правила

Элемент — это часть блока. Размещать элементы вне блока нельзя.

Также нельзя создавать элемент элемента. Запись block\_\_element\_\_element некорректна. Элементы могут быть вложены друг в друга, но название класса всегда привязано к имени блока:

Скопировать кодHTML

*<!-- ПРАВИЛЬНО -->*

<div class="main-content">

<p class="main-content\_\_text">

В этом параграфе <a href="#" class="main-content\_\_link">ссылка</a>

</p>

</div>

*<!-- НЕПРАВИЛЬНО -->*

<div class="main-content">

<p class="main-content\_\_text">

В этом параграфе <a href="#" class="main-content\_\_text\_\_link">ссылка</a>

</p>

</div>

Второй пример неправильный, поскольку содержит элемент элемента — класс main-content\_\_text\_\_link.

Имя элемента описывает содержание, а не внешний вид или состояние: main-content\_\_text — хорошее имя, main-content\_\_red — плохое.

Блок может не содержать элементов. Но если элементы нужны, они всегда являются частью блока.

# Модификаторы в БЭМ

Вы уже знаете, что такое блоки и элементы, но мы напомним.

**Блоки** — основные единицы проектирования веб-страницы: шапка сайта, меню, подвал, карточка проекта. Блоки существуют независимо, их можно вырвать из контекста и расположить в другой части страницы или сайта.

**Элементы** — это части блоков. Например, пункт меню или картинка, сопровождающая текст. Без блоков элементы не имеют смысла: их нельзя никуда переместить, они потеряют смысл.

Переходим к модификаторам — третьей букве в аббревиатуре БЭМ.

**Модификаторы —** это не фрагменты страницы. Модификаторы определяют состояние и поведение блоков или элементов.

Скопировать кодHTML

<nav class="menu">

<a href="#" class="menu\_\_link">Первые шаги в HTML и CSS</a>

<a href="#" class="menu\_\_link">Библиотеки JavaScript</a>

<a href="#" class="menu\_\_link">Практика HTML, CSS и JavaScript</a>

</nav>

Здесь menu — блок, menu\_\_link — элемент.

Предположим, мы находимся на странице «Первые шаги в HTML и CSS». Хорошо бы показать пользователю, что первая ссылка ведёт на текущую страницу. Например, подчеркнуть эту ссылку.

Для этой задачи нам понадобятся модификаторы — они задают элементам уникальные свойства. Так мы можем изменить внешний вид ссылки только на одной из страниц сайта.

В нашем случае подойдёт модификатор active, который будет указывать на то, что ссылка активна.

Скопировать кодHTML

<nav class="menu">

<a href="#" class="menu\_\_link menu\_\_link\_active">Первые шаги в HTML и CSS</a>

<a href="#" class="menu\_\_link">Библиотеки JavaScript</a>

<a href="#" class="menu\_\_link">Практика HTML, CSS и JavaScript</a>

</nav>

Название класса модификатора составляют так: сначала идёт имя блока или элемента, за ним — имя модификатора через нижнее подчёркивание.

Для модификатора блока: блок\_модификатор.

Для модификатора элемента: блок\_\_элемент\_модификатор.

## Виды модификаторов

Модификаторы делятся на два типа — булевы и вида «ключ-значение».

**Булевы модификаторы** используют, когда у элемента или блока может быть только два состояния. Например, у пункта меню может быть два состояния: обычное и активное (если пользователь находится на странице этого пункта меню).

Обычное состояние описывают стилями элемента menu\_\_link, а активное состояние — модификатором menu\_\_link\_active.

Скопировать кодHTML

<nav class="menu">

<a href="index.html" class="menu\_\_link menu\_\_link\_active">Главная</a>

<a href="about.html" class="menu\_\_link">О нас</a>

<a href="contacts.html" class="menu\_\_link">Контакты</a>

</nav>

**Модификаторы «ключ-значение»** показывают различные варианты внешнего вида, состояния или поведения.

Блок с результатом решения задачи может вести себя по-разному: сообщать статус успешного прохождения, сообщать об ошибке или содержать дополнительное сообщение. Эти три модификатора связаны одним ключом — типом сообщения, но с разными значениями.

Скопировать кодHTML

<div class="solution-status solution-status\_type\_success"></div>

<div class="solution-status solution-status\_type\_error"></div>

<div class="solution-status solution-status\_type\_message"></div>

Запомните разницу в записи модификаторов.

Булевый: блок\_\_элемент\_модификатор или блок\_модификатор.

Ключ-значение: блок\_\_элемент\_ключ\_значение или блок\_ключ\_значение.

## Важное о модификаторах

Модификаторы могут быть и у блоков, и у элементов, причём их может быть несколько для разных случаев.

Примерно так может выглядеть код маленькой отключённой кнопки в блоке «форма подписки на новости»:

Скопировать кодHTML

<button class="signup-form\_\_button signup-form\_\_button\_disabled signup-form\_\_button\_size\_m">Текст кнопки</button>

При записи модификаторов в атрибут class записывают 2 значения: имя блока или элемента и название модификатора.

Если на странице у элемента всего два состояния, но потенциально их может стать больше — используйте модификаторы вида ключ-значение.

# Способы модификации блоков

БЭМ построен вокруг идеи создания независимых блоков, которые можно использовать повторно. Но редко можно встретить интерфейс, в котором ни один блок не меняет своих размеров или расположения в разных контекстах.

Посмотрите, как сделана шапка [на сайте Авито](https://www.avito.ru/). При прокрутке страницы с поиском, шапка следует за пользователем. Но если открыть страницу товара, шапка остаётся прижата к верхней части сайта:

Описывать такое поведение на уровне блока не выйдет — нужен другой подход.

Ещё один распространённый случай — блок с иконками социальных сетей. В подвале его могут разместить слева, а в блоке с контактами — справа. Или наоборот.

В общем, разработчикам постоянно требуется изменить внешний вид блока в каком-то контексте или дополнить его поведение. Есть два способа сделать это, и один из них вам уже знаком — использовать модификатор блока.

## **Использование модификатора**

Модификаторы используют, чтобы изменить один конкретный экземпляр блока. Хороший пример — модификатор, который меняет цвет текста с темного на светлый при тёмном фоне страницы. Установка и снятие модификатора должны изменять только сам блок и не влиять на его окружение — другие блоки.

Скопировать кодHTML

<button class="button button\_disabled"></button>

<div class="logo logo\_theme\_dark"></div>

<div class="response-status response-status\_type\_rejected"></div>

Модификаторы отлично подходят, чтобы обозначать статус или состояние конкретного блока. Но есть и другие ситуации.

## Когда модификаторы не подойдут

**Модификаторы не всегда подходят для расположения элементов.** Технически их можно использовать, но код становится менее понятным, потому что модификатор не сообщает контекста.

Например, мы делаем интернет-журнал. Все картинки на сайте мы обрамили рамкой, добавили небольшую тень:

Скопировать кодCSS

.picture {

border: 1px solid grey;

box-shadow: 0 0 3px grey;

}

Допустим, картинки внутри статей нужно расположить по центру. В принципе мы можем применить для этого модификатор:

Скопировать кодCSS

.picture\_pos\_centered {

margin: auto;

}

*/\* Но не стоит так делать \*/*

При таком подходе модификатор не сообщает контекста, в котором расположен блок. Модификатор говорит лишь «это стили картинки, расположенной по центру». А полезнее было бы знать «это стили картинки, находящейся в зоне контента».

**Модификатор также не годится для стилизации однотипных элементов, расположенных в разных блоках.** Например, мы верстаем шапку сайта и хотим активные ссылки окрасить в жёлтый. Для этого создаём модификатор:

Скопировать кодCSS

.header\_\_link\_active {

color: yellow;

}

Затем мы хотим окрасить в жёлтый цвет ссылки в контенте. И тут модификатор нам не поможет. Имя модификатора связано с именем блока, поэтому мы не можем добавить одному блоку модификатор другого.

Скопировать кодHTML

<div class="content">

*<!-- так делать нельзя -->*

<a class="content\_\_link header\_\_link\_active">

</div>

Поэтому для позиционирования и стилизации однотипных элементов модификаторы не годятся. Для этих целей лучше подходят «миксы».

## Микс: смешивание стилей

Микс — это смешение стилей двух блоков. В этом случае мы задаём одному блоку два имени и получаем в результате новый блок, который обладает свойствами двух смешанных блоков.

Микс можно сделать двумя способами.

### Микс для общих правил

Первый способ предназначен для того, чтобы определить общие правила для разных блоков. Очень распространённая задача — задать разным блокам одинаковые стили текста и шрифта. Для этого мы стилизуем сразу несколько блоков:

Скопировать кодHTML

<nav class="menu"></nav>

<section class="content"></section>

<footer class="footer"></footer>

Мы хотим задать блокам content и footer одинаковые текстовые свойства. Для этого создадим новый селектор и опишем в нём, как должен выглядеть текст. А затем добавим этот класс всем блокам, где хотим видеть такой текст.

Создадим БЭМ-блок main-text и стилизуем его:

Скопировать кодCSS

.main-text {

color: #000;

font-family: 'Arial', 'Helvetica', sans-serif;

font-size: 18px;

line-height: 1.5;

font-weight: normal;

}

И примешаем созданный блок к нужным блокам. Это и называется «сделать микс»:

Скопировать кодHTML

<nav class="menu"></nav>

<section class="content main-text"></section>

<footer class="footer main-text"></footer>

Нужным блокам присвоены нужные стили. Мы произвели микс на уровне классов и тем самым стилизовали группу блоков.

### **Примешивание элемента блоку**

Вернёмся к примеру с иконками социальных сетей. Их часто размещают в подвале сайта. Например, такой блок может располагаться по центру подвала. Таким образом, блок социальных сетей нужно сдвинуть внутри подвала свойством margin.

Скопировать кодHTML

<footer class="footer">

<div class="social-icons"></div>

</footer>

Согласно БЭМ описывать внешние отступы для блока не рекомендуется. Но нам нужно задать соцсетям margin: 50px auto;.

Хорошая идея — подмешать к стилям блока social-icons класс БЭМ-элемента, который относится к родительскому блоку:

Скопировать кодHTML

<div class="footer">

<div class="social-icons footer\_\_social-icons"></div>

</div>

Блок social-icons теперь представляет собой одновременно и самостоятельный блок, и элемент блока footer по имени footer\_\_social-icons.

Так мы можем уточнить поведение нужного блока, сохранив его независимость: ведь margin мы указываем не для блока, а для элемента.

Скопировать кодCSS

.footer\_\_social-icons {

margin: 50px auto;

}

Все эти миксы могут казаться сложными поначалу. Но со временем вы наберётесь опыта в именовании селекторов и модифицировании блоков и элементов.

**Приведём в порядок код**

В прошлых уроках мы говорили про БЭМ. Одним из основных требований этого стандарта является использование исключительно селекторов классов. Согласно БЭМ нельзя использовать селекторы по тегам.

Но БЭМ-правила не единственные. Есть дополнительные правила, которых следует придерживаться при написании кода.

Одно из таких правил вы уже знаете и, мы надеемся, применяете. Это использование табуляции для сдвига вложенных элементов вправо в коде.

Ещё одно важное правило касается порядка записи CSS-селекторов. Браузер читает CSS-код сверху вниз и применяет только после того, как дочитает все стили до конца. Поэтому правила, описанные в коде ниже, имеют приоритет перед теми, что описаны выше.

Скопировать кодCSS

.menu {

background-color: #fff;

}

*/\* styles \*/*

.menu {

background-color: #000; */\* этот селектор более приоритетный, потому что находится ниже в коде\*/*

}

Технически в этом примере нет ошибки, но явно видна небрежность разработчика. В результате у блока menu будет чёрный фон: именно это правило написано последним.

Старайтесь писать селекторы в том же порядке, в каком они расположены в HTML-коде: не стоит размещать стили для footer выше стилей для header.

Придерживайтесь примерно такого порядка:

* Напишите самые общие правила. Например, для body. Но не забудьте, что даже к уникальным тегам следует обращаться по селектору класса.
* Затем напишите стили вспомогательных блоков — например, тех, которые вы используете только для микса.
* После этого записывайте первый блок, встретившийся в HTML.
* Потом его модификатор, если он есть.
* Затем первый элемент этого блока.
* Затем модификаторы этого элемента.

Таким же образом описывайте в CSS все блоки, читая HTML-код сверху вниз.

# Идентификаторы

Статьи на Википедии начинаются с содержания. Содержание состоит из ссылок, которые не ведут на другую страницу. Вместо этого по ссылке пользователь переходит на другую часть той страницы, на которой находится:

Такие гиперссылки называют якорными. Они ссылаются на конкретный объект на странице. Причём этот объект может находиться как на текущей странице, так и на любой другой.

Для установки якорной ссылки необходимо указать HTML-элемент, на который она ведёт. Например, на определённый заголовок в статье.

Как указать этот элемент? Использовать имя тега или имя класса не выйдет: эти имена могут повторяться. Нам же нужна уникальная метка элемента: идентификатор.

**Идентификатор элемента** задаётся атрибутом id — от англ. identifier, идентификатор. Идентификатор элементу присваивает разработчик. Идентификатор должен быть уникален в пределах одной страницы.

Например, идентификатор part-11 в коде выглядит так:

Скопировать кодHTML

<h2 id="part-11">Часть 11: идентификаторы</h2>

Когда мы установили id для элемента, можем ссылаться на него. Например, в адресе ссылки, установив якорную ссылку на той же веб-странице или на какой-то другой.

Если якорная ссылка и элемент, на который она ссылается, находятся на одной странице, то в качестве адреса ссылки нужно прописать хеш — символ «решётка», #, — а затем имя идентификатора без пробела:

Скопировать кодHTML

<a href="#part-11">Якорная ссылка с той же страницы</a>

А чтобы сослаться на этот элемент с любой другой страницы, нужно указать полный адрес страницы, на которой находится объект, а после — хеш и идентификатор элемента:

Скопировать кодHTML

<a href="https://my-site.com/article#part-11">Якорная ссылка с другой страницы</a>

При переходе на [страницу с официальным описанием якорных ссылок на W3Org](https://www.w3.org/TR/html401/struct/links.html#h-12.2.3), страница автоматически прокручивается вниз. Это якорная ссылка ведёт на элемент с id="h-12.2.3", который расположен примерно в середине страницы. Адрес ссылки выглядит так: <https://www.w3.org/TR/html401/struct/links.html#h-12.2.3>.

Хотя идентификаторы не следует использовать в селекторах правил CSS, технически это возможно. В CSS-селекторе перед названием идентификатора ставится хеш:

Скопировать кодCSS

#part-11 {

color: #ff0000;

}

Придумывая идентификатор, соблюдайте три правила, чтобы код был читаемым и для людей, и для роботов:

* не начинайте id с цифр или спецсимволов;
* если в названии несколько слов, разделяйте их дефисами или нижними подчёркиваниями вместо пробелов;
* подбирайте простые слова из английского языка.

Соблюдайте эти правила, и ваш код будет понятен любому.

# Псевдоклассы

До сих пор мы определяли внешний вид элементов при загрузке страницы: давали блоку имя, затем описывали стили для этого блока. Но часто элементы нужно стилизовать в зависимости от контекста.

Например, покрасить ссылку в другой цвет при наведении мыши. Чтобы решить эту задачу, нужно чтобы кто-то делал умозаключения: «указатель мыши находится на этой ссылке, значит, следует поменять её цвет». Или: «указатель не на этой ссылке, значит, её цвет менять не нужно». Кто-то должен анализировать ситуацию и принимать решения.

К счастью, браузер способен к анализу, пусть и очень несложному. Браузер умеет находить CSS-селекторы и добавлять в них сведения, которые вы в документе не прописывали. Такие сведения называются псевдоклассы, то есть «классы, которых как бы нет».

Браузер определяет их, исходя из своих знаний:

1. об HTML-разметке: типе элементов, порядке их расположения, вложенности, языке;
2. о действиях пользователя: кликах по ссылкам, заполнении форм, перемещении элемента мышкой.

### **Стилизация на основе действий пользователя**

Можно стилизовать элементы, реагируя на действия пользователя. Обычно нужно определить специфический внешний вид элемента:

* если пользователь ещё не посещал ссылку — псевдокласс link;
* уже посещал ссылку — visited;
* навёл на элемент указатель мыши — hover;
* нажал на элемент — active.

CSS-селекторы псевдоклассов записывают через двоеточие:

Скопировать кодCSS

*/\* все ссылки, которые пользователь не трогал, будут синие \*/*

a:link {

color: #00f;

}

*/\* ссылки, по которым пользователь переходил, сделаем серыми, бледными \*/*

a:visited {

color: #808080;

}

*/\* ссылка, на которую пользователь навёл курсор, сменит цвет на зеленый \*/*

a:hover {

color: #0f0;

}

*/\* в момент клика ссылка станет красной \*/*

a:active {

color: #f00;

}

Есть мнемоническое правило, которое поможет запомнить эти четыре псевдокласса. Оно составлено из двух английских слов: love и hate: LoVeHAte.

**L** — :link — ссылка, по которой пользователь не переходил;

**V** — :visited — ссылка, по которой пользователь уже переходил;

**H** — :hover — элемент, на который навели указатель мыши;

**A** — :active — элемент, по которому кликают (вид элемента в момент нажатия).

### **Стилизация, исходя из знаний о разметке**

Браузер умеет анализировать разметку. Например, понимает, сколько вложенных элементов у блока, и может посчитать, сколько однотипных блоков на странице.

Вот как это применяют. На сайте интернет-магазина часто располагают товары «плиткой».



Зелёным цветом показаны отступы

Каждой карточке товара мы задаём внешний отступ справа, чтобы между карточками был промежуток. Но у последней карточки в каждом ряду не должно быть отступа — иначе 3 карточки не влезут в контейнер и начнут переноситься на следующую строку.

Чтобы карточки уместились в ряд, нужно обнулить правый отступ у каждой третьей карточки. Для этой задачи есть псевдокласс nth-of-type, то есть «n-й элемент этого типа». Он назначает стили каждому n-му элементу: например, второму, третьему, десятому или пятидесятому.

В скобках после имени псевдокласса указывают число и букву n. 2n означает, что свойство нужно применить к каждому второму элементу, подходящему под селектор. 3n — каждому третьему, 4n — каждому четвёртому. И так далее по аналогии.

Скопировать кодCSS

*/\* правый отступ будет сброшен у каждого третьего элемента с классом card \*/*

.card:nth-of-type(3n) {

margin-right: 0;

}

Чтобы стилизовать самый первый элемент, подходящий под селектор, есть псевдокласс first-of-type, что переводится «первый такого типа». Аналогично есть псевдокласс last-of-type (последний такого типа) для самого последнего элемента.

### **Функциональные псевдоклассы**

Эти псевдоклассы позволяют стилизовать элементы, анализируя разметку или историю действий пользователя. Например, псевдокласс has позволяет стилизовать только элементы, которые содержат внутри себя заданные теги. Имена тегов прописывают в скобках после псевдокласса:

Скопировать кодCSS

*/\* стили применятся только к секциям, внутри которых есть заголовок второго уровня \*/*

section:has(h2) {

*/\* стили \*/*

}

Другой полезный функциональный псевдокласс — not. То, что он принимает в скобках, должно не выполняться:

Скопировать кодCSS

*/\* выберет все ссылки, по которым пользователь ещё не переходил \*/*

a:not(:visited) {

*/\* стили \*/*

}

В официальной документации <https://www.w3.org/TR/selectors-4/#pseudo-class> вы можете узнать подробности обо всех псевдоклассах. Мы разберём лишь основные принципы их применения.

# Псевдоэлементы и селекторы атрибутов

Браузер может не только анализировать ваш контент, но и добавлять что-то от себя до или после элемента.

Этому служат псевдоэлементы before (англ. «до») и after (англ. «после»). Их принято писать через два двоеточия в конце селектора:

Скопировать кодCSS

.selector::before {

*/\* стили для псевдоэлемента before \*/*

}

.selector::after {

*/\* стили для псевдоэлемента after \*/*

}

Браузер вставляет псевдоэлемент ::before перед содержимым указанного элемента, а ::after — после.

Псевдоэлементы ::before и ::after всегда содержат свойство content. Оно определяет, каким содержимым будут наполнены псевдоэлементы.

На примере это колдовство станет понятнее.

HTML-код:

Скопировать кодHTML

<h1 class="title">недаром</h1>

CSS-код:

Скопировать кодCSS

.title::before {

content: 'Зима ';

}

.title::after {

content: ' злится';

}

Этот код выведет на страницу заголовок «Зима недаром злится». При этом слова «Зима» и «злится» будут не частью элемента h1, а вложенными в h1 особыми псевдоэлементами, то есть «элементами, которых как бы нет».

Отображением псевдоэлементов управляют посредством CSS точно так же, как и отображением обычных элементов: задают им фон, шрифт, размер и прочие стили.

# Селекторы по атрибутам

Ещё один интересный и полезный способ работы с CSS — использование селекторов, привязанных к атрибутам. Такие селекторы выбирают элементы, имеющие определённые значения атрибутов. Для этого в квадратных скобках нужно написать имя атрибута, знак равенства и нужное значение в кавычках.

Скопировать кодCSS

img[src='logo.png'] {

border: 1px solid #0000ff;

}

Селектор img[src='logo.png'] выберет только изображение с источником по адресу logo.png и только его обведет голубым контуром. А все прочие картинки оставит без внимания.

Подобные селекторы применяются для решения специфических задач. А такие задачи перед вами обязательно встанут, тогда-то это знание вам и пригодится.