Содержание:

Введение

БЭМ (Блок-Элемент-Модификатор) — *методология web-разработки*, а также набор интерфейсных библиотек, фреймворков и вспомогательных инструментов.

Конечно, все началось с собственных потребностей Яндекса. Вместе с тем, как он рос, росло и количество сотрудников, которые занимаются фронтендом. Постепенно команда увеличилась настолько, что стало очевидно — без единых стандартов работать будет сложно. К тому же, мы находимся в офисах Яндекса в разных городах. Возникла идея создать общую методологию, которая поможет организовать процессы в большой команде, работающей над разными проектами. А главное то, что мы хотели не только упорядочить и ускорить разработку, но и снизить порог входа в проект для нового разработчика.

БЭМ включает в себя:

- Методологические рекомендации по разработке сайтов — простые советы по организации проекта, который нужно сделать быстро, а поддерживать долгие годы.
- Технологии и библиотеки с открытым исходным кодом — готовая реализация рекомендаций БЭМ.
- Инструменты для автоматизации работы с методологией БЭМ.

Возможности БЭМ

- Простая поддержка структуры кода при росте проекта.
- Повторное использование кода.
- Точечные изменения с минимальными затратами: обновление дизайна, добавление функциональных элементов и т. д.

Для чего нужна БЭМ-методология

Какие требования мы сформулировали:

- Разработчик должен понимать свой код (даже вернувшись к нему через год) и код любого программиста в команде БЭМ-проекта.
- Любой блок кода может быть использован повторно: необходимо создать общую базу знаний и не писать каждый раз всё с нуля, а использовать готовые наработки.
- Работая в одной команде, разработчики, менеджеры, дизайнеры и верстальщики должны называть одни и те же вещи одинаково. То есть говорить на одном языке.
- Команды могут обмениваться специалистами для реализации какой-то конкретной функциональности.
- Порог входа при переходе на новый проект должен быть снижен за счет одинаковой структуры организации всех БЭМ-проектов и одинаковых правил именования всех сущностей.

стория развития БЭМ

Как верстали 10 лет назад

Но когда все начиналось, ни о каких компонентных подходах и модульности в веб-разработке речи не шло. Все верстали сайты, складывая CSS в один файл project.css, скрипты, которых было очень мало, — в project.js, а картинки — в папку images.

В 2005 году обычный, с точки зрения интерфейса, проект был набором статических HTML-страниц. Вот такой была типичная структура проекта того времени:

about.html *# Для каждой страницы создавался отдельный HTML-файл*

index.html

…

project.css *# Стили находились в одном файле для всего проекта*

project.js *# Скрипты хранились в одном файле для всего проекта*

images/ *# Картинки складывались в отдельную директорию*

yandex.png

В CSS использовались id, классы и теги.

Пример

#foot div div div div

{

background-position: 54%;

background-image: url(../i/foot-4.png);

}

Типичный CSS того времени в большинстве случаев содержал длинный каскад.

Малейшие изменения требовали длительного рефакторинга. Свёрстанные статические HTML-страницы нарезались в шаблоны. Если HTML изменялся, все правки было необходимо переносить вручную в шаблон.

Вёрстка в больших проектах была неуправляемой.

Основы БЭМ-методологии

Технологии (HTML, CSS, JavaScript), которые мы использовали, изменялись в зависимости от требований проекта, а принципы БЭМ должны были быть универсальны.

Мы сформулировали основные правила, по которым будут жить и развиваться наши проекты, и которые никак не будут зависеть от технологий и инструментов.

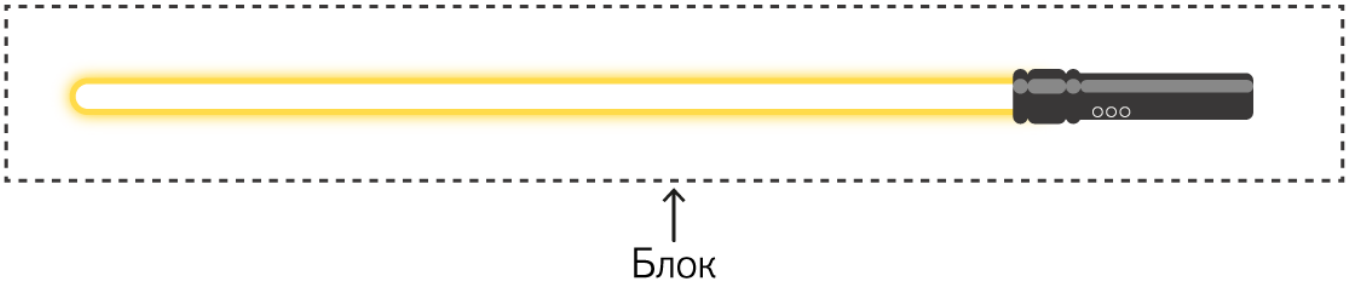
Чтобы ускорить разработку, необходимо было облегчить поддержку HTML и CSS отдельных компонентов страницы, сделать код менее связанным. Для этого мы разбили страницу на части. Так появилось новое понятие — блок. Блок мог состоять из различных элементов, которые не использовались вне самого блока. Состояния и поведение блока и элемента можно было задавать с помощью модификатора.

Это были три ключевых понятия, на которых основывалось большинство правил. Аббревиатура от трех слов Блок, Элемент и Модификатор стала названием методологии — БЭМ.

Блок

Логически и функционально независимый компонент страницы. Блок полностью самодостаточен: у него может быть свое поведение, шаблоны, стили, документация и не только. Блоки могут использоваться в любом месте страницы, повторно, даже в другом проекте.

Одни блоки можно вкладывать в другие, компоновать, использовать для создания более сложных блоков.



Элемент

Часть блока, которая не может использоваться в отрыве от него и имеет смысл только в рамках своего родителя. Элементы могут быть обязательными и опциональными.

Работая с элементами, важно помнить правило: не рекомендуется создавать элементы элементов. Если вложить один элемент в другой, будет невозможно изменить внутреннюю структуру блока: элементы нельзя будет поменять местами, удалить или добавить без корректировки существующего кода.

Модификатор

Свойство блока или элемента, которое меняет их внешний вид, состояние или поведение.
Модификатор имеет имя и может иметь значение. Использование модификаторов опционально. У блока/элемента может быть несколько разных модификаторов одновременно.

Так, например, с помощью модификатора можно изменить не только цвет меча, но и его функциональность (как показано в случае с красным мечом):

Правила именования CSS-селекторов

Все принципы БЭМ формировались и внедрялись постепенно. Мы начали с того, что сформулировали жесткие правила именования CSS-селекторов.

По БЭМ-методологии блоки не уникальны и их всегда можно использовать повторно, поэтому в описании CSS-правил отказались от использования id.

Блок не должен зависеть от окружающих его блоков и сам не должен влиять на соседние блоки, поэтому в CSS отказались от:

- тегов;
- вложенных селекторов;
- глобального сброса правил для всей страницы.

Важной определяющей сущностью в именовании селекторов стал блок:

- Полное имя элемента/модификатора формируется так, чтобы из него можно было определить принадлежность данного элемента/модификатора к конкретному блоку.

- По имени модификатора элемента должно быть возможно определить принадлежность данного модификатора к конкретному элементу конкретного блока.

Правила формирования имени БЭМ-сущности

- Каждая БЭМ-сущность должна иметь свой класс.
- CSS-свойства для блоков, элементов и модификаторов описываются только через классы.
- Для разделения слов в именах используется дефис (-).
- Элемент отделяется от блока двумя подчеркиваниями (__).
  Модификатор — одним (_).
- Имена БЭМ-сущностей записываются с помощью цифр и латинских букв в нижнем регистре.

Мы долго экспериментировали с префиксами в именах, но в итоге отказались от них.

Пример

- Имя блока — header.
- Имя элемента блока — header__search-form — элемент search-form блока header.
- Имя модификатора блока — header_theme_green-forest — модификатор theme в значении green-forest блока header.
- Имя модификатора элемента — header__search-form_disabled — булев модификатор disabledэлемента search-form блока header.

HTML

...

CSS

.header { color: red; }

Существует ряд альтернативных схем именования. Выбор всегда остается за вами.

Но мы рекомендуем придерживаться описанной выше схемы, так как инструменты БЭМ-платформы умеют работать именно с данным вариантом именования.

БЭМ в HTML

Мы хотели упорядочить HTML и в итоге пришли к тому, что больше не пишем HTML руками. Подробнее читайте в разделе про описание инструментов БЭМ.

В HTML каждая БЭМ-сущность определяется своим классом.

...

В простейшем случае блок соответствует DOM-узлу, один к одному. Но DOM-узел и блок — это не всегда одно и то же. На одном DOM-узле может совмещаться

несколько сущностей. Это называется миксом.

С помощью миксов можно:

- объединять поведение и стили нескольких БЭМ-сущностей без дублирования кода;
- создавать семантически новые компоненты интерфейса на основе имеющихся блоков, элементов и модификаторов;
- задавать позицию вложенного блока в родительском, не создавая дополнительных модификаторов. Подробнее, о том, как создавать обёртки в HTML, читайте на форуме.

Пример

В проекте кнопки реализованы блоком button. Необходимо поместить кнопку в форму поиска (search-form) и задать для кнопки отступы. Для этого воспользуемся миксом блока button и элемента buttonблока search-form:

```
<div class="search-form">

<div class="button search-form__button"></div>

</div>
```

Микс позволяет использовать универсальную кнопку, которая ничего не знает об отступах от границ конкретной формы. В данном случае в форме поиска есть элемент search-form__button, который знает, где ему надо находиться, и блок button, который нужно отображать.

Вместо микса можно создать дополнительный модификатор блоку button, но мы не рекомендуем этот способ, так как позиционирование блока button по смыслу не является частью универсального блока, а подходит только для его конкретного места использования.

Организация файловой системы

Нас не устраивала первоначальная структура проекта в файловой системе: в ней было сложно ориентироваться и находить нужные технологии сущностей.

Что мы хотели получить от новой структуры:

- Унифицированную файловую систему любого БЭМ-проекта.
- Универсальную расширяемую структуру репозитория. При добавлении в проект дополнительной технологии заранее известно, где будут находиться новые файлы.
- Быстрый поиск по файловой системе проекта.
- Повторное использование кода.
- Неограниченную возможность переноса кода всего блока из проекта в проект.

Сначала мы попробовали разделить репозиторий проекта по технологиям:

css/

html/

js/

xml/

xsl/

Такой подход не показал кардинальных изменений. Поэтому мы вынесли общую часть кода, подходящую для всех проектов и платформ, в отдельную директорию common. Специфические реализации, необходимые только определённым проектам, складывали отдельно — в директорию service. А примеры — в директорию example.

common/

css/

js/

xml/

xsl/

example/

html/

service/

auto/

css/

xml/

Так мы быстрее находили нужный код для отдельных проектов. Но эта структура всё равно не отвечала всем нашим требованиям.

Чего мы добились:

Ускорения разработки

- Блоки могут использоваться повторно.
- Реализация блоков может быть изменена на новом уровне переопределения, не затрагивая при этом базовую функциональность и стили.
- Блок — независимый компонент страницы и в его директории находится всё, что необходимо для корректного функционирования. Поэтому блоки легко переносить из проекта в проект, достаточно просто скопировать директорию блока.

Ускорение рефакторинга

- Разработчик работает с небольшими блоками кода.
- Технологии реализации одного блока не связаны с технологиями другого.
- Одинаковая структура репозитория позволяет быстро ориентироваться в проекте и находить нужные файлы.

Универсальная расширяемая система

- Появились уровни переопределения.
- Количество технологий не ограничено. Любая новая технология реализации находится в файле конкретного блока. Так, когда мы создавали новую файловую структуру, мы не планировали писать unit-тесты на JavaScript. Но, когда появилась такая необходимость, мы знали, где разместим эти файлы в проекте.

Правила организации файловой системы БЭМ-проекта

Блок — отдельная директория в файловой системе. Имя блока и его директории совпадают.

Реализация блока разделяется на отдельные файлы.

Файлы, относящиеся к блоку, всегда находятся в его директории.

Опциональные элементы и модификаторы выносятся в отдельные файлы.

Проект разделяется на уровни переопределения.

Пример

blocks/

input/ # Директория блока input

_theme/ # Директория опционального модификатора theme

input_theme_forest.css # Реализация модификатора theme в значении forest в технологии CSS

__clear/ # Директория опционального элемента clear

input__clear.css # Реализация элемента clear в технологии CSS

input__clear.png # Реализация элемента clear в технологии PNG

input.css # Блок input в технологии CSS

input.js # Блок input в технологии JavaScript

button/ # Директория блока button

button.css

button.js

button.png

https://yandex.ru/dev/bem/

https://ru.wikipedia.org/wiki/%D0%91%D0%AD%D0%9C

https://habr.com/ru/company/yandex/blog/276035/