Яндекс

История создания БЭМ
Кратко, сбивчиво и неполно

Виталий Харисов

@harisov #b_ #yasubbotnik

Я.Субботник, Минск, 2 июня 2012 года

Сегодня я вам расскажу историю создания БЭМ, откуда взялись сущности БЭМ и почему они сейчас именно такие.

В конце покажу, что БЭМ гибок и может применяться в том объёме, в каком это возможно в вашем проекте.

Статья

clubs.ya.ru/bem/replies.xml?item_no=1398

Всё, что сейчас расскажу подробно описано в статье, опубликованной в нашем клубе на Яру.

Доисторические времена

Итак, начнём с самого начала.

Яндекс.Адреса, 2005

Когда я начал работать над Яндекс.Адресами в далёком 2005 году, вёрстку я делал так.

Яндекс.Адреса, 2005
Файловая система

```
about.html
index.html
…
adresa.css
adresa.js
i/
    yandex.png
```


На файловой системе:

- Делались статические html-странички
- Стили для них складывались в один файл на весь проект — adresa.css
- Скрипты — в adresa.js. Скриптов было очень мало
- Картинки — в директорию в отдельную директорию

Яндекс.Адреса, 2005

Структура CSS-файла

```
/* Content container (begin) */
    #body
    {
        font: 0.8em Arial, sans-serif;

        margin: 0.5em 1.95% 0.5em 2%;
    }
/* Content container (end) */
```


Все стили были в одном файле и для отделения стилей разных частей страницы использовались комментарии с указанием начала и конца.

В вёрстке использовались как id, так и классы.

Яндекс.Музыка, 2006

Адреса были маленьким проектом с несколькими страницами и такой подход к вёрстке был удобен.

При вёрстке первой версии Яндекс.Музыки в начале 2006 стало понятно, что для проекта с большим количеством разных страниц этот подход работает плохо.

Тяжело подбирать названия классам, код всего проекта сложно держать в голове и писать так, чтобы изменения на одной странице ничего не ломали на другой.

Яндекс.Музыка, 2006
Типичный CSS код

```
/* Albums (begin) */
   .result .albums .info {…}

   .result .albums .info i {…}

   .result .albums .album .listen {…}

/* Albums (end) */
```


Типичный код того времени.

Используется длинный каскад.

Яндекс.Музыка, 2006
Типичный CSS код

```
/* Картинки на фоне (begin) */
    .#foot div {…}

    .#foot div div {…}
    …
    .#foot div div div div div div {…}

/* Картинки на фоне (end) */
```


Или вот ещё пример.

Используются id и тэги.

Яру, 2006

Одновременно с Музыкой началась вёрстка Яру.

Это проект с десятками страниц и такой подход не работает — вёрстка становится неуправляемой.

Появление блоков

И мы ввели понятие блока для частей страниц, которые имеют самостоятельное значение в дизайне.

Яру, 2006
Появление блоков

b- block
независимый блок, может использоваться
в любом месте страницы

c- control
контрол (независимый блок) с которым
ассоциирован javascript-объект

g- global
глобальное определение, используется по
необходимости, количество сведено к минимуму

Классы блоков получили префиксы (b–, c–, g–), чтобы отличаться от классов внутри.

Появление элементов

Внутренние классы мы стали называть элементами.

Яру, 2006

Появление элементов

```
/* Head (begin) */
    .b-head {…}

    /* Search (begin) */
        .b-head .search {…}
        .b-head .search div div,
        .b-head .search div div i {…}
    /* Search (end) */
```


Элементы, у которых много кода, выделялись комментариями.

Структура вёрстки проекта

Изменилась структура вёрстки проекта.

Яру, 2006

Структура вёрстки проекта

```
index.html
css/
    yaru.css
    yaru-ie.css
js/
    yaru.js
i/
    yandex.png
```

Ввели директории для технологий css и js.

Яру, 2006
yaru.css

```
/* Common definitions (begin) */
    body {…}
    * html body {…}
/* Common definitions (end) */

/* Head (begin) */
   .b-head {…}
/* Head (end) */
```


Валидные хаки для MSIE писали в основном CSS-файле.

Яру, 2006
yaru-ie.css

```
.b-artist .i i {
  top: expression(7 + (90 -
       this.parentNode
       .getElementsByTagName('img')[0]
       .height)/2);

  filter:progid:DXImageTransform.
    Microsoft.AlphaImageLoader(src='…’)
```

Невалидные — в файле -ie.css.

Зачатки общепортального фреймворка

При верстке нескольких проектов с похожим дизайном есть общие блоки.

Очень быстро понимаешь, что copy/paste этих блоков с проекта на проект не выход и надо делать common.

Так появилось общее хранилище для стилей шапки, подвала, статического текста на специальном сервере.

Зачатки общепортального фреймворка

Структура CSS-файла

```
@import url(http://…/css/head/common.css);
@import url(http://…/css/foot/common.css);

/* Header (begin) */
  /* Service (begin) */
    .b-head .service h1 {…}
    .b-head .service h1,
    .b-head .service h1 a {…}
```


Стили с него подключались в основной проектный файл с помощью импортов.

Чтобы с этим можно было уйти в продакшн, появился процесс сборки, во время которого вместо @import подставляется содержимое этих файлов.

Вёрстка независимыми блоками

К осени 2007 года правила вёрстки уже устоялись, и о них захотелось рассказать вне Яндекса.

Доклад на ClientSide‘2007 «Вёрстка независимыми блоками»

На ClientSide'2007 я сделал доклад **«Вёрстка независимыми блоками»**

Доклад на ClientSide‘2007
Статья

vitaly.harisov.name/article/independent-blocks.html

Содержимое доклада можно прочитать в статье, а я сейчас расскажу самое важное.

Доклад на ClientSide‘2007
Блок

> *Блоком будем называть фрагмент страницы, который описывается своей разметкой и стилями.*

В докладе вводится понятие «Блок».

Доклад на ClientSide‘2007
Правила независимости блока

> *1) для описания элемента используется class, но не id*
>
> *2) каждый блок имеет префикс*
>
> *3) в таблице стилей нет классов вне блоков*

clubs.ya.ru/bem/replies.xml?item_no=4

Формулируются правила независимости блока.

Отказ от id даёт возможность использовать один и тот же блок несколько раз на странице.

А также позволяет использовать на одном DOM-узле несколько классов, что в дальнейшем нам пригодилось.

Доклад на ClientSide‘2007
Простые и составные блоки

> *В простые блоки нельзя вкладывать другие блоки, в составные — можно.*

clubs.ya.ru/bem/replies.xml?item_no=42

Блоки делятся на простые и составные.

В простые блоки нельзя вкладывать другие блоки, в составные — можно.

Это было очень наивное деление, мы неоднократно сталкивались с тем, что даже в самые простые блоки вкладывались другие и приходилось переделывать вёрстку.

Доклад на ClientSide‘2007
Полная независимость блоков

«

1) никогда не опираться на элементы, только на классы:

.b-user b => .b-user .first-letter

2) всем классам внутри блока давать имена начинающиеся с имени этого блока:

.b-user .first-letter => .b-user__first-letter

Так же были сформулированы правила полной независимости блоков:

1) Нельзя использовать теги в селекторах
2) Нужно задавать каждому DOM-узлу свой класс

Доклад на ClientSide‘2007
Абсолютно-независимые блоки

clubs.ya.ru/bem/replies.xml?item_no=43

Позже мы назвали это АНБ (абсолютно-независимые блоки): clubs.ya.ru/bem/replies.xml?item_no=43

Но класс у каждого DOM-узла существенно увеличивает HTML-код.

На тот момент мы считали, что это дорого и применяли такой подход в исключительных случаях.

Доклад на ClientSide‘2007
Префиксы

« *Имена блоков начинаются с префиксов*

b- обычные блоки

h- обёртки

l- раскладки

g- глобальные стили

Тогда же, сформулировали правила для префиксов.

Доклад на ClientSide‘2007
Модификация блоков

« *1. Модификация блока от контекста*

2. Модификация постфиксом не зависимо от контекста, добавляя постфикс к имени блока:
b-block b-block-postfix

Ввели понятие модификации:

1. Блок может изменить свой внешний вид в зависимости от того, где он находится.
   Это модификация от контекста.

2. Можно добавить блоку второй класс, это модификация постфиксом, она не зависит от контекста.

Общепортальный фреймворк

Весной 2008 года была поставлена задача сделать брендбук, описывающий наш портальный стиль.

Я начал с того, что умел лучше всего — делать HTML/CSS код.

Если делать просто описание — оно устареет, не успев дописаться до конца.

Лего 1.0, 2008

Проект получил название Лего.

Лего 1.0, 2008

Структура репозитория

```
css/
html/
js/
xml/
xsl/
```


Первичное разделение на верхнем уровне репозитория идёт по технологиям.

В каждой технологии своя структура.

Лего 1.0, 2008

Структура репозитория: CSS

```
css/
    block/
        b-dropdown/
            b-dropdown.css
    service/
        auto/
            block/
                b-head-logo-auto.css
            head.css
    util/
        b-hmenu/
            b-hmenu.css
```


CSS раскладывается в три разные директории.

Лего 1.0, 2008

Структура репозитория: CSS

```
css/
    block/
        b-dropdown/
            b-dropdown.css
    service/
        auto/
            block/
                b-head-logo-auto.css
            head.css
    util/
        b-hmenu/
            b-hmenu.css
```


block — общепортальные блоки

util — блоки, которые имеют смысл вне Яндекса, их можно выложить в Open Source

service — стили для конкретных сервисов Яндекса, подключив их на сервис можно отобразить шапку, подвал

Лего 1.0, 2008

Структура репозитория: HTML

```
html/
    block/
        b-dropdown.html
    service/
        auto/
            l-head.html
    util/
        b-hmenu.html
```


Структура директории html аналогична css.

Лего 1.0, 2008

Структура репозитория: JS

```
js/
    check-is-frame.js
    check-session.js
    clean-on-focus.js
    dropdown.js
    event.add.js
    event.del.js
```


JS находится в зачаточном состоянии.

Лего 1.0, 2008

Структура репозитория: XML

```
xml/
    block/
        b-head-tabs-communication.xml
        common-services.ru.xml
        head-messages.ru.xml
    service/
        auto/
            head.xml
```


У каждого сервиса есть XML-файл, использующийся для построения шапки.

Лего 1.0, 2008

Структура репозитория: XSL

```
xsl/
    block/
        b-dropdown.xsl
        b-head-line.xsl
        i-common.xsl
        i-locale.xsl
        l-foot.xsl
        l-head.xsl
```


XSL блоков лежит в одной директории.

Каждому блоку соответствует один файл.

Лего 1.0, 2008

Что ещё можно выделить

Подключение через svn:externals

Директории из Лего подключались на проект

Статическая линковка

Подключение Лего на проекты делалось через svn:externals.

При сборке для продакшена, код библиотеки полностью включается в проект, что можно сравнить со статической линковкой.

Такой подход позволяет выпускать версии сервисов с разными версиями Лего и переходить на новую версию, когда это удобно команде проекта.

Мы используем его до сих пор.

Лего 1.0, 2008

Что ещё можно выделить

```
@import url(../../block/l-head/
                l-head.css);
@import url(../../block/b-head-logo/
                b-head-logo.css);
@import url(../../block/b-head-logo/
                b-head-logo_name.css);
@import url(block/b-head-logo-auto.css);
```


Файлы, подключавшиеся на страницах, состояли из @import'ов реализации блоков.

Эти import’ы писались вручную.

Лего 1.0, 2008

Что ещё можно выделить

Именование файлов ещё не устоялось

Именование файлов ещё не устоялось, мы пробуем разные варианты.

Лего 1.0, 2008

Что ещё можно выделить

Префиксы:

b- визуальные блоки

l- раскладка блоков

i- вспомогательные блоки

Используются три типа префиксов:

b- для визуальных блоков
l- для блоков задающих раскладку
i- для вспомогательных блоков

Лего 1.2, 2008

Чуть позже в рамках версии 1.2 был сделан рефакторинг…

Лего 1.2, 2008

Структура репозитория

```
common/
    css/
    js/
    xml/
    xsl/
example/
    html/
service/
    auto/
        css/
        xml/
```


…и структура Лего изменилась.

Лего 1.2, 2008

Структура репозитория: CSS

```
common/
    css/
        b-dropdown/
            arr/
                b-dropdown.arr.css
                b-dropdown.arr.ie.css
            b-dropdown.css
            b-dropdown.ie.css
```


Убрано разделение на util и block, весь общий CSS лежит в common/css.

От идеи выноса кода в Open Source на тот момент отказались.

Вернулись к ней только через два года.

Лего 1.2, 2008

Структура репозитория: CSS

```
common/
    css/
        b-dropdown/
            arr/
                b-dropdown.arr.css
                b-dropdown.arr.ie.css
            b-dropdown.css
            b-dropdown.ie.css
```


Файлы для MSIE переименованы.

Расширения у файлов теперь могут быть из нескольких слов.

Лего 1.2, 2008

Структура репозитория: CSS

```
common/
    css/
        b-dropdown/
            arr/
                b-dropdown.arr.css
                b-dropdown.arr.ie.css
            b-dropdown.css
            b-dropdown.ie.css
```


Опциональный CSS блоков вынесен в директории.

В основном файле блока стало меньше кода.

Лего 1.2, 2008

Структура репозитория: CSS

```
.b-head-tabs-grey

.b-head-tabs_grey

 b-head-tabs_grey.css
```


Для модификации постфиксом вместо минуса начали использовать подчёркивание.

Это позволило визуально отделить имя блока от модификатора, что потом пригодилось при реализации инструментов, помогающих работать с кодом.

БЭМ

В марте 2009 выходит версия Лего 2.0.

Этим событием заканчивается «Вёрстка независимыми блоками» и начинается «БЭМ».

Лего 2.0, 2009

Что же принципиально изменилось с выходом версии 2.0?

Лего 2.0, 2009
Блоки первичны

Блоки первичны

Технологии вторичны

clubs.ya.ru/bem/replies.xml?item_no=237

Самое главное изменение — мы вывели вперёд блоки, а не технологии.

Блоки стали первичны, а технологии их реализации — вторичны.

Реализация блока делалась в отдельной директории, а технологии — это файлы внутри неё.

В том числе появилась документация к блоку — это файл .wiki внутри блока.

Терминология

Какими терминами мы тогда оперировали?

Лего 2.0, 2009

Терминология: Блоки

```
XML:      <lego:l-head>
          <lego:b-head-logo>

HTML:     <table class="l-head">
          <div class="b-head-logo">

CSS:      .l-head
          .b-head-logo
```


Независимый блок,

который может быть использован в любом месте страницы.

В XML он представлен тегом в неймспейсе lego, в HTML класс у блока, такой же, как имя этого тега.

Лего 2.0, 2009

Терминология: Блоки

```
common/
    block/
        b-head-logo/
            b-head-logo.css
            b-head-logo.xsl
            b-head-logo.js
            b-head-logo.wiki
```


Все файлы (css, js, html, xsl), относящиеся к блоку, кладутся в его директорию.

Лего 2.0, 2009
Терминология: Элементы

```
XML:    <lego:b-head-logo>
            <lego:name/>
        </lego:b-head-logo>

HTML:   <div class="b-head-logo">
            <span class="name">Авто</span>
        </div>

CSS:    .b-head-logo .name
```

Элементы в XML пишутся в неймспейсе lego без префикса.

Класс в HTML — тоже без префикса.

Лего 2.0, 2009

Терминология: Элементы

```
common/
    block/
        b-head-logo/
            name/
                b-head-logo.name.css
                b-head-logo.name.png
                b-head-logo.name.wiki
```

Файлы для вложенного элемента кладутся в отдельную директорию.

Лего 2.0, 2009
Терминология: Модификаторы

```
XML:    <lego:b-head-tabs lego:theme="grey">

HTML:   <div class="b-head-tabs
                    b-head-tabs_grey">

CSS:    .b-head-tabs_grey
```


Модификаторы в XML представлены атрибутами в неймспейсе lego.

В HTML используется дополнительный класс.

Лего 2.0, 2009

Терминология: Модификаторы

```
common/
    block/
        b-head-logo/
            _theme/
                b-head-logo_gray.css
                b-head-logo_gray.png
                b-head-logo_gray.wiki
```

Файлы для модификатора кладутся в отдельную директорию, с подчёркиванием в начале имени.

Модификатор имеет тип и значение.

Декларация используемых блоков

Все лего-компоненты проекта описываются…

Лего 2.0, 2009
Декларация

```
<lego:page>
    <lego:l-head>
        <lego:b-head-logo>
            <lego:name/>
        </lego:b-head-logo>

        <lego:b-head-tabs
            type="search-and-content"/>
```


...в XML-файле.

Из него генерируются...

Лего 2.0, 2009
Генерируется CSS

```
@import url(../../common/block/
            b-head-logo/b-head-logo.css);
@import url(../../common/block/
            b-head-logo/name/
            b-head-logo.name.css);
@import url(block/
            b-head-logo/b-head-logo.css);
```


…CSS-файлы.

Сначала идёт общий код, а потом добавляются стили, чтобы привести лего-блоки к дизайну проекта.

Имена файлов элементов пишутся через точку: b-head-logo.name.css

Лего 2.0, 2009
Генерируется JS

```
include("../../common/block/
         i-locale/i-locale.js");
include("../../common/block/
         b-dropdown/b-dropdown.js");
include("../../common/block/b-search/
         sample/b-search.sample.js");
```


Из XML-декларации генерируются и JS-файлы.

Лего 2.0, 2009
Генерируется XSL

```
<xsl:import href="../../common/block/
      l-head/l-head.xsl"/>
<xsl:import href="../../common/block/
      b-head-logo/b-head-logo.xsl"/>
<xsl:import href="../../common/block/
      b-head-logo/name/
      b-head-logo.name.xsl"/>
```


А также XSL-файлы.

Мы перестали писать эти файлы руками, началась генерация кода.

Проблема скорости селекторов, 2009

При реализации новой версии Яндекс.Почты была задача сделать её такой быстрой как программа на компьютере.

Но возникли проблемы.

При исследовании обнаружили, что тормозят селекторы CSS.

Скорость селекторов, 2009

clubs.ya.ru/bem/replies.xml?item_no=338

Результаты исследования подробно описаны в статье.

Скорость селекторов, 2009
Решение

Абсолютно-Независимые Блоки (АНБ)

clubs.ya.ru/bem/replies.xml?item_no=43

Мы поняли, что решение этой проблемы у нас есть давно — это абсолютно-независимые блоки.

Скорость селекторов, 2009
Решение

```
<div class="b-head-logo">
    <span class="b-head-logo__name">
        Авто
    </span>
</div>
```


В классы элементов вносится имя блока, селекторы получаются простыми и быстрыми.

Стабилизация нотации

Постепенно мы пришли к тому, что нотация в коде и структура на файловой системе устоялась и с тех пор уже не меняется.

Стабилизация нотации
Разделитель элемента

Было

```
b-block.elem.css
```

Стало

```
b-block__elem.css
```


В именах файлов разделитель . был заменён на __.

Теперь они совпадают с CSS-селекторами.

Стабилизация нотации
Модификаторы элементов

```
.b-block__elem_theme_green

по аналогии с

.b-block_theme_green
```


Были реализованы модификаторы у элементов по аналогии с модификаторами блоков.

Стабилизация нотации
Тип модификатора

Было

```
.b-menu__item_current
```

Стало

```
.b-menu__item_state_current
```


В имя файла модификатора и в его класс внесён тип модификатора.

Причина этого изменения — работа с модификаторами из JS.

Open Source (2010)

github.com/bem

В 2010 году мы создали организацию bem на github’е чтобы вести разработку в open source.

Библиотека блоков bem-bl

Мы начали выносить блоки из Лего в bem-bl, проводя одновременно с этим рефакторинг.

Библиотека блоков bem-bl

github.com/bem/bem-bl

clubs.ya.ru/bem/posts.xml?tag=8486525

Вместе с выносом блоков публиковали информацию про них в клубе.

Работы по выносу блоков в Open Source пока не закончены.

Инструменты

github.com/bem/bem-tools

Начали реализацию инструментов bem-tools, которые помогают работать с файлами по БЭМ-методу.

Инструменты реализуются на JavaScript под Node.js.

Инструменты
Уровни переопределения

```
bem-bl/
    b-logo/
lego/
    b-logo/
auto/
    blocks/
        b-logo/
```


Директории с реализацией блоков стали называть «уровнем переопределения».

Например, на проекте может быть

1) публичная библиотека блоков с github
2) внутренняя библиотека lego
3) и блоки самого проекта

Инструменты
Схемы именования

```
.bem/
    level.js
```


На уровне переопределения можно задать другую схему именования папок/файлов, отличную от нашей.

Например, вы можете задать другие разделители между именем блока и элемента, или не раскладывать всё по директориям, а сделать плоскую структуру из файлов.

BEMHTML
JavaScript шаблонизатор

После экспериментов с разными шаблонизаторами, был разработан шаблонизатор BEMHTML.

BEMHTML
JavaScript шаблонизатор

— Шаблоны в БЭМ-терминах

— Уровни переопределения

— На сервере и в браузере

Этот шаблонизатор

1) позволяет писать шаблоны в БЭМ-терминах
2) доопределять их на уровнях переопределения
3) исполнять эти шаблоны, как на сервере, так и в браузере, поскольку шаблоны компилируются в простой и быстрый JavaScript

BEMHTML
JavaScript шаблонизатор

clubs.ya.ru/bem/replies.xml?item_no=898

clubs.ya.ru/bem/replies.xml?item_no=899

clubs.ya.ru/bem/replies.xml?item_no=1153

clubs.ya.ru/bem/replies.xml?item_no=1172

clubs.ya.ru/bem/replies.xml?item_no=1391

Подробности в клубе на Яру.

Варианты использования БЭМ

Как вы можете видеть, БЭМ не появился сразу. У нас был долгий период проб и подбора наиболее подходящего нам варианта.

Но хочу обратить внимание, что всё это время это был БЭМ.

То, что мы используем сейчас — не единственно верное решение.

Наоборот, вы можете использовать БЭМ на своих проектах в том объёме, в котором он принесёт наибольшую пользу.

БЭМ методология очень гибкая и позволяет вам настраивать её под свои процессы, под свои текущие технологии.

Давайте посмотрим на примерах.

Блоки в одном файле

У вас есть проект, в котором вы хотите применить БЭМ для вёрстки и ни для чего более.

Хорошо, мы тоже с этого начинали.

Выбирайте подходящую вам схему…

Блоки в одном файле

Префиксы и классы без них

```
.b-block
.b-block .elem
.b-block_size_l
.b-block .elem_size_l
```


С префиксами и каскадом.

Блоки в одном файле
АНБ с префиксами

```
.b-block
.b-block__elem
.b-block_size_l
.b-block__elem_size_l
```


Или задайте класс каждому DOM-узлу и используйте абсолютно-независимые блоки.

Блоки в одном файле
АНБ без префиксов

```
.block
.block__elem
.block_size_l
.block__elem_size_l
```


Или можно убрать префиксы.

Блоки в одном файле
И верстаете

```
myfacebook/
    myfacebook.css
    myfacebook.js
    myfacebook.html
```


И начинайте делать вёрстку на проекте по БЭМ.

Используйте самую простую схему на файловой системе, когда реализация блоков лежит в одном файле.

При использовании этого варианта всё делается руками, без bem-tools.

Блоки в директории

Когда вы поймёте, что ваш проект вырос, можно начать раскладывать реализацию блоков по файлам и использовать сборку.

Блоки в директории

```
blocks/
    b-myblock.css
    b-myblock.js
    b-yourblock.css
    b-yourblock.js
```


В этом случае можно начинать использовать bem-tools для сборки проекта.

Не обязательное в файлах

Если в ваших блоках есть элементы/модификаторы, которые используются не на всех страницах…

Не обязательное в файлах

```
blocks/
   b-myblock/
      b-myblock_mod_val1.css
      b-myblock__opt-elem.css
      b-myblock__opt-elem_mod_val1.css
      b-myblock.css
```


...можно вынести их реализацию в отдельные файлы, чтобы подключать только то, что нужно.

Модификаторы в директориях

Если модификаторов много...

Модификаторы в директориях

```
blocks/
  b-myblock/
    _mod/
      b-myblock_mod_val1.css
    b-myblock__opt-elem.css
    b-myblock__opt-elem_mod_val1.css
    b-myblock.css
```


…для каждого можно сделать отдельную директорию. Это разгрузит корневую директорию блока.

Опциональные элементы в директориях

Предпоследний по сложности вариант, когда...

Опциональные элементы в директориях

```
blocks/
  b-myblock/
    _mod/
      b-myblock_mod_val1.css
    __opt-elem/
      b-myblock__opt-elem.css
    b-myblock.css
```


…в директории выносятся ещё и необязательные элементы, а код обязательных элементов лежит в основных файлах блока.

Мы используем сейчас именно этот вариант при разработке Лего и bem-bl.

ВСЕ элементы/модификаторы в директориях

Самый сложный в разработке, но самый понятный вариант по конечной структуре, когда ВСЕ элементы блока и ВСЕ модификаторы имеют свои директории.

Все элементы/модификаторы в директориях

```
blocks/
  b-myblock/
    _mod/
      b-myblock_mod_val1.css
    __elem/
      b-myblock__elem.css
    b-myblock.css
```


Этот вариант очень наглядный, при взгляде на файловую систему можно увидеть всю структуру блока.

Мы сейчас спорим, надо нам переходить на него в Лего или нет.

Подведём итог

Итак, давайте подведём итог.

Подведём итог

методология организации работы

нет единственно правильного варианта

набор правил

команда встраивает в свой процесс

БЭМ — это методология организации работы над проектом, которая позволяет команде работать с единым кодом и говорить на одном языке.

При этом нет **единственно** правильного варианта и мы не стремимся его получить.

Наоборот, мы рассматриваем БЭМ как набор правил.

Каждая конкретная команда встраивает его в свой процесс разработки и использует так, как им удобно.

Статья

clubs.ya.ru/bem/replies.xml?item_no=1398

Ещё раз обращаю ваше внимание, что всё, что я сейчас рассказывал более подробно описано в статье, опубликованной в нашем клубе на Яру.

Виталий Харисов

vitaly@yandex-team.ru

@harisov

#b_ #yasubbotnik

Спасибо за внимание.

Я готов ответить на вопросы сейчас и после доклада.