
B_E_M

Hi, there!
I’m Luciano Battagliero
@battaglr

Block,
Element,
Modifier

Yandex

~2009

A little bit of context

OOCSS ~2010

SMACSS ~2011

What is it?

It’s a **methodology**

Heavily complemented by a set of **tools**, **libraries** and a complete **technology stack**

At its **core** BEM is an **unified semantic** for **different implementations**

That being said, this talk will be focused mainly on CSS

What does BEM solve?

Chaos

/ˈkāˌäs/

noun

1. Complete disorder and confusion

There are **two types** of **problems** we face in **CSS**

Layout or **cosmetic** problems

Architectural problems

BEM attempts to **help solving architecture** related **problems**

Entity

A generic **term** to **refer** to **blocks**, **elements** or **modifiers**

Block

An **independent** and **self-sufficient component** of an interface

Provides structure, behaviour and appearance **encapsulation**

```
<dialog class="modal"> ... </dialog>
```

```
.modal { … }
```

Blocks must be **context independent**, thus they should **not** have direct **influence** over **other blocks**

```
<dialog class="modal">
  <button class="btn"> ... </button>
</dialog>
```

```
// Don’t do this
.modal .btn { … }
```

Blocks **can contain** other **blocks**

```
<header class="header">
  <nav class="nav"> ... </nav>
</header>
```

```
<header class="header">
  <nav class="nav"> ... </nav>
</header>
```

Multiple instances of a block could be used across the interface

```
<button class="btn"> ... </button>

...

<dialog class="modal modal--message">
  <button class="modal__confirm btn">
    ...
  </button>
</dialog>
```

```
<button class="btn"> ... </button>

...

<dialog class="modal modal--message">
  <button class="modal__confirm btn">
    ...
  </button>
</dialog>
```

Element

An **internal part** of a block that can **not** be **used outside** of it

Elements **can contain** other **elements** or **blocks**

```
<ul class="paginator">
  <li class="paginator__page">
    <a class="paginator__link"> ... </a>
  </li>
</ul>
```

```
<ul class="paginator">
  <li class="paginator__page">
    <a class="paginator__link"> ... </a>
  </li>
</ul>
```

Elements should **not** attempt to be a **representation** of the **DOM structure**

```
<ul class="paginator">
  <li class="paginator__page">
    <!-- Don’t do this -->
    <a class="paginator__page__link"> … </a>
  </li>
</ul>
```

```
<ul class="paginator">
  <li class="paginator__page">
    <!-- Don’t do this -->
    <a class="paginator__page__link"> … </a>
  </li>
</ul>
```

A block may not contain any element

Modifier

A **variation** on the **appearance** or **behavior** of a **block** or an **element**

```
<a class="btn btn--large"> ... </a>
```

```
<a class="btn btn--large"> ... </a>
```

```
<a class="btn btn--is-disabled"> ... </a>
```

```
<a class="btn btn--is-disabled"> ... </a>
```

Multiple modifiers can be **used simultaneously** on the same block or element

```
<a class="btn btn--large btn--is-disabled"> … </a>
```

```
<a class="btn btn--large btn--is-disabled"> ... </a>
```

Naming conventions

The **main purpose** of a **class name** is to be used as a **hook** for adding style or behaviour

A class name should **communicate information** which helps to **understand** its **purpose**

Remember that a class name can not be “unsemantic”[*]

The “official” syntax

```
// Basic syntax
.block__element_modifier
```

```
// Basic syntax
.block__element_modifier
```

```
// Basic syntax
.block__element_modifier
```

```
// Basic syntax
.block__element_modifier
```

```
// Key-value modifier
.block-or-element_modifier_value
```

```
// Key-value modifier
.block-or-element_modifier_value
```

```
// Entities with compound names
.block-name__element-name_modifier-name
```

```
// Entities with compound names
.block-name__element-name_modifier-name
```

```
////
// All possible combinations
//

.block
.block_modifier
.block__element
.block__element_modifier
```

```
////
// Don’t do any of these
//

.element
.block_modifier__element
.block__element__element
```

The “popular” syntax

```
// Basic syntax
.block__element--modifier
```

```
// Basic syntax
.block__element--modifier
```

```
// Basic syntax
.block__element--modifier
```

```
// Basic syntax
.block__element--modifier
```

```
// Entities with compound names
.block-name__element-name--modifier-name
```

```
// Entities with compound names
.block-name__element-name--modifier-name
```

```
////
// All possible combinations
//

.block
.block--modifier
.block__element
.block__element--modifier
```

```
////
// Don’t do any of these
//

.element
.block--modifier__element
.block__element__element
```

The “CamelCase” syntax

```
// Basic syntax
.Block-element--modifier
```

```
// Basic syntax
.Block-element--modifier
```

```
// Basic syntax
.Block-element--modifier
```

```
// Basic syntax
.Block-element--modifier
```

```
// Entities with compound names
.BlockName-elementName--modifierName
```

```
// Entities with compound names
.BlockName-elementName--modifierName
```

```
////
// All possible combinations
//

.Block
.Block--modifier
.Block-element
.Block-element--modifier
```

```
////
// Don’t do any of these
//

.element
.Block--modifier-element
.Block-element-element
```

Syntax comparison

```
// “Official”
block-name__element-name_modifier-name

// “Popular”
block-name__element-name--modifier-name

// “CamelCase”
BlockName-elementName--modifierName
```

```
// “Official”
block-name__element-name_modifier-name

// “Popular”
block-name__element-name--modifier-name

// “CamelCase”
BlockName-elementName--modifierName
```

```
// “Official”
block-name__element-name_modifier-name

// “Popular”
block-name__element-name--modifier-name

// “CamelCase”
BlockName-elementName--modifierName
```

```
// “Official”
block-name__element-name_modifier-name

// “Popular”
block-name__element-name--modifier-name

// “CamelCase”
BlockName-elementName--modifierName
```

It has been scientifically proven
that BEM class names are ugly

Mix

A **combination** of different **entities** on a **single DOM node**

```
<input class="search__input input" />
```

```
<input class="search__input input" />
```

Implementation guidelines

Only **define entities** using **class selectors**

```
.icon { … }
```

Use **descendant selectors** only **when needed**

```
// Don’t do this
.menu .menu__item { … }
```

```
// Do this
.menu--horizontal .menu__item { … }
```

Do **not** use **type** or **id selectors**

```
// Don’t do this
button { … }
```

```
// Do this
.button { … }
```

```
// Don’t do this
#breadcrumb { … }
```

```
// Do this
.breadcrumb { … }
```

Do **not** declare **styles outside** of **blocks**

```
// Don’t do this
ul {
  list-style: none;
}
```

```
// Do this
.menu {
  list-style: none;
}
```

This usually means no global styles and no global “resets”

That’s just a part of it!

There’s a lot **more** about **BEM**. Go to bem.info and find out!

Questions?

Thanks!