

[Pages](#) / [Home](#)

Using BEM at the META+LAB

Created by Grano, Andrew Gordon, last modified on Jul 19, 2018

What is BEM?

BEM (Block, Element, Modifier), is a simple naming convention that makes front-end code easier to understand, easier to work with, and easier to scale.

Using proper naming convention:

- Ensures everyone who participates in the development of websites at the META+LAB can easily read and understand the front-end codebase.
- Will make it easier to modify parts of a website when design changes need to be made in the future
- Will ease the process of onboarding new team members to a project.

Working with BEM

Use Blocks, Elements, and modifiers to build reusable components. The naming convention is as follows:

.block

.block__element

.block--modifier

.block__element--modifier

Block

A **block** is a functionally independent page component that can be reused.

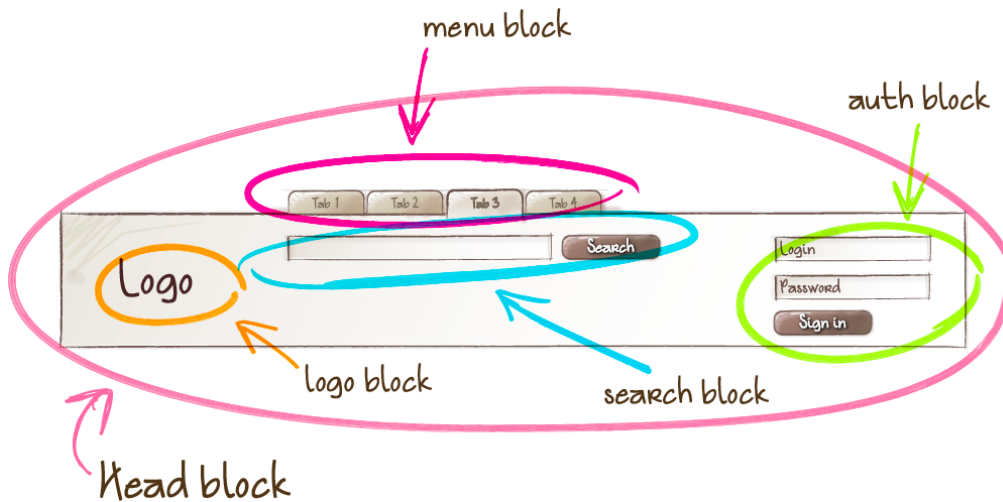
Rules for usage:

- The block name describes its purpose ("What is it?" — menu or button), not its state ("What does it look like?" — red or big).
- Blocks can be nested in each other. You can have any number of nesting levels.
- Block shouldn't influence its environment, meaning you shouldn't set the external geometry (margin) or positioning for the block.

Example:

```
<!-- `header` block -->
<header class="header">
  <!-- Nested `logo` block -->
  <div class="logo"></div>

  <!-- Nested `search-form` block -->
  <form class="search-form"></form>
</header>
```



(Pictured above: a block with many other blocks nested inside)

Element

An **element** is a composite part of a block that cannot be used separately.

Rules for usage:

- The element name describes its purpose ("What is it?" — menu or button), not its state ("What does it look like?" — red or big).
- The structure of an element's full name is block-name__element-name. The element name is separated from the block name with a double underscore (__).

```
<!-- `search-form` block -->
<form class="search-form">
  <!-- `input` element in the `search-form` block -->
  <input class="search-form__input">

  <!-- `button` element in the `search-form` block -->
  <button class="search-form__button">Search</button>
</form>
```

- Elements can be nested inside each other. You can have any number of nesting levels.
 - However, an element is always part of a block, not another element. This means that element names cannot define a hierarchy such as block__elem1__elem2.

```
<div class="block">
  <div class="block__elem1">
    <div class="block__elem2">
      <div class="block__elem3"></div>
    </div>
  </div>
</div>
```

- This block structure is always represented as a flat list of elements in the BEM methodology:

```
.block {}
.block__elem1 {}
.block__elem2 {}
.block__elem3 {}
```

- This allows you to change a block's DOM structure without making changes in the code for each separate element.
- Not all blocks have elements. Elements are optional parts of a block.

Modifier

A **modifier** is an entity that defines the appearance, state, or behavior of a block or an element.

Rules for usage:

- The modifier name describes its appearance ("What size?" or "Which theme?" and so on; --sm or --theme-islands), its state ("How is it different from the others?"; --disabled, --focused, etc.) and its behavior ("How does it behave?" or "How does it respond to the user?"; such as --top).
- The modifier name is separated from the block or element name by a two dashes (--).
- A modifier cannot be used in isolation from the modified block or element. A modifier should change the appearance, behavior, or state of the entity, not replace it.
- A modifier is optional. Not all blocks or elements have modifiers.

```
<!-- The `search-form` block has the `focused` Boolean modifier -->
<form class="search-form search-form--focused">
  <input class="search-form__input">

  <!-- The `button` element has the `disabled` Boolean modifier -->
  <button class="search-form__button search-form__button--disabled">Search</button>
</form>
```

Mixing

A single HTML element can have a block class as well as an element class. This will allow you to combine the behavior and styles of multiple entities without duplicating code, and create semantically new UI components based on existing ones.

```
<!-- `header` block -->
<div class="header">
  <!-- The `search-form` block is mixed with the `search-form` element from the `header` block -->
  <div class="search-form header__search-form"></div>
</div>
```

This is the appropriate way to add margin and positioning to a block.

```
.search-form {} //don't add margin or positioning to the block
.header__search-form {} //add your margin and positioning here
```

You can also use mixing to make modifications per-page:

```
<body class="aboutPage">
  <!-- header and navigation-->
```

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

<!-- footer -->
<footer class="footer aboutPage__footer">...</footer>
</body>
```

Writing BEM in CSS

- Don't use tag selectors or ID selectors. The styles of blocks and elements are defined by class selectors only.
- Don't nest rules unless necessary. Nested selectors increase code complexity and make reuse difficult.
- Don't combine selectors (such as `.button.button--themelands`)

//good

```
.card {}
.card__title {}
.card--active {}
```

//bad

```
.card {}
.card .card__title {}
.card.card--active {}
```

- Sometimes, nesting is necessary and appropriate, such as when you need to change the styles of elements relative to the state of the block or the theme set:

```
.button--hovered .button__text {
  text-decoration: underline;
}

.button--themelands .button__text {
  line-height: 1.5;
}
```

Writing BEM in Sass

- You can take advantage of nesting in Sass to make your BEM rules easy to read:

```
.card {
  border: 1px solid black;
  Border-radius: 10px;
```

```
Padding: 1rem;
```

```
&__title {
```

```
Font-size: 2rem;
```

```
Font-weight: bold;
```

```
}
```

```
&--active {
```

```
Border-color: red;
```

```
}
```

```
}
```


This Sass code will compile to the following CSS:

```
.card {...}
```

```
.card__title {...}
```

```
.card--active {...}
```

Note: most of this content was taken from <https://en.bem.info/methodology/quick-start/> (but our naming conventions for modifiers differ from theirs). Check out that documentation for even more information.

 Like Be the first to like this

No labels

Powered by a free **Atlassian Confluence Community License**
granted to California State University, Northridge. Evaluate Confluence
today.