Bootstrap CSS Coding Standards

grammar

* With two spaces instead of tabs (tab) - This is the only way to ensure a consistent approach to show in all environments.
* When the selector is grouped separately on a line selector.
* For code readability, in front of the block each statement brace to add a space.
* Brace block statement alone should make the trip.
* Each statement is the statement : after should insert a space.
* In order to obtain more accurate error reporting, each statement should be a separate line.
* All declaration statement should end with a semicolon. Semicolon behind last declaration statement is optional, but if you omit the semicolon, your code may be more wrong.
* For property values separated by commas, each comma should be inserted into a space (for example, box-shadow ).
* Do not rgb() , rgba() , hsl() , hsla() or rect() behind the *internal* values comma insert a space. Such benefits from a plurality of attribute values ​​(also both comma and space) to distinguish between a plurality of color values ​​(only a comma, no spaces).
* For the color attribute value or parameter omitted less than 1 in front of the decimal 0 (for example, .5 instead of 0.5 ; -.5px replaced -0.5px ).
* Hexadecimal values should be all lowercase, for example, #fff . When scanning a document, lowercase characters easy to distinguish, because their form is more easily distinguishable.
* Try to use a short form of the hexadecimal value, for example, with #fff instead of #ffffff .
* Add double quotes for the selection of a property, for example, input[type="text"] . [Only in certain circumstances it is optional](https://mathiasbynens.be/notes/unquoted-attribute-values#css/index.html) , however, for consistency code proposals in double quotes.
* 0 units to the specified value to avoid, for example, with a margin: 0; instead of margin: 0px; .

For terms used herein have questions? Please refer to Wikipedia on [Cascading Style Sheets - grammar](https://zh.wikipedia.org/wiki/%E5%B1%82%E5%8F%A0%E6%A0%B7%E5%BC%8F%E8%A1%A8#.E8.AF.AD.E6.B3.95) .

/\* Bad CSS \*/

.selector, .selector-secondary, .selector[type=text] {

padding:15px;

margin:0px 0px 15px;

background-color:rgba(0, 0, 0, 0.5);

box-shadow:0px 1px 2px #CCC,inset 0 1px 0 #FFFFFF

}

/\* Good CSS \*/

.selector,

.selector-secondary,

.selector[type="text"] {

padding: 15px;

margin-bottom: 15px;

background-color: rgba(0,0,0,.5);

box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;

}

Declaration order

Property-related statements should be grouped and arranged in the following order:

1. Positioning
2. Box model
3. Typographic
4. Visual

As the positioning (positioning) can remove the element from the normal document flow, and also cover the box model (box model) related to style, so in the first row. The box model in second place, because it determines the size and position of components.

Other properties affect only *the internal* components *(inside)* or does not affect the first two groups attribute, so at the back.

Complete list of properties and their order refer to [Recess](https://twitter.github.com/recess/index.html) .

.declaration-order {

/\* Positioning \*/

position: absolute;

top: 0;

right: 0;

bottom: 0;

left: 0;

z-index: 100;

/\* Box-model \*/

display: block;

float: right;

width: 100px;

height: 100px;

/\* Typography \*/

font: normal 13px "Helvetica Neue", sans-serif;

line-height: 1.5;

color: #333;

text-align: center;

/\* Visual \*/

background-color: #f5f5f5;

border: 1px solid #e5e5e5;

border-radius: 3px;

/\* Misc \*/

opacity: 1;

}

Do not use @import

And <link> compared tag, @import command much slower, not only increases the number of additional requests, but also lead to unexpected problems. There are several alternatives:

* Using multiple <link> element
* By Sass or Less like CSS preprocessor multiple CSS files are compiled into one file
* By Rails, Jekyll or other system provided the CSS file merge feature

Refer to [Steve Souders article](http://www.stevesouders.com/blog/2009/04/09/dont-use-import/index.html) to learn more knowledge.

<!-- Use link elements -->

<link rel="stylesheet" href="core.css">

<!-- Avoid @imports -->

<style>

@import url("more.css");

</style>

Media queries (Media query) position

The media queries placed as close to the relevant rules. Do not pack them in a single style file, or at the bottom of the document. If you take them apart, and in the future everyone will be forgotten. Here's a typical example.

.element { ... }

.element-avatar { ... }

.element-selected { ... }

@media (min-width: 480px) {

.element { ...}

.element-avatar { ... }

.element-selected { ... }

}

With attributes prefix

When using vendor-specific attributes prefixed by indentation, which allows the value of each property are aligned in the vertical direction, it is easy to multi-line editing.

In Textmate, using **Text → Edit Each Line in Selection (** ^⌘A). In Sublime Text 2, use the **Selection → Add Previous Line (^⇧ ↑** ) and **Selection → Add Next Line (^⇧ ↓** ).

/\* Prefixed properties \*/

.selector {

-webkit-box-shadow: 0 1px 2px rgba(0,0,0,.15);

box-shadow: 0 1px 2px rgba(0,0,0,.15);

}

Single-line rule declaration

For styles that **contain only a statement,** in order to facilitate legibility and quick editing, we recommend that the statement on the same line. For a number of styles with a declaration or statement should be divided into multiple lines.

A key factor in doing this is to detect errors - for example, CSS validator pointed out that 183 line has a syntax error. If it is a single-line single statement, you will not ignore this error; if it is more than a single-line statement, you must carefully analyze the mistakes to avoid missing.

/\* Single declarations on one line \*/

.span1 { width: 60px; }

.span2 { width: 140px; }

.span3 { width: 220px; }

/\* Multiple declarations, one per line \*/

.sprite {

display: inline-block;

width: 16px;

height: 15px;

background-image: url(../img/sprite.png);

}

.icon { background-position: 0 0; }

.icon-home { background-position: 0 -20px; }

.icon-account { background-position: 0 -40px; }

Shorthand property declarations

In the need to explicitly set the value of all cases, we should try to limit the use of shorthand property declarations. Situation commonly abused shorthand property declared as follows:

* padding
* margin
* font
* background
* border
* border-radius

In most cases, we do not need to specify all values ​​for the attribute declaration abbreviated form. For example, HTML heading elements only on the set top and bottom margins (margin) value, therefore, when necessary, just covering these two values ​​can be. Excessive use the short form of the property declaration can lead to confusing code, property value and would bring unnecessary overlap causing unexpected side effects.

MDN (Mozilla Developer Network) on a very good idea about [shorthand properties](https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand_properties/index.html) of the article, for the less familiar shorthand property declaration and user behavior is useful.

/\* Bad example \*/

.element {

margin: 0 0 10px;

background: red;

background: url("image.jpg");

border-radius: 3px 3px 0 0;

}

/\* Good example \*/

.element {

margin-bottom: 10px;

background-color: red;

background-image: url("image.jpg");

border-top-left-radius: 3px;

border-top-right-radius: 3px;

}

Less and Sass nested

Avoid unnecessary nesting. This is because although you can use the nest, but that does not mean you should use nested. Only in the style you must limit within the parent element (ie descendant selector), and need only a plurality of nested elements using nested exist.

// Without nesting

.table > thead > tr > th { … }

.table > thead > tr > td { … }

// With nesting

.table > thead > tr {

> th { … }

> td { … }

}

Note

The code is written and maintained by the people. Please make sure that your code can be self-describing, well-commented and easy to understand others. Good code comments can convey context and purpose of the code. Do not simply reiterate component or class name.

For longer comments, be sure to write a complete sentence; for general comments, can write simple phrases.

/\* Bad example \*/

/\* Modal header \*/

.modal-header {

...

}

/\* Good example \*/

/\* Wrapping element for .modal-title and .modal-close \*/

.modal-header {

...

}

class name

* class names can appear only lowercase characters and dashes (dashe) (not underlined, nor hump nomenclature). Dash should be used for the relevant class named (like namespaces) (for example, .btn and .btn-danger ).
* Avoid excessive arbitrary shorthand. .btn representatives *button,* but .s not express any meaning.
* class name should be as short and clear meaning.
* Use meaningful names. Organized or purposeful use of the name, do not use expressions (presentational) name.
* Based on the nearest parent class or basic (base) class as a prefix for the new class.
* Use .js-\* class to identify the behavior (as opposed to style), and do not contain these class to the CSS file.

Sass and Less for variable naming is also read all the specifications listed above.

/\* Bad example \*/

.t { ... }

.red { ... }

.header { ... }

/\* Good example \*/

.tweet { ... }

.important { ... }

.tweet-header { ... }

Selector

* For common elements use class, so conducive to optimize rendering performance.
* For components often avoid using the property selectors (for example, [class^="..."] ). Browser performance will be affected by these factors.
* Selectors as short as possible, and try to limit the number of elements of the selector, it is recommended not to exceed 3.
* **Only** when it is necessary to limit in the last class of the parent element (ie descendant selector) (For example, do not use the class with the prefix - namespace prefix is similar).

Further reading:

* [Scope CSS classes with prefixes](http://markdotto.com/2012/02/16/scope-css-classes-with-prefixes/index.html)
* [Stop the cascade](http://markdotto.com/2012/03/02/stop-the-cascade/index.html)

/\* Bad example \*/

span { ... }

.page-container #stream .stream-item .tweet .tweet-header .username { ... }

.avatar { ... }

/\* Good example \*/

.avatar { ... }

.tweet-header .username { ... }

.tweet .avatar { ... }

Code Organization

* Component-unit organization code.
* Development of consistent annotation specification.
* Use consistent whitespace separated into blocks of code, so conducive to scan large documents.
* If more than one CSS file, in the form of spin-off assembly rather than the page, because the page will be reorganized, and the assembly will be moved.

/\*

\* Component section heading

\*/

.element { ... }

/\*

\* Component section heading

\*

\* Sometimes you need to include optional context for the entire component. Do that up here if it's important enough.

\*/

.element { ... }

/\* Contextual sub-component or modifer \*/

.element-heading { ... }

Configuration Editor

Your editor in accordance with the following configuration settings to avoid common code inconsistencies and differences:

* Two spaces instead of tabs (soft-tab that is representative of tab characters with spaces).
* When you save a file to remove the trailing whitespace.
* Set the file encoding to UTF-8.
* At the end of the file add a blank line.