**Grammar**

* All declaration statement ends with a semicolon.
* We use two spaces instead of tabs (tab) – as 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 is braced to add a space.
* Braced block statement alone makes the trip.
* Each statement in the statement after inserts a space.
* In order to obtain more accurate error reporting, each statements are on separate line.
* For property values separated by commas, each comma is inserted into a space.
* We didn’t use rgb(), rgba(), hsl(), hsla() or rect() behind the *internal* values comma insert a space.
* For the color attribute value or parameter it’s omitted less than 1 in front of the decimal 0.
* Hexadecimal values are all lowercase, for example, #fff. When scanning a document, lowercase characters easy to distinguish, because their form is more easily distinguishable.
* Used a short form of the hexadecimal value, for example, with #fff instead of #ffffff .
* Added double quotes for the selection of a property, for example, input[type="text"] .
* 0 units are specified value to avoid, for example, with a margin: 0; instead of margin: 0px;.

**Declaration order**

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

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

As the positioning can remove the element from the normal document flow, and also cover the 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 internalcomponents or does not affect the first two groups attribute, so at the back.

## Don’t use

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

## 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.

## 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.

## 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:

1. padding
2. margin
3. font
4. background
5. border
6. 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 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.

## 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.

## 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.

## 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.

## Selector

* For common elements use class, so conducive to optimize rendering performance.
* For components often avoid using the property selectors. 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).

## 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.