# **HTML Style Guide and Coding Conventions**

A consistent, clean, and tidy HTML code makes it easier for others to read and understand your code.

Here are some guidelines and tips for creating good HTML code.

Always Declare Document Type

Always declare the document type as the first line in your document.

The correct document type for HTML is:

<!DOCTYPE html>

## Use Lowercase Element Names

HTML allows mixing uppercase and lowercase letters in element names.

However, we recommend using lowercase element names, because:

* Mixing uppercase and lowercase names looks bad
* Developers normally use lowercase names
* Lowercase looks cleaner
* Lowercase is easier to write

## Close All HTML Elements

In HTML, you do not have to close all elements (for example the <p> element).

However, we strongly recommend closing all HTML elements.

## Use Lowercase Attribute Names

HTML allows mixing uppercase and lowercase letters in attribute names.

However, we recommend using lowercase attribute names, because:

* Mixing uppercase and lowercase names looks bad
* Developers normally use lowercase names
* Lowercase look cleaner
* Lowercase are easier to write

## Always Quote Attribute Values

HTML allows attribute values without quotes.

However, we recommend quoting attribute values, because:

* Developers normally quote attribute values
* Quoted values are easier to read
* You MUST use quotes if the value contains spaces

## Always Specify alt, width, and height for Images

Always specify the alt attribute for images. This attribute is important if the image for some reason cannot be displayed.

Also, always define the width and height of images. This reduces flickering, because the browser can reserve space for the image before loading.

## Spaces and Equal Signs

HTML allows spaces around equal signs. But space-less is easier to read and groups entities better together.

## Avoid Long Code Lines

When using an HTML editor, it is NOT convenient to scroll right and left to read the HTML code.

Try to avoid too long code lines.

## Blank Lines and Indentation

Do not add blank lines, spaces, or indentations without a reason.

For readability, add blank lines to separate large or logical code blocks.

For readability, add two spaces of indentation. Do not use the tab key.

# **CSS Style Guide and Coding Conventions**

#### CSS Validity

Use valid CSS where possible.

Unless dealing with CSS validator bugs or requiring proprietary syntax, use valid CSS code.

Use tools such as the [W3C CSS validator](https://jigsaw.w3.org/css-validator/) to test.

Using valid CSS is a measurable baseline quality attribute that allows to spot CSS code that may not have any effect and can be removed, and that ensures proper CSS usage.

#### ID and Class Naming

Use meaningful or generic ID and class names.

Instead of presentational or cryptic names, always use ID and class names that reflect the purpose of the element in question, or that are otherwise generic.

Names that are specific and reflect the purpose of the element should be preferred as these are most understandable and the least likely to change.

Generic names are simply a fallback for elements that have no particular or no meaning different from their siblings. They are typically needed as “helpers.”

Using functional or generic names reduces the probability of unnecessary document or template changes.

#### ID and Class Name Style

Use ID and class names that are as short as possible but as long as necessary.

Try to convey what an ID or class is about while being as brief as possible.

Using ID and class names this way contributes to acceptable levels of understandability and code efficiency.

#### Type Selectors

Avoid qualifying ID and class names with type selectors.

Unless necessary (for example with helper classes), do not use element names in conjunction with IDs or classes.

Avoiding unnecessary ancestor selectors is useful for [performance reasons](http://www.stevesouders.com/blog/2009/06/18/simplifying-css-selectors/).

#### Shorthand Properties

Use shorthand properties where possible.

CSS offers a variety of [shorthand](https://www.w3.org/TR/CSS21/about.html#shorthand) properties (like font) that should be used whenever possible, even in cases where only one value is explicitly set.

Using shorthand properties is useful for code efficiency and understandability.

#### 0 and Units

Omit unit specification after “0” values, unless required.

Do not use units after 0 values unless they are required.

#### Hexadecimal Notation

Use 3 character hexadecimal notation where possible.

For color values that permit it, 3 character hexadecimal notation is shorter and more succinct.

#### Prefixes

Prefix selectors with an application-specific prefix (optional).

In large projects as well as for code that gets embedded in other projects or on external sites use prefixes (as namespaces) for ID and class names. Use short, unique identifiers followed by a dash.

Using namespaces helps preventing naming conflicts and can make maintenance easier, for example in search and replace operations.

#### ID and Class Name Delimiters

Separate words in ID and class names by a hyphen.

Do not concatenate words and abbreviations in selectors by any characters (including none at all) other than hyphens, in order to improve understanding and scannability.

#### Block Content Indentation

Indent all block content.

Indent all [block content](https://www.w3.org/TR/CSS21/syndata.html#block), that is rules within rules as well as declarations, so to reflect hierarchy and improve understanding.

#### Declaration Stops

Use a semicolon after every declaration.

End every declaration with a semicolon for consistency and extensibility reasons.

#### Property Name Stops

Use a space after a property name’s colon.

Always use a single space between property and value (but no space between property and colon) for consistency reasons.

#### Rule Separation

Separate rules by new lines.

Always put a blank line (two line breaks) between rules.

#### CSS Quotation Marks

Use single ('') rather than double ("") quotation marks for attribute selectors and property values.

Do not use quotation marks in URI values (url()).

Exception: If you do need to use the @charset rule, use double quotation marks—[single quotation marks are not permitted](https://www.w3.org/TR/CSS21/syndata.html#charset).

#### Section Comments

Group sections by a section comment (optional).

If possible, group style sheet sections together by using comments. Separate sections with new lines.

[GitHub CSS Style Guide →](http://primercss.io/)

*As a rule of thumb, don’t nest further than 3 levels deep. If you find yourself going further, think about reorganizing your rules (either the specificity needed, or the layout of the nesting).*

*Unit-less line-height is preferred because it does not inherit a percentage value of its parent element, but instead is based on a multiplier of the font-size.*

[Google HTML/CSS Style Guide →](https://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml)

*Use ID and class names that are as short as possible but as long as necessary.*

E.g. #nav not #navigation, .author not .atr

*Do not concatenate words and abbreviations in selectors by any characters (including none at all) other than hyphens, in order to improve understanding and scannability.*

E.g. .demo-image not .demoimage or .demo\_image

[ThinkUp CSS Style Guide →](https://github.com/ginatrapani/ThinkUp/wiki/Code-Style-Guide:-CSS)

*If the value of the width or height is 0, do not specify units.*

*Comments that refer to selector blocks should be on a separate line immediately before the block to which they refer.*

[WordPress CSS Coding Standards →](https://make.wordpress.org/core/handbook/coding-standards/css/)

*Add two blank lines between sections and one blank line between blocks in a section.*

*Broad selectors allow us to be efficient, yet can have adverse consequences if not tested. Location-specific selectors can save us time, but will quickly lead to a cluttered stylesheet. Exercise your best judgement.*

# **JavaScript Style Guide and Coding Conventions**

# Variable Names

Use **camelCase** for identifier names (variables and functions).

All names start with a **letter**.

## Spaces Around Operators

Always put spaces around operators ( = + - \* / ), and after commas:

## Code Indentation

Always use 2 spaces for indentation of code blocks.

## Statement Rules

General rules for simple statements:

* Always end a simple statement with a semicolon.

General rules for complex (compound) statements:

* Put the opening bracket at the end of the first line.
* Use one space before the opening bracket.
* Put the closing bracket on a new line, without leading spaces.
* Do not end a complex statement with a semicolon.

## Object Rules

General rules for object definitions:

* Place the opening bracket on the same line as the object name.
* Use colon plus one space between each property and its value.
* Use quotes around string values, not around numeric values.
* Do not add a comma after the last property-value pair.
* Place the closing bracket on a new line, without leading spaces.
* Always end an object definition with a semicolon.

## Line Length < 80

For readability, avoid lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it, is after an operator or a comma.

## Naming Conventions

Always use the same naming convention for all your code. For example:

* Variable and function names written as **camelCase**
* Global variables written in **UPPERCASE**(We don't, but it's quite common)
* Constants (like PI) written in **UPPERCASE**

Hyphens can be mistaken as subtraction attempts. Hyphens are not allowed in JavaScript names.

Do not start names with a $ sign. It will put you in conflict with many JavaScript library names.

Accessing HTML Elements

A consequence of using "untidy" HTML styles, might result in JavaScript errors.

These two JavaScript statements will produce different results:

var obj = getElementById("Demo")  
  
var obj = getElementById("demo")

If possible, use the same naming convention (as JavaScript) in HTML.

## General Guide

## File Extensions

HTML files should have a **.html** extension (**.htm** is allowed).

CSS files should have a **.css** extension.

JavaScript files should have a **.js** extension.

## Use Lower Case File Names

Most web servers (Apache, Unix) are case sensitive about file names:

london.jpg cannot be accessed as London.jpg.

Other web servers (Microsoft, IIS) are not case sensitive:

london.jpg can be accessed as London.jpg or london.jpg.

If you use a mix of upper and lower case, you have to be extremely consistent.

If you move from a case insensitive, to a case sensitive server, even small errors can break your web site.

To avoid these problems, always use lower case file names (if possible).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CODING LANGUAGE** | | | | |
| **S.No.** | **NAME** | **Previous Version** | **Used Version** | **Remark** |
| 1 | HTML | HTML5 | HTML5 | No Change |
| 2  3 | CSS | CSS-5 | CSS-5 | No Change  No Change |
|  |  |  |  |  |
|  | | | | |
| **S.No.** | **NAME** | **Previous Version** | **Used Version** | **Remark** |
| 1 | react | 16.9.0 | 16.14.0 | Modification |
| 2 | react-dom | 16.9.0 | 16.14.0 | Modification |
| 3 | react-router-dom | 5.0.1 | 5.2.0 | Modification |
|  |  |  |  |  |

4 react-scripts 3.1.1 3.1.1 No Change

5 react-mdl 1.11.0 1.11.0 No Change