HTML AND CSS CODING GUIDELINES FOR DEVELOPER

Author: *Mohammed Raja*

Date	Version	Description
09-04-2014	**1.0**	**Initial Version**

Introduction:

This document defines formatting and style rules for HTML and CSS. The general concept behind standards is to provide predictable building blocks for others to build with so that the need for re-invention is minimized, to lower the costs of creation and innovation, and to reap the benefits of shared codes.

Table of contents

1. General Principle

Separation of Concerns: Separate structure from presentation from behavior.

Strictly keep structure (markup), presentation (styling), and behavior (scripting) apart, and try to keep the interaction between the three to an absolute minimum.

That is, make sure documents and templates contain only HTML and HTML that is solely serving structural purposes. Move everything presentational into style sheets, and everything behavioral into scripts.

Separating structure from presentation from behavior is important for maintenance reasons. It is always more expensive to change HTML documents and templates than it is to update style sheets and scripts.

1. A Whitespace

For all code languages, we require indentation to be done via soft tabs (using the space character). Hitting Tab in your text editor shall be equivalent to two spaces.

Readability vs. Compression:
We prefer readability over file-size savings when it comes to maintaining existing files. Plenty of whitespace is encouraged. There is no need for any developer to purposefully compress HTML or CSS, nor obfuscate JavaScript.
We will use server-side or build processes to automatically minify and gzip all static client-side files, such as CSS and JavaScript.

1. B Comments

A good program is beautiful in both its concept -- the algorithm used, the design, the flow of control -- but also in its readability. Proper use of commenting can make code maintenance much easier, as well as helping make finding bugs faster. Further, commenting is very important when writing functions that other people will use. Remember, well documented code is as important as correctly working code.

Where to Comment:
Comments should occur in the following places:

1. *The top of any file.*
   This is called the "Header Comment". It should include all the defining information about who wrote the code, and why, and when, and what it should do. (See Header Comment below)
2. *Above every function.*
   This is called the function header and provides information about the purpose of this "sub-component" of the program.
3. *In line.*
   Any "tricky" code where it is not immediately obvious what you are trying to accomplish should have comments right above it or on the same line with it.

2. A Divide and conquer

Divide your code in multiple files to maintain overview of single groups of code fragments. In such situations **master stylesheet** is used to import groups. Using master-stylesheet you generate some unnecessary server requests, but the approach produces a clean and elegant code which is easy to reuse, easy to understand and also easy to maintain. And you also need to include only the master-file in your documents.

2. B Define color and typography

This gives a quick reference to the colors used in the site to avoid using alternates by mistake and, if you need to change the colors, you have a quick list to go down and do a search and replace.”

2. C Indentation

Use 2 spaces at a time. Do not use tabs or mix both tabs and space

2. D Naming Convention

ID name should be in lowerCamelCase

#pageContainer {
Class name should be in lowercase, with words separated by underscore.

.my_class_name {
Property and value should be in lower case.

```
/* Not recommended */
color: #E5E5E5;

/* Recommended */
color: #e5e5e5;
```

Always use meaningful or generic ID and class name. Use ID and class names that are as short as possible but as long as necessary.

2. E Use external stylesheet

Do not write inline styles or embedded styles unless unavoidable. For performance reasons, always link to external stylesheets using the <link> rather than @import

2. F Comments

Explain code as needed, where possible. Comments that refer to selector blocks should be on a separate line immediately before the block to which they refer. Short inline comments may be added after a property-value pair, preceded with a space.

```
/* Comment about this selector block. */

selector {

  property: value; /* Comment about this property-value pair. */

}
```

C style comments (/* Comment goes here. */) are valid for CSS code.

2. G Selector

Selectors should be on a single line, with a space after the last selector, followed by an opening brace. A selector block should end with a closing curly brace that is unindented and on a separate line. A blank line should be placed between each selector block. Selectors should never be indented.

```
selector {

}



selector {

}
```

Multiple selectors should each be on a single line, with no space after each comma.

```
selector1,

selector2,

selector3,

selector4 {

}
```

When selecting HTML elements, write the selector in lowercase.

```
div { /* Okay */

DIV { /* Not okay
```

2. H Property-Value Pairs

Property-value pairs should be listed starting on the line after the opening curly brace. Each pair should:

- be on its own line;
- be indented one level;
- have a single space after the colon that separates the property name from the property value;

- end in a semicolon.

```
selector {
  name: value;
  name: value;
}
```

Multiple property-value pairs should be listed in alphabetical order by property.
For properties with multiple values, separate each value with a single space following the comma(s).

```
font-family: Helvetica, sans-serif;
```

If a single value contains any spaces, that value must be enclosed within double quotation marks.

```
font-family: "Lucida Grande", Helvetica, sans-serif;
```

2. I Colors

If it's possible to specify the desired color using three-digit hexadecimal notation, do so as you'll save the end-user a few bytes of download time.

2. J Dimensions

When denoting the dimensions — that is, the width or height — of an element or its margins, borders, or padding, specify the units in either em, px, or %. If the value of the width or height is 0, do not specify units.

```
width: 12px;  /* Okay */
width: 12%;   /* Okay */
width: 12em;  /* Okay */
width: 12rem; /* Okay */
width: 0;     /* Okay */
width: 12;    /* Not okay */
```

```
width: 0px;   /* Not okay */
```

2. K Avoid universal rules

Make sure a rule doesn't end up in the universal category! Avoid to style on HTML tag.

2. L Do not qualify ID rules with tag names or classes

If a rule has an ID selector as its key selector, don't add the tag name to the rule. Since IDs are unique, adding a tag name would slow down the matching process needlessly.

2. M Use scoped style sheet

If you specify a stylesheet as an XBL resource, the styles only apply to the bound elements and their anonymous content. This reduces the inefficiency of universal rules and child selectors because there are fewer elements to consider.

2. N File Naming and Organization

Filenames should contain lowercase letters and words should be separated with a hyphen.

3. A General Markup Guidelines

- Use actual P elements for paragraph delimiters as opposed to multiple BR tags.
- Make use of DL (definition lists) and BLOCKQUOTE, when appropriate.
- Items in list form should always be housed in a UL, OL, or DL, never a set of DIVs or Ps.
- Use label fields to label each form field, the for attribute should associate itself with the input field, so users can click the labels. cursor:pointer; on the label is wise, as well. note 1 note 2
- Do not use the size attribute on your input fields. The size attribute is relative to the font-size of the text inside the input. Instead use css width.
- Place an html comment on some closing div tags to indicate what element you're closing. It will help when there is lots of nesting and indentation.
- Tables shouldn't be used for page layout.
- Use micro formats and/or Microdata where appropriate, specifically hCard and adr.
- When quoting attribute values, use double quotation marks.
- Provide alternative content for multimedia.

```
<!-- Not recommended -->
<img src="spreadsheet.png">
```

```
<!-- Recommended -->
<img src="spreadsheet.png" alt="Spreadsheet screenshot.">
```

Bootstrap Coding guidline

General

1. Items must use index.html as the filename for the index page. Other pages must use the .html file structure.
2. Custom styles and scripts must be kept separate from the default Bootstrap assets. For example, when modifying the CSS, leave the bootstrap.min.css file intact and create an external stylesheet.
3. HTML needs to be validated via the W3C validator. However, browser prefixes and any other cutting edge code will be exempt.
4. Coding should be tabbed, properly formatted and easy to follow.
5. Complex items should include proper documentation
6. Items should be free from spelling and grammatical errors.
7. JavaScript shouldn't raise any errors or notices.
8. IDs and classes must be appropriately named and follow a naming convention.
9. Third-party assets used to build your item must be properly licensed or free for commercial use.
10. Sufficient information regarding third-party assets must be present in your documentation. This includes the author, license info and a direct link to the asset online.
11. Complex items should include documentation or help files to assist-end users especially when customizations are included in the item. This may include, but is not limited to, information on how to add/edit/remove specific elements, create new pages and apply styles to elements.
12. Documentation must include a change log for each version if applicable.
13. Items should not implement custom components or JavaScript plugins that duplicate the functionality already included in the Bootstrap framework unless it is not possible to achieve a desired outcome using the standard components or plugins

Syntax

1. Use soft tabs with two spaces—they're the only way to guarantee code renders the same in any environment.
2. Nested elements should be indented once (two spaces).
3. Always use double quotes, never single quotes, on attributes.
4. Don't include a trailing slash in self-closing elements—the HTML5 spec says they're optional.
5. Don’t omit optional closing tags (e.g. or</body>).
6. When grouping selectors, keep individual selectors to a single line.
7. Don't prefix property values or color parameters with a leading zero (e.g., .5 instead of 0.5 and -.5px instead of -0.5px).
8. Lowercase all hex values, e.g., #fff. Lowercase letters are much easier to discern when scanning a document as they tend to have more unique shapes.

CSS and JavaScript includes

Per HTML5 spec, typically there is no need to specify a typewhen including CSS and JavaScript files as text/css andtext/javascript are their respective defaults.

```
<!-- External CSS -->
<link rel="stylesheet" href="code-guide.css">

<!-- In-document CSS -->
<style>
  /* ... */
</style>

<!-- JavaScript -->
<script src="code-guide.js"></script>
```

Attribute order

HTML attributes should come in this particular order for easier reading of code**.**

- class
- id, name
- data-*
- src, for, type, href
- title, alt
- aria-*, role

Reducing markup

Whenever possible, avoid superfluous parent elements when writing HTML. Many times this requires iteration and refactoring, but produces less HTML.

Take the following example:

```
<!-- Not so great -->
<span class="avatar">
  <img src="...">
</span>

<!-- Better -->
<img class="avatar" src="...">
```

Shorthand notation

Strive to limit use of shorthand declarations to instances where you must explicitly set all the available values. Common overused shorthand properties include:

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

```
/* 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;
}
```

Selectors

- Use classes over generic element tag for optimum rendering performance.
- Avoid using several attribute selectors (e.g., [class^="..."]) on commonly occuring components. Browser performance is known to be impacted by these.
- Keep selectors short and strive to limit the number of elements in each selector to three.
- Scope classes to the closest parent only when necessary (e.g., when not using prefixed classes).

```
/* Bad example */
span { ... }
.page-container #stream .stream-item .tweet .tweet-header .username { ... }
.avatar { ... }

/* Good example */
.avatar { ... }
.tweet-header .username { ... }
.tweet .avatar { ... }
```

Editor preferences

Set your editor to the following settings to avoid common code inconsistencies and dirty diffs:

- Use soft-tabs set to two spaces.
- Trim trailing white space on save.
- Set encoding to UTF-8.
- Add new line at end of files.

Resources *http://codeguide.co/#css*
http://bootstrapbay.com/submission-guidelines