# Standards

General best practices as outlined in the Cognizant Interactive Best Practices should be adhered to during the development of Bootstrap. This document does not attempt to restate those best practices. Instead it outlines additional best practices specific to Scripps and Bootstrap.

Since the code for Bootstrap is the result of a POC, it may not match all of these standards at this time. When modifying a file, please take time to ensure that all standards are followed.

## HTML

* HTML should be XHTML compliant with closing tags and proper nesting.
* Element and attribute names should be in lower case.
* Element IDs should use camel casing.
* Indentation should be 2 spaces per tab.
* HTML should not have inline style attributes or use deprecated style elements or attributes. Appearances should be applied through CSS.
* As appropriate, use comments on ending tags to denote the end of a section of the document. The comment should use a forward slash followed by a CSS selector that could be used to apply a style. Example:

|  |  |
| --- | --- |
| **Class based closing comment** | **ID based closing comment** |
| **<div class=”showTitle”>**  **...**  **</div> <!-- /.showTitle -->** | **<div id=”footer”>**  **...**  **</div> <!-- /#footer -->** |

* + Do not use comments on every end tag. These types of comments should add value when scanning the document to find a particular section or area of the HTML.
* Code should validate using the W3C HTML5 validator.
* HTML in Bootstrap is based largely on [HTML5Boilerplate](http://html5boilerplate.com/). This open source project should be used as the model for best practices for areas not covered by this document.

## LESS/CSS

* Selectors and rules should appear on separate lines.

**Example:**

.someContent,

.otherContent {

color: #ff0000;

padding-top: 24px;

}

* CSS classes should be use camel casing.
* Indentation should be 4 spaces per tab.
* Properties should be sorted alphabetically
* The open bracket should be on the same line as the last selector.
* Use LESS mixins for properties that require browser prefixes.
* Each LESS file should:
  + Have a header that identifies the source file name
  + Define any configuration variables
  + List rules specific to the feature/component
* Selectors should have the minimum specificity required to apply the style. In general this should be limited to two levels of class selectors.

|  |  |
| --- | --- |
| **Correct** | **Incorrect** |
| .container .content | html div#id .container .content |

* In general, element selectors should only be used to apply base appearances. Element selectors may be used after a class selector if there is no chance of un-intended consequences.

|  |  |
| --- | --- |
| **Correct** | **Incorrect** |
| .globalNavigation li {  ...  } | .pageContent li {  ...  } |
| The <li> selector is scoped to the global navigation component. Since this component is not a container for other content, the chance of side effects is low. | The <li> selector is scoped to the content area of the page. Since this is a container for other content, it has a high chance of side effects. A second class selector should be used instead of the element selector. |

## JavaScript

* JavaScript files should include a header with the author, date created, application (Scripps Bootstrap) and a brief description.
* Functions and variables should use camel casing
* Constructor functions and “namespaces” should use Pascal casing
* Indentation should be 4 spaces per tab.
* First open bracket should always be on the same line as the control statement.
* Code should validate using JSLint/JSHint
* JavaScript files that are used to initialize components or provide business logic for a specific page should have the same name as the page. Example:

|  |  |
| --- | --- |
| **JavaScript Widgets Page** | |
| HTML | javascript.html |
| JavaScript | javascript.js |

* Custom JavaScript libraries should have the following naming conventions:
  + Development library

Pattern:               sniui.{component}.js

Example:             sniui.dialog.js

* + Release library

Pattern:               sniui.{component}.{major}.{minor}.{build}.min.js

Example:             sniui.dialog.1.1.0.min.js

## Components

* Most parts of Bootstrap will consist of HTML, CSS and JavaScript. This content should be maintained as follows:
  + HTML should be documented in the Bootstrap documentation on the appropriate page:
    - **Scaffolding** for layouts and responsive features
    - **Base CSS** for fonts, colors, typography and basic HTML elements
    - **Components** for design patterns that did not need to leverage JavaScript
    - **JavaScript widgets** for design patterns that leverage JavaScript
  + Each component should have a unique LESS/CSS file. The LESS/CSS file should be placed in one of the following folders as appropriate:
    - css\core - Basic appearances (fonts, colors, layouts, etc.)
    - css\components - Design patterns implemented as HTML/CSS only
    - css\widgets - Appearance overrides for JavaScript widgets
    - css\responsive - Overridden appearances for responsive layouts
  + JavaScript
* Third party libraries may also contain CSS and JavaScript.
  + CSS should be placed in a subfolder in the CSS folder based on the library name.

Example: css\jquery-ui

* + JavaScript should be placed in the libs\js folder.
* Each component should have a single root element. The root element should have a CSS class applied based on the name of the element.
* CSS classes for sub-components or extensions to the base CSS class should have a naming format of {base CSS class name}-{sub-component/extension name}.

Example:

<header class="pageHeader pageHeader-noNavigation">

<div class="pageHeader-portal">

<div class="pageHeader-logo">

<a href="/"></a>

</div>

</div>

</header>

* + <header> is the root element
  + pageHeader is the base CSS class
  + pageHeader-noNavigation is an extension CSS class that customizes the component
  + pageHeader-portal and pageHeader-logo are sub-components