TATA CONSULTANCY SERVICES
Experience certainty.

CSS Coding Standards & best Practice

July, 2013

Version 2.0

1. Structure

There are plenty of different methods for structuring a style sheet. With the CSS in core, it is important to retain a high degree of legibility. This enables subsequent contributors to have a clear understanding of the flow of the document.

•Use tabs, not spaces, to indent each property.

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

•Each selector should be on its own line, ending in either a comma or an opening curly brace. Property-value pairs should be on their own line, with one tab of indentation and an ending semicolon. The closing brace should be flush left, using the same level of indentation as the opening selector.

Correct:

#selector-1, #selector-2, #selector-3 { background: #fff; color: #000; }

Incorrect:

#selector-1, #selector-2, #selector-3 { background: #fff; color: #000; } #selector-1 { background: #fff; color: #000; }

Declare you're most generic items first, then the not-so-generic and so on, your structure should be based on your layout most generic structure is given below:

A. Resets and overrides
B. Links and type
C. Main layout
D. Secondary layout structures
E. Form elements
F. Miscellaneous

Title, Project, Version, Date: - let other know who wrote CSS

```
/*******************************************
---------------------------------------------------------------
---------------------------------------------------------------
Title                   : - Master.CSS
Author                  : - Prerit Ahuja
Project                 : - Pfizer.com
Description             :- mother marketing website for pfizer.com
Version                 : - 1.0
---------------------------------------------------------------
---------------------------------------------------------------
*******************************************/
```

2. Selectors

With specificity, comes great responsibility. 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 style sheet. Exercise your best judgment to create selectors that find the right balance between contributing to the overall style and layout of the DOM.

• use lowercase and separate words with hyphens when naming selectors. Avoid camel case and underscores.

•Use human readable selectors that describe what element(s) they style.

•Attribute selectors should use double quotes around values

•Refrain from using over-qualified selectors, div.container can simply be stated as .container

Correct:

```
#comment-form {     margin: 1em 0; }   input[type="text"] {     line-height: 1.1; }
```

Incorrect:

```
#commentForm { /* Avoid camelcase. */    margin: 0; }   #comment_form { /* Avoid underscores. */ margin: 0; }   div#comment_form { /* Avoid over-qualification. */    margin: 0; }   #c1-xr { /* What is a c1-xr?! Use a better name. */    margin: 0; }   input[type=text] { /* Should be [type="text"] */    line-height: 110% /* Also doubly incorrect */}
```

3.Properties

Similar to selectors, properties that are too specific will hinder the flexibility of the design. Less is more. Make sure you are not repeating styling or introducing fixed dimensions (when a fluid solution is more acceptable).

•Properties should be followed by a colon and a space.

•All properties and values should be lowercase, except for font names and vendor-specific properties.

•Use hex code for colors, or rgba() if opacity is needed. Avoid RGB format and uppercase, and shorten values when possible: #fff instead of #FFFFFF.

•Use shorthand (except when overriding styles) for background, border, font, list-style, margin, and padding values as much as possible. (For a shorthand reference, see CSS Shorthand.)

Property Ordering #

“Group like properties together, especially if you have a lot of them.”

Above all else, choose something that is meaningful to you and semantic in some way. Random ordering is chaos, not poetry. Our recommended way is logical or grouped ordering, wherein properties are grouped by meaning and ordered specifically within those groups. The properties within groups are also strategically ordered to create transitions between sections, such as background directly before color. The baseline for ordering is:

•Display

•Positioning

•Box model

•Colors and Typography

•Other

Top/Right/Bottom/Left (TRBL/trouble) should be the order for any relevant properties (e.g. margin), much as the order goes in values. Corner specifiers (e.g. border-radius-*-*) should be top-left, top-right, bottom-right, bottom-left. This is derived from how shorthand values would be ordered.

Example:

#overlay { position: absolute; z-index: 1; padding: 10px; background: #fff; color: #777; }

4.Vendor Prefixes

Vendor prefixes should go longest (-webkit-) to shortest (unprefixed). Values should be right aligned with spaces after the colon provided that all the values are the same across all prefixes.

Preferred method:

.koop-shiny { -webkit-box-shadow: inset 0 0 1px 1px #eee; -moz-box-shadow: inset 0 0 1px 1px #eee; box-shadow: inset 0 0 1px 1px #eee; -webkit-transition: border-color 0.1s; -moz-transition: border-color 0.1s; -ms-transition: border-color 0.1s; -o-transition: border-color 0.1s; transition: border-color 0.1s; }

Not preferred:

.okay { -webkit-box-shadow: inset 0 0 1px 1px #eee; -moz-box-shadow: inset 0 0 1px 1px #eee; box-shadow: inset 0 0 1px 1px #eee; } .bad { -webkit-box-shadow: inset 0 0 1px 1px #eee; -moz-box-shadow: inset 0 0 1px 1px #eee; box-shadow: inset 0 0 1px 1px #eee; }

Special case for CSS gradients:

.gradient { background: #fff; background-image: -webkit-gradient(linear, left bottom, left top, from(#fff), to(#000)); background-image: -webkit-linear-gradient(bottom, #fff, #000); background-image: -moz-linear-gradient(bottom, #fff, #000); background-image: -o-linear-gradient(bottom, #fff, #000); background-image: linear-gradient(to top, #fff, #000); }

5. Values

There are numerous ways to input values for properties. Follow the guidelines below to help us retain a high degree of consistency.

•Space before the value, after the colon

•Do not pad parentheses with spaces

•Always end in a semicolon

•Use double quotes rather than single quotes, and only when needed, such as when a font name has a space.

•0 values should not have units unless necessary, such as with transition-duration.

•Line height should also be unit-less, unless necessary to be defined as a specific pixel value. This is more than just a style convention, but is worth mentioning here. More information: http://meyerweb.com/eric/thoughts/2006/02/08/unitless-line-heights/

•Use a leading zero for decimal values, including in rgba().

•Multiple comma-separated values for one property should be separated by either a space or a newline, including within rgba(). Newlines should be used for lengthier multi-part values such as those for shorthand properties like box-shadow and text-shadow. Each subsequent value after the first should then be on a new line, indented to the same level as the selector and then spaced over to left-align with the previous value.

Correct:

.class { /* Correct usage of quotes */ background-image: url(images/bg.png); font-family: "Helvetica Neue", sans-serif; } .class { /* Correct usage of zero values */ font-family: Georgia, serif; text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.5), 0 1px 0 #fff; }

Incorrect:

.class { /* Avoid missing space and semicolon */ background:#fff} .class { /* Avoid adding a unit on a zero value */ margin: 0px 0px 20px 0px; }

6. Media Queries

Media queries allow us to gracefully degrade the DOM for different screen sizes. If you are adding any, be sure to test above and below the break-point you are targeting.

•It is generally advisable to keep media queries grouped by media at the bottom of the style sheet.

•An exception is made for the wp-admin.css file in core, as it is very large and each section essentially represents a style sheet of its own. Media queries are therefore added at the bottom of sections as applicable.

•Rule sets for media queries should be indented one level in.

Example:

@media all and (max-width: 699px) and (min-width: 520px) { /* Your selectors */}

7. Commenting

•Comment, and comment liberally. If there are concerns about file size, utilize minified files and the SCRIPT_DEBUG constant. Long comments should manually break the line length at 80 characters.

• Section/subsection headers should have newlines before and after. Inline comments should not have empty newlines separating the comment from the item to which it relates.

For sections and subsections:

Write proper comments & indentation with blocks so it's easy to recognize & maintain sections.

```
/* ----------------------------------------*/
/* ---------->>> GLOBAL <<<----------*/
/* ----------------------------------------*/
```

/** * #.# Section title * * Description of section, whether or not it has media queries, etc. */ .selector { float: left; }

For inline:

/* This is a comment about this selector */ .another-selector { position: absolute; top: 0! important; /* I should explain why this is so !important */}

8. Best Practices

Style sheets tend to get long in length. Focus slowly gets lost whilst intended goals start repeating and overlapping. Writing smart code from the outset helps us retain the overview whilst remaining flexible throughout change.

- If you are attempting to fix an issue, attempt to remove code before adding more.
- DOM will change over time, target the element you want to use as opposed to "finding it" through its parents. Example: Use .highlight on the element as opposed to .highlight a (where the selector is on the parent)
- Know when to use the height property. It should be used when you are including outside elements (such as images). Otherwise use line-height for more flexibility.
- Do not restate default property & value combinations (for instance display: block; on block-level elements).

- **Don't Repeat Yourself**: - don't declare selectors multiple times in CSS
- **Optimize for Lightweight Style Sheets**. Try to minify CSS
- **Write Your Base for Gecko** (Firefox Mozilla), Then Tweak for Webkit and IE
- Use Ems instead of Pxs for font's size. Set font-size on the body-tag with 62.5%. Default-value of the **font-size** is 16px; applying the rule, you'll get one Em standing for roughly ten pixels (16 x 62.5% = 10). This allows you to use EMs to specify sizes while thinking in PX terms, e.g. 1.3em is approximately 13px.
- **Don't use @import** for calling other CSS
- **Minimal use of !important**.
- Don't use hacks unless it's a known and documented bug
- Take care of margins and padding on all elements that you use
- Avoid using too much absolute positioning
- Avoid using negative margins
- Validate Your CSS and XHTML. Refer http://csslint.net/ or http://jigsaw.w3.org/css-validator/ for validating your CSS