|  |
| --- |
|  |
|  |
|  |
| CA/CST  CSS Coding Standards  Version 1.0 |
| February 15, 2014 |
|  |
|  |
|  |

Record of Changes

| No. | Date | | | | Reference: Page, Table, Figure, Paragraph | A = Add. M = Mod. D = Del. | | Change Description | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | | | 02/12/2014 | Entire Document | |  | | Initial Draft | | |
|  | |  | | |  | |  | |  |
|  |  | | | |  |  | |  | | |
|  |  | | | |  |  | |  | | |
|  |  | | | |  |  | |  | | |
|  |  | | | |  |  | |  | | |
|  |  | | | |  |  | |  | | |
|  |  | | | |  |  | |  | | |
|  |  | | | |  |  | |  | | |
|  |  | | | |  |  | |  | | |
|  |  | | | |  |  | |  | | |
|  |  | | | |  |  | |  | | |
|  |  | | | |  |  | |  | | |
|  |  | | | |  |  | |  | | |

Contents

[1 Introduction 1](#_Toc380436794)

[1.1 Purpose 1](#_Toc380436795)

[1.2 Scope 1](#_Toc380436796)

[1.3 Code Change Scope 1](#_Toc380436797)

[2 References 3](#_Toc380436798)

[3 Validation 4](#_Toc380436799)

[4 Self-closing Elements 4](#_Toc380436800)

[5 Attributes and Tags 4](#_Toc380436801)

[6 Quotes 4](#_Toc380436802)

[7 Indentation 5](#_Toc380436803)

[8 Protocol 7](#_Toc380436804)

[9 Indentation 7](#_Toc380436805)

[10 Capitalization 7](#_Toc380436806)

[11 Trailing Whitespace 8](#_Toc380436807)

[12 Encoding 8](#_Toc380436808)

[13 Comments 8](#_Toc380436809)

[14 Action Items 8](#_Toc380436810)

[15 Document Type 8](#_Toc380436811)

[16 HTML Validity 9](#_Toc380436812)

[17 Semantics 9](#_Toc380436813)

[18 Multimedia Fallback 9](#_Toc380436814)

[19 Separation of Concerns 10](#_Toc380436815)

[20 Optional Tags 10](#_Toc380436816)

[21 type Attributes 11](#_Toc380436817)

[22 General Formatting 11](#_Toc380436818)

[23 HTML Quotation Marks 12](#_Toc380436819)

# Introduction

This document is derived from HTML style guides available from Google and Wordpress. The document is intentionally kept minimal to give flexibility to the developer in implementing code and to account for code generated by applications which can be configured.

## Purpose

Coding conventions help make sure that project code has a consistent structure and style. They are intended to make the code easier to read, understand, review, and maintain and further reduce the complexity of the code.

Additional coding guidelines in the form of code metrics (measurements) and coding rules are provided in this document for developers to use. The code metrics guidelines consist of “hand-calculable” measurements within each method and each class. The coding rules are based on information gathered from multiple sources to enhance the security, reliability, maintainability, testability and performance of the code.

This document is intended for the following uses:

* Desk-side reference for CA/CST HTML developers during coding.
* Source for GTM code review checklists and criteria.
* Reference for developers who must develop extend and maintain the CA/CST Web Applications.
* As a Governance Tool for CA/CST GTM’s.

## Scope

This document describes the following for the CA/CST:

* Naming conventions for projects, files, objects, variables, and other code constructs.
* Formatting conventions for code modules and their comments.
* Error handling conventions.
* Complexity conventions.
* Section 508 compliance.
* Security standards.
* Logging conventions.
* Coding practices and recommendations.

## Code Change Scope

The HTML coding standards described in this document apply to new applications (new code) and existing code in the following ways:

* New code in the middle of an existing file should follow new coding standards with explanations regarding the change being made to be consistent with new guidelines, ensuring no disruption to the existing code structure. Exceptions can be made to this rule if following new guidelines creates significant and unnecessary and potentially dangerous re-work.
* New files within existing application must follow the Java coding standards documented in this document.
* New applications must follow the Java coding standards documented in this document.

# References

The following sources were used in creation of the original version of this standard.

* <http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml>
* <http://make.wordpress.org/core/handbook/coding-standards/html/>

# Structure [#](http://make.wordpress.org/core/handbook/coding-standards/css/#structure)

There are plenty of different methods for structuring a stylesheet. 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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | #selector-1,  #selector-2,  #selector-3 {      background: #fff;      color: #000;  } |

Incorrect:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | #selector-1, #selector-2, #selector-3 {      background: #fff;      color: #000;      }    #selector-1 { background: #fff; color: #000; } |

[Top ↑](http://make.wordpress.org/core/handbook/coding-standards/css/#top)

# Selectors [#](http://make.wordpress.org/core/handbook/coding-standards/css/#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 stylesheet. Exercise your best judgement to create selectors that find the right balance between contributing to the overall style and layout of the DOM.

* + Similar to the [WordPress Coding Standards](http://codex.wordpress.org/WordPress_Coding_Standards) for file names, use lowercase and separate words with hyphens when naming selectors. Avoid camelcase 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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #comment-form {      margin: 1em 0;  }    input[type="text"] {      line-height: 1.1;  } |

Incorrect:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | #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 \*/  } |

[Top ↑](http://make.wordpress.org/core/handbook/coding-standards/css/#top)

# Properties [#](http://make.wordpress.org/core/handbook/coding-standards/css/#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](http://codex.wordpress.org/CSS_Shorthand).)

# Property Ordering [#](http://make.wordpress.org/core/handbook/coding-standards/css/#property-ordering)

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

Above all else, choose something that is meaningful to you and semantic in some way. Random ordering is chaos, not poetry. In WordPress Core, our choice 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

Things that are not yet used in core itself, such as CSS3 animations, may not have a prescribed place above but likely would fit into one of the above in a logical manner. Just as CSS is evolving, so our standards will evolve with it.

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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #overlay {      position: absolute;      z-index: 1;      padding: 10px;      background: #fff;      color: #777;  } |

Another method that is often used, including by the Automattic/WordPress.com Themes Team, is to order properties alphabetically, with or without certain exceptions.

Example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #overlay {      background: #fff;      color: #777;      padding: 10px;      position: absolute;      z-index: 1;  } |

[Top ↑](http://make.wordpress.org/core/handbook/coding-standards/css/#top)

# Vendor Prefixes [#](http://make.wordpress.org/core/handbook/coding-standards/css/#vendor-prefixes)

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

Preferred method:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | .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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | .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](http://core.trac.wordpress.org/ticket/16461#comment:43) for CSS gradients:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | .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);  } |

[Top ↑](http://make.wordpress.org/core/handbook/coding-standards/css/#top)

# Values [#](http://make.wordpress.org/core/handbook/coding-standards/css/#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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | .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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | .class { /\* Avoid missing space and semicolon \*/      background:#fff  }    .class { /\* Avoid adding a unit on a zero value \*/      margin: 0px 0px 20px 0px;  } |

[Top ↑](http://make.wordpress.org/core/handbook/coding-standards/css/#top)

# Media Queries [#](http://make.wordpress.org/core/handbook/coding-standards/css/#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 stylesheet.
    - An exception is made for the wp-admin.css file in core, as it is very large and each section essentially represents a stylesheet 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:

|  |  |
| --- | --- |
| 1  2  3  4 | @media all and (max-width: 699px) and (min-width: 520px) {            /\* Your selectors \*/  } |

[Top ↑](http://make.wordpress.org/core/handbook/coding-standards/css/#top)

# Commenting [#](http://make.wordpress.org/core/handbook/coding-standards/css/#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.
  + A table of contents should be utilized for longer stylesheets, especially those that are highly sectioned. Using an index number (1.0, 1.1, 2.0, etc.) aids in searching and jumping to a location.
  + Comments should be formatted much as PHPDoc is. The [CSSDoc](http://cssdoc.net/) standard is not necessarily widely accepted or used but some aspects of it may be adopted over time. 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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | /\*\*  \* #.# Section title  \*  \* Description of section, whether or not it has media queries, etc.  \*/    .selector {      float: left;  } |

For inline:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | /\* This is a comment about this selector \*/    .another-selector {      position: absolute;      top: 0 !important; /\* I should explain why this is so !important \*/  } |

[Top ↑](http://make.wordpress.org/core/handbook/coding-standards/css/#top)

# Best Practices [#](http://make.wordpress.org/core/handbook/coding-standards/css/#best-practices)

Stylesheets 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.
  + Magic Numbers are unlucky. These are numbers that are used as quick fixes on a one-off basis. Example: .box { margin-top: 37px }.
  + 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).

# 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](http://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

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=ID_and_Class_Naming#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.

/\* Not recommended: meaningless \*/ #yee-1901 {} /\* Not recommended: presentational \*/ .button-green {} .clear {}

/\* Recommended: specific \*/ #gallery {} #login {} .video {} /\* Recommended: generic \*/ .aux {} .alt {}

# ID and Class Name Style

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=ID_and_Class_Name_Style#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.

/\* Not recommended \*/ #navigation {} .atr {}

/\* Recommended \*/ #nav {} .author {}

# Type Selectors

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Type_Selectors#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/).

/\* Not recommended \*/ ul#example {} div.error {}

/\* Recommended \*/ #example {} .error {}

# Shorthand Properties

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Shorthand_Properties#Shorthand_Properties)▽

Use shorthand properties where possible.

CSS offers a variety of [shorthand](http://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.

/\* Not recommended \*/ border-top-style: none; font-family: palatino, georgia, serif; font-size: 100%; line-height: 1.6; padding-bottom: 2em; padding-left: 1em; padding-right: 1em; padding-top: 0;

/\* Recommended \*/ border-top: 0; font: 100%/1.6 palatino, georgia, serif; padding: 0 1em 2em;

# 0 and Units

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=0_and_Units#0_and_Units)▽

Omit unit specification after “0” values.

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

margin: 0; padding: 0;

# Leading 0s

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Leading_0s#Leading_0s)▽

Omit leading “0”s in values.

Do not use put 0s in front of values or lengths between -1 and 1.

font-size: .8em;

# Hexadecimal Notation

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Hexadecimal_Notation#Hexadecimal_Notation)▽

Use 3 character hexadecimal notation where possible.

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

/\* Not recommended \*/ color: #eebbcc;

/\* Recommended \*/ color: #ebc;

# Prefixes

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Prefixes#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.

.adw-help {} /\* AdWords \*/ #maia-note {} /\* Maia \*/

# ID and Class Name Delimiters

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=ID_and_Class_Name_Delimiters#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.

/\* Not recommended: does not separate the words “demo” and “image” \*/ .demoimage {} /\* Not recommended: uses underscore instead of hyphen \*/ .error\_status {}

/\* Recommended \*/ #video-id {} .ads-sample {}

# Hacks

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Hacks#Hacks)▽

Avoid user agent detection as well as CSS “hacks”—try a different approach first.

It’s tempting to address styling differences over user agent detection or special CSS filters, workarounds, and hacks. Both approaches should be considered last resort in order to achieve and maintain an efficient and manageable code base. Put another way, giving detection and hacks a free pass will hurt projects in the long run as projects tend to take the way of least resistance. That is, allowing and making it easy to use detection and hacks means using detection and hacks more frequently—and more frequently is too frequently.

# Declaration Order

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Declaration_Order#Declaration_Order)▽

Alphabetize declarations.

Put declarations in alphabetical order in order to achieve consistent code in a way that is easy to remember and maintain.

Ignore vendor-specific prefixes for sorting purposes. However, multiple vendor-specific prefixes for a certain CSS property should be kept sorted (e.g. -moz prefix comes before -webkit).

background: fuchsia; border: 1px solid; -moz-border-radius: 4px; -webkit-border-radius: 4px; border-radius: 4px; color: black; text-align: center; text-indent: 2em;

# Block Content Indentation

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Block_Content_Indentation#Block_Content_Indentation)▽

Indent all block content.

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

@media screen, projection { html { background: #fff; color: #444; } }

# Declaration Stops

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Declaration_Stops#Declaration_Stops)▽

Use a semicolon after every declaration.

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

/\* Not recommended \*/ .test { display: block; height: 100px }

/\* Recommended \*/ .test { display: block; height: 100px; }

# Property Name Stops

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Property_Name_Stops#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.

/\* Not recommended \*/ h3 { font-weight:bold; }

/\* Recommended \*/ h3 { font-weight: bold; }

# Declaration Block Separation

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Declaration_Block_Separation#Declaration_Block_Separation)▽

Use a space between the last selector and the declaration block.

Always use a single space between the last selector and the opening brace that begins the [declaration block](http://www.w3.org/TR/CSS21/syndata.html#rule-sets).

The opening brace should be on the same line as the last selector in a given rule.

/\* Not recommended: missing space \*/ #video{ margin-top: 1em; } /\* Not recommended: unnecessary line break \*/ #video { margin-top: 1em; }

/\* Recommended \*/ #video { margin-top: 1em; }

# Selector and Declaration Separation

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Selector_and_Declaration_Separation#Selector_and_Declaration_Separation)▽

Separate selectors and declarations by new lines.

Always start a new line for each selector and declaration.

/\* Not recommended \*/ a:focus, a:active { position: relative; top: 1px; }

/\* Recommended \*/ h1, h2, h3 { font-weight: normal; line-height: 1.2; }

# Rule Separation

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Rule_Separation#Rule_Separation)▽

Separate rules by new lines.

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

html { background: #fff; } body { margin: auto; width: 50%; }

# CSS Quotation Marks

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=CSS_Quotation_Marks#CSS_Quotation_Marks)▽

Use single quotation marks for attribute selectors and property values.

Use single ('') rather than double ("") quotation marks for attribute selectors or 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](http://www.w3.org/TR/CSS21/syndata.html#charset).

/\* Not recommended \*/ @import url("//www.google.com/css/maia.css"); html { font-family: "open sans", arial, sans-serif; }

/\* Recommended \*/ @import url(//www.google.com/css/maia.css); html { font-family: 'open sans', arial, sans-serif; }

# CSS Meta Rules

## Section Comments

[link](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=Section_Comments#Section_Comments)▽

Group sections by a section comment (optional).

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

/\* Header \*/ #adw-header {} /\* Footer \*/ #adw-footer {} /\* Gallery \*/ .adw-gallery {}

# CSS formatting guidelines

Last updated February 7, 2014. Created by*[JohnAlbin](https://drupal.org/user/32095" \o "View user profile.)*on January 14, 2013.Edited by*[LewisNyman](https://drupal.org/user/751964" \o "View user profile.)*,*[hmmdinger](https://drupal.org/user/234516" \o "View user profile.)*,*[hass](https://drupal.org/user/85918" \o "View user profile.)*,*[echoz](https://drupal.org/user/38665" \o "View user profile.)*.[*Log in to edit this page*](https://drupal.org/user?destination=node/1887862).

## Table of Contents

[Whitespace](https://drupal.org/node/1887862#whitespace)  
[Comments](https://drupal.org/node/1887862#comments)  
[Format](https://drupal.org/node/1887862#format)  
[Miscellaneous](https://drupal.org/node/1887862#miscellaneous)  
[Practical Example](https://drupal.org/node/1887862#example)

### Terminology

For those unfamiliar with [CSS terminology](http://www.w3.org/TR/CSS2/syndata.html#statements), these are the concise terms used in these standards.

selector {  
  property: value;  
}

A **rule set** (also called a **rule**) consists of a selector followed by a declaration block.

The **selector** consists of everything up to (but not including) the first left curly brace ({).

A **declaration block** starts with a left curly brace ({) and ends with the matching right curly brace (}). In between there must be a list of zero or more semicolon-separated (;) declarations.

A **declaration** consists of a **property name**, followed by a colon (:), followed by a **value**.

## Whitespace

### Indentation

Using tabs for indentation leads to inconsistent display of the source code, since many text editors and most text viewers (like web browsers) cannot have their “tab size” configured.

For lines needing indenting, use 2 spaces for each level of indentation, the same standard as Drupal’s PHP and JavaScript code.

* Declarations (property/value pairs) should be indented one level relative to their selector.
* Rulesets within a media block or a media query should be indented one level relative to the media statement.
* Comments should be indented the same amount as the declaration or ruleset they describe.

@media print {  
  /\* This line is indented with 2 spaces, 2 spaces x 1 level of indentation. \*/  
  .example {  
    /\* This line is indented with 4 spaces, 2 spaces x 2 levels of indentation. \*/  
    padding: 0;  
  }  
}

### Blank lines

* In general, do NOT separate each ruleset by a blank line.
* If a ruleset has a proceeding Doxygen-style or single-line-style comment that describes it, place a blank line before the comment.
* If two rulesets have no interleaving blank line, they must be logically related. If they are not logically related to each other, add a blank line and a comment describing the second ruleset.

/\* A comment describing the ruleset. \*/  
.selector-1,  
.selector-2,  
.selector-3[type="text"] {  
  -webkit-box-sizing: border-box;  
     -moz-box-sizing: border-box;  
          box-sizing: border-box;  
  display: block;  
  margin: 0;  
  font-family: Times, "Times New Roman", sans-serif;  
  color: #333;  
  background: #fff;  
  background: linear-gradient(#fff, rgba(0, 0, 0, 0.8));  
}  
/\*\*  
\* A longer comment describing this ruleset. Note  
\* the blank line before the docblock.  
\*/  
.selector-4,  
.selector-5 {  
  padding: 10px;  
}  
/\* This logical grouping of rulesets has no interleaving blank lines. \*/  
.profile {  
  margin: 1em 0;  
}  
.profile\_\_picture {  
  float: right; /\* LTR \*/  
}

### Line endings

There MUST NOT be any whitespace (spaces or tabs) at the end of lines. This means blank lines should also not contain any spaces or tabs. Inconsistent trailing whitespace can add lines to diffs/patches and makes changes harder to notice.

All text files should end with a single blank line. This makes git commits easier to read since it's clearer what is being changed when lines are added to the end of a file and it avoids the verbose \ No newline at end of file warning in patches.

Files should be formatted with Unix line endings (a newline character, denoted as \n or LF), which is also the default in Mac OS X. Do not use Windows line endings (a carriage return plus a newline, denoted as \r\n or CRLF).

Tip: configure your editor to “show invisibles”. This will allow you to eliminate end-of-line whitespace, eliminate unintended blank-line whitespace, and avoid polluting commits.

Drupal 8 includes an [EditorConfig](http://editorconfig.org/) file in [its root directory](http://drupalcode.org/project/drupal.git/tree/refs/heads/8.x) to help maintain these whitespace conventions.

## Comments

Well commented code is extremely important. Take time to describe components, how they work, their limitations, and the way they are constructed. Don't leave others guessing as to the purpose of uncommon or non-obvious code.

To stay consistent with the rest of Drupal's code base, we borrow some of the CSS comment styles from the [Doxygen and comment formatting conventions](http://drupal.org/node/1354) for PHP files.

### File comments

Each file should start with a comment describing what the file does. For example:

/\*\*  
\* @file  
\* Short description describing the file.  
\*  
\* The first sentence of the long description starts here and continues on this  
\* line for a while finally concluding here at the end of this paragraph.  
\*/

Note that a blank line should follow a @file documentation block. And keep line-lengths to 80 columns, when possible. For more information, see the [PHP file comment standards](http://drupal.org/node/1354#files).

### Multi-line comments

When describing a ruleset or set of rulesets, any comment that requires 2 or more lines (wrapped to 80 characters) must follow the Doxygen comment style (also called a “docblock”).

/\*\*  
\* Short description using Doxygen-style comment format.  
\*  
\* The first sentence of the long description starts here and continues on this  
\* line for a while finally concluding here at the end of this paragraph.  
\*  
\* The long description is ideal for more detailed explanations and  
\* documentation. It can include example HTML, URLs, or any other information  
\* that is deemed necessary or useful.  
\*/  
.example-rule {  
}

Place the comment on the line immediately above the ruleset (or rulesets) it describes. Place a blank line before the docblock comment. See the [Doxygen and comment formatting conventions](http://drupal.org/node/1354)for more info.

### Single-line comments

When describing a property or ruleset, any comment that can be written inside the 80 character line length limit can use a simple CSS comment style.

.example {  
  /\* Override the default margins. \*/  
  margin: 0;  
}  
/\* This is a variant of the .example component \*/  
.example--item {  
  display: inline;  
}

Place the comment on the line immediately above the property or ruleset it describes. The comment should be indented the same amount as the property or ruleset it describes.

If the comment is describing a ruleset, place a blank line before the comment.

### Styling for Right-To-Left Languages

It is common for RTL language websites to have their designs flipped in the left/right direction. For direction specific property/values, add the comment /\* LTR \*/ on the same line preceded by a single space. In [Drupal 6 and 7](https://drupal.org/node/222782), the inclusion of a separate RTL stylesheet is automated. In Drupal 8, follow with an additional ruleset containing the inverse property/values, beginning with the attribute selector [dir="rtl"].

###### Example Rulesets for Drupal 6 and Drupal 7

[example.css]  
.example-rule {  
  float: left; /\* LTR \*/  
  margin-right: 1.5em; /\* LTR \*/  
  padding: 0 0.25em;  
}

[example-rtl.css]  
.example-rule {  
  float: right;  
  margin-left: 1.5em;  
  margin-right: 0;  
}

###### Example Rulesets for Drupal 8

[example.css]  
.example-rule {  
  float: left; /\* LTR \*/  
  margin-right: 1.5em; /\* LTR \*/  
  padding: 0 0.25em;  
}  
[dir="rtl"] .example-rule {  
  float: right;  
  margin-left: 1.5em;  
  margin-right: 0;  
}

* when you use the keywords, 'left' or 'right' in a property, e.g. float: left;
* where you use unequal margin, padding or borders on the sides of a box, e.g. margin-left: 1em; or padding: 0 0 0 2em;
* where you specify the direction of the language, e.g. direction: ltr;

## Format

Our CSS formatting ensures the code is easy to read, easy to clearly comment, minimizes the chance of accidentally introducing errors, and results in useful Git diffs and blames.

### Rulesets

* Use one selector per line when a ruleset has a group of selectors separated by commas.
* The opening brace ({) of a ruleset’s declaration block should be on the same line as the selector (or the same line as the last selector in a group of selectors.) The opening brace should include a single space before it.
* Place the closing brace (}) of a ruleset in the same column as the first character in the selector of the ruleset.
* Include one declaration per line in a declaration block.
* Each declaration should be indented one level relative to its selector.

###### Example Ruleset

.selector-alpha,  
.selector-beta {  
  counter-reset: section;  
  text-transform: small-caps;  
}

### Properties

* In a declaration, the property name should be immediately followed by a colon, then a single space, and then the property’s value.
* Include a semi-colon at the end of all declarations, including the last declaration in a declaration block.
* When hex values are used for colors, use lowercase and, if possible, the shorthand syntax, e.g. #aaa. Colors may be expressed with any valid CSS value, such as hex value, color keyword, rgb() or rgba(). Note that IE8 does not support all color syntaxes and will require a fallback value.
* For property values that require quotes, use double quotes instead of single quotes, e.g. font-family: "Arial Black", Arial, sans-serif; and content: " ";.
* Quote attribute values in selectors, e.g. input[type="checkbox"].
* Where allowed, avoid specifying units for zero-values, e.g. use margin: 0; instead of margin: 0px;.
* Include a space after each comma in comma-separated property or function values.
* Do not use spaces around the parentheses in a function, e.g. color: rgba(0, 0, 0, 0.8);

###### Example Properties

|  |  |
| --- | --- |
| display: block; | Basic syntax |
| color: #fff color: #df7dcf | Use shorthand syntax for hexadecimal colors when possible Always use lowercase |
| font-family: "Frutiger Ultra" | Use double quotes instead of single quotes |
| text-shadow: 0 0 2px #ddd | Do not attach units to zero-values |
| color: rgba(0, 136, 18, 0.8) | Spaces MUST follow commas in property or function values |

### Declaration order

The declarations in a ruleset should be ordered so that the purpose of the declaration block is most obvious. Clarity should be the guiding principle. We can help to achieve this goal by placing structurally important properties before others: positioning, box model, then other properties.

1. Positioning properties include: position, float, clear, top, right, bottom, left, direction, and z-index.
2. Box model properties include: display, [(max|min)-]height, [(max|min)-]width, margin, padding, border and their various longhand forms (margin-top, etc.) Plus box-sizing.
3. Other declarations.

Within each of the above groups, properties can be grouped alphabetically or grouped with like properties next to each other, e.g. putting font and text properties next to each other. Drupal’s coding standards are purposefully vague here because there is no consensus on this issue (as of 2013), but we respect each other’s abilities and preferences.

Vendor prefixed properties should be directly before their non-prefixed version. This allows the official version of the property to override any inconsistencies in the vendor-prefixed versions once those browsers implement the official property. If browser bugs or cross-browser issues necessitate any deviation from this ordering, it should be clearly documented.

Again, the order of properties are meant to reinforce the purpose of the ruleset. As such, **it is much more important to add comments to the ruleset then to worry about property ordering.**

.selector {  
  /\* Positioning declarations \*/  
  position: absolute;  
  top: 0;  
  left: 0;  
  z-index: 10;  
  /\* Box model declarations \*/  
  display: inline-block;  
  width: 100px;  
  padding: 10px;  
  border: 1px solid #333;  
  /\* Other declarations \*/  
  background: #000;  
  color: #fff;  
  font-family: sans-serif;  
  font-size: 1.1em;  
}

Tools like [CSScomb](http://csscomb.com/) may help with automating the order of properties.

### Exceptions and slight deviations

Large blocks of single declarations can use a slightly different, single-line format. In this case, a space should be included after the opening brace and before the closing brace.

.selector-1 { width: 10%; }  
.selector-2 { width: 20%; }  
.selector-3 { width: 30%; }

Long, comma-separated property values—such as collections of gradients or shadows—can be arranged across multiple lines in an effort to improve readability and produce more useful diffs.

.selector {  
  background-image:  
    linear-gradient(#fff, #ccc),  
    linear-gradient(#f3c, #4ec);  
  box-shadow:  
    1px 1px 1px #000,  
    2px 2px 1px 1px #ccc inset;  
}

### Media Queries

Media queries should be written in the same style as ruleset. Any containing rulesets are indented by two spaces.

* One space between the media feature and the value.
* All values to be written in ems unless it is inappropriate.
* Add the pixel value in a comment directly after the the opening brace.

@media screen and (min-width: 28.125em) { /\* 450px \*/  
  #page {  
    margin-left: 1.25em;  
    margin-right: 1.25em;  
  }  
}

## Miscellaneous

### @charset statements

Character set statements (like @charset "UTF-8";) are only valid if they are at the very top of a CSS file. Since Drupal’s CSS aggregator combines multiple CSS files into one file, Drupal will strip all @charset statements so that the aggregated file remains valid CSS.

This means CSS files MUST NOT include any @charset statements. The default encoding for CSS files is UTF-8. Any CSS comments or content property values MUST be encoded with UTF-8.

## Practical example

An example of various conventions.

/\*\*  
\* @file  
\* Layouts for this theme.  
\*/  
/\*\*  
\* Column layout with horizontal scroll.  
\*  
\* This creates a single row of full-height, non-wrapping columns that can be  
\* browsed horizontally within their parent.  
\*  
\* Example HTML:  
\*  
\* <div class="grid">  
\*   <div class="cell cell-3"></div>  
\*   <div class="cell cell-3"></div>  
\*   <div class="cell cell-3"></div>  
\* </div>  
\*/  
/\*\*  
\* Grid container  
\*  
\* Must only contain '.cell' children.  
\*/  
.grid {  
  height: 100%;  
  /\* Remove inter-cell whitespace \*/  
  font-size: 0;  
  /\* Prevent inline-block cells wrapping \*/  
  white-space: nowrap;  
}  
/\*\*  
\* Grid cells  
\*  
\* No explicit width by default. Extend with '.cell-n' classes.  
\*/  
.cell {  
  position: relative;  
  display: inline-block;  
  overflow: hidden;  
  box-sizing: border-box;  
  height: 100%;  
  /\* Set the inter-cell spacing \*/  
  padding: 0 10px;  
  border: 2px solid #333;  
  vertical-align: top;  
  /\* Reset white-space \*/  
  white-space: normal;  
  /\* Reset font-size \*/  
  font-size: 16px;  
}  
/\* Cell states \*/  
.cell.is-animating {  
  background-color: #fffdec;  
}  
/\* Cell dimensions \*/  
.cell-1 { width: 10%; }  
.cell-2 { width: 20%; }  
.cell-3 { width: 30%; }  
.cell-4 { width: 40%; }  
.cell-5 { width: 50%; }  
/\* Cell modifiers \*/  
.cell--detail,  
.cell--important {  
  border-width: 4px;  
}

References

<http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml#CSS_Style_Rules>

<http://make.wordpress.org/core/handbook/coding-standards/css/>

<https://drupal.org/node/1887862>

References

<http://make.wordpress.org/core/handbook/coding-standards/javascript/>

<http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml>

<https://github.com/rwaldron/idiomatic.js/>