WordPress Coding Standard For Developers

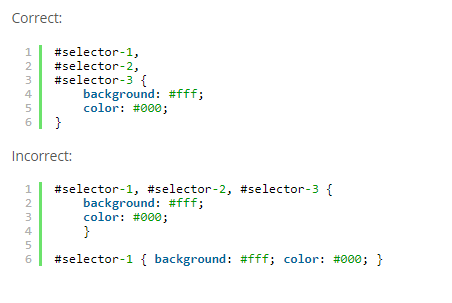
In this WordPress document the standards are divided into four main used languages that one use while creating a WordPress website.

CSS Coding Standards

1. Structure:

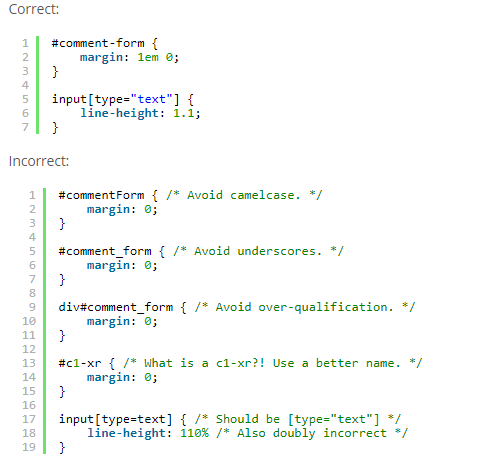
Structuring of a stylesheet is important to have a clear understanding of the flow of the document.

* 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 indentation and an ending semicolon.
* The closing brace should be flush left, using the same level of indentation as the opening selector.

For example:

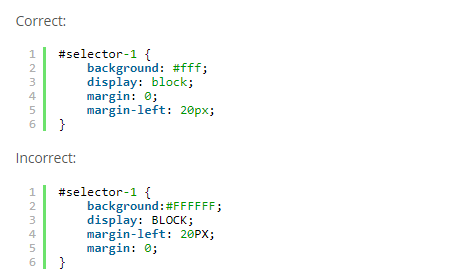
1. Selectors:

* For file names conventions, Use lowercase and separate words with hyphens when naming selectors. Avoid camel case and underscores.
* Use human readable selectors that describe what elements they style.
* Attributes selectors should use double quotes around values.
* Avoid over-qualified selectors, div.container can simple be stated as .container.

For example:

1. Properties:

* 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 (-moz-,-webkit-).
* Use hex code for colors, or rgba () if opacity is needed.
* Avoid RGB format and uppercase.
* Shorten values when possible #fff instead of #ffffff.
* Use shorthand for background, border, font, list-style, margin and padding values as much as possible.(shorthand references https://codex.wordpress.org/CSS\_Shorthand)

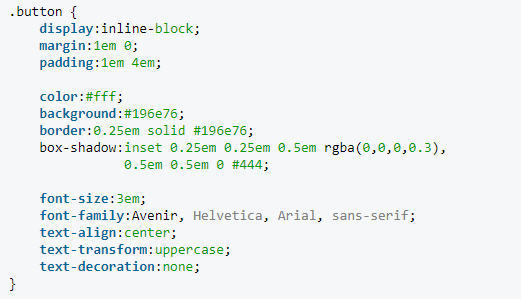
For example:

1. Property Ordering:

The baseline for ordering is:

* Layout Properties (position, float, clear, display)
* Box Model Properties (width, height, margin, padding)
* Visual Properties (color, background, border, box-shadow)
* Typography Properties (font-size, font-family, text-align, text-transform)
* Misc Properties or other (cursor, overflow, z-index).

For example:

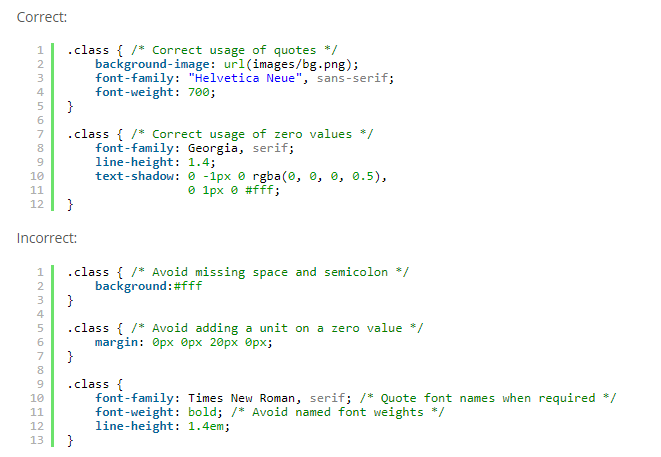


1. Vendor Prefixes :

Use Autoprefixer (https://github.com/postcss/autoprefixer) as a pre-commit tool to easily manage unnecessary browser prefixes.

1. Values:

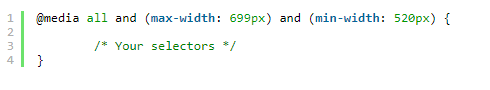
* Space before the value, after the colon.
* Always end in a semicolon
* Use double quotes rather than single quotes whenever needed. For eg: font-family: “Helvetica Neue”.
* Font weights should be defined using numeric values (eg. 400 instead of normal, 700 rather than bold).
* 0 values should not have 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.
* 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.



1. Media Queries:

* Keep media queries grouped by media at the bottom of the stylesheet.
* Rules sets for media queries should be indented one level in.

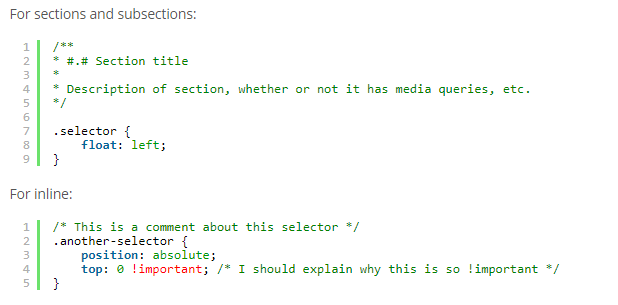
For example:



1. Commenting:

* Long comments should manually break the line length at 80 characters.
* A table of contents should be utilized for longer stylesheet, especially those that are highly sectioned.
* 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 example:



1. Best Practices:

* If you are attempting to fix an issue, attempt to remove code before adding more.
* Magic number are unlucky. These are numbers that are used as quick fixes on a one-off basis. Ex: .box {margin-box: 37px}.
* DOM changes over time, target the element you want to use through its parents. For eg: .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 element (for eg: images). Otherwise use line-height for more flexibility.
* Do not restate default property and value combinations (for eg: display: block; on block-level elements).

HTML Coding Standards:

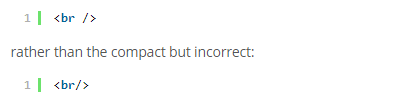
1. Validation:

All HTML pages should be verified against [the W3C validator](http://validator.w3.org/) to ensure that the markup is well formed. This in and of itself is not directly indicative of good code, but it helps to weed out problems that are able to be tested via automation.

1. Self-closing Elements:

* Tags that are self-closed, the forward slash should have exactly one space preceding it.

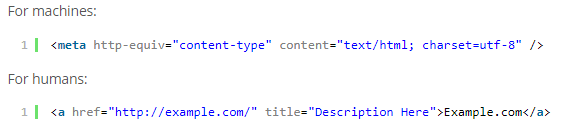
For example:



1. Attributes and Tags:

* All tags and attributes must be written in lowercase.
* To interpret by machine then attributes value should be lowercase.
* To interpret by human, then proper title capitalization should be followed.

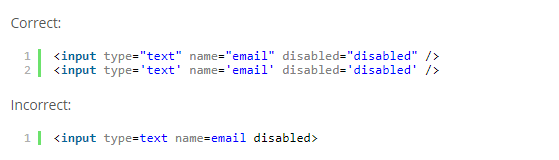
For example:



1. Quotes:

* All attributes must have a value .and must use double or single quotes.

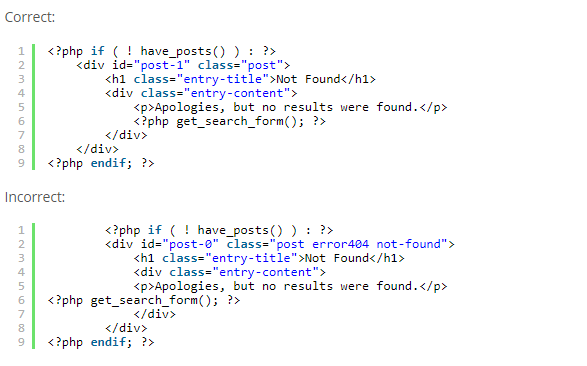
For example:



1. Indentation:

* Always use tabs and not spaces.
* When mixing PHP and HTML together, indent PHP blocks to match the surrounding HTML code. Closing PHP blocks should match the same indentation level as the opening block.

For example:



PHP Coding Standards:

1. Single and Double Quotes:

* Use single and double quotes when appropriate.
* If not evaluating anything in the string, use single quotes.
* Never escape quotes in a string, rather than that alternate your quoting style.

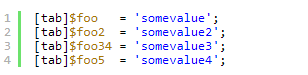
For example:

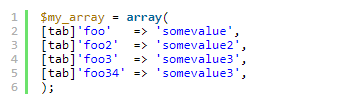
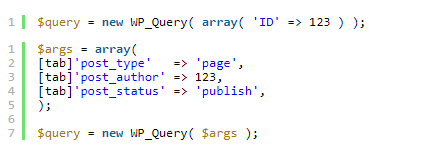


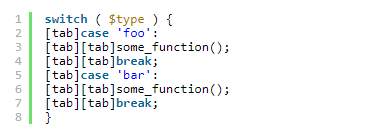
1. Indentation:

* Use real tabs and not spaces, as this allows the most flexibility across clients.

For example:

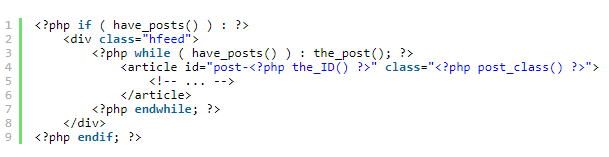




1. Brace Style:

* PHP offers an Alternative syntax for some of its control structures; namely if, while, for, foreach, and switch.
* The basic form of the alternate syntax is to change the opening brace to a colon (:) and the closing brace to endif; endwhile;, endfor;, endforeach;, or endsswitch;, respectively.
* Alternative syntax especially used where php code is embedded within HTML.



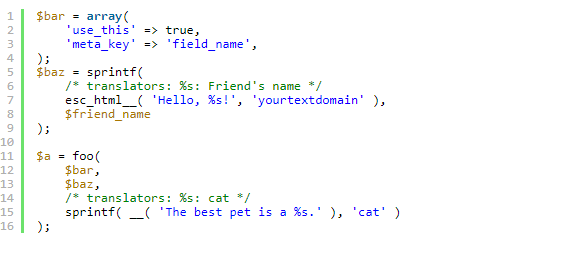
1. Use elseif, not else if:

* “else if” is not compatible with the colon syntax for “if|elseif” blocks. For this reason, use “elseif” for conditionals.

1. Multiline Function Calls:

* When splitting a function call over multiple lines, each parameter must be on a separate line.
* Single line inline comments can take up their own line.
* Each parameter must take up no more than a single line.
* Multiline parameter values must be assigned to a variable and then that variable and then that variable should be passed to the function call.

For example:



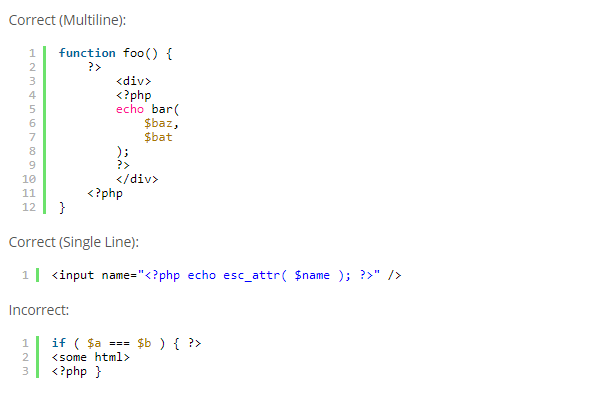
1. Regular Expressions:

* Use of Regular expressions are nothing more than a sequence or pattern of character itself. They provide the foundation for pattern-matching functionality.
* Using regular expression you can search a particular string inside another string, you can replace one string by another string and you can split a string into many chunks.

(<https://www.tutorialspoint.com/php/php_regular_expression.htm>)

1. Opening and Closing PHP Tags:

* When embedding multi-line PHP snippets within a HTML block, the PHP open and close tags must be on a line by themselves.



1. No Shorthand PHP Tags:

* Never use shorthand PHP start tags. Always use full php tags.

For example:



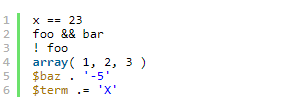
1. Remove Trailing Spaces:

* Remove trailing whitespace at the end of each line of code. Omitting the closing PHP tag at the end of a file is preferred. If you use the tag, make sure you remove trailing whitespace.

1. Space Usage:

* Always put spaces after commas, and on both sides of logical, comparison, string and assignment operators.

For example:



* Put spaces on both sides of the opening and closing parenthesis of if, elseif, foreach, for and switch blocks.

For example:



* When defining a function.

For example:



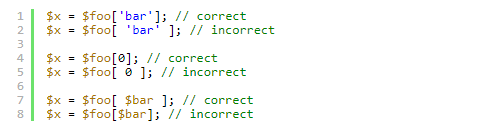
* When calling a function.
* When performing logical comparison.



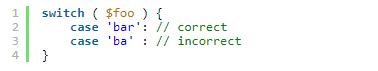
* [Type casts](http://www.php.net/manual/en/language.types.type-juggling.php#language.types.typecasting) must be lowercase. Always prefer the short form of type casts, (int) instead of (integer) and (bool) rather than (boolean). For float casts use (float).



* When referring to array items, only include a space around the index if it is a variable, for example:



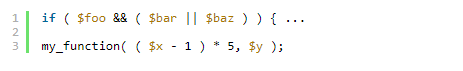
* In a switch block, there must be no space before the colon for a case statement.



* Similarly, there should be no space before the colon on return type declarations.



* Unless otherwise specified, parentheses should have spaces inside of them



1. Formatting SQL Statement:

* To format long SQL statement you may break it into several lines.
* Always capitalize the SQL parts of the statements like CREATE, UPDATE, WHERE.
* SQL slash escaping should be done as close to the time of the query as possible, preferably by using $wpdb->prepare ().
* $wpdb->prepare() is a method that handles escaping, quoting and int-casting for SQL queries.

For example:

$var = ‘dangerous’; // raw data that may not need to be escaped

$id = some\_foo\_number (); //data we expect to be an integer, but we’re not certain

$wpdb->query ($wpdb->prepare ( "UPDATE $wpdb->posts SET post\_title = %s WHERE ID = %d", $var, $id) );

1. Database Queries:

* Avoid touching the database directly. If there is a defined function that can get the data you need, use it.
* Database abstraction (using functions instead of queries) helps keep your code forward-compatible and, in cases where results are cached in memory, it can be many times faster.

1. Naming Conventions:

* Use lowercase letters in variable, action/filter, and function names (never camelCase).
* Separate words via underscores.
* Don’t abbreviate variable names unnecessarily; let the code be unambiguous and self-documenting.

For example:



* Class names should use capitalized words separated by underscores. Any acronyms should be all upper case.



* Constants should be in all upper-case with underscores separating words:



* Files should be named descriptively using lowercase letters. Hyphens should separate words.



* Class file names should be based on the class name with class- prepended and the underscores in the class name replaced with hyphens, for example WP\_Error becomes:



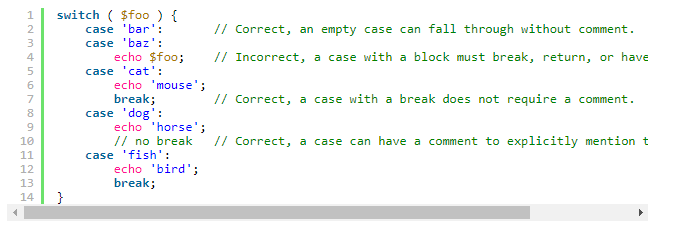
* Files containing template tags in wp-includes should have -template appended to the end of the name so that they are obvious.



1. Clever Code:
   * Readability of code is important



* + In a switch statement, it’s okay to have multiple empty cases fall through to a common block. If a case contains a block, then falls through to the next block, however, this must be explicitly commented.

For example:

* + Note: goto statement must never be used.
  + Create\_function() and eval() is deprecated in php 7.2. Both must not be used.

1. Error Control Operator:

* PHP supports one error control operator, the at sign @. When prepended to an expression in PHP, any error messages that might be generated by that expressions will be ignored.
* If you have set a custom error handler function with set\_error\_handler () then it will still get called, but this custom error handler can call error\_reporting () which will return 0 when the call that triggered the error was preceded by an @.