|  |
| --- |
|  |
|  |
|  |
| CA/CST  HTML 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/>

# Validation

All HTML pages should be verified against the W3C validator 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. It is no substitute for manual code review. (For other validators, see HTML Validation in the Codex.) Code Example

# Self-closing Elements

All tags must be properly closed. For tags that can wrap nodes such as text or other elements, termination is a trivial enough task. For tags that are self-closing, the forward slash should have exactly one space preceding it:

|  |  |
| --- | --- |
|  | <br /> |

Rather than the compact but incorrect:

|  |  |
| --- | --- |
|  | <br/> |

# Attributes and Tags

All tags and attributes must be written in lowercase. Additionally, attribute values should be lowercase when the purpose of the text therein is only to be interpreted by machines. For instances in which the data needs to be human readable, proper title capitalization should be followed.

For machines:

|  |  |
| --- | --- |
|  | <meta http-equiv="content-type" content="text/html; charset=utf-8" /> |

For humans:

|  |  |
| --- | --- |
|  | <a href="http://example.com/" title="Description Here">Example.com</a> |

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

# Quotes

According to the W3C specifications for XHTML, all attributes must have a value, and must use double- or single-quotes ([source](http://www.w3.org/TR/xhtml1/#h-4.4)). The following are examples of proper and improper usage of quotes and attribute/value pairs.

Correct:

|  |  |
| --- | --- |
| 1  2 | <input type="text" name="email" disabled="disabled" />  <input type='text' name='email' disabled='disabled' /> |

Incorrect:

|  |  |
| --- | --- |
| 1 | <input type=text name=email disabled> |

In HTML, attributes do not all have to have values, and attribute values do not always have to be quoted. While all of the examples above are valid HTML, failing to quote attributes can lead to security vulnerabilities. Always quote attributes.

# Indentation

As with PHP, HTML indentation should always reflect logical structure. 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.

Correct:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php if ( ! have\_posts() ) : ?>      <div id="post-1" class="post">          <h1 class="entry-title">Not Found</h1>          <div class="entry-content">              <p>Apologies, but no results were found.</p>              <?php get\_search\_form(); ?>          </div>      </div>  <?php endif; ?> |

Incorrect:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php if ( ! have\_posts() ) : ?>          <div id="post-0" class="post error404 not-found">              <h1 class="entry-title">Not Found</h1>              <div class="entry-content">              <p>Apologies, but no results were found.</p>  <?php get\_search\_form(); ?>              </div>          </div>  <?php endif; ?> |

# Protocol

Omit the protocol from embedded resources.

Omit the protocol portion (http:, https:) from URLs pointing to images and other media files, style sheets, and scripts unless the respective files are not available over both protocols.

Omitting the protocol—which makes the URL relative—prevents mixed content issues and results in minor file size savings.

<!-- Not recommended --> <script src="http://www.google.com/js/gweb/analytics/autotrack.js"></script>

<!-- Recommended --> <script src="//www.google.com/js/gweb/analytics/autotrack.js"></script>

/\* Not recommended \*/ .example { background: url(http://www.google.com/images/example); }

/\* Recommended \*/ .example { background: url(//www.google.com/images/example); }

# Indentation

Indent by 2 spaces at a time.

Don’t use tabs or mix tabs and spaces for indentation.

<ul> <li>Fantastic <li>Great </ul>

.example { color: blue; }

# Capitalization

Use only lowercase.

All code has to be lowercase: This applies to HTML element names, attributes, attribute values (unless text/CDATA), CSS selectors, properties, and property values (with the exception of strings).

<!-- Not recommended --> <A HREF="/">Home</A>

<!-- Recommended --> <img src="google.png" alt="Google">

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

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

# Trailing Whitespace

Remove trailing white spaces.

Trailing white spaces are unnecessary and can complicate diffs.

<!-- Not recommended --> <p>What?\_

<!-- Recommended --> <p>Yes please.

# Encoding

Use UTF-8 (no BOM).

Make sure your editor uses UTF-8 as character encoding, without a byte order mark.

Specify the encoding in HTML templates and documents via <meta charset="utf-8">. Do not specify the encoding of style sheets as these assume UTF-8.

(More on encodings and when and how to specify them can be found in [Handling character encodings in HTML and CSS](http://www.w3.org/International/tutorials/tutorial-char-enc/).)

# Comments

Explain code as needed, where possible.

Use comments to explain code: What does it cover, what purpose does it serve, why is respective solution used or preferred?

(This item is optional as it is not deemed a realistic expectation to always demand fully documented code. Mileage may vary heavily for HTML and CSS code and depends on the project’s complexity.)

# Action Items

Mark todos and action items with TODO. All TODO items should be removed from final delivery to IV&V (and production)

{# TODO(john.doe): revisit centering #} <center>Test</center>

<!-- TODO: remove optional tags --> <ul> <li>Apples</li> <li>Oranges</li> </ul>

# Document Type

Use HTML5.

HTML5 (HTML syntax) is preferred for all HTML documents: <!DOCTYPE html>.

(It’s recommended to use HTML, as text/html. Do not use XHTML. XHTML, as [application/xhtml+xml](http://hixie.ch/advocacy/xhtml), lacks both browser and infrastructure support and offers less room for optimization than HTML.)

# HTML Validity

Use valid HTML where possible.

Use valid HTML code unless that is not possible due to otherwise unattainable performance goals regarding file size.

Use tools such as the [W3C HTML validator](http://validator.w3.org/nu/) to test.

Using valid HTML is a measurable baseline quality attribute that contributes to learning about technical requirements and constraints, and that ensures proper HTML usage.

<!-- Not recommended --> <title>Test</title> <article>This is only a test.

<!-- Recommended --> <!DOCTYPE html> <meta charset="utf-8"> <title>Test</title> <article>This is only a test.</article>

# Semantics

Use HTML according to its purpose.

Use elements (sometimes incorrectly called “tags”) for what they have been created for. For example, use heading elements for headings, p elements for paragraphs, a elements for anchors, etc.

Using HTML according to its purpose is important for accessibility, reuse, and code efficiency reasons.

<!-- Not recommended --> <div onclick="goToRecommendations();">All recommendations</div>

<!-- Recommended --> <a href="recommendations/">All recommendations</a>

# Multimedia Fallback

Provide alternative contents for multimedia.

For multimedia, such as images, videos, animated objects via canvas, make sure to offer alternative access. For images that means use of meaningful alternative text (alt) and for video and audio transcripts and captions, if available.

Providing alternative contents is important for accessibility reasons: A blind user has few cues to tell what an image is about without @alt, and other users may have no way of understanding what video or audio contents are about either.

(For images whose alt attributes would introduce redundancy, and for images whose purpose is purely decorative which you cannot immediately use CSS for, use no alternative text, as in alt="".)

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

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

# 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.

In addition, keep the contact area as small as possible by linking as few style sheets and scripts as possible from documents and templates.

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.

<!-- Not recommended --> <!DOCTYPE html> <title>HTML sucks</title> <link rel="stylesheet" href="base.css" media="screen"> <link rel="stylesheet" href="grid.css" media="screen"> <link rel="stylesheet" href="print.css" media="print"> <h1 style="font-size: 1em;">HTML sucks</h1> <p>I’ve read about this on a few sites but now I’m sure: <u>HTML is stupid!!1</u> <center>I can’t believe there’s no way to control the styling of my website without doing everything all over again!</center>

<!-- Recommended --> <!DOCTYPE html> <title>My first CSS-only redesign</title> <link rel="stylesheet" href="default.css"> <h1>My first CSS-only redesign</h1> <p>I’ve read about this on a few sites but today I’m actually doing it: separating concerns and avoiding anything in the HTML of my website that is presentational. <p>It’s awesome!

# Optional Tags

Omit optional tags (optional).

For file size optimization and scanability purposes, consider omitting optional tags. The [HTML5 specification](http://www.whatwg.org/specs/web-apps/current-work/multipage/syntax.html#syntax-tag-omission) defines what tags can be omitted.

(This approach may require a grace period to be established as a wider guideline as it’s significantly different from what web developers are typically taught. For consistency and simplicity reasons it’s best served omitting all optional tags, not just a selection.)

<!-- Not recommended --> <!DOCTYPE html> <html> <head> <title>Spending money, spending bytes</title> </head> <body> <p>Sic.</p> </body> </html>

<!-- Recommended --> <!DOCTYPE html> <title>Saving money, saving bytes</title> <p>Qed.

# type Attributes

Omit type attributes for style sheets and scripts.

Do not use type attributes for style sheets (unless not using CSS) and scripts (unless not using JavaScript).

Specifying type attributes in these contexts is not necessary as HTML5 implies [text/css](http://www.whatwg.org/specs/web-apps/current-work/multipage/semantics.html#attr-style-type) and [text/javascript](http://www.whatwg.org/specs/web-apps/current-work/multipage/scripting-1.html#attr-script-type) as defaults. This can be safely done even for older browsers.

<!-- Not recommended --> <link rel="stylesheet" href="//www.google.com/css/maia.css" type="text/css">

<!-- Recommended --> <link rel="stylesheet" href="//www.google.com/css/maia.css">

<!-- Not recommended --> <script src="//www.google.com/js/gweb/analytics/autotrack.js" type="text/javascript"></script>

<!-- Recommended --> <script src="//www.google.com/js/gweb/analytics/autotrack.js"></script>

# General Formatting

Use a new line for every block, list, or table element, and indent every such child element.

Independent of the styling of an element (as CSS allows elements to assume a different role per display property), put every block, list, or table element on a new line.

Also, indent them if they are child elements of a block, list, or table element.

(If you run into issues around whitespace between list items it’s acceptable to put all li elements in one line. A linter is encouraged to throw a warning instead of an error.)

<blockquote> <p><em>Space</em>, the final frontier.</p> </blockquote>

<ul> <li>Moe <li>Larry <li>Curly </ul>

<table> <thead> <tr> <th scope="col">Income <th scope="col">Taxes <tbody> <tr> <td>$ 5.00 <td>$ 4.50 </table>

# HTML Quotation Marks

When quoting attributes values, use double quotation marks.

Use double ("") rather than single quotation marks ('') around attribute values.

<!-- Not recommended --> <a class='maia-button maia-button-secondary'>Sign in</a>

<!-- Recommended --> <a class="maia-button maia-button-secondary">Sign in</a>