Coding Style Guide

HTML

Use Lower Case Element Names

HTML5 allows mixing uppercase and lowercase letters in element names.
We recommend using lowercase element names because:

- Mixing uppercase and lowercase names is bad
- Developers normally use lowercase names (as in XHTML)
- Lowercase look cleaner
- Lowercase are easier to write

Good:

```
<section>

  <p>This is a paragraph.</p>

</section>
```

Close All HTML Elements

In HTML5, you don't have to close all elements (for example the <p> element).
We recommend closing all HTML elements.

Good:

```
<section>

  <p>This is a paragraph.</p>

  <p>This is a paragraph.</p>
```

```
</section>
```

Close Empty HTML Elements

In HTML5, it is optional to close empty elements.

Allowed:

```
<meta charset="utf-8">
```

Also Allowed:

```
<meta charset="utf-8" />
```

However, the closing slash (/) is REQUIRED in XHTML and XML.
If you expect XML software to access your page, it is a good idea to keep the closing slash!

Use Lower Case Attribute Names

HTML5 allows mixing uppercase and lowercase letters in attribute names.
We recommend using lowercase attribute names because:

- Mixing uppercase and lowercase names is bad
- Developers normally use lowercase names (as in XHTML)
- Lowercase look cleaner
- Lowercase are easier to write

Good:

```
<div class="menu">
```

Quote Attribute Values

HTML5 allows attribute values without quotes.
We recommend quoting attribute values because:

- Mixing uppercase and lowercase values is bad
- Quoted values are easier to read
- You MUST use quotes if the value contains spaces

Good:

```
<table class="striped">
```

Image Attributes

Always add the "alt" attribute to images. This attribute is important when the image for some reason cannot be displayed. Also, always define image width and height. It reduces flickering because the browser can reserve space for the image before loading.

Good:

```
<img src="html5.gif" alt="HTML5"
style="width:128px;height:128px">
```

Spaces and Equal Signs

HTML5 allows spaces around equal signs. But space-less is easier to read, and groups entities better together.

Good:

```
<link rel="stylesheet" href="styles.css">
```

Avoid Long Code Lines

When using an HTML editor, it is inconvenient to scroll right and left to read the HTML code.
Try to avoid code lines longer than 80 characters.

Blank Lines and Indentation

Do not add blank lines without a reason.
For readability, add blank lines to separate large or logical code blocks.
For readability, add two spaces of indentation. Do not use the tab key.
Do not use unnecessary blank lines and indentation. It is not necessary to indent every element:

Better:

```
<body>

<h1>Famous Cities</h1>

<h2>Tokyo</h2>

<p>Tokyo is the capital of Japan, the center of the Greater
Tokyo Area,

and the most populous metropolitan area in the world.

It is the seat of the Japanese government and the Imperial
Palace,

and the home of the Japanese Imperial Family.</p>
```

```
</body>
```

Table Example:

```
<table>
  <tr>
    <th>Name</th>
    <th>Description</th>
  </tr>
  <tr>
    <td>A</td>
    <td>Description of A</td>
  </tr>
  <tr>
    <td>B</td>
    <td>Description of B</td>
  </tr>
</table>
```

List Example:

```
<ol>
  <li>London</li>
```

```
  <li>Paris</li>

  <li>Tokyo</li>

</ol>
```

Omitting <html> and <body>?

In the HTML5 standard, the <html> tag and the <body> tag can be omitted. The following code will validate as HTML5:

Example

```
<!DOCTYPE html>

<head>

  <title>Page Title</title>

</head>



<h1>This is a heading</h1>

<p>This is a paragraph.</p>
```

We do not recommend omitting the <html> and <body> tags.

The <html> element is the document root. It is the recommended place for specifying the page language:

```
<!DOCTYPE html>

<html lang="en-US">
```

Declaring a language is important for accessibility applications (screen readers) and search engines.
Omitting <html> or <body> can crash DOM and XML software.
Omitting <body> can produce errors in older browsers (IE9).

HTML Comments

Short comments should be written on one line, like this:

```
<!-- This is a comment -->
```

Comments that spans more than one line, should be written like this:

```
<!--

  This is a long comment example. This is a long comment
example.

  This is a long comment example. This is a long comment
example.

-->
```

Long comments are easier to observe if they are indented two spaces.

Style Sheets

Use simple syntax for linking to style sheets (the type attribute is not necessary):

```
<link rel="stylesheet" href="styles.css">
```

Short rules can be written compressed, on one line, like this:

```
p.intro {font-family: Verdana; font-size: 16em;}
```

Long rules should be written over multiple lines:

```
body {
```

```
  background-color: lightgrey;

  font-family: "Arial Black", Helvetica, sans-serif;

  font-size: 16em;

  color: black;

}
```

- Place the opening bracket on the same line as the selector
- Use one space before the opening bracket
- Use two spaces of indentation
- Use semicolon after each property-value pair, including the last
- Only use quotes around values if the value contains spaces
- Place the closing bracket on a new line, without leading spaces
- Avoid lines over 80 characters

Loading JavaScript in HTML

Use simple syntax for loading external scripts (the type attribute is not necessary):

```
<script src="myscript.js">
```

Accessing HTML Elements with JavaScript

A consequence of using "untidy" HTML styles, might result in JavaScript errors. These two JavaScript statements will produce different results:

Example

```
var obj = getElementById("Demo")

var obj = getElementById("demo")
```

Javascript

Variable Names

All names start with a **letter**. (Variables and functions)

```
firstName = "John";

lastName = "Doe";

price = 19.90;

tax = 0.20;

fullPrice = price + (price * tax);
```

Spaces Around Operators

Always put spaces around operators (= + - * /), and after commas:

```
var x = y + z;

var values = ["Volvo", "Saab", "Fiat"];
```

Code Indentation

Always use tab for indentation of code blocks:

```
function toCelsius(fahrenheit) {

     return (5 / 9) * (fahrenheit - 32);

}
```

Statement Rules

General rules for simple statements:

- Always end a simple statement with a semicolon.

```
var values = ["Volvo", "Saab", "Fiat"];

var person =

{

     firstName: "John",

    lastName: "Doe",

     age: 50,

     eyeColor: "blue"

};
```

General rules for complex (compound) statements:

- Put the opening bracket after the first line.
- Put the closing bracket on a new line, without leading spaces.
- Do not end a complex statement with a semicolon.

```
function toCelsius(fahrenheit)

{
```

```
    return (5 / 9) * (fahrenheit - 32);
}
```

Loops:

```
for (i = 0; i < 5; i++)
{
     x += i;
}
```

Conditionals:

```
if (time < 20)
{
     greeting = "Good day";
}
else
{
    greeting = "Good evening";
}
```

Object Rules

General rules for object definitions:

- Place the opening bracket after the object name.
- Use colon plus one space between each property and its value.
- Use quotes around string values, not around numeric values.
- Do not add a comma after the last property-value pair.
- Place the closing bracket on a new line, without leading spaces.
- Always end an object definition with a semicolon.

Example

```
var person =

{

     firstName: "John",

    lastName: "Doe",

     age: 50,

     eyeColor: "blue"

};
```

Short objects can be written compressed, on one line, using spaces only between properties, like this:

```
var person = {firstName:"John", lastName:"Doe", age:50,
eyeColor:"blue"};
```

Line Length < 80

For readability, avoid lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it, is after an operator or a comma.

Example

```
document.getElementById("demo").innerHTML =

    "Hello Dolly."
```

Naming Conventions

Always use the same naming convention for all your code. For example:

- Variable and function names written as **camelCase**
- Global variables written in **UPPERCASE** (We don't, but it's quite common)
- Constants (like PI) written in **UPPERCASE**

Underscores:
Many programmers prefer to use underscores (date_of_birth), especially in SQL databases.
Underscores are often used in PHP documentation.
camelCase:
camelCase is used by JavaScript itself, by jQuery, and other JavaScript libraries.

Do not start names with a $ sign. It will put you in conflict with many JavaScript library names.

Loading JavaScript in HTML

Use simple syntax for loading external scripts (the type attribute is not necessary):

```
<script src="myscript.js"></script>
```

File Extensions

HTML files should have a **.html** extension (not **.htm**).
CSS files should have a **.css** extension.
JavaScript files should have a **.js** extension.

Use Lower Case File Names

Most web servers (Apache, Unix) are case sensitive about file names:
london.jpg cannot be accessed as London.jpg.
Other web servers (Microsoft, IIS) are not case sensitive:
london.jpg can be accessed as London.jpg or london.jpg.
If you use a mix of upper and lower case, you have to be extremely consistent.
If you move from a case insensitive, to a case sensitive server, even small errors can break your web site.
To avoid these problems, always use lower case file names (if possible).