



HTML & CSS Coding Style

This document sets the guidelines HTML/CSS markdown of MindTouch websites. These are not hard-and-fast rules, but rather guidelines which each developer should aim to follow. By following the same mentality for achieving effects, it will be easier for different people to maintain projects.

All MindTouch projects should follow [Yahoo's Graded Browser Support](#) table when testing for compatibility.

HTML Markdown

All pages should be declared against the XHTML Transitional or XHTML Strict doctype and should validate. The markup should never contain presentational elements (no tables for formatting, no inline styles), and should do its best to be semantically rigorous (content comes first, use semantic markup like ``).

There's a [great graphic floating around on the web that outlines clean HTML](#).

IDs should be used sparingly, and selector rules should be taken advantage of when possible. Unless necessary, try to keep IDs to delimit "sections" of the site. Following [common standardized names](#), we should use: **header**, **content**, **nav**, **sidebar**, and **footer** as IDs. Use classes as sparingly as possible to prevent tag soup:

Worse:

```
<div id="nav">
  <ul class="nav">
    <li class="nav"><a href="http://wiki.opengarden.org/home" class="home">Home</a></li>
    <li class="nav"><a href="http://wiki.opengarden.org/about"
class="about">About</a></li>
  </ul>
</div>
```

Better:

```
<div id="nav">
  <ul>
    <li class="home"><a href="http://wiki.opengarden.org/home">Home</a></li>
    <li class="about"><a href="http://wiki.opengarden.org/about">About</a></li>
  </ul>
</div>
```

Note that unless a second `` exists inside the **nav** section, there is no need to explicitly define classes on the elements underneath. Also notice that the class has moved onto the `` element - semantically, the list element itself is

specific to the home section and not just the link element. With the list element having the specific home, it also allows additional formatting around the <a> (without losing specificity).

CSS

All MindTouch projects should utilize [YUI's reset.css](#). The **advantages of resetting** are many for individual developers, but the main reason we're pursuing this is because each developer has their own way of "zeroing" out the designs, which is unacceptable for long-term maintenance. Zeroing the styles also gets developers in the habit of not designing for one browser, by forcing them to explicitly define styles.

Note that usage of reset.css will require you to define some common styles; we'll maintain a common MindTouch post-reset.css file on this page which will add back some default styling (example: font-weight: bold; to).

<link> vs. <style type="text/css">

When possible, store CSS files externally. This improves cacheability. Embedding style attributes into the HTML markup should be avoided - the one exception to this rule is using display: none; when creating UI elements which toggle on/off.

Naming conventions

Just like our file naming conventions, class & id names should be lowercased. If you want to add separation, hyphens are legal. On the UI, we are actively trying to scope ids with "deki-". Examples of this convention can be seen in tags and page alerts.

```
#deki-page-tags {  
}  
  
#deki-page-alerts {  
}
```

Formatting CSS

Curly braces should end on the same line as the selector classes. In the absence of tools like Firebug, we need an effective way of finding the closest selector sets that define styles, and using { as the delimiter helps for searching.

Worse:

```
#nav ul li.home a  
{  
  font-weight: bold;  
}
```

Better:

```
#nav ul li.home a {  
  font-weight: bold;  
}
```

Selector specificity

There's a lot of gray area when it comes to how specific one should get when defining selectors. As a general rule of thumb, if you're using elements, follow the full tree; for divs, use the closest div.

Example markup:

```
<div id="list">
  <div class="block">
    <div class="user">
      <ul>
        <li class="user"><a href="#">User Page</a></li>
        <li class="logout"><a href="#">Logout</a></li>
      </ul>
    </div>
  </div>
</div>
```

Selecting against the user page link:

Better:

```
#list ul li.user a { }
```

Worse:

```
#list ul div.block div.user li a { }
```

Unless `div.block` contains other elements which require more specificity, avoid using it as a part of the selectors.

Selector rules

Selector rules should try to remain as concise as possible. When relevant, they should use the closest id as an anchor.

Sample HTML:

```
<body>
<div id="nav">
  <ul>
    <li class="home"><a href="http://wiki.opengarden.org/home">Home</a></li>
    <li class="about"><a href="http://wiki.opengarden.org/about">About</a></li>
  </ul>
  <div class="block">
    Feel free to <a href="http://wiki.opengarden.org/contact">contact us for some more
information</a>.
  </div>
</div>

... (snip)
</body>
```

Worse:

```
body #nav ul li.home a {
  font-weight: bold;
}
```

Better:

```
#nav li.home a {
  font-weight: bold;
}
```

CSS Attributes

Use short-hand notation when possible, and rely on inheritance whenever possible. When forced to make a decision on inheritance vs. short-hand notation, rely on inheritance.

Worse:

```
body {
  font-size: 12px;
  font-family: Verdana, Sans-Serif;
}
#nav li.home a {
  font: bold 14px Verdana, Sans-Serif;
  padding-top: 4px;
  padding-right: 8px;
  color: #555555;
}
```

Better:

```
body {
  font: 12px Verdana, Sans-Serif;
}
#nav li.home a {
  font-weight: bold;
  font-size: 14px;
  padding: 4px 8px 0 0;
  color: #555;
}
```

Separate with linebreaks multiple attributes:

```
#nav li.home,
#nav li.about {
  font-weight: bold;
}
```

Browser Workarounds

When implementing browser-specific workarounds, they should be kept in separate files if possible. For IE browsers, you can selectively target them in your HTML:

```
<!--[if IE 6]><link rel="stylesheet" type="text/css" media="screen"
href="ie6hacks.css"/><![endif]-->
```

If hacks must be kept in the same CSS files, be sure you comment the workaround.

Browser-specific styles

Browser-specific styles are OK, if they are only for polishing. The same goes for advanced selectors not supported by all browsers (CSS2, CSS3). When possible, **use styles which work in all browsers before attempting to use specific-selectors**.

With our **current browser support**, CSS2 and CSS3 selectors are generally **not in use** and should be avoided when possible.

Position: absolute;

Should be avoided when doing layouts. The z-index should be reserved for Javascript functionality ... factoring in the z-index of a layout in JS is difficult and can cause erratic behavior.

Specific implementations

Forms

```
<form class="deki-form">
  <div class="field"></div>
  <div class="field"></div>
  <div class="submit"></div>
</form>
```

Use DekiForm to build input fields (DekiForm::singleInput()) and wrap those elements with "field" or "submit".

