COMP519
Web Programming

Autumn 2015

Cascading Style Sheets

Content vs. Presentation

Most HTML tags define content type, independent of how that content is to be presented.

There are a few obvious exceptions such as . . . for bold text and . . . for italicized text.

The trend in web development has been towards an increasing separation of the content of webpages from the presentation of them.

Style sheets allow us to maintain this separation, which facilitates easier maintenance of webpages, and for a consistent look across a collection of webpages.

Content vs. Presentation (cont.)

Style sheets associate presentation formats with HTML elements.

- CSS1: Developed in 1996 by W3C.
- CSS2: Released in 1998, but still not fully supported by all browsers.
- CSS3: Specification still under development by the W3C (since 1998), “completely backwards compatible with CSS2” (according to the W3C).
  CSS3 has split the development into separate modules, allowing for independent development.
- CSS4: Several “level 4” modules exist.

Content vs. Presentation (cont.)

Style sheets associate presentation formats with HTML elements.

- CSS1: Developed in 1996 by W3C.
- CSS2: Released in 1998, but still not fully supported by all browsers.
- CSS3: Specification still under development by the W3C (since 1998), “completely backwards compatible with CSS2” (according to the W3C).
  CSS3 has split the development into separate modules, allowing for independent development.
- CSS4: Several “level 4” modules exist.

Style sheets can be used to specify how tables should be rendered, how lists should be presented, what colors should be used on the webpage, what fonts should be used and how big/small they are, etc.

Content vs. Presentation (cont.)

HTML style sheets are known as *Cascading Style Sheets*, since can be defined at three different levels:

1. Inline style sheets apply to the content of a single HTML element.
2. Document style sheets apply to the whole <body> of a document.
3. External style sheets can be linked and applied to numerous documents.

Content vs. Presentation (cont.)

HTML style sheets are known as *Cascading Style Sheets*, since can be defined at three different levels:

1. Inline style sheets apply to the content of a single HTML element.
2. Document style sheets apply to the whole <body> of a document.
3. External style sheets can be linked and applied to numerous documents.

Lower-level (i.e. lower-numbered) style sheets can override higher-level style sheets.

Content vs. Presentation (cont.)

HTML style sheets are known as *Cascading Style Sheets*, since can be defined at three different levels:

1. Inline style sheets apply to the content of a single HTML element.
2. Document style sheets apply to the whole <body> of a document.
3. External style sheets can be linked and applied to numerous documents.

Lower-level (i.e. lower-numbered) style sheets can override higher-level style sheets.

(Actually, there are about nine levels of priority.)

Content vs. Presentation (cont.)

HTML style sheets are known as *Cascading Style Sheets*, since can be defined at three different levels:

1. Inline style sheets apply to the content of a single HTML element.
2. Document style sheets apply to the whole <body> of a document.
3. External style sheets can be linked and applied to numerous documents.

Lower-level (i.e. lower-numbered) style sheets can override higher-level style sheets.

(Actually, there are about nine levels of priority.)

Style sheets can also specify how things should be presented on different screen types (e.g. resolution sizes) and/or in print.

Content vs. Presentation (cont.)

Finally, web users (i.e. clients) can even specify personalized style sheets that can be used to override the styles shown on web pages when viewed on their own browser.

This could be useful, for example, to specify larger type sizes for visually-impaired users.

Inline Style Sheets

Using the style attribute, you can specify a presentation style for a single HTML element.

Within the HTML style specification, list a sequence of “property: value” pairs separated by semi-colons.

Inline Style Sheets

Using the style attribute, you can specify a presentation style for a single HTML element.

Within the HTML style specification, list a sequence of "property: value" pairs separated by semi-colons.

Example property: value pairs include:

font-family: Courier,monospace
font-style: italic font-weight: bold
font-size: 12pt font-size: large font-size: larger

Inline Style Sheets

Using the style attribute, you can specify a presentation style for a single HTML element.

Within the HTML style specification, list a sequence of "property: value" pairs separated by semi-colons.

Example property: value pairs include:

font-family: Courier,monospace
font-style: italic font-weight: bold
font-size: 12pt font-size: large font-size: larger
color: red color: #000080
background-color: white

Inline Style Sheets

Using the style attribute, you can specify a presentation style for a single HTML element.

Within the HTML style specification, list a sequence of "property: value" pairs separated by semi-colons.

Example property: value pairs include:

font-family: Courier,monospace
font-style: italic font-weight: bold
font-size: 12pt font-size: large font-size: larger

color: red color: #000080
background-color: white

text-decoration: underline text-decoration: none

text-align: left text-align: center
text-align: right text-align: justify
vertical-align: top vertical-align: middle vertical-align: bottom

text-indent: 5em text-indent: 0.2in

Inline Style Sheets (cont.)

```
<html>
<!-- COMP519  page17.html  2015.09.28 -->

<head>
  <title>Inline Style Sheets</title>
</head>

<body>
  <p style="font-family:Arial,sans-serif;
                  text-align:right">
         This is a right-justified paragraph in a sans serif
         font (preferably Arial), with some
        <span style="color:green">green text</span>.
  </p>

  <p>And <a style="color:red;
                   text-decoration:none;
                   font-size:larger;"
           href="page01.html">here</a>
     is a formatted link.
  </p>
</body>
</html>
```

view page

Inline Style Sheets (cont.)

There are many(!) CSS properties that can be manipulated.

margin-left: 0.1in margin-right: 5%
margin: 3em

padding-top: 0.1in padding-bottom: 5%
padding: 3em

border-width: thin border-width: thick
border-width: 5

border-color: red

border-style: dashed border-style: dotted
border-style: double border-style: none

whitespace: pre

list-style-type: square list-style-type: decimal
list-style-type: lower-alpha list-style-type: upper-roman

CSS Properties

As I stated, there are many CSS properties that can be changed.

Jens Oliver Meiert maintains an excellent CSS Properties Index (along with an HTML Elements Index).

An online search will reveal many sites offering tutorials and examples. Here are a few examples:

- W3Schools CSS Tutorial
- A Brief Introduction to the CSS “Box Model”
- W3C Web Style Sheets, CSS tips & tricks
- A self-study course given in a Wiki-style format

Inline Style Sheets (cont.)

```
<html>
<!-- COMP519  page18.html  2015.09.28 -->
<head>  <title>Inline Style Sheets</title>   </head>
<body>
     <p>Here is an image
       <img src="VictoriaBldg.jpeg"
       alt="image of Victoria Building"
       height="300"
       title="Victoria Building"
       style="margin-left:0.3in; margin-right:0.3in;
            vertical-align:middle;  border-style:double;
            border-color:blue">  embedded in text.
     </p>
     <ol style="list-style-type:upper-alpha">
            <li>one thing</li><li> or another</li>
            <ul style="list-style-type:square; whitespace:pre">
                  <li> with  this</li>
                  <li> or                          that</li>
            </ul>
     </ol>
</body>
</html>
```

view page

Inline Style Sheets (cont.)

```
<html>
<!-- COMP519  page19.html 2015.09.29 -->
<head>
  <title> Inline Style Sheets </title>
</head>
<body>
  <table style="font-family: Arial,sans-serif">
    <caption style="color: red;
                    font-style: italic;
                    text-decoration: underline">
       Student data. </caption>
    <tr style="background-color: red">
      <th> name </th> <th> age </th>
    </tr>
    <tr>
      <td> Chris Smith </td> <td> 19 </td>
    </tr>
    <tr>
      <td> Pat Jones </td> <td> 20 </td>
    </tr>
    <tr><td>Doogie Howser</td><td>9</td></tr>
</table>
</body>   </html>
```

Styles can be applied to <table> elements (and sub-elements) for interesting effects.

This is better handled using document or external style sheets.

view page

Document Style Sheets

Inline style sheets apply to individual elements in the page.

Using inline style directives can lead to inconsistencies, as similar elements might end up formatted differently, e.g. we might like for all <h1> elements to be centered, and manually editing individual webpages means we could miss altering some <h1> elements on some pages.

Document Style Sheets

Inline style sheets apply to individual elements in the page.

Using inline style directives can lead to inconsistencies, as similar elements might end up formatted differently, e.g. we might like for all <h1> elements to be centered, and manually editing individual webpages means we could miss altering some <h1> elements on some pages.

Also, inline definitions mix content and presentation, which violates the (current) general philosophy of HTML and CSS.

Document Style Sheets

Inline style sheets apply to individual elements in the page.

Using inline style directives can lead to inconsistencies, as similar elements might end up formatted differently, e.g. we might like for all <h1> elements to be centered, and manually editing individual webpages means we could miss altering some <h1> elements on some pages.

Also, inline definitions mix content and presentation, which violates the (current) general philosophy of HTML and CSS.

As a general rule, inline style sheet directives should be used as sparingly as possible.

Document Style Sheets

Inline style sheets apply to individual elements in the page.

Using inline style directives can lead to inconsistencies, as similar elements might end up formatted differently, e.g. we might like for all <h1> elements to be centered, and manually editing individual webpages means we could miss altering some <h1> elements on some pages.

Also, inline definitions mix content and presentation, which violates the (current) general philosophy of HTML and CSS.

As a general rule, inline style sheet directives should be used as sparingly as possible.

Alternatively, document style sheets allow for a cleaner separation of content and presentation. Style definitions are placed in the <head> of the page, within <style> tags.

Document Style Sheets

Inline style sheets apply to individual elements in the page.

Using inline style directives can lead to inconsistencies, as similar elements might end up formatted differently, e.g. we might like for all <h1> elements to be centered, and manually editing individual webpages means we could miss altering some <h1> elements on some pages.

Also, inline definitions mix content and presentation, which violates the (current) general philosophy of HTML and CSS.

As a general rule, inline style sheet directives should be used as sparingly as possible.

Alternatively, document style sheets allow for a cleaner separation of content and presentation. Style definitions are placed in the <head> of the page, within <style> tags.

Document style sheets allow us to apply style definitions to all elements, or a subclass of elements, throughout the page.

Document Style Sheets (cont.)

```
<html>
<!-- COMP519  page20.html  2015.09.29 -->
<head>
  <title>Document Style Sheets</title>
  <style type="text/css">
     h1 {  color: blue;
           text-align: center;  }
     p.indented {  text-indent: 0.2in;   }
  </style>
</head>
<body>
  <h1> Centered Title </h1>

  <p class="indented">This paragraph will have the first
  line indented, but subsequent lines will be flush.  <br>
  This next line starts unindented.  </p>

  <p>This paragraph will not be indented.
  </p>

 <h1> The End </h1>
</body>
</html>
```

view page

Selectors, Properties, and Values

Document (and external) style sheet directives consist of a "selector", together with one or more "property: value" pairs, where the pairs are enclosed inside of braces, and separated by semi-colons.

Examples include:

```
h1 { color: blue; text-align: center; }

.alert { text-decoration: underline;
         color: red;
         font-size: 150%; }

ol, a { background-color: yellow;
        font-style: bold;
        font-family: "Times New Roman"; }
```

These directives are placed in a <style> element in the <head> element, in a manner similar to the previous example.

Selectors, Properties, and Values (cont.)

Note that the example

```
.alert { text-decoration: underline;
         color: red;
         font-size: 150%; }
```

creates a class, which can (in principle) be applied to any HTML element, in a manner similar to the example below (assumed to be part of a larger valid HTML document).

```
<p class="alert">Help me! I'm falling down!</p>
```

This would create a paragraph with text that is red, underlined, and 150% of the normal text size of the webpage.

The first example below would apply the “alert” class to both items in the list, so both would be red, underlined, and in a larger font. (The number of the list item is also in a larger red font and could appear underlined or not, depending upon the browser used, it seems.)

```
<ol class="alert">
    <li>Help!</li>
    <li>Beware!</li>
</ol>
```

The first example below would apply the “alert” class to both items in the list, so both would be red, underlined, and in a larger font. (The number of the list item is also in a larger red font and could appear underlined or not, depending upon the browser used, it seems.)

```
<ol class="alert">
    <li>Help!</li>
    <li>Beware!</li>
</ol>
```

This second example would apply the “alert” to only the first item in the list.

```
<ol>
    <li class="alert">Help!</li>
    <li>Beware!</li>
</ol>
```

Also note the very subtle (but important!) distinction between the following two selector definitions:

```
ol, a { background-color: yellow;
        font-style: bold;
        font-family: "Times New Roman"; }
```

This makes a style declaration that applies to both <ol> and <a> elements. (Note the comma between the ol and a.)

Also note the very subtle (but important!) distinction between the following two selector definitions:

```
ol, a { background-color: yellow;
        font-style: bold;
        font-family: "Times New Roman"; }
```

This makes a style declaration that applies to both <ol> and <a> elements. (Note the comma between the ol and a.)

```
ol a { background-color: yellow;
       font-style: bold;
       font-family: "Times New Roman"; }
```

This second example makes a style declaration that applies to <a> elements that are children of (i.e. contained in) elements. (There is no comma between the ol and a.)

Classes and Ids

It is possible to define classes that apply only to specific HTML elements.

One example like this was shown already.

```
p.indented { text-indent: 0.2in; }
```

This allows you to ensure that certain styles aren't accidentally applied where they shouldn't be (for example).

Classes and Ids

It is possible to define classes that apply only to specific HTML elements.

One example like this was shown already.

```
p.indented { text-indent: 0.2in; }
```

This allows you to ensure that certain styles aren't accidentally applied where they shouldn't be (for example).

An id can also be defined, which is supposed to be used to specify a unique element within a webpage. For example:

```
#bigorange { color: #ffa500;
             font-size: 200%;
             font-weight: bold; }
```

Then apply it to the HTML element using `id="bigorange"` in the HTML element's tag.

Formatting Tables Using CSS (an example)

```
<html>
<!-- COMP519  page21.html  2015.10.01 -->

<head>
  <title> Inline Style Sheets </title>
 <style type="text/css">
     table {  font-family: Arial,sans-serif;  }
     caption {  color: red;  font-style: italic;
                text-decoration: underline;  }
     th {  background-color:red;  }
     td {  padding:  0 1em 0 2em;  }
  </style>
</head>
<body>
  <table>
    <caption> Student data. </caption>
    <tr><th> name </th>           <th> age</th></tr>
    <tr><td> Chris Smith </td>   <td> 19 </td></tr>
    <tr><td> Pat Jones </td>     <td> 20 </td></tr>
    <tr><td> Doug MacKenzie </td> <td> 32 </td></tr>
</table>
</body>
</html>
```

view page

External Style Sheets

Modularity is key to the development and reuse of software.

- E.g., design/implement/test useful routines and classes.
- Then, package and make available for repeated use in various ways.
- Saves in development cost and time.
- Central libraries make it possible to make a single change and propagate the changes quickly and easily.

External Style Sheets

Modularity is key to the development and reuse of software.

- E.g., design/implement/test useful routines and classes.
- Then, package and make available for repeated use in various ways.
- Saves in development cost and time.
- Central libraries make it possible to make a single change and propagate the changes quickly and easily.

External style sheets place the style definitions in a separate file (or files).

- Multiple pages can link to the same style sheet.
- This gives a consistent look across the multiple pages of a website.
- Possible to make a single change in a style sheet and it propagates automatically to all pages that use that style sheet.

External style sheets represent the ultimate in separation of content and its representation.

Modularity and Style Sheets

```
<html>
<!-- COMP519  page26.html 2015.10.01 -->
<head>
  <title>Title for Page</title>
  <link rel="stylesheet"
        type="text/css"
        href="myStyle.css"
        title="myStyle">
</head>
<body>
    <h1>Centered Title</h1>
    <p class="indented">This paragraph will have the first
       line indented, but subsequent lines will be flush.  <br>
       Here is a second line in the first paragraph.  </p>
    <p>This paragraph will not be indented.  </p>
    <h1>The End</h1>
</body>
</html>
```

```
/* myStyle.css  COMP519 2015.10.01 */

h1 {color : blue; text-align : center}
p.indented {text-indent: 0.5in}
```

view page

Modularity and Style Sheets (cont.)

Ideally, the developer(s) of a website will place all formatting options in an external style sheet.

All webpages for a particular website will then link to that same style sheet, in order to maintain a consistent look through the set of webpages.

Using an external style sheet can help simplify webpage design, since the developer(s) need to only specify the structure and content of the pages, and the styling is left to the style sheet declarations.

Note that no <style> tags are used in the external style sheet file.

Modularity and Style Sheets (cont.)

It is possible to use several style sheets, and it is quite common to use one that specifies styles for a web browser and another for styles to use when a webpage is printed.

For example, you could include two links in the <head> element of the form

```
<link rel="stylesheet" type="text/css" media="screen"
      href="browser.css">
<link rel="stylesheet" type="text/css" media="print"
      href="print.css">
```

So style sheets can be tailored in this manner for different media.

The Tag

Problem: Font properties apply to whole elements, which are often too large.

Solution: A new tag to define an element in the content of a larger element, the tag. The default meaning of is to leave the content as it is (i.e. unchanged). But with a class definition (or inline style declaration), you can apply it to a portion of text (or some other element too).

```
...
<style type = "text/css">
 .bigred {  font-size: 24pt;
            font-family: Ariel;
            color: red   }
 </style>
...

 <p> Now is the <span class="bigred">
     best time </span> ever!
 </p>
...
```

Like most other HTML tags, a tag can be nested, have a style attribute, a class, and/or an id assigned to it to apply style effects to its contents.

The Tag

The tag is similar to the tag, useful for defining a section of content to which you may apply style definitions (using a class, id, or inline style declaration attribute with the tag).

The main difference between the two tags is that, by default, a web browser will place a line break before and after a element. (That default behavior can, of course, be changed using other CSS declarations to alter it.)

Media Queries (or Media Rules)

In addition to (or instead of) style sheets for different types of media, you can also include “media queries” or “media rules” in a style sheet that can be used to define styles for different types of media and/or devices.

Media queries were introduced in the CSS3 specification, extending the media types of CSS2.

Media Queries (or Media Rules)

In addition to (or instead of) style sheets for different types of media, you can also include “media queries” or “media rules” in a style sheet that can be used to define styles for different types of media and/or devices.

Media queries were introduced in the CSS3 specification, extending the media types of CSS2.

Media queries look at the capability of the device being used by the client, and can check many things including (according to the W3C explanation of the “media rule”):

- width and height of the viewport
- width and height of the device
- orientation (is the phone/tablet in portrait or landscape mode?)
- resolution

Media Queries (cont.)

Interested people can find more information about media queries using the provided link (and, of course, other online searches).

A typical example might be something like the following:

```
...
<style type="text/css">

    ....

   @media screen and (max-width: 500px)  {
                     body  {  background-color:  green;  }

                     ul  {  list-style-type:  square;  }
    }

</style>
...
```

Bootstrap

Bootstrap is a free and open-source framework and collection of tools that can be used for creating websites.

Bootstrap

Bootstrap is a free and open-source framework and collection of tools that can be used for creating websites.

Bootstrap has design templates using HTML and CSS declarations for forms, navigation elements, buttons, typography, and other interface elements. It also includes optional JavaScipt extensions (utilizing jQuery).

Bootstrap

Bootstrap is a free and open-source framework and collection of tools that can be used for creating websites.

Bootstrap has design templates using HTML and CSS declarations for forms, navigation elements, buttons, typography, and other interface elements. It also includes optional JavaScipt extensions (utilizing jQuery).

Bootstrap has been designed to work with all major browsers (but the alpha release, in August 2015, of Bootstrap 4 has dropped support for Internet Explorer version 8). And it has also been designed to support mobile devices through the use of media queries in all of its CSS templates.

Bootstrap

Bootstrap is a free and open-source framework and collection of tools that can be used for creating websites.

Bootstrap has design templates using HTML and CSS declarations for forms, navigation elements, buttons, typography, and other interface elements. It also includes optional JavaScipt extensions (utilizing jQuery).

Bootstrap has been designed to work with all major browsers (but the alpha release, in August 2015, of Bootstrap 4 has dropped support for Internet Explorer version 8). And it has also been designed to support mobile devices through the use of media queries in all of its CSS templates.

I won't be covering Bootstrap here in this module, but want to make you aware of its existence. You are free to utilize it in making your own websites for this course (properly attributed, of course), but I would strongly recommend you to familiarize yourself with basic CSS concepts first.

Web Rules of Thumb (Ok, my rules of thumb. . .)

HTML and CSS provide lots of neat features, but just because you can add a feature doesn't mean you should!

Web Rules of Thumb (Ok, my rules of thumb. . .)

HTML and CSS provide lots of neat features, but just because you can add a feature doesn't mean you should!

Don't add features that distract from the content of the page.

Web Rules of Thumb (Ok, my rules of thumb. . .)

HTML and CSS provide lots of neat features, but just because you can add a feature doesn't mean you should!

Don't add features that distract from the content of the page.

Some suggestions that I offer:

Web Rules of Thumb (Ok, my rules of thumb. . .)

HTML and CSS provide lots of neat features, but just because you can add a feature doesn't mean you should!

Don't add features that distract from the content of the page.

Some suggestions that I offer:

- Use (non-default) colors and fonts sparingly, and be careful how elements fit together.
  E.g. no purple text on a pink background, no "weird" fonts.
  E.g. I find bright white text on a black background difficult to read.

Web Rules of Thumb (Ok, my rules of thumb. . .)

HTML and CSS provide lots of neat features, but just because you can add a feature doesn't mean you should!

Don't add features that distract from the content of the page.

Some suggestions that I offer:

- Use (non-default) colors and fonts sparingly, and be careful how elements fit together.
  E.g. no purple text on a pink background, no "weird" fonts.
  E.g. I find bright white text on a black background difficult to read.

  Consider the needs of visually impaired users of your website!! (For example, use "em"s or percentages to specify font sizes, not fixed pixel sizes.)

Web Rules of Thumb (Ok, my rules of thumb. . .)

HTML and CSS provide lots of neat features, but just because you can add a feature doesn't mean you should!

Don't add features that distract from the content of the page.

Some suggestions that I offer:

- Use (non-default) colors and fonts sparingly, and be careful how elements fit together.
  E.g. no purple text on a pink background, no "weird" fonts.
  E.g. I find bright white text on a black background difficult to read.

  Consider the needs of visually impaired users of your website!! (For example, use "em"s or percentages to specify font sizes, not fixed pixel sizes.)
  Remember that an estimated 8-10% of people have some type of color-blindness, so choose color combinations appropriately.

Web Rules of Thumb (Ok, my rules of thumb. . .)

HTML and CSS provide lots of neat features, but just because you can add a feature doesn't mean you should!

Don't add features that distract from the content of the page.

Some suggestions that I offer:

- Use (non-default) colors and fonts sparingly, and be careful how elements fit together.
  E.g. no purple text on a pink background, no "weird" fonts.
  E.g. I find bright white text on a black background difficult to read.

  Consider the needs of visually impaired users of your website!! (For example, use "em"s or percentages to specify font sizes, not fixed pixel sizes.)
  Remember that an estimated 8-10% of people have some type of color-blindness, so choose color combinations appropriately.
  Some people use screen readers to read content on webpages (so, e.g., include "alt" properties on images).

- Use images only where appropriate.
  E.g. bright background images can make text hard to read.
  E.g. using clickable images instead of standard HTML buttons or links can slow access as images have to be loaded.

- Use images only where appropriate.
  E.g. bright background images can make text hard to read.
  E.g. using clickable images instead of standard HTML buttons or links can slow access as images have to be loaded.
- Don’t rely on window or font size for layout.
  E.g. the font size may be adjusted by the client.
  Consider using CSS media queries to help account for different devices people use.

- Use images only where appropriate.
  E.g. bright background images can make text hard to read.
  E.g. using clickable images instead of standard HTML buttons or links can slow access as images have to be loaded.
- Don't rely on window or font size for layout.
  E.g. the font size may be adjusted by the client.
  Consider using CSS media queries to help account for different devices people use.
- Don't be annoying!
  E.g. avoid lots of pop-up windows, silly music, etc.

- Use images only where appropriate.
  E.g. bright background images can make text hard to read.
  E.g. using clickable images instead of standard HTML buttons or links can slow access as images have to be loaded.
- Don't rely on window or font size for layout.
  E.g. the font size may be adjusted by the client.
  Consider using CSS media queries to help account for different devices people use.
- Don't be annoying!
  E.g. avoid lots of pop-up windows, silly music, etc.
- Break a large document (webpage) into several smaller ones or provide a menu for navigation.

- Use images only where appropriate.
  E.g. bright background images can make text hard to read.
  E.g. using clickable images instead of standard HTML buttons or links can slow access as images have to be loaded.
- Don't rely on window or font size for layout.
  E.g. the font size may be adjusted by the client.
  Consider using CSS media queries to help account for different devices people use.
- Don't be annoying!
  E.g. avoid lots of pop-up windows, silly music, etc.
- Break a large document (webpage) into several smaller ones or provide a menu for navigation.
- Utilize style sheets to make changes easy and ensure consistency across a set of webpages.
  Consider using a tool such as Bootstrap to assist with different cross-browser CSS implementations.

- Use images only where appropriate.
  E.g. bright background images can make text hard to read.
  E.g. using clickable images instead of standard HTML buttons or links can slow access as images have to be loaded.
- Don't rely on window or font size for layout.
  E.g. the font size may be adjusted by the client.
  Consider using CSS media queries to help account for different devices people use.
- Don't be annoying!
  E.g. avoid lots of pop-up windows, silly music, etc.
- Break a large document (webpage) into several smaller ones or provide a menu for navigation.
- Utilize style sheets to make changes easy and ensure consistency across a set of webpages.
  Consider using a tool such as Bootstrap to assist with different cross-browser CSS implementations.
- Stick to standard features and test many browsers if possible (and versions of the same browser).