Slightly more advanced CSS

Comments

Comments in CSS are applied using a forward slash and star:

/* COMMENT */

class and **id**

- The **class** and **id** attributes in HTML completely eliminate the need for inline CSS, so we can export every style specification to an external style sheet.
- So far, we have only looked at HTML selectors in CSS (selectors which represent HTML tags).
- For example:

```
p {color: #f00;}
a {text-decoration: none;}
h2 {font-size: 200%;}
```

class and **id**

- However, we can also define our own selectors with **class** and **id**.
- The benefit of this is that we can present the same HTML element differently depending on its class or id (meaning, for example, that we can make some paragraphs red and some blue).

class and **id** examples

- In the style sheet, a class selector is a name preceded by a **period** (**.**) and an id selector is a name preceded by a **hash character** (**#**)
- So the CSS will look something like this:

#dictionary {background-color: #ffa; font-size: 1.5em;}

.noun {color: red; font-weight: bold;}

- The HTML refers to the CSS by using the **id** and **class** attributes. This will look something like this:

<div **id="dictionary"**><h1>Letter A</h1>

<p **class="noun"**>Aardvark: some kind of animal

Acute: Sharp, smart, pointed

<p **class="noun"**>Apple: a delicious fruit

class and id

- The difference between an id and a class is that an id can be used to identify one single element, whereas a class can be used to identify more than one.
- You can also apply a selector to a pre-defined HTML element by writing the HTML selector first, so **p.makeBlue {color: blue;}** will apply a blue color to all the paragraphs in the HTML document which have the class makeBlue.

Grouping

- You can give the same properties to a number of selectors without having to repeat them by separating the selectors with **commas**.
- For example:

```
h2 {color: red;}
.someOtherClass {color: red;}
.thirdClass {color: red;}

h2, .someOtherClass, .thirdClass {color: red;}
```

Nesting

- If the CSS is structured well, there shouldn't be a need to use too many class or id selectors. This is because you can specify properties to selectors within other selectors. For example:

#chores {background-color: #ccc; padding: 1em;}

#chores h1 {color: #ff0;}

#chores p {color: red; font-weight: bold;}

- Applying this to the following HTML removes the need for additional classes or ids:

** **

<h1>My chores**</h1>**

** **First I have to clean my house, starting with my room.** **

** **Then I have to rake the leaves outside and mow the lawn.** **

** **

- By separating the selectors with **spaces**, we are saying "**h1** inside the id **chores** is color #ff0" and "**p** inside the id **chores** is red and bold".
- This can get complicated because it can go for more than two levels, so it takes practice to make efficient.

Pseudo classes

- **Pseudo classes** get attached to selectors to specify a state or relation to the selector.
- They take the following form, with a **colon** in between the selector and the pseudo class:

selector: pseudo class {property: value;}

- Some of these are not supported by all browsers, but the following pseudo classes can be used safely when applied to links:

link is for an unvisited link.

visited is for a link to a page that has already been visited.

active is for a link when it is clicked on.

hover is for a link when the cursor is held over it.

Pseudo class example

a.homework: **link** {color: blue;}
a.homework: **visited** {color: purple;}
a.homework: **active** {color: red;}
a.homework: **hover** {
text-decoration: none;
color: blue;
background-color: yellow;
}

hover

- You should be able to use the **hover** pseudo class with elements other than links.
- However, Internet Explorer usually doesn't support this method.

Shorthand Properties

- Some CSS properties allow a string of values, replacing the need for a bunch of separate properties.
- These are represented by values separated by **spaces**.
- **margin**, **padding** and **border-width** allow you to combine **margin-top-width**, **margin-right-width**, **margin-bottom-width,** etc. in the following way:

property: top right bottom left;

Shorthand example

```
p {
border-top-width: 2px;
border-right-width: 7px;
border-bottom-width: 10px;
border-left-width: 8px;
}
```

Can be replaced by:

p {**border-width: 2px 7px 10px 8px;**}

More shorthand

- **border-width**, **border-color** and **border-style** can also be combined; for example:

p {**border: 1px red solid;**}

- This can also be applied to **border-top**, **border-right**, etc.
- If only two values are stated (**margin: 1em 10em;**), the first value will be for the top and bottom and the second value will be for the left and right.

More shorthand

- Font-related properties can also be gathered together with the font property:

p {**font: italic bold 1em/1.5 courier;**}

- **/1.5** specifies the line-height.

Mixed example

```
p {
font: 1em/1.5 "Times New Roman", times, serif;
padding: 3em 1em;
border: 1px black solid;
border-width: 1px 5px 5px 1px;
border-color: red green blue yellow;
margin: 1em 5em;
}
```

Background images

- The **background** property deals with the manipulation of **background images**.

body {
background: white
url(http://www.somewebsite.com/images/sample.jpg)
no-repeat top right;
}

- This combines the following properties:

background-color, which specifies the color of the background.

background-image, which specifies the location of the background image.

background-repeat, which specifies how the image repeats itself. This can be **repeat**, **repeat-y** (repeating on the y-axis), **repeat-x** (repeating on the x-axis) or **no-repeat** (which shows just one instance of the image).

background-position, which can be **top**, **center**, **bottom**, **left**, **right** or a combination.

- Background-images can be used in most HTML elements - not just in the body, and can be used for simple but effective results such as curved corners.

Background images

- Don’t get carried away with background images and plaster them all over your web pages.
- Having a distracting, brightly colored picture tiled across the background of a page makes it a lot harder to read the foreground text.
- The most user-friendly, readable text is black on a plain white background (or a slightly off-white background and off-black text for reducing glare).
- So, background images should either have no content on top of them, or be made very light (which would also reduce their size).

Specificity

- If you have two or more conflicting CSS specifications that concern the same element, there are rules to determine which one gets applied.
- The rule of thumb is that the most **specific** CSS style specification is the one that gets applied.
- Such conflicts are not very typical, but the larger and more complex your CSS files become, the more likely it is that some conflicts turn up.

Specificity

- If the selectors are the same then the one that comes latest in the style sheet will take precedence.
- For example, in the following code, paragraphs would be colored blue because that specification comes last.

```
p { color: red; }
p { color: blue; }
```

- However, you usually won't have identical selectors with conflicting declarations on purpose.

Specificity

- Conflicts usually come up when you have nested selectors.

```
div p { color: red; }

p { color: blue; }
```

- With this code, paragraphs will get colored red, due to the specificity of the first selector. Since **div p** targets only paragraphs within a div element, this is a more specific selector, and is therefore given more preference when there is a conflicting style.

Specificity

- To calculate the actual specificity of a group of nested selectors, you may use the following rules.
- Give every id selector (#someID) a value of 100
- Give every class selector (.someClass) a value of 10
- Give every HTML selector (someTag) a value of 1
- Add up the values and obtain a specificity value.
- The selectors with the highest values get applied.

Specificity example

- **p** has a specificity of 1 (one HTML selector)
- **div p** has a specificity of 2 (two HTML selectors)
- **.noun** has a specificity of 10 (one class selector)
- **div p.noun** has a specificity of 12 (two HTML selectors and one class selector)
- **#dictionary** has a specificity of 100 (one id selector)
- **body #dictionary .noun p** has a specificity of 112 (two HTML selectors, one id selector, one class selector)