CSS - Cascading Style Sheets

CSS defines how HTML elements are displayed. Styling can happen in a few different places. The most common place for CSS is an external style sheet (which is a file with a .css extension). You place a <link> between your HTML <head> tags like so:

```
<head>
    <link href="stylesheet.css" rel="stylesheet" type="text/css">
</head>
```

Note: You can use relative or absolute paths to your style sheets for the 'href' attribute of <link>.

You can also place a <style> tag in the <head> portion of your HTML document and define your CSS there; this is called an internal style sheet.

```
<head>
   <style type=”text/css”>
       …
   </style>
</head>
```

Finally you can add a 'style' attribute to any HTML element and define your styles there; this is called an inline style.

```
<div style=”width:50px;height:50px;”>…</div>
```

The Cascading in Style Sheets

When an HTML element has multiple styles defined on it the one with the highest priority will be chosen and override the rest. An inline style (a style defined on the HTML element) has the highest priority and will override any other CSS defined. Next is an internal style sheet (one defined in the header of your HTML document), then External style sheets (you reference these with a link tag in the header of your HTML document, which are typically declared before the internal style sheets). Finally, browser default options are at the bottom of the list, and will have the lowest priority.

1. inline styles
2. internal / external style sheets (last one defined determines style)
3. browser defaults

Note that overwriting happens only if the *specificity* of the selectors is the same. So for example, let's say you have a style that applies to p elements in a div, and then later on you have a style that applies to all p elements.

p elements inside a div will get the first style, because the most specific style wins even if there's a more general one later.

Style declarations aren't monolithic. When something gets "overridden," what's really happening is that any declarations that are the same level of specificity *and* the same property are overridden, but all the other properties remain.

So for example, let's say you have this:

```
div p { /* applies to p elements inside a div */
 color: blue;
}
```

And then later on, you have this:

```
p { /* applies to all p elements */
 color: black;
 text-decoration: underline;
}
```

<p> elements in a div will be blue and underlined and all other <p> elements will be black. The more-specific declaration has a color so that overrides the general color, even though it is defined first. Because it doesn't say anything about text-decoration, that style is determined from the more-general set.

CSS Syntax

A typical CSS statement looks like this:

SELECTOR {DECLARATION[PROPERTY: VALUE];DECLARATION[PROPERTY:VALUE]; }

For example:

h1 { color: #FFFFFF; }

Selector-> h1
Declaration-> color: #FFFFFF;
Property-> color
Value-> #FFFFFF

CSS declarations ***always*** end with a semicolon, and curly brackets surround declaration groups.

Note: Do not leave spaces between property values and units

incorrect	top: 20 px;
correct	top: 20px;

Comments

A CSS comment begins with "/*", and ends with "*/", like this:

```
/* This is a comment */

/*
This is a
multiline
comment
```

```
*/
```

Identifying elements with ID and Class

ID defines a special and unique case for an element (this means that it can only be used once per document). These should be treated like global variables and used sparingly.

```
<html>
   <head>
      <style type=”text/css”>
         #unique_box {
             width: 50px;
             height: 50px;
             background-color: blue;
          }
       </style>
   </head>
   <body>
      <div id=”unique_box”></div>
   </body>
</html>
```

In CSS an id is declared with a pound sign ‘#’ followed by a unique name. Such as ‘#unique_box’ in the example above. In CSS if you follow a class declaration with a selector you can define specific declarations for that element.

CSS classes define a special non-unique case for elements. Classes should be used when multiple elements require the same styling.

```
<html>
   <head>
      <style type=”text/css”>
         .box {
             width: 50px;
             height: 50px;
          }
       </style>
   </head>
   <body>
      <div class=”box”></div>
      <div class=”box”></div>
      <div class=”box”></div>
   </body>
</html>
```

CSS classes are declared with a period ‘.’ followed by a unique name. Such as ‘.box’ in the example above. In CSS if you follow a class declaration with a selector you can define specific declarations for that element.

```
.box p {
    color: green;
}
```

Where ‘.box’ is the class, ‘p’ is the selector and ‘color: green;’ is the declaration.

Common patterns

Generally you won't be writing CSS that applies to all <p> elements, or all <a> elements. You will write CSS that applies only to certain elements based on how they are placed relative to other elements. For example, you might have a specific style for all <p> elements inside any <div> with class 'bounding-box.'

Examples of nesting selectors:

```
div p { /* all p elements that are inside a div */
 color: green;
}
div p.box { /* all p elements with class box that are inside a div */
 color: black;
}
div.main-text p.box { /* all p elements with class box that are inside
a div with class main-text */
 color: blue;
}
```

Examples of grouping selectors:

```
/* all p and h1 elements inside the div with class main-text */
div.main-text p, div.main-text h1 {
  color: black;
}
```

Common CSS Attributes

Display: block verses inline

The display property controls how an element is displayed. It does this with two properties called block and inline. The block property tells the element to take up the full width available and forces line breaks in text. While the inline property tells the element to take up just as much width as necessary and doesn't force line breaks.

Note: 'display: none;' will hide an element, making it invisible.

These HTML elements have a 'display: block;' by default

<p>, <h1>…<h4>, <div>

These HTML elements have a display: inline; by default

<a>, <span>

Visibility has two values visible or hidden to control whether an element is visible or not. [visibility: hidden;]

Margin clears the area outside of the container. Margin takes 4 values in a clockwise rotation: MARGIN TOP RIGHT BOTTOM LEFT. Each value must be defined in pixels, pt, em, or %.

Note: Negatives values are allows, so that you may overlap content.

Similar Elements:

margin-left: VALUE;
margin-right: VALUE;
margin-top: VALUE;
margin-bottom: VALUE;

Padding clears the area inside the container. Padding takes 4 values in a clockwise rotation: PADDING TOP RIGHT BOTTOM LEFT. Each value must be defined in pixels, pt, em, or %. [p { padding: 0px 10px 0px 10px;}]

Note: Negative values are not allowed.

Similar Elements:

padding-left: VALUE;
padding-right: VALUE;
padding-top: VALUE;
padding-bottom: VALUE;

Background controls the background color or image of an HTML element. Has options BACKGROUND: COLOR IMAGE REPEAT ATTACHMENT POSITION. [body { background: #00ff00 url('image.png') no-repeat fixed top; }]

Similar Elements:

background-color: VALUE;
background-image: VALUE;
background-repeat: VALUE;
background-attachment: VALUE;
background-position: VALUE;

Color controls text color. Colors can be defined by name [color: red;], RGB [color: `rgb(255,0,0);`] or hex representation [color: #ff0000;].

Text-Align is used to set the horizontal alignment of text. [p {text-align: center;}]

Text-Decoration allows you to over-line, under-line, line-through or blink text. The blink option will flash the text and hide it at a fixed rate. It is not supported in IE, Safari or Chrome. It is most commonly used to remove the decoration for link elements [a { text-decoration: none; }]

Text-Transform used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word. [h1 { text-transform: uppercase; }]

Float specifies how elements lay out relative to each other. Elements can be told to move as far left or right as they can, allowing other elements to wrap around them. Floating <div>(s) or <img>(s) is common.

```
        <html>
           <head>
              <style type="text/css">
                 img {
                     float: right;
                  }
              </style>
           </head>
           <body>
        <p> This text is only here to show wrapping around the image. You will see that
the text will continue to flow on the left around the image on the right. You will also
see that the image has floated as far right as possible.
        <img src="image.jpg" width="50" height="50" alt="some image" /></p>
           </body>
        </html>
```

TIP: Elements after the floated element(s) will continue to wrap. To avoid this use the 'clear' property on the elements you do not want floated. Values of clear are: left, right, both, none, inherit.

```
        .foo {
           clear: both;
        }
```