CSS Basics.com

Chapter: 1 - Introduction to CSS

A CSS (cascading style sheet) file allows you to separate your web sites (X)HTML content from it's style. As always you use your (X)HTML file to arrange the content, but all of the presentation (fonts, colors, background, borders, text formatting, link effects & so on...) are accomplished within a CSS.

At this point you have some choices of how to use the CSS, either internally or externally.

Internal Stylesheet

First we will explore the internal method. This way you are simply placing the CSS code within the <head></head> tags of each (X)HTML file you want to style with the CSS. The format for this is shown in the example below.

```
<head>
<title><title>
<style type="text/css">
CSS Content Goes Here
</style>
</head>
<body>
```

With this method each (X)HTML file contains the CSS code needed to style the page. Meaning that any changes you want to make to one page, will have to be made to all. This method can be good if you need to style only one page, or if you want different pages to have varying styles.

External Stylesheet

Next we will explore the external method. An external CSS file can be created with any text or HTML editor such as "Notepad" or "Dreamweaver". A CSS file contains no (X)HTML, only CSS. You simply save it with the .css file extension. You can link to the file externally by placing one of the following links in the head section of every (X)HTML file you want to style with the CSS file.

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

Or you can also use the @import method as shown below

```
<style type="text/css">@import url(Path To
stylesheet.css)</style>
```

Either of these methods are achieved by placing one or the other in the head section as shown in example below.

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

or

<head>
<title><title>
<style type="text/css"> @import url(Path To stylesheet.css)
</style>
</head>
<body>
```

By using an external style sheet, all of your (X)HTML files link to one CSS file in order to style the pages. This means, that if you need to alter the design of all your pages, you only need to edit one .css file to make global changes to your entire website.

Here are a few reasons this is better.

- Easier Maintenance
- Reduced File Size
- Reduced Bandwidth
- Improved Flexibility

Are you getting the idea? It's really cool.

Cascading Order

In the previous paragraphs, I have explained how to link to a css file either internally or externally. If you understood, than I am doing a good job. If not don't fret, there is a long way to go before we are finished. Assuming you have caught on already, you are probably asking, well can I do both? The answer is yes. You can have both internal, external, and now wait a minute a third way? Yes inline styles also.

Inline Styles

I have not mentioned them until now because in a way they defeat the purpose of using CSS in the first place. Inline styles are defined right in the (X)HTML file along side the element you want to style. See example below.

```
<p style="color: #ff0000;">Some red text</p>
```

Some red text

Inline styles will NOT allow the user to change styles of elements or text formatted this way

---

So, which is better?

So with all these various ways of inserting CSS into your (X)HTML files, you may now be asking well which is better, and if I use more than one method, in what order do these different ways load into my browser?

All the various methods will cascade into a new "pseudo" stylesheet in the following order:

1. Inline Style (inside (X)HTML element)
2. Internal Style Sheet (inside the <head> tag)
3. External Style Sheet

As far as which way is better, it depends on what you want to do. If you have only one file to style then placing it within the <head></head> tags (internal) will work fine. Though if you are planning on styling multiple files then the external file method is the way to go.

Choosing between the <link related=> & the @import methods are completely up to you. I will mention that the @import method may take a second longer to read the CSS file in Internet Explorer than the <link related=> option. To combat this see Flash of unstyled content

---

Users with Disabilities

The use of external style sheets also can benefit users that suffer from disabilities. For instance, a user can turn off your stylesheet or substitute one of there own to increase text size, change colors and so on. For more information on making your website accessible to all users please read Dive into accessibility

---

Power Users

Swapping stylesheets is beneficial not only for users with disabilities, but also power users who are particular about how they read Web documents.

Browser Issues

You will discover as you delve farther into the world of CSS that all browsers are not created equally, to say the least. CSS can and will render differently in various browsers causing numerous headaches.

Chapter 2 - CSS Syntax

The syntax for CSS is different than that of (X)HTML markup. Though it is not too confusing, once you take a look at it. It consists of only 3 parts.

```
selector { property: value }
```

The selector is the (X)HTML element that you want to style. The property is the actual property title, and the value is the style you apply to that property.

Each selector can have multiple properties, and each property within that selector can have independent values. The property and value are seperated with a colon and contained within curly brackets. Multiple properties are seperated by a semi colon. Multiple values within a property are sperated by commas, and if an individual value contains more than one word you surround it with quotation marks. As shown below.

```
body {
  background: #eeeeee;
  font-family: "Trebuchet MS", Verdana, Arial, serif;
}
```

As you can see in the above code I have seperated the color from the font-family with a semi-colon, seperated the various fonts with commas and contained the "Trebuchet MS" within quotations marks. The final result sets the body color to light grey, and sets the font to ones that most users will have installed on there computer.

I have changed the way I layout my code, but you can arrange it in one line if you choose. I find that it is more readable if I spread each property to a seperate line, with a 2 space indention.

Inheritance

When you nest one element inside another, the nested element will inherit the properties assigned to the containing element. Unless you modify the inner elements values independently.

For example, a font declared in the body will be inherited by all text in the file no matter the containing element, unless you declare another font for a specific nested element.

```
body {font-family: Verdana, serif;}
```

Now all text within the (X)HTML file will be set to Verdana.

If you wanted to style certain text with another font, like an h1 or a paragraph then you could do the following.

```
h1 {font-family: Georgia, sans-serif;}
p {font-family: Tahoma, serif;}
```

Now all <h1> tags within the file will be set to Georgia and all <p> tags are set to Tahoma, leaving text within other elements unchanged from the body declaration of Verdana.

There are instances where nested elements do not inherit the containing elements properties.

For example, if the body margin is set to 20 pixels, the other elements within the file will not inherit the body margin by default.

```
body {margin: 20px;}
```

Combining Selectors

You can combine elements within one selector in the following fashion.

```
h1, h2, h3, h4, h5, h6 {
  color: #009900;
  font-family: Georgia, sans-serif;
}
```

As you can see in the above code, I have grouped all the header elements into one selector. Each one is seperated by a comma. The final result of the above code sets all headers to green and to the specified font. If the user does not have the first font I declared it will go to another sans-serif font the user has installed on there computer.

Comment tags

Comments can be used to explain why you added certain selectors within your css file. So as to help others who may see your file, or to help you remember what you we're thinking at a later date. You can add comments that will be ignored by browsers in the following manner.

```
/* This is a comment */
```

You will note that it begins with a / (forward slash) and than an * (asterisks) then the comment, then the closing tag which is just backward from the opening tag * (asterisks) then the / (forward slash).

Chapter 3: CSS Classes

The class selector allows you to style items within the same (X)HTML element differently. Similiar to what I mentioned in the introduction about inline styles. Except with classes the style can be overwritten by changing out stylesheets. You can use the same class selector again and again within an (X)HTML file.

To put it more simply, this sentence you are reading is defined in my CSS file with the following.

```
p {
```

```
  font-size: small;
  color: #333333
}
```

Pretty simple, but lets say that I wanted to change the word "sentence" to green bold text, while leaving the rest of the sentence untouched. I would do the following to my (X)HTML file.

```
<p>
To put it more simply, this <span
class="greenboldtext">sentence</span> you are reading is styled
in my CSS file by the following.
</p>
```

Then in my CSS file I would add this style selector:

```
.greenboldtext{
  font-size: small;
  color: #008080;
  font-weight: bold;
}
```

The final result would look like the following:

To put it more simply, this **sentence** you are reading is styled in my CSS file by the following.

Please note that a class selector begins with a (.) period. The reason I named it "greenboldtext" is for example purposes, you can name it whatever you want. Though I do encourage you to use selector names that are descriptive. You can reuse the "greenboldtext" class as many times as you want.

Chapter 4: CSS IDs

IDs are similar to classes, except once a specific id has been declared it cannot be used again within the same (X)HTML file.

I generally use IDs to style the layout elements of a page that will only be needed once, whereas I use classes to style text and such that may be declared multiple times.

The main container for this page is defined by the following.

```
<div id="container">
Everything within my document is inside this division.
</div>
```

I have chosen the id selector for the "container" division over a class, because I only need to use it one time within this file.

Then in my CSS file I have the following:

```
#container{
  width: 80%;
  margin: auto;
```

```
  padding: 20px;
  border: 1px solid #666;
  background: #ffffff;
}
```

You will notice that the id selector begins with a (#) number sign instead of a (.) period, as the class selector does.

Chapter 5: CSS Divisions

Ok so you have finished the first 4 chapters in my series. You have learned the very basics of CSS, how the syntax works and a bit about classes and IDs. Now we are gonna take a quick break from CSS and focus on the (X)HTML side of using it.

Divsions

Divisions are a block level (X)HTML element used to define sections of an (X)HTML file. A division can contain all the parts that make up your website. Including additional divisions, spans, images, text and so on.

You define a division within an (X)HTML file by placing the following between the <body></body> tags:

```
<div>
Site contents go here
</div>
```

Though most likely you will want to add some style to it. You can do that in the following fashion:

```
<div id="container">
Site contents go here
</div>
```

The CSS file contains this:

```
#container{
  width: 70%;
  margin: auto;
  padding: 20px;
  border: 1px solid #666;
  background: #ffffff;
}
```

Now everything within that division will be styled by the "container" style rule, I defined within my CSS file. A division creates a linebreak by default. You can use both classes and IDs with a division tag to style sections of your website.

Chapter 6: CSS Spans

Spans are very similar to divisions except they are an inline element versus a block level element. No linebreak is created when a span is

declared.

You can use the span tag to style certain areas of text, as shown in the following:

```
<span class="italic">This text is italic</span>
```

Then in my CSS file:

```
.italic{
  font-style: italic;
}
```

The final result is: *This text is italic.*

The purpose of the last 2 chapters was to provide you with a basis for using CSS in an (X)HTML file. For a more detailed explaination of XHTML please visit W3Schools

Chapter 7: CSS Margins

Inherited: No

As you may have guessed, the margin property declares the margin between an (X)HTML element and the elements around it. The margin property can be set for the top, left, right and bottom of an element. (see example below)

```
margin-top: length percentage or auto;
margin-left: length percentage or auto;
margin-right: length percentage or auto;
margin-bottom: length percentage or auto;
```

As you can also see in the above example you have 3 choices of values for the margin property

- length
- percentage
- auto

You can also declare all the margins of an element in a single property as follows:

```
margin: 10px 10px 10px 10px;
```

If you declare all 4 values as I have above, the order is as follows:

1. top
2. right
3. bottom
4. left