CSS – Cascading Style Sheets

DM2113 Web Development – Lesson 3

In this lesson

- Styles and Cascading Style Sheets
- Span and Division
- ID and Class Selectors

INTRODUCTION TO STYLE SHEETS

An HTML page should consist of *structure and content* only. Style and layout should really be performed with *Cascading Style Sheets*.

Refer to cssExample1.html & cssExample2.html

A large part of HTML coding involves application of styling characteristics to page elements.

Styling includes positioning of elements on the page along with their visual appearance as font faces, sizes, colours, and other display characteristics.

There are two basic ways to control the layout and appearance of page elements

- with *tag attributes* and
- *style sheets*.

Tag Attributes

Attributes give certain characteristics to an HTML element by providing additional information about it. For example, links are defined with the <a> tag with the address specified in the *href attribute.*

Attribute names and attribute values are case-insensitive. However, newer versions of HTML will demand *lowercase attributes*.

Below is a list of some common attributes that can be used on any HTML element:

Attribute	Description
class	Specifies one or more class names for an element (refers to a class in a style sheet)
id	Specifies a unique id for an element
style	Specifies an inline CSS style for an element
title	Specifies extra information about an element (displayed as a tool tip)

Styling attributes for element (deprecated)

For example, a horizontal rule is displayed across the page wherever the *<hr/>* tag is coded.

```
<hr />
```

This tag produces a line often used to separate sections of content on the page.

To give it a different appearance, we can assign values to its attributes inside the tag.

```
<hr size="10" width="50%" color="red" align="center"/>
```

The line is 10 pixels in height (size="10"), 50% of the width of the browser window (width="50%"), red in colour (color="red"), and centred horizontally on the page (align="center").

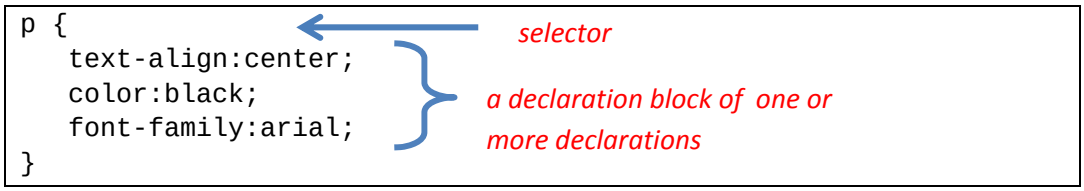
Different tag has attributes that is specific to it and which control the appearance of whatever page content they contain.

Tag attributes are coded in the general format *attribute="value" (e.g. color="red"),* naming the attribute and supplying a quoted value for its setting. The attribute values give styling characteristics to display the rule differently from its normal default style.

The styling attributes for element has been deprecated in HTML 4.01 in favour of style sheets.

CSS SYNTAX

A CSS rule has two main parts: a *selector*, and a block of *one or more declarations*



CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers.

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

```
/*This is a comment*/
p {
text-align:center;
/*This is another comment*/
color:black;
font-family:arial;
}
```

WORKSHOP: APPLYING TEXT STYLE TO A CONTAINER

Text colours are applied by assigning a colour value in the *style sheet* associated with the container tag enclosing the text. *workshop1.html*

Colour

colour.html

Colours can be applied to various page elements with the *color* and *background-color* properties. Colour values can be given by a *colour name*, a *hexadecimal value* representing the combination of red, green, and blue hues to be applied, or by an *RGB (red, green, blue)* decimal value.

There are 16 basic colour names recognized as standard Windows colours: *aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow*.

The following are the equivalent

```
color=red;
color=#ff0000;
color=#f00;
color=rgb(255,0,0);
```

For instance, a heading can be displayed in blue by assigning this colour name in a style sheet for the <hn> tag. All the <h1> elements will be blue.

```
h1 {color:blue}
```

Note the spelling *color* and NOT *colour*. All the <h1> elements will be blue.

If an entire paragraph is to have a particular text colour, then that colour name can be applied with a style sheet associated with its <p> tag.

```
p {color:red}
```

All the <p> elements will be red.

To display all text on a page in a particular colour, assign the colour property to the *<body>* tag.

```
body {color:green}
```

STYLE SHEETS

stylessheet.html

The preferred way to apply styling to page elements is with *Cascading Style Sheets* (CSS). A style sheet is a set of style properties and accompanying style values that describe the appearance of HTML elements to which they are applied.

There are three methods of creating style sheets.

- An *in-line style sheet* appears inside the tag to which *its style declarations apply*;
- an *embedded style sheet* is a separate style section of a Web page which applies its styles to all designated tags on a page;
- A *linked style sheet* is an external document containing style settings that apply to all pages that link to it.

In-line Style Sheets

StylesSheet.html

An in-line style sheet is coded inside a tag to apply styling to that particular tag.

```
<hr style="height:10px; width:50%; background-color:red; text-
align:center"/>
<hr style="height:10px; width:50%; background-color:red; text-
align:center"/>
<hr style="height:10px; width:50%; background-color:red; text-
align:center"/>
<hr style="height:10px; width:50%; background-color:red; text-
align:center"/>
```

One or more *attribute:value* pairs give style settings within a *style attribute* coded in the tag. Multiple style properties settings are separated from each other with *semicolons*; spaces can be included between the settings to help with readability.

The properties and values that can be coded for a tag depend to a large extent on the particular tag being styled.

Limitations

When applied as in-line style sheets the *attributes* and *values* pertain *only to the tag* in which they are coded.

- If there are several <hr/> tags on a page, each would need to include the same in-line style sheet to share the same style.
- This may not present a problem with only a few tags;
- with many tags, it can be tedious and error-prone to code the same style sheet in all of the individual tags.

A way is needed, then, for declaring style settings that can be shared by multiple tags.

Embedded Style Sheets

To avoid duplicate coding of in-line style sheets an embedded style sheet can be used. An embedded style sheet is defined within a *<style>* tag, coded in the *<head>* section of the page.

Within this *<style>* section is a list of tag names, called *selectors*, to which style declarations (*attributes* and *values*) are assigned.

```
<style>
    hr { height:10px; width:50%; background-color:red; text-
align:center }
</style>
```

Once these declarations are made, they are automatically applied to the *identified tags* wherever they are located *on the page*.

A *selector* is a tag name (e.g. hr, without the enclosing "<" and ">" symbols). Style *attributes and values* for the tag are enclosed within a list of style declarations, separated by semicolons, all enclosed within a pair of braces "{ }".

```
General format: selector {attribute:value; attribute:value….}
```

The hr selector is associated with four *attribute:value* declarations to style a horizontal rule. These styles are applied automatically wherever the browser encounters an *<hr/>* tag within the document.

It is *not necessary* to code separate in-line style sheets within each and every <hr/> tag. The tag itself carries the styling described in the embedded style sheet.

Instead of coding the declarations on a single line, you may prefer to code *attribute:value* settings on separate lines for ease of reading.

```
<style>
  hr {height:10px;
      width:50%;
      backgroundcolor:red;
      text-align:center}
</style>
```

Linked Style Sheets

linkedStylesheet.css

If your Web site contains *multiple pages*, a third style sheet alternative is probably the best solution. It permits you to conveniently apply the same styles to all pages without having to duplicate in-line or embedded style sheets on each page.

A linked style sheet is a separate document containing *attribute:value* settings in the same format as an embedded style sheet. The only difference is that the linked document does not include *<style>* tags surrounding the entries.

A linked style sheet document containing specifications for the *<hr/>* tag

```
hr {height:10px;width:50%; background:gold; text-align:center}
```

This separate document is a *standard text file*, usually with the file extension *.css*.

Linking

All pages that use the style sheet *must "link"* to it to make the styles accessible to those pages. This linkage occurs with a <link> tag coded in the <head> section of the page.

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

- The *href* gives the location of the linked style sheet.
- The *type* identifies the type of document to which the link is made (always "text/css").
- The *rel* specifies the relationship of the external document to the current page (always "stylesheet").

The above code is used to link the css document to a Web page. The css document and the Web page reside in the same directory.

Linking to different Directory

For documents in a different directory, there are three kinds of pathnames that can be used in the hypertext reference (href):

- *Absolute Pathnames*
  - Points to root directory eg. "/css/linkedStylesheet.css"
  - Specifies the web address of a file, eg. "http://myWebSite.com/css/linkedStylesheet.css"
- *Relative Pathname* - Points to files based on their locations relative to the current file,
  - eg. "css/linkedStylesheet.css"

As in the case with embedded style sheets, any tags identified in a linked style sheet carry with them the declared styles.

eLEARNING: LINKED, EMBEDDED OR IN-LINE STYLE SHEETS

Whether to employ an embedded style sheet or in-line style sheets is a matter of coding efficiencies and personal choice.

- If your Web site contains *multiple pages*, a *linked style sheet* is probably the best solution.
- If a style is applied to *multiple occurrences* of a tag, it involves less coding and fewer errors to code an *embedded style sheet*, declaring the style one time for sharing by all the tags.
- If a style is applied *one time to a single tag*, then there is convenience in coding an *in-line style sheet* for that particular tag.

Style inheritance

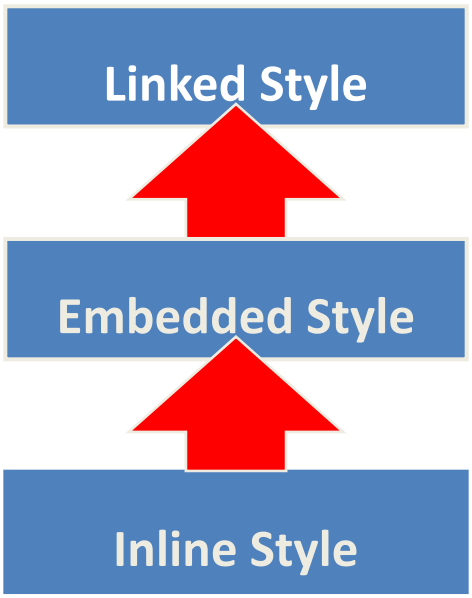
CSSDemo.html

It is *not uncommon* to employ all three types of style sheets for a single Web page.

What style will be used when there is more than one style specified for an HTML element?

The browser handles these multiple style sheets in the following *precedence* order.

- In-line style sheets.
- Embedded style sheets.
- Linked style sheets.



Cascading Style Sheets

Style inheritance is what makes style sheets "*cascading*." Linked styles are inherited by, or cascade across, all Web pages that link to them; embedded styles are inherited by, or cascade across, an entire Web page containing a <style> section; in-line styles are inherited by, or cascade across, individual tags containing in-line style sheets.

STYLING BACKGROUND WITH CSS

StyleBackground.html

Background Properties

Any text container can also be given a background colour by applying the *background-color* property. Just make sure that the colour chosen is complementary to the text colour and does not make it difficult to read the text.

```
p {color:green; background-color:yellow}
```

There are other properties for the background:

- background-image. Allow you to choose an image.
- background-repeat. Can have any one of the 4 different values
- repeat
    - no-repeat
    - repeat-x
    - repeat-y
- background-attachment. Sets whether a background image is fixed or scrolls with the rest of the page.
- background-position. The position/alignment of the image. The first indicates the x-axis (left, centre, right) and the second indicates the y-axis (top, centre, and bottom). You may also enter a number for precise positioning.

Shorthand property - Background

To shorten the code, we specify all the properties in one single property called a shorthand property.

When using the shorthand property the order of the property values is:

- background-image
- background-repeat
- background-attachment
- background-position

CSS3 Background Properties

- background-clip
    - painting area of the background images
- background-origin
    - positioning area of the background images
- background-size
    - size of the background images

WORKSHOP: CHANGING THE BACKGROUND COLOUR AND IMAGE OF A WEB PAGE

StyleBackground.html

You can change the background colour and image of your web page.

It is just as easy to change the background of a page using CSS.

```
body {
   color: #fff;
   background-color: #00c;
   background-image: url(images/skyline.jpg);
   background-repeat:no-repeat;
   background-attachment:fixed;
   background-position:top center;
}
```

Or

```
body {
   background: #fff url(images/skyline.jpg) no-repeat fixed top
center;
}
```

eLEARNING: STYLING LIST WITH CSS

Unordered List

```
<ul style="list-style-type: disc;"> discs: default.
<ul style="list-style-type: circle;"> hollow circles.
<ul style="list-style-type: square;"> square.
```

Ordered List

```
<ol style="list-style-type: lower-alpha;"> Alphabet (lower-alpha
or upper-alpha)
<ol style="list-style-type: upper-roman;"> Roman (lower-roman or
upper-roman).
<ol style="list-style-type: decimal;"> 1,2,3 default.
```

Specify an image as the list item marker

```
list-style-image: url('marker1.png');
```

eLEARNING: STYLING TABLE WITH CSS

The following HTML codes display a 2x2 tables.

```
<table>
   <tr>
      <th>Header 1</th>
      <th>Header 2</th>
   </tr>
   <tr>
      <td>row 1, cell 1</td>
      <td>row 1, cell 2</td>
   </tr>
   <tr>
      <td>row 2, cell 1</td>
      <td>row 2, cell 2</td>
   </tr>
</table>
```

Borders

CSS also provides border properties in the form of the border property. There are also three sub-properties.

- *border-width*. Specify the width of the border. Normally the width is specified in the number of pixels.
- *border-style*. Specify the style of the border. This takes a keyword, such as solid, or dashed.
- *border-color*. Specify a colour for the border. As with all other colour properties in style sheets, it is preferred that you use a hexadecimal value to set the colour.

When you set border properties, they apply to the element selected, not for everything inside the table element. The <table>, <th> and <td> elements have separate borders.

Try it.

To set the border around the table to a two pixel, solid red line:

```
table {border: 2px solid #ff0000;}
```

The "*px*" in the above statement says that the measure is in pixels. The line is a solid line and is pure red. The values need to occur in that order and all need to be present for the command to work. If you just want to set one value, then use the sub-properties.

```
table {border-width: 2px;}
```

For same borders around everything:

```
table, th, td { border: 2px solid #ff0000; }
```

To specify properties for *only some of the sides*, specify *border-left*, *border-right*, *border-top*, or *border-bottom* for the *border-width* sub-properties. These work the same way as the border attribute.

```
{border-width: 2px 4px 6px 4px;}
```

The above statement sets the top border to 2 pixels, the left and right border to four pixels each, and the bottom border to six pixels.

You may also use the following to set only certain sides.

```
border-width: 2px;
```

Set all four sides to the size specified.

```
border-width: width1 width2;
```

Set the top and bottom border to width1 and the left and right borders to width2.

```
border-width: width1 width2 width3;
```

Set the top border to width1, set the left and right borders to width2, and set the bottom border to width3.

```
border-width: width1 width2 width3 width4;
```

Set the top border to width1, set the right border to width2, set the bottom border to width3, and set the left border to width 4.

There are also properties for the individual sides by sub-property.

- border-top-width
- border-right-width
- border-bottom-width
- border-left-width

Try it.

Size

The height and width style properties.

They can take an absolute value in *pixels*, *a relative value* (normally based on *em-units*, a typographic measure), or a *percentage* value.

If using a percentage, beware that some browsers treat it as a percentage of the table or available space after margins and others as a percentage of the page.

The recommended method for setting width is to specify an absolute width for some columns and have others that have no size defined and expand to fill in the extra space.

- When you specify a width, you specify a maximum width. This will only be overridden if the contents of the cell or table are too wide to fit in that space. For instance, while text will wrap to fit available space, if you set a cell width to 400 pixels and then try to put an image that is 600 pixels wide in it, then the cell will expand to fit the image.
- You only need to specify a width once in a given column. All the cells in a column are the same width unless you are spanning columns.
- It is generally not recommended you specify a height, but when you do, it sets the minimum height for that element. You only need to set height once in a row.

border-collapse

On most browsers, there is an automatic margin of 2 pixels inserted between each cell.

To remove the spaces between the cells,

```
table { border-collapse: collapse; }
```

The border-collapse property takes one of three values:

- *collapse* -- Put no space between adjacent cell borders.
- *separate* -- Put some space between adjacent cell borders. In this mode you can then use border-spacing to specify how much space.
- *inherit* -- Inherit this property.

Spacing and Padding

- *margins* is the space *between cells* in a table.
- *padding* is the extra space *inside a cell* between the contents and the border of the cell.

The *padding* property can take *one to four* values to specify the padding, or the space within the cells, the same rules apply as with specifying individual border sides above. *padding-left*, *padding-right, padding-top* or *padding-bottom* specify each individual side.

The *margin* property is a little more limited in its use. It can only be applied to the *table as a whole* specifying the same amount of spacing between every element in the table.

Aligning the table

To center a table on a page, set the margins of the table to auto. This will cover your bases for almost all newer browsers.

Aligning the contents within the cells

Contents can be aligned by applying attributes or properties to individual cells or to the entire row or table division (i.e., header, body, footer).

To align cell contents horizontally you can use the *text-align* property. It takes a value of *left, right, center, or justify.* Justification is not well-supported on the Web.

The default alignment for content cells is *left aligned*. The default alignment for header cells is *centered*.

You can also align cell contents vertically with the *vertical-align* property. The vertical-align property is not as well supported as the align property, but it is getting there.

It can take any of the following values.

- top. Aligns contents with the top of the cell.
- middle. Aligns contents with the middle of the cell. In other words, it centres it vertically.
- bottom. Aligns contents with the bottom of the cell.
- baseline. For right now, this can be treated the same as bottom, though it is slightly different.

The default vertical alignment on a cell of any kind is middle.

Using Tables for Layout

Many Web sites use <table> elements to design page layout. However, this is not recommended.

SPAN AND DIVISION

span_div.html

So far, styles are applied to individual page elements by associating style sheets with their tags. However, the following styling situations are not covered by this method.

- Applying a style sheet to the <p> tag does not give you the ability to select which "*substring*" of the paragraph to style. All characters in the paragraph receive the same styling.
- Apply the same style to a *consecutive group* of paragraphs. We can duplicate and apply the same in-line style sheet individually to the separate *<p>* tags to give them the same appearance. However, it would be more convenient to be able to apply the same style to the paragraphs as a group, without having to style them individually.

HTML provides "marker" tags, * * and **, to identify and isolate particular sections or subsections of a page to which styling can be applied.

The Tag

A ** tag is a container tag placed around text for the purpose of identifying *a string of characters*. The tag can enclose a single letter, a word, a phrase, a sentence, or any other substring of text.

```
<span>World of Warcraft (WoW)</span> is a class-based <span>
massively multiplayer online role-playing game (MMORPG)</span>
developed by Blizzard Entertainment.
```

A ** tag is nothing more than a *marker* to isolate text to which its style sheet can be applied. It has *no built-in formatting* characteristics of its own. It does not force a line break nor does it produce a blank space. Therefore, it can be used in-line, within the normal flow of text simply to style its enclosed content.

Like other tags, a style sheet can also be applied to this tag. Without a style sheet, the enclosed string appears similar to other text.

```
<span style="color:red">World of Warcraft (WoW)</span> is a
class-based <span style="color:blue">massively multiplayer online
role-playing game (MMORPG)</span> developed by Blizzard
Entertainment.
```

Two "styles" are isolated within ** tags. When this paragraph is displayed in the browser, the two words are rendered in their assigned colours.

The Tag

A * * (division) tag is a similar type of marker to the tag. Its purpose is to enclose and designate a *collection of page elements* for application of styling to the enclosed set.

The * * tag is called a "block-level" tag because it *encloses other tags* and, importantly, it forces a line break on the page. Because it creates a line break before and after the content it encloses, it cannot appear inside other tags as can the tag.

```
<div style="margin-left:30px; margin-right:30px; text-
align:justify; color:aqua; background:blue" >
   <p>The game ...
November 2004. The game was released in Europe on February 11th,
2005 with English, French and German language versions.</p>
   <p>On March 2... including Gamer's Pulse's Best of Show
award.</p>
</div>
```

The * * tag does not have any visible formatting characteristics of its own other than the fact that it creates a line break before and after its enclosed content.

STILL A PROBLEM

The following simple selectors set page-wide formatting for page margins, headings, paragraphs, and blockquotes. Anywhere one of these tags is encountered on the page, the browser applies the associated style.

```
  body       {margin:20px; color:black}
  h1         {color:blue; text-align:center}
  h2         {color:blue; text-align:left}
  p          {text-align:justify; text-indent:25px}
  blockquote {margin-left:20px; margin-right:20px; text-
align:justify}
```

Question: What other properties of the selectors can we change?

EXCEPTIONAL STYLING

stylingExample.html

If one or more of the paragraphs in the document require a different alignment or indention, or a particular heading may use a different style to emphasize it differently from other headings, then we need to designate *exceptional styling* *within* the embedded style sheet itself.

In-line style sheets *should not be used* to override embedded style declarations.

THE ID AND CLASS SELECTORS

In addition to setting a style for a HTML element, CSS allows you to specify your own selectors called "id" and "class".

ID Selectors

stylingExample_ID.html

One way to apply special styles to individual page elements is to provide the element with a *unique identifier*. The unique identifier, or *id selector*, is used to specify a style for a *single, unique element*. Within the page, special styling can be applied only to the element with that identifier.

The first step in styling this paragraph is to assign it an id value. *An id is a unique identifier* (alphabetic and numeric characters).

```
  body       {margin:20px; color:black}
  h1         {color:blue; text-align:center}
  h2         {color:blue; text-align:left}
  p          {text-align:justify; text-indent:25px}
  p#Special  {text-indent:0px; margin-left:25px; margin-
right:25px}
  blockquote {margin-left:20px; margin-right:20px; text-
align:justify}
```

With an id assigned to this <p> tag it can be designated for special styling by using an ID selector in the embedded style sheet.

Example

```
<p id="Special">Despite the genre's focus on multiplayer gaming,
AI-controlled characters are still common. NPCs and mobs that
give out quests or serve as opponents are typical mostly in
MMORPGs. AI-controlled characters are not as common in action-
based MMOGs.</p>
```

Styling is applied only to the selector with the designated (#) id value.

This type of tag identification and styling can be used for any tag. Just assign a unique id to the tag, and differentiate it from other tags of the same type by adding this *#id value* to the tag selector.

and

 or elements often appear multiple times on a page to isolate and style different portions of text and different collections of page elements, all of which usually require different styling.

ID selectors can be used to apply different styles to different and tags without committing them to one particular style.

```
<style>
  div#Justify {text-align:justify}
  span#Red    {color:red}
  span#Blue   {color:blue}
</style>
```

```
<div id="Justify">
<p>This paragraph is styled by inheriting the styling of its
container division which is formatted with the <span
id="Red">"Justify"</span> style.</p>

<p>This paragraph is also styled by inheriting the styling of its
container division. In addition, <span id="Blue">"Blue"</span>
styling is applied to one of its words.</p>
</div>
```

"General-purpose" ID selector

For even greater flexibility, an ID style *does not* have to be associated with a particular type of tag selector. A style sheet entry can include a "general-purpose" ID selector,

```
<style>
  #general {color:green;text-align:justify}
</style>
```

which can be associated with any tag by assigning this id value to it.

```
<p id="general">This paragraph is styled by inheriting the
styling of a general ID selector.</p>
```

```
<blockquote id="general">This blockquoted paragraph is also
styled by inheriting the styling of a general ID
selector.</blockquote>
```

It is not advisable to use ID selectors as general-purpose styling techniques.

ID selectors should be reserved for applying unique styles to a single, unique element (within the page) as given by its standard format, *selector#id*.

General-Purpose Class Selectors

stylingExample_class_gen.html

A style class named *.Offset* is defined and is assigned to any tag with the class="class" attribute.

```
<style>
  p       {margin:20px}
  .Offset {margin-left:60px; margin-right:60px}
</style>
```

```
<p class="Offset">This paragraph requires special styling
different from normal paragraphs on the page. It has left and
right margins of 60 pixels.</p>
```

A style class can be

- applied to any type of tag simply by assigning the class to the tag
- assigned to any number of tags.

Any paragraphs, divisions, blockquotes, or other tag types can reflect the above styling by assigning them to the *.Offset class*. Of course, the tag to which the class is applied must be receptive to the particular properties and values declared for the class.

Class Selectors

Similar to ID style, you may also associate the style with a particular type of tag selector. Any tag can be associated by assigning this class value to it.

To associate the class selector to a particular type of tags you many use the following

```
<style>
   h2.hilite {color:blue;}
   p.hilite  {color:red;}
</style>
```

GENERAL ID AND CLASS SELECTORS

The tags become carriers for many different styles simply by assigning different style classes to them.

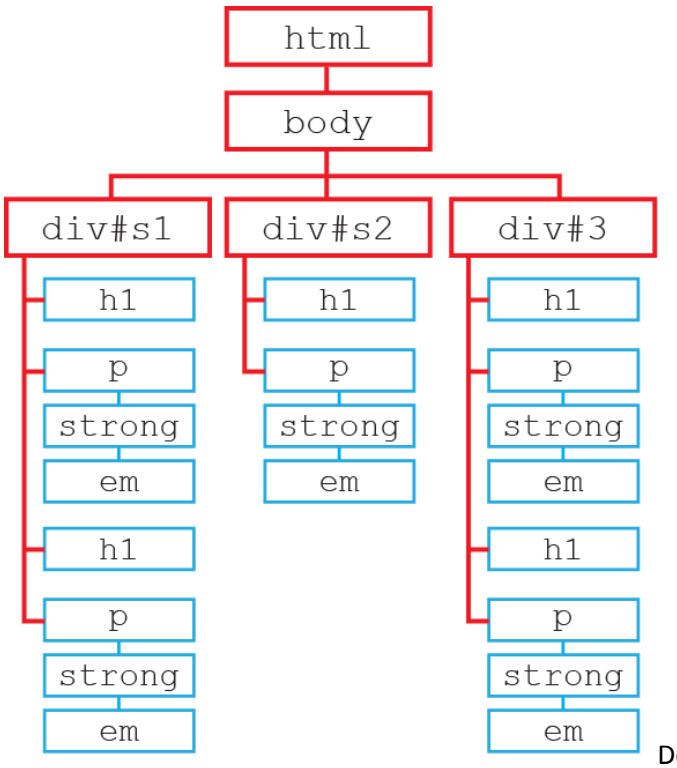
```
<style>
p#Special {
   text-indent:0px;
   margin-left:50px;
   margin-right:50px;
   font-weight: bold;
   background-color: #cdf;
   padding: 5px;
   border: 1px solid #999999;
}
span#Red { color:red }
span.Blue { color:blue }
#yellow {color:yellow }
.green {color:green}
</style>

<body>
<h1>World of Warcraft</h1>
<p ><!--- stylesheet for p is applied here --->
  <span id="Red">World of Warcraft (WoW)</span> is a class-based
<span class="Blue" massively multiplayer online role-playing game
(MMORPG)</span> developed by Blizzard Entertainment. </p>
<p class="Blue">Does this paragraph change to blue?</p>
<p id="Special"><!--- stylesheet with p#Special is applied here -
-->
  It is the 5th Blizzard game - not including expansion packs -
  if you take the cancelled Warcraft Adventures: Lord of the
Clans as being one of them,
  set in the <span class="green">Warcraft Universe</span>, a
fantasy setting introduced by Warcraft: Orcs & Humans in 1994.
</p>
<p class="green">Does this paragraph change to green?</p>
<p><span id="Red">World of Warcraft</span> is set four years
after the events at the conclusion of Blizzard's previous
release, Warcraft III: The Frozen Throne.
<h1>General information </h1>
<p>The game was simultaneously released on November 23, 2004 in
North America, Australia and New Zealand, on both PC and
Macintosh systems. </p>
</body>
```

The *.blue* class style is assigned to 2 different selectors. Since the *.blue* class is assigned to the element, the <p> element with the *.blue* class assigned will not inherit the styling.

The .green class style is a general class selector. Both the <p> and elements inherit the styling

DESCENDANT SELECTORS & SELECTOR PATTERNS

The descendant selector is a powerful way to target specific elements within specific areas of your HTML documents. Using a descendant selector, you could, for instance, easily target any *em* elements that reside within *p* elements. There is no need to create a separate class and then apply it to the tags. You can simply target them from the style sheet. There is a space between each element within the descendant selector.



Document Tree

- Parent and Child
  - body is the child of html
  - div is the child of body
  - div is the parent of h1 and p
- Siblings
  - h1 and p are siblings within the div
  - Descendants of div
- h1, div, p and em are all descendant of body
  - body is the ancestor of all elements
  - p, div, body and html are all ancestors of em

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>Example Document</title>
</head>
<body>
<div id="s1">
  <h1>Hickory Dickory Dock</h1>
  <p> Hickory, dickory, dock. </p>
  <p>The mouse ran up the clock. </p>
  <p>The clock struck one. </p>
  <p>The mouse ran down. </p>
  <p>Hickory, dickory, dock! </p>

  <h1> Mary had a little lamb </h1>
  <p> Mary had a little lamb. </p>
  <p>It's fleece was white as snow. </p>
  <p>And everywhere that Mary went
    the lamb was sure to go. </p>
</div> <!-- end of div#s1 -->

<div id="s2">
  <h1>Itsy Bitsy Spider</h1>
  <p>The itsy bitsy spider climbed up the water spout.</p>
  <p>Down came the rain
    and washed the spider out. </p>
  <p>Out came the sun
    and dried up all the rain. </p>
  <p>And the itsy bitsy spider
    went up the spout again!.</p>
</div><!-- end of div#s2 -->

<div id="s3">
  <h1> Jack and Jill</h1>
  <p>Jack and Jill went up the hill
    to fetch a pail of water. </p>
  <p>Jack fell down
    and broke his crown. </p>
  <p>And Jill came tumbling after!</p>

  <h1> Little Miss Muffet </h1>
  <p> Little Miss Muffet
    sat on a tuffet
    eating her curds and whey.</p>
  <p> Along came a spider
    and sat down beside her. </p>
  <p>And frightened Miss Muffet away! </p>
</div><!-- end of div#s3 -->
</body>
</html>
```

HTML document (descendent.html)