5.1 Introduction

We have learned how to structure the content of the documents using XHTML's wide variety of elements and attributes; we are now going to start making web pages look a lot more exciting. We are going to learn how to use *cascading style sheets* (or CSS for short) to take control of the style of the pages, including the colors and size of fonts, the width and colors of lines, and the amount of space between items on the page. The cascading style sheets specification works by specifying *rules* that say how the content of elements within the document should appear. For example, we can specify that the background of the page is a black color, the contents of all <p> elements should be displayed in red using the Arial font, and that all <h1> elements should be in white using the Times New Roman font.

5.2 What is CSS?

HTML was originally developed as a language to enable scientists to exchange information electronically with one another. The early versions of the language focused entirely on structural elements, such as paragraphs, headings, lists, and tables. Fortunately, HTML and the Web did not stay solely for long. As its popularity grew in the mid-1990s, better design control, such as specifying fonts and colors, adding borders, and laying out pages, became necessary. Adapting HTML to simultaneously fill both the needs of defining structure and defining presentation turned out to be close to impossible, to the point that many sites were horribly bloated with unnecessary code. Making matters worse, not all browsers supported all the presentational elements, forcing some designers to create multiple copies of their site for different browsers.

The World Wide Web Consortium, which oversees the development of HTML and other Web technologies, including XML, decided that the solution to the problem was to remove anything presentational from HTML, in essence taking it back to its original design goals, and instead created an entirely new language for the presentation of documents on the Web. Thus CSS (Cascading style sheets) was born.

DEFINITION CSS

CSS (cascading style sheets) is a language that applies styles to a HTML document and its elements to change the look and feel and is usually stored in separate .css style sheets which can be re-used for all the web pages. A website is made of HTML for content plus CSS for appearance.

HTML	+	CSS	=	WEB PAGE
(Content)		(Presentation)		

5.3 Advantages and Disadvantages of CSS

Advantages

- **CSS saves time** - We can write CSS once and then reuse same sheet in multiple HTML pages. We can define a style for each HTML element and apply it to as many Web pages as we want.
- **Pages load faster** - If we are using CSS, we do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply to all the occurrences of that tag. So less code means faster download times.

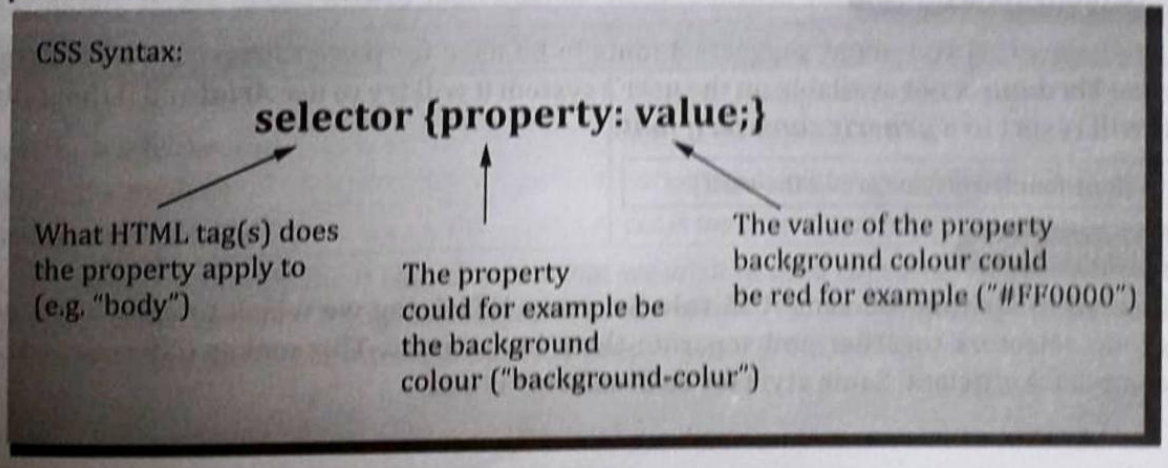
- **Easy maintenance** - To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Smaller file size**: Separating content from presentation helps keep HTML file sizes down and improves page loading times. Once CSS files have been loaded for the first time they will be cached by the browser and loaded instantly for all following pages that use the same style sheets.
- **Consistency**: With CSS it's easy to keep a consistent look throughout the website.
- **Accessibility**: Having content separated from presentation is a great help to people with viewing disabilities who often benefit from well structured HTML to understand the page.
- **Design without tables**: Before CSS lots of web designers mis-used tables to position elements on their page which is a bad thing for accessibility and also makes the code unnecessarily big and complicated. With CSS we can position any element exactly where we want without using tables.
- **Superior styles to HTML** - CSS has a much wider array of attributes than HTML so we can give far better look to the HTML page in comparison of HTML attributes.
- **Multiple Device Compatibility** - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** - Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

Disadvantages

- **Browser compatibility**: Browsers have varying levels of compliance with Style Sheets. This means that some Style Sheet features are supported and some aren't. To confuse things more, some browser manufacturers decide to come up with their own proprietary tags. Fortunately, browser compatibility is becoming less of an issue as the latest browser versions are much more standards-compliant than their earlier counterparts.

5.4 Basic CSS Syntax

The CSS syntax consists of a set of rules. These rules have 3 parts: a selector, a property, and a value. To keep in line with the *XHTML formatting rules* all CSS code should be written in **lowercase**.



- **Selector**: Defines which HTML tags are going to receive the CSS rules. To change the style of all the paragraphs we can write

```
p {property:value}.
```

- **Property**: This is the CSS style we wish to apply. To change the text color of all paragraphs we can write

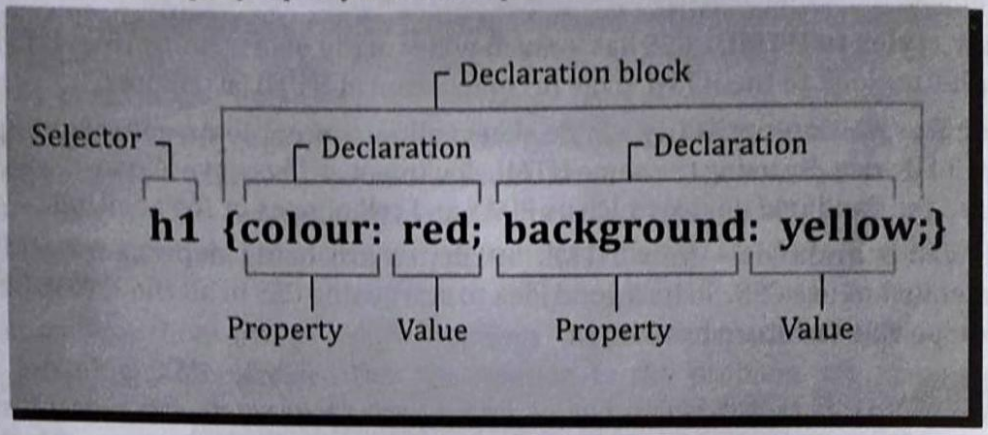
```
p {color:value}.
```

- **Value**: The value of the property. To make all paragraph text red we can write

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

Multiple property:value pairs

- We can create multiple property and value pairs as shown below



Space-separated values

- **Multiple values** can either be **space-separated** which means that all values are used for the property. Or they can be **comma-separated** which means that a series of values is being suggested and the first applicable one is used.

```
p {border:5px solid red}
```

Above CSS statement uses all 3 values to render paragraphs with a 5 pixel wide red border.

Comma-separated values

- The below CSS statement suggests 3 fonts to be used for paragraphs, from left to right. In case ***Verdana*** is not available on the user's system it will try to use ***Arial*** and if that fails, too, it will resort to a ***generic sans-serif font***.

```
p {font-family:verdana,arial,sans-serif}
```

Multiple selectors

- Instead of writing the same CSS rule for every HTML tag we whish to apply it to, we can group selectors together and separate them by commas. This makes CSS code extremely compact & efficient. Same style for each selector is bad

```
h1 {color:black}
h2 {color:black}
h3 {color:black}
h4 {color:black}
h5 {color:black}
h6 {color:black}
```

The Grouped selectors are good and it is shown below.

```
h1, h2, h3, h4, h5, h6 {color: black}
```

Contextual selectors

- If we just apply styles to a paragraph selector, all paragraphs will be affected. What if we wish to only change the paragraphs inside list items? In that case we need contextual selectors which only apply a CSS rule if a tag occurs in a certain context. A contextual selector is created by listing the tags in the nested order in which they should appear in the document.

```
li p {color:red}
h1 em {background:gray}
```

In above example only paragraph elements <p> inside list items <li> get red text and only emphasized text in <h1> headings gets a gray background.

Comments

- Sometimes it can be helpful to put CSS comments next to some statements to explain what they're doing. These comments are ignored by web browsers and it is used only for readability and maintainability.

```
/* this is a CSS comment */
p {font-weight:bold} /* this makes all paragraphs bold */
```

Example: To write the style rules so all H1 headings are rendered in red text, we will create the rule in the style sheet like this:

```
body { background-color: red;}
```

Let's compare HTML to CSS to help you visualize the similarities and differences.

XHTML Example	CSS Example
<body bgcolor="red">	body {background-color: red;}
body is the HTML element	body is the CSS selector
bgcolor is the attribute	background-color is the property
red is the attribute value	red is the property value

The CSS ***selector*** is merely the HTML element that we wish to set a rule for. The HTML ***attribute*** and CSS ***property*** are the same thing, although the actual terms (bgcolor vs. background-color) can be different. And both the HTML attribute and CSS property are given a value.

Write the basic CSS syntax.

The syntax is : selector {property:value}

What is grouping?

Grouping is gathering two or more selectors that share the same style.

Example: h1, h2, h3, h4, h5, h6 {color: black}

How to add Comments in CSS?

The comments are added using /* and */

What are Contextual selectors?

Contextual selector is a selector that addresses specific occurrence of an element. It is a string of individual selectors separated by white space, a search pattern, where only the last element in the pattern is selected for styling.

5.5 Levels of Style Sheets

CSS can be added to web pages at 3 different levels. They are ordered from lowest level to highest level.

1. Inline Styles
2. Internal or Document Styles
3. External Styles

Inline: It is possible to create CSS styles that only work for the single tag it is defined for. Single tag CSS is used when the style is used in a single place on the entire site.

Internal: Usually a certain style appears more than once in the web page, and thus we should use the internal technique: adding styles that are defined once for the entire page.

External: If, however, that certain style is used on more than a single page, we should use the third - and most powerful - technique called external: adding styles that are defined once for the entire site.

What are the different levels of CSS? Or What are the different ways of applying a style to XHTML element?

There are three ways of applying style to XHTML element; They are Inline, Internal or Document, and External.

5.5.1 Inline Styles

- The **simplest** and most **direct** way of applying CSS to an element is to write it into the tag itself as a **style attribute** `style="..."`. CSS styles applied this way have the highest priority over other style definitions, i.e. they will **overwrite** existing styles.
- We are applying CSS to a XHTML element directly and hence a **selector is not necessary**.
- To use inline styles we use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:
- **Example 1**: In the below code, we have used style attribute for <b> (bold) tag. This tells the browser to display "**Web Programming**" in bold and font-size 16px.

```
<b style="font-size:16px;">Web Programming</b>
```

- **Example 3:** In the below example, we have used style attribute for <p> tags. It is noticeable that there are two paragraphs with different styles. The first text "**My first paragraph with style**" will be displayed in red color and keeping 20 pixels left margin. The second text "**My second paragraph with style**" will be displayed in yellow color and keeping 40 pixels left margin.

```
<p style="color: red; margin-left: 20px">
My first paragraph with style
</p>
<p style="color: yellow; margin-left: 40px">
My second paragraph with style
</p>
```

IMPORTANT TO KNOW

- Inline styles apply only to the particular element in which they appear.
- Remember that we're trying to separate content from presentation? Inline styles are the exact opposite. They're inefficient because they'll combine up the code and presentation and are difficult to maintain. Only use them for some quick testing of CSS styles or for exceptions where a style is only used at a particular place and nowhere else on the homepage.
- Furthermore, if we wanted to change a certain style, we have to change it all over in the document, rather than in one place.
- The purpose of inline styles is to override an internal or external style rule, or to make a quick change of style where we don't want a reusable rule. For example, we may have an external rule that specifies all **h1** headings should be the color **blue**. If we have a single place on our page where we want an **h1** heading in **red** and all other **h1** should be in **blue** itself, an inline style rule allows overriding an internal or external style rule.

Example: Let us create an XHTML document with different paragraphs of different font styles and sizes

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5    <head>
6      <title>Thanks Message</title>
7    </head>
8    <body>
9      <p style="font-size: xx-small;"> This is paragraph one with font size xx-small</p>
10     <p style="font-size: x-small;"> This is paragraph two with font size x-small</p>
11     <p style="font-size:small;"> This is paragraph three with font size small</p>
12     <p style="font-size: medium;"> This is paragraph four with font size medium</p>
13     <p style="font-size: large;"> This is paragraph five with font size large</p>
14     <p style="font-style: italic;"> This is paragraph six with font style italic</p>
15     <p style="font: bold; font-family: cursive;"> This is paragraph seven with font bold and font family cursive</p>
16   </body>
17 </html>
```

How to apply style using Inline Styles Method?

The inline styles are applied to each individual element using style attribute of the tag.

Example: `<b style="font-size:16px;">Web Programming</b>`

What are the disadvantages of Inline Method?

Applying style to each individual element increases the code and it is difficult to maintain.

When Inline is useful?

Whenever we want to override the rule applied in Internal or External for a particular element.

5.5.2 Internal or Document Styles

- An internal style sheet should be used when a single document has a unique style. When applying internal **CSS** we put all CSS rules in the <head>...</head> part of the XHTML document and enclose it with <style type="text/css">...</style> tags as shown below.

```
<head>
        <style type="text/css">
              h1 {color: red;}
              p {color: yellow; }
              body {background-color: black; }
        </style>
</head>
```

The above code says that, all **h1** headings should be styled with **red** color, all **paragraphs** should be styled with **yellow** color, and **body** of the document must be **black** color. The browser will now read the style definitions, and format the document according to it.

- Similar to an inline style rule, embedded styles allows overriding the rules of an external style sheet. The difference is that with an internal rule, we don't have to create a rule with each tag of an XHTML element. An h1 heading given the color red in an internal style rule will render the h1 heading in red every time we use it on the page without having to code the rule into each heading tag as we must do with inline rules or with XHTML attributes.

IMPORTANT TO KNOW

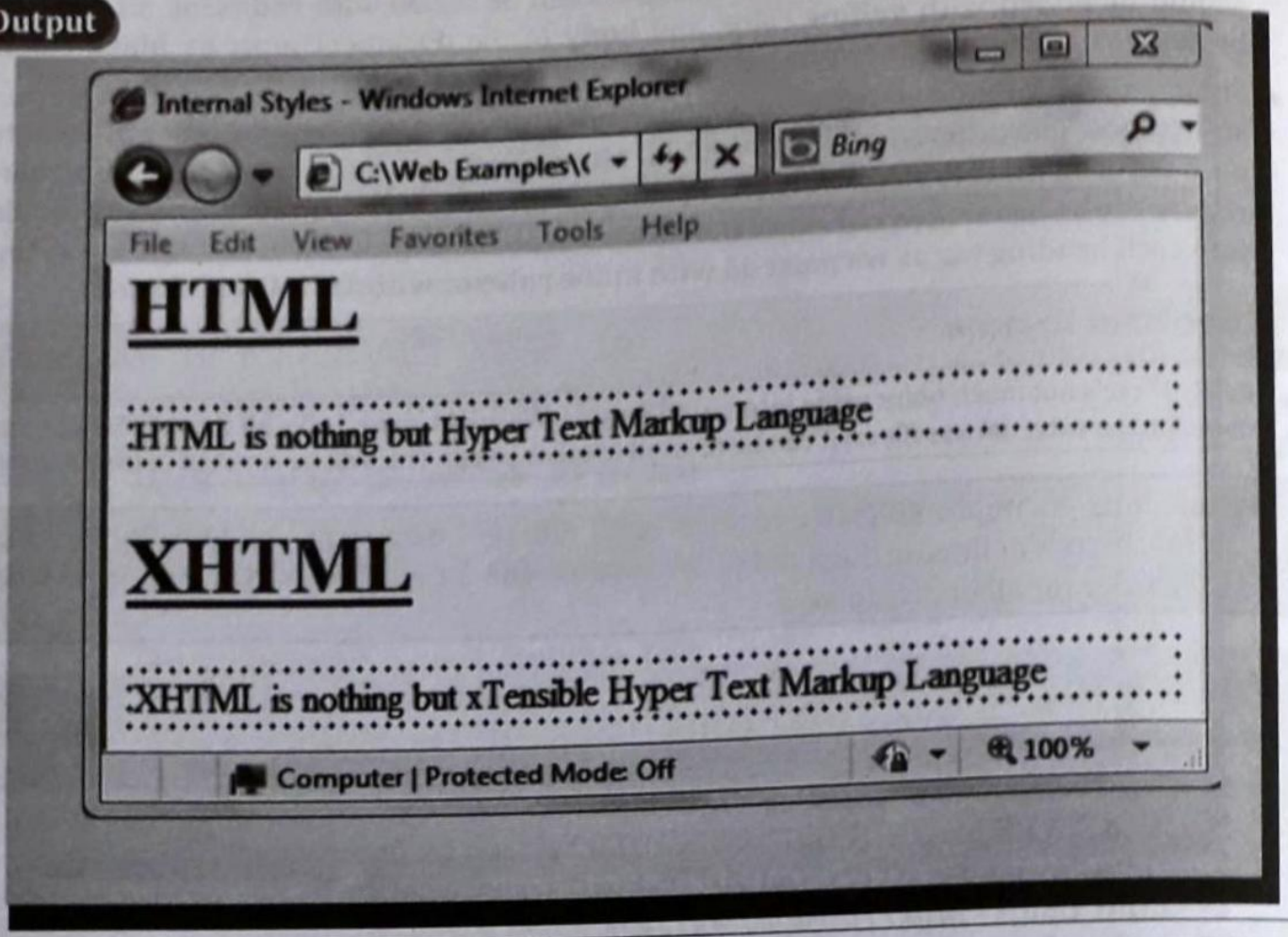
- Really there's not much point using internal CSS styles. We might as well take all those CSS rules and put them into an external .css file so it can be re-used in other pages. This will be covered in the next section.

Example: Let us create an XHTML document with different paragraphs and headings using the same style defined in the style tag. We have used underline for all <h1> headings and dotted border for all paragraphs.

InternalStyles.html

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6          <title>Internal Styles </title>
7          <style type="text/css">
8              p {border-style: dotted;}
9              h1 {text-decoration: underline;}
10         </style>
11     </head>
12     <body>
13         <h1> HTML</h1>
14             <p> HTML is nothing but Hyper Text Markup Language</p>
15         <h1> XHTML</h1>
16             <p> XHTML is nothing but xTensible Hyper Text Markup Language</p>
17     </body>
18 </html>
```

Output



What is Internal or Document Styles?

The Internal or Document Level Styles are used to define the style for whole page. This works only for the page it is defined.

How to apply style using Internal Styles?

To apply internal CSS, we need put all CSS rules in the <head>...</head> part of the XHTML document and enclose it with <style type="text/css">...</style> tags as shown below.

```
<head>
    <style type="text/css">
        h1 {color: red;}
        p {color: yellow; }
        body {background-color: black; }
    </style>
</head>
```

When Internal Styles are useful?

Whenever we want to override the rule applied in External Sheets for a particular web page. For example, If we have 10 pages in the web site, all 9 pages require same style and 1 page require different style, then it is useful.

5.5.3 External Styles

- The true power of CSS is separating presentation from content. **External stylesheets** are the tool for doing exactly this. An external style sheet is ideal when the style is applied to many pages.
- CSS can be defined for entire sites by simply writing the CSS definitions in a plain text file (file with extension .css) that is referred to from each of the pages in the site.Rather than writing the entire CSS definition on each page, as in the previous examples, we can write it to a text file that is only loaded on the first page that a visitor sees at web site.

  When the visitor jumps to other pages, the CSS text file will be cached and thus doesn't have to be transferred via the internet for subsequent pages. This means that web pages will load faster while at the same time we will have extreme flexibility to change the style for the entire site even after it has been made.
- By putting all the CSS rules into a separate **.css file** (a simple text file) we can **re-use** them for all the pages without rewriting. All the styles are in a centralized place making it **easy to maintain** and change the look of the website.
- To **include** one or more style sheets into the XHTML document we can use the

  `<link ... />` element inside the HTML `<head>`.
- **For example**, let's say that the style sheet is named **style.css** and is located in a folder named **style**. The situation can be illustrated like this:

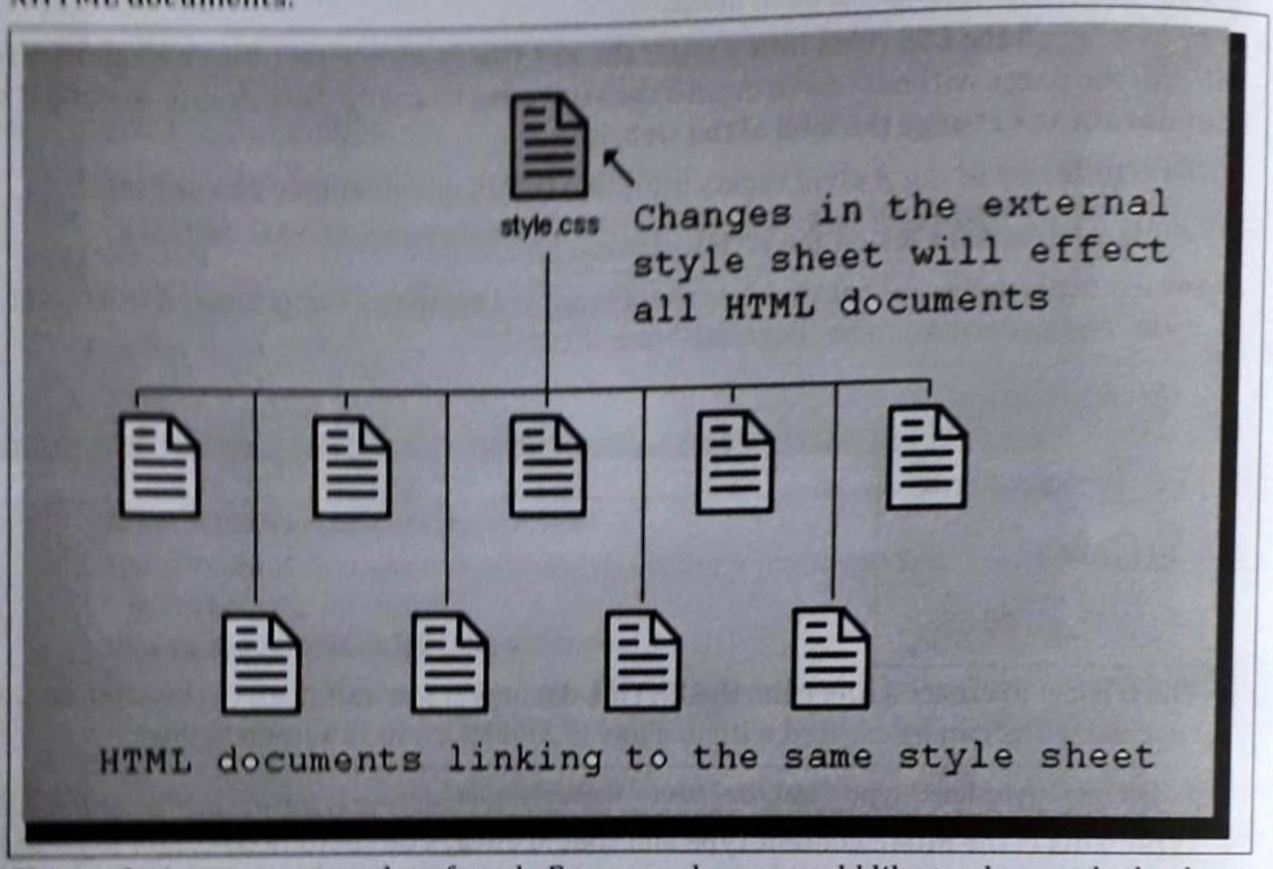
- The trick is to create a link from the XHTML document (default.htm) to the style sheet (style.css). Such link can be created with one line of XHTML code as shown below.

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

- **type**: This is the MIME content-type and should always be set to text/css for stylesheets.
- **rel**: This attribute describes the relationship between the actual document and the one we are linking to. In this case it's always stylesheet.
- **href**: This is the link to the stylesheet.
- Notice how the path to our style sheet is indicated using the attribute href. The line of code must be inserted in the header section of the XHTML code i.e. between the `<head>` and `</head>` tags. Like this:

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

- This link tells the browser that it should use the layout from the CSS file when displaying the XHTML file. The really smart thing is that several XHTML documents can be linked to the same style sheet. In other words, one CSS file can be used to control the layout of many XHTML documents.



- This technique can save a lot of work. For example, we would like to change the background color of a website with 100 pages, a style sheet can save from having to manually change all 100 XHTML documents. Using CSS, the change can be made in a few seconds just by changing one code in the single style sheet.

IMPORTANT TO KNOW

1. **Use as much as possible**: Try to put all your CSS styles into stylesheets and minimize your **inline CSS** and **internal CSS**.
2. **Use Comments in CSS**: It's good practice to comment the code so that we can remember and others can understand what it's doing. Don't go overboard with adding **CSS comments** though, as it will increase the file size unnecessarily. Just comment the main sections of the CSS file and any special rules that are not easily understood.

3. **Use *F5* to display recent changes**: When we are making changes to the stylesheets we have to press the *F5* key (sometimes even *CTRL+F5* which causes a more thorough refresh in some browsers) to force the web browser to refresh the page and load the newest version instead of using the cached one.

Example: Let us create simple CSS file to define the styles for headings, body and paragraphs. And link the CSS file to an XHTML document.

Step 1: Open Notepad and type the below code and save it as MyStyle.css

MyStyle.css

```
1  body   { background: silver;   }
2  h1     { border: thick, background-color: gray;   }
3  h2     { border: medium, background-color: yellow;  }
4  p      { border: thin,text-decoration: underline;   }
```

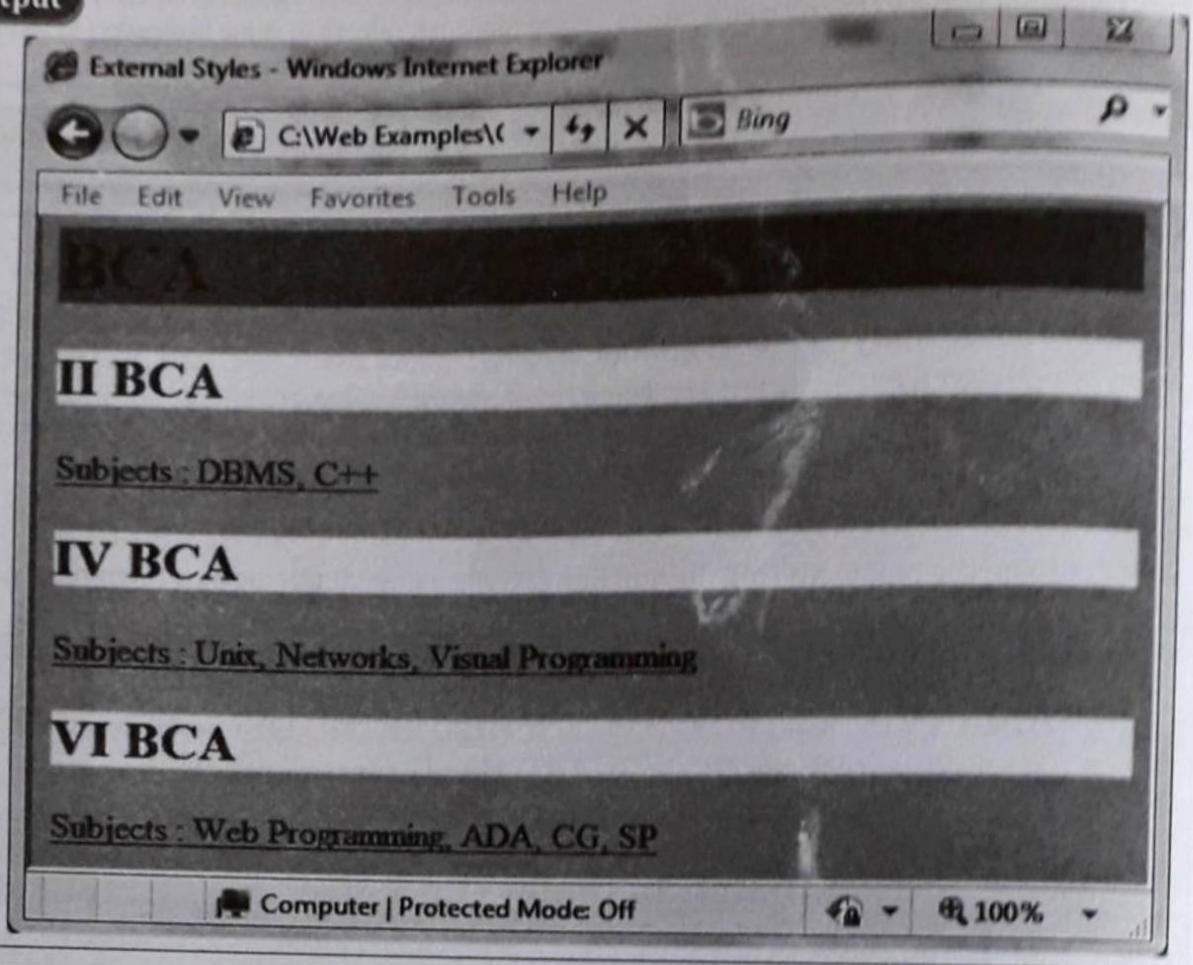
Step 2: Create the below XHTML file and link CSS file using <link> tag as shown below.

ExternalStyles.html

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml">
5      <head>
6        <title>External Styles </title>
7        <link rel="stylesheet" type="text/css" href="MyStyle.css" />
8      </head>
9      <body>
10       <h1> BCA</h1>
11           <h2> II BCA</h2>
12               <p> Subjects : DBMS, C++</p>
13           <h2> IV BCA</h2>
14               <p> Subjects : Unix, Networks, Visual Programming</p>
15           <h2> VI BCA</h2>
16               <p> Subjects : Web Programming, ADA, CG, SP</p>
17     </body>
18 </html>
```

If we would like to use the same CSS file for other pages, then we need to link CSS file in all other web pages.

Output



IMPORTANT TO KNOW

It is highly suggested to use external style sheets approach. We will use the same approach in rest of the chapter.

What are External Style Sheets?

The External style sheets can be defined for entire web pages by simply writing the CSS definitions in a plain text file (file with extension .css) that is referred to from each of the pages in the web site.

How to link external style sheet in the web page?

The link can be created with one line of XHTML code as shown below.

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

What are the attributes of <link> tag?

The attributes are rel, type, and href.

5.6 Importing External Style Sheets

- We have already seen how to link external style sheet into XHTML document using <link> tag. The import is another way of including the external sheets into XHTML document. The general syntax is :

How to import external style sheet?

The external style sheet can be imported using either <link> tag or @import. The example is:

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

What are the advantages/disadvantages of the various style methods?

External Style Sheets

Advantages

* Can control styles for multiple documents at once
* Classes can be created for use on multiple XHTML element types in many documents
* Selector and grouping methods can be used to apply styles under complex contexts

Disadvantages

* An extra download is required to import style information for each document
* The rendering of the document may be delayed until the external style sheet is loaded
* Becomes slightly unwieldy for small quantities of style definitions.

Internal Style Sheets

Advantages

* Classes can be created for use on multiple tag types in the document
* Selector and grouping methods can be used to apply styles under complex contexts
* No additional downloads necessary to receive style information

Disadvantages

* This method cannot control styles for multiple documents at once

Inline Styles

Advantages

* Useful for small quantities of style definitions
* Can override other style specification methods

Disadvantages

* Does not separate style information from content (a main goal of SGML/HTML)
* Cannot control styles for multiple documents at once
* Developer cannot create or control classes of elements to control multiple element types within the document
* Selector grouping methods cannot be used to create complex element addressing scenarios

5.7 STYLE SPECIFICATION FORMATS

We have already seen how to use style specification format in Inline, Internal, and External style sheets. Let us look into the general structure of using style specifications.

How to apply style to XHTML element?

Using style attribute of a tag as shown below.

```
<h1 style="color:yellow;"> This is inline CSS </h1>
```

Which tag is used to apply styles to whole page?

Using <style> tag as shown below.

```
<head>
    <style type="text/css" media="all">
        h1{ color: yellow; }
    </style>
</head>
```

What are the attributes of <style> tag?

The type and media are the two attributes of <style> tag.

What is the MIME type for CSS?

text/css

What is the format for external style sheet?

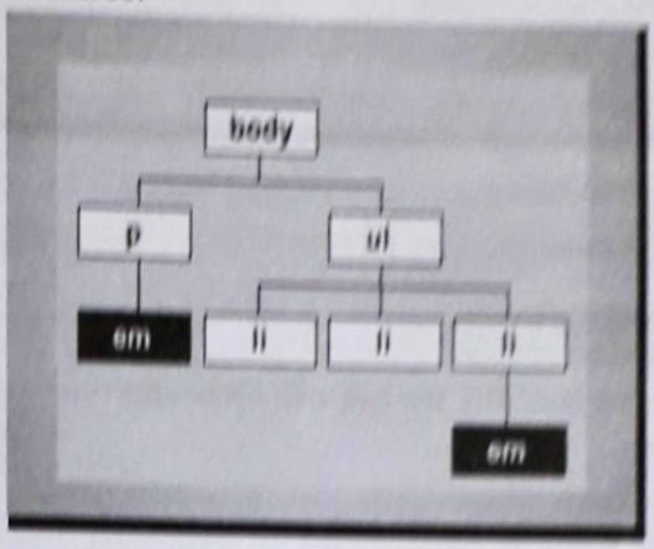
selector {property_1:value_1; property_2:value_2;----; property_n:value_n;}

5.8.1 Type Selectors

- The most common and easy to understand selectors are type selectors. Type selectors will select any HTML element on a page that matches the selector, regardless of their position in the document tree. For example:

```
em {color: blue; }
```

- This rule will select any element on the page and color it blue. As we can see from the document tree diagram below, all elements will be colored blue, regardless of their position in the document tree.



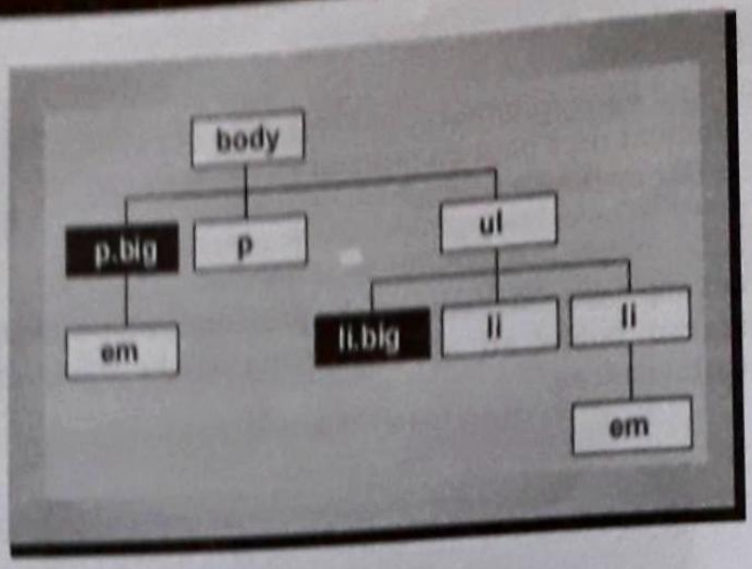
- There are a huge range of elements that we can select using type selectors, which means we can change the appearance of any or every element on the page using only type selectors.

5.8.2 Class Selectors

- While type selectors target every instance of an element, class selectors can be used to select any XHTML element that has a class attribute, regardless of their position in the document tree.
- For example, if we want to target the first paragraph and first list items on a page to make them stand out, we could mark up the page in the following way:

```
<body>
    <p class="big">This is some <em>text</em></p>
    <p>This is some text</p>
    <ul>
        <li class="big">List item</li>
        <li>List item</li>
        <li>List <em>item</em></li>
    </ul>
</body>
```

- The tree diagram would be:



- The rule used could then be:

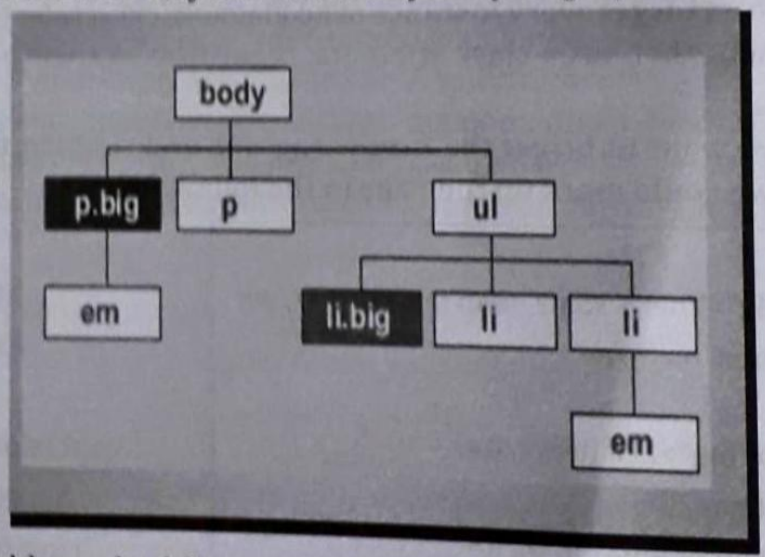
```
.big { font-size: 110%; font-weight: bold; }
```

Combining class and type selectors

- If we want to be more specific, we can use class and type selectors together. Any type selectors can be used.

```
div.big { color: blue; }
td.big { color: yellow; }
label.big { color: green; }
form.big { color: red; }
```

- For example, we may want to target the first paragraph and first list items on a page to give them a larger font size. We may also want only the paragraph to be bold.



- To do this, we could use the following rules:

```
.big { font-size: 110%; }          /* affects p and li */
p.big { font-weight: bold; }       /* affects p only */
```

- Perhaps the most powerful aspect of class selectors is that multiple classes can be applied to one XHTML element. For example, we may wish to use two rules on one particular element like this:

```
<p class="big indent">                    /* There is a space between big and indent */
.big { font-weight: bold; }
.indent { padding-left: 2em; }
```

- A Class Selector is indicated by a period (.).
- For instance, **.bold_text** is a class selector.
- Class Selectors must begin with a period followed by a letter. Cannot use a number immediately after the period.

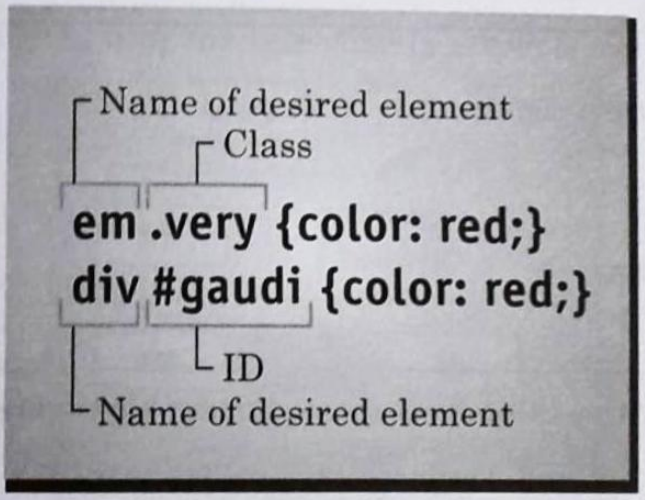
- A Class Selector can be used as many times as needed on a single web page.
- We are required to use the period at the beginning, but we can use any words we would like to name the selector.
- It is common for class selectors to be used to style only a portion of an element. However, they can also be used to style any element that will occur on a web page more than one time.

5.8.3 ID Selectors

- ID selectors are similar to class selectors. They can be used to select any HTML element that has an ID attribute, regardless of their position in the document tree. Examples of ID selectors:

```
#navigation { width: 12em; color: #333; }
div#navigation { width: 12em; color: #333; }
```

- ***An ID Selector is indicated by using the pound # sign***. Use an ID Selector for page elements that occur one time on a web page. In other words, ID Selectors must be unique. A perfect example is the Layout. Use an ID Selector for each div of the CSS Layout
- The major difference is that IDs can only be applied once per page, while classes can be used as many times on a page as needed. Look at the way of using class and ID selectors in the below diagram.

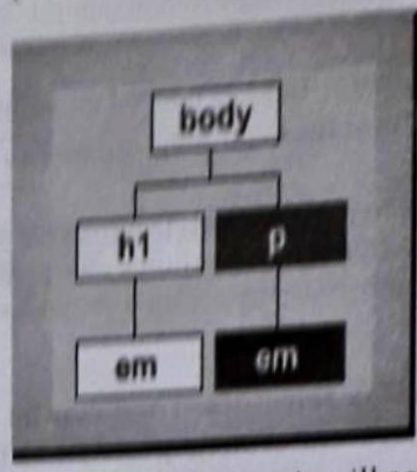


5.8.4 Descendant selectors

- Descendant selectors are used to select elements that are descendants of another element in the document tree.
- For example, we may wish to target a specific element on the page, but not all elements. A sample document could contain the following code:

```
<body>
    <h1>Heading <em>here</em> </h1>
    <p>Skyward <em>Publishers</em> Books.</p>
</body>
```

- The document tree diagram (with the element to be targeted) would be:



- If we use a type selector like the example below, it will select all elements on the page:

```
em {color: blue; }
```

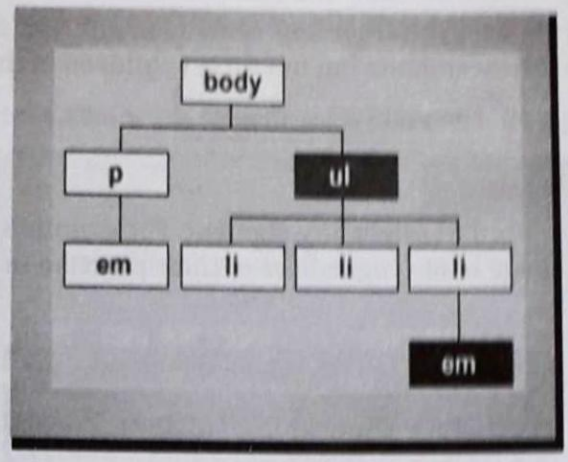
- However, if we use a descendant selector, we can refine the elements that we select. The rule below will only select elements that are descendants of <p> elements. If this rule is applied, the element within the <h1> will not be colored blue.

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

- We can also jump levels in the document tree structure to select descendants. For example, the following code:

```
<body>
<p>Skyward <em>Publishers</em> Books.</p>
    <ul>
        <li>item 1</li>
        <li>item 2</li>
        <li><em>item 3</em></li>
    </ul>
</body>
```

- The document tree (with a third-level element highlighted) would be:



- Using the following rule we can isolate any element inside a <ul> element, without having to describe the <li> element. If this rule is applied, any element within a <ul> element will be colored blue. However, the element within the <p> will not be colored blue:

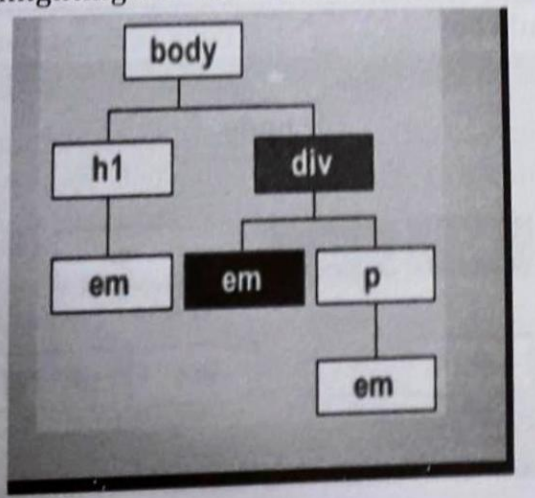
```
ul em {color: blue; }
```

5.8.5 Child selectors

- A child selector is used to select an element that is a direct child of another element (parent). Child selectors will not select all descendants, only direct children.
- For example, we may wish to target an that is a direct child of a , but not other elements that are descendants of the . A sample document could contain the following code:

```
<body>
        <h1>Heading <em>text</em></h1>
        <div>
            This is some <em>text</em>
            <p>This is a paragraph of <em>text</em></p>
        </div>
</body>
```

- The document tree (highlighting the that is a child of the) would be:



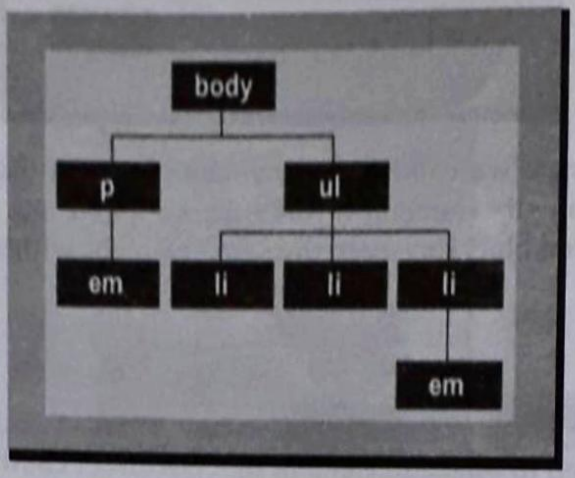
- Using the following rule we can target any element that is a child of the . Other elements that are descendants but not direct children of the will not be targeted.

```
div > em { color: blue; }   OR   div>em { color: blue; }
```

5.8.6 Universal selectors

- Universal selectors are used to select any element. For example, the rule below will color all XHTML elements on a page blue - regardless of their position in the document tree.

```
* {color: blue; }
```

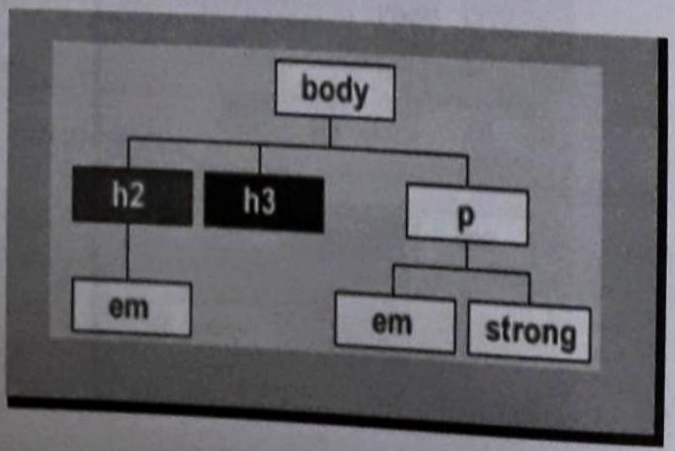


5.8.7 Adjacent sibling selectors

- Adjacent sibling selectors will select the sibling immediately following an element.
- For example, we may wish to target an <h3> element, but only <h3> elements that immediately follow an <h2> element. This is a commonly used example, as it has a real-world application. There is often too much space between <h2> and <h3> elements when they appear immediately after each other. The code would be:

```
<body>
    <h2>Heading 2 <em>text</em></h2>
    <h3>Heading 3 text</h3>
    <p>This is <em>text</em> and more <strong>text</strong></p>
</body>
```

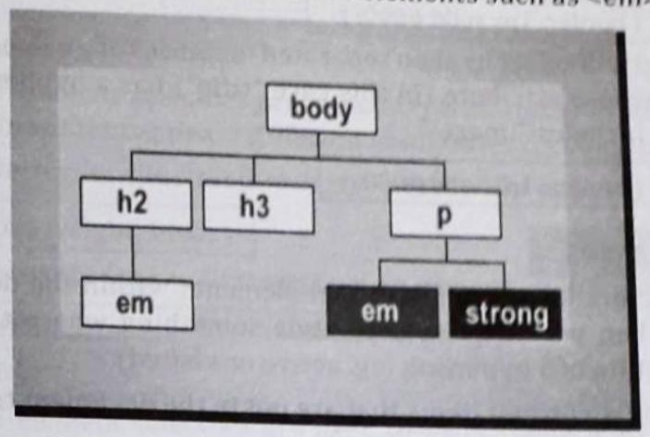
- The document tree would be:



- Using the following rule, we can target any <h3> that follows an <h2>:

```
h2 + h3 {margin: -1em; }
```

- Adjacent sibling selectors also work for inline elements such as and



- Using the following rule, we can target text that follows text:

```
em + strong {color: blue; }
```

5.8.8 Attribute selectors

- We know that all XHTML elements can have associated properties, called attributes. These attributes generally have values. Any number of attribute/value pairs can be used in an element's tag - as long as they are separated by spaces. They may appear in any order. In the example below, the code segments highlighted in bold are attributes .

```
<h1 id="section1">
<img src="small.gif" width="100" height="100">
<img title="mainimage" alt="main image">
<a href="foo.htm">
<p class="maintext">
<form style="padding: 10px">
```

- Attribute selectors are used to select elements based on their attributes or attribute valu For example, we may want to select any image on an XHTML page that is called "small.gi This could be done with the rule below that will only target images with the chosen name

```
img[src="small.gif"] { border: 1px solid #000; }
```

- There are four types of attribute selectors.
- The first option is to select based on attribute. The example below will select an element this case "img") with the relevant attribute. Examples could include:

```
img[title] { border: 1px solid #000; }
img[width] { border: 1px solid #000; }
img[alt] { border: 1px solid #000; }
```

- The second option is to select based on value. The example below will select any image wh attribute (in this case "src") has a value of "small.gif".

```
img[src="small.gif"] { border: 1px solid #000; }
```

What do you mean by Selector?

In CSS, a selector is the target element or elements to which each CSS rule is applied

What are the different types of Selectors?

The different types of selectors are:

- Type selectors
- Class selectors
- ID selectors
- Descendant selectors
- Child selectors
- Universal selectors
- Adjacent sibling selectors
- Attribute selectors
- Pseudo-classes
- Pseudo-elements

What is ID selector?

ID selector is an individually identified (named) selector to which a specific style is declared. Using the ID attribute the declared style can then be associated with one and only one XHTML element per document as to differentiate it from all other elements. ID selectors are created by a character # followed by the selector's name.

```
#MyPara {color: red; background: black}
<p id="MyPara">This and only this element can be identified as MyPara </p>
```

What is attribute selector?

Attribute selector is a selector defined by 1) the attribute set to element(s), 2) the attribute and value(s), 3) the attribute and value parts.

What is class selector?

The class selector is a "stand alone" class to which a specific style is declared. Using the class attribute the declared style can then be associated with any XHTML element. The class selectors are created by a period followed by the class's name.

What is the difference between ID and CLASS?

ID identifies and sets style to one and only one occurrence of an element while class can be attached to any number of elements.

What is attribute selector?

Attribute selector is a selector defined by 1) the attribute set to element(s), 2) the attribute and value(s), 3) the attribute and value parts.

What is class selector?

The class selector is a "stand alone" class to which a specific style is declared. Using the class attribute the declared style can then be associated with any XHTML element. The class selectors are created by a period followed by the class's name.

What is the difference between ID and CLASS?

ID identifies and sets style to one and only one occurrence of an element while class can be attached to any number of elements.

5.9 CSS Units of Measurement (Property Value Forms)

CSS includes many properties in seven categories: fonts, lists, text, margins, colors, backgrounds, and borders. Every property has a value and value can include some measurements like %, cm, mm etc, Let us discuss some of the measurement units of CSS. CSS supports a number of measurements including absolute units such as inches, centimetres, points, and so on, as well as relative measures such as percentages and em units. We need these values while specifying various measurements in the Style rules. **Example**: border="1px solid red".

Number Values

- **There are two types of numbers in CSS**: integers ("whole" numbers) and reals fractional numbers). These number types serve primarily as the basis for other value types, but, in a few instances, raw numbers can be used as a value for a property.
- **The following are all valid number values**: 10.5, -980.00004, and 5. Both integers and reals may be either positive or negative, although properties can (and often do) restrict the range of numbers they will accept.

Percentage Values

- Percentage units are always relative to another value, often the element's font size or the size of the parent element.100% takes the current value unchanged, 200% doubles it and so on. Percentages are set by appending a %.

Example:	p {font-size:200%}

Length Values

- CSS supports many different ways of measuring the length of an element. Some are absolute length units (such as inches, centimeters, or points) and others are relative length units (such as pixels and em units). The table below shows all of the different length units available in CSS and what each one means.

Unit	Description	Example
em	1em is equal to the current ont size. 2em means 2 times the size of the current font. E.g., if an element is displayed with a font of 12 pt, then `2em' is 24 pt. The `em' is a very useful unit in CSS, since it can adapt automatically to the font that the reader uses.	p{ font-size: 1.5em; }
ex	x-height (height of the lowercase letter x). The x-height is determined by the height of the font's lowercase letter x.	p { font-size: 2ex; }
px	pixcels (a dot on the computer screen).	p { font-size: 13px; }
in	defines a measurement in inches.	p { font-size: 0.15in; }
cm	Defines a measurement in centimeters.	p {font-size: 1cm; }
mm	Defines a measurement in millimeters.	p { font-size: 4mm; }
pc	Picas (1 pica = 12 points, 1 inch = 6 picas)	p { font-size: 1pc; }
pt	Points (1 point = 1/72 inches)	p { font-size: 12pt; }

Color Values using Color Keywords

- Colors can be specified using names such as black, white, red etc. For example:

```
body { color: white; background-color: Sienna; }
h1   { color: DarkGreen; }
```

Color Values using RGB Values

- The best way to specify colors in CSS is using RGB values, which ensures that the exact colour will be used. Colors can be specified in any of the following ways:

Unit	Description	Example
#RRGGBB	Standard XHTML red-green-blue 6-digit hex number	color: #FF0000
#RGB	Short-hand red-green-blue 3-digit hex number: each digit is replicated to produce the final colour, e.g. #123 -> #112233	color: #0CC
rgb(R, G, B)	Decimal representation, each value being an integer between 0 and 255	color: rgb(255,128,255)
rgb(R%, G%, B%)	Percentage representation, each value being a percentage between 0.0% and 100.0%	color: rgb(40%,70%,40%)

URLs

- A URL value is given by url(foo), where foo is the URL. The URL may be optionally quoted with either single (') or double (") quotes and may contain whitespace before or after the (optionally quoted) URL.