**CSS (Cascading Style Sheets)**

* CSS stands for cascading style sheets.
* Developed in 1997.
* Define the **look and feel** of Web pages.
* It is used to separate style from content.

A CSS rule has two main parts: a *selector* and one or more *declarations*. **Selector** is normally the HTML element you want to style and each **declaration** consists of a *property* and *value*. The **property** is the style attribute we want to use and each property has a **value** associated with it.

**Example:**

p {color:red;text-align:center;}

**Inserting CSS**

We can use style sheets in three different ways in out HTML document. There are **external style sheet**, **internal style sheet** and **inline style**.

**External Style Sheet**

If we want to apply the same style to many pages, we use external style sheet. With an external style sheet, you can change the look of an entire Web site by changing one style sheet file. Each page must link to the style sheet using the **<link>** tag. The <link> tag goes inside the head section. For example,

<head>

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

An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a **.css** extension. An example of a style sheet file is shown below:

hr {color:red;} p {margin-left:20px;} /\*Do not leave space between property value and units\*/ body {background-image:url("images/back40.gif");}

If you want a unique style to a single document, an internal style sheet should be used. You define internal styles in the head section of an HTML page, by using the **<style>** tag. For example,

<head>

<style type="text/css">

hr {color:red;}

p {margin-left:20px;} body {background-image:url("images/back40.gif");}

</style> </head>

**Inline Styles**

If you want a unique style to a single element, an inline style sheet should be used. An inline style loses many of the advantages of style sheets by mixing content with presentation. To use inline styles you use the **style attribute** in the relevant tag. The style attribute can contain any CSS property. For example,

<p style="color:yellow; margin-left:20px">This is a paragraph.</p>

**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 "\*/".

**Id and Class Selectors**

The **id** selector is used to specify a style for a single, unique element. The id selector uses id attribute of the HTML element and is defined with **“#”**. For example,

<head>

<style type="text/css">

#para1

{

text-align:center;

color:red;

}

</style>

</head>

<body>

<p id="para1">Hello World!</p>

<p>This paragraph is not affected by the style.</p>

</body>

The **class** selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements. This allows you to set a particular style for any HTML elements with the same class. The class selector uses the HTML class attribute, and is defined with a ".". For example,

<head>

<style type="text/css">

.center

{

text-align:center;

}

</style>

</head>

<body>

<h1 class="center">Center-aligned heading</h1>

<p class="center">Center-aligned paragraph.</p>

</body>

You can also specify that only specific HTML elements should be affected by a class. For example,

<head>

<style type="text/css">

p.center

{

text-align:center;

}

</style>

</head>

<body>

<h1 class="center">This heading will not be affected</h1>

<p class="center">This paragraph will be center-aligned.</p>

</body>

**Multiple Styles Will Cascade into One**

Styles can be specified:

 inside an HTML element

 inside the head section of an HTML page

 in an external CSS file

**Tip:** Even multiple external style sheets can be referenced inside a single HTML document.

**Cascading order**

What style will be used when there is more than one style specified for an HTML element? Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

1. Browser default

2. External style sheet

3. Internal style sheet (in the head section)

4. Inline style (inside an HTML element)

So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or in a browser (a default value).

**Note:** If the link to the external style sheet is placed after the internal style sheet in HTML <head>, the external style sheet will override the internal style sheet!

**Pseudo Classes and Pseudo Elements**

The **pseudo-class** concept is introduced to permit selection based on information that lies outside of the document tree or that cannot be expressed using the other simple selectors. Pseudo-classes are allowed in all sequences of simple selectors contained in a selector.

***Syntax***

selector:pseudo-class{ property: value; }

***Example***

**Example**

<html>

<head>

<style>

p::first-line {

color: #ff0000;

font-variant: small-caps;

}

</style>

</head>

<body>

<p>You can use the ::first-line pseudo-element to add a special effect to the first line of a text.</p>

</body>

</html>

**Pseudo-elements** create abstractions about the document tree beyond those specified by the document language. For instance, document languages do not offer mechanisms to access the first letter or first line of an element's content. Pseudo-elements allow authors to refer to this otherwise inaccessible information.

**CSS Background**

Background properties are used to define the background effects of an HTML element. CSS properties used to define background effects are: **background-color**, **background-image**, **background-repeat**, **background-attachment**, and **background-position**.

**Background Image**

The background-image property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element.

The background image for a page can be set like this:

body {background-image:url('paper.gif');}

**Background Image - Repeat Horizontally or Vertically**

By default, the background-image property repeats an image both horizontally and vertically. Some images should be repeated only horizontally or vertically, or they will look strange, like this:

**Example**

body {

background-image:url('gradient2.png'); }

If the image is repeated only horizontally (repeat-x), the background will look better:

**Example**

body {

background-image:url('gradient2.png');

background-repeat:repeat-x; }

**Background Image - Set position and no-repeat**

When using a background image, use an image that does not disturb the text. Showing the image only once is specified by the background-repeat property:

**Example**

body {

background-image:url('img\_tree.png'); background-repeat:no-repeat;

}

In the example above, the background image is shown in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much.

The position of the image is specified by the background-position property:

**Example**

body {

background-image:url('img\_tree.png'); background-repeat:no-repeat; background-position:right top;

}

**CSS Display and Visibility**

The display property specifies if/how an element is displayed, and the visibility property specifies if an element should be visible or hidden.

**Hiding an Element - display:none or visibility:hidden**

Hiding an element can be done by setting the display property to "none" or the visibility property to "hidden". However, notice that these two methods produce different results:

visibility: hidden hides an element, but it will still take up the same space as before. The element will be hidden, but still affect the layout.

**Example**

h1.hidden {visibility:hidden;}

**display: none** hides an element, and it will not take up any space. The element will be hidden, and the page will be displayed as the element is not there:

**Example**

h1.hidden {display:none;}

**CSS Display - Block and Inline Elements**

A block element is an element that takes up the full width available, and has a line break before and after it.

Examples of block elements:

 <h1>

 <p>

 <div>

An inline element only takes up as much width as necessary, and does not force line breaks.

Examples of inline elements:

 <span>

 <a>

**Changing How an Element is Displayed**

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow web standards. The following example displays list items as inline elements:

**Example**

li {display:inline;}

The following example displays span elements as block elements:

**Example**

span {display:block;}

Changing the display type of an element changes only how the element is displayed, NOT what kind of element it is. For example: An inline element set to display:block is not allowed to have a block element nested inside of it.

**CSS Padding Property**

**Example**

Set the padding of a p element:

p {

padding:2cm 4cm 3cm 4cm; }

**Definition and Usage**

The padding shorthand property sets all the padding properties in one declaration. This property can have from one to four values.

Examples:

 **padding:10px 5px 15px 20px;**

o top padding is 10px

o right padding is 5px

o bottom padding is 15px

o left padding is 20px

 **padding:10px 5px 15px;**

o top padding is 10px

o right and left padding are 5px

o bottom padding is 15px

 **padding:10px 5px;**

o top and bottom padding are 10px

o right and left padding are 5px

 **padding:10px;**

o all four paddings are 10px

**Note:** Negative values are not allowed.

**CSS Float**

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it. Float is very often used for images, but it is also useful when working with layouts. Elements are floated horizontally; this means that an element can only be floated left or right, not up or down. A floated element will move as far to the left or right as it can. Usually this means all the way to the left or right of the containing element. The elements after the floating element will flow around it. The elements before the floating element will not be affected. If an image is floated to the right, a following text flows around it, to the left.

**Example**

img {

float:right; }

**Floating Elements Next to Each Other**

If you place several floating elements after each other, they will float next to each other if there is room. Here we have made an image gallery using the float property:

**Example**

.thumbnail { float:left; width:110px; height:90px; margin:5px;

}

**Turning off Float - Using Clear**

Elements after the floating element will flow around it. To avoid this, use the clear property.

The clear property specifies which sides of an element other floating elements are not allowed.

Add a text line into the image gallery, using the clear property:

**Example**

.text\_line {

clear:both; }

**Grouping and Nesting Selectors**

We can group and nest items to minimize the amount of code you write, this will minimize our style-sheet size and will allow pages to load faster. When grouping we separate each selector by a comma. You can give the same properties to a number of selectors without having to repeat them.

***Example***

<html> <head> <title>CSS Grouping</title>

<style type="text/css">

h1,h2,h3,h4,h5,h6,p

{

color:green;

font-family: Verdana

}

</style>

</head>

<body>

<h1>Working with Grouping and Nesting in CSS - H1 </h1> <h2>Working with Grouping and Nesting in CSS - H2</h2> <h3>Working with Grouping and Nesting in CSS - H3</h3> <h4>Working with Grouping and Nesting in CSS - H4</h4> <h5>Working with Grouping and Nesting in CSS - H5</h5> <h6>Working with Grouping and Nesting in CSS - H6</h6> <p>This is block of content - Paragraph</p>

</body>

</html>

It is possible to apply a style for a selector within a selector. For example, we can style headings within two divs differently.

#first {

background-color: red;

padding: 1cm

}

#second {

color:blue;

}

<div id="first">

<h1>Chocolate curry</h1>

</div>

<div id="second">

<h1>Chocolate curry</h1>

</div>

**Pseudo Classes and Pseudo Elements**

Pseudo-classes are allowed anywhere in selectors and are used to add special effects to some selectors. They are used to add styles to things that are not part of document tree.

***Syntax***

selector:pseudo-class

{ property:value; }

***Example:*** These pseudo-classes let you style unvisited links differently from visited ones.

<html>

<head>

<style>

/\* unvisited link \*/

a:link {

color: #FF0000;

}

/\* visited link \*/

a:visited {

color: #00FF00;

}

/\* mouse over link \*/

a:hover {

color: #FF00FF;

}

</style>

</head>

<body>

<a href="http://cdcsit.edu.np/">Visit CDCSIT</a></body>

</html>

Just like pseudo-classes, pseudo-elements are added to selectors but instead of describing a special state, they allow you to style certain parts of a document. For example, the ::first-line pseudo-element targets only the first line of an element specified by the selector.

***Syntax***

selector::pseudo-element { property: value; }

***Example***

<html>

<head>

<style>

p::first-line

{

color: #ff0000;

font-variant: small-caps;

}

p::first-letter

{ color: #ff0000; font-size: xx-large; }

</style>

</head>

<body>

<p>You can use the ::first-line pseudo-element to add a special effect to the first line of a text.</p>

</body>

</html>