Bachelor of Computer Application
Semester – II
US02MIBCA03: Web Application Development – II
UNIT-I: DHTML & Cascading Style Sheets

UNIT III: DHTML & CASCADING STYLE SHEETS	
Sr No.	**Topics**
1.	Introduction to DHTML, Applications of DHTML, Components of DHTML
2.	Introduction to Cascading Style Sheets (CSS), Ways of specifying style – inline, internal, external
3.	Basic Syntaxes, ID and CLASS selectors, SPAN, DIV
4.	Fonts, Color, Background, Text, Border, Lists
5.	Layers, Margin, Links, Position

Reference Books:

1. Ivan Bayross, "Web enabled Commercial Application Development using HTML, DHTML, Java script, perl CGI" BPB 2004
2. Xavier C: World Wide Web Design with HTML, Tata Mcgraw hill publication 2000

❖ What is DHTML?

- DHTML stands for **Dynamic Hypertext markup language**.
- It is an extension of HTML and it is not a language.
- DHTML is the combination of HTML, Cascading Style Sheets, DOM and JavaScript used to create dynamic Web content which can move, hide, or animate as a result of user events.
- Dynamic HTML" is typically used to describe the combination of HTML, style sheets scripts and Dom that allows documents to be animated.
- Dynamic HTML allows changing after web page loaded into the browser there doesn't have to be any communication with the web server for an update.
- Dynamic HTML is a new web technology that enables elements inside your web page to be well dynamic. For example, a piece of text can change from one size or color to another, or a graphic can move from one location to another, in response to some kind of user action, such as clicking a button.
- Dynamic HTML gives authors creative control so they can manipulate any page element and change styles, positioning, and content at any time.
- DHTML is a TERM describing the art of making dynamic and interactive web pages.

- DHTML is a new tool for designers to create Web pages with special effects and animation such as flying headlines and news tickers.

❖ USES / APPLICATIONS OF DHTML

- Animate text and images in their document independently moving each element from any starting point to any ending point. Following a predetermined path or one chosen by the user.
- Embed a ticker (timer) that automatically refreshes its content with the latest news stock quotes or other data
- Include rollover buttons or drop down menus
- A less common use is to create browser based action games.
- **Form validation**: use a form to capture user input and then process and respond to that data without having to send data back to the server.
- **Image swaps**: When the mouse is moved over an image, it changes to another image.
- **Pop up windows**: content can be shown in a smaller window
- **Dynamic content**: it is used for showing content on the page based on a user preference.
- **Backgrounds**: CSS allows background color, background image in document.
- **Setting text properties**: CSS allow you to set rules for font properties
- **Changing the visibility of objects**: the visibility of an element can be turned on or off.
- **Changing the tags and properties**: it allows you to change the qualities of a html tag depending on an event outside the browser. Such as mouse click, time, date, etc.
- **Real time positioning**: Object images and text moving around the web page. It can allow you to play interactive games with your readers or animate portions of your screen.
- **Browser detection**: JavaScript can be made to detect things like operating systems, screen resolutions, and even color depth.

Examples of DHTML

- Object and text animation
- Pop-up and pull-down menus
- Roll-over buttons and images

- Web page content retrieved from external data sources
- DHTML form validator
- Floating gallery with drag and drop
- Calendar
- Slider
- Tab view
- Menu bar
- Dynamic tooltip
- Dynamic content
- Folder tree with drag and drop
- Drag and drop demo
- Progress bar
- Image enlarger, Image enlarger(with draggable images)
- Resize widget
- Context menu
- Auto complete
- Chained select

It is an extension of HTML that enables, among other things, the inclusion of small animations and dynamic menus in Web pages.

Uses of DHTML Applications

Building a DHTML application in Visual Basic provides several advantages over other methods of Internet development. DHTML applications give you:

- **Dynamic HTML**. When you create a DHTML application, you have full access to the richness of, integrated with the power of Visual Basic code and controls. **Lessen server load**. DHTML applications conserve server resources because each request or user action does not have to be routed through the Web server.

- **Fewer refreshes, faster responses**. When an end user's actions initiate changes to a typical Web page, the browser must refresh the page from the server. In a DHTML application, the browser can process user data, make changes to the page's layout and appearance, and process code all without refreshing the page.

- **Dynamic interaction**. Visual Basic code on a Web page can directly manipulate any element on the page and create and manage new elements on the fly, allowing for truly dynamic user interfaces.

- **Improved state management**. Typically, HTML pages are *stateless* — that is, no information about an HTTP request is maintained after the response is received from the server. Visual Basic DHTML applications allow you to store

state between requests, without using the server. Therefore, multiform or multipage applications are possible without requiring server interaction, complex URL-based state, or cookies.

- **Offline capability**. For a DHTML application, users can browse to and use a DHTML application on their corporate intranet. Later, when disconnected, the same users can still make use of their Web-based application through the browser's cached storage.

- **Code security**. When you embed scripts within an HTML page, anyone can access your page, read the script, and make changes to it. Using Visual Basic to develop your DHTML application, your code is compiled, is not part of the HTML page itself, and cannot be tampered with as easily.

STRUCTURE OF A WEB PAGE

Typically a web page using DHTML is set up the following way:

```
<html>
<head>
    <title>dhtml example</title>
    <link rel=stylesheet type="text/css" href="ext1.css">
    <script language="JavaScript">
        function addition()
        {
        var a=parseFloat(window.document.calc.no1.value);
        var b=parseFloat(window.document.calc.no2.value);
        var c;
        c=a+b;
        window.document.calc.ans.value=c;
        }
    </script>
    </HEAD>
    <body style="color:red">
    <form name=calc>
    No 1:<input type=text name="no1" size=4><br>
    No 2:<input type=text name="no2" size=4><br>
    <input type=button value=Add onClick=addition() name=add><br>
    Result:<input type=text name="ans" size=10>
    </form>
</html>
```

COMPONENTS OF DHTML

1. **HTML**

2. **SCRIPTING LANGUAGE**
3. **CSS**
4. **DOM**

HTML:

- It is the core of DHTML. You have to understand the basics before you even consider making your plain HTML dynamic.

- HTML defines the structure of a Web page, using such basic elements as headings, forms, tables, paragraphs and links.

2. Cascading style sheets (CSS):

- CSS is easy to learn and understand but it provides precise control over the presentation of an HTML document.

- Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized what background images or colors are used, as well as a variety of other effects.

3. Scripting Language:

- It is the most commonly used scripting language for creating DHTML effects.

- A scripting language is a lightweight programming language and it is easy to learn.

- JavaScript programming code is written into HTML pages.
- JavaScript code can be executed by all web browsers.
- JavaScript is most commonly used as a client side scripting language. this means that JavaScript code is written into an HTML page.

- A user requests an HTML page When JavaScript language that runs on the Client's browser to bring special effects.

- Basically some script function is called to execute the required effect when events like 'Mouse Over', 'Mouse Out', 'Click', etc. occurs.

4. Document Object Model (DOM):

- “Document Object Model (DOM) is allows programs and scripts to dynamically access and update the content, structure, and style of a document."
- It takes the individual components of the HTML document and organizes them into a hierarchal structure of named objects.

- Dom allows you to access any part of your webpage to change with dhtml. Dom defines a standard way for accessing, changing web document by using object, properties, method, user event (on click, on load).

- DOM is a fully object-oriented representation of the web page, and it can be modified with a scripting language such as JavaScript
- The DOM is not a programming language but it's written in JavaScript.
- JavaScript uses the DOM to access the document and its elements of the web pages, but without it, the JavaScript language wouldn't have access the document and its elements of the web pages,
- The Document Object Model, or DOM, is the interface that allows you to programmatically access and manipulate the contents of a web page (or document).
- It provides a structured, object-oriented representation of the individual elements and content in a page with methods for retrieving and setting the properties of those objects.
- It also provides methods for adding and removing such objects, allowing you to create dynamic content.
- The DOM also provides an interface for dealing with events, allowing you to capture and respond to user or browser actions.

❖ **Introduction to css:**

- CSS are powerful mechanism for adding styles to web document for e.g. font, color, spacing, border etc.
- it provides powerful control over the presentation of an HTML document. Style sheets are used for adding styles to web document for e.g. font, color, spacing, border etc.
- In style sheets, text and image formatting properties can be predefined in a single list.
- The advantage of style sheet includes the ability to make global changes to all documents from a single location.

ADVANTAGES OF CSS

- CSS, in combination with HTML and XML, truly separates content from its presentation.
- Will allow designers to create tighter, more dynamic content for the Web
- Pages using CSS will transfer faster to users and will be easy to maintain
- Will lead to improved accessibility, maintainability and performance on the Web
- The idea behind style sheets is to make a distinction between the content of a web page and the style/formatting of a page.
- Consistent design (e.g. changing a style definition in a “style sheet” can change every page on your web site!)
- More flexibility
- Less work to maintain large web sites
- It allows presentation to be specified separately.

- This makes it possible to use the same presentational specifications for several pages.
- This also makes it easier to change the presentation of those pages.
- And it makes it possible to have different presentational info applied to the same page under different circumstances.
- As an added benefit, it results in smaller, simpler XHTML documents for those pages.

There are several reasons to recommend CSS:

- You have more choices: Using CSS, you can add color, positioning, and special effects not available with standard HTML.
- You save time: CSS save time in many ways. As a Web developer, you will save time creating pages in FrontPage because once you have established the style elements, formatting is one click away, rather than several. In addition, you will save in site maintenance time because one change to one style sheet can update all of your pages at once. Finally, your client will save time because CSS saves on code, so pages download more quickly.
- Your site is more consistent. CSS allows you to standardize the look and feel of your site in all operating systems and all CSS-capable browsers.
- A development team can use CSS to maximize its members' talents. CSS facilitates the logical division of labor into *content* and *appearance*. Thus, a team can delegate these tasks according to the skills of the members more easily.
- CSS increases access to your pages. CSS is easier for browsers to read, allowing easier access by people with disabilities and those using languages that do not rely on Latin-style alphabets.

CSS Syntax

Every language including computer languages have certain rules for how words should be arranged to form sentences or statements. Computer languages in particular must be very specific because a computer can only do what you tell it to do. Unlike humans, computers cannot discern meaning if your sentence structure is incorrect!

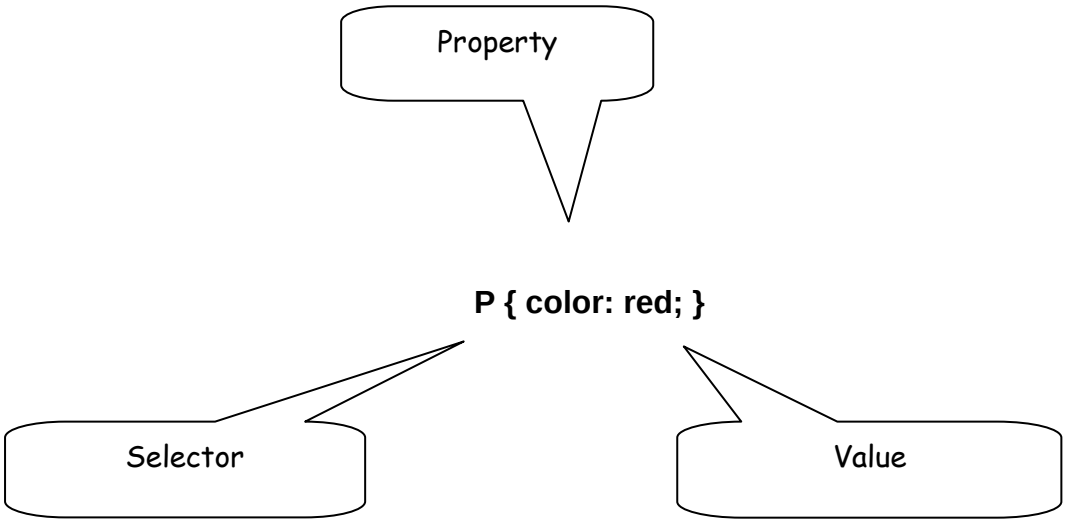
CSS syntax differs from what we have seen with html. It is a different language! Where HTML was designed to define the structure of a web page, CSS was designed to add style to a web page.

There are three parts to a CSS statement:

1. selector
2. property

3. value

The whole statement is called a *rule*. **Syntax: Selector { property:value; }**



CSS selects the elements and tells the browser how to display it in the web page.

In the example above, CSS is telling the browser to display all paragraph elements with red text.

Selector

Selector is normally the HTML element you want to style. In our example, P is the paragraph element. Notice, in CSS syntax you type the element abbreviation only. Do not include the left and right angle brackets…that's HTML. You may have more than one selector. To group elements, simply separate the selectors with a comma, like this:

P, h1 { color: red; }

Property and Value

The property allows you to manipulate style. In our example above, color is the property that will be assigned a value. Correct syntax is important. Surround the property: value statement with braces, separate the property and value with a colon and end the statement with a semi-colon.

How to add style in HTML document

There are *three* ways to add CSS style to your web page.

1. Inline Style
2. Internal (Embedded) Style
3. External Style

Example – Inline Style

- An inline style is applied to a single tag or section of the current HTML file.
- Inline style declarations are placed inside the tag.

```
<HTML>
<HEAD><TITLE>INLINE STYLE SHEET</TITLE></HEAD>
<BODY>
<h2> Day 1 </h2>

<img src="pictures/alligator.jpg">

<p style=”color: red;”> After a long but very cool boat ride we arrived at the island! First we unloaded our stuff and met Jim Bitler. He told us a lot about the island and how to be safe. After lunch at the clubhouse we got our island IT kits. It had a GPS, camera and laptop that we got to use while on the island. I took this picture of an alligator during our tour. </p>
</BODY>
</HTML>
```

Using inline style to add CSS is very specific. It only affects the element in the line. Inline styling should only be used when you want to change one particular element.

Suppose you want all paragraphs in your document to have style. Using the inline style means you have to type the code into every paragraph element in your document.

Example - Internal (Embedded) style

- An internal style is applied to the entire current HTML file but is not applied to other files on the web site.
- Internal style declarations are placed in the header of the HTML file.

```
<HTML>
<head>
    <title> Brooke Hooker </title>
    <style type=”text/css”>
    P { color: Green;
        Margin: 20px;
        }
```

```
        </style>
</head>
<BODY>
<h2> Day 1 </h2>

<img src="pictures/alligator.jpg">

<p> After a long but very cool boat ride we arrived at the island! First we unloaded our stuff and met Jim Bitler. He told us a lot about the island and how to be safe. After lunch at the clubhouse we got our island IT kits. It had a GPS, camera and laptop that we got to use while on the island. I took this picture of an alligator during our tour. </p>
</BODY>
</HTML>
```

Using the internal style method, type your CSS inside the <style></style> tags located in the head section of your html document. The style tags let the browser know that everything in between will be CSS syntax.

Example - External Style

- An external style is applied to the entire HTML file and may be applied to any or all pages on the web site.
- External style declarations are written in a separate text file which is then linked to the HTML file.
- External style sheets make it easy to give a consistent look to any number of web pages.

To use an external style sheet there are 3 steps.

1. Create .css file
2. Write CSS code
3. Link the .css to .html

Create a CSS file

On the notepad menu bar click File > Save As, then type “yourfilename.css” , in the save as type box select All Files, Click Save.

Write CSS Code

With your .css file open type the highlighted text.

```
p { color: blue;
        margin: 30px;
        font-family: monotype corsiva;
        }
```

Click File > Save>Save the file with ext1.css.

Link the ext1.css to your html file
Type the following code in the html file.

```
<html>
        <head>
        <title> Your Name </title>
        <link type=”text/css” rel=”stylesheet” href=”ext1.css” />
        </head>
<BODY>
<h2> Day 1 </h2>

<img src="pictures/alligator.jpg">

<p> After a long but very cool boat ride we arrived at the island! First we unloaded our stuff and met Jim Bitler. He told us a lot about the island and how to be safe. After lunch at the clubhouse we got our island IT kits. It had a GPS, camera and laptop that we got to use while on the island. I took this picture of an alligator during our tour. </p>
</BODY>
</HTML>
```

Note: In the value for href, type whatever you named your CSS file.

Multiple Style Sheets

Multiple style sheets can be used / cascaded into one html file. 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 suggests the priority level:

1. Inline style (inside an HTML element)
2. Internal (Embedded) style sheet (in the head section)
3. External style sheet (in the head section)
4. Browser default

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 an internal style sheet, 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!

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 "*/".

e.g: /* This is a CSS comment */

SELECTORS

ID

- Use an id to distinguish something, like a paragraph, from the other paragraph in a document.
  - For example, to identify a paragraph as “head”, use the code:
  …

- To create an ID for a specific tag, use the property:
  - <tag ID=id_name>
- To apply a style to a specific ID, use:
  - #id_name {style attributes and values}

CLASS

Using a class selector allows you to define different styles for the same element. For instance, if your HTML document contains multiple paragraph elements you can style them individually using classes.

To use a class there are two steps

1. Define the class in CSS
   CSS syntax:
   selector.classname {property:value; }
   OR
   .classname [property:value;}

2. Call the class in HTML.
   HTML syntax:
   <element class="classname">
   OR
   <tag class="classname">

Example:

Following is the .css file:

```
h2 { font-size: 15px; }
p.cursive {font-family: cursive;}
```

Following is the index.html file using class in <p> tag:

```
<h2>CLASS EFFECT</h2>
<p class="cursive"> The font change affects only the paragraph elements. The font change affects only the paragraph elements. The font change affects only the paragraph elements.
The font change affects only the paragraph elements. </p>
```

- The **difference between the Class property and the ID property** is that the value of the ID property must be unique:
  - You can't have more than one tag with the same ID value.
  - You can apply the same Class value to multiple document tags

❖ ** tag:**

- A web page can be divided into segments or divisions using . these segments can be position anywhere on the page. each segment start with and end with .
- Div tag has a **position attribute** that can take two values.

(1) absolute position:

- This segment with respect to the top/left edge of the browser window.

Relative position:

- This segment in relation to other elements on the page.

- **Example:**

.css file

#box1{**position:relative**;left:30;top:15;width:5;height:20;z-index:1;visibility:visible}

Example:

```
<div style="color:olive">
    <h2>Table of Contents.</h2>
   <ol>
     <li>Chapter 1</li>
     <li>Chapter 2</li>
   </ol>
</div>
```

❖ **Layer :**

- To segment or divisions of a web page there is also use of another tag start with<layer>and end with </layer>.
- These tags are designed to behave like a piece of transparent paper laid on top of a web page
- <layer> is provided with an **id** and with attributes and value that specify its appearance and position .

SPAN

SPAN tag is usually used to format a small amount of text, such as a single word or sentence. Some people say that span tags are inline elements (because they can be used in a line of text)

Example:

```
<span style="font-style:italic"> Hey! </span> This is the DHTML World!
```

❖ **MEASUREMENT UNITS / LENGTH VALUES unit:**

- Length can be specified, such as **IN** (inches), **CM** (centimeters), **PT** (points) and PX (pixels)

1. **IN (inches):**
   - It specifies the value in inches and **absolute** unit.
   - It is used in printing unit
2. **CM (centimeters):**
   - It specifies the unit in centimeters and **absolute** unit.
   - It is used in printing unit

3. **PT (points):**
   - It specifies the value in **PT** (points) and **absolute** unit.
   - 72pt equals 1**in** (inches)
   - It is used in printing unit

4. **PX** (**pixels**):
   - It specifies the value in **PX** (pixels) and **relative unit.**
   - It is used one dot on the screen.(one dot space)

- **Example:**

  H1 {font-size: 1in}

CSS BACKGROUND PROPERTIES

CSS background properties are used to define the background effects of an element. CSS properties used for background effects:

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

Property	Description	Values
background	Sets all the background properties in one declaration	background-color background-image background-repeat background-attachment background-position
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page	scroll fixed
background-color	Sets the background color of an element	color-rgb color-hex color-name
background-image	Sets the background image for an element	url(URL) none
background-position	Sets the starting position of a background image	left top left center

		left bottom right top right center right bottom center top center center center bottom x% y% xpos ypos
background-repeat	Sets if/how a background image will be repeated	repeat repeat-x repeat-y no-repeat

Background Color

The background-color property specifies the background color of an element. The background color of a page is defined in the body selector:

Example

```
body { background-color:#b0c4de;}
```

The background color can be specified by:

- name - a color name, like "red"
- RGB - an RGB value, like "rgb(255,0,0)"
- Hex - a hex value, like "#ff0000"

In the example below, the h1, p, and div elements have different background colors:

Example

```
h1 {background-color:#6495ed;}
p {background-color:#e0ffff;}
div {background-color:#b0c4de;}
```

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:

Example

```
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.

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; }
```

TEXT PROPERTIES

The text can be styled with some of the text formatting properties like text-align, text-transform, color properties, indentation, alignment, and the space between characters.

Property	Description	Values
letter-spacing	Increase or decrease the space between characters	normal length
text-align	Aligns the text in an element	left right center justify
text-decoration	Adds decoration to text	none underline overline line-through blink
text-indent	Indents the first line of text in an element	length %
text-transform	Controls the letters in an element	none capitalize uppercase lowercase

Text Color

The color property is used to set the color of the text. The color can be specified by:

- name - a color name, like "red"
- RGB - an RGB value, like "rgb(255,0,0)"
- Hex - a hex value, like "#ff0000"

The default color for a page is defined in the body selector.

Example

```
body { color:blue;}
h1 { color:#00ff00;}
h2 { color:rgb(255,0,0);}
```

Text Alignment

The text-align property is used to set the horizontal alignment of a text. Text can be centered, or aligned to the left or right, or justified. When text-align is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

Example

```
h1 { text-align:center;}
p.date { text-align:right;}
p.main { text-align:justify;}
```

Text Decoration

The text-decoration property is used to set or remove decorations from text. The text-decoration property is mostly used to remove underlines from links for design purposes:

Example

```
a { text-decoration:none;}
```

It can also be used to decorate text:

Example

```
h1 { text-decoration:overline;}
h2 { text-decoration:line-through;}
h3 { text-decoration:underline;}
h4 { text-decoration:blink;}
```

Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

Example

```
p.uppercase { text-transform:uppercase;}
p.lowercase { text-transform:lowercase;}
p.capitalize { text-transform:capitalize;}
```

Text Indentation

The text-indentation property is used to specify the indentation of the first line of a text.

Example

```
p { text-indent:50px;}
```

FONT PROPERTIES

CSS font properties define the font family, boldness, size, and the style of a text.

Font Properties

Property	Description	Values
font	Sets all the font properties in one declaration	font-style font-variant font-weight font-size font-family
font-family	Specifies the font family for text	family-name
font-size	Specifies the font size of text	Smaller small medium large larger

		length %
font-style	Specifies the font style for text	normal italic oblique
font-variant	Specifies whether or not a text should be displayed in a small-caps font	normal small-caps
font-weight	Specifies the weight of a font	normal bold bolder lighter 100px

Font Family

The font family of a text is set with the font-family property. If the browser does not support the first font, it tries the next font. Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Note: If the name of a font family is more than one word, it must be in quotation marks, like font-family: "Times New Roman". More than one font family is specified in a comma-separated list:

Example

```
P { font-family:"Times New Roman", Times, serif;}
```

Font Style

The font-style property is mostly used to specify italic text. This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Example

```
p.normal { font-style:normal;}
p.italic { font-style:italic;}
p.oblique { font-style:oblique;}
```

Font Size

The font-size property sets the size of the text. Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs. Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs. The font-size value can be an absolute or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

The default size for normal text is 16px (16px=1em).

Set Font Size With Pixels

Setting the text size with pixels, gives you full control over the text size:

Example

```
h1 { font-size:40px;}
h2 { font-size:30px;}
p { font-size:14px;}
```

Font-Variant

Font-Variant property is used to display the text in a small-caps font.

Example

```
h1 { font-variant:normal;}
h2 { font-variant:small-caps;}
```

Font-Weight

Font-Weight property is used to display the text in a bold, light, bolder, lighter font.

Example

```
h1 { font-weight:bold;}
h2 { font-weight:light;}
h3 { font-weight:lighter;}
h4 { font-weight:bolder;}
h5 { font-weight:200px;}
```

LIST PROPERTIES

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker

In HTML, there are two types of lists:

- unordered lists - the list items are marked with bullets
- ordered lists - the list items are marked with numbers or letters

With CSS, lists can be styled further, and images can be used as the list item marker.

List Properties

Property	Description	Values
list-style	Sets all the properties for a list in one	list-style-type

	declaration	list-style-position list-style-image
list-style-image	Specifies an image as the list-item marker	URL none
list-style-position	Specifies if the list-item markers should appear inside or outside the content flow	inside outside
list-style-type	Specifies the type of list-item marker	none disc circle square decimal decimal-leading-zero lower-alpha upper-alpha lower-roman upper-roman

Different List Item Markers

The type of list item marker is specified with the list-style-type property:

Example

```
ul.a {list-style-type: circle;}
ul.b {list-style-type: square;}
ol.c {list-style-type: upper-roman;}
ol.d {list-style-type: lower-alpha;}
```

Some of the property values are for unordered lists, and some for ordered lists.

Values for Unordered Lists

Value	Description
none	No marker
disc	Default. The marker is a filled circle
circle	The marker is a circle
square	The marker is a square

Values for Ordered Lists

Value	Description
decimal	The marker is a number
decimal-leading-zero	The marker is a number padded by initial zeros (01, 02, 03, etc.)
lower-alpha	The marker is lower-alpha (a, b, c, d, e, etc.)
lower-roman	The marker is lower-roman (i, ii, iii, iv, v, etc.)
upper-alpha	The marker is upper-alpha (A, B, C, D, E, etc.)
upper-roman	The marker is upper-roman (I, II, III, IV, V, etc.)

Note: No versions of Internet Explorer (including IE8) support the property values "decimal-leading-zero" UNLESS a DOCTYPE is specified!

List - Shorthand property

It is also possible to specify all the list properties in one, single property. This is called a shorthand property. The shorthand property used for lists is the list-style property:

Example

```
Ul { list-style: square url("sqpurple.gif"); }
```

When using the shorthand property, the order of the values are:

- list-style-type
- list-style-position
- list-style-image

It does not matter if one of the values above are missing, as long as the rest are in the specified order.

An Image as The List Item Marker

To specify an image as the list item marker, use the list-style-image property:

Example

```
Ul { list-style-image: url('sqpurple.gif'); }
```

The example above does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari.

BORDER PROPERTIES

The CSS border properties allow you to specify the style and color of an element's border.

Border Properties

Property	Description	Values
border	Sets all the border properties in one declaration	border-width border-style border-color
border-bottom	Sets all the bottom border properties in one declaration	border-bottom-width border-bottom-style border-bottom-color
border-bottom-color	Sets the color of the bottom border	border-color
border-bottom-style	Sets the style of the bottom border	border-style
border-bottom-width	Sets the width of the bottom border	border-width
border-color	Sets the color of the four borders	color_name hex_number rgb_number transparent
border-left	Sets all the left border properties in one declaration	border-left-width border-left-style border-left-color
border-left-color	Sets the color of the left border	border-color
border-left-style	Sets the style of the left border	border-style
border-left-width	Sets the width of the left border	border-width
border-right	Sets all the right border properties in one declaration	border-right-width border-right-style border-right-color

border-right-color	Sets the color of the right border	border-color
border-right-style	Sets the style of the right border	border-style
border-right-width	Sets the width of the right border	border-width
border-style	Sets the style of the four borders	none hidden dotted dashed solid double groove ridge inset outset
border-top	Sets all the top border properties in one declaration	border-top-width border-top-style border-top-color
border-top-color	Sets the color of the top border	border-color
border-top-style	Sets the style of the top border	border-style
border-top-width	Sets the width of the top border	border-width
border-width	Sets the width of the four borders	thin medium thick length

Border Style

The border-style property specifies what kind of border to display. None of the border properties will have ANY effect unless the border-style property is set!

border-style values:

none: Defines no border

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders are the same as the border-width value

groove: Defines a 3D grooved border. The effect depends on the border-color value

ridge: Defines a 3D ridged border. The effect depends on the border-color value

inset: Defines a 3D inset border. The effect depends on the border-color value

outset: Defines a 3D outset border. The effect depends on the border-color value

Border Width

The border-width property is used to set the width of the border. The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.

Note: The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

```
p.one { border-style:solid; border-width:5px; }
p.two { border-style:solid; border-width:medium; }
```

Border Color

The border-color property is used to set the color of the border. The color can be set by:

- name - specify a color name, like "red"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- Hex - specify a hex value, like "#ff0000"

You can also set the border color to "transparent".

Note: The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

```
p.one { border-style:solid; border-color:red; }
p.two { border-style:solid; border-color:#98bf21; }
```

Border - Individual sides

In CSS it is possible to specify different borders for different sides:

Example

```
P { border-top-style:dotted; border-right-style:solid; border-bottom-style:dotted;
border-left-style:solid; }
```

The above example can also be set with a single property:

Example

```
border-style:dotted solid;
```

The border-style property can have from one to four values. For example:

- border-style:dotted solid double dashed;
  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed
- border-style:dotted solid double;
  - top border is dotted
  - right and left borders are solid
  - bottom border is double
- border-style:dotted solid;
  - top and bottom borders are dotted
  - right and left borders are solid
- border-style:dotted;
  - all four borders are dotted

Border - Shorthand property

As you can see from the examples above, there are many properties to consider when dealing with borders. To shorten the code, it is also possible to specify all the border properties in one property. This is called a shorthand property. The shorthand property for the border properties is "border":

Example

```
border:5px solid red;
```

When using the border property, the order of the values is:

- border-width
- border-style
- border-color

MARGIN PROPERTIES

The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent. The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

Margin Properties

Property	Description	Values
margin	A shorthand property for setting the margin properties in one declaration	margin-top margin-right margin-bottom margin-left
margin-bottom	Sets the bottom margin of an element	auto length %
margin-left	Sets the left margin of an element	auto length %
margin-right	Sets the right margin of an element	auto length %
margin-top	Sets the top margin of an element	auto

		length %

Possible Values

Value	Description
auto	The browser sets the margin. The result of this is dependant of the browser
length	Defines a fixed margin (in pixels, pt, em, etc.)
%	Defines a margin in % of the containing element

It is possible to use negative values, to overlap content.

Margin - Individual sides

In CSS, it is possible to specify different margins for different sides:

Example

```
margin-top:100px;
margin-bottom:100px;
margin-right:50px;
margin-left:50px;
```

Margin - Shorthand property

To shorten the code, it is possible to specify all the margin properties in one property. This is called a shorthand property. The shorthand property for all the margin properties is "margin":

Example

```
margin:100px 50px;
```

The margin property can have from one to four values. For example:

- margin:25px 50px 75px 100px;
  - top margin is 25px
  - right margin is 50px

    - bottom margin is 75px
    - left margin is 100px
- margin:25px 50px 75px;
    - top margin is 25px
    - right and left margins are 50px
    - bottom margin is 75px
- margin:25px 50px;
    - top and bottom margins are 25px
    - right and left margins are 50px
- margin:25px;

POSITIONING / LAYERS

To create a layer you need to do is assign the **position** attribute to your style. The position can be either **absolute** or **relative**. The position itself is defined with the **top** and **left** properties. Finally, which layer is on top is defined with the **z-index** attribute.

The CSS positioning properties allow you to position an element. It can also place an element behind another, and specify what should happen when an element's content is too big.

Elements can be positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the positioning method.

There are four different positioning methods.

Static Positioning

HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page. Static positioned elements are not affected by the top, bottom, left, and right properties.

Fixed Positioning

An element with fixed position is positioned relative to the browser window. It will not move even if the window is scrolled:

Example

```
p.pos_fixed { position:fixed; top:30px; right:5px; }
```

Note: Internet Explorer supports the fixed value only if a !DOCTYPE is specified.

Fixed positioned elements are removed from the normal flow. The document and other elements behave like the fixed positioned element does not exist. Fixed positioned elements can overlap other elements.

Relative Positioning

A relative positioned element is positioned relative to its normal position.

Example

```
h2.pos_left { position:relative; left:-20px; }
h2.pos_right { position:relative; left:20px; }
```

The content of a relatively positioned elements can be moved and overlap other elements, but the reserved space for the element is still preserved in the normal flow.

Example

```
h2.pos_top { position:relative; top:-50px; }
```

Relatively positioned element is often used as container blocks for absolutely positioned elements.

Absolute Positioning

An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>:

Example

```
h2 { position:absolute; left:100px; top:150px; }
```

Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist. Absolutely positioned elements can overlap other elements.

Overlapping Elements (z-index property)

When elements are positioned outside the normal flow, they can overlap other elements. The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others). An element can have a positive or negative stack order:

Example

```
Img { position:absolute; left:0px; top:0px; z-index:-1 }
```

An element with greater stack order is always in front of an element with a lower stack order.

Note: If two positioned elements overlap, without a z-index specified, the element positioned last in the HTML code will be shown on top.

Positioning Properties

Property	Description	Values
bottom	Sets the bottom margin edge for a positioned box	auto *length* *%*
left	Sets the left margin edge for a positioned box	auto *length* *%*
position	Specifies the type of positioning for an element	absolute fixed relative static
right	Sets the right margin edge for a positioned box	auto *length* *%*
top	Sets the top margin edge for a positioned box	auto *length* *%*
z-index	Sets the stack order of an element	*number* auto

USE OF LAYERS

Layers offer certain possibilities for precise positioning of static elements on your pages. In reality layers are often used in more dynamic ways:

- Flying elements/banners on the page
- Games where you move an object around
- Menus that pop out when triggered

- Menus that become visible when triggered

LINKS

Links can be styled with any CSS property (e.g. color, font-family, background, etc.). links can be styled differently depending on what state they are in. Usually, all these properties are kept in the header part of the HTML document.

The **four links states** are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user moves mouse over it
- a:active - a link the moment it is clicked

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

Example

```
/* unvisited link */
a:link { color: red; }

/* visited link */
a:visited { color: green; }

/* mouse over link */
a:hover { color: hotpink; }

/* selected link */
a:active { color: blue; }
```

Bachelor of Computer Application
Semester – II
US02MIBCA03: Web Application Development – II
UNIT-II: Introduction to Scripting

UNIT I & II:	
Sr No.	**Topics**
1.	Introduction to Scripting
2.	Client Side Scripting vs. Server Side Scripting
3.	How the Web works, Introduction to JavaScript, Applications and Advantages of JavaScript, Using JavaScript on a web

Reference Book:

1. **Ivan Bayross**, "Web Enabled Commercial Applications Development using HTML, DHTML, Javascript, Perl CGI", BPB, 2004.
2. **Douglas E Comer**: The Internet, PHI, Second Edition, May 2000.

INTRODUCTION TO SCRIPTING

Scripting languages are programming languages that are typically written using high-level programming constructs, which makes them easy to learn. While there is no fixed definition of what constitutes a scripting language, some of the common distinguishing traits of these languages include:

- **Interpreted:** Scripting languages are typically converted into machine level code during runtime by an interpreter, rather than being compiled into an executable before running. While this leads to a performance hit as each line has to be interpreted on the fly, it makes for easier portability between systems.
- **Typeless:** Variables can be used to hold any type of data without having to explicitly declare their type. While this can make it easy to run into typecasting errors, it makes the language easier to learn and can improve readability of the script.
- **Native Complex Types**: Most scripting languages also natively provide certain complex data types like strings, arrays, lists and hashes.
- **Garbage Collection:** Most scripting languages automate garbage collection (freeing of memory used by data). This can reduce the likelihood of memory leaks occurring.

SERVER-SIDE SCRIPTING

Normally when a browser requests an HTML file, the server returns the file, but if the file contains a server-side script, the script inside the HTML file is executed by the server before the file is returned to the browser as plain HTML.

What can Server Scripts Do?

- Dynamically edit, change or add any content to a Web page
- Respond to user queries or data submitted from HTML forms
- Access any data or databases and return the results to a browser
- Customize a Web page to make it more useful for individual users
- Provide security since your server code cannot be viewed from a browser

CLIENT-SIDE SCRIPTING

It generally refers to the class of computer programs on the web that are executed client-side, by the user's web browser, instead of server-side (on the web server). This type of computer programming is an important part of the Dynamic HTML (DHTML) concept, enabling web pages to be scripted; that is, to have different and changing content depending on user input, environmental conditions (such as the time of day), or other variables.

Web authors write client-side scripts in languages such as JavaScript (Client-side JavaScript) and VBScript.

SERVER SIDE VS CLIENT SIDE

There are two distinct operations involved in displaying any web page to the visitor, the first being server side operations and the second client side operations.

Server Side Operations

Server side operations are concerned with the sending of the web page data from the server to the web page visitors' browser. In the case of Static Web Pages, the data is simply served immediately upon request for the data from the visitors' browser. If the requested page is a Dynamic Web Page, then any preprocessing of the page is carried out and the output is then served to the visitor. PHP and ASP are server side scripting languages that are used to pre process pages and output HTML before the page is sent to the visitor. HTML is the language that the browser understands that tells it how to display the page.

Client Side Operations

Client side operations are performed on the visitors' computer by the users Internet browser to display the web page as the data is received from the server. HTML is interpreted as it is read by the browser resulting in the display of the web page within the browser. Once the page has loaded HTML cannot be

reprocessed without refreshing the page. The visitors experience on the web page can however be enhanced by means of a client side scripting language, typically Javascript used in conjunction with dynamic html and cascading style sheets, which enable interactive menu systems, hi-lighting effects, image effects, data manipulation and many other actions to be performed on the page without reloading or refreshing the page.

	Server-side scripting	Client-side scripting
Basic	Works in the backend which could not be visible at the client end	Works at the front end and script are visible among the user
Prosecccing	Requires server interaction	Does not need interaction with the server
Languages involved	PHP, ASP.Net, Ruby on Rails, ColdFusion, Python, etc	Javascript, VBScript
Affect	Could effectively customize the web pages and provide dynamic website	Can reduce the load to the server
Security	Relatively secure	Insecure

Difference between Server-side scripting and Client-side scripting

JavaScript

- A scripting language is a lightweight programming language
- JavaScript is a scripting language
- A JavaScript consists of lines of executable computer code
- A JavaScript is usually embedded directly into HTML pages
- JavaScript was designed to add interactivity to HTML pages

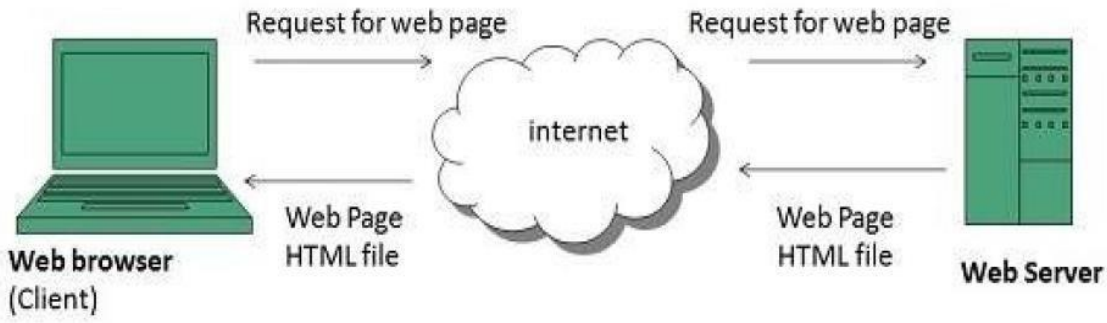
HOW THE WEB WORKS

Web works on client- server approach. Following steps explains how the web works:

- User enters the URL (say, http://www.tutorialspoint.com) of the web page in the address bar of web browser.
- Then browser requests the Domain Name Server for the IP address corresponding to www.tutorialspoint.com.
- After receiving IP address, browser sends the request for web page to the web server using HTTP protocol which specifies the way the browser and web server communicates.
- Then web server receives request using HTTP protocol and checks

its search for the requested web page. If found it returns it back to the web browser and close the HTTP connection.

- Now the web browser receives the web page, It interprets it and display the contents of web page in web browser's window.



WHAT IS JAVASCRIPT?

- Created by Brendan Eich of Netscape Communications in 1995
- Originally called LiveScript.
- JavaScript runs through a web browser
  - ✓ to give dynamic operation and control to web pages
  - ✓ executed on the user's or client's computer, i.e., the interpreter is part of the browser, not the server software
- JavaScript is an object-oriented language
  - ✓ program elements are organized into "objects"
- JavaScript is case sensitive
- JavaScript is NOT Java

  Java and JavaScript share many characteristics, but Java is a compiled language and it has a number of capabilities, instructions, and libraries not available to JavaScript.

INTRODUCTION TO JAVASCRIPT

- ✓ JavaScript is a high-level language
- ✓ JavaScript is an interpreted language
  - ◦ A program called an interpreter runs on the client's computer.
  - ◦ One instruction at a time, the interpreter figures out what to do by reading your *text* program file.
  - ◦ Interpreted programming language can run on any platform or O/S as long as there is an interpreter that runs on that particular setup.
  - ◦ Interpreted languages are usually slower.
  - ◦ Speed may not be a big factor because programs written in JavaScript

are not usually that complex

JAVASCRIPT USES / APPLICATION

- Browser Detection
  - Use JavaScript to detect the browser used by a visitor at your page. Depending on the browser, another page specifically designed for that browser can then be loaded.
- Dynamic HTML
  - Any code that allows you to change the content or appearance of a document in a Web browser.
- Cookies
  - Use JavaScript to store information on the visitor's computer, then retrieving this information automatically next time the user visits your page. This technique is called "cookies".
- Control Browsers
  - Opening pages in customized windows, where you specify if the browser's buttons, menu line, status line or whatever should be present.
- Data Processing
  - Validating inputs to fields before submitting a form.
    An example would be validating the entered email address to see if it has an @ in it, since if not, it's not a valid address.

What can a JavaScript Do?

- **JavaScript gives HTML designers a programming tool -** JavaScript is a scripting language with a very simple syntax. We can put small "snippets" of code into the HTML pages.
- **JavaScript can put dynamic text into an HTML page -** A JavaScript statement can write a variable text into an HTML page
- **JavaScript can react to events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element.
- **JavaScript can read and write HTML elements -** A JavaScript can read and change the content of an HTML element.
- **JavaScript can be used to validate data -** A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing.
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser.
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer.

An HTML file can contain text, HTML tags and scripts. Scripts in an HTML file can be executed on the Web server.

ADVANTAGES OF JAVASCRIPT

- An Interpreted language
- Embedded within HTML
- Minimal syntax-easy to learn
- Quick development
- Designed for simple, small programs
- Performance
- Procedural capabilities
- Designed for programming user events
- Easy debugging and testing
- Platform independence / architecture neutral

USING JAVASCRIPT ON A WEBPAGE

There are 3 places where scripts may be inserted:

- In the head of the page: between <head> tags
  - ✓ Scripts in the head section will be executed when CALLED.
  - ✓ Scripts in the header can't create output within the HTML document, but can be referred to by other scripts.
  - ✓ Header is often used for functions.
- In the body of the page: between <body> tags
  - ✓ Scripts in the body section will be executed WHILE the page loads.
  - ✓ Output from scripts contained in the body is displayed as part of the HTML document when the browser loads the page
- In a separate file (External file)

JAVASCRIPT BASICS (Syntax)

- Scripts in the head section:

```
<html>
<head>
<script type="text/javascript">
some statements
</script>
</head>
```

- Scripts in the body section:

```
<html>
<head>
</head>
 <body>
<script type="text/javascript">
some statements
```

```
</script>
</body>
```

- Scripts in both the body and the head section:

```
<html>
<head>
<script type="text/javascript">
some statements
</script>
</head>
<body>
<script type="text/javascript">
some statements
</script>
</body>
```

- Scripts in the external file:

```
<script language="javascript" src="jsfile.js"> </script>
```