1

HTML5

Unit Structure :

1.0 OBJECTIVES

After completing this chapter, you will be able to:

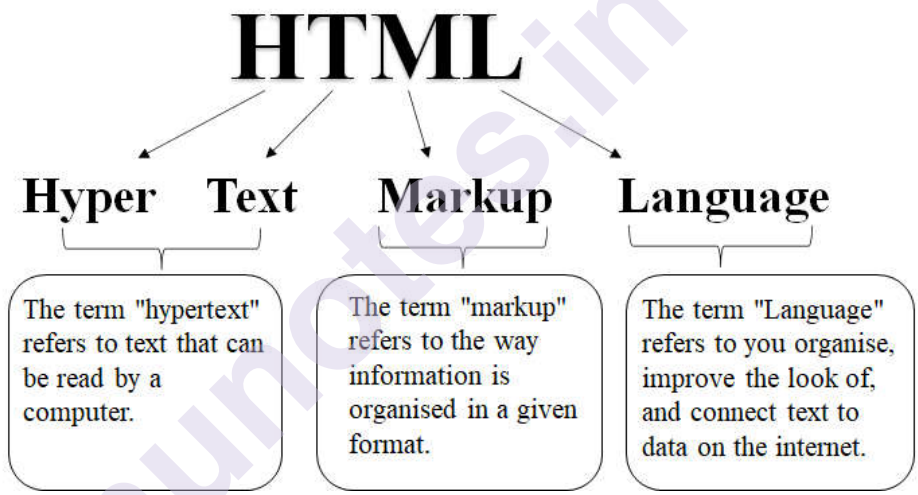
- To understand the basic concept of HTML5.
- To understand the HTML document structure.
- To understand the how to create HTML elements, attributes, tags.
- To understand the how to create HTML list, image, table, map.

- To understand the how to create FORMs in HTML.
- To understand the how to working with Multimedia.

1.1 INTRODUCTION

1.1.1 HTML

HTML stands for Hyper Text Markup Language, which is used to create web pages and apps. HTML is made up of a number of different elements. HTML components specify how the content should be displayed in the browser. Tags are used to represent HTML elements. HTML is a markup language. Notepad was used to write the text. A web browser, such as Google Chrome, interprets HTML code. HTML is case insensitive (but still lower case is generally considered easy to read) language. We can design static web pages only with HTML.



1.1.2 HTML5

HTML5, which replaces HTML 4.01, XHTML 1.0, and XHTML 1.1, is the next significant revision of the HTML standard. HTML5 is a web standard for organising and presenting content on the web. HTML5 is essentially a term for a professional touch web technologies. The HTML Living Standard is included, as well as JavaScript APIs for storage, multimedia, and device access. HTML5 supports both audio and video. HTML does not allow JavaScript to run within the web browser, whereas HTML5 does. HTML5 allows inline mathML and SVG to be used in a text. HTML5 adds new form controls including date and time, email, number, category, title, Url, search, and more. HTML5 includes a plenty of new elements. Time, footer, description, audio, article, canvas, shape and so on are some of the most essential.

1.2 FUNDAMENTAL ELEMENTS OF HTML

Tags are the parts of an HTML element that begin and end. They begin with the < symbol and terminate with the > symbol. Tags are the contents of the written inside < and >.

Between the tags, elements surround the contents. They have some type of structure or expression to them. A start tag, content, and end tag are the most common elements.

There are two types of HTML tags:

- Container Elements or Paired Tags
- Empty Elements or Singular Tags

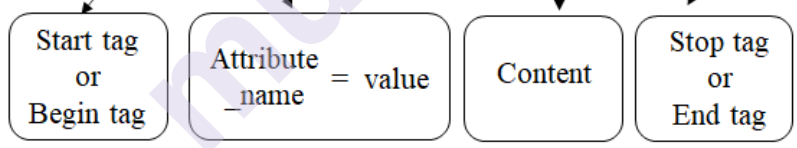
Elements form an HTML file. These elements are capable of developing web pages and defining the content on those pages. A start tag <tagName>, a closing tag </tagName>, and content added between them make an element in HTML. A collection of start tag, attributes, end tag, and content between them forms an element in HTML. All HTML elements can have attributes. In HTML, an attribute offers additional information about an element and is added within the start tag. There are two fields in an HTML attribute: name and value.

Syntax:

<tagName attribute_name= "value"> Content </ tagName>

Ex.

<h1 class="country"> India is my country </h1>



1.2.1 HTML Document Structure

The structure of a typical HTML document is as follows:

```
<!DOCTYPE html>
<html>

<head>
Tags associated to document head
</head>
```

```
<body>
Tags associated to document body
</body>


</html>
```

Where

- The declaration<!DOCTYPE html> indicates that this is an HTML5 document.
- The root element of an HTML page is the <html> element.
- The html page's meta data is stored in the <head> element.
- The <title> element gives the HTML page a title (which appears in the browser's title bar or in the tab).
- The <body> element specifies the document's body and serves as a container for all visible information, including headers, paragraphs, pictures, hyperlinks, tables, lists, and so on.

1.2.2 Creating an HTML Document and Saving It:

Step 1 : Open a text editor and write HTML code (here we are using Notepad)

Step 2 : Select File from the File Menu.

Step 3 : Select Save As.

Step 4 : Type a Name with extension .HTM or .HTML.

Step 5 : Select the location where you want to save the file.

Step 6 : Click on Save.

1.2.3 Viewing an HTML Document:

Step 1 : Navigate to the HTML file's location.

Step 2 : Right-click on the HTML file and select the option to open with.

Step 3 : Choose a browser (Google Chrome).

Step 4 : The webpage will be seen.

1.2.4 Viewing an HTML Document in Another Way:

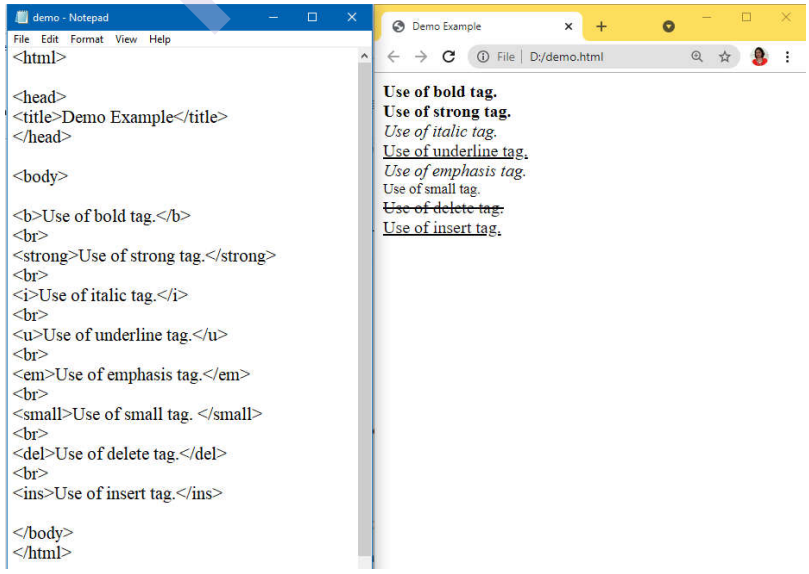
If you have Google Chrome selected as your default browser, you can just double-click the HTML file and it will open in Chrome.

1.3 FORMATTING TEXT IN HTML

Formatting elements were designed to display special types of text:

- ** tag** : It's known as Bold text tag and used to displayed content in bold. …….. is the syntax for the tag.
- ** tag** : It's known as Important text tag and used to displayed text as strong text. < strong >……..</ strong > is the syntax for the < strong > tag.
- ** tag** : It's known as Italic text tag and used to displayed content in italic. …….. is the syntax for the tag.
- **<u> tag** : It's known as Underline text tag and used to displayed content with underline. <u>……..</u> is the syntax for the <u> tag.
- ** tag** : It's known as Emphasized text tag and used to displayed text as emphasized text. …….. is the syntax for the tag.
- **<small> tag** : It's known as Smaller text tag and used for identifying secondary importance such as copyright, side comments, and legal notices. <small>……..</ small > is the syntax for the < small > tag.
- **<del> tag** : It's known as Deleted text tag and used to indicate the area of the document where text has been deleted/removed. ~~……..~~ is the syntax for the <del> tag.
- **<ins> tag** : It's known as Inserted text tag and used to represent the newly inserted text It normally appears with a text underline, however this may be altered using a CSS attribute. <ins>……..</ins> is the syntax for the <ins> tag.

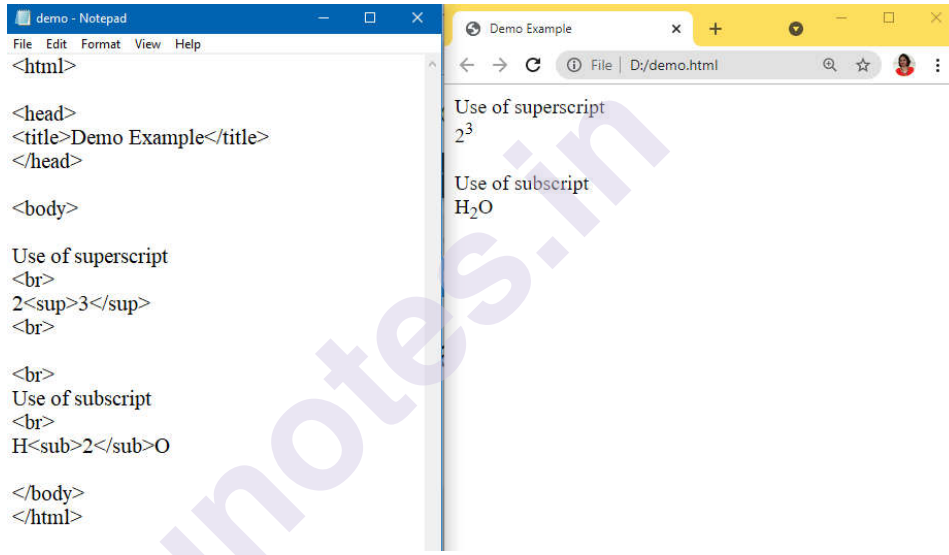
Example of formatting text tags-



<sub> tag : It's known as Subscript text tag and used to specify text that is subscripted. The text inside a <sub> has a lower baseline and a smaller font than the surrounding content. The <sub> tag can be used to display mathematical or chemical formulas. < sub >……..</ sub > is the syntax for the < sub > tag.

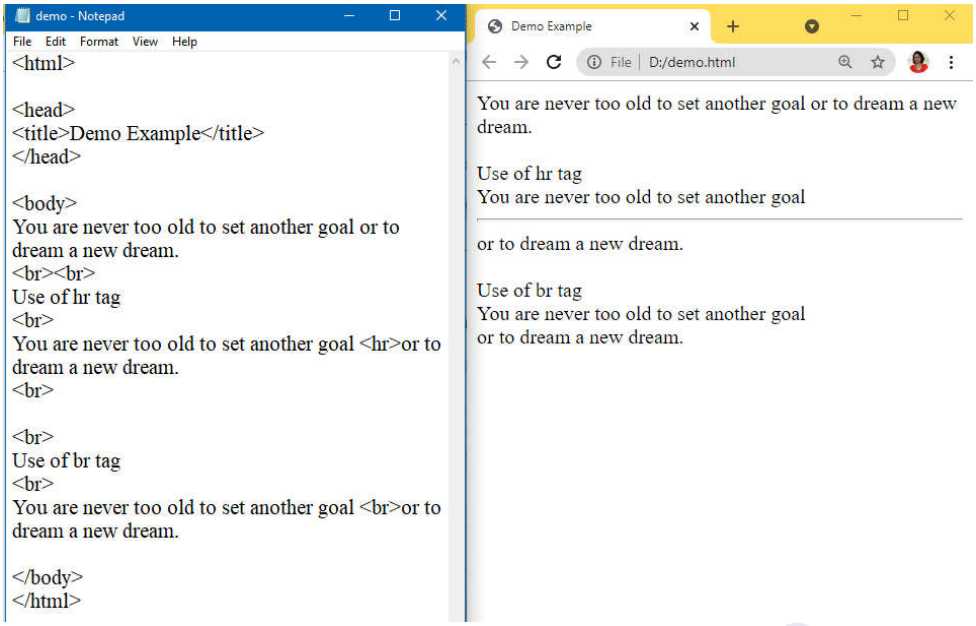
- **<sup> tag** : It's known as Superscript text tag and used to define content that is written in superscript. The content included within the <sup> element has a higher baseline and a smaller font size than the surrounding text. Mathematical formulae and footnotes can be defined with the <sup> tag. < sup >……..</ sup > is the syntax for the < sup > tag.

Example of <sub> and <sup> tag-



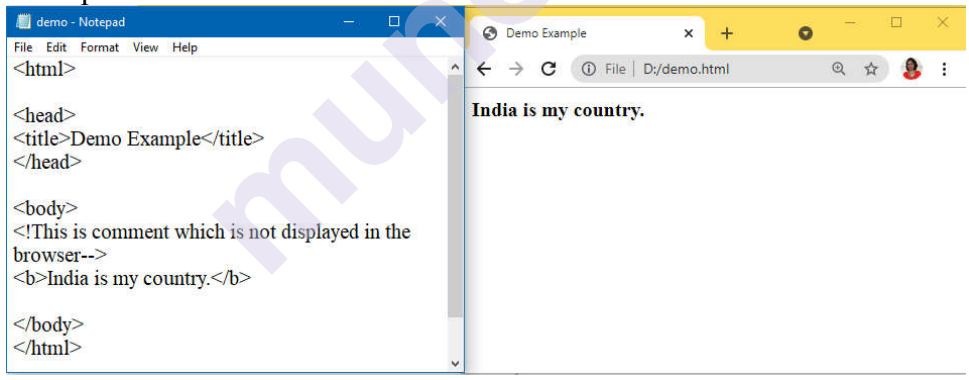
- **
 tag** : It's known as Line Break tagand used to break a line and display the text from the next line. The HTML br tag can be used in two ways
 or
. The closed br tag
 is suggested because it is supported by both HTML and XHTML.
is the syntax for the
tag.

- **<hr> tag** : It's known as Horizontal Ruleand used to produce the horizontal line. It creates a horizontal divide in the content. In HTML, it's also known as a Horizontal Rule. <hr>is the syntax for the <hr>tag. Example of <hr> and
 tag:



- **COMMENT Tag**: HTML comments are not visible in the browser, however they can assist in the documentation of your HTML source code. You can use comments to describe your code, which will help enormously when you go back to update the source code. If you have a lot of code, this is very helpful. Comments are given between <!-- type text-- >

  Example of comments in html-



1.4 ORGANIZING TEXT IN HTML

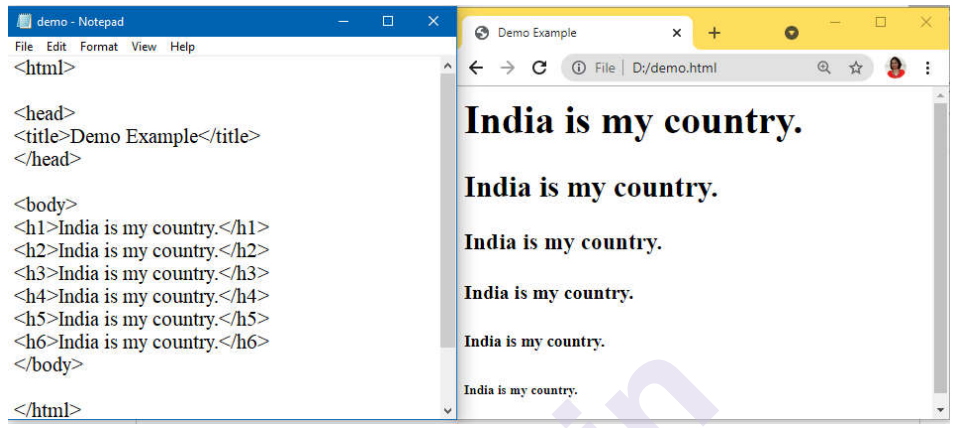
1.4.1 HTML Heading

A title or a subtitle that you wish to appear on the web page may be defined using an HTML header tag. When you use the heading tags <h1>.....</h1>, the text is shown in bold on the browser, and the size of the text is decided by the number of headings. When a heading is displayed, the browser inserts one line before and one line after it.

The <h1> to <h6> tags are used to specify the six different HTML headers.

The greatest heading tag is h1 and the lowest is h6. As a result, the most significant heading is h1 and the least important is h6.

Example of heading tags in html-

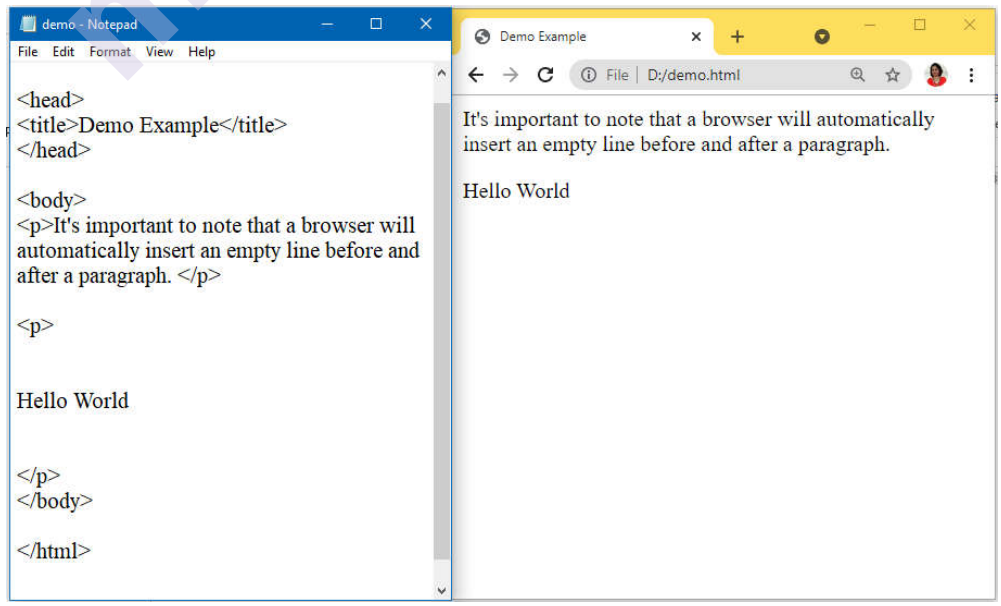


1.4.2 HTML Paragraph

A paragraph in a webpage is defined by the HTML paragraph tag. It's important to note that a browser will automatically insert an empty line before and after a paragraph. The <p> element in HTML denotes the start of a new paragraph.

When you use a lot of spaces inside the HTML p> tag, the browser eliminates the excess spaces and lines when the page is shown. The browser considers the number of spaces and lines to be one. < p >……..</ p > is the syntax for the < p > tag.7.5

Example of paragraph tag in html-

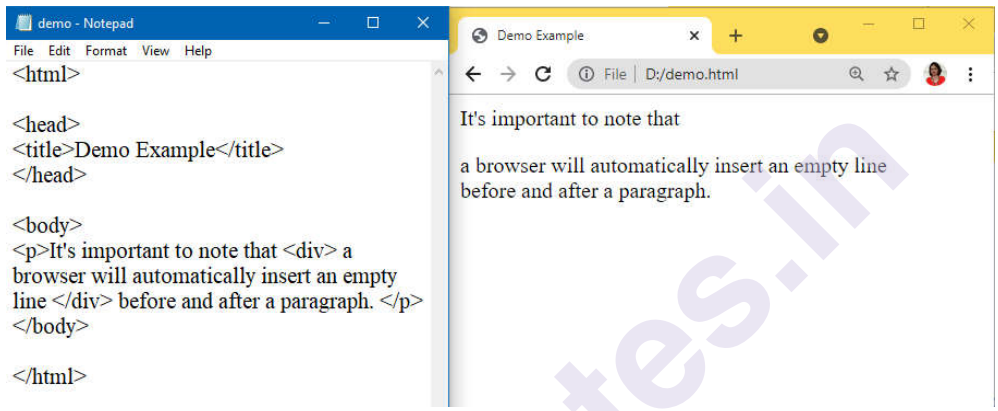


1.4.3 HTML Division

In an HTML document, the div> element denotes a division or section. The tag serves as a container for HTML components, which may then be decorated using CSS or modified using JavaScript. The class or id attribute can be used to style the element. It is a block level tag

It's a term that refers to a collection of HTML tags that may be used to build sections and apply styles to them. < div >……..</ div > is the syntax for the < div > tag.

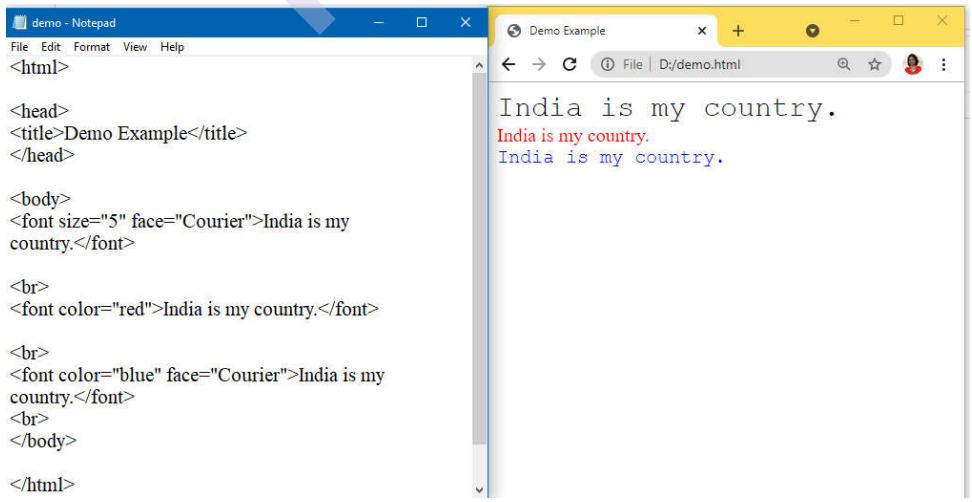
Example of division tag in html-



1.4.4 Font

HTML5 does not support this. The font style for the text within it is defined by the HTML element. In an HTML document, it specifies the font size, colour, and face of the text. …. is the syntax for the <font> tag with attributes size, color and face.

Example of font tag in html-



1.4.5. List in HTML

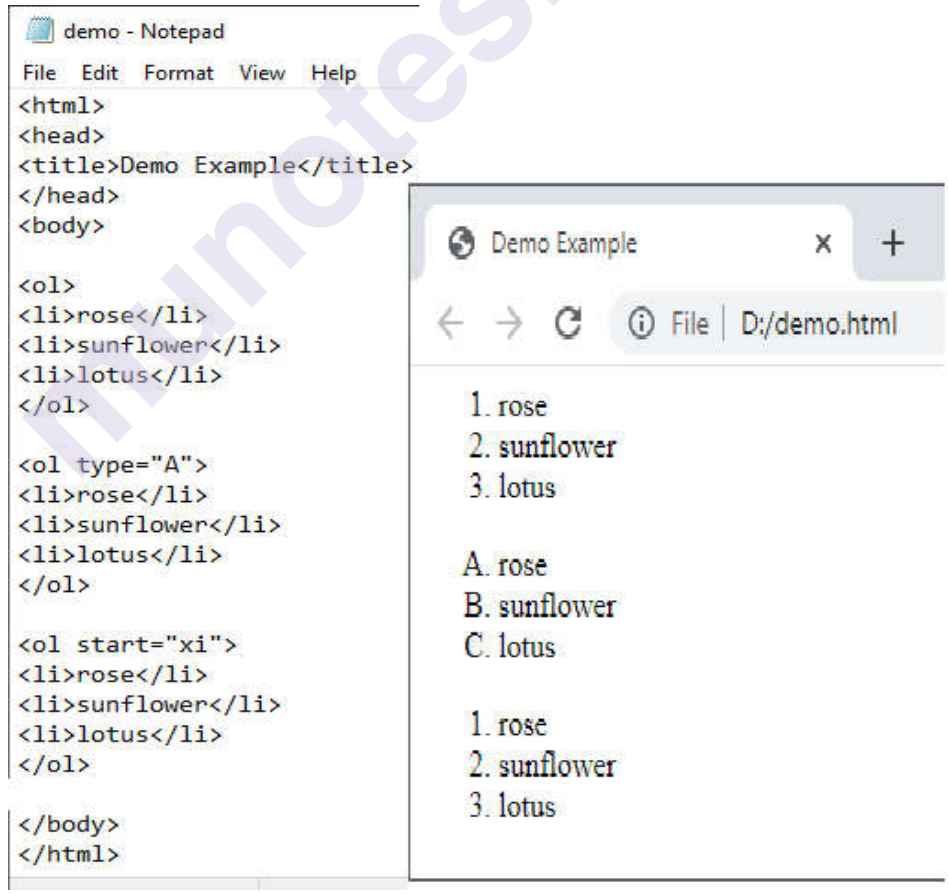
HTML List HTML lists allow web developers to group a set of related items in lists.HTML Lists are used to specify lists of information. All lists may contain one or more list elements. There are three different types of HTML lists:

- Ordered List or Numbered List (ol)
- Unordered List or Bulleted List (ul)
- Description List or Definition List (dl)

Ordered List or Numbered List:

By default, all list elements in ordered HTML lists are marked with numbers. It's also known as a numbered list. The <ol> tag starts the ordered list, while the <li> tag starts the list items. The number starts at 1 and increases by 1 each time. The Kind element of the <ol> tag is used to specify the type of number for list items. 1 is the default type. The TYPE attribute of the <ol> tag has several options. 1-Number, I-Upper Roman, i-Lower Roman, A-Upper Case, a- Lower case.

Example of order list in html-



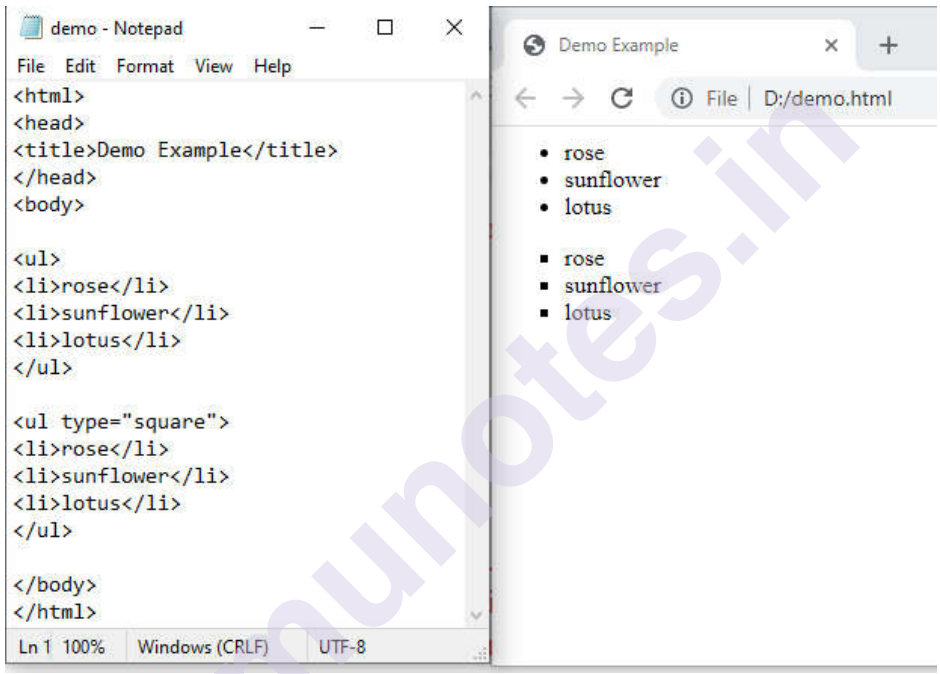
Unordered List or Bulleted List:

All list items in an HTML Unordered list are marked with bullets. It's also referred to as a bulleted list. The ul> tag starts the unordered list, while the li> tag starts the list items. The ul> tag has a TYPE attribute that specifies the type of bullet for each list item.

Disc is the default type. The ul> tag's TYPE attribute has the following options::

1.Square 2.Circle 3.Disc

Example of unorder list in html-



Description List or Definition List:

HTML Description List is a list style that HTML and XHTML both support. It's also known as a definition list, because the words are listed alphabetically, like in a dictionary.

The following three tags are included in the HTML definition list:

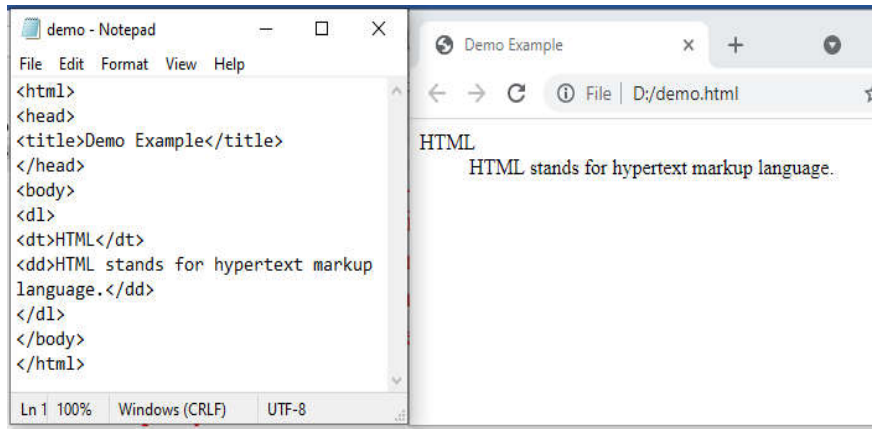
<dl>-Defines the beginning of the list

<dt>-Specify a term

<dd>-Specify term definition

</dl>-Defines the end of the list

Example of unorder list in html-



1.5 LINKS AND URLS IN HTML

A link is connection between two web pages. Hyperlinks are HTML links. You can jump to another document by clicking on a link. The mouse arrow will change into a tiny hand when you move the mouse over a link.

The href property, which identifies the link's destination, is the most important attribute of the <a> element. The section of the link text that will be visible to the reader is the link text.

The reader will be directed to the given URL address by clicking on the link text.

In all browsers, links will display as follows by default:

- link : A link that has not been visited is highlighted in blue.
- vlink : A visited link is highlighted in purple.
- alink : An active link is highlighted in red and underlined.

Syntax:

link text

where

href= it is the link's destination address

link text = it is the visible part of the link

Ex.

 Google

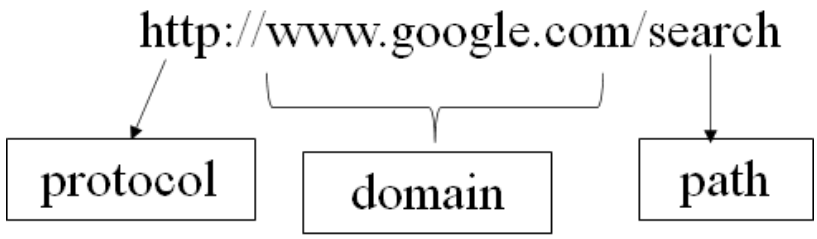
Links will open in the current window or frame by default. If you want the link to open in a different window or frame than the one it's in, you'll need to add a target attribute in <a> element

The following are predefined targets:

- _blank: opens the page in a new browser window.
- _self: the page is loaded into the current window.
- _parent: loads the page into the frame above the one in which the hyperlink is located.

- _top: closes all frames and loads the page in the whole browser window.

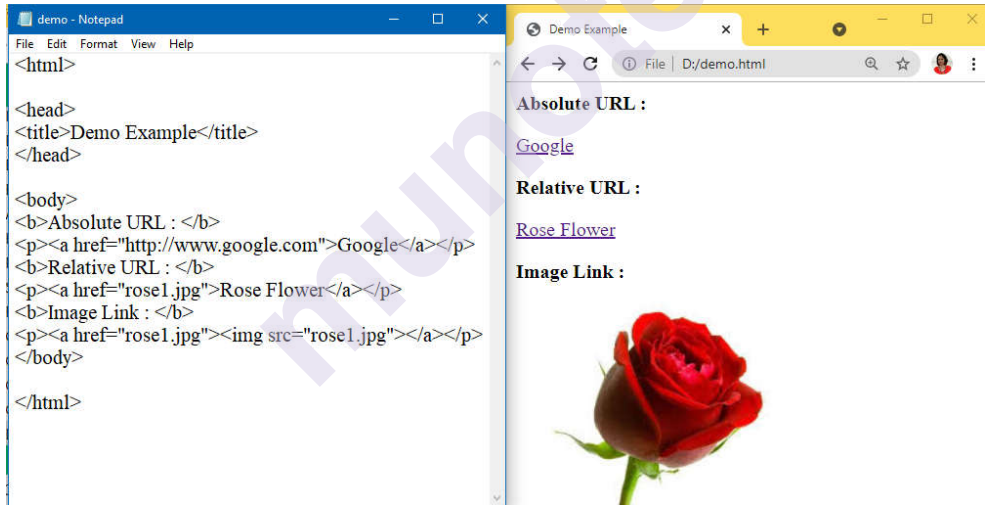
URL stands for Uniform Resource Locator. It refers to an internet/web address for a resource. The following elements make up a URL:



URL having two types - A relative URL and an absolute URL. A relative URL begins with the forward slash and directs the browser to stay within the current site. An absolute URL is the full URL of the page that you link to.

Put the <img> element within the <a> tag to use an image as a link. With the SRC property, the img> element is used to display an image in an HTML page.

Example of <a>tag in html-



1.6 TABLES IN HTML

HTML tables allow website designers to organise data such as text, photos, links, and other tables into rows and columns of cells. The <table> element is used to build HTML tables, with the <tr> tag used to produce table rows and the <td> tag used to create data cells. By default, the items under td> are regular and left aligned. The <th> element is used to specify the table heading.

<table> tag:

<table>: specifies a table.

<tr>: specifies a row in a table.

<th> : specifies a header cell in a table.

<td> : specifies a cell in a table.

<caption> : specifies the table caption.

<colgroup> : specifies a group of one or more columns in a table for formatting.

<col> : used with <colgroup> element to specify column properties for each column.

<table> attribute:

border: pixels Specifies the border width. A value of "0" means no border.

align: Used to define Visual alignment in right, left, centre or justify.

bgcolor: The background colour of the tableis defined in rgb(x,x,x) , #hexcode or colorname.

cellpadding: The distance between the cell borders and their contents is defined in pixels or %.

cellspacing : The distance between cells is defined in pixels or %.

width : The width is defined in pixels or %.

Example of <table> in html-

demo - Notepad

File Edit Format View Help

```
<html>
<head>
<title>Demo Example</title>
</head>
<body>
<table border="1" cellpadding="7" cellspacing="7" bgcolor="Yellow">
<tr>
<th>Class</th>
<th>Year</th>
</tr>
<tr>
<td>FY</td>
<td>Comp. Sci.</td>
</tr>
<tr>
<td>SY</td>
<td>Comp. Sci.</td>
</tr>
</table>
</body>
</html>
```

Demo Example

File | D:/demo.html

Class	Year
FY	Comp. Sci.
SY	Comp. Sci.

1.6.1 Colspan and Rowspan attributes of <table>

A table is divided into rows, with each row containing cells. We may need the table to spread across (merge) multiple columns or rows. We can use the Colspan and Rowspan attributes in this case.

1.Colspan

The colspan parameter specifies how many columns a cell should display horizontally.

Eg:<td colspan=2>

This code will merge two cells horizontally into one.

2.Rowspan

The rowspan parameter is used to define how many rows a cell should display vertically.

Eg:<td rowspan=2>

This code will merge two cells vertically into one.

Before Rowspan After Rowspan.

1.7 IMAGES ON A WEB PAGE

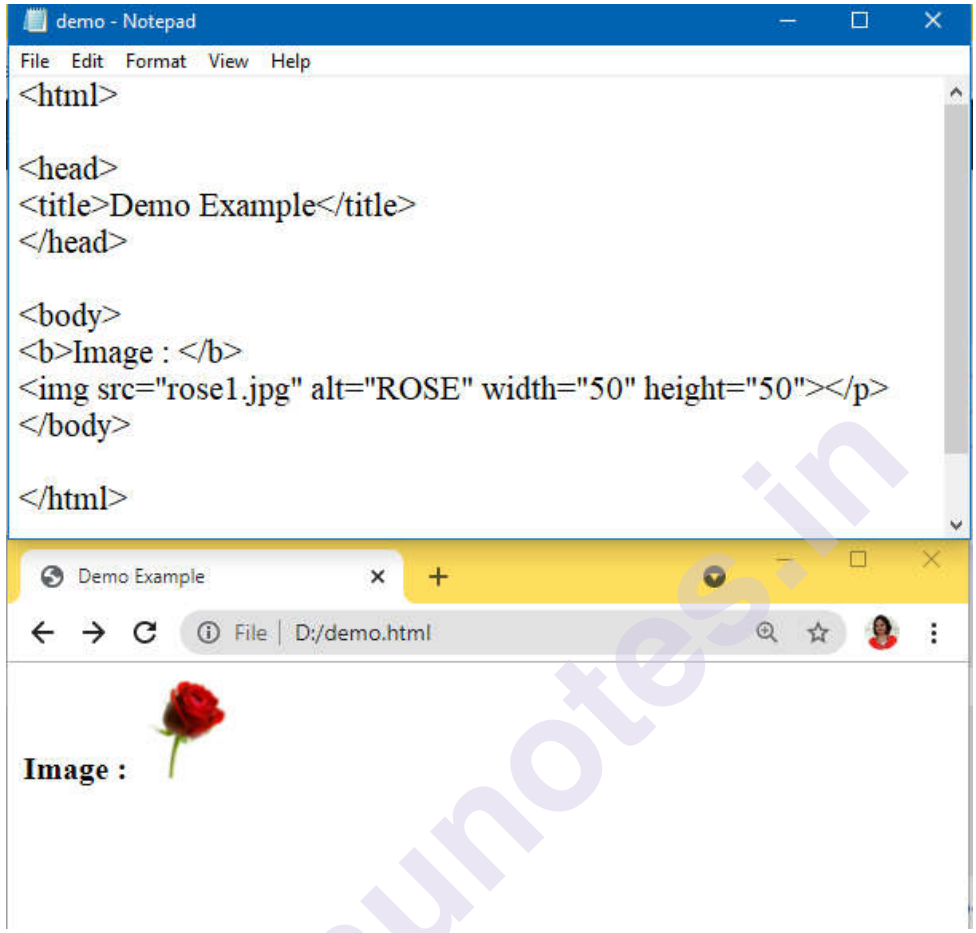
The <img> tag in HTML is used to show an image on a web page. The HTML <img> tag is an empty tag with just attributes; no closing tags are used in the HTML image element.

<img> has the following attributes:

- src: is used to indicate the image's path.
- alt: is used to specify an image's alternate text. It is useful since it tells the viewer about the picture's meaning and also because if the image cannot be displayed due to a network issue, this alternate text will be displayed.
- height: is used to determine the image's height.
- width: is used to specify the image's width.
- usemap: is used to specify an image as a client-side image map.
- ismap: is used to specify an image as a server-side image map.

- align: is used to specify the image alignment with value top, bottom, middle, left and right.
- border: is used to specify the width of image border.

Example of image in html-



1.8 IMAGE FORMATS

- **BMP** : Microsoft Windows produced the bitmap (.bmp) format, which is widely used on the internet because to Internet Explorer's popularity. Supports 16.7 million colours and is primarily uncompressed, making it somewhat big.

- **JPEG** : JPEG (Joint Photographers Expert Group) is a compression standard (not an image format) with the extension.jpg or .jpeg , .jpg. .jpg offers 16.7 million colours with a tiny file size, and is frequently used for high-color photography. However, when compressed, certain details are lost (lossy compression).

- **GIF**: GIF (Graphic Interchange Format) is a popular image format that loses no information when compressed (i.e lossless compression). The disadvantage of.gif is that it only supports an 8-bit colour palette, which means that each picture may only have 256 colours, making it unsuitable for images with numerous colours. Most importantly,.gif

allows for animation; a single.gif file may contain several frames, which can then be played in order to produce an animation effect.

- **PNG**: PNG (Portable Network Graphics) is an open-source image format that may be used for any sort of image on the Internet. File sizes are greater than.jpg (since compression is lossless). .png allows for 16.7 million colour combinations and 256 transparent colours. Suitable for images that just require a few colours. Low colour depth data may be compressed into very tiny files.

1.9 IMAGE MAPS

The HTML mapping image is a client-side image map with clickable areas in an HTML document that act as hyperlinks. The HTML <map> tag is used with the <area> tag to create a client-side image map. An image map is made up of a picture with clickable regions. When you click on the image, it will open to a new or previously selected location. The <map> tag can include several <area> components, each of which defines the area's coordinates and type. You may simply link any portion of a image to other documents using the <map> tag without splitting the image.

Syntax:

```
<img src=" " alt=" "  usemap="#nm_val">
<map name= " nm_val ">
<area shape=" " coords=" " href=" " alt=" ">
</map>
```

You must define the shape of the clickable region, and one of the following values can be used:

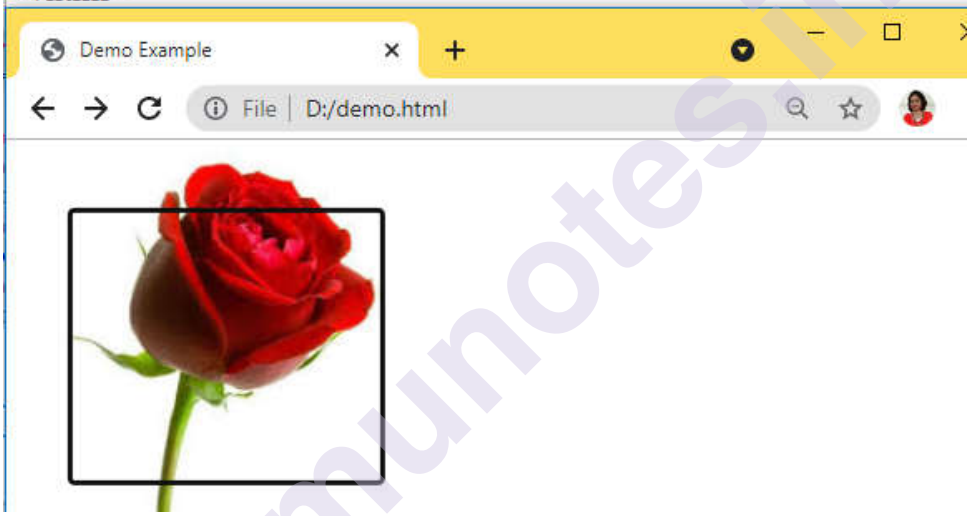
- rect – used to define a rectangular region
- circle - used to define a circular region
- poly - used to define a polygonal region
- default - used to define the entire region

Example of image map in html-

demo - Notepad

File Edit Format View Help

```
<html>
<head>
<title>Demo Example</title>
</head>

<body>
<img src="rose1.jpg" alt="Workplace" usemap="#abc" width="300"
height="300">
<map name="abc">
  <area shape="rect" coords="40,45,275,250" alt="ROSE"
href="rose.html">
 </map>
</body>
</html>
```



1.10 COLORS

Colors play a significant role in giving your website a pleasing appearance. The <body> tag can be used to set colours for the entire page, or the bgcolor attribute can be used to set colours for individual tags.

The following attributes of the <body> tag can be used to set different colours:

- bgcolor: specifies the colour of the page's background.
- text: specifies the colour of the body text is chosen by text.
- alink: specifies changes colour of active or selected links.
- link: specifies the colour of linked text is determined by the link command.
- vlink: assigns a colour to visited links, or linked text that has already been clicked.

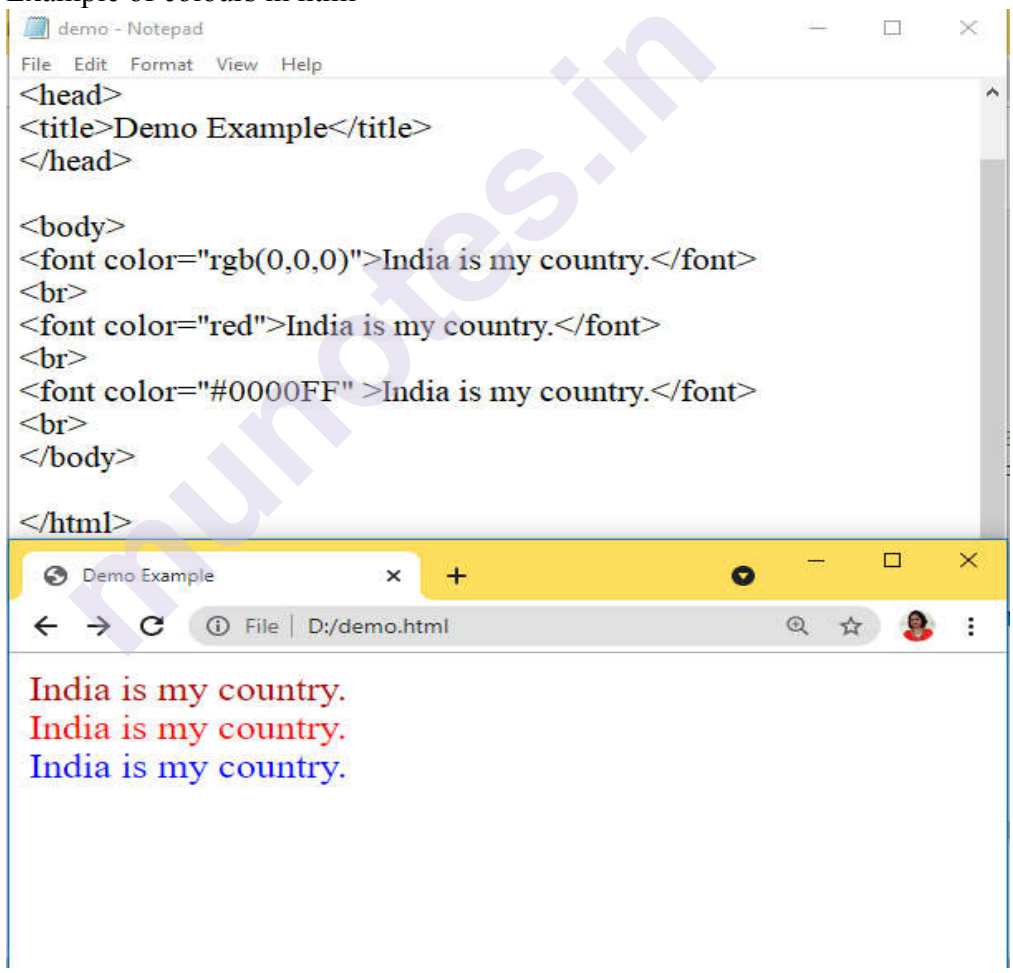
Methods of HTML Color Coding:

- Color name: Color names can be specified directly, such as green, blue, or red. W3C has listed 16 basic color names. Ex. Yellow, Red etc.

- Hex code: Hex codes are a six-digit code that represents the proportion of red, green, and blue in a colour. A hexadecimal is a 6 digit representation of a color as #RRGGBB Ex. #110033, #111111 etc.

- rgb(): The rgb( ) parameter is used to specify colour decimal or percentage values. This property accepts three values: one for red, one for green, and one for blue. The value might be a percentage or an integer between 0 and 255. Ex. rgb(0,0,0), rgb(255,0,0) etc.

Example of colours in html-



1.11 FORMs in HTML

An HTML form is a part of a document that contains controls like text fields, password fields, checkboxes, radio buttons, submit buttons, menus, and so on. An HTML form allows a user to enter data that will be transmitted to a server for processing, such as a user's name, email address, password, phone number, and other information.

Syntax for creating form:

```
<form action="url" method="GET/POST">
  input controls
</form>
```

<form> attributes:

- action: When a form is submitted, the action specifies where the form data should be sent.
- name: Specifies the name of the form.
- method: Specifies the HTTP method to use when delivering form-data. Get or Post is the option.
- target: Indicates where the response received after submitting the form should be displayed.

1.12 INTERACTIVE ELEMENTS

Anything on a web page that the user interacts with is considered an interactive element. Consider an interactive element to be something that forces the user to take action. You can use many types of form controls to collect data using an HTML form which are nothing but interactive elements.

Some very common attributes regarding form elements:

- name : The name attribute is a unique identifier that is used to send data via HTML form elements.
- id : The id attribute is a unique identifier for HTML tags that allows the browser to identify the element when it is displayed.
- value : An <input> element's value is specified via the value attribute.

One or more of the following form elements can be contained in the HTML <form> element:

1) Text Input Controls

2) Checkboxes Controls

3) Radio Button Controls

4) Select Box Controls

5) File Upload boxes

6) Button Controls

7) Hidden Form Fields

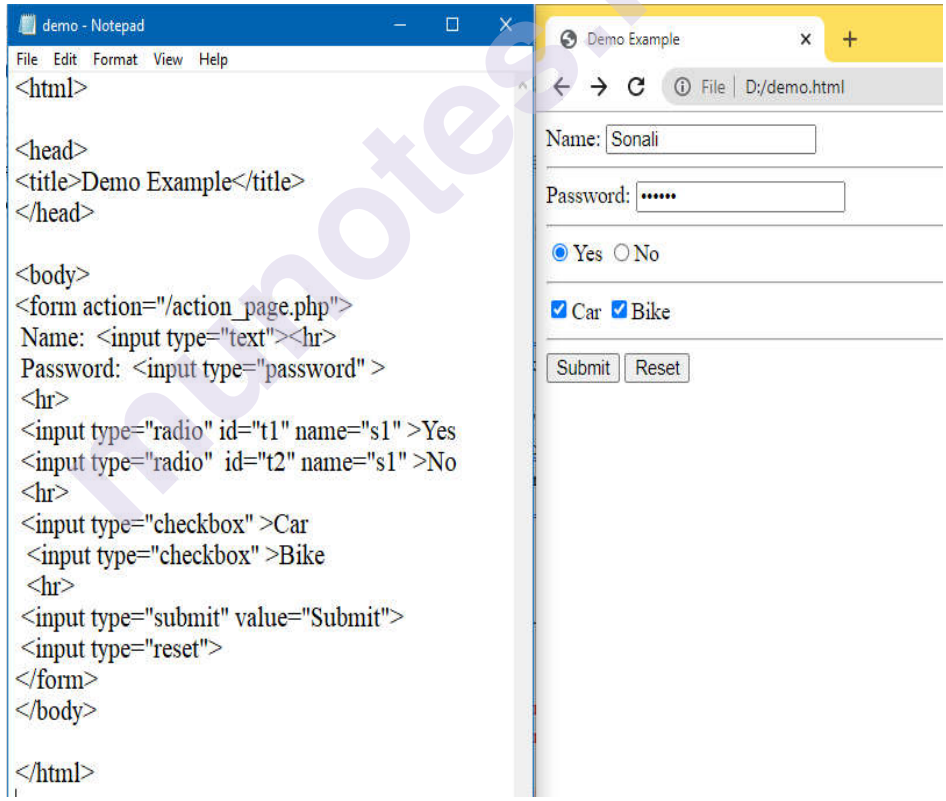
1. <input> :

The input> element in HTML is the most often used form element. Depending on the type attribute, an input> element can be presented in a number of ways.

The following are the various HTML input types:

- <input type="text"> : It specifies a single-line text input field
- <input type="radio"> : It specifies a radio button. A user can select ONE of a limited number of options using radio buttons.
- <input type="checkbox"> : It specifies a checkbox (for selecting zero or more of many choices)
- <input type="submit"> : It specifies a submit button (for submitting the form). A button for submitting form data to a form-handler is defined by the <input type="submit">.
- <input type="button"> : It specifies a clickable button
- <input type="password"> : It specifies a single-line text password field
- <input type="reset"> : It specifies a reset button that will reset all form values to their default values.

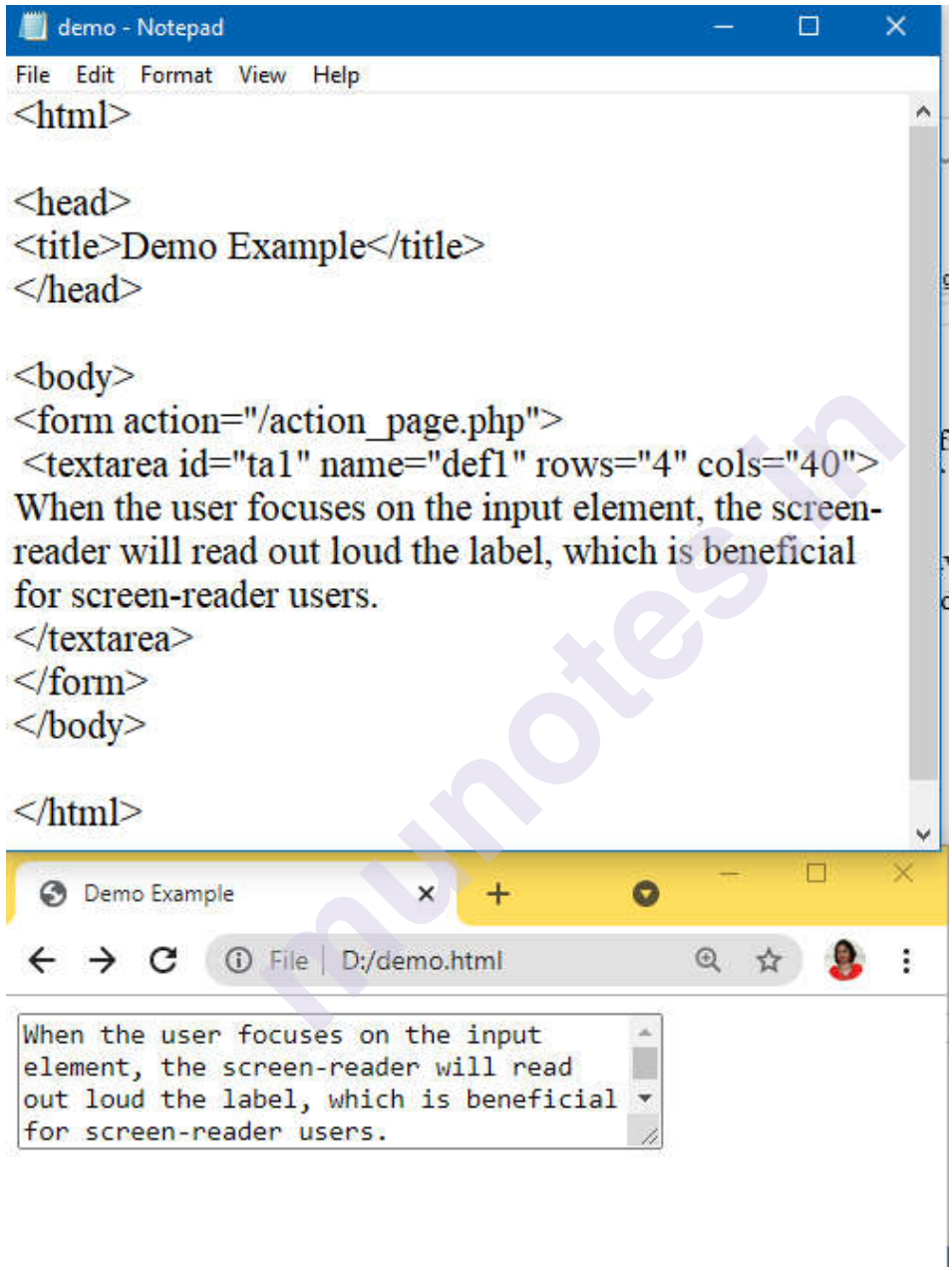
Example of form elements in html-



2. <textarea> :

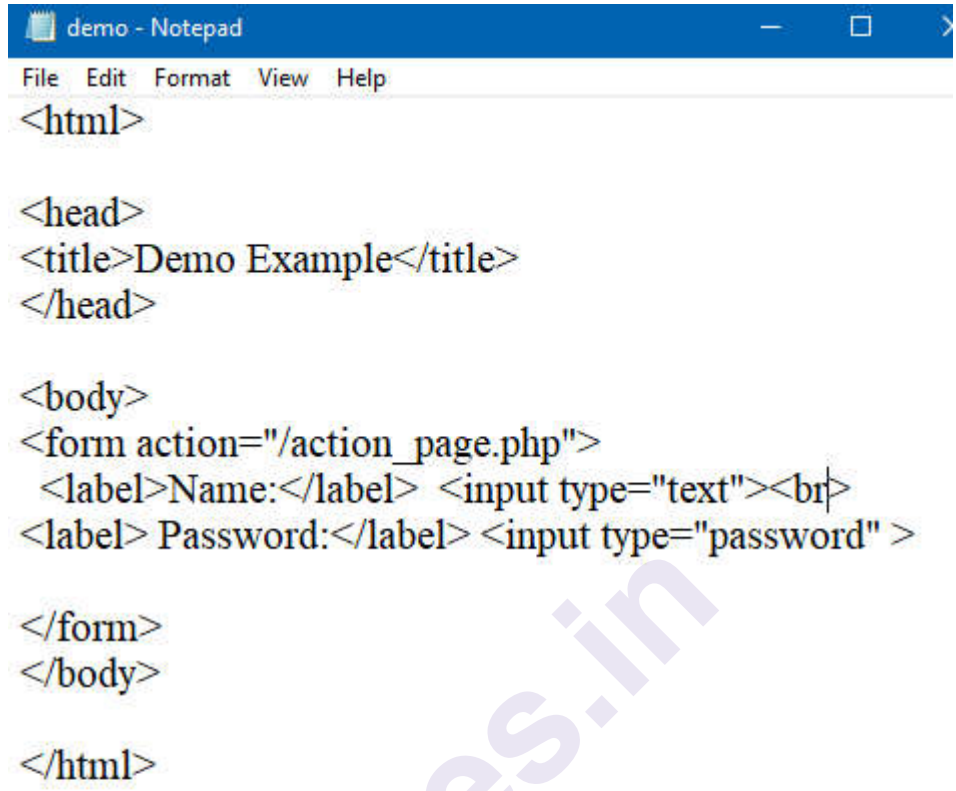
A multi-line text input control is defined by the <textarea> tag. The <cols> and <rows> characteristics determine the size of a text area.

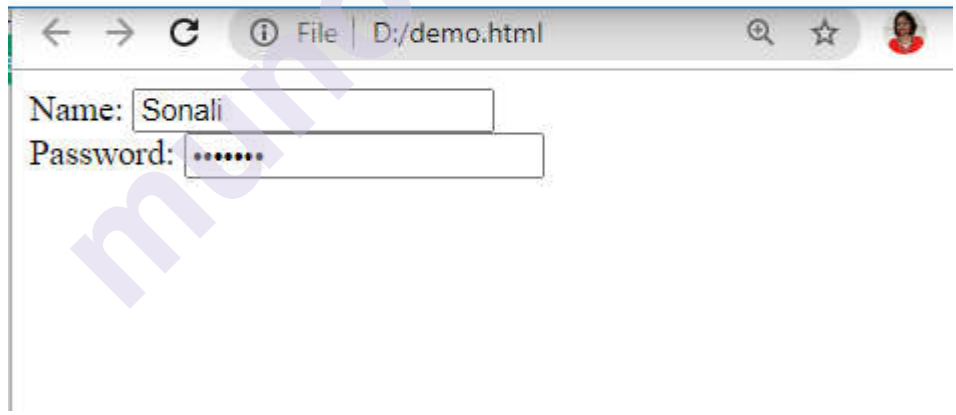
Example of <textarea> in html-



3. <label> : Many form elements have a label defined by the <label> tag. When the user focuses on the input element, the screen-reader will read out loud the label, which is beneficial for screen-reader users.

Example of <label> in html-





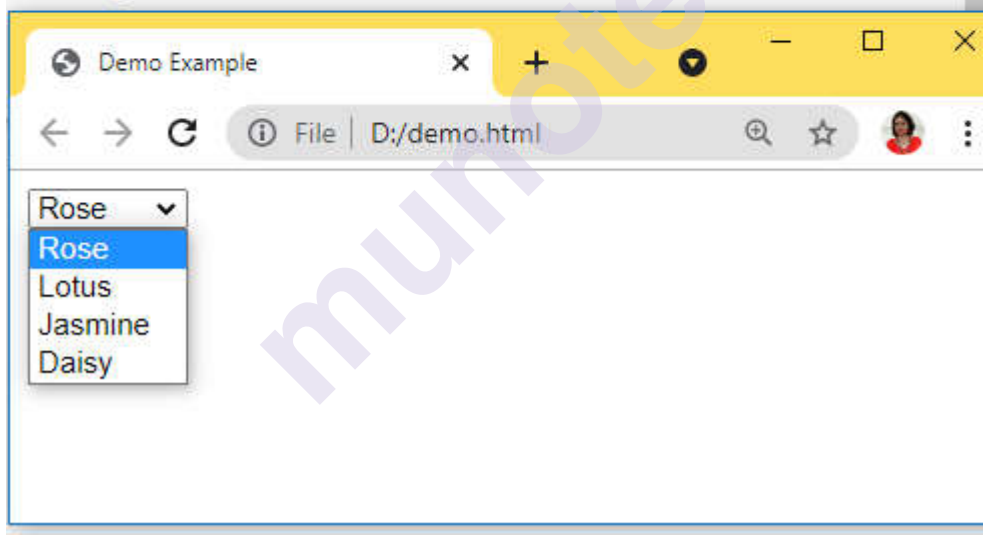
4. <select> : A drop-down list is created with the <select> element. In most cases, the <select> element is used in a form to collect user input. The available options in the drop-down list are defined by the <option> tags inside the <select> element.

Example of <select> in html-

demo - Notepad

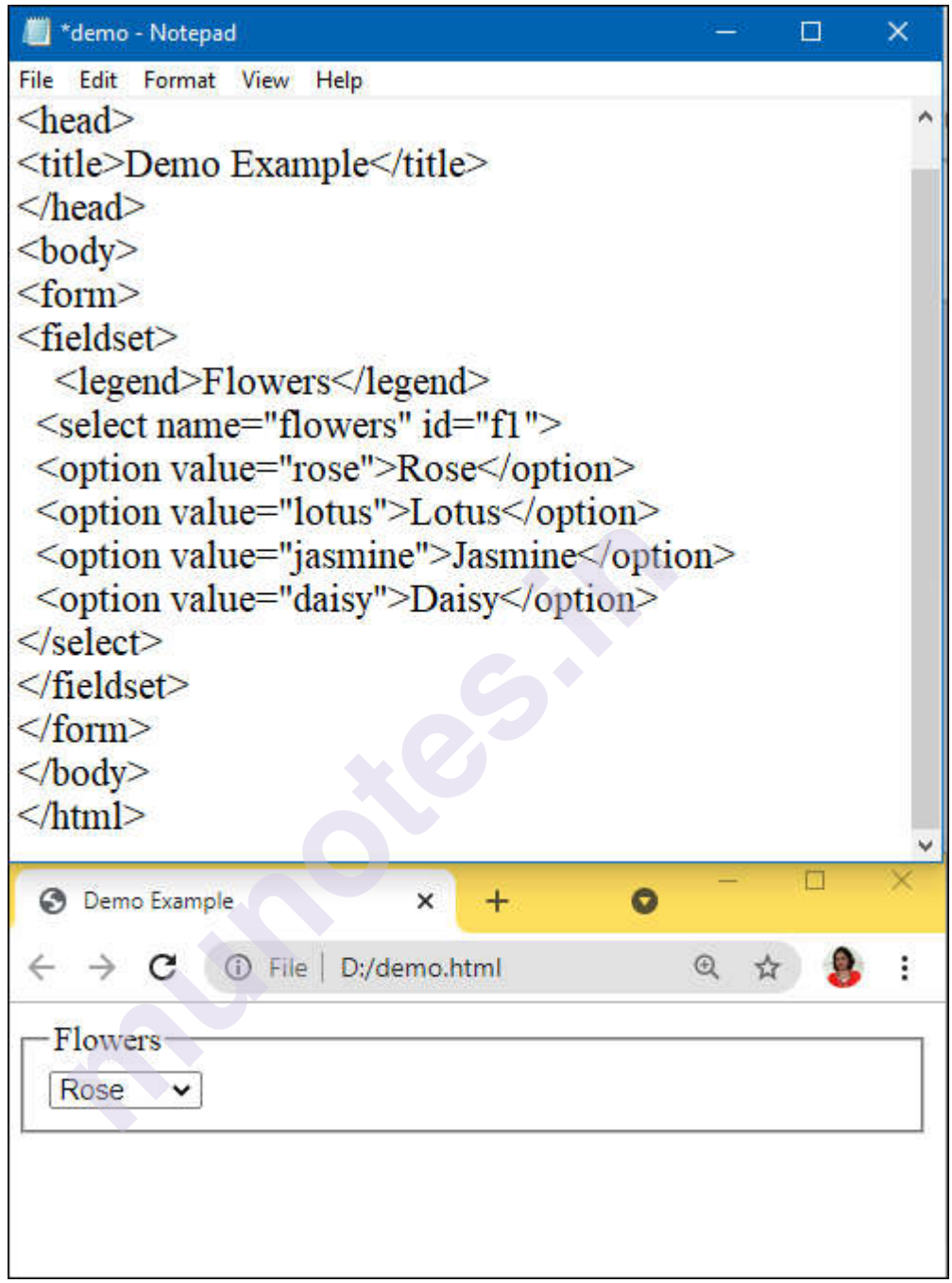
File Edit Format View Help

```
<html>
<head>
<title>Demo Example</title>
</head>

<body>
<form action="/action_page.php">
  <select name="flowers" id="f1">
  <option value="rose">Rose</option>
  <option value="lotus">Lotus</option>
  <option value="jasmine">Jasmine</option>
  <option value="daisy">Daisy</option>
</select>
</form>
</body>
```



5. <fieldset> and <legend> : In a form, the <fieldset> element is used to group together relevant data. The caption for the <fieldset> element is defined by the <legend> element.

Example of <fieldset> and <legend> in html-



1.13 WORKING WITH MULTIMEDIA

Almost everything that can be heard or seen may be considered multimedia (like- sound, music, images, records, videos, films, animations, etc.). It is available in a variety of forms. Multimedia components in many forms and types can be found on web sites. Various multimedia tags in HTML allow you to add a variety of multimedia files to your website. The following are some of the tags:

- <audio>: Use for displaying a audio file on the web page,
- <video>: Use for displaying a video file on the web page,

- <embed>: Use for embedding multimedia files on the web page,
- <object>: Use for embedding multimedia files on the web page.
- <iframe>: Use for embedding other web pages.

Media files include multimedia components such as audio and video. Looking at the file extension is the most common approach to figure out what type of file it is. Formats and extensions for multimedia files include: .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

1.13.1 HTML elements for inserting Audio on a web page

Music and other audio clips are created using the HTML audio tag. The HTML 5 audio element currently supports three file formats which are given below:

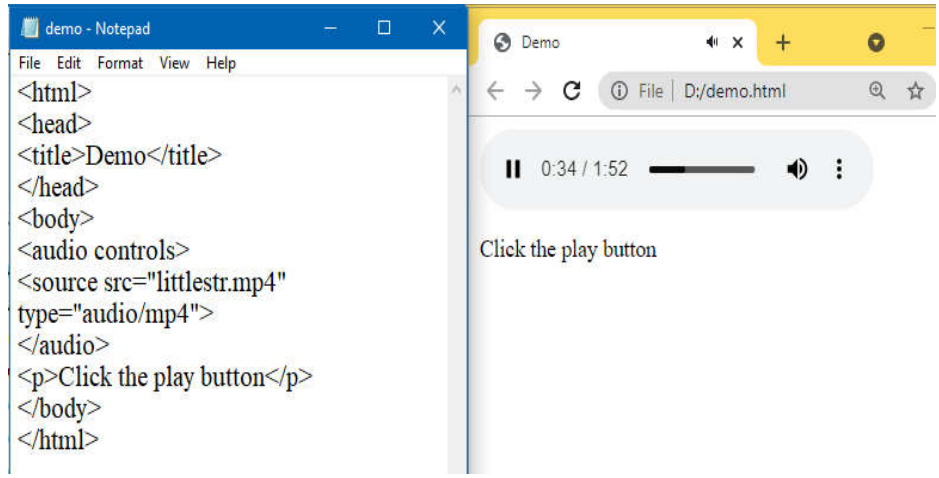
a). mp3
b). wav
c). ogg

HTML5 allows you to use video and audio controls. The multimedia elements are played using Flash, Silverlight, and other comparable technologies.

<audio> Tag Attributes:

- controls: It specifies the audio controls (play/pause buttons) which are shown.
- autoplay: When enabled, the audio will begin playing as soon as it is ready.
- autoplay: When enabled, the audio will begin playing as soon as it is ready. When an audio file is finished, loop signals that it will begin playing again.
- muted: It's being used to turn off the audio output while it's muted.
- preload: This indicates the author view that when the page loads, it should upload an audio file.
- src: This specifies the audio file's source URL.

Example of inserting audio-



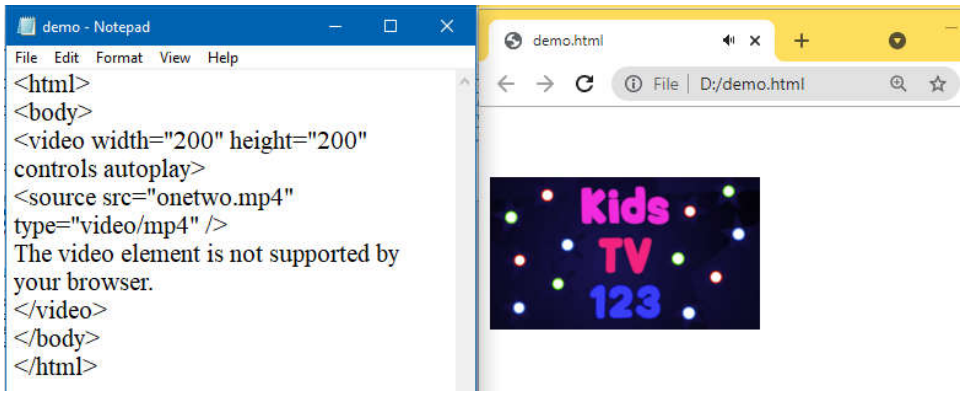
1.13.2 HTML elements for inserting Video on a web page

The <video> tag in HTML allows you to embed video in your document. MP4 and OGG are the video formats that HTML supports. The <source> element is used to identify media, as well as the kind of media and other properties. In the video element, several source elements are allowed, and the browser will use the first format it recognises.

<video> Tag Attributes:

- controls: It describes the video controls, which include the play/pause buttons..
- height: It's used to change the height of the video player.
- width: The width of the video player is specified with this attribute.
- poster: When the video isn't playing, the poster is used to define the image that appears on the screen.
- autoplay : The video will start playing as soon as it is available, according to autoplay.
- loop : When the video file is ended, the loop specifies that the video file will begin again from the beginning.
- muted: This option is used to silence the visual output.
- preload: The author is instructed to upload a video file when the page loads.
- src: This specifies the URL of the video file's source.

Example of inserting video-



1.14 SUMMARY

In this chapter we have covered Fundamental Elements of HTML, structure of html document, how to Formatting Text in HTML, Organizing Text in HTML. How to use Links and URLs in HTML, Tables in HTML, Images on a Web Page, Image Formats, Image Maps, Colors.

How to create FORMs in HTML. How to insert HTML elements for inserting Audio on a web page and Video on a web page.

1.15 REFERENCE FOR FURTHER READING

- HTML 5 Black Book, Covers CSS 3, JavaScript, XML, XHTML, AJAX, PHP and jQuery, 2ed, Dreamtech Press.
- HTML, XHTML, and CSS Bible Fifth Edition, Steven M. Schafer, WILEY
- Learn to Master HTML 5, scriptDemics, StarEdu Solutions Pvt Ltd.

1.16 QUESTIONS

- Which tag is used to create a list that displays items with bullets?
- How to make an image map?
- Explain HTML document structure.
- Explain <font> in html.
- Explain <sub> and <sup> tags.
- How to create HTML elements, attributes, tags?
- How to create HTML list, image, table, map?
- How to create Forms in HTML?
- How to working with Multimedia in HTML?
- Explain HTML elements for inserting Audio on a web page.
- Explain HTML elements for inserting Video on a web page.
- Explain any 2 form elements with an example.

❖❖❖❖

2

CSS

Unit Structure :

2.1 OBJECTIVES

After completing this chapter, you will be able to:

- Understand the function of CSS in the creation of user interfaces (for mobile) and websites.
- Selectors, CSS properties, CSS code structure, CSS declarations, and other fundamental CSS topics will be covered.

2.2 INTRODUCTION

Cascading Style Sheets (CSS) is an abbreviation for Cascading Style Sheets. Styles determine how HTML elements are displayed. It was created to allow developers to separate content from design in order to create a well-designed HTML page. External Style Sheets can help you save time and effort. CSS files contain external style sheets. In 1997, CSS was created as a tool for Web designers to determine the appearance and feel of their pages. If the designer adds additional instructions, the browser must determine which ones take precedence.

The official logo of the latest version CSS 3

CSS is typically used in combination with HTML to customize the appearance of web pages and user interfaces. CSS defines how HTML elements should appear on a screen, in print, or in other media. CSS helps you save time and effort. It has the ability to control the layout of several web pages at the same time. CSS files contain external stylesheets. CSS is a style sheet language for the web. It can be used with any XML-based markup language and is independent on HTML.

2.2.1 Use Of CSS

CSS is used to control the appearance and feel of web pages. CSS is used to control the colour of text, font style, different background pictures or colours, paragraph spacing, layout design, display changes for various devices, and a range of other effects. CSS is simple to learn and understand, yet it has a lot of power over how a web page looks. CSS is commonly used in conjunction with markup languages such as HTML. CSS, in simple terms, is a declaration of a style sheet that can be used repeatedly.

2.2.2 Features, Need and Advantages of CSS:

1. CSS allows for reusability.
2. CSS makes changing the style of a document relatively simple.
3. We may have a consistent layout and location of navigation across a site by using CSS.
4. It helps save time.
5. CSS-enabled pages load quicker than non-CSS-enabled pages. Etc.

2.2.3 Disadvantages of CSS

1. CSS doesn't always operate the same way in various browsers.
2. CSS debugging is extremely difficult.
3. Requires time while the webpage is being created.

2.3 UNDERSTANDING THE SYNTAX OF CSS

The selector identifies the HTML element that needs to be styled. A selector is a tool that allows you select or point to one or more specified

items on your page. One or more declarations are separated by semicolons in the declaration block. A colon separates the name of the CSS property and its value in each declaration. A semicolon always ends a CSS declaration, and declaration blocks are enclosed by curly braces.

Selector { property : value ; property2 : value2;}

For example, the following CSS style rule specifies that, any text in h1 elements should be centered and has a font color of blue.

h1 {text-align:center; color:blue;}

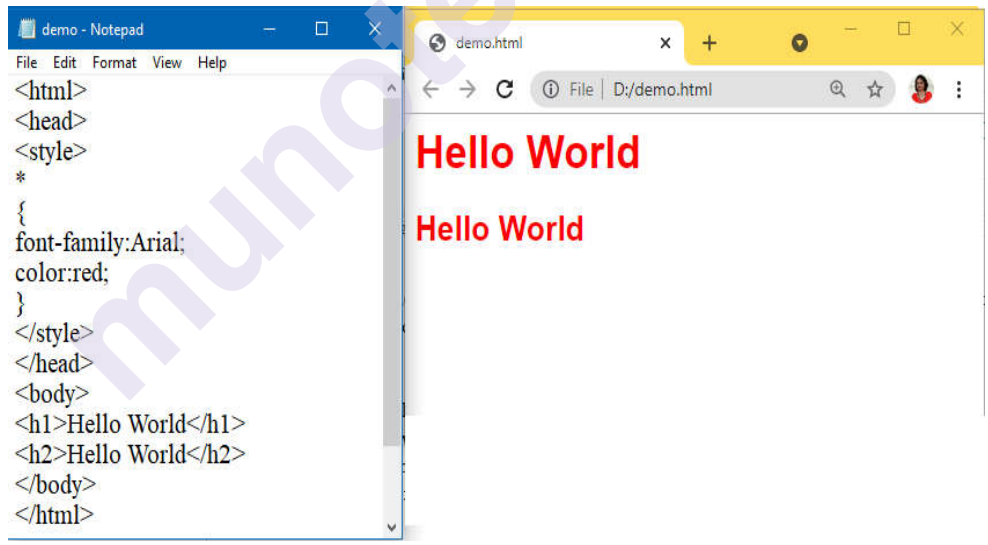
2.4 CSS SELECTORS

1. **Universal Selector:** Universal selector indicate by asterisk (*) sign applies to all elements in your page. It is used to set global setting like a font family.

Example:

```
*
{
font-family:Arial;
}
```

Example of Universal selector -

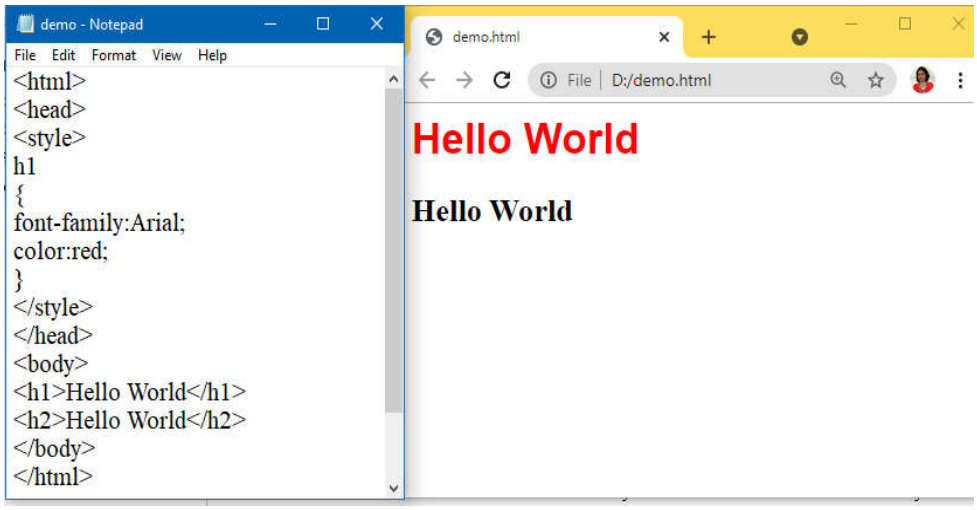


2. **Type Selector:** Type selector is the most basic type of selector which is used by the users. This selector helps the user to select any HTML element on a page that matches the selector, irrespective of the position of the elements in the document tree.

Example:

```
h1
{
   text-align: center;
   color: red;
}
```

Example of Type Selector:

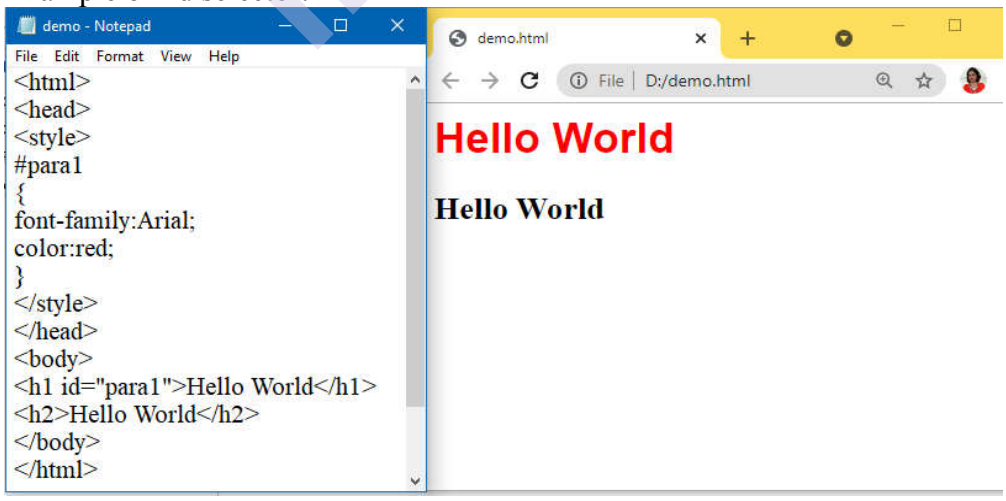


3. **ID Selector:** ID selectors are very similar to the class selectors. The only difference is that ID selector can only be applied once per page, while class selector can be applied as many times as required by the user. These types of selectors are well supported across standards – compliant browsers.

Example:

```
#para1
{
font-family:Arial;
color:red;
}
<h1 id="para1">Hello World</h1>
<h2>Hello World</h2>
```

Example of id selector:

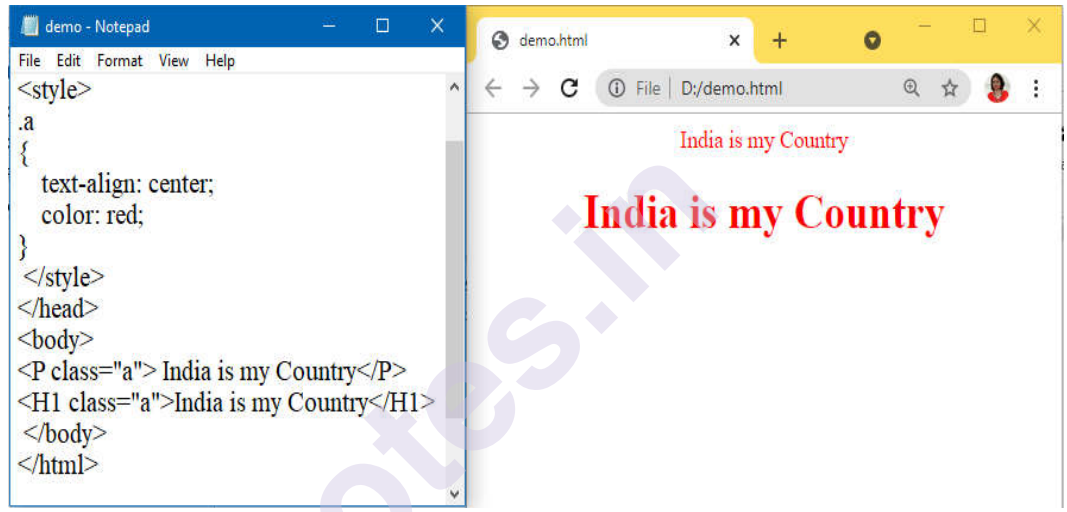


4. **Class Selector:** Class selector is used to style multiple elements. It is prefixed by dot (.). It is useful when you want to give the same type of formatting to related or unrelated HTML element.

Example:

```
.a
{
   text-align: center;
   color: red;
}

<P class=".a"> Hello World</P>
<H1 class=".a">How are you?</H1>
```

Example of class selector:

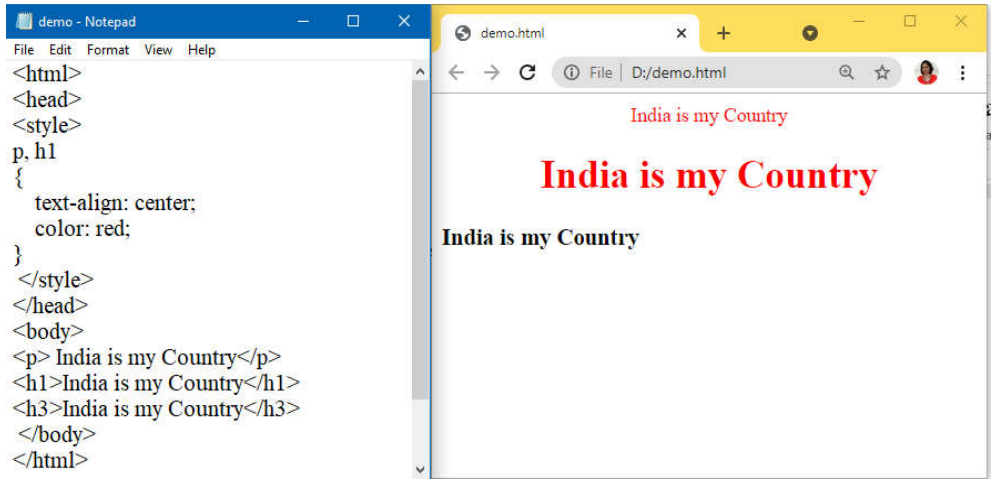


5. **Grouping & Combining Selector:** Grouping & Combining Selector used to group multiple selectors by separating them with a comma.

Example:

```
h1, h2, p
{
   text-align: center;
   color: red;
}

<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>
```

Example of Grouping & Combining Selector-



2.5 INSERTING CSS IN AN HTML DOCUMENT

You can change the design of a website entirely with an external style sheet.

2.5.1 Types of CSS

1. Inline style

An inline style rule is defined in an element's opening tag by using the style attribute. Use an inline style when we want to define properties for a single element in a Web page and we do not want to re-use that style.

The following example shows an inline style.

<p style="font-weight: bold; font-style: italic; color: #FF0000">

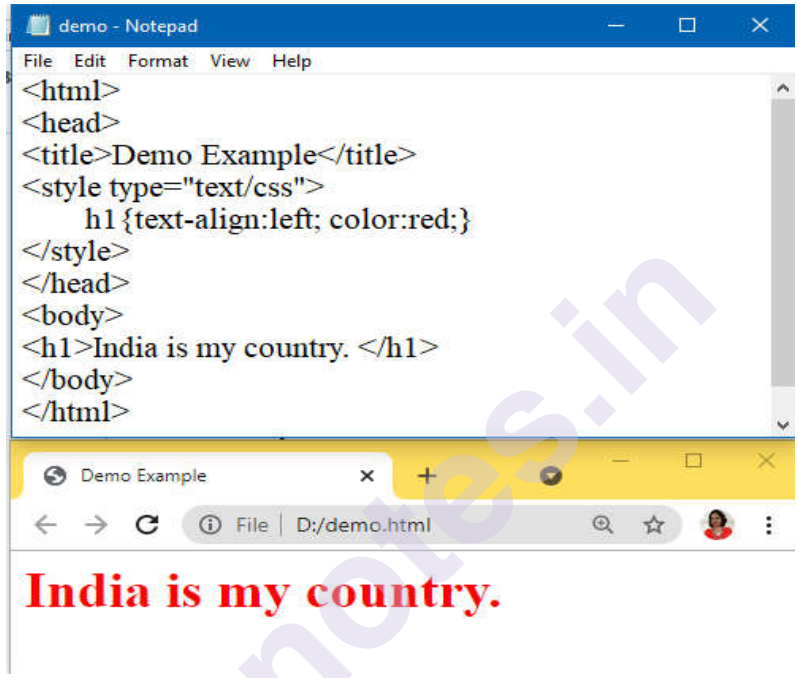
Example of inline style-

2. Internal style sheet

CSS style rules can be defined in a style element inside the head element of a Web page. In that case, the style rules apply only to elements in that page.

The following example shows how to define and apply a CSS style rule to all the h1 elements in a Web page.

Example of internal style sheet-



In this Web page, any text that appears between the <h1> and </h1> tags will be centered and blue. We do not have to reassign these style attributes for each h1 element in the document. If we want to change the color (or any property) of all text in h1 elements, we can edit one style rule.

3. External style sheet

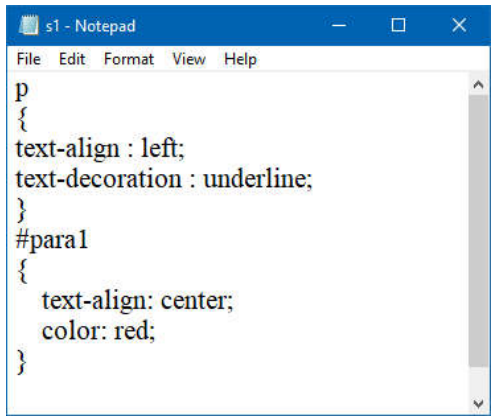
An external style sheet is a text file that has a .css file name extension and that contains only style rules. We can link a style sheet to a Web page by using a link element, as shown in the following example.

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

This link element applies the style rules in the external style sheet myStyles.css to the current page.

Example of external style sheet:

Create s1.css as follows:



Create demo.html and add s1.css in it with <link> tag as follows:

2.6 CSS PROPERTIES TO WORK WITH BACKGROUND OF A PAGE

The CSS background property is used to specify the element's background effects. The HTML elements are affected by five CSS background properties:

1) background-color
2) background-image
3) background-repeat
4) background-attachment
5) background-position

1) CSS background-color

The element's background colour is defined using the background-color property.

2) CSS background-image

The background-image attribute is used to make an image the element's backdrop.

3) CSS background-repeat

The background-image attribute is set to repeat the background picture horizontally and vertically by default. Some photos are only repeated vertically or horizontally.

If the image is simply repeated horizontally, the background appears better.

background-repeat: repeat-x; and background-repeat: repeat-y; are used in it.

4) CSS background-attachment

The background-attachment attribute controls whether the background image in the browser window is fixed or scrolls with the rest of the page. The image will not move while scrolling in the browser if the background image is set to fixed.

background: white url('bbb.gif');

background-repeat: no-repeat;

background-attachment: fixed;

5) CSS background-position

The background-position property is used to specify the background image's initial location. The background image is put in the top-left corner of the webpage by default.

The following positions can be set:

- center
- top
- bottom

- left
- right

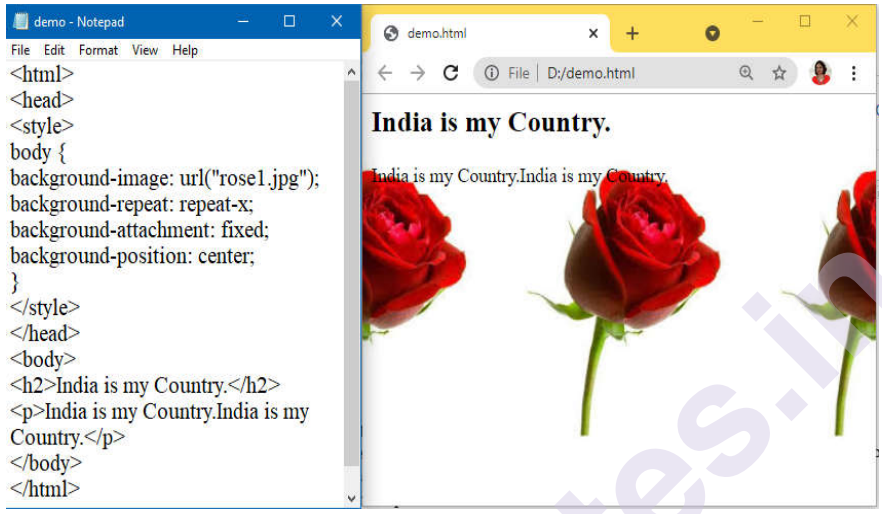
background: white url('rose1.jpg');

background-repeat: no-repeat;

background-attachment: fixed;

background-position: center;

Example of CSS properties to work with background of a Page-



2.7 CSS PROPERTIES TO WORK WITH FONTS AND TEXT STYLES

The font property in CSS is used to customise the appearance of text. You can modify the text size, colour, style, and more using the CSS font property. You've previously learned how to bold or underline text. You'll also learn how to use percentages to resize your font.

Here are some important font attributes:

1) **CSS Font color**: This property is used to change the color of the text.
2) **CSS Font family**: This property is used to change the face of the font.
3) **CSS Font size**: This property is used to increase or decrease the size of the font.
4) **CSS Font style**: This property is used to make the font bold, italic or oblique.
5) **CSS Font variant**: This property creates a small-caps effect.
6) **CSS Font weight**: This property is used to increase or decrease the boldness and lightness of the font.

1) CSS Font Color

CSS font colour is a separate attribute in CSS, despite the fact that it appears to be part of CSS fonts. It's used to adjust the text colour.

A colour can be defined in one of three ways:

- By a color name ex. red
- By hexadecimal value ex. 00FF11
- By RGB ex. rgb(0,100,255)

2) CSS Font Family

There are two types of fonts in the CSS font family:

- Generic family: It includes Serif, Sans-serif, and Monospace.
- Font family: It specifies the font family name like Arial, New Times Roman etc.

Serif: Serif fonts include small lines at the end of characters. Example of serif: Times new roman, Georgia etc.

Sans-serif: A sans-serif font doesn't include the small lines at the end of characters. Example of Sans-serif: Arial, Verdana etc.

3) CSS Font Size

CSS font size property is used to change the size of the font. These are the possible values that can be used to set the font size:

- xx-small: used to display the extremely small text size.
- x-small: used to display the extra small text size.
- Small: used to display small text size.
- Medium: used to display medium text size.
- Large: used to display large text size.
- x-large: used to display extra large text size.
- xx-large : used to display extremely large text size.
- Smaller: used to display comparatively smaller text size.
- Larger:used to display comparatively larger text size.
- size in pixels or %: used to set value in percentage or in pixels.

4) CSS Font Style

The CSS Font style parameter specifies the font you wish to use. It might be italic, oblique, or normal.

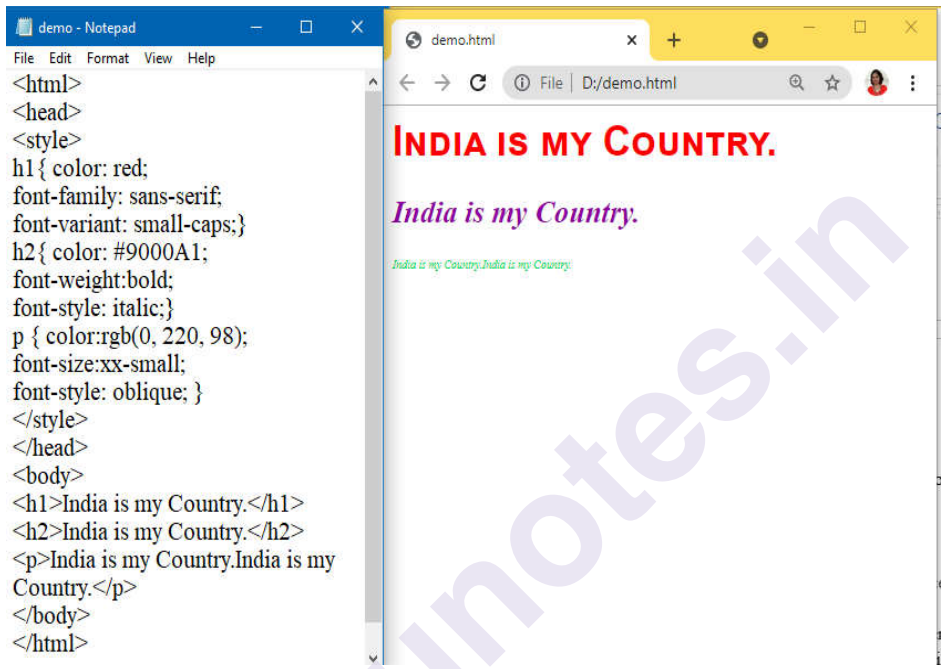
5) CSS Font Variant

CSS font variant property specifies how to set font variant of an element. It may be normal and small-caps.

6) CSS Font Weight

The font weight property in CSS specifies the font's weight and how bold it is. Normal, bold, bolder, lighter, or numeric font weights are all options (100, 200..... upto 900).

Example of CSS properties to work with Fonts and Text Styles-



2.8 CSS PROPERTIES FOR POSITIONING AN ELEMENT

The CSS position property is used to define an element's position. It's also useful for planned animation effects and to arrange an element behind another. The top, bottom, left, and right attributes can be used to position an element. These properties can only be used once the position property has been set. The computed position attribute of a position element can be relative, absolute, fixed, or sticky. Let's have a look at the CSS positioning that follows:

1) CSS Static Positioning

2) CSS Fixed Positioning

3) CSS Relative Positioning

4) CSS Absolute Positioning

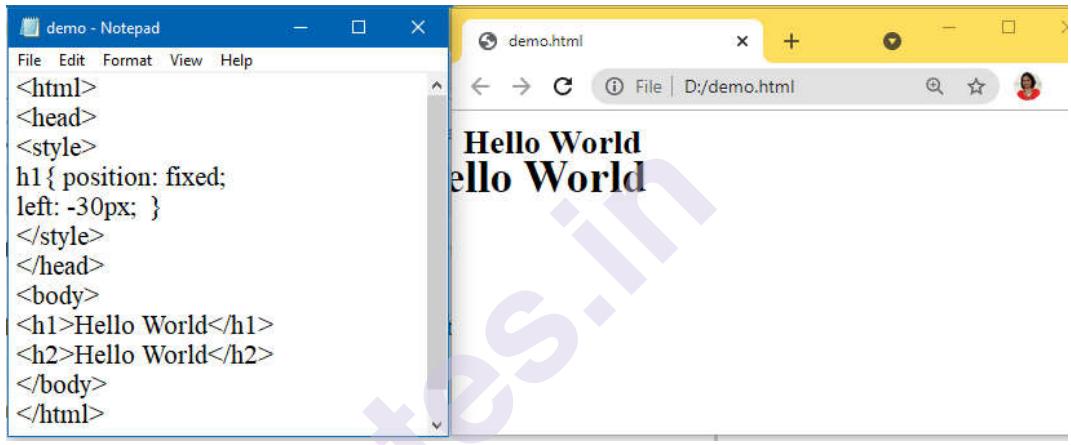
1) CSS Static Positioning

HTML components are placed in this position by default. It always places an element in accordance with the page's natural flow. The top, bottom, left, and right attributes have no effect on it.

2) CSS Fixed Positioning

The fixed position attribute helps in the positioning of text on the browser. This fixed test is positioned in relation to the browser window and will not move even if the window is scrolled.

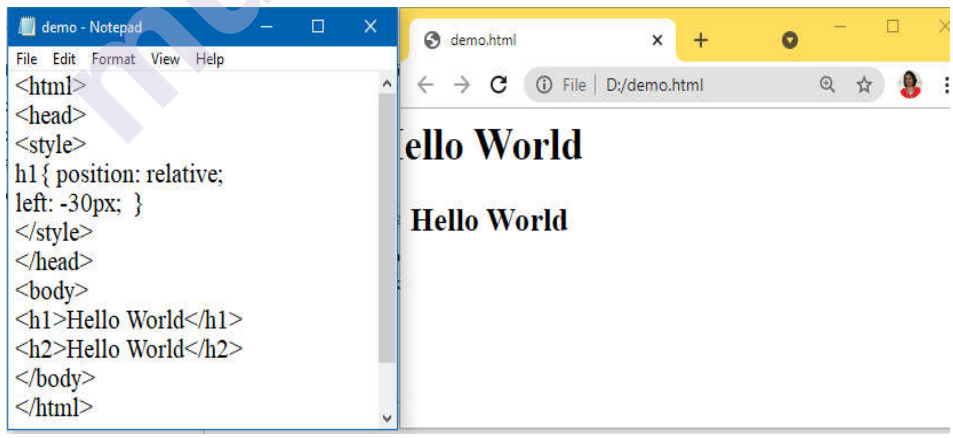
Example of CSS fixed position-



3) CSS Relative Positioning

The relative positioning property is used to set the element relative to its normal position.

Example of CSS Relative Positioning-

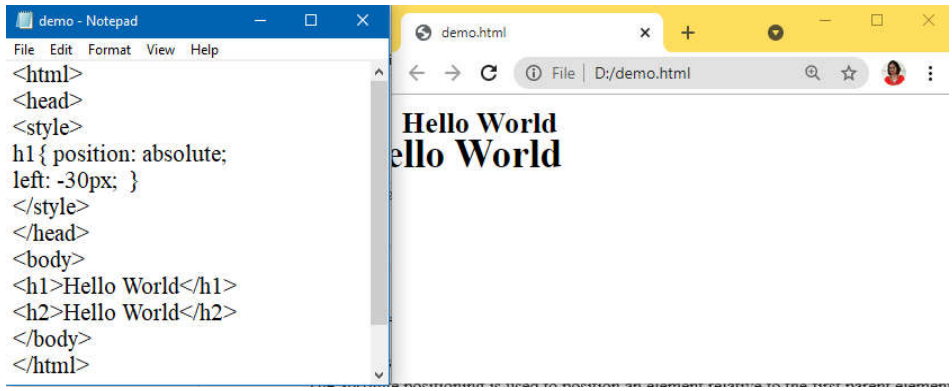


4) CSS Absolute Positioning

Absolute positioning is used to position an element in relation to the initial parent element with a non-static location. The containing block is HTML if no such element is detected.

You can position an element anywhere on a page with absolute positioning.

Example of CSS Absolute Positioning-



All CSS Position Properties

No.	property	description	values
1)	bottom	specify the bottom margin edge for a positioned box.	auto, length, %, inherit
2)	Clip	specify clip an absolutely positioned element.	shape, auto, inherit
3)	cursor	specify the type of cursors to be displayed.	url, auto, crosshair, default, pointer, move, e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize, text, wait, help
4)	Left	specify a left margin edge for a positioned box.	auto, length, %, inherit
5)	overflow	specify what happens if content overflow an element's box.	auto, hidden, scroll, visible, inherit
6)	position	specify the type of positioning for an element.	absolute, fixed, relative, static, inherit
7)	right	specify a right margin edge for a positioned box.	auto, length, %, inherit
8)	Top	specify a top margin edge for a positioned box.	auto, length, %, inherit
9)	z-index	specify set stack order of an element.	number, auto, inherit

2.9 SUMMARY

In this chapter we have covered Syntax of CSS, how to use CSS Selectors , how to insert CSS in an HTML Document , Use of CSS properties to work with background of a Page , Fonts , Text Styles and positioning an element with examples. CSS is essential knowledge for today's web developers and designers.

2.10 REFERENCE FOR FURTHER READING

- HTML 5 Black Book, Covers CSS 3, JavaScript, XML, XHTML, AJAX, PHP and j Query, 2ed, Dreamtech Press.
- HTML, XHTML, and CSS Bible Fifth Edition, Steven M. Schafer, WILEY
- Learn to Master HTML 5, script Demics, Star Edu Solutions Pvt Ltd.

2.11 QUESTIONS

1. What is CSS?
2. What is the full form of CSS?
3. Why was CSS developed?
4. What are the advantages of using CSS?
5. What are the disadvantages of using CSS?
6. What are the types of CSS?
7. How to Insert CSS in an HTML Document?
8. Write a note on CSS selectors.
9. Explain CSS properties to work with background of a Page.
10. Explain CSS properties to work with Fonts and Text Styles.
11. Explain CSS properties for positioning an element.

3

JAVASCRIPT

Unit Structure

3.1 WHAT IS JAVASCRIPT ?

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**. JavaScript is a lightweight, interpreted programming language.

- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform

Client-Side JavaScript

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

Advantages of JavaScript

The merits of using JavaScript are −

- **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features −

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multi-threading or multiprocessor capabilities.

3.2 USING JAVASCRIPT IN AN HTML DOCUMENT

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags.

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

<script ...>

JavaScript code

</script>

The script tag takes two important attributes −

- **Language** − This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **Type** − This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

  So your JavaScript segment will look like –

  <script language = "javascript" type = "text/javascript">
  JavaScript code
  </script>

Your First JavaScript Code

Let us take a sample example to print out "Hello World". We added an optional HTML comment that surrounds our JavaScript code. This is to save our code from a browser that does not support JavaScript. The comment ends with a "//-->". Here "//" signifies a comment in JavaScript, so we add that to prevent a browser from reading the end of the HTML comment as a piece of JavaScript code. Next, we call a function **document.write** which writes a string into our HTML document.

```
<html>
<body>
<scriptlanguage="javascript"type="text/javascript">
<!--
document.write("Hello World!")
//-->
</script>
</body>
</html>
```

This code will produce the following result −

Hello World!

Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

```
<scripttype="text/javascript">
<!--
var money;
var name;
//-->
</script>
```

You can also declare multiple variables with the same **var** keyword as follows −

```
<scripttype="text/javascript">
<!--
var money, name;
//-->
</script>
```

Note − Use the **var** keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

JavaScript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

- **Global Variables** − A global variable has global scope which means it can be defined anywhere in your JavaScript code.
- **Local Variables** − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.
- Within the body of a function, a local variable takes precedence over a global variable with the same name. If you declare a local variable or function parameter with the same name as a global

variable, you effectively hide the global variable. Take a look into the following example.

```
<html>
<bodyonload=checkscope();>
<scripttype="text/javascript">
<!--
varmyVar="global";// Declare a global variable
functioncheckscope(){
varmyVar="local";// Declare a local variable
document.write(myVar);
}
//-->
</script>
</body>
</html>
```

JavaScript Variable Names

While naming your variables in JavaScript, keep the following rules in mind.

- You should not use any of the JavaScript reserved keywords as a variable name. These keywords are mentioned in the next section. For example, **break** or **boolean** variable names are not valid.
- JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, **123test** is an invalid variable name but **_123test** is a valid one.
- JavaScript variable names are case-sensitive. For example, **Name** and **name** are two different variables.
- JavaScript Reserved Words
- A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

JavaScript Operator:

JavaScript includes operators same as other languages. An operator performs some operation on single or multiple operands (data value) and produces a result. For example, in 1 + 2, the + sign is an operator and 1 is left side operand and 2 is right side operand. The + operator performs the addition of two numeric values and returns a result.

Syntax:

<*Left operand*> operator <*right operand*>

<*Left operand*> operator

JavaScript includes following categories of operators.

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators

4. Assignment Operators
5. Conditional Operators
6. Ternary Operator

Arithmetic Operators

Arithmetic operators are used to perform mathematical operations between numeric operands.

Operator	Description
+	Adds two numeric operands.
-	Subtract right operand from left operand
*	Multiply two numeric operands.
/	Divide left operand by right operand.
%	Modulus operator. Returns remainder of two operands.
++	Increment operator. Increase operand value by one.
--	Decrement operator. Decrease value by one.

The following example demonstrates how arithmetic operators perform different tasks on operands.

Example: Arithmetic Operation

```
var x = 5, y = 10;

var z = x + y; //performs addition and returns 15

z = y - x; //performs subtraction and returns 5

z = x * y; //performs multiplication and returns 50

z = y / x; //performs division and returns 2

z = x % 2; //returns division remainder 1
```

The ++ and -- operators are unary operators. It works with either left or right operand only. When used with the left operand, e.g., x++, it will increase the value of x when the program control goes to the next statement. In the same way, when it is used with the right operand, e.g., ++x, it will increase the value of x there only. Therefore, x++ is called post-increment, and ++x is called pre-increment.

Example: Post and Pre Increment/Decrement

```
var x = 5;
x++; //post-increment, x will be 5 here and 6 in the next line
++x; //pre-increment, x will be 7 here
x--; //post-decrement, x will be 7 here and 6 in the next line
--x; //pre-decrement, x will be 5 here
```

3.3 STRING CONCATENATION

The + operator performs concatenation operation when one of the operands is of string type. The following example demonstrates string concatenation even if one of the operands is a string.

Example: + Operator with String

```
var a = 5, b = "Hello ", c = "World!", d = 10;
a + b; //returns "5Hello "
b + c; //returns "Hello World!"
a + d; //returns 15
b + true; //returns "Hello true"
c - b; //returns NaN; - operator can only used with numbers
```

Comparison Operators

JavaScript provides comparison operators that compare two operands and return a boolean value true or false.

Operators	Description
==	Compares the equality of two operands without considering type.
===	Compares equality of two operands with type.
!=	Compares inequality of two operands.
>	Returns a boolean value true if the left-side value is greater than the right-side value; otherwise, returns false.
<	Returns a boolean value true if the left-side value is less than the right-side value; otherwise, returns false.
>=	Returns a boolean value true if the left-side value is greater than or equal to the right-side value; otherwise, returns false.
<=	Returns a boolean value true if the left-side value is less than or equal to the right-side value; otherwise, returns false.

The following example demonstrates the comparison operators.

Example: JavaScript Comparison Operators

```
var a = 5, b = 10, c = "5";

var x = a;

a == c; // returns true

a === c; // returns false

a == x; // returns true

a != b; // returns true

a > b; // returns false

a < b; // returns true

a >= b; // returns false

a <= b; // returns true
```

Logical Operators

In JavaScript, the logical operators are used to combine two or more conditions. JavaScript provides the following logical operators.

Operator	Description
&&	&& is known as AND operator. It checks whether two operands are non-zero or not (0, false, undefined, null or "" are considered as zero). It returns 1 if they are non-zero; otherwise, returns 0.
||	|| is known as OR operator. It checks whether any one of the two operands is non-zero or not (0, false, undefined, null or "" is considered as zero). It returns 1 if any one of of them is non-zero; otherwise, returns 0.
!	! is known as NOT operator. It reverses the boolean result of the operand (or condition). !false returns true, and !true returns false.

Example: Logical Operators

```
var a = 5, b = 10;

(a != b) && (a < b); // returns true

(a > b) || (a == b); // returns false

(a < b) || (a == b); // returns true

!(a < b); // returns false

!(a > b); // returns true
```

Assignment Operators

JavaScript provides the assignment operators to assign values to variables with less key strokes.

Assignment operators	Description
=	Assigns right operand value to the left operand.
+=	Sums up left and right operand values and assigns the result to the left operand.
-=	Subtract right operand value from the left operand value and assigns the result to the left operand.
*=	Multiply left and right operand values and assigns the result to the left operand.
/=	Divide left operand value by right operand value and assign the result to the left operand.
%=	Get the modulus of left operand divide by right operand and assign resulted modulus to the left operand.

Example: Assignment operators

```
var x = 5, y = 10, z = 15;

x = y; //x would be 10

x += 1; //x would be 6

x -= 1; //x would be 4

x *= 5; //x would be 25

x /= 5; //x would be 1

x %= 2; //x would be 1
```

Ternary Operator

JavaScript provides a special operator called ternary operator :? that assigns a value to a variable based on some condition. This is the short form of the if else condition.

Syntax:

<condition> ? <value1> :<value2>;

The ternary operator starts with conditional expression followed by the ? operator. The second part (after ? and before :) will be executed if

the condition turns out to be true. Suppose, the condition returns false, then the third part (after :) will be executed.

Example: Ternary operator

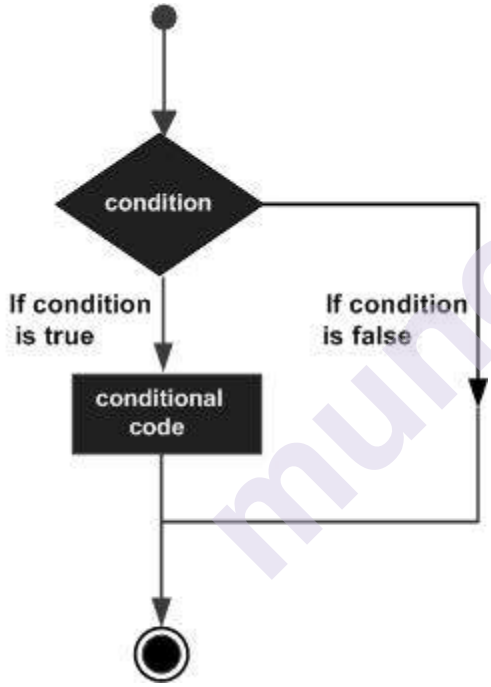
```
var a = 10, b = 5;

var c = a > b? a : b; // value of c would be 10

var d = a > b? b : a; // value of d would be 5
```

Control Flow Statement:

JavaScript supports conditional statements which are used to perform different actions based on different conditions. Here we will explain the **if..else** statement.

Flow Chart of if-else

The following flow chart shows how the if-else statement works.



JavaScript supports the following forms of **if..else** statement −

JavaScript supports the following forms of **if..else** statement −

- if statement
- if...else statement
- if...else if... statement.

if statement

The **if** statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax

The syntax for a basic if statement is as follows −

if (expression) {

Statement(s) to be executed if expression is true

}

Here a JavaScript expression is evaluated. If the resulting value is true, the given statement(s) are executed. If the expression is false, then no statement would be not executed. Most of the times, you will use comparison operators while making decisions.

Example

```
<html>
<body>
<scripttype="text/javascript">
<!--
var age =20;


if( age >18){
document.write("<b>Qualifies for driving</b>");
}
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

Output

Qualifies for driving

Set the variable to different value and then try...

if...else statement

The **'if...else'** statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

Syntax

if (expression) {

Statement(s) to be executed if expression is true

} else {

Statement(s) to be executed if expression is false

}

Here JavaScript expression is evaluated. If the resulting value is true, the given statement(s) in the 'if' block, are executed. If the expression is false, then the given statement(s) in the else block are executed.

Example

Try the following code to learn how to implement an if-else statement in JavaScript.

```
<html>
<body>
<scripttype="text/javascript">
<!--
var age =15;
if( age >18){
document.write("<b>Qualifies for driving</b>");
}else{
document.write("<b>Does not qualify for driving</b>");
}
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```


Output

Does not qualify for driving

Set the variable to different value and then try...

if...else if... statement

The **if...else if...** statement is an advanced form of **if…else** that allows JavaScript to make a correct decision out of several conditions.

Syntax

The syntax of an if-else-if statement is as follows −

```
if (expression 1) {
Statement(s) to be executed if expression 1 is true
} else if (expression 2) {
Statement(s) to be executed if expression 2 is true
} else if (expression 3) {
Statement(s) to be executed if expression 3 is true
} else {
Statement(s) to be executed if no expression is true
}
```

There is nothing special about this code. It is just a series of **if** statements, where each **if** is a part of the **else** clause of the previous statement. Statement(s) are executed based on the true condition, if none of the conditions is true, then the **else** block is executed.

Example

Try the following code to learn how to implement an if-else-if statement in JavaScript.

```
<html>
<body>
<scripttype="text/javascript">
<!--
var book ="maths";
if( book =="history"){
document.write("<b>History Book</b>");
}elseif( book =="maths"){
document.write("<b>Maths Book</b>");
}elseif( book =="economics"){
document.write("<b>Economics Book</b>");
}else{
document.write("<b>Unknown Book</b>");
```

```
}
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
<html>
```

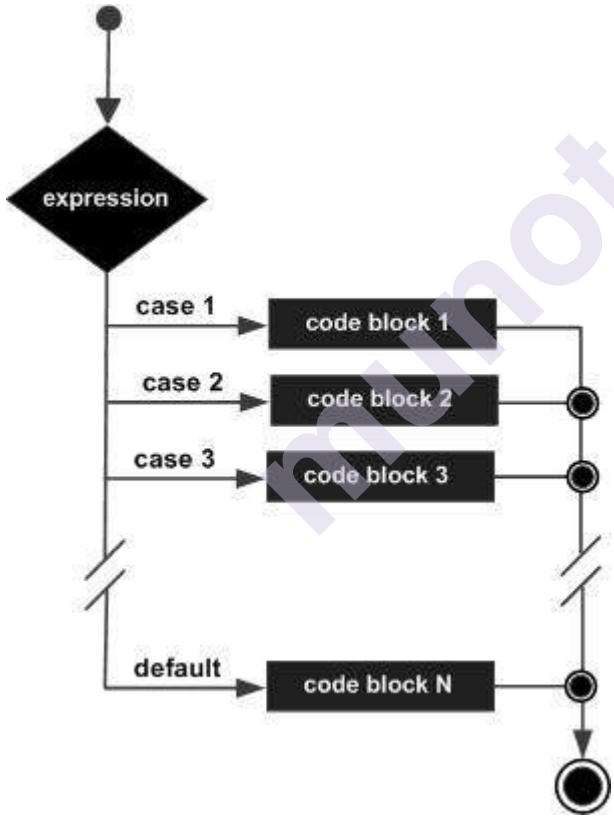
Output

3.4 MATHS BOOK

Set the variable to different value and then try.

Flow Chart

The following flow chart explains a switch-case statement works.



Syntax

The objective of a **switch** statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each **case** against the value of the expression until a match is found. If nothing matches, a **default** condition will be used.

```
switch (expression) {
case condition 1: statement(s)
break;
case condition 2: statement(s)
break;
...
case condition n: statement(s)
break;
default: statement(s)
}
```

The **break** statements indicate the end of a particular case. If they were omitted, the interpreter would continue executing each statement in each of the following cases.

We will explain **break** statement in **Loop Control** chapter.

Example

Try the following example to implement switch-case statement.

```
<html>
<body>
<scripttype="text/javascript">
<!--
var grade ='A';
document.write("Entering switch block<br />");
switch(grade){
case'A':document.write("Good job<br />");
break;
case'B':document.write("Pretty good<br />");
break;
case'C':document.write("Passed<br />");
break;
```

```
case'D':document.write("Not so good<br />");
break;
case'F':document.write("Failed<br />");
break;
default:document.write("Unknown grade<br />")
}
document.write("Exiting switch block");
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

Output

Entering switch block

Good job

Exiting switch block

Set the variable to different value and then try...

Break statements play a major role in switch-case statements. Try the following code that uses switch-case statement without any break statement.

```
<html>
<body>
<scripttype="text/javascript">
<!--
var grade ='A';
document.write("Entering switch block<br />");
switch(grade){
case'A':document.write("Good job<br />");
case'B':document.write("Pretty good<br />");
case'C':document.write("Passed<br />");
```

```
case'D':document.write("Not so good<br />");
case'F':document.write("Failed<br />");
default:document.write("Unknown grade<br />")
}
document.write("Exiting switch block");
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

Output

Entering switch block

Good job

Pretty good

Passed

Not so good

Failed

Unknown grade

Exiting switch block

Set the variable to different value and then try...

3.5 JAVASCRIPT LOOPING STATEMENTS:

Very often when you write code, you want the same block of code to run a number of times. You can use looping statements in you code to do this.

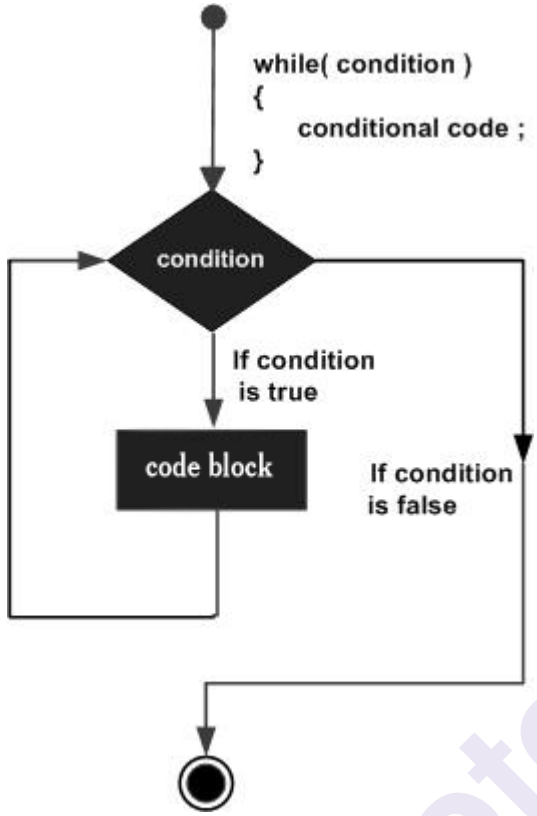
1. While statement
2. do…while statement
3. for statement

1. while statement:

   The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is true. Once the expression becomes **false,** the loop terminates.

Flow Chart

The flow chart of **while loop** looks as follows −



Syntax

The syntax of **while loop** in JavaScript is as follows −

```
while (expression) {
Statement(s) to be executed if expression is true
}
```

Example

```
<html>
<body>
<h2>JavaScript While Loop</h2>
<p id="demo"></p>
<script>
let text = "";
leti = 0;
while (i< 10) {
text += "<br>The number is " + i;
i++;
}
```

```
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

OUTPUT:

JavaScript While Loop

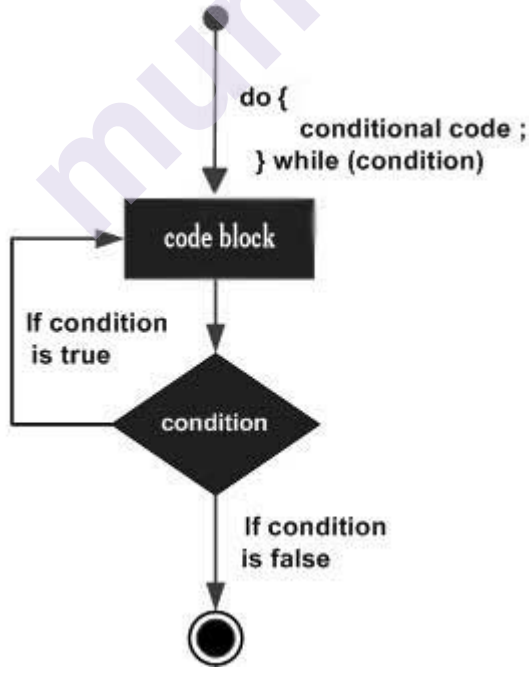
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9

The do...while Loop

The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is **false**.

Flow Chart

The flow chart of a **do-while** loop would be as follows −



Syntax

The syntax for **do-while** loop in JavaScript is as follows −

```
do {
Statement(s) to be executed;
} while (expression);
```

```
<html>
<body>
<h2>JavaScript Do While Loop</h2>
<p id="demo"></p>
<script>
let text = ""
leti = 0;
do {
text += "<br>The number is " + i;
i++;
}
while (i< 10);
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

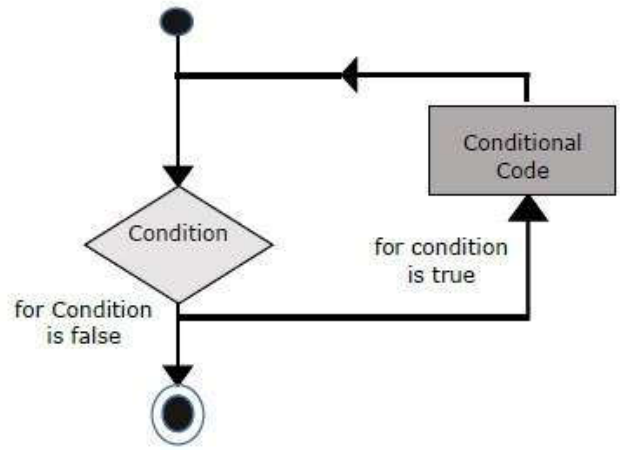
for loop:

The '**for**' loop is the most compact form of looping. It includes the following three important parts −

- The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The **iteration statement** where you can increase or decrease your counter.

Flow Chart

The flow chart of a **for** loop in JavaScript would be as follows −



Syntax

The syntax of **for** loop is JavaScript is as follows −

```
for (initialization; test condition; iteration statement)
{
Statement(s) to be executed if test condition is true
}
<html>
<head></head>
<title>for loop</title>
<body>
<script type=”text/javascript”>
vari;
for(i=1;i<=10;i++)
{
document.write(i);
}
</script>
</body>
</html>
```

Working with JavaScript jump statements:

The break statement "jumps out" of a loop.

The continue statement "jumps over" one iteration in the loop.

You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch() statement.

The break statement can also be used to jump out of a loop.

The **break statement** breaks the loop and continues executing the code after the loop (if any):

```
<html>

<body>

<p>A loop with a break.</p>

<p id="demo"></p>

<script>

var text = "";

vari;

for (i = 0; i< 10; i++)

 {

if (i === 3)

{

break;

}

text += "The number is " + i + "<br>";

}

document.getElementById("demo").innerHTML = text;

</script>


</body>

</html>
```

A loop with a break.

The number is 0
The number is 1
The number is 2

The Continue Statement

The **continue statement** breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of 3:

<html>

<body>

A loop which will skip the step where i = 3.

<script>

var text = "";

vari;

for (i = 0; i< 10; i++) {

if (i === 3) { continue; }

text += "The number is " + i + "
";

}

document.getElementById("demo").innerHTML = text;

</script>

</body>

</html>

A loop which will skip the step where i = 3.

The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9

javaScript Popup Boxes:

JavaScript Message Boxes: alert(), confirm(), prompt()

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed

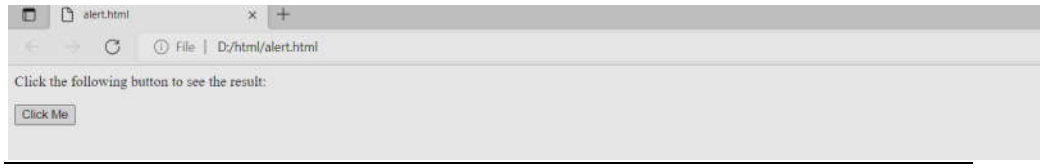
Syntax

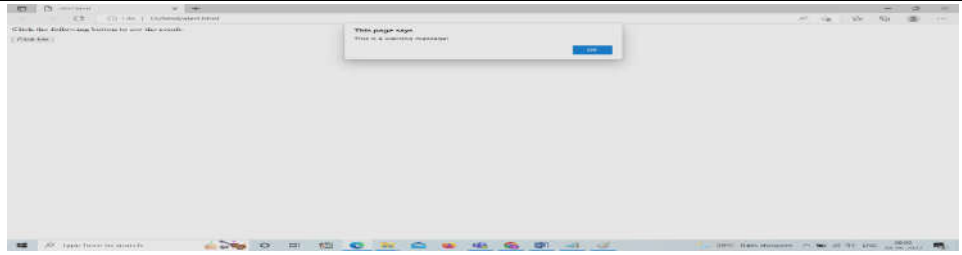
window.alert("*sometext*");

The window.alert() method can be written without the window prefix.

```
<html>
<head>
<scripttype="text/javascript">
<!--
functionWarn(){
alert("This is a warning message!");
document.write("This is a warning message!");
}
//-->
</script>
</head>

<body>
<p>Click the following button to see the result: </p>
<form>
<inputtype="button"value="Click Me"onclick="Warn();"/>
</form>
</body>
</html>
```

Output





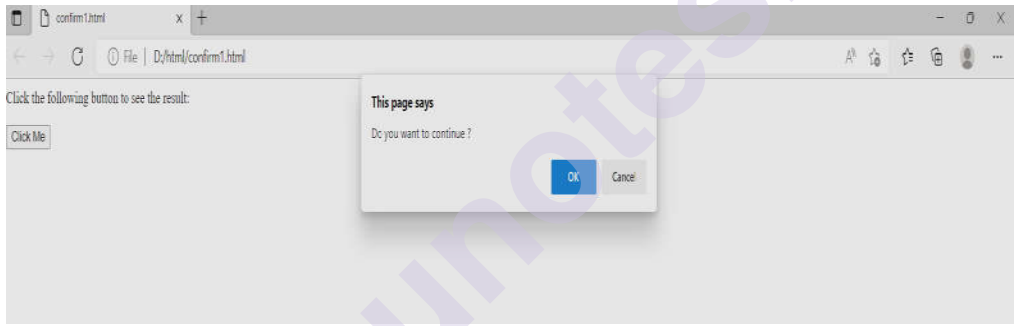
3.6 CONFIRMATION DIALOG BOX

A confirmation dialog box is mostly used to take user's consent on any option. It displays a dialog box with two buttons: **OK** and **Cancel**.

If the user clicks on the OK button, the window method **confirm()** will return true. If the user clicks on the Cancel button, then **confirm()** returns false. You can use a confirmation dialog box as follows.

```
<html>
<head>
<scripttype="text/javascript">
<!--
functiongetConfirmation(){
varretVal= confirm("Do you want to continue ?");
if(retVal==true){
document.write("User wants to continue!");
returntrue;
}else{
document.write("User does not want to continue!");
returnfalse;
}
}
//-->
</script>
</head>

<body>
```

```
<p>Click the following button to see the result: </p>
<form>
<inputtype="button"value="Click Me"onclick="getConfirmation();"/>
</form>
</body>
</html>
```



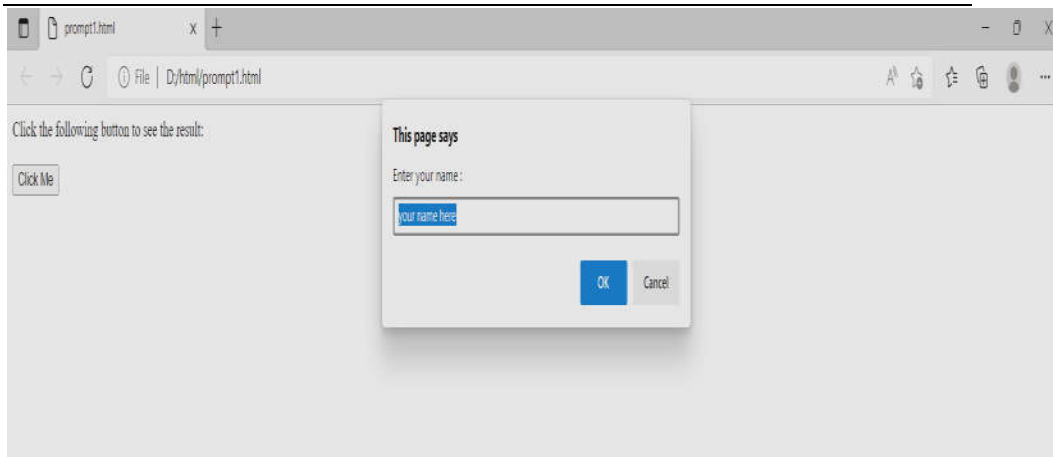
Prompt Dialog Box

The prompt dialog box is very useful when you want to pop-up a text box to get user input. Thus, it enables you to interact with the user. The user needs to fill in the field and then click OK.

This dialog box is displayed using a method called **prompt()** which takes two parameters: (i) a label which you want to display in the text box and (ii) a default string to display in the text box.

This dialog box has two buttons: **OK** and **Cancel**. If the user clicks the OK button, the window method **prompt()** will return the entered value from the text box. If the user clicks the Cancel button, the window method **prompt()** returns **null**.

```
<html>
<head>
<scripttype="text/javascript">
<!--
functiongetValue(){
varretVal= prompt("Enter your name : ","your name here");
document.write("You have entered : "+retVal);
}
//-->
</script>
</head>

<body>
<p>Click the following button to see the result: </p>
<form>
<inputtype="button"value="Click Me"onclick="getValue();"/>
</form>
</body>
</html>
```



JavaScript Functions

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don't need to write many lines of code each time to perform a common task

Defining and invoking a JavaScript Function

JavaScript Function Syntax

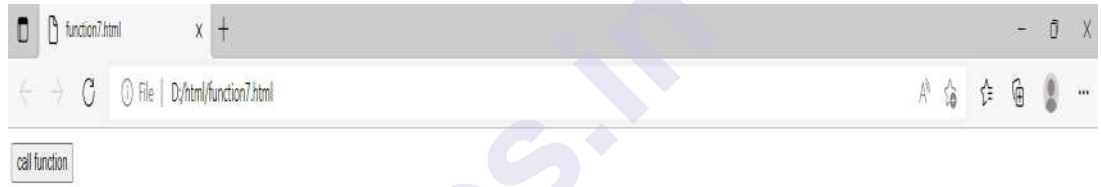
The syntax of declaring function is given below.

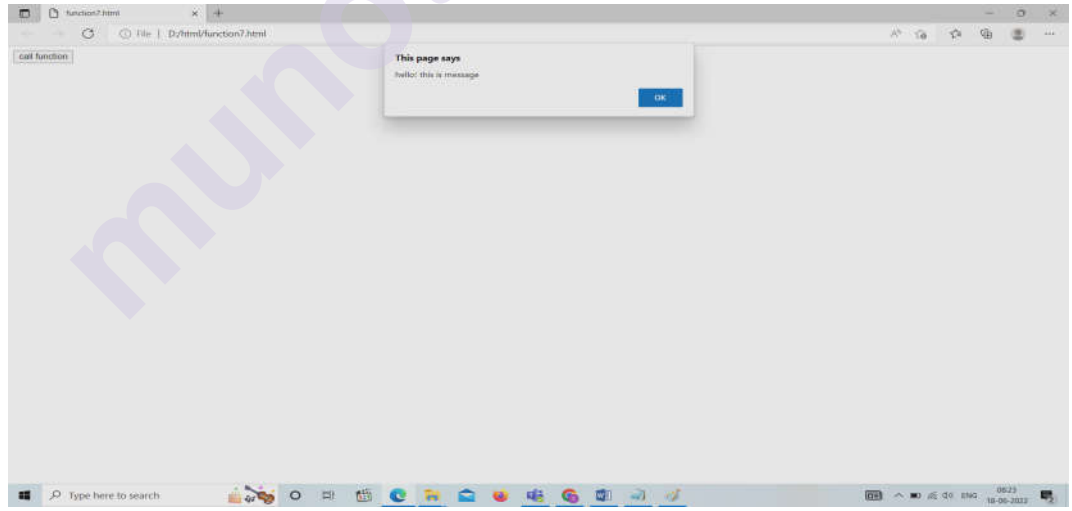
```
function functionName([arg1, arg2, ...argN])
{
 //code to be executed
}
```

JavaScript Functions can have 0 or more arguments.

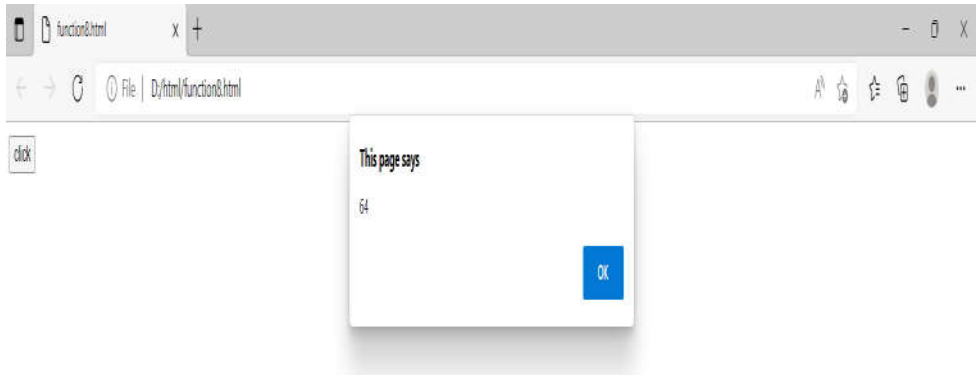
```
<html>
```

```
<body>
<script>
functionmsg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
</body>
</html>
```





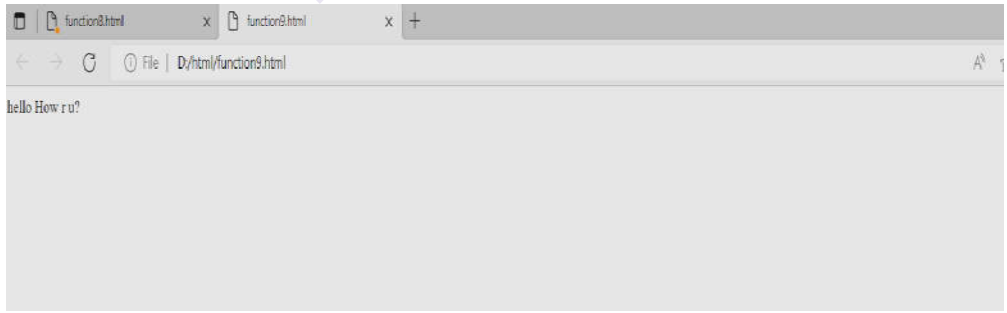
JavaScript Function Arguments



Function with Return Value

We can call function that returns a value and use it in our program. Let's see the example of function that returns value.

```
<html>
<body>
<script>
functiongetInfo(){
return "hello How r u?";
}  </script>
<script>
document.write(getInfo());
</script>
</body>
</html>
```



JavaScript timer

In JavaScript, a timer is created to execute a task or any function at a particular time. Basically, the timer is used to delay the execution of the program or to execute the JavaScript code in a regular time interval. With the help of timer, we can delay the execution of the code. So, the code

does not complete it's execution at the same time when an event triggers or page loads.

The best example of the timer is advertisement banners on websites, which change after every 2-3 seconds. These advertising banners are changed at a regular interval on the websites like Amazon. We set a time interval to change them. In this chapter, we will show you how to create a timer.

JavaScript

offers two timer functions **setInterval()** and **setTimeout()**, which helps to delay in execution of code and also allows to perform one or more operations repeatedly. We will discuss both the timer functions in detail as well as their examples.

setTimeout()

The setTimeout() function helps the users to delay the execution of code. The setTimeout() method accepts two parameters in which one is a user-defined function, and another is the time parameter to delay the execution. The time parameter holds the time in milliseconds (1 second = 1000 milliseconds), which is optional to pass.

The basic syntax of setTimeout() is:

1. setTimeout(function, milliseconds)

We will use the setTimeout() function to delay the printing of message for 3 seconds. The message will display on the web after 3 seconds of code execution rather than immediately. Now, let's look at the code below to see how it works:

3.7 EXECUTION OF CODE AFTER A DELAY

```
<html>

<body>

<script>

Function delay Function() {

   //display the message on web after 3 seconds on calling delay Function

document.write('<h3> Welcome to JavaScript <h3>');

}

</script>

<h4> Example of delay the execution of function <h4>
```

<!?button for calling of user-defined delay Function having 3 seconds of delay -->

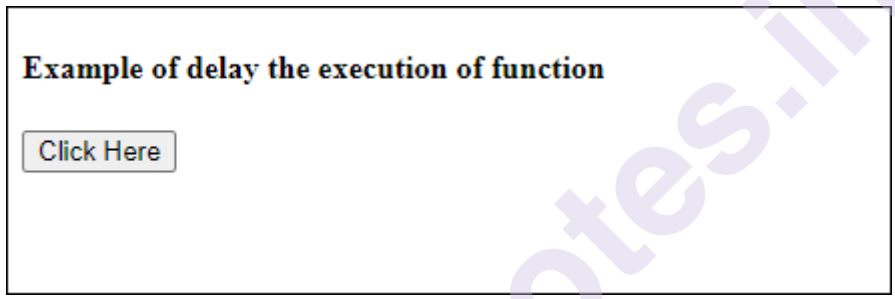
<button onclick = "set Timeout (delay Function, 3000)"> Click Here </button>

</body>

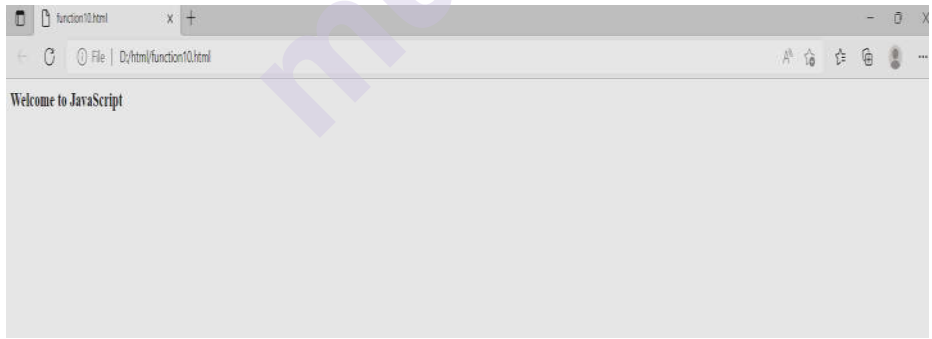
</html>

Output

The above code will execute in two sections. Firstly, the HTML part of the code will execute, where by clicking on **Click Here** button the remaining JavaScript code will execute after 3 seconds. See the output below:



On clicking the **Click Here** button, the remaining code will execute after 3 seconds. A message **Welcome to JavaScript** will display on the web after 3 seconds (3000 milliseconds).



setInterval()

The setInterval method is a bit similar to the setTimeout() function. It executes the specified function repeatedly after a time interval. Or we can simply say that a function is executed repeatedly after a specific amount of time provided by the user in this function. **For example -** Display updated time in every five seconds.

The basic syntax of setInterval() is:

setInterval(function, milliseconds)

Similar to setTimeout() method, it also accepts two parameters in which one is a user-defined function, and another is a time-interval parameter to wait before executing the function. The time-interval parameter holds the amount of time in milliseconds (1 second = 1000 milliseconds), which is optional to pass. Now, see the code below how this function works:

Execution of code at a regular interval

```
<html>
<body>
<script>
function waitAndshow() {
      //define a date and time variable
      var systemdate = new Date();
           //display the updated time after every 4 seconds
     document.getElementById("clock").innerHTML = systemdate.toLocaleTimeString();
}
      //define time interval and call user-defined waitAndshow function
    setInterval(waitAndshow, 4000);
</script>
 <h3> Updated time will show in every 4 seconds </h3>
<h3> The current time on your computer is: <br>
<span id="clock"></span> </h3>
</body>
</html>
```

Cancel or Stop the timer

JavaScript offers two functions **clear Timeout()** and **clear Interval()** to cancel or stop the timer and halt the execution of code. The set Timeout() and set Interval() both return a unique IDs. These IDs are used by the clear Timeout() and clear Interval() to clear the timer and stop the code execution beforehand. They both take only one parameter, i.e., ID.

Example

In this example, we will use clear Timeout() to clear the timer set by with set Timeout() function. Look at the example how clear Interval() work with set Interval().

Disable the regular interval

```
<html>
<body>
<script>
function waitAndshow() {
  //define a date and time variable
  var systemdate = new Date();
        //display the updated time after every 4 seconds
  document.getElementById("clock").innerHTML = systemdate.toLocaleTimeString();
}
   //function to disable the time interval
function stopClock() {
    clearInterval(intervalID);
}
  //define time interval and call user-defined waitAndshow function
var intervalID = setInterval(waitAndshow, 3000);
</script>
   <p>Current system time: <span id="clock"> </span> </p>
  <!-- button to stop showing time in a regular interval -->
<button onclick = "stopClock();" > Stop Clock </button>
</body>
</html>
```

3.8 JAVASCRIPT OBJECTS

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Creating Objects in JavaScript

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

1)JavaScript Object by object literal

The syntax of creating object using object literal is given below:

object={property1:value1,property2:value2.....propertyN:valueN}

As you can see, property and value is separated by : (colon).

Let's see the simple example of creating object in JavaScript.

```
<html>
<body>
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
</body>
```

```
</html>
```

Output:

102 Shyam Kumar 40000

2) By creating instance of Object

The syntax of creating object directly is given below:

```
var objectname=new Object();
<html>
<body>
<script>
varemp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
</body>
</html>
```

101 Ravi Malik 50000

3) By using an Object constructor

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

```
<html>
<body>
<script>
functionemp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);
```

```
document.write(e.id+" "+e.name+" "+e.salary);
</script>
</body>
</html>
```

OUTPUT:

103 Vimal Jaiswal 30000

JavaScript String Methods:

String Methods

Method	Description
charAt(position)	Returns the character at the specified position (in Number).
charCodeAt(position)	Returns a number indicating the Unicode value of the character at the given position (in Number).
concat([string,,])	Joins specified string literal values (specify multiple strings separated by comma) and returns a new string.
indexOf(SearchString, Position)	Returns the index of first occurrence of specified String starting from specified number index. Returns -1 if not found.
lastIndexOf(SearchString, Position)	Returns the last occurrence index of specified SearchString, starting from specified position. Returns -1 if not found.
localeCompare(string,position)	Compares two strings in the current locale.
match(RegExp)	Search a string for a match using specified regular expression. Returns a matching array.
replace(searchValue, replaceValue)	Search specified string value and replace with specified replace Value string and return new string. Regular expression can also be used as searchValue.
search(RegExp)	Search for a match based on specified regular expression.
slice(startNumber, endNumber)	Extracts a section of a string based on specified starting and ending index and returns a new string.
split(separatorString,	Splits a String into an array of strings by

Method	Description
limitNumber)	separating the string into substrings based on specified separator. Regular expression can also be used as separator.
substr(start, length)	Returns the characters in a string from specified starting position through the specified number of characters (length).
substring(start, end)	Returns the characters in a string between start and end indexes.
toLocaleLowerCase()	Converts a string to lower case according to current locale.
toLocaleUpperCase()	Converts a sting to upper case according to current locale.
toLowerCase()	Returns lower case string value.
toString()	Returns the value of String object.
toUpperCase()	Returns upper case string value.
valueOf()	Returns the primitive value of the specified string object.

RegExp Object

A regular expression is an object that describes a pattern of characters.

The JavaScript **RegExp** class represents regular expressions, and both String and **RegExp** define methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.

Syntax

A regular expression could be defined with the **RegExp ()** constructor, as follows −

var pattern = new RegExp(pattern, attributes);

or simply

var pattern = /pattern/attributes;

Here is the description of the parameters −

- **pattern** − A string that specifies the pattern of the regular expression or another regular expression.
- **attributes** − An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multi-line matches, respectively.

Brackets

Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

Sr.No.	Expression & Description
1	**[...]** Any one character between the brackets.
2	**[^...]** Any one character not between the brackets.
3	**[0-9]** It matches any decimal digit from 0 through 9.
4	**[a-z]** It matches any character from lowercase **a** through lowercase **z**.
5	**[A-Z]** It matches any character from uppercase **A** through uppercase **Z**.
6	**[a-Z]** It matches any character from lowercase **a** through uppercase **Z**.

The ranges shown above are general; you could also use the range [0-3] to match any decimal digit ranging from 0 through 3, or the range [b-v] to match any lowercase character ranging from **b** through **v**.

Quantifiers

The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character has a specific connotation. The +, *, ?, and $ flags all follow a character sequence.

Sr.No.	Expression & Description
1	**p+** It matches any string containing one or more p's.
2	**p*** It matches any string containing zero or more p's.
3	**p?** It matches any string containing at most one p.
4	**p{N}** It matches any string containing a sequence of **N** p's
5	**p{2,3}** It matches any string containing a sequence of two or three p's.
6	**p{2, }** It matches any string containing a sequence of at least two p's.
7	**p$** It matches any string with p at the end of it.
8	**^p** It matches any string with p at the beginning of it.

Examples

Following examples explain more about matching characters.

Sr.No.	Expression & Description
1	**[^a-zA-Z]** It matches any string not containing any of the characters ranging from **a** through **z** and **A** through Z.
2	**p.p** It matches any string containing **p,** followed by any character, in turn followed by another **p**.

3	**^.{2}$** It matches any string containing exactly two characters.
4	**(.*)** It matches any string enclosed within and .
5	**p(hp)*** It matches any string containing a **p** followed by zero or more instances of the sequence **hp**.

Literal characters

Sr.No.	Character & Description
1	**Alphanumeric** Itself
2	**\0** The NUL character (\u0000)
3	**\t** Tab (\u0009
4	**\n** Newline (\u000A)
5	**\v** Vertical tab (\u000B)
6	**\f** Form feed (\u000C)
7	**\r** Carriage return (\u000D)
8	**\xnn** The Latin character specified by the hexadecimal number nn; for example, \x0A is the same as \n
9	**\uxxxx** The Unicode character specified by the hexadecimal number xxxx; for example, \u0009 is the same as \t
10	**\cX** The control character ^X; for example, \cJ is equivalent to the newline character \n

Met characters

A metacharacter is simply an alphabetical character preceded by a backslash that acts to give the combination a special meaning.

For instance, you can search for a large sum of money using the '\d' metacharacter: **/([\d]+)000/**, Here **\d** will search for any string of numerical character.

The following table lists a set of metacharacters which can be used in PERL Style Regular Expressions.

Sr.No.	Character & Description
1	**.** a single character
2	**\s** a whitespace character (space, tab, newline)
3	**\S** non-whitespace character
4	**\d** a digit (0-9)
5	**\D** a non-digit
6	**\w** a word character (a-z, A-Z, 0-9, _)
7	**\W** a non-word character
8	**[\b]** a literal backspace (special case).
9	**[aeiou]** matches a single character in the given set
10	**[^aeiou]** matches a single character outside the given set
11	**(foo|bar|baz)** matches any of the alternatives specified

Modifiers

Several modifiers are available that can simplify the way you work with **regexps,** like case sensitivity, searching in multiple lines, etc.

Sr.No.	Modifier & Description
1	**i** Perform case-insensitive matching.
2	**m** Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of a string boundary
3	**g** Performs a global matchthat is, find all matches rather than stopping after the first match.

RegExp Properties

Here is a list of the properties associated with RegExp and their description.

Sr.No.	Property & Description
1	constructor Specifies the function that creates an object's prototype.
2	global Specifies if the "g" modifier is set.
3	ignoreCase Specifies if the "i" modifier is set.
4	lastIndex The index at which to start the next match.
5	multiline Specifies if the "m" modifier is set.
6	source The text of the pattern.

In the following sections, we will have a few examples to demonstrate the usage of RegExp properties.

RegExp Methods

Here is a list of the methods associated with RegExp along with their description.

Sr.No.	Method & Description
1	exec() Executes a search for a match in its string parameter.
2	test() Tests for a match in its string parameter.
3	toSource() Returns an object literal representing the specified object; you can use this value to create a new object.
4	toString() Returns a string representing the specified object.

In the following sections, we will have a few examples to demonstrate the usage of RegExp methods.

The **math** object provides you properties and methods for mathematical constants and functions. Unlike other global objects, **Math** is not a constructor. All the properties and methods of **Math** are static and can be called by using Math as an object without creating it.

Thus, you refer to the constant **pi** as **Math.PI** and you call the *sine* function as **Math.sin(x)**, where x is the method's argument.

Syntax

The syntax to call the properties and methods of Math are as follows

```
varpi_val = Math.PI;

varsine_val = Math.sin(30);
```

Math Properties

Here is a list of all the properties of Math and their description.

Sr.No.	Property & Description
1	E \ Euler's constant and the base of natural logarithms, approximately 2.718.
2	LN2 Natural logarithm of 2, approximately 0.693.
3	LN10 Natural logarithm of 10, approximately 2.302.
4	LOG2E Base 2 logarithm of E, approximately 1.442.
5	LOG10E Base 10 logarithm of E, approximately 0.434.
6	PI Ratio of the circumference of a circle to its diameter, approximately 3.14159.
7	SQRT1_2 Square root of 1/2; equivalently, 1 over the square root of 2, approximately 0.707.
8	SQRT2 Square root of 2, approximately 1.414.

In the following sections, we will have a few examples to demonstrate the usage of Math properties.

Math Methods

Here is a list of the methods associated with Math object and their description

Sr.No.	Method & Description
1	abs() Returns the absolute value of a number.
2	acos() Returns the arccosine (in radians) of a number.
3	asin() Returns the arcsine (in radians) of a number.
4	atan() Returns the arctangent (in radians) of a number.

5	atan2() Returns the arctangent of the quotient of its arguments.
6	ceil() Returns the smallest integer greater than or equal to a number.
7	cos() Returns the cosine of a number.
8	exp() Returns E^N, where N is the argument, and E is Euler's constant, the base of the natural logarithm.
9	floor() Returns the largest integer less than or equal to a number.
10	log() Returns the natural logarithm (base E) of a number.
11	max() Returns the largest of zero or more numbers.
12	min() Returns the smallest of zero or more numbers.
13	pow() Returns base to the exponent power, that is, base exponent.
14	random() Returns a pseudo-random number between 0 and 1.
15	round() Returns the value of a number rounded to the nearest integer.
16	sin() Returns the sine of a number.
17	sqrt() Returns the square root of a number.
18	tan() Returns the tangent of a number.
19	toSource() Returns the string "Math".

The **math** object provides you properties and methods for mathematical constants and functions. Unlike other global objects, **Math** is not a

constructor. All the properties and methods of **Math** are static and can be called by using Math as an object without creating it.

Thus, you refer to the constant **pi** as **Math.PI** and you call the *sine* function as **Math.sin(x)**, where x is the method's argument.

Syntax

The syntax to call the properties and methods of Math are as follows

varpi_val = Math.PI;

varsine_val = Math.sin(30);

Math Properties

Here is a list of all the properties of Math and their description.

Sr.No.	Property & Description
1	E \ Euler's constant and the base of natural logarithms, approximately 2.718.
2	LN2 Natural logarithm of 2, approximately 0.693.
3	LN10 Natural logarithm of 10, approximately 2.302.
4	LOG2E Base 2 logarithm of E, approximately 1.442.
5	LOG10E Base 10 logarithm of E, approximately 0.434.
6	PI Ratio of the circumference of a circle to its diameter, approximately 3.14159.
7	SQRT1_2 Square root of 1/2; equivalently, 1 over the square root of 2, approximately 0.707.
8	SQRT2 Square root of 2, approximately 1.414.

JavaScript Date Object

Methods	Description
getDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time.
getDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.
getHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of local time.
getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.
getUTCDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of universal time.
getUTCDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of universal time.
getUTCFullYears()	It returns the integer value that represents the year on the basis of universal time.
getUTCHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of universal time.
getUTCMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of universal time.
getUTCMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of universal time.

getUTCSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of universal time.
setDate()	It sets the day value for the specified date on the basis of local time.
setDay()	It sets the particular day of the week on the basis of local time.
setFullYears()	It sets the year value for the specified date on the basis of local time.
setHours()	It sets the hour value for the specified date on the basis of local time.
setMilliseconds()	It sets the millisecond value for the specified date on the basis of local time.
setMinutes()	It sets the minute value for the specified date on the basis of local time.
setMonth()	It sets the month value for the specified date on the basis of local time.
setSeconds()	It sets the second value for the specified date on the basis of local time.
setUTCDate()	It sets the day value for the specified date on the basis of universal time.
setUTCDay()	It sets the particular day of the week on the basis of universal time.
setUTCFullYears()	It sets the year value for the specified date on the basis of universal time.
setUTCHours()	It sets the hour value for the specified date on the basis of universal time.
setUTCMilliseconds()	It sets the millisecond value for the specified date on the basis of universal time.
setUTCMinutes()	It sets the minute value for the specified date on the basis of universal time.
setUTCMonth()	It sets the month value for the specified date on the basis of universal time.
setUTCSeconds()	It sets the second value for the specified date on the basis of universal time.

toDateString()	It returns the date portion of a Date object.
toISOString()	It returns the date in the form ISO format string.
toJSON()	It returns a string representing the Date object. It also serializes the Date object during JSON serialization.
toString()	It returns the date in the form of string.
toTimeString()	It returns the time portion of a Date object.
toUTCString()	It converts the specified date in the form of string using UTC time zone.
valueOf()	It returns the primitive value of a Date object.

3.9 BROWSER OBJECT MODEL

1. Browser Object Model (BOM)

The **Browser Object Model** (BOM) is used to interact with the browser.

The default object of browser is window means you can call all the functions of window by specifying window or directly. For example:

```
1.  window.alert("hello javatpoint");
```

is same as:

```
1. alert("hello javatpoint");
```

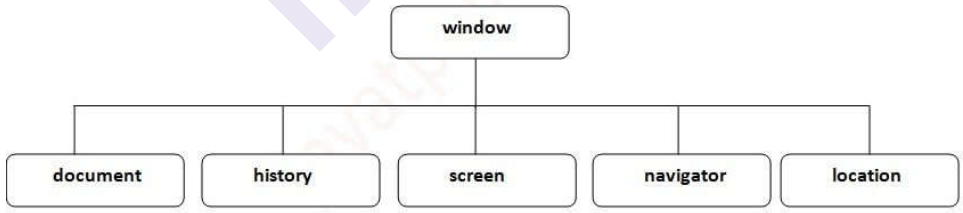


Fig: JavaScript's Window Object

Window Object

The **window object** represents a window in browser. An object of window is created automatically by the browser.

Window is the object of browser, **it is not the object of javascript**. The javascript objects are string, array, date etc.

Methods of window object

The important methods of window object are as follows:

Method	Description
alert()	displays the alert box containing message with ok button.
confirm()	displays the confirm dialog box containing message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.
close()	closes the current window.
setTimeout()	performs action after specified time like calling function, evaluating expressions etc.

Navigator Object Properties

The properties of the navigator object are the variables created inside the Navigator Object.

We can access Navigator Object Property as: **navigator.propertyname** where **propertyname** is the name of property.

Properties	Description
appcodename	specifies the code name of the browser (*experimental property - can return incorrect value*)
appname	specifies the name of the browser (*experimental property - can return incorrect value*)
appversion	specifies the version of browser being used (*experimental property - can return incorrect value*)
cookieEnabled	specifies whether cookies are enabled or not in the browser
platform	contains a string indicating the machine type for which the browser was compiled.

Properties	Description
useragent	contains a string representing the value of **user-agent header** sent by the client to server in HTTP protocol
geolocation	returns object of GeoLocation which can be used to get the location information of the device.
onLine	specifies whether the browser is online or not.
language	returns a string with the preferred browser language, for example, **en-US** for English.
languages	returns a string with all the languages supported by the browser in order of user preference.

JavaScript Object Property

```
<html>
<head>
        <title>JS Navigator Object Properties</title>
    </head>
    <body>
        <h3>Navigator Object Example</h3>
<script>
letappc =  navigator.appCodeName;
letappn =  navigator.appName;
letappv =  navigator.appVersion;
letappco = navigator.cookieEnabled;
letlan   = navigator.language;
letonl = navigator.onLine;
letpla = navigator.platform;
letusra = navigator.userAgent;
document.write(appc +"<br>");
```

```
document.write(appn +"<br>");
document.write(appv +"<br>");
document.write(appco +"<br>");
document.write(lan +"<br>");
document.write(onl +"<br>");
document.write(pla +"<br>");
document.write(usra +"<br>");
</script>
    </body>
</html>
```

```
navigator.appName: Netscape
navigator.appVersion: 5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
navigator.cookieEnabled: true
navigator.language: en-US
navigator.userAgent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
navigator.platform: Win32
navigator.onLine: true
navigator.appCodeName: Mozilla
```


The location object contains information about the current URL.

The location object is part of the window object and is accessed through the window.location property.

Note: There is no public standard that applies to the location object, but all major browsers support it.

Location Object Properties

Property	Description
hash	Sets or returns the anchor part (#) of a URL
host	Sets or returns the hostname and port number of a URL
hostname	Sets or returns the hostname of a URL
href	Sets or returns the entire URL
origin	Returns the protocol, hostname and port number of a URL
pathname	Sets or returns the path name of a URL
port	Sets or returns the port number of a URL
protocol	Sets or returns the protocol of a URL
search	Sets or returns the querystring part of a URL

Location Object Methods

Method	Description
assign()	Loads a new document
reload()	Reloads the current document
replace()	Replaces the current document with a new one

```
navigator.appVersion: 5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36

(KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36

navigator.cookieEnabled: true

navigator.language: en-US

navigator.userAgent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36

(KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36

navigator.platform: Win32

navigator.onLine: true
```

Document Object Model

The **document object** represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

What is Document Object Model (DOM)

The Document Object Model (DOM) is an application programming interface (API) for manipulating HTML documents.

The DOM represents an HTML document as a tree of nodes. The DOM provides functions that allow you to add, remove, and modify parts of the document effectively.

Note that the DOM is cross-platform and language-independent ways of manipulating HTML and XML documents.

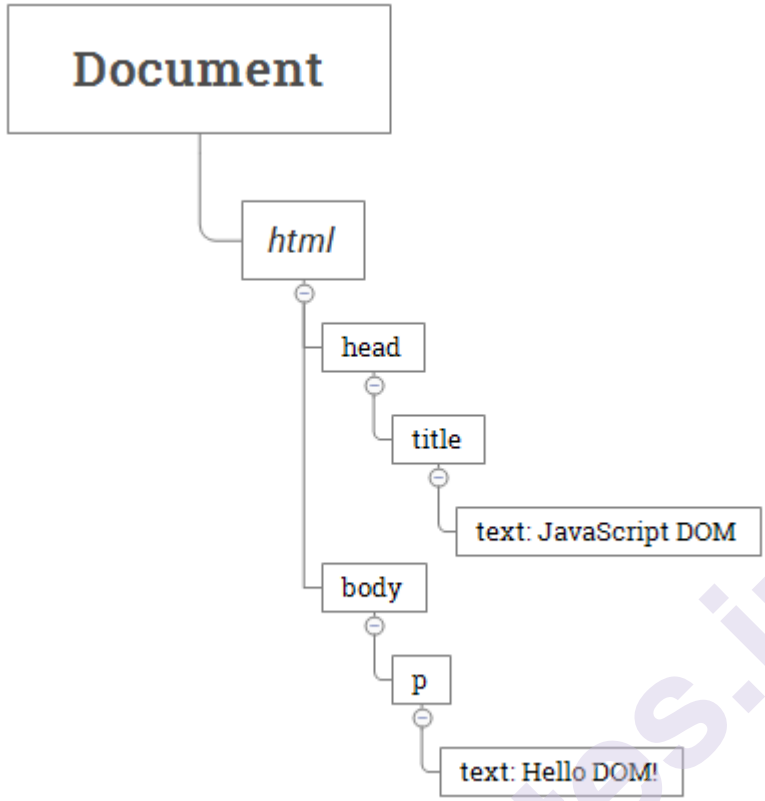
A document as a hierarchy of nodes

The DOM represents an HTML document as a hierarchy of nodes. Consider the following HTML document:

```
<html>
<head>
<title>JavaScript DOM</title>
</head>
<body>
<p>Hello DOM!</p>
</body>
</html>
```

Code language: HTML, XML (xml)

The following tree represents the above HTML document:



In this DOM tree, the document is the root node. The root node has one child node which is the <html> element. The <html> element is called the *document element*.

Each document can have only one document element. In an HTML document, the document element is the <html> element. Each markup can be represented by a node in the tree.

A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to other objects which allow access to and modification of document content.

The way a document content is accessed and modified is called the **Document Object Model**, or **DOM**. The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.

- **Window object** − Top of the hierarchy. It is the outmost element of the object hierarchy.
- **Document object** − Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.
- **Form object** − Everything enclosed in the <form>...</form> tags sets the form object.

- **Form control elements** − The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.

9.10 JAVASCRIPT FORM VALIDATION

It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.

JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation.

Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

JavaScript Form Validation Example

In this example, we are going to validate the name and password. The name can't be empty and password can't be less than 6 characters long.

```
<html>

<body>

<script>

functionvalidateform(){

var name=document.myform.name.value;

var password=document.myform.password.value;

if (name==null || name==""){

alert("Name can't be blank");

return false;

}else if(password.length<6){

alert("Password must be at least 6 characters long.");

return false;

  }

}

</script>

<body>

<form name="myform" method="post"
action="http://www.javatpoint.com/javascriptpages/valid.jsp"
onsubmit="return validateform()" >
```

```
Name: <input type="text" name="name"><br/>

Password: <input type="password" name="password"><br/>

<input type="submit" value="register">

</form>

</body>

</html>
```

JavaScript Number Validation

Let's validate the textfield for numeric value only. Here, we are using isNaN() function.

```
<html>

<head>

<script type="text/javascript">

functionmatchpass(){

varfirstpassword=document.f1.password.value;

varsecondpassword=document.f1.password2.value;

if(firstpassword==secondpassword){

return true;

}

else{

alert("password must be same!");

return false;

}

}

</script>

</head>

<body>

<form name="f1"
action="http://www.javatpoint.com/javascriptpages/valid.jsp"
onsubmit="return matchpass()">

Password:<input type="password" name="password" /><br/>
```

```
Re-enter Password:<input type="password" name="password2"/><br/>
<input type="submit">
</form></body>
</html>
```

JavaScript email validation

We can validate the email by the help of JavaScript.

There are many criteria that need to be follow to validate the email id such as:

- email id must contain the @ and . character
- There must be at least one character before and after the @.
- There must be at least two characters after . (dot).

```
<html>
<body>
<script>
functionvalidateemail()
{
var x=document.myform.email.value;
varatposition=x.indexOf("@");
vardotposition=x.lastIndexOf(".");
if (atposition<1 || dotposition<atposition+2 || dotposition+2>=x.length){
alert("Please enter a valid e-mail address \n atpostion:"+atposition+"\n dotposition:"+dotposition);
return false;
  }
}  </script>
<body>
<form name="myform"  method="post" action="http://www.javatpoint.com/javascriptpages/valid.jsp" onsubmit="return validateemail();">
Email: <input type="text" name="email"><br/>
<input type="submit" value="register">
</form>
</body>
</html>
```

❖❖❖❖

4

FUNDAMENTALS OF AJAX

Unit Structure

4.0 OBJECTIVES

After going through this unit you will be able to :

- Describe the working Ajax
- Demonstrate asynchronous communication between client and server
- Update the web page without refreshing it
- Develop more interactive, attractive and dynamic web pages
- Handle the data through cookies and sessions

4.1 INTRODUCTION

A website is a means of communication for an individual, company or any organization with the help of internet. It not only contains information like text, numbers, but also images and videos. Today's technological era demands the tools which will allow the management of dynamic contents with the help of databases. It also need the features like user friendliness as well as increased efficiency and speed. However the user is always attracted to the websites where the contents are well organized, eye catchy with animation effects, presentable, and satisfy the need of the user in no time. In order to make our web applications enriched with all the above mentioned features we need various tools to be used. Whereas in this unit we are going to learn Ajax, PHP and JQuery.

Ajax is a tool which dynamically communicates with web server with XmlHttpRequest object which preventsreloading the unnecessary

contents. It only refreshes the specific part of a page which needs to be reloaded. This increases the efficiency.

PHP allows to build dynamic pages with the use of various databases like MySQL, Oracle, Sql and many more.

Jquery is nothing but a Javascript library which works on a motive "Write Less Do More" and provides plenty of built in animations which can be used in our website.

So overall these thee technologies together allows us to build dynamic, attractive and more responsive web pages.

In the consequent chapters we are going to learn all those topics in detail.

4.2 AJAX INTRODUCTION

Ajax stands for Asynchronous Javascript and XML. It is a combination of technologies suchas HTML, XML, Javascript and CSS for web application development. The technology is popular at it makes the web pages more responsive. All this is possible because data exchange process in client and server also can happen after the page has been loaded and hence it is fast. It means the data can be sent to the server in background. The reason behind this is Ajax is a Data Driven web browser technology independent of any web server software. This technology is used in various web applications such as google maps, youtube, facebook, twitter etc. Though it is a Rich Internet Application (RIA) technology it has browser incompatibility and is supported by Javascript.

The technologies used in Ajax are

1. Javascript : It used to bind the data requests and displays the information. It is also used for validating the data provided by the user i.e user inputs in HTML and then sends the data to the server.

2. DOM : Document Object Model is used for dynamic interaction and displaying the presented information

3. CSS : CSS is an acronym for Cascading Style Sheet and is mainly used for displaying the content and style i. e. for presentation purpose

4. XmlHttpRequest :It allows asynchronous communication between client and server.

5. XML, HTML and XSLT :XSLT is a style sheet language for XML which allows transformation. The purpose of these technologies is for displaying the content in a presentable manner.

Applications of Ajax :

We can see the use of Ajax in various areas such as:

1. Google Autocomplete Feature:It assists to complete the keywords while they are being typed. The keywords used to change in real time, whereas the page remains the same.

   Eg: In the below screen in the search box the keyword entered is places but it is showing various suggestion starting from the keyword places.

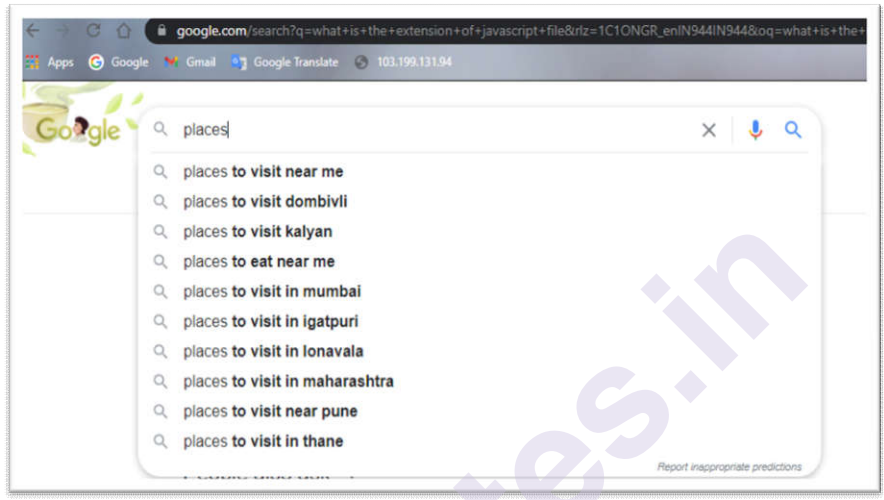


   Figure 4.1 Autocomplete feature by google

2. Voting and Rating System:
   Whenever we purchase anything online it used to ask rating for the product. We also have seen various online voting forms. So when we click on the rating or voting button the website updates the calculation but the whole page remains unchanged.

3. Chat Rooms:
   Some websites uses built in chat room which provides s support services by enabling the communication with their customer support officer. However there is no need to explore the page at the same time because Ajax sends and receives new message without reloading the page.

4. Twitter's Trending Notification:
   Twitter has recently used AJAX for their updates. Whenever new tweets are made regarding trending topic; Twitter updates the new figures without affecting the main page.

5. Form Validation : The credentials such as user Id and Password can be validated using Ajax

   So we can see that applications such as google maps, twitter, Gmail, Youtube, Google Suggest, Facebooketc are using this technology.

Advantages of Ajax :

1. Easy and Simple: Ajax is very easy to understand if the user is having strong knowledge of Javascript
2. Speed : It allows to reduce the server traffic and therby increase the speed.
3. Responsive : It helps the user to create more responsive, Faster and interactive web applications
4. Asynchronous Calls : Through asynchronous calls to the web server it sees how the data arrival time can be reduced
5. Form Validation : Validation is the most necessary part of any form data. Ajax usesprompt and proper validation methods.
6. Bandwidth Usage : Ajax based applications needs very less bandwidth as they are not refreshing the whole page. They are just reloading the part of the page which is required to be updated.

Diadavantages of Ajax :

1. Browser Support : It does not run all browser. Rather before actual execution we need to check the browsers which are supporting it.
2. Security and User Privacy :Issues regarding user security and privacy are need to be taken care while developing the ajax applications.
3. Accessibility : Since all browsers does not support Javascript and XmlHttpRequst Object, one must look into the part how to make the web applications accessible to all the users.
4. Bookmark :Since Ajax uses asynchronous calls it loads bits of contents on the existing web page instead of reloading the contents of whole page. Whereas the Browser history and bookmarks may not behave correctly since URL is unchanged and only some part of it has been changed.
5. Navigation : When the back button is pressed the user thinks that it will switch over to the last change which he has made. However Ajax does not support that.
6. Search Engine : It is possible to search Ajax features and elements within an application but we can not search for Ajax applications. Search Engines like google can not index Ajax pages

4.2.1 AJAX WEB APPLICATION MODEL

Web application model of Ajax uses XmlHttpRequest and Javascript for asynchronous data communication over client server architecture.

It provides solutions to the problem of synchronous request-response model of communication related to classical web application models where the user is kept on wait state and thus do not provide better experience to the user.

New approach of Ajax with regard to web applications is based on various technologies which provides better user experience and helps to develop interactive web applications.

The applications of Ajax rejects start-stop or click and wait actions. It creates an intermediary layer between the user and the web server to eliminate the refresh criteria of client server interaction.

At the beginning of the session it does not load web page instead the the browser loads the Ajax engine which is written in javascript. However each and every action of the user generates HTTP request which in turn makes a javascript call to the Ajax Engine.

The response provided by the server contains only data but not presentation which indicates that data required by the client is provided by the server as the response and presentation implemented on that data with the help of XML from Ajax Engine.

Javascript does not executes all the instructions rather it updates the web page dynamically

Javascript make it possible to fill the forms and click on the buttons even when javascriptsends request to the web server and at the same time server is working on the requests in the background. When server completes its processing code updates only that part of page which has changed. This is how the client need not wait around. This is nothing but called as asynchronous requests.

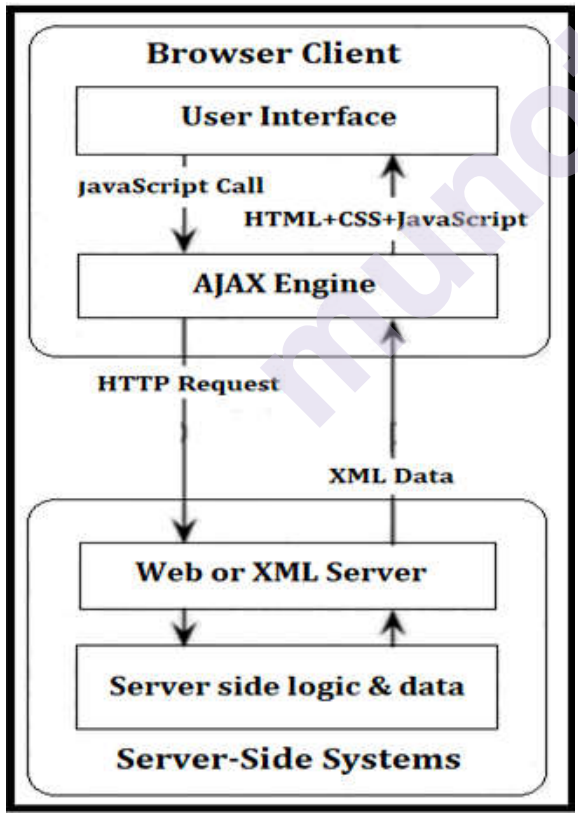


Figure 4.2 Ajax Web Application Model

Irrespective of server Ajax does asynchronous communication between client and the application. This prevents the user from waiting for the server to complete its processing.

The Ajax engine displays the user interface and interact with server on behalf of user

Following figure shows synchronous mode of communication between client and server in traditional web applications

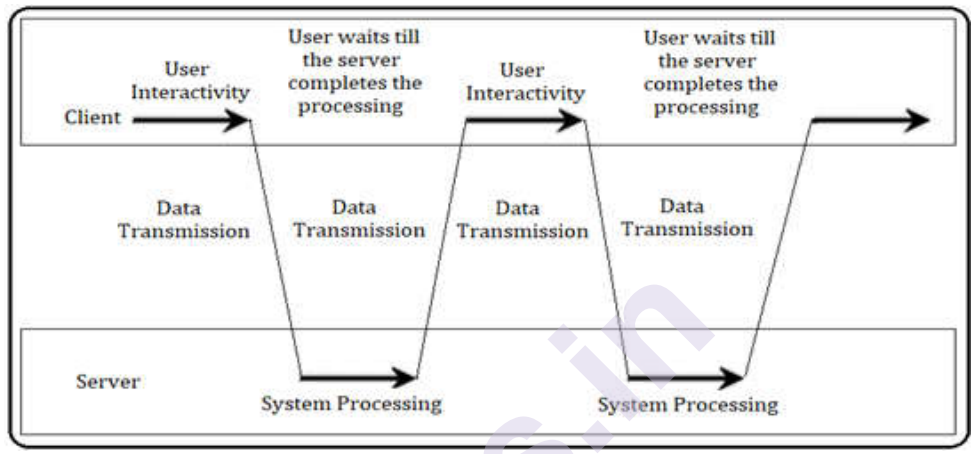


Figure 4.3 Synchronous Mode of Communication

Essentially Ajax is used for partial update and asynchronous communication, the Ajax model is used for programming and not restricted for any specifications such as specific data exchange format or specific programming language or specific communication mechanism

Following figure shows Asynchronous mode of communication

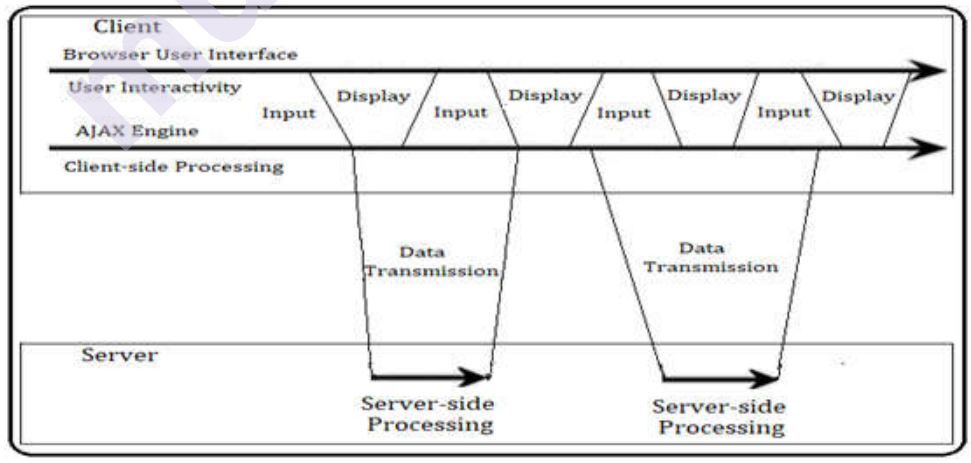


Figure 4.4 Asynchronous Mode of Communication

This figure clarifies every action done by the user results in generation of HTTP request which takes the form of a JavaScript to call the Ajax Engine.

Ajax engine handles Data validation, Navigation and data editing in memory on its own as compared to classical web application model where responses to the user action do not involve the trip back to the server.

Ajax engine makes an asynchronous interaction with server using JavaScript and XmlHtttpRequest Object. This communication does not interrupt user interaction with application.

4.2.2 Working of Ajax

The communication between Ajax and Web Server happens using XmlHttpRequest Object

Following sequence show the workflow of Ajax

1. Through the User Interface user sends a request and a javascript call goes to XmlHttpRequest Object
2. XmlHttpRequest Object then sends the HTTP request to the server
3. Server then interacts with database using programming languages such as PHP ASP.net, JSP etc.
4. Data is retrieved
5. Server sends XML data to the XmlHttpRequest callback function
6. On the web browser contents of HTML and CSS are displayed.

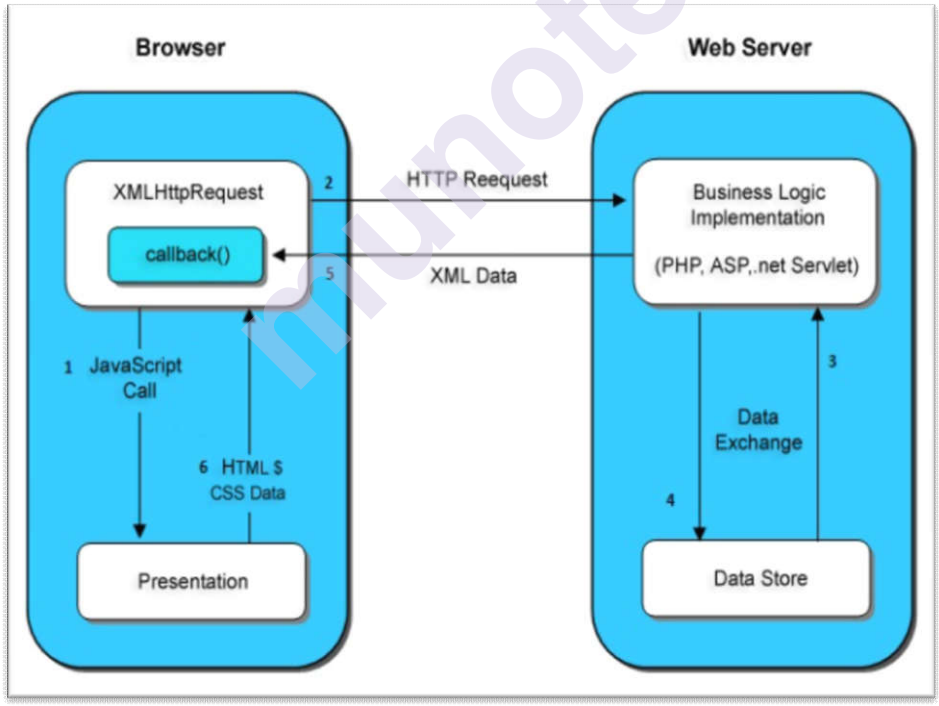


Figure 4.5 Working of Ajax

In the Ajax model Ajax Engine is involved between client and server which eliminates the process of communication from user to server and then again back from server to user. The Ajax Engine is written in Javascript. It handles user from by communicating to user and handles server side by itself. This is how eradicates waiting time faced by the user.

Comparison between Convention Model and Ajax Model

Sr. No.	Conventional Model	Ajax Modes
1.	Web Browser send HTTP request to the server	The browser creates javascript call which results in activating XmlHttpRequest
2.	The server receives the request processes it and retrieves the data	In the background HTTP request is created by the browser and sent to the server
3.	Web browser gets the requested data which is sent by the server	The server receives, retrieves and sends the data back to the web browser
4.	After receiving the data to the web browser the page is to be reloaded to make the data appear	The requested data is received by the browser and without reloading it appears on the page
5	This process is time consuming as well as puts	This process is fast as compared to conventional model

4.2.3 XmlHttpRequest Object

All the modern browsers supports XmlHttpRequest Object which is used to request data from the server. Following illustration will help us to understand uses of this object

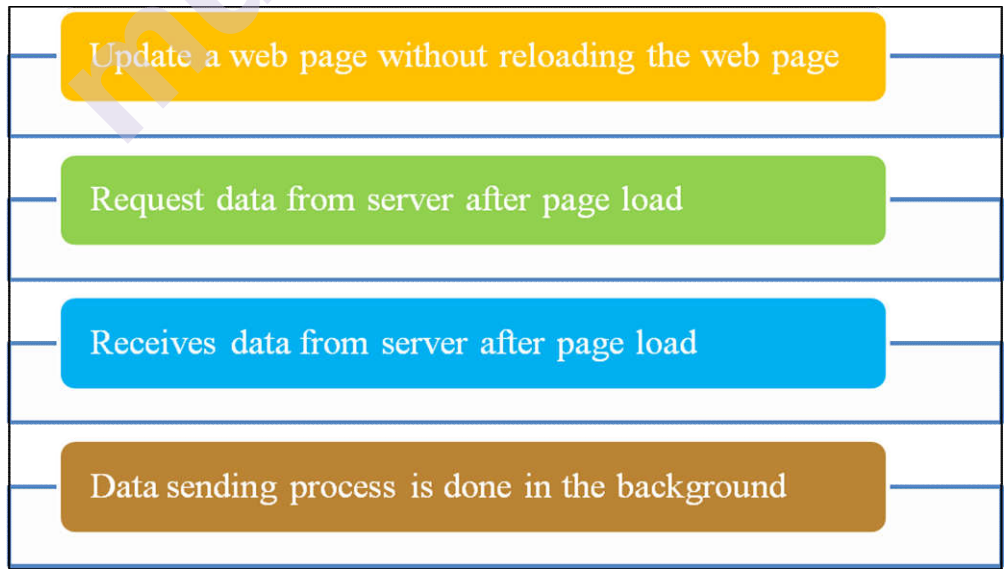


Figure 3.6 Features of XmlHttpRequest Object

This object consists of various methods and propertires.

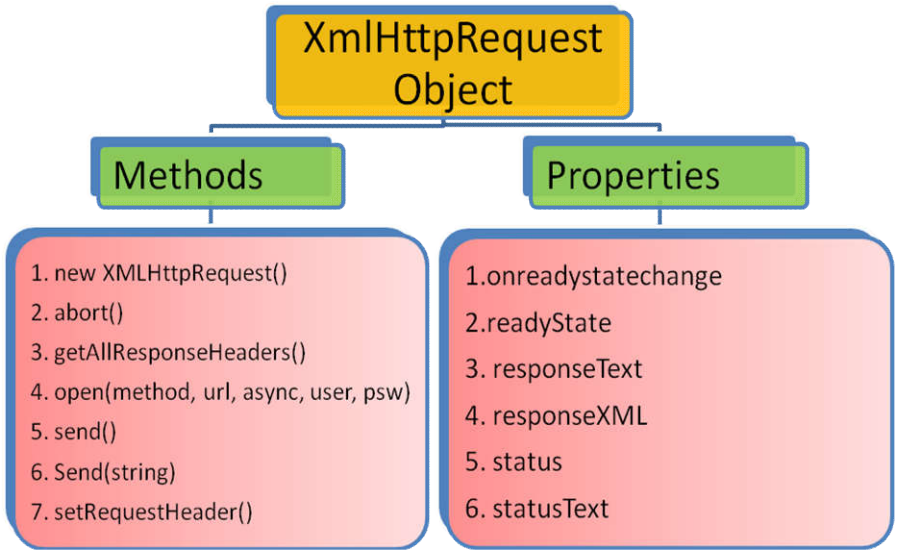


Figure 3.7 Methods and properties of XmlHttpRequest Object

We will understand all those methods and properties one by one.

4.2.3.1 XmlHttpRequest Object Methods:

Sr.No	Method	Description
1.	new XML Http Request()	Allows to create new XmlHttpRequest object
2.	abort()	Cancels the current request
3.	Get All Response Headers()	Returns header information
4.	Get Response Header()	Returns specific header information
5.	open(method,url,async,user,psw)	Opens the request with following parameter specifications Method : The request type GET or POST Async : true (asynchronous) or false (synchronous User : username Psw : password
6.	Send()	Sends the request to the server Used for GET requests
7.	Send(string)	Sends the request to the server. Used for POST requests

4.2.3.2 XmlHttpRequest Object Properties:

Sr. No	Propery	Description
1.	Onreadystatechange	Defines a function to be called when the readyState property changes
2.	readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
3.	responseText	Returns the response data as a string
4.	responseXML	Returns the response data as XML data
5.	Status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to theHttp Messages Reference
6.	statusText	Returns the status-text (e.g. "OK" or "Not Found")

4.2.4 Handling asynchronous requests using Ajax:

We already know that Ajax technology allows to make asynchronous HTTP request without reloading the full page. To make this process more easy as compared to in pure javascript,JQuery libraries can be used. JQuery's most common Ajax shorthand methods are $.get(), $.post() and $.load().

Whenever we specify an event and if Ajax call fails or Ajax request could not give us the requested result in certain time period in that case we can rely on other function provided by JQuery i.e. $.ajax(). This function is used to perform an asynchronous HTTP request.

Following is the syntax of the $.ajax() function

```
$.ajax(url [ , settings])

$.ajax([settings])
```

The url is the location where we want rich with the help of ajax call. Whereas settings allows to make configurations for the Ajax request.

Following are some of the setting parameters of $.ajax()

Sr.No	Setting	Description
1.	accepts	Specifies the type of response will get in return to the request sent
2.	async	Makes asynchronous request. If we want to make synchronous request it can be set false
3.	cache	If this option is set as false it forces the requested page not to be cached by the browser
4.	beforeSend	Pre request call back function
5.	complete	A function to be called when the request finishes
6.	contents	This object determines how the library will parse the responses.
7.	contentType	Data content type sent to the server
8.	context	Context **this** is used for all ajax related call backs
9.	converters	Data type to data type converter
10.	crossDomain	To force a cross domain request can set this as True
11.	Data	Data which is sent to the server while making ajax request
12.	dataFilter	handles the raw response data of XMLHttpRequest.
13.	dataType	The type of data expected back from the server.
14.	Error	A function to be called if the request fails.
15.	global	Whether to trigger global Ajax event handlers for this request.
16.	headers	An object of additional headers to send to the server.
17.	statusCode	An object of numeric HTTP codes and functions to be called when the response has the corresponding code
18.	success	A function to be called if the request succeeds
19.	timeout	Specifies a timeout (in milliseconds) for the request.

Summary:

Asynchronous JavaScript and XML (Ajax) refer to a group of technologies that are used to develop web applications.

Ajax enables a web application user to interact with a web page without the interruption of constant web page reloading.

Ajax is made up of various technologies such as XHTML, CSS, Document Object Model (DOM), XMLHttpRequest object, XML, HTML, and XSLT and JavaScript

XMLHttpRequest Object is used to send and receive to and from the server in the background . It works with various methods and properties

Practice Questions:

1. Explain what is Ajax and also state the technologies through which the ajax is made up of
2. Write advantages and limitations of ajax technology
3. Explain Ajax web application model with the help of diagram
4. Explain various modes of communication in client and server in web applications
5. How does ajax works?
6. Write the difference between conventional model and ajax model
7. Explain XmlHttpRequest object with its methods and properties

Web References:

https://www.mageplaza.com/blog/advantages-and-disadvantages-of-ajax.html

https://way2tutorial.com/ajax/ajax-advantages-and-disadvantages.php

https://www.sitesbay.com/ajax/ajax-features

https://www.ques10.com/p/29472/explain-in-detail-ajax-web-application-model-wit-1/

5

FUNDAMENTALS OF PHP

Unit Structure

5.1 PHP

PHP. It is a recursive acronym of Hypertext Preprocessor. It is developed by RasmusLerdorf in the year 1994

PHP is an open source, object oriented, interpreted, server side scripting language embedded in HTML which allows to develop dynamic web pages. Software Applications are developed using this language. To bring the dynamism in this language it is integrated with various popular databases such MySQL, Oracle, Microsoft SQL server, Sybase, PostgreSQL etc. PHP is simple and easy to learn as the syntax of PHP is very similar to c language. PHP is faster than any other scripting languages such as ASP or JSP

Following are the features of PHP:

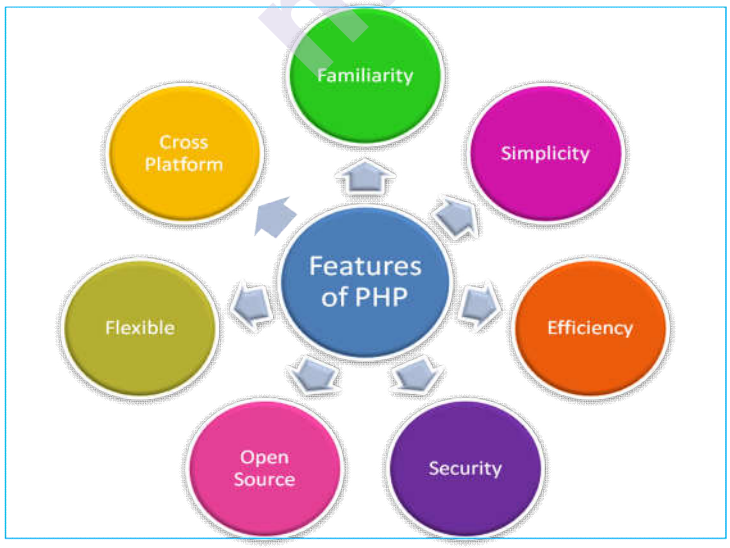


Figure 5.8 Features of PHP

1. Familiarity : The person having programming background can easily understand syntax of PHP as the syntax is inherited by C language.

2. Simplicity: Like C language there is no need to include any libraries. It provides lot many predefined functions which helps to secure the data.PHP programs can be written without creating classes.

3. Efficiency: PHP supports object oriented programming and it also supports session management which makes it more efficient as it does not require unnecessary memory allocation.

4. Security: PHP's predefined function sets supports trusted data encryption options and thereby provides security.

5. Open Source :PHP is an open source programming language so you can download freely there is no need to buy a licence or anything.

6. Flexible: PHP cam be embedded with various languages such as HTML, JAVA SCRIPT, WML, XML, and many others. PHP scripts can run on any devices ie scripts are executed on the server and the result is sent to the browser of a device. Hence PHP is flexible.

7. Cross Platform : PHP scripts can be executed on any operating system such as windows, Linux, Mac, Solaris etc.

PHP Installation

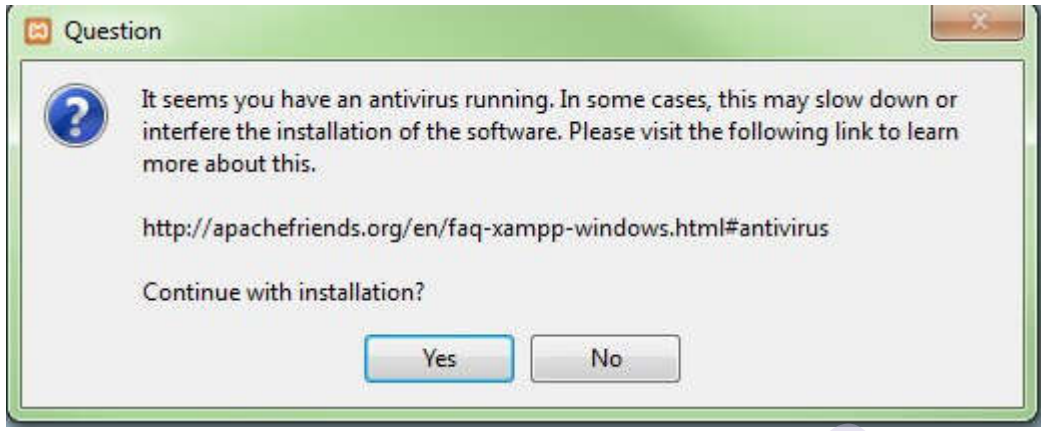
Various AMP (Apache, MySQL and PHP) software stacks are available for PHP installations such as

1. WAMP : Used in Windows
2. LAMP : Used in Linux
3. MAMP : Used in Mac
4. SAMP : Used in Solaris
5. FAMP : Used in FreeBSD
6. XAMPP : It is cross platform and is used as acronym for (Cross, Apache, MySQL, PHP and Perl) It can be used in any operating system. It also includes the components such as FileZilla, OpenSSL, Webalizer, Mercury Mail, etc.

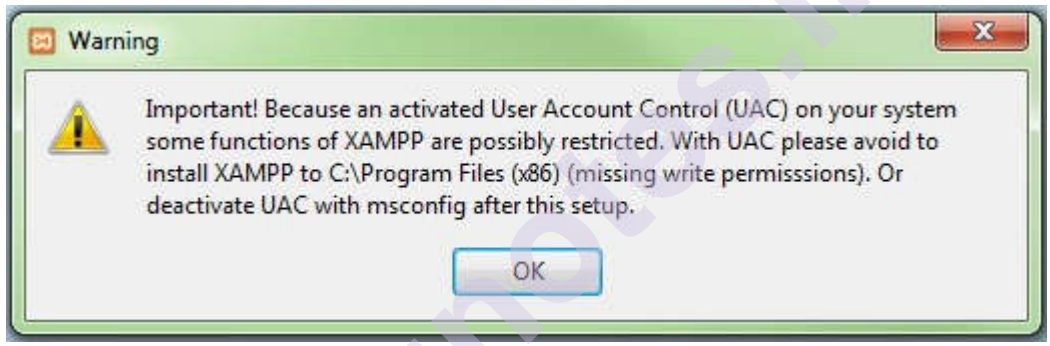
Now we will see how to install XAMPP in our computer having Windows operating System

Step 1 : Download the software using link
https://www.apachefriends.org/download.html

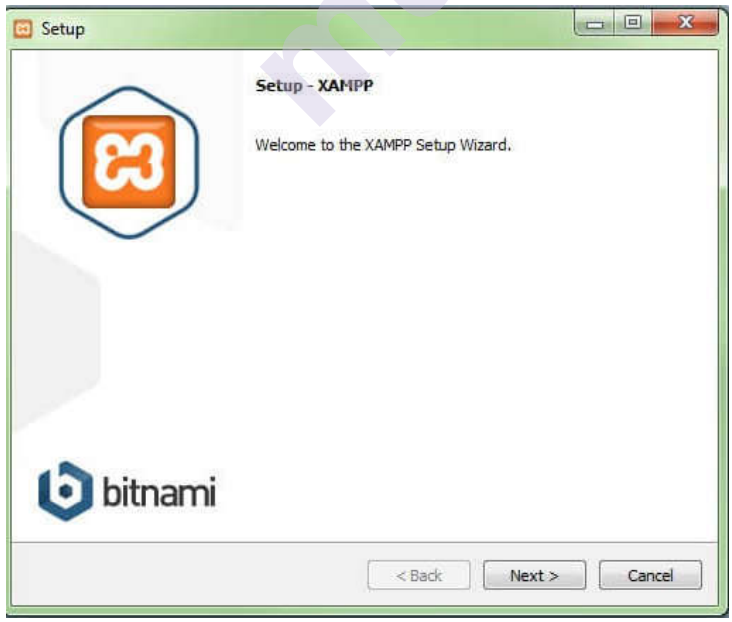
Step 2 : Run the exe. To allow the installation we have to deactivate the antivirus software



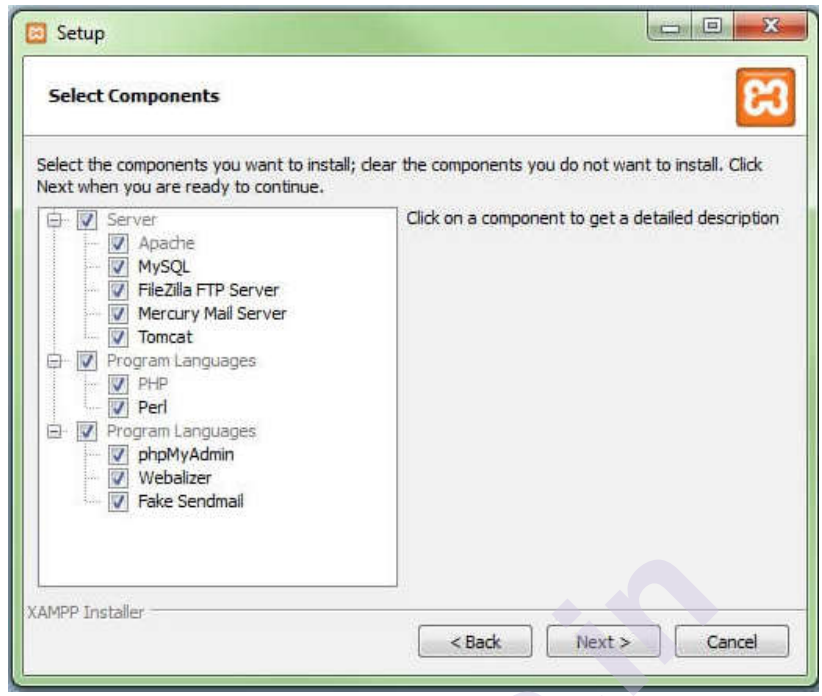
Step 3 : Deactivate User Account Control



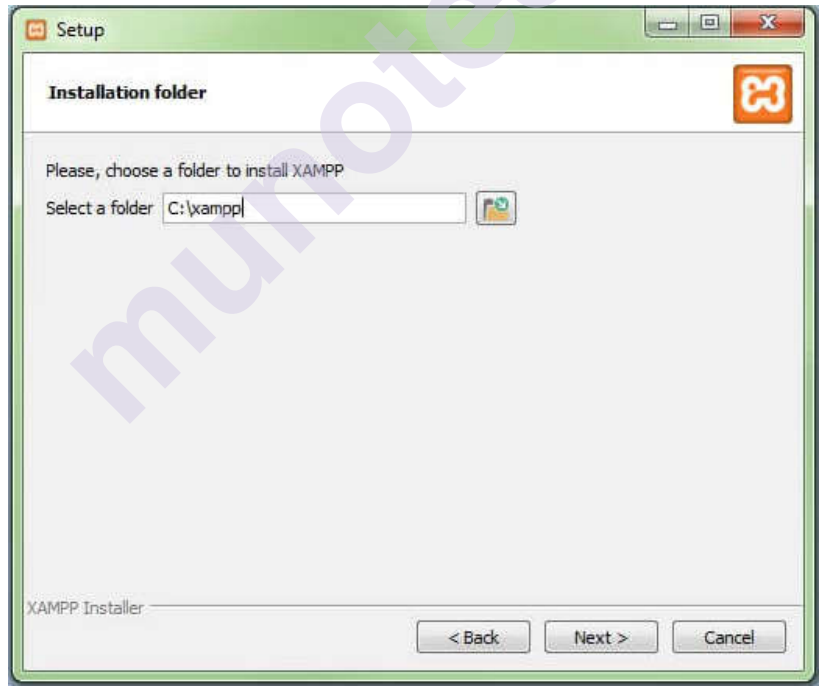
Step 4 : Start the setup wizard



Step 5 : Select software components

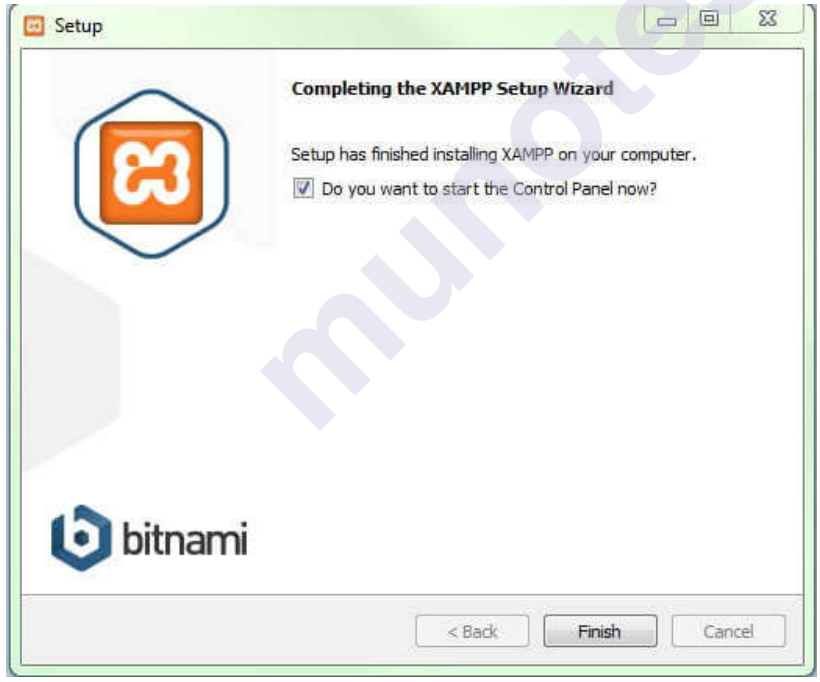


Step 6 : Select the installation folder

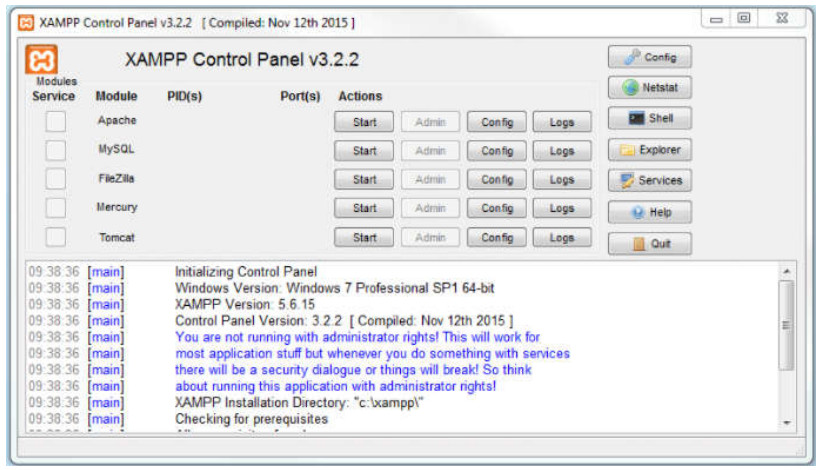


Setp7 : Start installation process

Step 8 : Complete the installation



Step 9 :Open the control panel



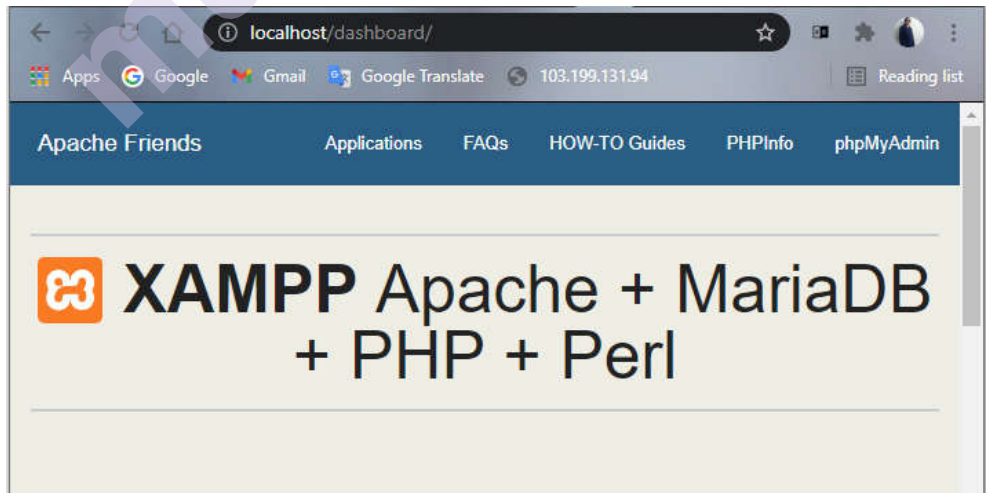
Here we can activate the required services. The default port for Apache server is 80.

After activating the services we can start writing the php program and execute it. The things to be noted that The PHP file should be saved with an extension .php and are always to be saved htdocs folder. As per step no 6 we can see that xampp folder is getting created in C drive so you will be able to find the htdocs folder in C:/xampp/htdocs path .

To view the program execution

We have to perform following steps

1. Open the browser and type localhost:80/ and press enter as it the default port for apache server.
2. It will show the dashboard



3. To execute a program click on address bar and remove dashboard and type name of your php file and press enter. The output will be displayed in the browser.

5.1.1 Variables and Operators

Basic PHP Syntax

A PHP script starts with <?php and ends with ?>

Single line comments will be written using // symbol or # symbol. However multiline comments are written within /* */ symbols

Rules to be kept in mind while declaring variables

1. A variable should always start with an alphabet or an underscore character only and may have (A-Z, a-z,0 to 9 and an underscore) in it
2. In PHP variable name always starts with $ symbol. Eg : $a=10;
3. A variable name can not start with number
4. Names of the variable are case sensitive eg $sname and $SNAME will be considered as different variables

PHP Datatypes

PHP Datatypes

PHP supports various data types. However whenever we define any variable with a value it used to consider the type of variable will be which is matching the value of the variable

Eg. If it is written as $a=10 then it automatically considered as datatype of variable will be integer. If we write $a=”mystring” then it considers the datatype of variable a as string.

var_dump() function allows us to check the datatype of the variables..

Eg. The code snippet given below shows us the result as **int(10)**

```
<html>
<body>

<?php
$x = 10;
var_dump($x);
?>

</body>
</html>
```

Here we will see all the datatypes briefly:

1. Integer : This datatype considers only non fractional positive or negative whole numbers they are ranging from -2,147,483,648 and 2,147,483,647. It is declared as $age=21;

2. Float : This datatype allows to store floating point real numbers means decimal values. It is declared as float $weight=15.30;

3. String : String is nothing but a group of single characters stored in consecutive memory locations. The value of string variable is written in double quotations. Eg: $name=”nisha”;

4. Boolean : This datatype used to represent only two possible states i.e. True or False. They are normally used while testing the conditions.Eg : $x=true;

5. Array : The datatype array is having capacity to store multiple similar datatype values inside one single variable. It is declared as

   Eg :$flowers=array(“lily”, “Jasmine”, “lotus”);

6. Object : Under object oriented programming classes and objects are two main concepts . A class defines structure of the object and and object is instance of the class. When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

7. Null : Whenever we want to create any variable without value or we have to remove the value of existing variable then null datatype is used.

Eg $x=”nisha”;

$x=null;

8. Resource : This datatype allows to store references to functions and resources external to PHP

PHP Operators

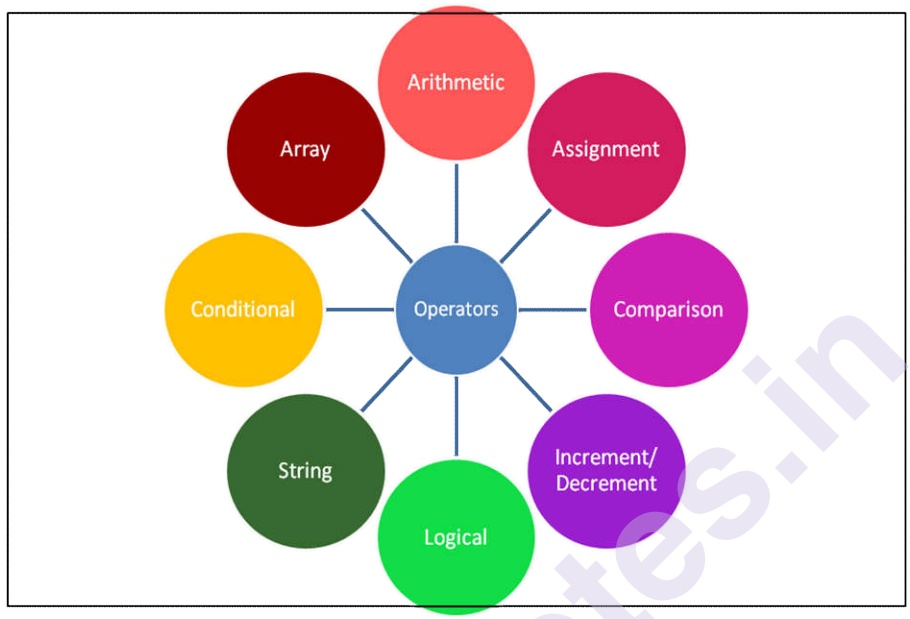


Figure PHP Operators

1. Arithmetic operators

Sr. No.	Operator	Name	Example	Description
1	+	Addition	$a + $b	Calculates Sum of operands
2	-	Subtraction	$a - $b	Calculates Difference of operands
3	*	Multiplication	$a * $b	Calculates Product of operands
4	/	Division	$a / $b	Calculates Quotient of operands
5	%	Modulus	$a % $b	Calculates Remainder of operands
6	**	Exponentiation	$a ** $b	Calculates $a raised to the power $b

2. AssignentOpeators

Sr.No	Operator	Name	Example	Description
1	=	Assign	$a = $b	Assigns the value of RHS Operand to LHS Operand
2	+=	Add and then Assign	$a += $b	Evaluates as $a = $a + $bt
3	-=	Subtract and then Assign	$a -= $b	Evaluates as $a = $a - $b
4	*=	Multiply and then Assign	$a *= $b	Evaluates as $a = $a * $b
5	/=	Divide and then Assign (quotient)	$a /= $b	Evaluates as $a = $a / $b
6	%=	Divide and then Assign (remainder)	$a %= $b	Evaluates as $a = $a % $b

3. Comparison Operators

Sr.No.	Operator	Name	Example	Description
1	==	Equal	$a == $b	Returns TRUE if $a is exactly equal to $b
2	===	Identical	$a === $b	Returns TRUE if $a is exactly equal to $b, and if their datatype is also same
3	!==	Not identical	$a !== $b	Returns TRUE if $a is not equal to $b, and they are not of same data type
4	!=	Not equal	$a != $b	Returns TRUE if $a is not equal to $b
5	<>	Not equal	$a <> $b	Returns TRUE if $a is not equal to $b
6	<	Less than	$a < $b	Returns TRUE if $a is less than $b
7	>	Greater than	$a > $b	Returns TRUE if $a is greater than $b
8	<=	Less than or equal to	$a <= $b	Returns TRUE if $a is less than or equal $b
9	>=	Greater than or equal to	$a >= $b	Returns TRUE if $a is greater than or equal $b
10	<=>	Spaceship	$a <=>$b	Returns -1 if $a is less than $b Returns 0 if $a is equal $b Returns 1 if $a is greater than $b

4. Increment / Decrement Operators

Sr.No.	Operator	Name	Example	Description
1	++	Pre Increment	++$a	Increases the value of $a by one, then returns the value of $a
2		Post Increment	$a++	First returns the value of $a, and then increases the value of $a by one
3	--	Pre Decrement	--$a	Decreases the value of $a by one, and then returns the value of $a
4		Post Decrement	$a--	First return the value of $a, and then decreases the value of $a by one

5. Logical Operatots

Sr. No.	Operator	Name	Example	Description
1	&&	And	$a and $b	Returns TRUE if both $a and $b are true
2	||	Or	$a or $b	Returns TRUE if either $a or $b is true
3	xor	Xor	$a xor $b	Returns TRUE if either $ or $b is true but not both
4	!	Not	! $a	Returns TRUE if $a is not true
5	and	And	$a and $b	Returns TRUE if both $a and $b are true
6	or	Or	$a or $b	Returns TRUE if either $a or $b is true

6. String Operators

Sr.No.	Operator	Name	Example	Description
1	.	Concatenation	$a . $b	Concatenate both $a and $b
2	.=	Concatenation and Assignment	$a .= $b	First concatenate $a and $b, then assign the concatenated string to $a, e.g. $a = $a . $b

7. Conditional Operators

Sr. No.	Operator	Name	Example	Description
1	? :	Ternary	$x = expr1 ? expr2 : expr3	Returns the value of $x. The value of $x is expr2 if expr1 = TRUE. The value of $x is expr3 if expr1 = FALSE
2	? ?	Null Coalescing	$x = expr1 ? ? expr2	Returns the value of $x. The value of $x is expr1 if expr1 exists, and is not NULL. If expr1 does not exist, or is NULL, the value of $x is expr2

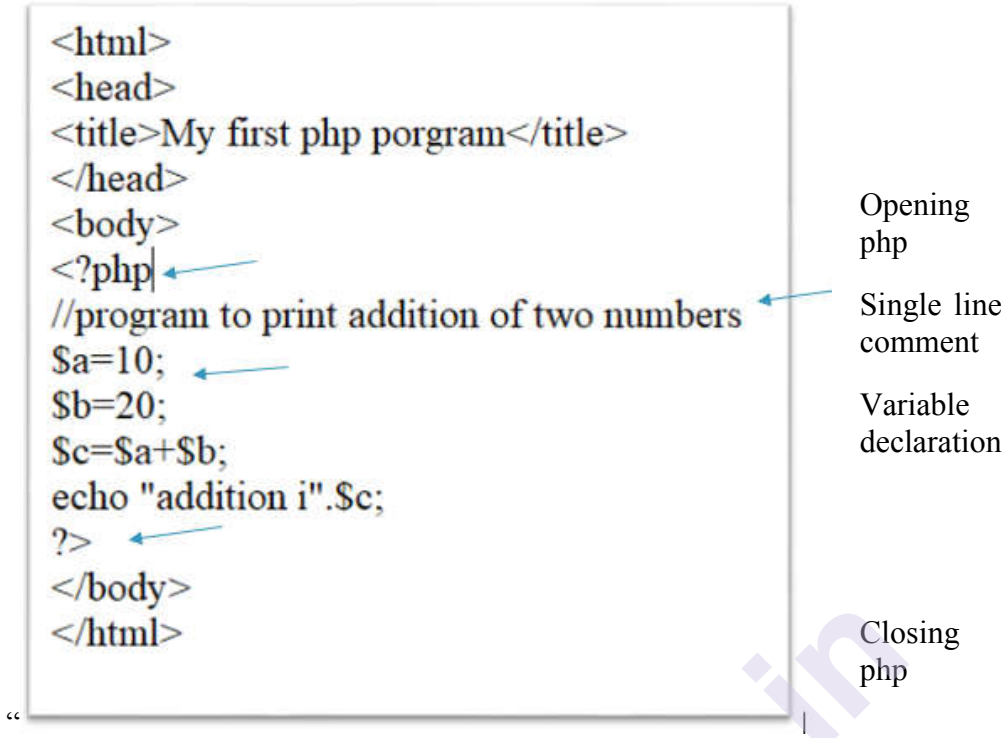
8. Array Operators

Sr. No.	Operator	Name	Example	Description
1	+	Union	$a + $b	Returns Union of $a and $b
2	==	Equality	$a == $b	Returns TRUE if $a and $b have same key/value pair
3	!=	Inequality	$a != $b	Returns TRUE if $a is not equal to $b
4	===	Identity	$a === $b	Returns TRUE if $a and $b have same key/value pair of same type in same order
5	!==	Non-Identity	$a !== $b	Returns TRUE if $a is not identical to $b
6	<>	Inequality	$a <> $b	Returns TRUE if $a is not equal to $b

5.1.2 Program Flow

First we will try to understand syntax of phpprogram.with the help of following example

Here is a code snippet which shows addition of two numbers in php



In the given code snippet

PHP code is embedded with HTML. <?php is used as opening php which makes us understand that now we are started writing the code using php script whereas ?> at the end denotes that here is the end of php script. So within <?php ?> block we are writing php code. In the line //Program to print addition of two numbers you can see the symbol // which is used as a single line comment. In PHP we can give single line comment using // symbol or # symbol. If we want to give multiline comment we can write the comment statements in /* */ block.

Then $a=10; $ b=20; are showing how we have declared the variables. So from these lines we can understand that $ symbol is used before name of the variables whenever we are using the variable.

However the statement echo "addition is" .$c; is indicating that we are printing the addition of a and b. Here . (dot) is the operator which is used for concatenation purpose.

PHP supports various programming structures such as Simple If, Nested If, Decision Making using ternary operators, Switch case, iterative Statements etc.

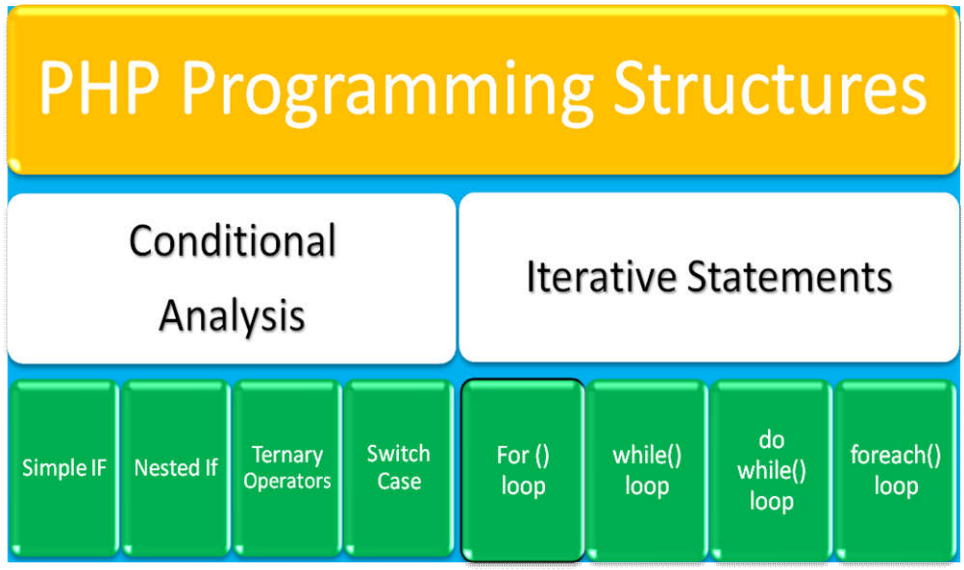


Figure PHP Programming Structure

Conditional Analysis :

This is the way how computers can make decisions based on conditions. In programming various structures are used such as if, if-else, if –else-if, nested if, Ternary operators and switch case. We will try to understand all those concepts with the help of programs

1. If -else :
Syntax :

```
if (condition)
        Statement     //if the condition appears to be true
else
        Statements     // if the condition appears to be false
```

Example : Write a program to print whether the number is even or odd

```
<html>
<body>
<?php
$n=10;
If($n%2= = 0)
        echo “Entered no is Even”;
else
        echo “Entered no is Odd”;
?>
</body>
</html>
```

2. Nested If

Syntax :

```
If (Condition)
        Statement   // If condition appears to be true
Else if (Condition)   //  if first condition is appearing to be false then applying 2nd condition
        Statement   // if second condition appears to be true
       Else
        Statement    // if Second condition appears to be false
```

Example : Write a program print whether a number is positive, negative or zero

```
<html>
<body>
<?php
$n=-10;
if($n>0)
echo "entered number is positive";
else if ($n<0)
        echo "Entered number is negative";
else
        echo "Enter number is zero";
?>
</body>
</html>
```

3. Ternary operators

Syntax

```
(condition) ? <Statements if condition is true> :<statements if condition is false>
```

Example : Write a program to print whether a number is even or odd

```
<html>
<body>
<?php
$x = 27;

echo "Input Number: ", $x, "\n\n";

$x = $x % 2 == 0 ?1 : 0;

if ($x) {
echo "The input number is even number.\n";
} else {
echo "The input number is odd number.\n";
}
?>
</body>
</html>
```

4. Switch Case

Syntax

```
switch(expression){
case value1:
 //code to be executed
 break;
case value2:
 //code to be executed
 break;
.....
default:
 code to be executed if all cases are not matched;
}
```

Example : Write a program to accept a day of week in number and print respective day

```
<html>
<body>
<form name="f1" method="Post">
<table>
<tr>
<th>Enter a number</th>
<td><input type="text" name="weekdaynum">
</tr>
</table>
<?php
$x=$_POST['weekdaynum'];
switch($x)
{
        case 1 :
        echo "sunday";
        break;
        case 2 :
        echo "monday";
        break;
        case 3 :
        echo "tuesday";
        break;
        case 4:
        echo "wednesday";
        break;
        case 5 :
        echo "thursday";
```

```
        break;
        case 6:
        echo "friday";
        break;
        case 7:
        echo "saturday";
        break;
        default:
        echo "not a proper number";
        break;
}
?>
</body>
</html>
```

Iterative Statements :

Iterative statements are also called as looping structures. When certain instructions are to be executed multiple times continuously in such cases we can make use of loops. While working with such loops we must have following thing such as

1. Initial value
2. Final Value
3. Condition
4. Increment / Decrement
5. Body of the Loop

In PHP we come across with 4 different types of loops which we are going to understand with the help of examples. These loops are :

1. For () loop

Syntax

```
For (initial value; final value and condition; incr/decr)
{
        Body of the loop;
}
```

Example : Write a program to print the series of number 1,2,3,…..,10

```
<html>
<body>
<?php
for($x=1;$x<=10;$x++)
{
        echo $x;
        echo "<br>";
}
?>
</body>
</html>
```

2. While () loop

Syntax

```
Initial value;
While (condition with final value)
{
        Body of the loop;
        Increment / decrement ;
}
```

Example : Write a program to print the series of number 1,2,3,…..,10

```
<html>
<body>
<?php
$x=1;
while($x<=10)
{
        echo $x;
        $x=$x+1;
        echo "<br>";
}
?>
</body>
</html>
```

3. Do while() loop

Syntax

```
Initial value;
do
{
        Body of the loop;
        Increment / decrement ;
}
While (condition with final value);
```

Example : Write a program to print the series of number 1,2,3,…..,10

```
<html>
<body>
<?php
$x=1;
do
{
        echo $x;
        $x=$x+1;
        echo "<br>";
}
while($x<=10);
?>
</body>
</html>
```

4. Foreach() loop : This type of loop is used in php to work with array elements.

Syntax

```
foreach ($array as $key => $element)
{
   Statemets to be executed;
}
```

Example : Write a program to print the series of number 1,2,3,…..,10

```
<html>
<body>
<?php
   //array declaration
   $no = array (1,2,3,4,5,6,7,8,9,10);

   //access array elements using foreach loop
foreach ($no as $element) {
echo "$element";
echo "</br>";
   }
?>
</body>
</html>
```


5.1.3. Working with arrays

An array is programming structure where we can stores one or more similar type of values in a single variable. In PHP there are 3 different types of array such as

1. Numeric Array: The above code snippet offoreach() loop is an example of numeric array. There are two different types of methods how we can declare numeric arrays.
Method 1 : $x=array(1,2,3,4,5);
Method 2 :
$x[0]=1;

```
$x[1]=2;
$x[2]=3;
$x[3]=4;
$x[4]=5;
```

2. Associative Array :The functionality of associative arrays are very similar to numeric arrays Just the difference is in their index. Associative array will have their index as string so that one can establish a strong association between indexes and values. These arrays can be declared as follows

$st_marks=array(“krish” => 50, “samiksha” =>67, “Shubhra” =>90);

Here krish, samiksha and shubhra are index keys however 50, 67 and 90 are the values of the index keys.

3. Multidimensional Array :A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Following example will show how multidimensional arrays can be declared.

```
$st_marks = array(
"krish" => array (
            "physics" => 35,
            "maths" => 30,
            "chemistry" => 39
          ),

"samiksha" => array (
            "physics" => 30,
            "maths" => 32,
            "chemistry" => 29
          ),

"shubhra" => array (
            "physics" => 31,
            "maths" => 22,
            "chemistry" => 39
          )
        );
```

5.1.4 Files and Directories

PHP has a file system which allows the user to create file, read contents of the file line by line, character by character, append the contents in the file as well as delete and close the file.

We will see following functions used for handling the files

1. fopen() : This function is used to open the file.

Example

```
<?php
$fileop = fopen("c:\\Shubhra\\myfile.txt", "r");
?>
```

Here "r" stands for file open mode .

Following table will show various modes of file opening

Sr. No.	Mode	Description
1	r	Opens file in **read-only** mode. It places the file pointer at the beginning of the file.
2	r+	Opens file in **read-write** mode. It places the file pointer at the beginning of the file.
3	w	Opens file in **write-only** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
4	w+	Opens file in **read-write** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
5	a	Opens file in **write-only** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
6	a+	Opens file in **read-write** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
7	x	Creates and opens file in **write-only** mode. It places the file pointer at the beginning of the file. If file is found, fopen() function returns FALSE.
8	x+	It is same as x but it creates and opens file in **read-write** mode.

9	c	Opens file in **write-only** mode. If the file does not exist, it is created. If it exists, it is neither truncated (as opposed to 'w'), nor the call to this function fails (as is the case withx'). The file pointer is positioned on the beginning of the file
10	c+	It is same as c but it opens file in **read-write** mode.

2. fclose () : This function is used to close the open file

Example

```
<?php
fclose($fileop);
?>
```

4. fread() : This function is used to read the contents of the file.This function comes with two parameters such as resource and file size.

Example

```
<?php
$filename="c:\\Shubhra\\myfile.txt";
$fileop = fopen($filename,"r");
$contents= fread($fileop, filesize($filename));
echo $contents;
fclose($fileop);
?>
```

5. fwrite() : This function is used to write the contents into the file.

Example

```
<?php
$fileop = fopen("c:\\Shubhra\\myfile.txt", "w");//open file in write mode
fwrite($fileop, 'hello ');
fwrite($fileop, 'php file');
fclose($fileop);
echo "File written successfully";
?>
```

6. unlink() : This function is used to delete the file.

Example

```
<?php
unlink("c:\\Shubhra\\myfile.txt");


echo "File deleted successfully";
?>
```

5.1.5 Working with Database

As we have already learnt that PHP helps us to create dynamic web pages. With PHP e can connect and manipulate with various databases. However MySQL is the most popular database which runs on the server.

To interact with MySQL database following functions are useful.

Sr. No	Function	Description
1	Mysqli_connect(hostname, user,password)	Connect the the server using userid and password
2	Mysqli_connect_error()	Returns and ID of error
3	Mysqli_query(conectionsring, querystring)	Allows the query string to be consider as mysql query
4	Mysqli_select_db(database name, connection string)	Activates a MySQL database on the server
5	Mysqli_fetch_array(resulted[result type])	Fetches the result set as an associative array

Following examples will show how we can interact with database

Prerequisites : There exists a table named users in database login with few records in it

Database Name : Login

Table Name : Users

srno	Name	UID	PASS
1	rudra	rudra@gmail.com	12345
2	samar	samar@gmail.com	samar@123
3	Spruha	Spruha@gmail.com	spruha@234
4	Omkar	omkar@gmail.com	87655
5	geeta	geeta@gmail.com	geeta@789

Figure Records of users table

Example : Considering the above data Write a PHP program to design a page for authenticating the user.

Code:

```
<html>
<body>
<form name="f1" method="POST">
<table border="1">
<tr><th>Enter user name</th>
<td><input type="text" name="u"></td></tr>
<tr><th>Enter password</th>
<td><input type="text" name="p"></td></tr>
</table>
</form>
<?php
 $userid=$_POST['u'];
$password=$_POST['p'];
$con=mysqli_connect("localhost","root","");
mysqli_select_db("login",$con);
$query="Select * from users where UID= '$u' and PASS= '$p';
$result=mysqli_query($query,$con);
$count=mysql_num_rows($result);
```

```
if($count==0)
echo "invalid user"  ;
else
echo "you have logged in successfully";
mysql_close($con);
?>
</body>
</html>
```

5.1.6 Cookies, Sessions, and Headers

PHP headers : header() sends a raw HTTP header. This header() must always be called before actual output is sent either by normal HTML tags, blank lines in a file, or from PHP. It is a very common error to read code with include, or require, functions, or another file access function, and have spaces or empty lines that are output before header() is called. The same problem exists when using a single PHP/HTML file.

Syntax

header(header, replace, http_response_code)

Sr.No.	Parameter	Description
1	Header	It is a required parameter Specifies the header string to send
2	Replace	It is a optional parameter. It indicates whether a header should replace a previous similar header or adds a new header of the same type. Default value is true and it will replace the header. However false value allows multiple headers of same type
3	http_response_code	Its an optional parameter. It forces the HTTP response code to the specified value.

PHP cookies :

Cookies are the text files stored on client computer and helps in tracking.

Following are the stepts through which users can be identified

1. Serve sends a chunk of data in the form of set of cookies to the browser such as name, age etc.
2. Browser stores the information on client computer for future use
3. When next time request is sent by the browser to the web server, browser sends those cookies information to the server and server uses that information for identifying the user.

PHP allows to create cookie using the function setcookie(). This function has following arguments and they must be called before html tag

Sr.No.	Argument	Description
1	Name	This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
2	Value	This sets the value of the named variable and is the content that you actually want to store.
3	Expiry	This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
4	Path	This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
5	Domain	This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
6	Security	This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Following example sets the cookie for name and age and those cookies are expired after one hour.

```
<?php
setcookie("name","samiksha", time()+3600,"/","",0);
setcookie("age","36", time()+3600,"/","",0);
?>
<html>

<head>
<title>Setting Cookies with PHP</title>
</head>

<body>
<?php echo "Set Cookies"?>
</body>

</html>
```

isset() function is used to check whether cookies are set or not. Following program will show how to check whether cookie is set or not

```
<html>
<head>
<title>Accessing Cookies with PHP</title>
</head>
<body>
<?php
if(isset($_COOKIE["name"]))
echo"Welcome ". $_COOKIE["name"]."<br />";

else
echo"cookie not found"."<br />";
?>
</body>
</html>
```

PHP sessions :

Allows to access the variable values across various pages of the website. It creates a file in temporary directory of the server where registered session variables and their values are stored.The location of the temporary file is determined by a setting in the php.ini file called session.save_path. Before using any session variable make sure you have setup this path.

Session_start() function is used to create the sessions. It is always written in the beginning of the page. Whereas isset() function is used to check whether session variables are set or not.

Session_destroy() function is used to destroy the session. Unset() function is used to unset the session variables.

Following example will show creation of session variables.

```
<?php
session_start();
if(isset( $_SESSION['counter'])){
    $_SESSION['counter']+=1;
}else{
    $_SESSION['counter']=1;
}
  $msg="You have visited this page ".  $_SESSION['counter'];
  $msg.="in this session.";
?>
<html>
<head>
<title>Setting up a PHP session</title>
</head>
<body>
<?php  echo ( $msg);?>
</body>
</html>
```

Summary :

php is acronym for Hypertext Preprocessor. It is a serverside scripting laguage. The syntax of language is similar to c language and hence easy to understand.

It is used to create dunamic web pages.

various databases can be used in the backend of the language.

PHP is also having its file system used to create files, read contents, append the contents and so on

PHP session allows to carry the values of the variable across the pages of the website

Cookies stores chunk of data for future use.

Practice Questions

1. Write features of PHP
2. Explain PHP datatypes
3. Explain various operators supported bu PHP
4. Write a note on Sessions
5. Write a note on Cookies
6. Explain various file handling modes in PHP
7. Explain MySQL functions used while handling the database

References:

https://www.javatpoint.com/php-tutorial

https://www.w3schools.com/php/default.asp

❖❖❖❖

6

INTRODUCTIONS TO JQUERY

Unit Structure

6.1 INTRODUCTION TO JQUERY

jQuery is a small, lightweight , fast, concise and a coss platform JavaScript Library created by John Resig in 2006. The main intention behind creating this library was "Write less, do more". jQuery simplifies HTML document traversing, event handling, animating, and Ajax calls and Document Object Model manipulations. jQuery is acting as a JavaScript toolkit designed to simplify various tasks by writing less code.

Features of jQuery

Figure Features of jQuery

1. Document Object Manipulation : Due to jQuery it is easy to select DOM elements and manipulate their contents by using cross-browser open source selector engine called Sizzle.
2. Event Handling : jQuery allows to capture various events such as a user clicking on the link and there is no need to clutter the HTML code with because of event handlers.
3. Ajax Support: jQuery also helps to develop responsive websites with the help of Ajax technology
4. Animation: jQuery have its own number of animations effects which can be used on website to make them more eye catchy
5. Lightweight : It is a very lightweight and hence fast library
6. Cross browser support : jQuery has a cross browser support and it works with number of browsers.

Methods to add jQuery

There are two methods to add jQuery in a website

1. Downloading the jQuery library from jQuery.com : While downloading there are two different versions are available ie. Production version which is a compressed one and used for live websites and the another version is Development version which is used for testing and developmet as it is uncompressed and helps us to read the code.

When we download jquery-2.1.3.min.js file from https://jquery.com/download/ link and store it in a directory of a website we will be able to jquery library in HTML. Following coderefelects how jquery library is included in HTML

```
<html>

<head>

<title>The jQuery Example</title>

<script type="text/javascript"

src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<script type="text/javascript">

 $(document).ready(function(){

document.write("Hello, We have included jquery library in html through local installation!");

 });
```

```
</script>
</head>
<body>
<h1>Hello</h1>
</body>
</html>
```

2. Including jQuery from a CDN (Content Delivery Network) : jQuery library can be included directly into HTML code from Content Delivery Network. Google and Microsoft provides latest versions for such type of content delivery

The following code displays usage of jQuery library from Google CDN

```
<html>
<head>
<title>The jQuery Example</title>
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
<script type="text/javascript">
 $(document).ready(function(){
document.write("Hello, we have used jQuery library using Google CDN!");
 });
</script>
</head>
<body>
<h1>Hello</h1>
</body>
</html>
```

6.1.1 Fundamentals

The jQuery library enables to manipulate an HTML page after it is displayed by the browser. It also allows us to use various animation effects and develops interactive web pages. It also provides tools for communication with a server without reloading the page. To do all those things . First, we should be aware of jQuery basics, and at how we can use jQuery to perform its core functionality:

1. Meaning of $: The jQuery library provides jQuery () function which allows us to select the elements using CSS selectors. We can do this by writing

varlistItems = jQuery('li');however the function jQuery() can be shortened and can be written as varlistItems = $('li');When we call $() function and pass a selector to it, it creates a new jQuery object. In javascript, functions are acting as objects also so they also have methods and properties as well .

2. $(document).ready() : Before using jQuery on our page we should make ensure that page is in ready state for being get manipulated. $(document).ready() accomplish this by putting the code in this function

3. Other ways to create jQuery Object : In addition to passing simple select $() there are some other ways to create jQuery object

Sr.No.	Way to create a jQuery object	Code
1	Creating jQuery object from DOM element	$(document.body.children[0]);
2	creating a jQuery object from a list of DOM elements	$([window, document]);
3	making a selection in the context of a DOM element	varfirstBodyChild = document.body.children[0]; $('li', firstBodyChild)
4	making a selection within a previous selection	var paragraph = $('p'); $('a', paragraph);

4. Creating new elements: When we pass HTML code snippet to $() it creates a new element in memory. That means the element will get created but it will not be placed on the page until we place it on the page. Following are some ways how we can create elements

Sr.No.	Way to create a jQuery Element	Code
1	creating a new element with no content	$(' ');
2	creating a new element with content	$(' Hello! ');
3	creates a new with content and class	$(' Hello!</ p>');
4	passing an object with information and creating an object	$(' ', { html: 'Hello!', 'class': 'greet' });

6.1.2 Selectors :

jQuery selector is a function which makes use of expressions and finds out matching elements from a Document Object Model. Following are some criteria mentioned in the table. These items can be used alone or with combination of other selectors. All jQuery selectors start with a dollor sign and parenthesis e.g. $(). It is known as the factory function.

Sr. No	Seelector	Description
1	Tag Name	Represents a tag name available in DOM e.g. $(‘p’) selects all paragraphs in tag in the document
2	Tag ID	Shows a tag available with the given ID in the DOM. For example $('#someid') selects the single element in the document that has an ID of some-id.
3	Tag Class	Shows a tag available with the given class in the DOM. For example $('.some-class') selects all theelements in the document that have a class of someclass.

Following example will illustrate the use of tag selector where all the paragraphs are set with background color teal

```
<html>
<head>
 <title>First jQuery Example</title>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
 </script>
 <script type="text/javascript" language="javascript">
 $(document).ready(function() {
 $("p").css("background-color", "teal");
 });
 </script>
 </head>
<body>
<p>This is first paragraph.</p>
<p>This is second paragraph.</p>
<p>This is third paragraph.</p>
</body>
</html>
```

Using Selectotrs :

Selectors are very useful and require at each step while making use of jQuery. They helps us to get the exact element which is needed from HTML document.

Sr.No	Selector	Description
1	Name	Selects all elements which match with the given element Name.
2	#ID	Selects a single element which matches with the given ID.
3	.class	Selects all elements which matches with the given Class.
4	Universal (*)	Selects all elements available in a DOM.
5	Multiple Elements X,Y,Z	Selects the combined results of all the specified selectors X,Y or Z

Following table gives information about different jQuery selectors with the help of examples

Sr.No.	Selector	Example	Description
1	*	$("*")	selects all elements.
2	#id	$("#firstname")	selects the element with id="firstname"
3	.class	$(".primary")	selects all elements with class="primary"
4	class,.class	$(".primary,.secondary")	selects all elements with the class "primary" or "secondary"
5	element	$("p")	selects all p elements.
6	el1,el2,el3	$("h1,div,p")	selects all h1, div, and p elements.
7	:first	$("p:first")	selects the first p element
8	:last	$("p:last")	selectsthe last p element
9	:even	$("tr:even")	selects all even tr elements
10	:odd	$("tr:odd")	selects all odd tr elements
11	:first-child	$("p:first-child")	Selects all p elements that are the first child of their parent
12	:first-of-type	$("p:first-of-type")	selects all p elements that are the first p element of their parent
13	:last-child	$("p:last-child")	selects all p elements that are the last child of their parent
14	:last-of-type	$("p:last-of-type")	selects all p elements that are the last p element of their parent
15	:nth-child(n)	$("p:nth-child(2)")	Selects all p elements that are the 2nd child of their parent
16	:nth-last-child(n)	$("p:nth-last-child(2)")	Selects all p elements that are the 2nd child of their parent, counting from the last child
17	:nth-of-type(n)	$("p:nth-of-type(2)")	Selects all p elements that are the 2nd p element of their parent

18	:nth-last-of-type(n)	$("p:nth-last-of-type(2)")	Selects all p elements that are the 2nd p element of their parent, counting from the last child
19	:only-child	$("p:only-child")	Selects all p elements that are the only child of their parent
20	:only-of-type	$("p:only-of-type")	Selects all p elements that are the only child, of its type, of their parent
21	parent > child	$("div > p")	Selects all p elements that are a direct child of a div element
22	parent descendant	$("div p")	Selects all p elements that are descendants of a div element
23	element + next	$("div + p")	It selects the p element that are next to each div elements
24	element ~ siblings	$("div ~ p")	It selects all p elements that are siblings of a div element
25	:eq(index)	$("ulli:eq(3)")	Selects the fourth element in a list (index starts at 0)
26	:gt(no)	$("ulli:gt(3)")	Select the list elements with an index greater than 3
27	:lt(no)	$("ulli:lt(3)")	Select the list elements with an index less than 3
28	:not(selector)	$("input:not(:empty)")	Select all input elements that are not empty
29	:header	$(":header")	Select all header elements h1, h2 ...
30	:animated	$(":animated")	Select all animated elements
31	:focus	$(":focus")	Select the element that currently has focus
32	:contains(text)	$(":contains('Hello')")	Select all elements which contains the text "Hello"
33	:has(selector)	$("div:has(p)")	Select all div elements that have a p element
34	:empty	$(":empty")	Select all elements that are empty

35	:parent	$(":parent")	Select all elements that are a parent of another element
36	:hidden	$("p:hidden")	Select all hidden p elements
37	:visible	$("table:visible")	Select all visible tables
38	:root	$(":root")	Selects the document's root element
39	:lang(language)	$("p:lang(de)")	Select all p elements with a lang attribute value starting with "de"
40	[attribute]	$("[href]")	Select all elements with a href attribute
41	[attribute=value]	$("[href='default.htm']")	Select all elements with a href attribute value equal to "default.htm"
42	[attribute!=value]	$("[href!='default.htm']")	Selects all elements with a href attribute value not equal to "default.htm"
43	[attribute$=value]	$("[href$='.jpg']")	Selects all elements with a href attribute value ending with ".jpg"
44	[attribute|=value]	$("[title|='Tomorrow']")	Select all elements with a title attribute value equal to 'Tomorrow', or starting with 'Tomorrow' followed by a hyphen
45	[attribute^=value]	$("[title^='Tom']")	Select all elements with a title attribute value starting with "Tom"
46	[attribute~=value]	$("[title~='hello']")	Select all elements with a title attribute value containing the specific word "hello"
47	[attribute*=value]	$("[title*='hello']")	Select all elements with a title attribute value containing the word "hello"
48	:input	$(":input")	Selects all input elements
49	:text	$(":text")	Selects all input elements with type="text"

50	:password	$(":password")	Selects all input elements with type="password"
51	:radio	$(":radio")	Selects all input elements with type="radio"
52	:checkbox	$(":checkbox")	Itwill select all input elements with type="checkbox"
53	:submit	$(":submit")	Selects all input elements with type="submit"
54	:reset	$(":reset")	Selects all input elements with type="reset"
55	:button	$(":button")	Selects all input elements with type="button"
56	:image	$(":image")	Selects all input elements with type="image"
57	:file	$(":file")	Selects all input elements with type="file"
58	:enabled	$(":enabled")	Selects all enabled input elements
59	:disabled	$(":disabled")	Selects all disabled input elements
60	:selected	$(":selected")	Selects all selected input elements
61	:checked	$(":checked")	Selects all checked input elements

6.1.3 Methods to access HTML attributes

The following table displays all the methods which can be used to manipulate HTML documents

Sr.No	Method	Description
1	addClass()	Adds one or more than one class names to selected elements
2	after()	Inserts the content after selected elements
3	append()	Inserts the content at the end of selected elements

4	appendTo()	Inserts the HTML elements at the end of selected elements
5	attr()	Sets or returns attributes/values of selected elements
6	before()	Inserts the content before selected elements
7	clone()	Makes a copy of selected elements
8	css()	Sets or returns one or more style properties for selected elements
9	detach()	Removes selected elements (keeps data and events)
10	empty()	Removes all child nodes and content from selected elements
11	hasClass()	Checks if any of the selected elements have a specified class name
12	height()	Sets or returns the height of selected elements
13	html()	Sets or returns the content of selected elements
14	innerHeight()	Returns the height of an element (includes padding, but not border)
15	innerWidth()	Returns the width of an element (includes padding, but not border)
16	insertAfter()	Inserts HTML elements after selected elements
17	insertBefore()	Inserts HTML elements before selected elements
18	offset()	Sets or returns the offset coordinates for selected elements (relative to the document)
19	offsetParent()	Returns the first positioned parent element
20	outerHeight()	Returns the height of an element (includes padding and border)
21	outerWidth()	Returns the width of an element (includes padding and border)
22	position()	Returns the position (relative to the parent element) of an element

23	prepend()	Inserts content at the beginning of selected elements
24	prependTo()	Inserts HTML elements at the beginning of selected elements
25	prop()	Sets or returns properties/values of selected elements
26	remove()	Removes the selected elements (including data and events)
27	removeAttr()	Removes one or more attributes from selected elements
28	removeClass()	Removes one or more classes from selected elements
29	removeProp()	Removes a property set by the prop() method
30	replaceAll()	Replaces selected elements with new HTML elements
31	replaceWith()	Replaces selected elements with new content
32	scrollLeft()	Sets or returns the horizontal scrollbar position of selected elements
33	scrollTop()	Sets or returns the vertical scrollbar position of selected elements
34	text()	Sets or returns the text content of selected elements
35	toggleClass()	Toggles between adding/removing one or more classes from selected elements
36	unwrap()	Removes the parent element of the selected elements
37	val()	Sets or returns the value attribute of the selected elements (for form elements)
38	width()	Sets or returns the width of selected elements
39	wrap()	Wraps HTML element(s) around each selected element
40	wrapAll()	Wraps HTML element(s) around all selected elements
41	wrapInner()	Wraps HTML element(s) around the content of each selected element

6.1.4 Traversing Methods

jQueryTraversing means moving through and is used to find or select the HTML elements on the basis of their relation with the other elements. It starts with one selection and move through that selection only until it reaches to its destination . The following image focuses HTML page as a tree structure. With jQuery traversing one can easily move up, down and sideway directions in the tree

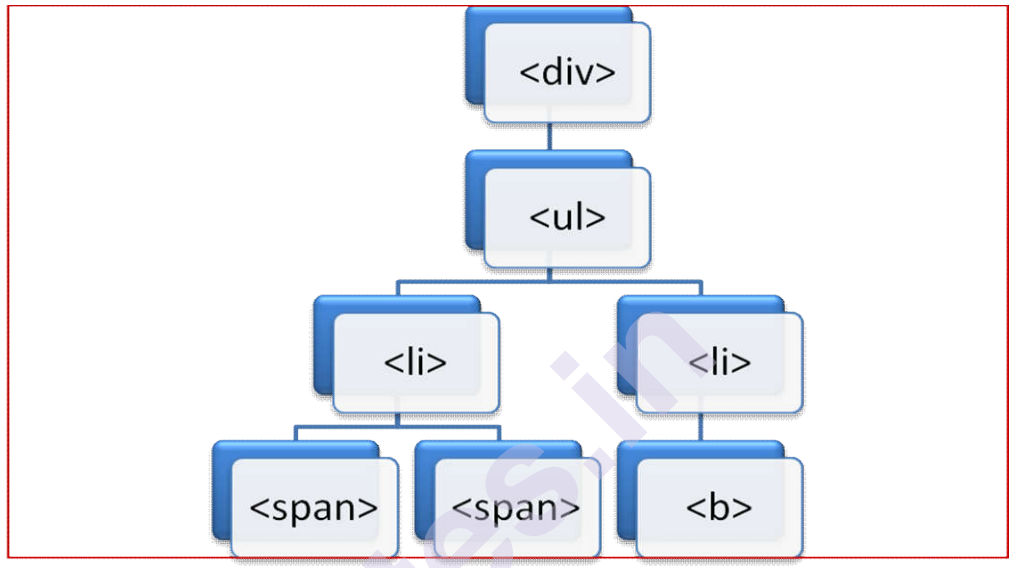


Figure HTML Page DOM Tree

The above figure illustrates the following relationship

1. is ancestor of all
2. <u> is descendant of and parent of both the
3. Left side <li> is child of <ul> and parent of both the as well as descendant of and
4. Right side <li> is child of <ul> and parent of both the <b>as well as descendant of and
5. Both the are siblings of each other as well as descendant of left <li>, and
6. <b> is descendant of right <li>, and

6.1.4.1 jQuery Traversing up to the Ancestors methods

Following are the useful jQuery methods for traversing upward directions to the ancestors

1. parent() : It displays the direct parent of the element.

Eg :

```
$(document).ready(function(){
  $("span").parent();
});
```

Here parent of is shown

2. parents() : This method returns all the ancestors up to the root element

Eg:

```
$(document).ready(function(){
  $("span").parents();
});
```

Here all parent, grandparent as well as great grand parents of are shown

3. parentsUntil() : This method returns all ancestors between given two arguments

Eg:

```
$(document).ready(function(){
                                    $("span").parentsUntil("div");
});
```

Here all parents between and are shown

6.1.4.2 jQuery Traversing down to the Descendants methods:

These methods are used to traverse in the downward direction of the DOM tree

Following are useful methods

1. children() : Displays all the direct descendants of the selected element

2.

Eg :

```
$(document).ready(function(){
                                                $("div").children();
});
```

This will display descendants of

3. find() : Finds descendant of a selected element upto the last descendant

Eg:

```
$(document).ready(function(){
  $("div").find("span");
});
```

This will display all the descendants of

6.1.4.3 jQuery Traversing sideways to the Siblings methods :

These methods traverse sideway across the DOM tree. Following table illustrates the methods used moving sideways towards the siblings

Sr. No	Method	Description
1	Siblings()	Displays all the sibling elements of the selected element
2	next()	Displays next sibling element of the selected element.
3	nextAll()	Displays all next sibling elements of the selected element
4	nextUntil()	Displays all next sibling elements between two given arguments.
5	prev()	Displays previous sibling element of the selected element.
6	prevAll()	Displays all previous sibling elements of the selected element
7	prevUntil()	Displays all previous sibling elements between two given arguments.

6.1.4.4 jQuery Traversing - Filtering :

Filtering traverse allows to select specific element on the basis of its position in a group of elements and on the basis of matching criteria.

Sr.No	Method	Description
1	Firs()	Displays the first element of the specified elements.
2	Last()	Displays the last element of the specified elements.
3	Eq()	Displays an element with a specific index number of the selected elements
4	Filter()	Elements that do not match the specified criteria are removed and those that match will be returned.
5	Not()	Displays all elements that do not match the criteria.

Following table shows all the Filtering traverse methods

6.1.5 Manipulators, Events, Effects

6.1.5.1 Manipulators :

jQuery provides various methods for DOM manipulation, Attributes Manipulation and Dimension manipulation

1. DOM Manipulation Methods

Sr. No	Method	Description
1	append()	Adds the content to the end of elementprovided by a selector.
2	before()	Adds content (new or existing DOM elements) before an elementprovided by a selector.
3	after()	Adds content (new or existing DOM elements) after an elementprovided by a selector.
4	prepend()	Adds content at the beginning of an elementprovided by a selector.
5	remove()	Removes element(s) from DOM which is provided by selector.
6	replaceAll()	Replaces target element with the specific element.
7	wrap()	Wraps an HTML structure around each element which is provided by selector.

2. Attributes Manipulation Methods

Sr. No	Method	Description
1	attr()	Get or set the value of provided attribute of the target element
2	prop()	Get or set the value of provided property of the target element
3	html()	Get or set html content to the provided target element
4	text()	Get or set text for the provided target element
5	val()	Get or set value property of the provided target element.

3. Dimension Manipulation Methods

S. No.	Method	Description
1	height()	Get or set height of the provided element
2	innerHeight()	Get or set inner height (padding + element's height) of the provided element
3	outerHeight()	Get or set outer height (border + padding + element's height) of the provided element
4	offset()	Get or set left and top coordinates of the provided element
5	position()	Get the current coordinates of the provided element
6	width()	Get or set the width of the provided element
7	innerWidth()	Get or set the inner width (padding + element's width) of the provided element
8	outerWidth()	Get or set outer width (border + padding + element's width) of the provided element

6.1.5.2 Events :

In many of the Web Applications to perform a specific operation the user has to trigger some action. So this triggered action is called as event. jQuery provides methods to handle Dom Events. They are categorized into Form Events, Keyboard Events, Mouse Events, Browser Events and Document Load Events

1. Form Events

Sr.No.	Jquery Method	Event
1	Blur	onblur
2	Change	onchange
3	Focus	onfocus
4	Focusin	onfocusin
5	Select	onselect
6	Submit	onsubmit

2. Keyboard Events

Sr.No.	Jquery Method	Event
1	keydown	onkeydown
2	keypress	onkeypress
3	Keyup	onkeyup
4	Focusout	

3. Mouse Events

Sr.No.	Jquery Method	Event
1	Click	onclick
2	Dblclick	ondblclick
3	Mousedown	onmousedown
4	Mouseenter	onmouseenter
5	Mouseleave	onmouseleave
6	Mousemove	onmousemove
7	Mouseout	onmouseout
8	Mouseover	onmouseover
9	mouseup	onmouseup
10	toggle	
11	focusout	
12	Hover	

4. Browser Events

Sr. No	jQuery Method	Event
1	error	onerror
2	resize	onresize
3	Scroll	onscroll

5. Document Load Events

Sr. No.	jQuery Method	Event
1	load	onload
2	unload	onunload
3	Ready	Onready

6.1.5.3 Effects:

jQuery allows to add effects such as fading, sliding, hiding showing and animation on the web page to make it more attractive.

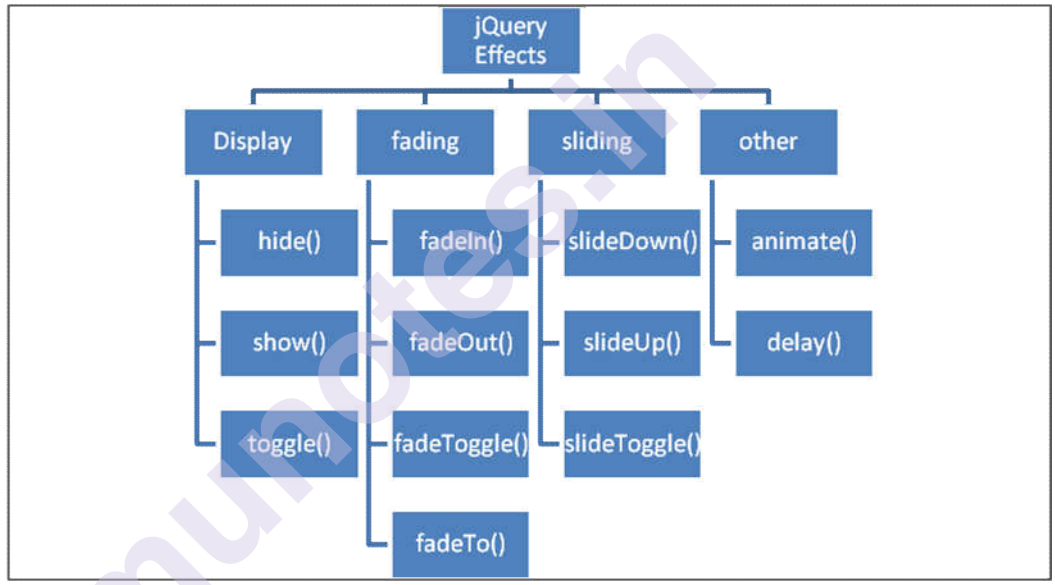


Figure jQuery Effects

Following table shows the complete list of jQuery effect methods

Sr. No.	Method	Description
1	hide()	Allows to hide the matched or selected elements.
2	show()	Reflects or shows the selected elements.
3	toggle()	shows or hides the matched elements. In other words, it toggles between the hide() and show() methods.
4	fadein()	Reflects the matched elements by fading it to opaque. In other words, it fades in the selected elements.

5	fadeout()	Reflects the matched elements by fading it to transparent. In other words, it fades out the selected elements.
6	fadetoggle()	Reflects or hides the matched element. means, toggles between the fadeIn() and fadeOut() methods.
7	fadeto()	Allows to adjust opacity for the matched element. In other words, it fades in/out the selected elements.
8	slidedown()	Projects the matched elements with slide.
9	slideup()	hides the matched elements with slide.
10	slidetoggle()	Reflects or hides the matched elements with slide. In other words, it is used to toggle between the slideUp() and slideDown() methods.
11	animate()	performs animation.
12	delay()	sets delay execution for all the queued functions on the selected elements.

Summary :

jQuery is a javascript library created with the intention Write less and do more. It is a toolkit which simplifies the work like document traversing, event handling, animtion, making ajax calls etc.

jQuery can be added by locally downloading or by including it from CDN.

$() is use synonymously for jQuery()

selectors are the functions used to find matching elements from DOM with the help of expressions

jQuery traversing finds the HTML elements based on their relations with other elements. It can travel into upward, downwar and side by side directions

jQuery also works for data manipulation, triggering events and animations

Practice Questins

1. Explain the characteristics of jQuery

2.How jQuery objec is created explain

3. What are selectors? explain how to use selectors

4. List down the methods used to acces HTML attributes

5. Explain traversing with the help of its different methods

6. Exaplain what is manipulation with its methods

7. Explain Events with the help of its methods

Referemces

http://jqfundamentals.com/chapter/jquery-basics

https://www.tutorialsteacher.com/jquery/jquery-event