Unit – I
Hypertext Markup Language (HTML)

Basics:

What Is the Internet?

The **Internet** is a worldwide collection of networks that links millions of businesses, government offices, educational institutions, and individuals. The Internet, as it is known today, was born in 1983 when **ARPANET** was split into two interconnected networks: ARPANET and MILNET. Data is transferred over the Internet using **servers** which are computers that manage network resources and provide centralized storage areas, and **clients**, which are computers that can access the contents of the storage areas.

Each computer or device on a communications line has a numeric address called an **IP** (Internet protocol) **address**, the text version of which is called a **domain name**. Every time you specify a domain name, a **DNS** (**domain name system**) **server** translates the domain name into its associated IP address, so data can route to the correct computer.

Internet services are:

- **Electronic mail (email**
- **Newsgroups**
- **Internet Relay Chat (IRC**
- **RIA, WOA and Social Web**
- **File Transfer Protocol (FTP and FTPS, SFTP**
- **World Wide Web (www)**

World Wide Web (WWW):

The World Wide Web is a set of programs, standards and protocols that allow the text, images, animations, sounds and videos to be stored,

accessed and linked together in form of web sites. The World Wide Web is a way of exchanging information between computers on the Internet, tying them together into a vast collection of interactive multimedia resources. It has a unique combination of flexibility, portability and user-friendly features that distinguish it from other services provided by the Internet. The main reason for its popularity is the use of a concept called hypertext. It uses the client-server model, and an Internet protocol called hypertext transport protocol (HTTP) for interaction between the computers on the Internet. An English scientist *Tim Berners-Lee* invented the **World Wide Web** in 1989 at **CERN** in **Geneva.**

URL

URL stands for **U**niform **R**esource **L**ocator, and is used to specify addresses on the World Wide Web. A URL is the fundamental network identification for any resource connected to the web (e.g., hypertext pages, images, and sound files). It is a reference (an address) to a resource on the Internet.

A URL will have the following format −

protocol://domain name:port/path

A URL has two main components:

- Protocol Identifier: The protocol specifies how information is transferred from a link. The protocol used for web resources is Hyper Text Transfer Protocol (HTTP).
- Domain name: The protocol is followed by a colon, two slashes, and then the resource name-domain name. The domain name is the complete address to the resource. The domain name is the computer on which the resource is located.

Ex: **http://google.com - here** the protocol identifier is http, the domain name is google.com

Internet Protocols: The different protocols like HTTP, TCP/IP, FTP, Telnet etc. are as follows

- **HTTP- Hyper Text Transfer Protocol:** HTTP is for accessing and transmitting World Wide Web documents. HTTP takes care of the communication between a web server and a web browser.
- **TCP – Transmission Control Protocol:** TCP is a **connection oriented** protocol and offers end-to-end packet delivery. It acts as back bone for connection. It handles communication between applications
- **IP – Internet Protocol:** Internet Protocol is **connectionless** and **unreliable** protocol. IP handles communication between computers. IP takes care of the communication with other computers.
- **FTP (File Transfer Protocol):** FTP creates two processes such as Control Process and Data Transfer Process at both ends i.e. at client as well as at server. FTP establishes two different connections: one is for data transfer and other is for control information.

Web Browser: When two computers communicate over some network, in many cases one acts as a client and the other as a server. The client initiates the communication, which is often a request for information stored on the server, which then sends that information back to the client. The Web, as well as many other systems, operates in this client-server configuration. Here the client can make use of web browser application to communicate.

Web browser is an application that provides a way to look at and interact with the information on the World Wide Web. It retrieves, presents, and traverses information resources. These include web pages, images, video, and other multimedia content.

Ex: Internet explorer, Mozilla Firefox, Opera, google chrome.

Web Server:

- A **web server** is a computer system that processes requests via HTTP to serve the files that form web pages to users, in response to their requests.
- The term can refer to the entire system, or specifically to the software that accepts and supervises the HTTP requests.
- The primary function of a web server is to store, process and deliver web pages to clients.
- When client sends request for a web page, the web server search for the requested page if requested page is found then it will send it to client with an HTTP response. If the requested web page is not found, web server will the send an HTTP response: Error 404 Not found.
- The most commonly used Web servers are Apache, which has been implemented for a variety of computer platforms, and Microsoft's Internet Information Server (IIS), which runs under Windows operating systems and XAMPP

Internet Service Providers

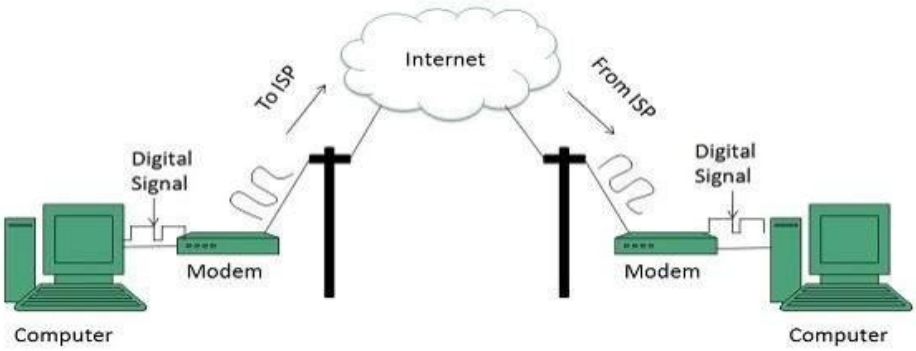
An **Internet Service Provider** (**ISP**) is a company provides internet access to individuals and businesses companies, families, and even mobile users for monthly or yearly fees. The type of Internet access varies depending on what the customer requires. In addition to internet connection, ISPs may also provide related services like web site hosting and development, email hosting, domain name registration etc. ISPs use fiber-optics, satellite, copper wire, and other forms to provide Internet access to its customers. Factors to consider while choosing ISP: Bandwidth (speed), Cost (setup and service fee), Availability (reach), Reliability (down time), Convenience (mobility) etc.

Examples of ISP's are AT&T, Verizon, Infocom, Telecom, UTL, MTN, Airtel, VSNL etc.

Types of internet access

Most ISPs offer several types of internet access which essentially differ in connection speed **–** the time taken for download and upload. There exist several ways to connect to the internet. Following are the connection types available:

1. **Dial-up Connection**- connection is probably the slowest connection and requires you to connect to the internet via your phone line by dialing a number specified by the ISP. It requires a modem to setup dial-up connection,
2. **ISDN** - **ISDN** is acronym of **Integrated Services Digital Network.** It establishes the connection using the phone lines which carry digital signals instead of analog signals.
3. **DSL** - is acronym of **Digital Subscriber Line.** It is a form of broadband connection as it provides connection over ordinary telephone lines. *It* is indeed very fast and these ISPs can offer different download speeds – quicker the speed, higher will be the price.
4. **Cable TV Internet connections** – this connection is provided through Cable TV lines. A cable modem is used to access this service, provided by the cable operator
5. **Satellite Internet connections** - Satellite Internet connection offers high speed connection to the internet. We need a dialup access through ISP over telephone line.
6. **Wireless Internet Connections**- Wireless Internet Connection makes use of radio frequency bands to connect to the internet and offers a very high speed. The wireless internet connection can be obtained by either Wi-Fi or Bluetooth.



URL

URL stands for **U**niform **R**esource **L**ocator, and is used to specify addresses on the World Wide Web. A URL is the fundamental network identification for any resource connected to the web (e.g., hypertext pages, images, and sound files). It is a reference (an address) to a resource on the Internet.

A URL will have the following format −

protocol://domain name:port/path

A URL has two main components:

- Protocol Identifier: The protocol specifies how information is transferred from a link. The protocol used for web resources is Hyper Text Transfer Protocol (HTTP). Other protocols compatible with most web browsers include FTP, telnet, newsgroups, and Gopher.
- Domain name: The protocol is followed by a colon, two slashes, and then the resource name-domain name. The domain name is the complete address to the resource. The domain name is the computer on which the resource is located. Links to particular files or subdirectories may be further specified after the domain name. The directory names are separated by single forward slashes.

Ex: **http://google.com** **-** her**e** the protocol identifier is http, the domain name is google.com

Domain Name:

To identify a computer resource on internet, each computer is assigned with an IP address. It is a series of numbers that identify a particular computer on the internet. A typical IP address looks like this: 66.249.66.1. Now an IP address like this is quite difficult to remember. Domain names were invented to solve this problem.

A domain name is a unique name that identifies a website. A domain name is the part of your Internet address of the website that comes after "www". These names begin with -

- The name of the host machine, followed by progressively larger enclosing collections of machines, called *domains*.
- There may be two or more domain names.
- The first domain name which appears immediately to the right of the *host name* is the domain of which the host is a part.
- All domain names have a domain suffix, such as .com, .net, or .org. The domain suffix helps identify the type of website the domain name represents. For example,

.com − Stands for company/commercial, but it can be used for any website.

.net − Stands for network and is usually used for a network of sites.

.org − Stands for organization and is supposed to be for non-profit bodies.

.us, .in − they are based on your country names so that you can go for country specific domain extensions

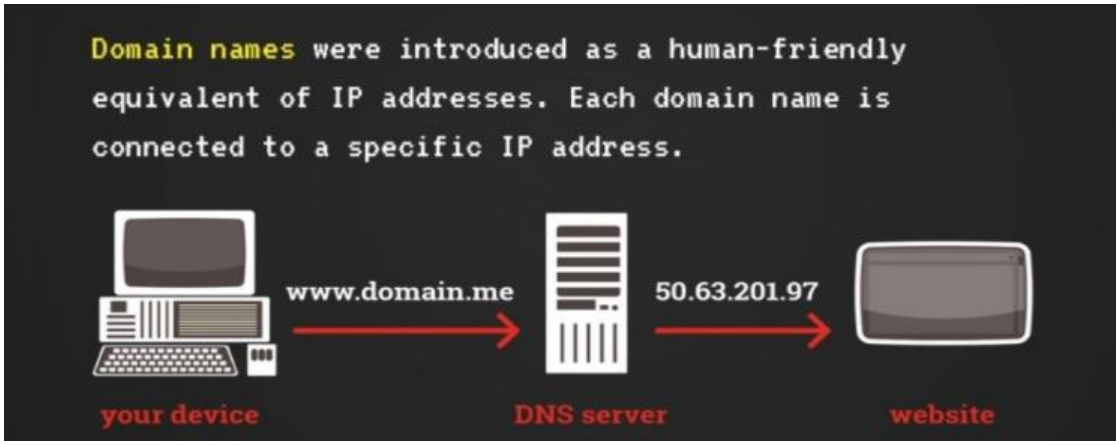
.info − Stands for information. This domain name extension can be very useful, and as a new comer it's doing well.

.edu/ac - educational/academic

.mil – military sites

.int – international organizations

.net – network providers



Ex: https://**techterms.com**/ here techterms.com is a domain name.

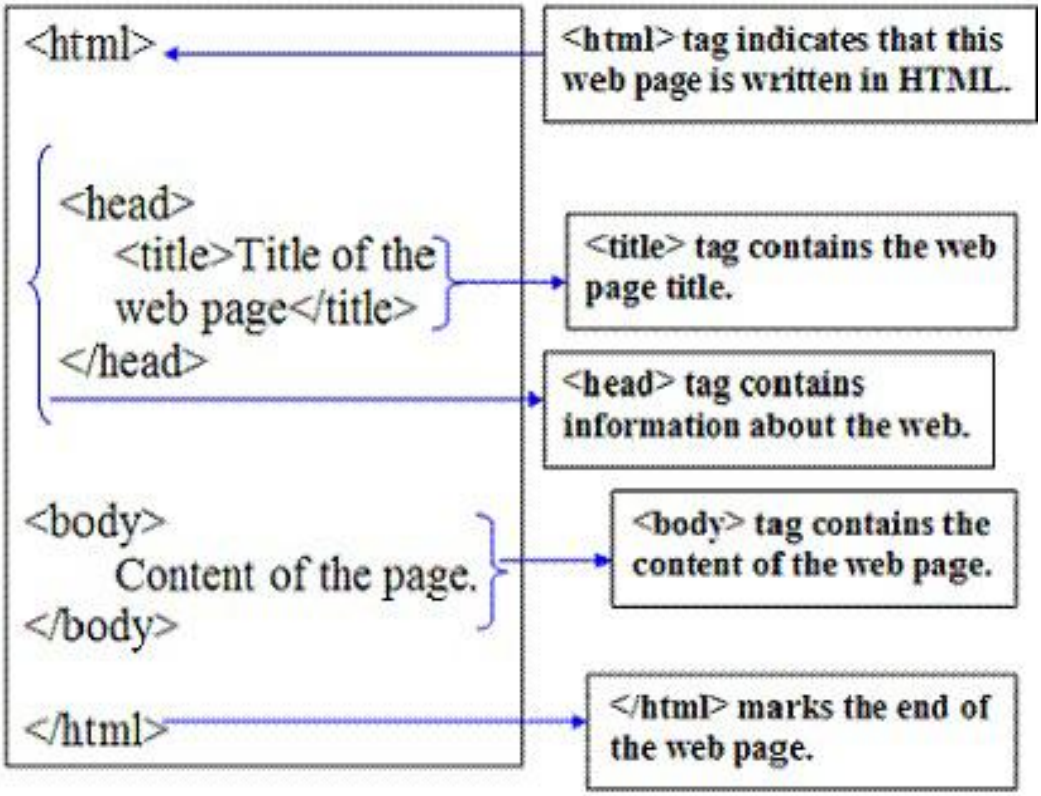
What is HTML?

- **HTML** stands for **Hyper Text Markup Language**, which is the most widely used language on Web to develop **web pages**.
- HTML was created by **Tim Berners-Lee** in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995.
- HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but currently we are having HTML-5 version which is an extension to HTML 4.01, and this version was published in 2012.
- It is a markup language and is a set of markup **tags.** The tags **describe** document content.
- HTML documents contain HTML **tags** and plain **text,** HTML documents are also called **web pages**

HTML Tags

- HTML markup tags are usually called HTML tags.
- HTML tags are keywords (tag names) surrounded by **angle brackets** like **<html>**
- HTML tags normally **come in pairs** like **<html**> and **</html>.**
- The first tag in a pair is the **start tag,** the second tag is the **end tag.** The end tag is written like the start tag, with a **slash** before the tag name.
- Start and end tags are also called **opening tags** and **closing tags.**
- In HTML, tags are paired tags and unpaired tags. Paired tags have opening tag and ending tag. Unpaired tags doesn't have end tag.
- HTML tags are not case sensitive, <b> means the same as <B>.
- In HTML there are both logical tags and physical tags. Logical tags are designed to describe (to the browser) the enclosed text's meaning. Physical tags on the other hand provide specific instructions on how to display the text they enclose.

HTML Page Structure: The following is the basic structure of HTML document.



An HTML document has two main parts:

1. **head:** the head element contains **title** of the web page - meta data of a web document.
2. **body**: The body element contains the information that you want to display on a web page.

In a web page, the first tag **<html>** indicates the markup language that is being used for the document. The **<head>** tag contains information about the web page. Lastly, the content appears in the **<body>** tag. The following illustration provides a summary.

Steps to create your first web page with Notepad?

Step 1: Open Notepad

To open Notepad in Windows 7 or earlier: Click **Start** (bottom left on your screen). Click **All Programs**. Click **Accessories**. Click **Notepad**.

To open Notepad in Windows 8 or later: Open the **Start Screen** (the

window symbol at the bottom left on your screen). Type **Notepad**

Step 2: Write Some HTML code

```
<! DOCTYPE html>
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Step 3: Save the HTML Page

Save the file on your computer.
Select **File Save as** in the Notepad menu.
When saving an HTML file, use either the .htm or the .html file extension.

Step 4: Execute HTML file / View HTML Page in Your Browser

Double-click your saved HTML file or right click on html file and open with any web browser.

Tag Attributes:

Tags can have attributes. Attributes can provide additional information about the HTML elements on your page. The <tag> tells the browser to do something, while the attribute tells the browser how to do it. For instance, if we add the **bgcolor** attribute, we can tell the browser that the background color of your page should be blue, like this: **<body bgcolor="blue">**. Attributes always come in name/value pairs like this: name="value". Attributes are always added to the start tag of an HTML element and the value is surrounded by quotes.

Comments in HTML : The comment tag is used to insert a comment in the HTML source code. A comment can be placed anywhere in the document and the browser will ignore everything inside the brackets. You can use comments to write notes to yourself, or write a helpful message to someone looking at your source code. A comment can be inserted with the following tag: **<!— - - comment -->.**

You don't see the text between the tags <!-- and -->

Ex: This html comment would <!-- This is a comment --> be displayed like this.

o/p: This HTML comment would be displayed like this.

Text Formatting Tags:

The following HTML tags are used to format the appearance of the text on web page.

Heading tags - <h1> </h1 >: Headings are defined with the **<h1> to <h6>** tags. <h1> defines the largest heading while <h6> defines the smallest.

Ex: <h1> This is my first Web Page </h1>
<h6> This is my first web page </h6>

HTML automatically adds an extra blank line before and after a heading. A useful heading attribute is having "**align**" attribute with 3 values as "left", "right", "center".
Ex: <h5 align="left">I can align headings </h5>
<h5 align="center">This is a centered heading </h5>
<h5 align="right">This is a heading aligned to the right </h5>

Paragraph tag : Paragraphs are defined with the and tag. Think of a paragraph as a block of text. Most content on a simple web page will appear in paragraphs or sections. A lot of text can appear within the and tags, and browsers will automatically wrap the text onto the next line once it reaches the edge of the screen. You can use the **align attribute** with a paragraph tag as well.
Ex: This is a paragraph

Bold - : The text in between the tags will be bold, and stand out against text around it, the same as in a word processor.

Italic - : italics displays the text at a slight angle.

Underline - <u> </u>: Underlines the text

Strike/del : <strike> </strike>: The **HTML** presentational inline **element** for strikethrough is <**strike**> or <s>. This **element** was, however, deprecated, and replaced by the **<del>** tag. Also works by using <s> </s> instead.
Ex: The HTML strike tag is like this. <strike>strike</strike> through

the middle of the text.
o/p: The HTML strike tag is like this. ~~strike~~ through the middle of the text .

Line Break Tag - </br> : Whenever you use the **
** element, anything following it starts from the next line. This tag is an example of an **empty** element, where you do not need closing tag, as there is nothing to go in between them.

Ex: `<p>This <br> is a para<br> graph with line breaks</p>`

o/p:
This
is a para
graph with line breaks

Horizontal tag - <hr>: Horizontal lines are used to visually break-up sections of a document. The **<hr>** tag creates a line from the current position in the document to the right margin and breaks the line accordingly. The horizontal rule does not have a closing tag. It takes attributes such as align and width. For instance:
Ex: <hr width="50%" align="center">

Preformatted Text - <pre> </pre>: Any text between the pre tags, including spaces, carriage returns and punctuation, will appear in the browser as it is typed in the text editor (normally browsers ignore multiple spaces)

Source Code - <code> </code>: The text is displayed in a fixed-width font, and is commonly used to show source code.

Small - <small> </small>: Instead of having to set a font size, you can use the small tag to render text slightly smaller than the text around it.
Big - <big> </big> : It displays the text in slightly bigger size than the text around it.

Centre - : It makes everything in between the tags centered (in the middle of the page).

Emphasis - : Used to emphasize text, which usually appears in italics, but can vary according to your browser.

Strong Emphasis - : Used to emphasize text more, which usually appears in bold, but can vary according to your browser.

Superscript Text - The : The content of a <sup> element is written in superscript; the font size used is the same size as the characters surrounding it but is displayed half a character's height above the other characters.
Ex: The following word uses a ^{superscript} typeface.
o/p: The following word uses a superscript typeface.

Subscript Text - The <sub> Element: The content of a <sub> element is written in subscript; the font size used is the same as the characters surrounding it, but is displayed half a character's height beneath the other characters.
Ex: The following word uses a _{subscript} typeface.
o/p: The following word uses a subscript typeface.

Font tag: : the font tag is used to change the color, size, style of the font tag.

Font Color -   : Change the color of a few words or a section of text. The color of font can be expressed in 2 ways either in name of the color or in code of the color. In color code the code can be expressed in 6 digit number to represent the hex color code.
Font Size - : Replace the ? with a number from 1 to 7 to change the size of the font. One being the smallest and seven the largest.
Font Size Change -   : For an immediate change of font size with respect to the font size preceding it, this tag increase or decreases the size of the font by the number you specify.
Eg: Some Text

Font Face - : To show text in a particular font, use the font name such "Helvetica" or "Arial" or "Courier".
Ex: welcome

Deleted Text: Anything that appears within **~~...~~** element, is displayed as deleted text.

Ex: I want to drink ~~cola~~ <ins>wine</ins>

I want to drink ~~cola~~ wine

Marked Text: Anything that appears with-in **<mark>...</mark>** element, is displayed as marked with yellow ink.

Ex: The following word has been <mark>marked</mark> with yellow

Text Abbreviation: You can abbreviate a text by putting it inside opening <abbr> and closing </abbr> tags. If present, the title attribute must contain this full description and nothing else.
Ex: <abbr title="Hyper Text Transfer Protocol" > HTTP</abbr>

Links in HTML

- A webpage can contain various links that take you directly to other pages or other websites, and even specific parts of a given page. These links are known as hyperlinks.
- Hyperlinks allow visitors to navigate between Web sites by clicking on words, phrases, and images. Thus you can create hyperlinks using text or images available on a webpage or any other HTML element.
- A link is specified using HTML **<a>**. This tag is called **anchor tag** and anything between the opening <a> tag and the closing **** tag becomes part of the link and a user can click that part to reach to the linked document.

  - In HTML links may be 1. External links or 2.Internal links.
  - External links point from one domain to an entirely separate domain. They may be links from your website to another website to provide additional information for readers, or they may be links from your website to an affiliate program. This link can be absolute path or relative path.
  - Internal links only point within your own specific website or

domain. The menu bar at the top of your site includes internal links. Links from pages on your site to your contact page are another simple example of internal links. This link can be absolute path or relative path.

- Syntax of anchor tag <a> …

Syntax: link text

Ex: My homepage

There are two main parts in link:

1. The **href** is an attribute that specifies the destination address (URL) where to go. In the above example home.html is destination address.
2. The **link text** that is the visible text in between <a> and . Clicking on the link text will send you to the specified address. Link text generally will be displayed in blue color with underline. In the above example MY homepage is link text.

Ex: Google Search

Target attribute of hyperlink: The **target** attribute specifies where to open the linked document.

The target attribute can have one of the following values:

- _blank - Opens the linked document in a new window or tab
- _self - Opens the linked document in the same window/tab as it was clicked (this is default)
- _parent - Opens the linked document in the parent frame
- _top - Opens the linked document in the full body of the window
- framename - Opens the linked document in a named frame

Ex1: Google
Ex2: Tutorial

Image as a Link - <img ...>

By placing an image tag between the <a> and tags, you can turn an image into a link, and clicking on that image will then load

the referenced page. You may notice that the image gets a blue border just as link text became underlined. This can be resolved by setting the border="0" attribute of the image, or using css.

Ex1: <img src="kites-thumb.jpg" alt="kites">
Ex2: <img src="sky-thumb.jpg" alt="Cloudy Sky">

Images in HTML:

Images are used in HTML documents

i. Images can improve the design and the appearance of a web page.
ii. Make the page visually effective and display information.
iii. Images can also be used as links

<img> tag: To display an image you need to specify the URL of the image using the src attribute, replacing url with the filename of image. it contains attributes only, and does not have a closing tag.There are several ways this can be done:

src="picture.jpg" - the filename if the image is in the same directory as the html file.
src="images/picture.jpg" - a relative path when the image is in another directory.
src="http://www.simplehtmlguide.com/images/photo.jpg" - a full URL can also be used.

Alternate Text - <img ... alt="?">: The alt attribute defines the text shown in place of an image when the image cannot load.

Image Size - <img ... width="?" height="?">: An image will normally be shown actual size, but by using the width and height attributes you can change the displayed size. You can specify the size in pixels or as a percentage.

Ex1: <img src="sky.jpg" alt="Cloudy Sky" width="250" height="150">

Border - <img ... border="?">

Add a border by specifying the thickness in pixels. You can also set border="0" to remove the border added when images are used as links.

Image Alignment - <img ... align="?">

By default an image appears at the place specified in the html

code(as with any other tag). However, you can align an image with the surrounding text or paragraph by setting any of align="left | right | top | bottom | middle".

HTML Tables

Table tags are used for displaying data in rows and columns. The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells.

The HTML tables are created using the **<table>** tag in which the A table is divided into rows with the <tr> tag, which stands for table row, and each row is divided into data cells with the <td> tag, which stands for table data. A <td> tag can contain text, links, images, lists, forms, other tables, etc.

<table> ... </table>: Used to define a table, it contains all row and column tags along with their content. It has some attributes to define the table layout.

border="?" - The size of the border (in pixels) surrounding the table
cellspacing="?" - The space (in pixels) between each cell, eg. between rows or columns.
cellpadding="?" - The space, or margin, between the content of a cell and its border.

<tr> </tr>: To start a table row, the tr tags must appear within the table tags.

<td> </td>:A table cell is where the content goes. Cells must exist within rows, where the number of cells in a row determines the number of columns in the table.

Cell properties can be set using the attributes:
align="?" - Alignment of text in the cell: left, center or right
valign="?" - Vertical alignment of the cell: top, middle or bottom.
width="?" - Specify a fixed with of a cell, by default they will only take up as much space as they need.

colspan="?" - Column spanning allows a cell to take up more than one column, in order to match layouts of other rows. Replace ? with the number of columns to span.
rowspan="?" - Row spanning, similar to column spanning, forces a cell to occupy more than one row.
nowrap - No text in the cell will be wrapped onto the next line. Similar to the nobr tag for paragraphs

<th> </th>: Similar to a table cell, a header cell must appear within a table row. Normally found in the first row, header cells are usually shown in bold and centered by the browser.

```
<html>
<body>
  <table border="1">
   <tr>
    <th>Header 1</td>
    <th>Header 2</td>
   </tr>
   <tr>
    <td>Cell A1</td>
    <td>Cell B1</td>
   </tr>
   <tr>
    <td>Cell A2</td>
    <td>Cell B2</td>
   </tr>
  </table>
</body>
</html>
```

HTML - Table Spanning Multiple Rows and Cells

Spanning allow you to extend columns and rows across multiple other columns and rows. Normally, when we creating a table cell, it cannot pass over into the space below or above another table cell. But, you can use the colspan attribute to span multiple columns and rowspan attribute to span multiple rows in a table. Here's is an example:

HTML Table Rowspan Attribute:

```
<table border="1">
<tr>
<td><b>Column 1</b></td>
<td><b>Column 2</b></td>
<td><b>Column 3</b></td>
</tr>
<tr>
<td rowspan="2">Row 1 Cell 1</td>
<td>Row 1 Cell 2</td>
<td>Row 1 Cell 3</td>
</tr>
<tr>
<td>Row 2 Cell 2</td>
<td>Row 2 Cell 3</td>
</tr>
<tr>
<td colspan="3">Row 3 Cell 1</td>
</tr>
</table>
```

HTML Colspan and Rowspan Attributes:

<table border="1">
<tr><th>Column 1</th><th>Column 2</th><th>Column 3</th></tr>
<tr><td rowspan="2">Row 1 Cell 1</td><td>Row 1 Cell 2</td><td>Row 1 Cell 3</td></tr>
<tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>
<tr><td colspan="3">Row 3 Cell 1</td></tr>
</table>

HTML Table Cell Padding and Spacing

The cell-padding and cell-spacing attributes are used to adjust white space inside a table.

- Cell-padding adjust the white space between table cell border and its content.
- Cell-spacing adjust the white space between table cells.

HTML Cellpadding/Cellspacing Code:

```
<table border="1" cellspacing="10" >
<tr>
<td><b>Column 1</b></td>
<td><b>Column 2</b></td>
</tr>
<tr>
<td>Row 1 Cell 1</td>
<td>Row 1 Cell 2</td>
</tr>
<tr>
<td>Row 2 Cell 1</td>
<td>Row 2 Cell 2</td>
</tr>
</table>
```

HTML Cellspacing and Padding:

Column 1	**Column 2**
Row 1 Cell 1	Row 1 Cell 2
Row 2 Cell 1	Row 2 Cell 2

HTML Forms

What is HTML Form

- HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.
- Forms contain special elements called controls like input box, check boxes, radio-buttons, submit buttons, etc.
- Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for processing.
- The <form> tag is used to create an HTML form.

Form - <form> ... </form>: All form elements such as inputs and buttons must go within the form tags. In most cases, a form **must** have the name, action & method **attributes** set.

- name="?" - A unique name identifying the form, used by the action script.
- action="*url*" - The address (URL) of the script that will process the form data when submitted.
- method="?" - The method used by the action script**, post or get**. For example, post would be used to submit data to a user-registration form, and get is used for searches or forms that must return information.

Input Field - <input>: Used to create a simple text-entry field for your form, but is also the basis for many other form input types using the type attribute. *An input element can be of type* text field, checkbox, password field, radio button, submit button, reset button, *etc.*

- name="?" - Unique name for the input to be used by the action script.
- **type="?"** - There are several types of form input fields, text, password, checkbox, radio, file, image, & hidden are among the most common.
- value="?" - Initial value or data displayed in the input field when the form is first loaded.

- size="?" - Defines the input size or width, typically defined in terms of number characters wide instead of pixels.
- maxlength="?" - Maximum length of input field, such as the maximum number of characters for a text input.
- checked - Used with checkbox and radio inputs, it sets the field default to be already checked.

Selection List - <select> ... </select>: A drop-down list, also refered to as a combo-box, allowing a selection to be made from a list of items.

- name="?" - Selector name
- size="?" - The minimum size (width) of the selection list, usually not required as the size of the items will define the list size.
- multiple - Allows a user to select multiple items from the list, normally limited to one.

Selection Item - <option> </option>: An option tag is needed for each item in the list, and must appear within the select tags. The text to be shown for the option must appear between the option tags.

- value="?" - The value is the data sent to the action script with the option is selected. This is **not** the text that appears in the list
- selected - Sets the default option that is automatically selected when the form is shown.

Large Text Area - <textarea> </textarea>: An input that allows a large amount of text to be entered, and allows the height of input box to be a specified unlike the standard input tag.

- name="?" - The unique name assigned to the form field.
- rows="?" - The number of rows of text, defines the vertical size of the text area.
- cols="?" - The horizontal size of the text box, defined as the number of characters (ie. columns).

Text Fields:
<input type="text"> defines a one-line input field that a user can enter text into:

Ex: <form>
First name: <input type="text" name="firstname">

Last name: <input type="text" name="lastname">
</form>

How the HTML code above looks in a browser:

First name:
Last name:

Note: The form itself is not visible. Also note that the default width of a text field is 20 characters.

Password Field:
<input type="password"> defines a password field:

Ex: <form>
Password: <input type="password" name="pwd">
</form>

How the HTML code above looks in a browser:

Password:

Note: The characters in a password field are masked (shown as asterisks or circles).

Radio Buttons:
<input type="radio"> defines a radio button. Radio buttons let a user select ONLY ONE of a limited number of choices:

Ex: <form>
<input type="radio" name="sex" value="male">Male
<input type="radio" name="sex" value="female">Female
</form>

How the HTML code above looks in a browser:

Male
Female

Checkboxes:
<input type="checkbox"> defines a checkbox. Checkboxes let a user select ZERO or MORE options of a limited number of choices.

Ex: <form>
<input type="checkbox" name="vehicle" value="Bike">I have a bike
<input type="checkbox" name="vehicle" value="Car">I have a car
</form>

How the HTML code above looks in a browser:

I have a bike
I have a car

Submit Button:
<input type="submit"> defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

Ex: <form name="input" action="demo_form_action.asp" method="get">
Username: <input type="text" name="user">
<input type="submit" value="Submit">
</form>

How the HTML code above looks in a browser:

Username:

If you type some characters in the text field above, and click the "Submit" button, the browser will send your input to a page called "demo_form_action.asp". The page will show you the received input.

Ex: <FORM action="http://somesite.com/prog/adduser" method="post">
<P> First name: <INPUT type="text" name="firstname">
Last name: <INPUT type="text" name="lastname">
email: <INPUT type="text" name="email">
<INPUT type="radio" name="gender" value="Male"> Male
<INPUT type="radio" name="gender" value="Female"> Female
<BUTTON name="submit" value="submit" type="submit">
Send<IMG src="/icons/wow.gif" alt="wow"></BUTTON>
<BUTTON name="reset" type="reset">
Reset<IMG src="/icons/oops.gif" alt="oops"></BUTTON>
</P>

</FORM>

Unit – II

What is CSS?

- CSS stands for Cascading Style Sheets. Styles define how to display HTML elements.
- CSS is a design language intended to simplify the process of making web pages presentable.
- CSS handles the look and feel part of a web page.
- Using CSS, 1) you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, and variations in display for different devices and screen sizes as well as a variety of other effects.

2) Saves a lot of time - CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file.

3) Provide more attributes - CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

Advantages of CSS:

- **CSS saves time** - You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS

rule of a tag and apply to all the occurrences of that tag. So less code means faster download times.

- **Easy maintenance** - To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** - CSS has a much wider array of attributes than HTML so you can give far better look to your HTML page in comparison of HTML attributes.
- **Multiple Device Compatibility** - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** - Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

CSS Syntax:

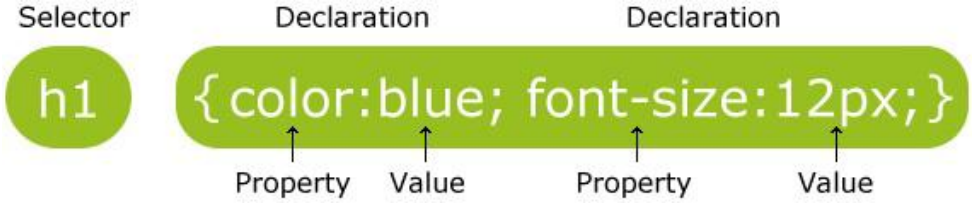
A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts:

- **Selector:** A selector is an HTML tag at which style will be applied. This could be any tag like <h1> or <table> etc.
- **Property:** A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS

properties. They could be *color* or *border* etc.

- **Value:** Values are assigned to properties. For example *color* property can have value either *red* or *#F1F1F1* etc.

Basic Structure of a Style



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

Example:

1. h1 { font-size:12; color:red; }
2. **p { color: red; text-align: center; }**

- CSS tags are also known as elements or selectors. They have a basic layout very similar to regular HTML tags.

Regular HTML tag	<tag property="value">
CSS command tag	element {property: value;}

CSS Types (CSS Styles): There three ways of applying a style sheet:

1. External style sheet
2. Internal style sheet
3. Inline style

1. **External Style Sheet**

An external style sheet is ideal when the style is applied to many pages. Most websites today use external style sheets. External styles are styles that are written in a separate document and then attached to various web documents. With an external style sheet, you can change the look of an entire Web site by changing one file. This makes long term site management much easier. Each page must link to the style sheet using the **<link> tag**. The <link> tag goes **inside the head section**.

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

Attributes of link tag:

rel: The **rel attribute** specifies the relationship between the current document and the the **linked** document

type: Specifies the style sheet language as a content-type. This attribute is required.

href: Specifies the style sheet file having Style rules. This attribute is a required.

<head>

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

</head>

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

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

2. **Internal Style Sheet (Embedded Style Sheet):**

Embedded styles are styles that are embedded in the head of the document. An internal style sheet should be used when a single document has a unique style. You define internal styles in the **head section** of an HTML page, by using the **<style> tag**, like this:

```
<html>
<head>
    <title> Embedded Style Sheet </title>
    <style type="text/css">
        hr {color:sienna;}
        p {margin-left:20px;}
        body {background-image: url("images/back40.gif");}
    </style>
</head>
<body>
```

```
<p> This is an Example of Embedded Style Sheet</p>
<hr>
</body>
</html>
```

3. **Inline Styles**

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly!

To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

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

CSS Selectors:

The different styles that we can use to apply on css elements can be done by using different types of css selectors.

1) CSS Element Selector: The element selector selects the HTML element by name.

```
Ex: p {
        text-align: center;
      color: blue;
    }
```

2) CSS Id Selector: The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is

chosen to select a single, unique element. It is written with the hash character (#), followed by the id of the element. Let’s take an example with the id "para1".

```
<html>
<head>
<style>
#para1 {      text-align: center;      color: blue;  }
</style>
</head>
<body>
<p id="para1">Hello CSS </p>
<p>This paragraph will not be affected.</p>
</body>
</html>
```

3) CSS Class Selector: The class selector selects HTML elements with a specific class attribute. It is used with a period character**.** (full stop symbol) followed by the class name.

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {      text-align: center;      color: blue;  }
</style>
</head>
<body>
<h1 class="center">This heading is blue and center-aligned.</h1>
<p class="center">This paragraph is blue and center-
```

aligned.</p>
</body>
</html>

4) CSS Universal Selector: The universal selector is used as a wildcard character. It selects all the elements on the pages.

```
<!DOCTYPE html>
<html>
<head>
<style>
* {     color: green;     font-size: 20px;  }
</style>
</head>
<body>
<h2>This is heading</h2>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

5) CSS Group Selector: The grouping selector is used to select all the elements with the same style definitions. Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

```
<html>
<head>
<style>
h1, h2, p {  text-align: center;  color: blue;  }
</style>
```

```
</head>
<body>
<h1>Hello Javatpoint.com</h1>
<h2>Hello Javatpoint.com (In smaller font)</h2>
<p>This is a paragraph.</p>
</body>
</html>
```

The Span and Div tags

The SPAN and DIV HTML tags are very useful for use with CSS.

The DIV Element

Div (short for division) divides the content into individual sections. Each section can then have its own formatting, as specified by the CSS. Div is a block-level container, meaning that there is a line feed after the tag. The DIV element defines logical divisions on your web page. It acts a lot like a P element, by placing newlines or carriage returns before and after the division. A division can have multiple paragraphs in it. The element is a block-level element.

Using the DIV Tag

To use the DIV element, surround the area of your page that you want as a separate division with the and tags:

```
<div>
<p>contents of div</p>
</div>
```

The DIV element allows you to define the style of entire sections of the HTML. You can define a division of your page as a callout and give that area a different style from the surrounding text. That area may have images, paragraphs, and headlines anything you wanted. The DIV element also gives you the ability to identify unique areas of

your documents. The most important attributes of the DIV element are:

- style
- class
- id

Example:

```
<html>
<body>
<div style="background-color: black;
color: white;
padding: 20px;">
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
<p>Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.</p>
</div>
</body>
</html>
```

The Span Tag

Span is similar to div in that they both divide the content into individual sections. The difference is that span goes into a finer level, so we can span to format a single character if needed. There is no line feed after the tag.

The main difference between the SPAN and DIV elements is that SPAN doesn't do any formatting of it's own. As mentioned above, the DIV element includes a paragraph break. The SPAN element simply tells the browser to apply the style rules to whatever is within the SPAN.

To use the SPAN element, simply surround the text that you want to add styles to with the and tags. The SPAN element has no required attributes, but the three that are the most useful are the same as for the DIV element:

- style
- class
- id

Use SPAN when you want to change the style of elements without placing them in a new block-level element in the document. For example, if you had a Level 3 Heading (H3) that you wanted the second word to be red, you could surround that word with 2ndWord and it would still be a part of the H3 tag, just red.

Ex:

- <h3>This is My Awesome Headline</h3>
- My mother has blue eyes and my father has dark green eyes.

CSS3:

Cascading Style Sheets Level 3 (CSS3) is the iteration of the CSS standard used in the styling and formatting of Web pages. CSS3

incorporates the CSS2 standard with some changes and improvements. A key change is the division of standard into separate modules, which makes it easier to learn and understand. CSS3 permits to pick out additional hypertext markup language tags and outline however they're displayed on an online browser. CSS3 is divided into several separate documents called "modules". Some of the most important **CSS3 modules are:** –

- Selectors
- Box Model
- **Backgrounds and Borders**
- **Text Effects**
- **2D/3D Transformations**
- **Animations**
- Multiple Column Layout
- User Interface

CSS3 Borders:

A CSS3 Border is such an afford of style sheet which reduces the human efforts of Photoshop and other graphical applications. An individual can create the rounded borders, border shadow, imaged based border and etc. with the help of CSS3 Border. Basically we use three features to create the border:

- border-radius
- box-shadow
- border-image

- **border-radius** is a such property of CSS3 by which we can

create the rounded corners. In CSS3, creating rounded corners is easy. In CSS3, the border-radius property is used to create rounded corners: div { border: 2px solid; border-radius: 25px; }

- **box-shadow** is a such property of CSS3 by which we can create the shadow of the border. Easy and cool, no need to write more code, just specify the location of image and assign the selector to the element.

- **border-image** is a such property of CSS3 by which we can create the customized border, as we can put our own image as a border. Easy and cool, no need to write more code, just specify the location of image and assign the selector to the element.

Creating CSS3 Rounded Corners

The border-radius property can be used to create rounded corners. This property typically defines the shape of the corner of the outer border edge. Prior to CSS3, sliced images are used for creating the rounded corners that was rather bothersome.CSS3 Rounded corners are used to add special colored corner to body or text by using the border-radius property.A simple **syntax**:

```
#rcorners7 {
border-radius: 60px;
background: #FF0000;
padding: 20px;
width: 200px;
```

height: 150px;
}

border-radius	Use this element for setting four boarder radius property
border-top-left-radius	Use this element for setting the boarder of top left corner
border-top-right-radius	Use this element for setting the boarder of top right corner
border-bottom-right-radius	Use this element for setting the boarder of bottom right corner
border-bottom-left-radius	Use this element for setting the boarder of bottom left corner

Ex:

```
<html>
 <head>
  <title>Title Name will go here</title>
</head>
<style>
 #border_radius
 {
    border:10px solid;
    font-size: 24px;
    color: #00ff00;
    font-weight: bold;
    padding: 10px;
    background: #000FCF;
    border-top-left-radius:25px;
```

```
        border-bottom-right-radius:25px;
      }
     #border_image
      {
        border-width: 15px;
        border-radius:15px;
        border-image:url(tulips.jpg) 30 30 round; /* Firefox */
        border-image:url(4.jpg) 30 30 round; /* Safari and
Chrome */
      }
    </style>
   <body>
      <div id="border_radius">  With the help of border-radius
properties, we can make the rounded corners border.  </div>
      <div id="border_image">  You can see the customized
border. This could be either *.png or *.jpg format.  </div>
   </body>
  </html>
```

CSS3 Background:

CSS3 provided several new background properties which facilitate background control. The newly specified properties in CSS3 for background are:

background-clip	Specifying the painting area of the background images
background-origin	Where the background will be painted
background-size	Determining the size of the background-image

Background-clip:

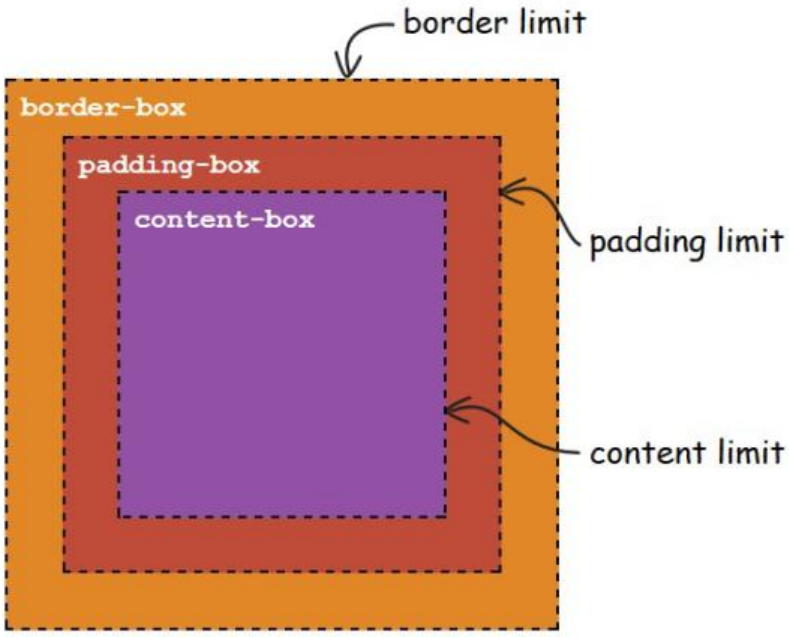
The background-clip property defines how far the background (color or image) should extend within an element. We can make the background cover just the padding-box or just the content-box with the help of background-clip. **Clipping** means cutting out and not displaying what falls outside the clipping region. It lets you control how far a background image or color extends beyond an element's padding or content.

Syntax:

background-clip: border-box|padding-box|content-box;

Values:

- border-box is the default value. This allows the background to extend all the way to the outside edge of the element's border.
- padding-box clips the background at the outside edge of the element's padding and does not let it extend into the border.
- content-box clips the background at the edge of the content box.



- If the padding is 0, then the padding-box is exactly the same size as the content-box, and the content limit coincides with the padding limit.
- If the border-width is 0, the border-box is the same size as the padding-box, and the border limit coincides with the padding limit.
- If both the padding and the border-width are 0, then all the three boxes (the content-box, the padding-box, and the border-box) have the same size, and the content limit, the padding limit, and the border limit all coincide.

Ex: Specify how far the background should extend within an element:

```
div {
  border: 10px dotted black;
  padding: 15px;
  background: lightblue;
  background-clip: padding-box;
```

}

Background-size:

It's sometime needed to specify a certain size to the background image.To control the background image size, all you need to do is to use background-size property in body selector as will be shown in the following code.

```
<!DOCTYPE html>
<html>
<head>
<title>CSS3 Modules: borders &background </title>
<style>
body {
background:url(tulips.jpg);
background-size:180px 160px;
background-repeat: no-repeat;
        }
</style>
</head>
<body>
</body>
</html>
```

Note: if you didn't specify background-repeat property to no-repeat, it will repeat the images several times. To use the original size of the image just set the background dimensions to auto, instead of using values.

Background-image:

The background-image property in CSS applies a graphic (e.g. PNG, SVG, JPG, GIF, WEBP) or gradient to the background of an element. The background-image property sets one or more background images for an element. By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally. There are two different types of images you can include with CSS: **regular images and gradients.**

Tip: The background of an element is the total size of the element, including padding and border (but not the margin).

Syntax:

background-image: url|none;

values:

- **URL:** the URL of the image. To specify more than one image, separate with comma.
- **None:** No background image will be displayed. This is default.
- **linear-gradient():**Sets a linear gradient as the background image. Define at least two colors (top to bottom)
- **radial-gradient( ):**Sets a radial gradient as the background image. Define at least two colors (center to edges)
- **repeat-linear-gradient( ):**repeats the linear gradient
- **repeat-radial-gradient( ):**repeats the radial gradient.

Ex1:

```
body {
  background-image: url("img_tree.gif"), url("paper.gif");
  background-color: #cccccc;
```

```
}
```
Ex2:
```
body {
   background-image: url("img_tree.gif"), url("paper.gif");
   background-repeat: no-repeat, repeat;
   background-color: #cccccc;
}
```
Ex3:
```
#grad1 {
   height: 250px;
   width:400px;
   padding:50px;
   background-image: linear-gradient(orange, white, green);
}
```

Text-Effects:

CSS3 contains several new text features. In this chapter we will learn about the following properties:

- text-overflow
- word-wrap
- word-break

Properties provided by CSS3 for text editing is showing below:

text-emphasis	Applies emphasis marks
text-justify	Justify text
text-outline	Specifies a text outline
text-overflow	Specify behavior is the text overflow it's container
text-shadow	Add shadow to the text , “like fire and ice”
text-wrap	Wrap text into multiple lines according to certain delimiter
word-break	Specifies line breaking rules

word-wrap	Break long words and divide them to more than one line like:immunosuppressive which can make it immune-suppressive

Text-Overflow:

- The text-overflow property specifies how overflowed content that is not displayed should be signaled to the user. It can be **clipped,** display an **ellipsis** (...), or display a **custom string**.
- Both of the following properties are required for text-overflow:
  - white-space: nowrap;
  - overflow: hidden;
- The CSS text-overflow property specifies how overflowed content that is not displayed should be signaled to the user.
- The text-overflow property only affects content that is overflowing a block container element in its *inline* progression direction (not text overflowing at the bottom of a box, for example).

Syntax:

text-overflow: clip | ellipsis | *string* ;

Values:

Clip	Default value. The text is clipped and not accessible
Ellipsis	Render an ellipsis ("...") to represent the clipped text
String	Render the given string to represent the clipped text

Ex: <style>

div.a {

```
    white-space: nowrap;
    width: 50px;
    overflow: hidden;
    text-overflow: clip;
    border: 1px solid #000000;
}
div.b {
    white-space: nowrap;
    width: 50px;
    overflow: hidden;
    text-overflow: ellipsis;
    border: 1px solid #000000;
}
</style>
```

Word-wrapping:

- Sometimes, while writing in a container on a web page, the end of line is not displayed properly. The solution was provided CSS3 in word-wrap property which wrap the long words to the next line.
- The CSS word-wrap property allows long words to be able to be broken and wrap onto the next line.
- If a word is too long to fit within an area, it expands outside; The word-wrap property allows you to force the text to wrap - even if it means splitting it in the middle of a word.

Syntax:

word-wrap: normal | break-word | initial;

Values:

- Normal: break words only at allowed break points.
- Break-word: allows unbreakable words to be broken
- Initial: sets this property to its default value.

Ex: Allow long words to be able to break and wrap onto the next line:

```
div {
   word-wrap: break-word;
     }
```

Word-break:

- The word-break property in CSS can be used to change when line breaks ought to occur.
- Normally, line breaks in text can only occur in certain spaces, like when there is a space or a hyphen. It specifies how words should break when reaching the end of a line.

Syntax:

word-break: normal | break-all | keep-all | break-word;

Values:

- normal: use the default rules for word breaking.
- break-all: any word/letter can break onto the next line.
- keep-all: for Chinese, Japanese and Korean text words are not broken. Otherwise this is the same as normal.
- break-word: to prevent overflow, word may be broken at arbitrary points.

Ex: p {

word-break: break-all;
}

Web Fonts:

Web fonts allow Web designers to use fonts that are not installed on the user's computer. When you have found/bought the font you wish to use, just include the font file on your web server, and it will be automatically downloaded to the user when needed. Your "own" fonts are defined within the CSS `@font-face` rule.

Different Font Formats:

When you purchase web fonts licensing, you receive a package of font files that typically include at least some of the following formats:

- **TrueType Fonts (TTF):** TrueType is the most common font format for both the Mac OS and Microsoft Windows operating systems.
- **OpenType Fonts (OTF):** OpenType is a format for scalable computer fonts. OpenType fonts are used commonly today on the major computer platforms.
- **The Web Open Font Format (WOFF):** WOFF is a font format for use in web pages. WOFF is essentially OpenType or TrueType with compression and additional metadata.
- **SVG Fonts/Shapes:** SVG fonts allow SVG to be used as glyphs when displaying text.
- **Embedded OpenType Fonts (EOT):** EOT fonts are a compact form of OpenType fonts designed by Microsoft for use as embedded fonts on web pages.

We make use of @font-face to include fonts in CSS.

```
Ex1: @font-face {
    font-family: myFirstFont;
    src: url(sansation_bold.woff);
    font-weight: bold;
}
```

following code shows the sample code of font face

```
<html>
<head>
<style>
        @font-face {
font-family: myFirstFont;
src: url(/css/font/SansationLight.woff);
        }
div {
font-family: myFirstFont;
        }
</Style>
</head>
<body>
<div>This is the example of font face with CSS3.</div>
<p><b>Original Text :</b>This is the example of font face with
CSS3.</p>
</body>
</html>
```

It will produce the following result –

This is the example of font face with CSS3.

Original Text: This is the example of font face with CSS3.

Fonts description

The following list contained all the fonts description which are placed in the @font-face rule −

Values	Description
font-family	Used to defines the name of font
Src	Used to defines the URL
font-stretch	Used to find, how font should be stretched
font-style	Used to defines the fonts style
font-weight	Used to defines the font weight(boldness)

CSS3 Transforms:

Using the new CSS3 transform property you can create element transformations and to change the shape, size and position of the element. 2D transforms are used to re-change the element structure. The transform property can get a set of transformation functions which can be composed if you write them separated by whitespace.

The 2D transform functions included:

- translate – given left and top parameters, the element will move from its position to the new point. There are also a translateX and translateY functions that get only one parameter and translate the element only in one axis.
- rotate – given a degree the element rotate clockwise according to the degree. Pay attention that the parameter should be in a specific format for example these are valid parameters: 60deg, 80deg and etc.

- scale – given a width and height, the element will increase or decrease its size. There are also scaleX and scaleY functions that get only one parameter and scale the element only in one axis.
- skew – given x degree and y degree parameters, the element will turn in the given angles first in the x-axis and then in the y-axis. There are also skewX and skewY functions that get only one parameter and skew the element only in one axis.
- matrix – given six a-f parameters apply the transformation matrix [a b c d e f] on the element.

3D Transforms: Using with 3d transforms, we can move element to x-axis, y-axis and z-axis. The basic functions are:

1. rotateX( )
2. rotateY( )
3. rotateZ( )

The 3D transform functions included:

- matrix3d – the same as the matrix function but now gets 16 parameters.
- translate3d – gets an additional z-axis parameter.
- scale3d – gets an additional z-axis parameter. There is also scaleZ function that scale the element only in the z-axis.
- rotate3d – gets four parameters – x, y and z that define the [x y z] direction vector and a degree to rotate in that direction. There is also a rotateZ function that rotate the element in the z-axis.

CSS transitions

Introduction:

Transitions are the grease in the wheel of CSS transforms. **CSS transitions** provide a way to control animation speed when changing CSS properties. Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time. By applying a transition you can control the change, making it smooth and gradual. With CSS transitions you have the potential to alter the appearance and behavior of an element whenever a state change occurs, such as when it is hovered over, focused on, active, or targeted.

Defining transitions

CSS Transitions are controlled using the shorthand transition property. You can control the individual components of the transition with the following sub-properties:

There are 4 sub-properties that are required in order for the transition to take effect:

1. transition-property
2. transition-duration
3. transition-timing *(optional)*
4. transition-delay *(optional)*

Here's the full shorthand sequence. Again, the first two properties are required.

```
div {
  transition: [property] [duration] [timing-function] [delay];
}
```

1. **transition-property *(required)***

The transition-property specifies the CSS property where the transition will be applied. You may apply a transition to an individual property (e.g., background-color or transform) or to all properties in the rule-set (i.e., all).

CSS syntax examples:

```
div {
  transition-property: all;
  transition-property: transform;
}
```

2. **transition-duration *(required)***

The transition-duration property specifies the time span of the transition. You can specify in seconds or milliseconds.

CSS syntax example:

```
 div {
   transition-duration: 3s;
 }
```

Shorthand example:

```
div {
  transition: all 3s;
}
```

3. **transition-timing *(optional)***

The transition-timing-function property allows you to define the speed of the transition over the duration. The default timing is ease, which starts out slow, quickly speeds up, and then slows down at the end. The other timing options are: linear, ease, ease-in, ease-out, and ease-

in-out.

Here’s an example of the different timing options (used with the transform: translateproperty):

For more advanced timing options, you can define a custom timing function with a cubic-bezier.

CSS syntax example:

```
div {
  transition-timing-function: ease-in-out;
}
```

Shorthand example:

```
div {
  transition: all 3s ease-in-out;
}
```

4. **transition-delay *(optional)***

The transition-delay property allows you to specify when the transform will start. By default, the transition starts as soon as it is triggered (e.g., on mouse hover). However, if you want to transition to start after it is triggered you can use the transition delay property.

Shorthand example:

```
div {
  transition: all 3s 1s;
}
```

A negative value will start the transition immediately, but part way through the transition process.

UNIT-III

Introduction:

JavaScript is a dynamic scripting language. JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Mozilla, Firefox, Netscape, Opera.

JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**.

1. JavaScript was designed to add interactivity to HTML pages
2. JavaScript is a scripting language (a scripting language is a lightweight programming language)
3. A JavaScript consists of lines of executable computer code, JavaScript is usually embedded directly into HTML pages
4. JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
5. It is client-side scripting language designed for creating network-centric applications.
6. JavaScript is a case sensitive language.
7. JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more.

Advantages of Java Script:

- **Client-Side execution**: No matter where you host JavaScript, Execute always on client environment to save a bandwidth and make execution process fast. Being client-side reduces the demand on the website server.

- **Interoperability**. JavaScript plays nicely with other languages and can be used in a huge variety of applications. Unlike PHP or SSI scripts, JavaScript can be inserted into any web page regardless of the file extension. JavaScript can also be used inside scripts written in other languages such as Perl and PHP.

- **Rapid Development**: JavaScript syntax's are easy and flexible for the developers. JavaScript small bit of code you can test easily on Console Panel (inside Developer Tools) at a time browser interpret return output result. In-short easy language to get pick up in development.

- **Browser Compatible**: The biggest advantages to a JavaScript having a ability to support all modern browser and produce the same result.

- **Speed**. Client-side JavaScript is very fast because it can be run immediately within the client-side browser. Unless outside resources are required, JavaScript is unhindered by network calls to a backend server. It also has no need to be compiled on the client side which gives it certain speed advantages.

- **Extended Functionality**. Third party add-ons like Greasemonkey enable JavaScript developers to write snippets of JavaScript which can execute on desired web pages to extend its functionality.

- **User Interface Interactivity**: JavaScript used to fill web page data dynamically such as drop-down list for a Country and State. Base on selected Country, State drop down list dynamically filled. Another one is Form validation, missing/incorrect fields you can alert to a users using alert box.

Where to place JavaScript code:

The <script> tag:

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 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".

There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows −

- Script in <head>...</head> section.

- Script in <body>...</body> section.

- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section

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.

So your JavaScript segment will look like −

```
<script language="javascript" type="text/javascript">

   JavaScript code

</script>
```

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

Data Types in JavaScript:

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

```
var a=40;//holding number
var b="Rahul";//holding string
```

Primitive data types: these are the primary data types in javaScript

- **Boolean-** A value which can ony be either true or false.

- **Number** – Any numeric value whether an integer number or float number.

Eg: 12 , 3.1415

- **String:** A group of characters represents string. i.e text. A string can be enclosed by a pair of single quotes (**'**) or double quote (**"**).

Eg: “welcome”

- **Null –** it defines a single value, the only value is "null" – to represent nothing.
- **Undefined**: represents undefined value. The only value is "undefined" – to represent the value of an uninitialized variable.

Non-Primitive data types (Reference data Types):

- **Object**: object is a named collection of data. An object is a collection of properties. Properties can be variables (Fields) or Functions (Methods)
- **Array:** Array is a sequence of values (an array is actually a predefined object)
- **Regexp**: Represents a regular expression.

JavaScript Variables:

Like many other programming languages, JavaScript has variables. 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.

We create variables and assign values to them in the following way:

```
var christianName = "Fred"      (string)
var surname = "Jones"           (string)
var age = 37                    (numeric)
var married = false             (Boolean)
```

- When a new variable is created (or *declared*) its name must be preceded by the word var
- The type of the variable is determined by the way it is declared:
  - if it is enclosed within quotes, it's a string
  - if it is set to true or false (without quotes) it's a boolean
  - if it is a number (without quotes) it's numeric
- We refer to the equals sign as the *assignment operator* because we use it to assign values to variables;
- Variable names must begin with a letter or an underscore
- Variable names must not include spaces

Dialog boxes in JavaScript:

JavaScript supports three important types of dialog boxes. These dialog boxes can be used to

raise an alert, or to get confirmation on any input or to have a kind of input from the users. Here we will discuss each dialog box one by one.

- Alert dialog box
- Confirmation dialog box
- Prompt dialog box

Alert Dialog Box: An alert dialog box is mostly used to give a warning message to the users. It is used to show a message in the dialog box, and there is an OK button. It is mostly used to prompt message if user missed input value or invalid data in given form or text. When an alert box pops up, the user will have to click "OK" to proceed.

Syntax: window.alert("*sometext*");

Here window is optional.

Eg: 1. alert("I am an alert box!");

Var price=10.00;

alert("The price is "+price);

Which produces:

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: **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.

```
Syntax:       window.confirm("sometext");
Eg:  if (confirm("Press a button!"))
    {
                txt = "You pressed OK!";
      }
```

```
else
{
            txt = "You pressed Cancel!";
  }
```

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 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.

Syntax: window.prompt("*sometext*","*defaultText*");

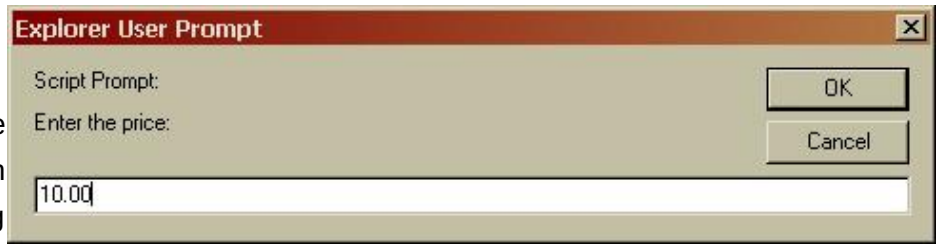
Window is optional here.

Eg:

price = prompt("Enter the price", "10.00");

var person = prompt("Please enter your name");

which produces -



Apart from all the above dialog boxes we can also use the following method to display.

document.write() method – this method is used to display the data on to the screen.

Eg: document.write("welcome")

document.write(23)

a=45; document.write("a = " + a); will display: a = 45

JavaScript Operators:

Operators are a type of command. They perform operations on variables and/or literals and produce a result.

- **Arithmetic Operators:** +(addition), - (Subtraction), * (multiplication), / (Division which gives quotient),and % (modulo division which gives remainder.)

Eg: 5/2=2 or 2.5 but 5%2=1

- **Relational (Comparison) Operators:** <, <=, >, >=, !=, !==, == and ===(this will check with type of data also).

===, !== (Strictly equals and strictly not equals) i.e., Type and value of operand must match / must not match

Eg: var v2 = ("5" === 5); // false
var v3 = (5 === 5.0); // true

- **Logical Operators:**

1. **!** – Logical NOT-

!OP1

!0=1 and !1=0 i.e !(true)=false, !(false)=true

2. **&&** – Logical AND

OP1 && OP2

If OP1 is true, expression evaluates to the value of OP2.

Otherwise the expression evaluates to the value of OP1.

Results may not be a boolean value.

3. **||** – Logical OR

OP1 || OP2

If OP1 is true, expression evaluates to the value of OP1. Otherwise the expression evaluates to the value of OP2.

- **Assignment Operators:** =, +=, -=,*=, /=, %=. These are shorthand assignment operators.

Eg: a=5 a+=2 means a=a+2 i.e a=5=2 a=7

- **Conditional (or ternary) Operator:**

"**? :**" ternary conditional statement. It works like if-else statement.

Exp1**?** Exp2 **:** Exp3 here exp1 is a condition if this conditions gives true value then exp2 will be evaluated. If exp1 condition is false then exp3 will be evaluated.

Eg: big = (a>b) ? a : b

- **Increment/Decrement operators: ++ and –**

++ will increment the variables value by one and – will decrement the value by one.

Eg: a=5 a++ a=a+1 a=5+1 a=6

a=5 a-- a=a-1 a=5-1 a=4

- **Other operators:**

New, delete, +(concatenation)etc.

Arrays:

- An array is an object
- Contains data elements in sequential order

- Elements need not be of the same type
- Elements can be primitive values or object references (possibly functions or other arrays)
- Has dynamic length
- Index of array runs from 0 to N-1.

Eg: Created via an Array literal:

1. var a3 = [“7”, 1, new Date(), false];

Eg: create an array via array object of N elements, you can write:

var myArray = new Array(N);

Eg: Can store values of different types

var a1 = new Array( );
a1[0] = 27; a1[45] = "Hello";

var Car = new Array(3);
Car[0] = "Ford";
Car[1] = "Toyota";
Car[2] = "Honda";
var Car2 = new Array("Ford", "Toyota", "Honda");

Array object methods:

As array is a **JavaScript object,** arrays have several methods associated with arrays via which the array content can be manipulated. Few of those properties are:

Join() – It returns all elements of the array joined together as a single string. This takes one argument.

Reverse(): It reverse the order of the elements in an array.

Length(): JavaScript array **length** property returns the number of elements in an array.

Push(): inserts/push a new element into the given array.

pop(): removes the last element from the array.

Eg:var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.pop();
Fruits.push(“strawberry”);

Conditional statements in JavaScript:

JavaScript supports conditional statements which are used to perform different actions based on different conditions.

In JavaScript we have the following conditional statements:

- Use **if** to specify a block of code to be executed, if a specified condition is true

- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be executed

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(*condition*)
{
block of code to be executed if the condition is true
}

Here a condition 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.

Eg: **<script>**
var a=20;
if(a**>**10){
document.write("value of a is greater than 10");
}
</script>

The else Statement: Use the **else** statement to specify a block of code to be executed if the condition is false. The syntax is as follows:

if (*condition*)
{
block of code to be executed if the condition is true
}
else
{
block of code to be executed if the condition is false
}

Here JavaScript condition 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.

Eg: **<script>**
var a=20;
if(a%2==0)
{
document.write("a is even number");

```
}
else
{
document.write("a is odd number");
}
</script>
```

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 (condition1) {
   block of code to be executed if condition1 is true
} else if (condition2) {
   block of code to be executed if the condition1 is false and condition2 is true
} else {
   block of code to be executed if the condition1 is false and condition2 is false
}
```

Eg:

```
if (time < 10) {
    greeting = "Good morning";
} else if (time < 20) {
    greeting = "Good day";
} else {
    greeting = "Good evening";
}
```

The switch-case statement:

The **JavaScript switch statement** is used *to execute one code from multiple expressions*. It is just like else if statement that we have learned in previous page. But it is convenient than *if..else..if* because it can be used with numbers, characters etc.

The break Keyword

- When JavaScript reaches a **break** keyword, it breaks out of the switch block.
- This will stop the execution of more code and case testing inside the block.
- When a match is found, and the job is done, it's time for a break. There is no need for more testing.

The default Keyword

- The **default** keyword specifies the code to run if there is no case match.

Syntax:

```
switch(expression)
{
  case value1:
        code to be executed;
        break;
  case value2:
        code to be executed;
        break;
  ......

default:
        code to be executed if above values are not matched;
}
```

Eg:
```
<script>
    var grade='B';
    var result;
    switch(grade)
    {
    case 'A':
        result="A Grade";
        break;
    case 'B':
        result="B Grade";
        break;
    case 'C':
        result="C Grade";
        break;
    default:
        result="No Grade";
    }
    document.write(result);
    </script>
```

Loops in JavaScript:

While writing a program, you may encounter a situation where you need to perform an action over and over again. In such situations, you would need to write loop statements to reduce the number of lines.

The **JavaScript loops** are used *to iterate the piece of code* using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

There are four types of loops in JavaScript.

1. while loop
2. do-while loop
3. for loop
4. for-in loop

1) **While loop:**

The most basic loop in JavaScript is the while loop. 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. The syntax is as follows:

```
while (condition)
{
   code to be executed
}
```

Eg:

```
<script>
var i=11;
while (i<=15)
{
document.write(i + "<br/>");
i++;
}
</script>
```

O/P:

```
11
12
13
14
15
```

2) The Do/While Loop:

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, and then it will repeat the loop as long as the condition is true. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested.

Syntax

```
do {
   code block to be executed
}
while (condition);
```

Eg:

```
<script>
var i=21;
do{
document.write(i + "<br/>");
```

```
i++;
}while (i<=25);
</script>
```

O/P:

```
21
22
23
24
25
```

3) JavaScript For loop:

The **JavaScript for loop** *iterates the elements for the fixed number of times*. It should be used if number of iteration is known. The syntax of for loop is given below.

```
for (initialization; condition; increment)
{
   code to be executed
}
```

For loop is having three parts:

- **Statement 1 (initialization)** is executed before the loop (the code block) starts. You can initiate many values in statement 1 (separated by comma) and it is optional.
- **Statement 2** (condition) defines the condition for running the loop (the code block). If statement 2 returns true, the loop will start over again, if it returns false, the loop will end.
- **Statement 3** (increment) is executed each time after the loop (the code block) has been executed. Statement 3 can do anything like negative increment (i--), positive increment (i = i + 15), or anything else.

Eg: <script>

```
for (i=1; i<=5; i++)
{
    document.write(i + "<br/>")
}
</script>
```

O/P :

```
1
2
3
4
5
```

4) The for…in loop:

The **for...in** loop is used to loop through an object's properties. Once you understand how objects behave in JavaScript, you will find this loop very useful.

Syntax:

```
for(variable-name in object)
{
```

```
Statement or block to execute
}
```

Functions in JavaScript:

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

JavaScript supports: 1. built-in functions and 2. User-defined functions.

1.Built-in functions in JavaScript:

JavaScript supports various built-in functions or methods.

- **Number methods**-The Number object supports built-in functions like valueOf(), toString(), toPrecision(), toFixed() etc.

parseInt() function: The parseInt() function converts a string argument and returns an integer of the specified radix (the base in mathematical numeral systems).

Eg: var p = parseInt(prompt(“enter a value”));

parseFloat(): The **parseFloat()** function parses an argument and returns a floating point number.

- **String methods**- charAt(), concat(), indexOf(), length() etc are the functions in strings.

concat(): Combines the text of two strings and returns a new string.

indexOf():Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.

charAt(): Returns the character at the specified index.

- **Boolean methods**- valueOf(), toSource() are few of these.

valueOf():Returns the primitive value of the Boolean object.

toSource():Returns a string containing the source of the Boolean object; you can use this string to create an equivalent object.

- **Array methods-** push(), pop(), reverse(), concat(), sort() etc are examples.

concat():Returns a new array comprised of this array joined with other array(s) and/or value(s).

join():Joins all elements of an array into a string.
push():Adds one or more elements to the end of an array and returns the new length of the array.
pop():Removes the last element from an array and returns that element.
reverse():Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.

2. **User-defined functions:** JavaScript also allows the users to create their functions.

Function declaration & definition:

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces. A function can return value of any type using the keyword **"return".**

Syntax: function *function-Name*(*parameters*)

```
{
  code to be executed
}
```

Eg: ***function myFunction(a, b)***

```
{
    return a * b;
}
```

Calling a Function

Declared functions are not executed immediately. They will be executed later, when they are invoked (called upon).To invoke a function somewhere later in the script, you would simply need to write the name of that function with parameters if any.

How to invoke a function:
Eg: we need to use html form objects to invoke a function by a click action on a button.

```
<html>
<head>
  <script type="text/javascript">
        function sayHello()
        {
           document.write ("Hello there!");
        }
  </script>

</head>
<body>
     <p>Click the following button to call the function</p>
```

```
<form>
        <input type="button" onclick="sayHello()"
value="Say Hello">
</form>
</body>
</html>
```

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.

Function Parameters

There is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma. We can call function by passing arguments. Let’s see the example of function that has one argument.

```
<script>
function getcube(number)
```

```
{
    alert (number*number*number);
}
</script>
<form>
<input type="button" value="click" onclick="getcu
</form>
```

Function with Return Value: We can call function that returns a value and use it in our program. A JavaScript function can have an optional return statement. This is required if you want to return a value from a function. This statement should be the last statement in a function. Let's see the example of function that returns value.

```
<script>
function getInfo(){
return "hello javatpoint! How r u?";
}
</script>
<script>
document.write(getInfo());
</script>
```

Objects in JavaScript:

- In JavaScript, an **object** is defined as

an **"unordered collection of properties each of which contains a primitive value, object, or function ".**

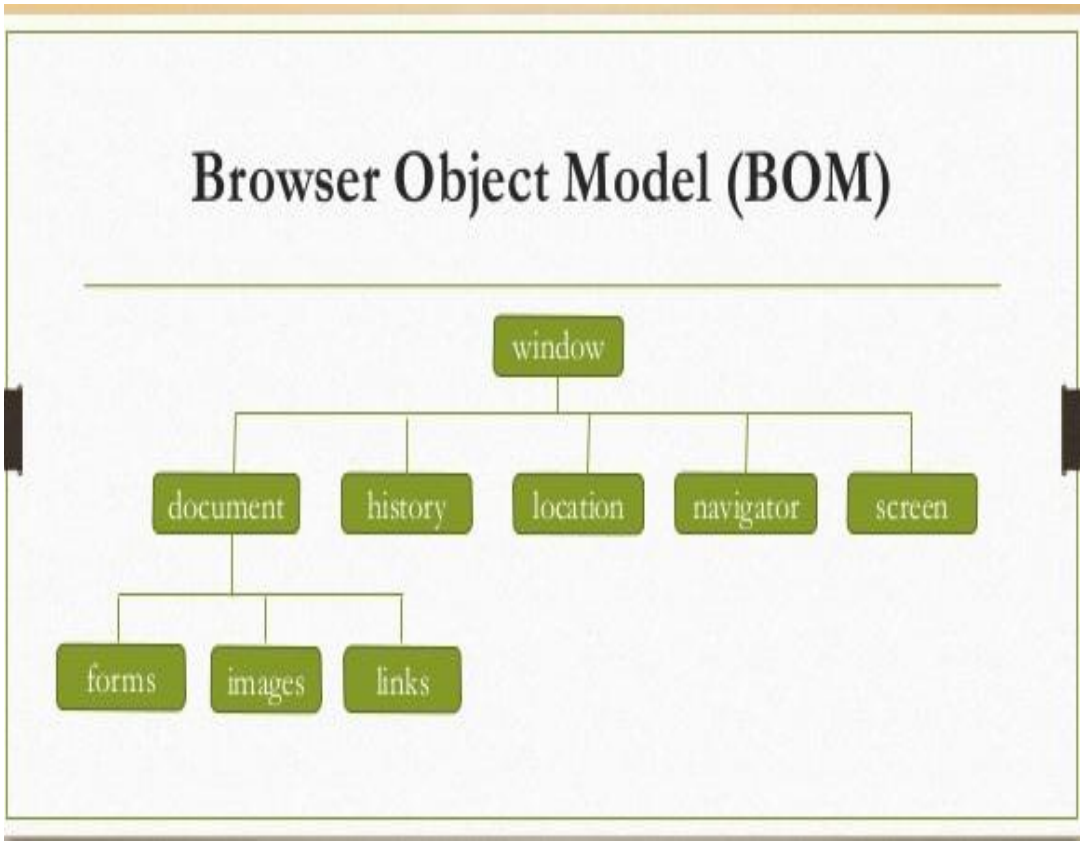
- The objects are described by **properties** and their behavior is defined by **methods**. An object is collection of these properties and methods which can be defined and altered and retrieved by the user.
- JavaScript objects are dynamic in nature, properties and methods can be added and deleted by the user. Each property or method is identified by the **name** that is mapped to a **value**.

The Browser Object:

- The Browser Object Model (BOM) is actually the central part of using JavaScript on the Web
- BOM - set of objects that comprise various elements of a Browser. The browser object model (BOM) is a hierarchy of browser objects that are used to manipulate methods and

properties associated with the Web browser itself.

- Web browser controlled through browser object model (BOM) whereas web page can be controlled through document object Model (DOM).
- Objects that make up the BOM include the window object, navigator object, screen object, history, location object, and the document object. The Document Object consists of objects that are used to manipulate methods and properties of the document or Web page loaded in the browser window.



The “window” Object represents the browser, and it is the default object.

Eg: document.write("a test message");

alert ("Hello");

has the same meaning as writing as follows:

window.document.write("a test message");

window.alert("Hello");

- Some of the methods of the window object are:

alert(), prompt(), confirm(): to read input and display output

open(): Create a new window

close(): close the current window
setTimeout(expression, time) : Evaluate "expression" after "time" (in millisecond).

- Some of the properties of the window object are:

location: Represents the URL loaded into the window
navigator: Contains info about the browser (**I**ts version, OS, etc.)
document: Holds the real content of the page
screen: Contains info about the client's display screen
history: Contains the visited URLs in the browser window

Eg: <!-- Opening a window with specified characteristics -->
<html>
<head>
<script type="text/JavaScript">
var myWin;

function open_close_win ()
{
    if (!myWin) // if not yet opened, open a new window
    myWin = window.open **(**

```
                "http://www.w3schools.com",       //
Document URL
                "my_new_window", // Window Name
                "toolbar=yes,location=yes,directories=n
+
                "status=no,menubar=yes,scrollbars=ye
+
                “resizable=no,copyhistory=yes,width=4
                        );
      else
      {                 // Otherwise close the opened
window
      myWin.close();
      myWin = null;
      }
}
</script>
</head>
<body>
<form>
<input type="button" value="Open/close Window"
onclick="open_close_win()">
</form>
</body>
</html>
```

Introduction to browser events

- Events are actions that can be detected by JavaScript, and the event object gives information about the event that has occurred.
- An event occurs when something happens in a browser window. The kinds of events that might occur are due to:
    - A document loading
    - The user clicking a mouse button
    - The browser screen changing size
- Events are normally used in combination with functions, and the function will not be executed before the event occurs! JavaScript event handlers are divided into two types:
    - Interactive event handlers - depend on user interaction with the HTML page; ex. clicking a button
    - Non-Interactive event handlers - do not need user interaction; ex. on load
- Events are *JavaScript code that are not added inside the <script> tags, but rather, inside the html tags,* that execute JavaScript when something happens, such as pressing a button, moving your mouse over a link, submitting a form etc.
- The basic syntax of these event handlers is:

- name_of_handler="JavaScript code here"

Event Handlers:
To react on events we can assign a *handler* – a function that runs in case of an event. Handlers is a way to run JavaScript code in case of user actions. There are several ways to assign a handler.

onclick:	Use this to invoke JavaScript upon clicking (a link, or form boxes)
onload:	Use this to invoke JavaScript after the page or an image has finished loading.
onunload:	Use this to invoke JavaScript right after someone leaves this page.
onmouseover:	Use this to invoke JavaScript if the mouse passes by some link
onmouseout:	Use this to invoke JavaScript if the mouse goes pass some link
onmousedown	The mouse button was pressed on the element
onmouseup	The mouse button was released on the element.
onkeydown	A key was pressed when an element has focus
onkeypress	A keystroke was received by the element
onkeyup	A key was released when the element has focus

onclick Event Type:

This is the most frequently used event type which occurs

when a user clicks the left button of his mouse. For instance, to assign a click handler for an input, we can use onclick. On mouse click, the code inside onclick runs. You can put your validation, warning etc., against this event type.

Ex:
```
<html>
  <head>
    <script type="text/javascript">
        function sayHello() {
          alert("Hello World")
        }
      </script>
    </head>
    <body>
    <p>Click the following button and see result</p>

    <form>
      <input type="button" onclick="sayHello()" value="Say
Hello" />
    </form>
    </body>
</html>
```

Onmousedown and onmouseup events:

The onmousedown, onmouseup, and onclick events are all parts of a mouse-click. First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is triggered, finally, when the mouse-click is completed, the onclick event is triggered.

```
<!DOCTYPE html>
<html>
<head>
<script>
```

```
function lighton() {
   document.getElementById('myimage').src = "bulbon.gif";
}
function lightoff() {
   document.getElementById('myimage').src = "bulboff.gif";
}
</script>
</head>
<body>
<img id="myimage" onmousedown="lighton()"
onmouseup="lightoff()" src="bulboff.gif" width="100" height="180"
/>
<p>Click mouse and hold down!</p>
</body>
</html>
```

The onload and onunload Events:

The onload and onunload events are triggered when the user enters or leaves the page. The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information. The onload and onunload events can be used to deal with cookies.

Ex:

```
<!DOCTYPE html>
<html>
<head>

<script>
function mymessage() {
   alert("This message was triggered from the onload event");
}
</script>
</head>

<body onload="mymessage()">
```

</body>

</html>

Form and form object in JavaScript:

Forms:

- A web form, also called an HTML form, is an online page that allows for user input. It is an interactive page that mimics a paper document or form, where users fill out particular fields.
- HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.
- Forms contain special elements called controls like input box, check boxes, radio-buttons, submit buttons, etc.
- Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for processing.
- The <form>… … </form> tags are used to create an HTML form.

JavaScript is widely used for form validation and to alter the default behavior of standard form controls.

Eg : A simple program to create a form with text boxes, radio buttons and submit button.

```
<body>
<FORM >
  <P>      First name: <INPUT type="text" name="firstname"><BR>
        Last name: <INPUT type="text" name="lastname"><BR>
        Email: <INPUT type="text" name="email"><BR>
        <INPUT type="radio" name="gender" value="Male"> Male<BR>
        <INPUT type="radio" name="gender" value="Female"> Female<BR>
        <BUTTN name="submit" value="submit" type="submit">
        Send <IMG src="/icons/wow.gif" alt="wow"></BUTTON>
        <BUTTON name="reset" type="reset">
        Reset<IMG src="/icons/oops.gif" alt="oops"></BUTTON>
  </P>
</FORM>
</body>
```

The Form object:

The Utility of JavaScript in forms is to validate the data (**data validation)** before it gets sent to server script for processing of data. Before submitting the data to the server data should be validated i.e., it should be checked for the correctness of the data. Once the form has been validated, the same script can be used to forward the data on to the server. It is used to check for empty form fields, improperly filled forms, verify the correct format of email address, credit card no, zipcode, telephone number etc.

JavaScript can also be used to submit the form on behalf of the user, using **form object** and its methods, properties. It enables users to handle multiple forms, call function to handle events, respond to various form related events etc.

- The "form" object belongs to the "document" object.
- Contains other objects that represent the form elements (text input field, radio buttons)
- This corresponds to an HTML input form constructed with the FORM tag.
- A form can be submitted by calling the JavaScript submit method or clicking the form submit button.

- **Form Object Methods**
  - reset() - Used to reset the form elements to their default values.
  - submit() - Submits the form as though the submit button were pressed by the user.
- **Form Object Properties**
  - action - This specifies the URL and CGI script file name the form is to be submitted to. It allows reading or changing the ACTION attribute of the HTML FORM tag.
  - length - The number of fields in the elements array. I.E. the length of the elements array.
  - method - This is a read or write string. It has the value "GET" or "POST".
  - name - The form name. Corresponds to the FORM Name attribute.
  - target - The name of the frame or window the form submission response is sent to by the server. Corresponds to the FORM TARGET attribute.
- **Form Events**
  - onReset
  - onSubmit
- Form elements can be accessed as : document.forms[idx] or document.forms[form_name] or document.form_name or document.getElementById(form_id)

```
<!– Validate the range of input in a text field -->
<html><head>
<script type="text/javascript">
function validate() {
        var x = document.myForm;
        var txt = x.myInput.value;
if (txt >= 1 && txt <= 5)
        return true;
else {
        alert("Must be between 1 and 5");
        return false;
}
```

```
}
</script>
</head>
<body>
<form name="myForm" action="tryjs_submitpage.htm"
onsubmit="return validate()"
>
Enter a value (1-5):
<input type="text" name="myInput" size="20">
<input type="submit" value="Submit">
</form></body></html>
```

UNIT-IV
PHP

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

Common uses of PHP:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Basic PHP Syntax

A PHP script can be placed anywhere in the document.
A PHP script starts with <?php and ends with ?>:

```
<?php
// PHP code goes here
?>
```

The default file extension for PHP files is ".php".
A PHP file normally contains HTML tags, and some PHP scripting code.
Example

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>
```

```
</body>
</html>
```

PHP 5 echo and print Statements

- In PHP there are two basic ways to get output: echo and print. echo and print are more or less the same. They are both used to output data to the screen.
- The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

The PHP echo Statement

- The echo statement can be used with or without parentheses like echo or echo( ) to display text.
- The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

Example

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

Example

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;
```

```
echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;
?>
```

The PHP print Statement

The print statement can be used with or without parentheses like print or print( ) to display text.

The following example shows how to output text with the print command (notice that the text can contain HTML markup):

Example

```
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

Display Variables: The following example shows how to output text and variables with the print statement:

Example

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

print "<h2>" . $txt1 . "</h2>";
print "Study PHP at " . $txt2 . "<br>";
print $x + $y;
?>
```

PHP 5 Form Handling:

The dynamic websites provide the functionalities that can use to store, update, retrieve, and delete the data in a database. Form is a

Document that containing black fields, that the user can fill the data or user can select the data. Forms are used to collect the data and casually the data will store in the data base. The PHP super-global $_GET and $_POST are used to collect form-data.

PHP - A Simple HTML Form

The example below displays a simple HTML form with two input fields and a submit button:

Example

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method. To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

The output could be something like this:

```
Welcome John
Your email address is john.doe@example.com
```

The same result could also be achieved using the HTTP GET method:
Example

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

Run example »

and "welcome_get.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.

GET:

Information sent from a form with the GET method is visible to everyone. GET also has limits on the amount of information to send.

The limitation is about 2000 characters. GET may be used for sending non-sensitive data. However, because the variables are displayed in the URL, it is possible to bookmark the page.

POST:

Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send. Developers prefer POST for sending form data. However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

GET vs. POST

- Both GET and POST create an array (e.g. array(key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.
- Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
- $_GET is an array of variables passed to the current script via the URL parameters.
- $_POST is an array of variables passed to the current script via the HTTP POST method.

Form Validation:

- Validation means check the input submitted by the user. There are two types of validation are available in PHP. They are as follows −
- Client-Side Validation − Validation is performed on the client machine web browsers.
- Server Side Validation − After submitted by data, The data has sent to a server and perform validation checks in server

machine.

- Some of Validation rules for field

Validate Form Data With PHP
The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.
When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:
<script>location.href('http://www.hacked.com')</script>

PHP Form Validation Example

* required field
Top of Form
Name: *
E-mail: *
Website:
Comment:
Gender: Female Male Other *

Bottom of Form

The validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

The HTML code of the form looks like this:

```
<form method="post" action="<?
php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

When the form is submitted, the form data is sent with method="post".

What is the $_SERVER["PHP_SELF"] variable?

The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.
So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

What is the htmlspecialchars() function?
The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

Validate Form Data With PHP

- The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.
- When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:
- <script>location.href('http://www.hacked.com')</script>
  - this would not be executed, because it would be saved as HTML escaped code, like this:

<script>location.href('http://www.hacked.com')</script>
The code is now safe to be displayed on a page or inside an e-mail.

- We will also do two more things when the user submits the form:
- Strip unnecessary characters (extra space, tab, newline) from

the user input data (with the PHP trim() function)

- Remove backslashes (\) from the user input data (with the PHP stripslashes() function)
- The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).
- We will name the function test_input().
- Now, we can check each $_POST variable with the test_input() function, and the script looks like this:

Example

```
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

URL and E-Mail Validation in PHP:

You can validate data in different ways. We have used PHP functions and regular expressions to create the validation rules.

Email address validation

We need to check if the email field is empty. If it is empty, an error message will be displayed. This message will be stored in the variable $email_error. We have used a PHP function called filter_var() to validate the email address entered by the user.

The easiest and safest way to check whether an email address is well-formed is to use PHP's filter_var() function.

In the code below, if the e-mail address is not well-formed, then store an error message:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL))
{
  $emailErr = "Invalid email format";
}
```

Website URL validation

The error message for the URL is stored in the variable $url_error. A regular expression validates the website URL entered through the contact form. Examine the following code to understand how we have performed the URL validation using PHP.

The code below shows a way to check if a URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

```
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?
=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "Invalid URL";
}
```

PHP Installation using Web Host:

To start using PHP, you can:

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host with PHP Support:

If your server has activated support for PHP you do not need to do anything. Just create some .php files, place them in your web directory, and the server will automatically parse them for you. You do not need to compile anything or install any extra tools. Because PHP is free, most web hosts offer PHP support. However, if your server does not support PHP, you must:

- install a database, such as MySQL
- install a web server
- install PHP

Step 1: Install MySQL

- Install the MySQL database server on your PC. We will do this using the 'MSI' one-click installer for Windows. Go to http://dev.mysql.com/downloads/ and download the 'MySQL Installer for Windows'.
- Run the installation. Click...
  - Install MySQL products
  - Accept the license, Allow the version check (optional)
  - At 'Choose a Setup Type' accept the "Developer Default" and click Next
  - A number of downloads of required software may be identified. Click Execute and follow onscreen instructions to install them.
  - At 'Installation progress' screen, hit Execute - the MySQL

software will be installed.

- At 'Configuration overview' hit Next to go to the basic configuration screen.
- Accept all the defaults on the 'MySQL Server Configuration' and hit Next.
- On the password screen, supply a password for the 'root' (main administrator) user.
- On the Service details page, accept the defaults and hit Next and then Next a couple more times for the configuration progress.
- Click Finish.

- MySQL Workbench will open. Under Server Administration (right hand column, double click 'Local MySQL56' (or whatever you called it). A box should pop up asking for the root password. Enter the password you supplied.
- The server management screen should appear. You don't have to worry too much about this. It just shows the install is working.

Step 2: Install Apache

- Install the Apache web server on your PC. Go to http://www.apachelounge.com/download/. Scroll down the page until you find the download for the 'Apache 2.4 win32 binaries' and download. You need to be careful that the module dll in PHP matches the version of Apache you install. Apache won't load otherwise.
- Unzip the file into C:\. You should end up with a directory 'Apache24' (or whatever the latest version is).
- Find Start > All programs > Accessories > Command Prompt..... BUT, right click, and select 'Run as administrator'.

- Enter the following commands

```
cd \Apache24\bin
httpd -k install
httpd -k start
```

...you may well get a warning about the server name. Don't worry about it. Don't close this window, you will need it again in a minute.

- To test it worked type 'http://localhost' into your browser. You should get a
- screen up to the effect that Apache is installed and working.

Step 3: Install PHP

- Now install the PHP scripting language on your PC. Go to http://www.php.net/download. In the current stable release section click on link for Windows 5.x.x binaries and source. Scroll down to the newest 'Zip' for **VC14 x86 Thread Safe** PHP (again, the newest versions of PHP didn't have this but it shouldn't matter) and download. *Don't* be tempted to use the Microsoft Installer version; it won't work.
- Open the zip file and extract to C:\PHP\
- In a console window, type php -v to see if it worked. (You may need to set up your PATH. Alos, if you get weired error messages, or no error messages at all, read the bit on the left of http://windows.php.net/ where it talks about installing "C++ Redistributable for Visual Studio")

Step 4: Configure Apache and PHP

You now need to edit Apache's ***httpd.conf file***. In the file explorer navigate to C:\Apache24\conf\httpd.conf. Open it in Notepad. At the end of this file (or wherever you like if you want to be more

organized) add the following lines:

```
LoadModule php5_module "C:/PHP/php5apache2_4.dll"
AddHandler application/x-httpd-php .php
PHPIniDir C:/PHP
```

The version of the module file matters (2_4 in this case). It MUST match the Apache version installed.

In the same file. Search for the line starting **DirectoryIndex**. Change it as follows

```
DirectoryIndex index.php index.html
```

Now, navigate to C:\PHP, and copy php.ini-development to php.ini. Edit this file, find the following lines and modify them as follows (all should exist already):

```
memory_limit = 256M
post_max_size = 128M
upload_max_filesize = 128M
```

You need to specify the extensions required for Moodle. Find the 'Dynamic Extensions' section and change the following lines (uncomment and add the correct path):

```
extension=c:/php/ext/php_curl.dll
extension=c:/php/ext/php_gd2.dll
extension=c:/php/ext/php_intl.dll
extension=c:/php/ext/php_mbstring.dll
extension=c:/php/ext/php_mysqli.dll
extension=c:/php/ext/php_openssl.dll
```

```
extension=c:/php/ext/php_soap.dll
extension=c:/php/ext/php_xmlrpc.dll
```

(these are a minimum. You may need others - e.g. LDAP - for specific functions) ...and save.

Back in the 'cmd' window for Apache, you need to restart it to load your changes...

```
httpd -k restart
```

Step 5: Test your install

Navigate to C:\Apache24\htdocs and create a file called 'test.php'. I had to change a file explorer setting to create .php files - Organise > Folder and search options > View and then untick 'Hide extensions for known file types'.

In this file enter the single line...

```
<?php phpinfo();
```

And then, in your browser, navigate to http://localhost/test.php. You should see a screen with masses of information and the PHP logo at the top. Check a few lines down for 'Loaded Configuration File' and make sure it says c:\php\php.ini.

That's PHP and Apache all working :)

1.3. Download and Install PHP Manually

If you decide to download PHP and install it manually, the procedures in this section guide you the following tasks:

- Download PHP and the WinCache extension.
- Install PHP and WinCache.

- Add the PHP installation folder to the Path environment variable.
- Set up a handler mapping for PHP.
- Add default document entries for PHP.
- Test your PHP installation.

To keep this procedure simple, install the WinCache extension but do not configure it. You will configure and test WinCache in Step 2: Configure PHP Settings.

To download and install PHP and WinCache

1. Open your browser to Windows for PHP Download Page and download the PHP non-thread-safe zip package.
2. Download the WinCache extension from the List of Windows Extensions for PHP.
3. Extract all files in the PHP .zip package to a folder of your choice, for example `C:\PHP\`.
4. Extract the WinCache .zip package to the PHP extensions folder (\ext), for example `C:\PHP\ext`. The WinCache .zip package contains one file (Php_wincache.dll).
5. Open **Control Panel**, click **System and Security**, click **System**, and then click **Advanced system settings**.
6. In the **System Properties** window, select the **Advanced** tab, and then click **Environment Variables**.
7. Under **System variables**, select **Path**, and then click **Edit**.
8. Add the path to your PHP installation folder to the end of the **Variable value**, for example `;C:\PHP`. Click **OK**.
9. Open IIS Manager, select the hostname of your computer in the **Connections** panel, and then double-click **Handler Mappings**.
10. In the **Action** panel, click **Add Module Mapping**.
11. In **Request path**, type ***.php**.
12. From the **Module** menu, select FastCgiModule.
13. In the **Executable** box, type the full path to Php-cgi.exe, for example `C:\PHP\Php-cgi.exe`.
14. In **Name**, type a name for the module mapping, for example **FastCGI**.
15. Click **OK**.
16. Select the hostname of your computer in the **Connections** panel, and double-click **Default Document**.

17. In the **Action** panel, click **Add**. Type **Index.php** in the **Name** box, and then click **OK**.
18. Click **Add** again. Type **Default.php** in the **Name** box, and then click **OK**.

To test your PHP installation

1. Open a text editor, for example Notepad, as Administrator.
2. In a new file, type the following text: `<?php phpinfo(); ?>`
3. Save the file as `C:\inetpub\wwwroot\Phpinfo.php`.
4. Open a browser and enter the following URL: `http://localhost/phpinfo.php`

   A nicely formatted webpage is displayed showing the current PHP settings.

To configure the WinCache PHP extension

1. In Windows Explorer, open your PHP installation folder, for example `C:\PHP`.
2. Choose either the **php.ini - development** or **php.ini - production** file, and rename it **php.ini**.
3. In a text editor, open the php.ini file and added the following line at the end of the file: `extension = php_wincache.dll`.
4. Save and close the php.ini file.
5. Recycle the IIS Application Pools for PHP to pick up the configuration changes.

To view WinCache configuration and other PHP settings

1. Open a text editor.
2. In a new file, type the following text: `<?php phpinfo(); ?>`
3. Save the file as `c:\inetpub\wwwroot\phpinfo.php`.
4. Open a browser and enter the following URL: `http://localhost/phpinfo.php`

   A nicely formatted web page is displayed showing the current PHP settings. The WinCache settings appear in a section called **wincache**.

Warning: Delete the phpinfo.php file when it's no longer needed.

Unit V

Introduction:

XML, or Extensible Markup Language, is a markup language that is used to create our own tags. It was created by the World Wide Web Consortium (W3C) to overcome the limitations of HTML, the Hypertext Markup Language is the basis for all Web pages.

- XML is widely used in the era of web development. It is also used to simplify data storage and data sharing.
- XML is based on SGML -- Standard Generalized Markup Language.
- XML is designed to store and transport data.
- Xml was released in late 90's. It was created to provide an easy to use and store self-describing data.
- XML is not a replacement for HTML.
- XML is designed to be self-descriptive.
- XML is designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML is platform independent and language independent.
- XML is a markup language which is like HTML. XML and HTML both use tags. But there are **some differences between them.**
    - HTML was designed for how to display data. And XML was designed for how to store data.
    - HTML tags are predefined. But XML tags are not predefined. You must define your own tags
    - With XML, data can also be easily exchanged between computer and database systems because XML data is stored in text format, this makes it easier to export data from a system to an XML file, and then import it into another system.

Creating XML Documents

- An XML tag is text that begins with a < and ends with a >.
- Data is placed between a matching start and end tags, and is called the 'element content'

- **Syntax:**
  <tagname> element content</tagname>
- The structure of xml tags is as follows:

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

Example:

```
<student>
        <rno>123</rno>
        <name>Radha</name>
        <age>20</age>
        <mobilenumber>7777777</mobilenumber>
        <address>
            <hno>7-90-456</hno>
            <city>hyd<city>
            <state>Telangana</state>
        </address>
</student>
```

Here student is a root element. And rno, name, age, mobile number, address are child elements of student tag.

Invalid, valid and well-formed documents:

There are three kinds of XML documents:

Invalid documents don't follow the syntax rules defined by the XML specification. If a developer has defined rules for what the document can contain in a DTD or schema, and the document doesn't follow those rules, that document is invalid as well. (See Defining document content for a proper introduction to DTDs and schemas for XML documents.)

Valid documents follow both the XML syntax rules and the rules defined in their DTD or schema.

Well-formed documents follow the XML syntax rules but don't have a DTD or schema.

Defining Data for Web Applications:

- We have seen how developers can use XML to create

documents with self-describing data, lets see how people are using those documents to improve the Web. Here are a few key areas:

- XML simplifies data interchange. Because different organizations (or even different parts of the same organization) rarely standardize on a single set of tools, it can take a significant amount of work for applications to communicate. Using XML, each group creates a single utility that transforms their internal data formats into XML and vice versa. Best of all, there's a good chance that their software vendors already provide tools to transform their database records (or LDAP directories, or purchase orders, and so forth) to and from XML.
- XML enables smart code. Because XML documents can be structured to identify every important piece of information (as well as the relationships between the pieces), it's possible to write code that can process those XML documents without human intervention. The fact that software vendors have spent massive amounts of time and money building XML development tools means writing that code is a relatively simple process.
- XML enables smart searches. Although search engines have improved steadily over the years, it's still quite common to get erroneous results from a search. If you're searching HTML pages for someone named "Chip," you might also find pages on chocolate chips, computer chips, wood chips, and lots of other useless matches. Searching XML documents for <first-name> elements that contained the text Chip would give you a much better set of results.

Well- formed XML documents:

XML document is said to be a well formed XML document, if it satisfies xml rules (XML document preparation rules) and element naming rules.A "Well Formed" Extensible Markup Language document that conforms to the XML syntax rules.A "Valid" XML document is a "Well Formed" XML document that conforms to the rules of a Document Type Definition (DTD).

XML rules:(OR XML document preparation rules)

- A well-formed XML document must have a corresponding end tag for all of its start tags.
- A tag in xml is called as element and the content of that tag is called as element content.
- XML documents must contain one element that is the parent of all other elements. This element is called the root element.
- Nesting of elements within each other in an XML document must be proper.
- An XML document can contain only one root element. So, the root element of an xml document is an element which is present only once in an xml document and it does not appear as a child element within any other element.
- A XML document must contain root tags, an element in XML can has its child element.
- XML tags are case sensitive.
- XML attribute values must be quoted.
- An element can contain other elements. That is child and sub child elements.

Elements in XML:

XML elements can be defined as building blocks of an XML.

An element can contain: text, attributes, other elements or a mix of the above

Each XML document contains one or more elements, the scope of which are either delimited by start and end tags, or for empty elements, by an empty-element tag.

Syntax:

<element-name attribute1 attribute2>

....content

</element-name>

Where element-name is the name of the element. The name its case in the start and end tags must match. Attribute1, attribute2 are attributes of the element separated by white spaces. An attribute defines a property of the element. It associates a name with a value, which is a string of characters. An attribute is written as −name = "value"

Eg:
```
<book category="web">
   <title>Learning XML</title>
   <author>Erik T. Ray</author>
   <year>2003</year>
   <price>39.95</price>
</book>
```

Element Naming Rules:

XML elements must follow these naming rules:

- Element Names can contain letters, numbers, and other characters
- Element Names cannot start with a number or punctuation character
- Element Names cannot start with the letters xml (or XML, or Xml, etc)
- Element Names cannot contain spaces.
- Element Names are case sensitive.

XML attributes

An attributes in XML Stores additional information about element

- Each attribute must have a name and a value
- To declare attribute in XML use = operator vale must be enclosed in quotes
- Example:

<student course=“B.Sc”>
<rno>123</rno>

```
    <name>Radha</name>
    <age>20</age>
    <mobilenumber>7777777</mobilenumber>
</student>
```

In the above example course is an attribute of student element.

Comments

- XML uses exactly the same syntax as HTML for comments so that any text that is inserted between **<!--** and **-->**
- To add comments in xml file we will use the following syntax:

Syntax:

```
<!- - comment here - ->
```

Example:

```
<!- - This is my xml file - ->
```

Empty XML Elements

- Elements that do not enclose any child elements or textual data are called ‘empty elements’.
- Empty elements can have a normal closing tag, or can use the shorthand method that combines both tags by adding a closing slash at the end of the opening tag.
- The following example demonstrates both methods being used to indicate the source of the image files in the element photo.

Example 1: Which has a normal closing tag

```
<student>
   <rno>123<rno>
   <photo filename=“jsmith.jpg”></photo>
</student>
```

(OR)

Example 2: closing slash at the end of the opening tag

```
<student>
   <rno>123<rno>
   <photo filename=“jsmith.jpg”/>
</student>
```

XML declaration

- The start of the very first line of every XML document should

contain the XML declaration.

- This identifies the document to be a XML document and defines the version of XML.
- The XML declaration is stated in a special tag starts **<?** and ends with **?>.**
- Inside the tag, the element is named ‘xml’ to denote that it is a part of the XML specification.
- To define XML version we will use version attribute and that is assigned the version number as its value. The version number is 1.0. that is as follows,

  <?xml version=“1.0”?>
- The XML declaration may include an attribute called ‘standalone’ to specify if that document uses other documents. It has two values either ‘yes’ or ‘no’.

  <?xml version=“1.0”? standalone=“yes”>

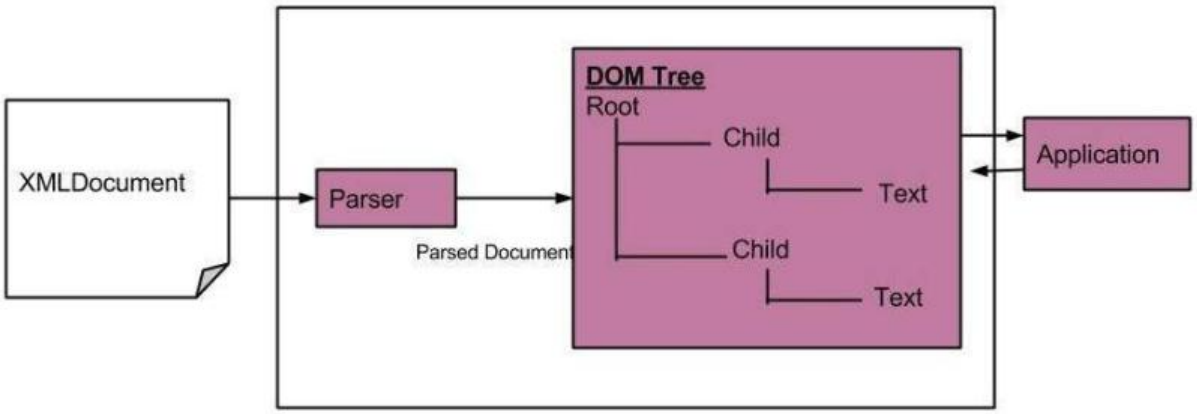
XML Document Object Model (DOM)

- The Document Object Model (DOM) is a W3C standard. The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.
- The DOM defines a standard for accessing and manipulating documents: *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*
- The XML DOM defines a standard way for accessing and manipulating XML documents. It presents an XML document as a tree-structure.
- We can access all elements through the DOM tree.
- We can modify or delete their content and also create new

elements. The elements, their content (text and attributes) are all known as nodes.

- XML DOM is a standard object model for XML. XML documents have a hierarchy of informational units called nodes;
    - Node object can have only one parent node object. This occupies the position above all the nodes
    - The parent node can have multiple nodes called as child nodes. These child nodes can have additional node called as attribute node
    - These child nodes in turn can have multiple child nodes. The text within the nodes is called as text node.
    - The node objects at the same level are called as siblings.

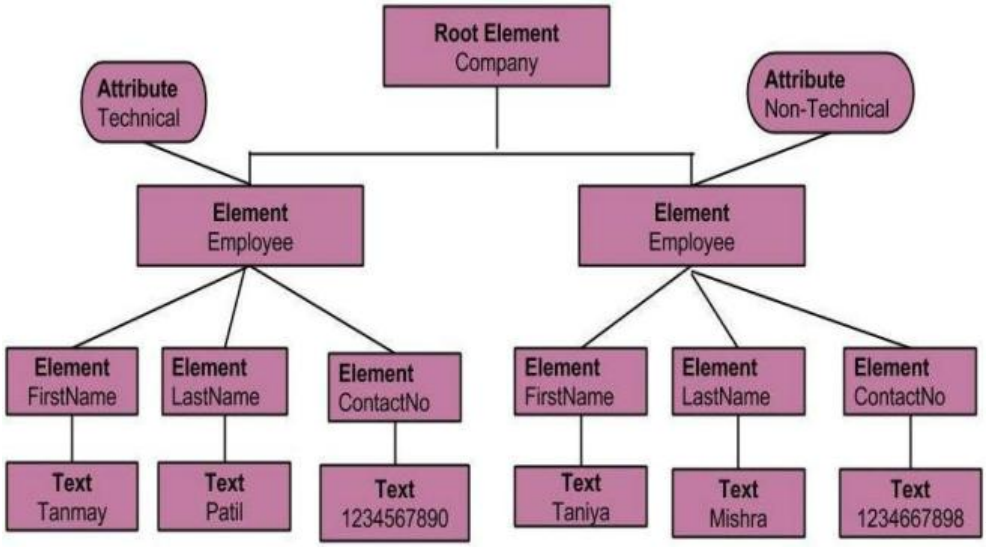
- It defines the objects and properties to access all XML elements. The DOM is separated into 3 different levels:
    - **Core DOM** - standard model for any structured document
    - **XML DOM** - standard model for XML documents
    - **HTML DOM** - standard model for HTML documents




```
<Company>
<Employeecategory="technical">
<FirstName>Tanmay</FirstName>
<LastName>Patil</LastName>
<ContactNo>1234567890</ContactNo>
</Employee>
<Employeecategory="non-technical">
```

```
<FirstName>Taniya</FirstName>
<LastName>Mishra</LastName>
<ContactNo>1234667898</ContactNo>
</Employee>
</Company>
```

The Document Object Model of the above XML document is:



Advantages of XML DOM

The following are the advantages of XML DOM.

- XML DOM is language and platform independent.
- XML DOM is **traversable** - Information in XML DOM is organized in a hierarchy which allows developer to navigate around the hierarchy looking for specific information.
- XML DOM is **modifiable** - It is dynamic in nature providing the developer a scope to add, edit, move or remove nodes at any point on the tree.

Disadvantages of XML DOM

- It consumes more memory (if the XML structure is large) as program written once remains in memory all the time until and unless removed explicitly.
- Due to the extensive usage of memory, its operational speed, compared to SAX is slower.

XML DTD (Data Type Definition):

- The XML Document Type Definition (DTD) is a way to describe XML language precisely.
- It defines the legal building blocks of an XML document. It is used to define document structure with a list of legal elements and attributes.
- DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language.
- An XML DTD can be either specified inside the document, or it can be kept in a separate document and then liked separately.
- Its main purpose is to define the structure of an XML document. It contains a list of legal elements and define the structure with the help of them.
- A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a DTD:

DTD Building Blocks:

1) Elements
2) Attributes
3) Entities
4) PCDATA
5) CDATA

1. Elements are the main building blocks of both XML and HTML documents.
   a. Eg: <body>some text</body>
      <message>some text</message>
2. Attributes provide extra information about elements.Attributes are always placed inside the opening tag of an element. Attributes always come in name/value pairs. The following "img" element has additional information about a source file:
   a.
3. Entities are expanded when a document is parsed by an XML parser.The following entities are predefined in XML:

Entity	Character	Meaning

&lt;	<	To display less than symbol
&gt;	>	To display less than symbol
&amp;	&	To display ampersand character
&quot;	"	To display " quotation mark
&apos;	'	To display ' apostrophe (single quote)

4. PCDATA means parsed character data.PCDATA is text that WILL be parsed by a parser. The text will be examined by the parser for entities and markup.Tags inside the text will be treated as markup and entities will be expanded.
5. CDATA means character data.CDATA is text that will NOT be parsed by a parser. Tags inside the text will NOT be treated as markup and entities will not be expanded.

Types of DTD:

There are 2 different types of DTD

1) Internal DTD
2) External DTD

1) **Internal DTD:**

A DTD is referred to as an internal DTD if elements are declared within the XML files. To refer it as internal DTD, *standalone* attribute in XML declaration must be set to **yes**. This means, the declaration works independent of an external source.

Syntax

```
<! DOCTYPE root-element [element-declarations]>
```

Where *root-element* is the name of root element and *element-declarations* is where you declare the elements.

Example: (XML document with an internal DTD)

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>HIIIIIIIIIII</body>
</note>
The DTD above is interpreted like this:

- !DOCTYPE note defines that the root element of this document is note
- !ELEMENT note defines that the note element must contain four elements: "to,from,heading,body"
- !ELEMENT to defines the to element to be of type "#PCDATA"
- !ELEMENT from defines the from element to be of type "#PCDATA"
- !ELEMENT heading defines the heading element to be of type "#PCDATA"
- !ELEMENT body defines the body element to be of type "#PCDATA"

2) **External DTD:**

In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal *.dtd* file or a valid URL. To refer it as external DTD, *standalone* attribute in the XML declaration must be set as **no**. This means, declaration includes information from the external source.

Syntax

```
<!DOCTYPE root-element SYSTEM "file-name">
```

where *file-name* is the file with *.dtd* extension.

Example:

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
```

```
  <body>Don't forget me this weekend!</body>
</note>
```

The DTD file as follows : (The file name as note.dtd)

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

DTD Entities

- Entities are used to define shortcuts to special characters.
- These are the place holders in XML.
- Entities can be declared internal or external.
- Entities are expanded when a document is parsed by an XML parser.
- Total 5 Character Entities
- The following entities are predefined in XML:
- There are few special characters or symbols which are not available to be typed directly from the keyboard. Character Entities can also be used to display those symbols/special characters.
- There are three types of character entities −
    - Predefined Character Entities
    - Numbered Character Entities
    - Named Character Entities
- The following are few character entities:

Entity	Character	Meaning
&lt;	<	To display less than symbol
&gt;	>	To display less than symbol
&amp;	&	To display ampersand character
&quot;	"	To display " quotation mark

&apos;	'	To display ' apostrophe (single quote)

Example:

<group>MPCS & MECS & MSCS</group>

Output:

<group>MPCS & MECS & MSCS</group>

XML Namespaces

XML Namespace Purpose:

XML Namespaces provide a method to avoid element name conflicts.

Name Conflicts:

In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.

Understanding namespaces (Example with explanation):

This XML carries HTML table information (a piece of table information):

<table>
<tr>
<td>**Apples**</td>
<td>**Bananas**</td>
</tr>
</table>

This XML carries information about a table (a piece of furniture information):

<table>
<name>**African Coffee Table**</name>
<width>**80**</width>
<length>**120**</length>
</table>

- If these XML were added together, there would be a name conflict.
- Both contain a <table> element, but the elements have different content and meaning.
- An XML application will not know how to handle these differences.
- To Solve the Name Conflict We have to use namespaces.

Solving the Name Conflict Using a Prefix:

- Name conflicts in XML can easily be avoided using a name prefix.

This XML carries information about an HTML table, and a piece of furniture:

```
<h:table>
      <h:tr>
            <h:td>Apples</h:td>
            <h:td>Bananas</h:td>
      </h:tr>
</h:table>
<f:table>
      <f:name>African Coffee Table</f:name>
      <f:width>80</f:width>
      <f:length>120</f:length>
</f:table>
```

In the example above, there will be no conflict because the two <table> elements have different names.

XML Namespaces - The xmlns Attribute: (or document name spaces)

- When using prefixes in XML (i.e. namespace) for the prefix must be defined.
- The namespace is defined by the **xmlns** attribute in the start tag of an element.
- The namespace declaration has the following syntax.

  xmlns: prefix="URI"
- **Example:**

```
<root>
 <h:table xmlns:h="http://www.w3.org/TR/fruits/">
       <h:tr>
            <h:td>Apples</h:td>
            <h:td>Bananas</h:td>
       </h:tr>
  </h:table>
   <f:table xmlns:f="http://www.w3schools.com/furniture/">
        <f:name>African Coffee Table</f:name>
        <f:width>80</f:width>
        <f:length>120</f:length>
   </f:table>
</root>
```

Explanation:

- In the example above, the xmlns attribute in the <table> tag give the h: and f: prefixes a qualified namespace.

- When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.

(OR)

We can declare all the namespaces at one place that is in root element.

```
<root  xmlns:h="http://www.w3.org/TR/html4/"
        xmlns:f="http://www.w3schools.com/furniture/">
<h:table>
          <h:tr>
            <h:td>Apples</h:td>
            <h:td>Bananas</h:td>
       </h:tr>
</h:table>


<f:table>
            <f:name>African Coffee Table</f:name>
          <f:width>80</f:width>
           <f:length>120</f:length>
  </f:table>
```