Concept of WWW.

- **WWW** is stands for World Wide Web.
- The **World Wide Web (WWW)** is a global information medium which users can read and write via computer connected to the internet.
- The Web, or World Wide Web, is basically a system of Internet servers that support specially formatted documents. The documents are formatted in a markup language called HTML (Hypertext Markup Language) that supports links to other documents, as well as graphics, audio, and video files.
- In short, **World Wide Web (WWW)** is collection of text pages, digital photographs, music files, videos, and animations you can access over the Internet.
- Web pages are primarily text documents formatted and annotated with Hypertext Markup Language (HTML). In addition to formatted text, web pages may contain images, video, and software components that are rendered in the user's web browser as coherent pages of multimedia content.
- The terms Internet and World Wide Web are often used without much distinction. However, the two are not the same.
- The Internet is a global system of interconnected computer networks. In contrast, the World Wide Web is one of the services transferred over these networks. It is a collection of text documents and other resources, linked by hyperlinks and URLs, usually accessed by web browsers, from web servers.
- There are several applications called **Web browsers** that make it easy to access the World Wide Web; For example: Firefox ,Microsoft's Internet Explorer, Chrome Etc.
- Users access the World-Wide Web facilities via a client called a browser, which provides transparent access to the WWW servers. User can access WWW via two way such us :

History of WWW:

- Tim Berners-Lee, in 1980 was investigating how computer could store information with random links. In 1989, while working at European Particle Physics Laboratory, he proposed to idea of global hypertext space in which any network-accessible information could be referred to by single **"universal Document Identifier".** After that in 1990, this idea expanded with further program and knows as **World Wide Web.**

Internet and WWW

- The Internet, linking your computer to other computers around the world, is a way of transporting content. The Web is software that lets you use that content…or contribute your own. The Web, running on the mostly invisible Internet, is what you see and click on in your computer's browser.

What is The Internet?

- The Internet is a massive network of networks, a networking infrastructure. It connects millions of computers together globally, forming a network in which any computer can communicate with any other computer as long as they are both connected to the Internet. Information that travels over the Internet does so via a variety of languages known as

protocols. So we can says that Internet is network of computer which connect to together and any computer communicate with any other computer.

What is The Web (World Wide Web)?

- The World Wide Web, or simply Web, is a way of accessing information over the medium of the Internet. It is an information-sharing model that is built on top of the Internet.
- The Web uses the HTTP protocol, only one of the languages spoken over the Internet, to transmit data. The Web also utilizes browsers, such as Internet Explorer or Firefox, to access Web documents called Web pages that are linked to each other via hyperlinks. Web documents also contain graphics, sounds, text and video.

Different between Internet and WWW

- **The Web** is a Portion of The Internet. The Web is just one of the ways that information can be disseminated over the Internet. **The Internet**, not the Web, is also used for email, which relies on SMTP, Usenet news groups, instant messaging and FTP. So the Web is just a portion of the Internet.

HTTP Protocol: Request and Response.

- HTTP stands for Hypertext Transfer Protocol.
- HTTP is based on the client-server architecture model and a stateless request/response protocol that operates by exchanging messages across a reliable TCP/IP connection.
- An HTTP "client" is a program (Web browser) that establishes a connection to a server for the purpose of sending one or more HTTP request messages. An HTTP "server" is a program (generally a web server like Apache Web Server) that accepts connections in order to serve HTTP requests by sending HTTP response messages.
- Errors on the Internet can be quite frustrating — especially if you do not know the difference between a 404 error and a 502 error. These error messages, also called HTTP status codes are response codes given by Web servers and help identify the cause of the problem.
- For example, "404 File Not Found" is a common HTTP status code. It means the Web server cannot find the file you requested. The file -- the webpage or other document you try to load in your Web browser has either been moved or deleted, or you entered the wrong URL or document name.
- HTTP is a stateless protocol means the HTTP Server doesn't maintain the contextual information about the clients communicating with it and hence we need to maintain sessions in case we need that feature for our Web-applications
- HTTP header fields provide required information about the request or response, or about the object sent in the message body. There are four types of HTTP message headers:
  - **General-header:**

  These header fields have general applicability for both request and response messages.
  - **Request-header:**

  These header fields have applicability only for request messages.
  - **Response-header:**

  These header fields have applicability only for response messages.
  - **Entity-header:**

  These header fields define Meta information about the entity-body.

- As mentioned, whenever you enter a URL in the address box of the browser, the browser translates the URL into a request message according to the specified protocol; and sends the request message to the server.
- For example, the browser translated the **URL http://www.test101.com/doc/index.html** into the following request message:

```
GET /docs/index.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

Here,Step by step communication between client and server mention into followingfigure.

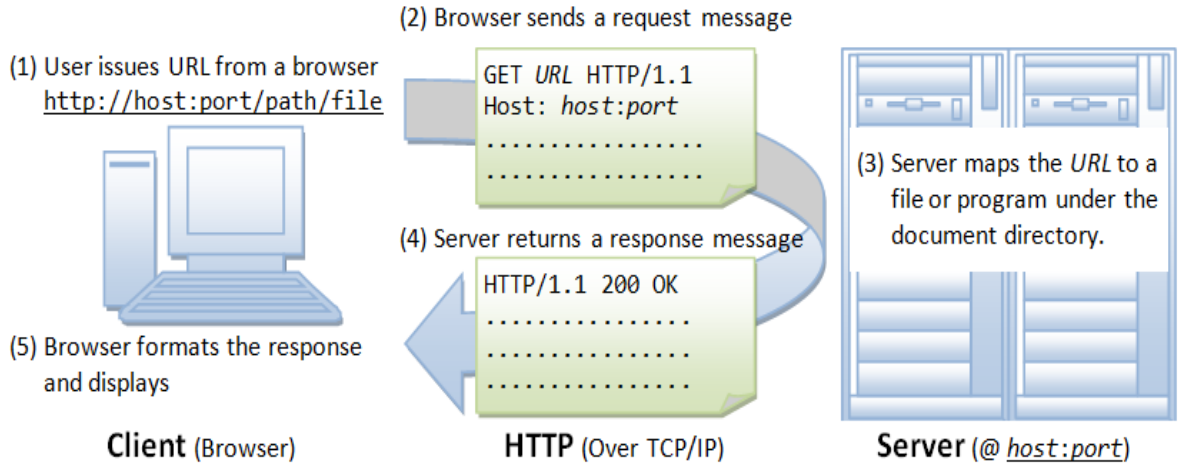


Fig 1: Communication between HTTP Client and HTTP Server

Web Browser and Web Server.

- Web server and web browser are the terms which are commonly used for website. The basic purpose of both is to develop a platform for internet web directory. So that any users can anytime access any kind of website. Major difference between them is on their function and how they perform their functions. Check for the detail of both topics before understanding the differences between them.

Web Browser

- Web browser is a client, program, software or tool through which we sent HTTP request to web server. The main purpose of web browser is to locate the content on the World Wide Web and display in the shape of web page, image, audio or video form.

- We can also call it a client server because it contacts the web server for desired information. If the requested data is available in the web server data then it will send back the requested information again via web browser.
- Microsoft Internet Explorer, Mozilla Firefox, Safari, Opera and Google Chrome are examples of web browser and they are more advanced than earlier web browser because they are capable to understand the HTML, JavaScript, AJAX, etc. Now days, web browser for mobiles are also available, which are called micro browser.

Web Server

- **Web server** is a computer system, which provides the web pages via HTTP (Hypertext Transfer Protocol). IP address and a domain name is essential for every web server.
- Whenever, you insert a URL or web address into your web browser, this sends request to the web address where domain name of your URL is already saved. Then this server collects the all information of your web page and sends to browser, which you see in form of web page on your browser.
- Lot of web server software is available in the market in shape of NCSA, Apache, Microsoft and Netscape. Storing, processing and delivering web pages to clients are its main function. All the communication between client (web browser) and server takes place via HTTP.
- Here, we can easily understand concept of web browser and web server by following figure.

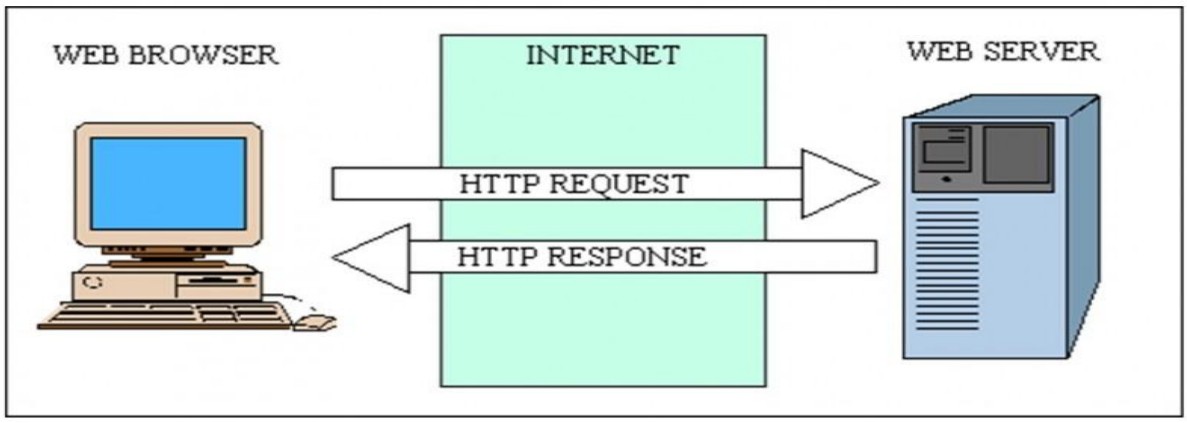


Fig 2: Communication between web Browser and Web Server

Feature of Web 2.0.

- Web 2.0 is term that was introduced in 2004 and refers to the second generation of the World Wide Web.The term "2.0" comes from the software industry, where new versions of software programs are labeled with an incremental version number.
- Some examples of features considered to be part of Web 2.0 are listed below:
  - **Blogs :**

    It also known as Web logs, these allow users to post thoughts and updates about their life on the Web.
  - **Wikis:**

    Wikis - sites like Wikipedia and others enable users from around the world to add and update online content.
  - **Social Networking:**

Sites like Facebook and MySpace allow users to build and customize their own profile sand communicate with friends.

- **Web Application:**

Web application is a broad range of new applications make it possible for users to run programs directly in a Web browser.as Web logs, these allow users to post thoughts and updates about their life on the Web.

- **User Participation:**

In traditional web the contents are solely provider by the web site owner or company, but in web 2.0 the users participate in content sourcing. This is also known as Crowd sourcing.
Examples: Wikipedia & You Tube.

- **Long Tail:**

The traditional web was like a retail business the product is sold directly to user and the revenue generated. But in web 2.0 the niche product is not sold directly but offered as a service on demand basis and income is generated as monthly fee and pay per consumption.

- **Rich User Experience :**

Traditional web are built with HTML and CSS CGI and had been offered as a static page. On the other hand Web 2.0 uses AjaxAsynchronous JavaScript + XML) presenting dynamic, rich user experience to users.
Example: Google Provided Google Maps and Google Suggest.

- Web 2.0 technologies provide a level user interaction that was not available before. Websites have become much more dynamic and interconnected, producing "online communities" and making it even easier to share information on the Web. Because most Web 2.0 features are offered as free services, sites like Wikipedia and Facebook have grown at amazingly fast rates.

Web Design Issues

Browser & Operating Systems

- Web pages are written using different HTML tags and viewed in browser window.
- The different browsers and their versions greatly affect the way a page is rendered, as different browsers sometimes interpret same HTML tag in a different way.
- Different versions of HTML also support different sets of tags.
- The support for different tags also varies across the different browsers and their versions.
- Same browser may work slightly different on different operating system and hardware platform.
- To make a web page portable, test it on different browsers on different operating systems.

Bandwidth and Cache

- Users have different connection speed, i.e. bandwidth, to access the Web sites.
- Connection speed plays an important role in designing web pages, if user has low bandwidth connection and a web page contains too many images, it takes more time to download.
- Generally, users have no patience to wait for longer time than 10-15 seconds and move to other site without looking at contents of your web page.
- Browser provides temporary memory called *cache* to store the graphics.
- When user gives the URL of the web page for the first time, HTML file together with all the graphics files referred in a page is downloaded and displayed.

Display Resolution

- Display resolution is another important factor affecting the Web page design, as we do not have any control on display resolution of the monitors on which user views our pages.
- Display or screen resolution is measured in terms of pixels and common resolutions are 800 X 600 and 1024 X 786.
- We have three choices for Web page design.
    - Design a web page with fixed resolution.
    - Make a flexible design using HTML table to fit into different resolution.
    - If the page is displayed on a monitor with a higher resolution, the page is displayed on left-hand side and some part on the right-hand side remains blank. We can use centered design to display page properly.
    - (Not For Exam) ~~Ideally we should use some frameworks for designing like Bootstrap/Material design.~~

Look & Feel

- Look and feel of the website decides the overall appearance of the website.
- It includes all the design aspects such as
    - Web site theme
    - Web typography
    - Graphics
    - Visual structure
    - Navigation etc…

Page Layout and Linking

- Website contains of individual web pages that are linked together using various navigational links.
- Page layout defines the visual structure of the page and divides the page area into different parts to present the information of varying importance.
- Page layout allows the designer to distribute the contents on a page such that visitor can view it easily and find necessary details.

Locating Information

- Webpage is viewed on a computer screen and the screen can be divided into five major areas such as center, top, right, bottom and left in this particular order.
- The first major area of importance in terms of users viewing pattern is the center, then top, right, bottom and left in this particular order.

Making Design user-Centric

- It is very difficult for any Web designer to predict the exact behavior of the Web site users.
- However, idea of general behavior of common user helps in making design of the Web site user-centric.
- Users either scan the information on the web page to find the section of their interest or read the information to get details.

Sitemap

- Many a times Web sites are too complex as there are a large number of sections and each section contains many pages.
- It becomes difficult for visitors to quickly move from one part to other.
- Once the user selects a particular section and pages in that section, user gets confused about where he/she is and where to go from there.
- To make it simple, keep your hierarchy of information to few levels or provide the navigation bar on each page to jump directly to a particular section.

Tips for Effective Navigation.

- Navigation links are either text based, i.e. a word or a phrase is used as a link, or graphical, i.e. a image, i.e. a icon or a logo is used as a link.
- Navigation links should be clear and meaningful.
- It should be consistent.
- Link should be understandable.
- Organize the links such that contents are grouped logically.
- Provide search link, if necessary, usually on top of the page. Use common links such as ‘about us’ or ‘Contact us’.
- Provide the way to return to first page.
- Provide the user with information regarding location
- Horizontal navigation bar can be provided on each page to directly jump to any section

HTML

What is HTML?

- Stands for Hypertext Markup Language.
- Most documents that appear on the World Wide Web were written in HTML.
- HTML is a markup language, not a programming language. In fact, the term HTML is an acronym that stands for Hypertext Markup Language.
- We can apply this markup language to your pages to display text, images, sound and movie files, and almost any other type of electronic information.
- We use the language to format documents and link them together, regardless of the type of computer with which the file was originally created.

HTML Elements

- An element consists of three basic parts: an opening tag, the element's content, and finally, a closing tag.

* - opening paragraph tag*
Element Content - paragraph words
* - closing tag*

- Every (web) page requires four critical elements: the html, head, title, and body elements.

1. <html> Element…</html>

- <html> begins and ends each and every web page.
- Its purpose is to encapsulate all the HTML code and describe the HTML document to the web browser.

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

2. <head> Element

- The <head> element is "next" as they say. As long as it falls somewhere between your <html> tag and your web page content (<body>).
- The head functions "behind the scenes." Tags placed within the head element are not directly displayed by web browsers.
- We will be placing the <title> element here.
- Other elements used for scripting (JavaScript) and formatting (CSS) will eventually be introduced and you will have to place them within your head element.

```
<html>
        <head>
        </head>
</html>
```

3. The <title> Element

- Place the <title> tag within the <head> element to title your page.
- The words you write between the opening and closing <title></title> tags will be displayed at the top of a viewer's browser.

```
<html><head><title>My WebPage!</title></head></html>
```

4. The <body> Element

- The <body> element is where all content is placed. (Paragraphs, pictures, tables, etc).
- The body element will encapsulate all of your webpage's viewable content.

```
<html>
<head><title>My WebPage!</title></head>
<body>
Hello World! All my content goes here!
</body>
</html>
```

HTML Tags

- A web browser reads an HTML document top to bottom, left to right.
- Each time the browser finds a tag, it is displayed accordingly (paragraphs look like paragraphs, tables look like tables, etc).
- Tags have 3 major parts: opening tag(s), content(s), and closing tag(s).
- Recall that a completed tag is termed an element.

1. Paragraph Tag <p>

- The <p> tag defines a paragraph. Using this tag places a blank line above and below the text of the paragraph.

```
<p>Avoid losing floppy disks with important school...</p>
<p>For instance, let's say you had a HUGE school...</p>
```

2. HTML - Headings 1:6

- A heading in HTML is just what we might expect, a title or subtitle.
- By placing text inside of <h1> (heading) tags, the text displays bold and the size of the text depends on the number of heading (1-6).
- Headings are numbered 1-6, with 1 being the largest heading and 6 being the smallest.

```
<html><body>
<h1>This is heading 1</h1><h2>This is heading 2</h2><h3>This is heading 3</h3>
<h4>This is heading 4</h4><h5>This is heading 5</h5><h6>This is heading 6</h6>
</body></html>
```

3. Line Breaks

- Line breaks are different then most of the tags we have seen so far. A line break ends the line you are currently on and resumes on the next line.

```
<p>Darshan<br />
Computer<br /></p>
```

HTML Lists

- There are 3 different types of lists.
- A <ol> tag starts an ordered list, <ul> for unordered lists, and <dl> for definition lists.
    1. - unordered list; *bullets*
    2. - ordered list; *numbers*
    3. <dl> - definition list; *dictionary*

1. HTML Ordered Lists

- Use the <ol> tag to begin an ordered list. Place the <li> (list item) tag between your opening <ol> and closing tags to create list items.
- Ordered simply means numbered, as the list below demonstrates.

```
<ol>
<li>Find a Job</li>
<li>Move Out</li>
</ol>
```

- Start your ordered list on any number besides 1 using the start attribute.

```
<ol start="4" >
<li>Buy Food</li>
<li>Get a Degree</li>
</ol>
```

- There are 4 other types of ordered lists. Instead of generic numbers you can replace them with Roman numerals or letters, both capital and lower-case. Use the type attribute to change the numbering.

```
<oltype="a">
<oltype="A">
<oltype="i">
<ol type="I">
</ol>
```

2. HTML Unordered Lists

- Create a bulleted list with the <ul> tag. The bullet itself comes in three subtypes: squares, discs, and circles.
- The default bullet displayed by most web browsers is the traditional full disc.

```
<ul>
<li>Milk</li>
<li>Chocolate</li>
</ul>
```

- There are 3 other types of unordered lists.

```
<oltype="square">
<ol type="disc">
<oltype="circle">
</ol>
```

3. HTML Definition Term Lists

- Make definition lists as seen in dictionaries using the <dl> tag. These lists displace the term word just above the definition itself for a unique look. It's wise to bold the terms to displace them further.
  - <dl> - defines the start of the list
  - <dt> - definition term
  - <dd> - defining definition

```
<dl>
<dt><b>Fromage</b></dt>
<dd>French word for cheese.</dd>
<dt><b>Voiture</b></dt>
<dd>French word for car.</dd>
</dl>
```

4. HTML Nested Lists

- You can also nest one list within another, so you could make an unordered list inside a numbered one:

```
<html>
<ol>
        <li> Clear out garage</li>
        <ul>
                <li> Tomatoes</li>
</ul>
        <li> repair fence </li>
</ol>
</html>
```

HTML - Formatting Elements

- Several tags exist to further amplify text elements. These formatting tags can make text bold, italic, sub/superscripted, and more.

Tag	Description	Example
<b>	The <b> tag specifies bold text.	<**b**>Bold Text</**b**>
<i>	The <i> tag specifies italic text.	<**i**>Italic Text</**i**>
	The tag specifies emphasis text	<**em**>Emphasized Text</**em**>
<sup>	The <sup> tag defines superscript text. Superscript text appears half a character above the baseline. Superscript text can be used for footnotes, like WWW [1].	This text contains <**sup**>superscript</**sup**> text.
<sub>	The <sub> tag defines subscript text. Subscript text appears half a character below the baseline. Subscript text can be used for chemical formulas, like H_2O.	An example of <**sub**>subscripted Text</**sub**>
<tt>	The <tt> tag defines teletype text.	<**tt**>This text is teletype text.</**tt**>
<blink>	The <blink> tag is used for blinking the text.	<**blink**> blinking text tag</**blink**>

HTML Color Coding System - Color Names

There are 3 different methods to set color.

- We can set color using three methods.
  a. Using color name

```
<body bgcolor="red">
<font color="red">
```

  b. Using RGB(Red, Green, Blue) value

```
<body bgcolor="rgb(72,0,0)">
<font color="rgb(72,0,0)">
```

  c. Using Hexadecimal value

```
<body bgcolor="#ffff00">
<font color="#ffff00">
```

HTML - Font and Basefont

- The <font> tag is used to add style, size, and color to the text on your site. Use the size, color, and face attributes to customize your fonts.
- Use a <basefont> tag to set all of your text to the same size, face, and color.

1. Font Size

- Set the size of your font with size. The range of accepted values is from 1(smallest) to 7(largest).The default size of a font is 3.

```
<p><font size="5">Here is a size 5 font</font></p>
```

2. Font Color

- Set the color of your font with color.

```
<font color="#990000">This text is hexcolor #990000</font><br />
<font color="red">This text is red</font>
```

3. Font Face

- Choose a different font face using any font you have installed.

```
<p><font face="Bookman Old Style, Book Antiqua, Garamond">This paragraph has
had its font...</font></p>
```

4. Basefont - Set a Solid Base

- With the basefont tag you will be able to set the default font for your web page.

HTML Code:

```
<html><body>
<basefont size="2" color="green"><p>This paragraph has had its font...</p>
</basefont>
</body></html>
```

HTML - Hypertext Reference (href) or Hyperlinks

- The href attribute defines reference that the link refers to. Basically this is where the user will be taken if they wish to click this link.
- Use the <a> tags to define the start and ending of an anchor.
- Decide what type of href attribute you need and place this attribute into the opening tag.
- The text you place between the opening and closing tags will be shown as the link on a page. Use the demonstration below as a reference.
- Hypertext references can be Internal, Local, or Global.
- **Internal** - Links to anchors on the current page
- **Local** - Links to other pages within your domain
- **Global** - Links to other domains outside of your site

```
Internal - href="#anchorname"
Local - href="../pics/picturefile.jpg"
Global - href=http://www.xyz.com/
```

```
<a href="http://www.google.com/" target="_blank" >Google Home</a>
<a href="http://www.espn.com/" target="_blank" >ESPN Home</a>
<a href="http://www.yahoo.com/" target="  blank" >Yahoo Home</a>
```

Link Targets

- The target attribute defines whether to open the page in a separate window, or to open the link in the current browser window.

HTML Code:	
target=" _blank"	Opens new page in a new browser window
target=" _self"	Loads the new page in current window
target=" _parent"	Loads new page into a frame that is superior to where the link lies
target=" _top"	Loads new page into the current browser window, cancelling all frames

Anchors

- To link to sections of your existing page a name must be given to the anchor.
- In the example below, we've created a mini Table of Contents for this page.
- By placing blank anchors just after each heading, and naming them, we can then create reference links to those sections on this page as shown below.
- First, the headings of this page contain blank, named anchors. They look like this.

```
<h2>HTML Links and Anchors<a name="top"></a></h2>
<h2>HTML Text Links<a name="text"></a></h2>
<h2>HTML Email<a name="email"></a></h2>
```

- Now create the reference links, placing the # symbol followed by the name of the anchor in the href of the new link.

```
<a href="#top">Go to the Top</a>
<a href="#text">Learn about Text Links</a>
<a href="#email">Learn about Email Links</a>
```

HTML – Images

- Use the <img /> tag to place an image on your web page.

<imgsrc="sunset.gif" />

1. Image src

- Above we have defined the src attribute.
- Src stands for source, the source of the image or more appropriately, where the picture file is located.
- There are two ways to define the source of an image. First you may use a standard URL. (src=http://www.Xyz.com/pics/htmlT/sunset.gif) As your second choice, you may copy or upload the file onto your web server and access it locally using standard directory tree methods. (src="../sunset.gif")
- The location of this picture file is in relation to your location of your .html file.

URL Types:

Local Src	Location Description
src="sunset.gif"	picture file resides in same directory as .html file
src="../sunset.gif"	picture file resides in previous directory as .html file
src="../pics/sunset.gif"	picture file resides in the pic directory in a previous directory as .html file

- A URL cannot contain drive letters
- Therefore something like src="C:\\www\web\pics\" will not work. Pictures must be uploaded along with your .html file to your web server.

2. Alternative Attribute

- The alt attribute specifies alternate text to be displayed if for some reason the browser cannot find the image, or if a user has image files disabled.

<imgsrc="http://example.com/brokenlink/sunset.gif" alt="Beautiful Sunset" />

3. Image Height and Width

- To define the height and width of the image, rather than letting the browser compute the size, use the height and width attributes.

<imgsrc="sunset.gif" height="50" width="100">

4. Vertically and Horizontally Align Images

- Use the align and valign attributes to place images within your body, tables, or sections.

1. align (Horizontal)
    1. right 2. left 3. center
2. valign (Vertical)
    1. top 2. bottom 3. center

- Below is an example of how to align an image to the right of a paragraph

* This is paragraph 1, yes it is... *
* <imgsrc="sunset.gif" align="right">The image will appear along the...isn't it? *

5. Images as Links

- Images are very useful for links and can be created with the HTML below.

<imgsrc="sunset.gif">

HTML Forms

- A form will take input from the viewer and depending on your needs, you may store that data into a file, place an order, gather user statistics, register the person to your web forum, or maybe subscribe them to your weekly newsletter.

Making a Form

- <form> is main tag to build a form.
- It has a few optional attributes too. Below is an example of the form element.

<form action=”processform.php” method=”post”>
</form>

- The action attribute tells the HTML where to send the collected information, while the method attribute describes the way to send it.

Type of Input

- The main tag for collecting information from the user is <input>.
- The tag itself contains a name attribute, so that we can refer to the input by a name, and the size of the entry box in characters.
- There are quite few different types of input to choose from:
- <input type=”text”/> this is the default input type and accepts characters and numbers into a text box. It can also have a value attribute attached to it, which will give it an initial value.
- <input type=”password”/> this is similar to the above text box but anything that is typed cannot be seen; instead an asterisk is printed to cover up the entry. As the name suggests, this is used for password entry.
- <input type=”checkbox”/> this gives a box that can be toggled between checked and unchecked. It can initially be set to one or the other with checked=”checked”.
- <input type=”radio”/> this is similar to checkbox but in group of radio buttons only one can be selected at a time. This can also have an initial checked state on one of the radio buttons.
- <input type=”file”/>This will give a box to allow you to choose a file similar to when you open or save files usually on your machine. It can be used to select a file on the local machine for upload to a server.

- <input type=”submit”/> this allows a form to be submitted. When pressed, the information will be passed on for processing, usually to a script mentioned in the action attribute option of the form.
- <input type=”image”/> this will also submit the form when selected and, like the img tag, requires the src attribute to specify an associated image.
- <input type=”button”/> this makes a button available.
- <input type=”reset”/> this will reset the form to its initial state when selected.
- <input type=”hidden”/> this allows hidden data(not seen by the user) to be passed along with the form.

HTML Text Fields

- The <input> has a few attributes that you should be aware of.
- **type** - Determines what kind of input field it will be. Possible choices are text, submit, and password.
- **name** - Assigns a name to the given field so that you may reference it later.
- **size** - Sets the horizontal width of the field. The unit of measurement is in blank spaces.
- **maxlength** - Dictates the maximum number of characters that can be entered.

```
<form method="post" action="mailto:youremail@email.com">
Name: <input type="text" size="10" maxlength="40" name="name"><br />
Password: <input type="password" size="10" maxlength="10"
name="password">
```

HTML Radio Buttons

- Radio buttons are a popular form of interaction. You may have seen them on quizzes, questionnaires, and other web sites that give the user a multiple choice question. that relate to the radio button.

```
<form method="post" action="mailto:youremail@email.com">
What kind of shirt are you wearing? <br />
 Shade:
<input type="radio" name="shade" value="dark">Dark
<input type="radio" name="shade" value="light">Light <br />
</form>
```

HTML Check Boxes

- Check boxes allow for multiple items to be selected for a certain group of choices. The check box's name and value attributes behave the same as a radio button.

```
<form method="post" action="mailto:youremail@email.com">
Select your favorite cartoon characters.
<input type="checkbox" name="toon" value="Goofy">Goofy
<input type="checkbox" name="toon" value="Donald">Donald
<input type="checkbox" name="toon" value="Bugs">Bugs Bunny
</form>
```

HTML Drop Down Lists

- Drop down menues are created with the <select> and <option> tags. <select> is the list itself and each <option> is an available choice for the user.

```
<form method="post" action="mailto:youremail@email.com">
College Degree?
<select name="degree">
<option>Choose One</option>
<option>Some High School</option>
<option>High School Degree</option>
</select>
</form>
```

HTML Selection List

- Yet another type of form, a highlighted selection list. This form will post what the user highlights. Basically just another type of way to get input from the user.
- The size attribute selects how many options will be shown at once before needing to scroll, and the selected option tells the browser which choice to select by default.

```
<form method="post" action="mailto:youremail@email.com">
Musical Taste
<select multiple name="music" size="4">
<option value="emo" selected>Emo</option>
<option value="metal/rock" >Metal/Rock</option>
<option value="hiphop" >Hip Hop</option><option value="ska" >Ska</option>
<option value="jazz" >Jazz</option>
</form>
```

HTML Text Areas

- Text areas serve as an input field for viewers to place their own comments onto forums and the like use text areas to post what you type onto their site using scripts. For this form, the text area is used as a way to write comments to somebody.
- Rows and columns need to be specified as attributes to the <textarea> tag.

- Another attribute to be aware of is the wrap. Wrap has 3 values.

```
<form method="post" action="mailto:youremail@email.com">
<textarea rows="5" cols="20" name="comments"> Enter Comments Here
</textarea>
</form>
```

HTML Tables

- The <table> tag is used to begin a table. Within a table element are the <tr> (table rows) and <td> (table columns) tags.

```
<table border="1">
<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>
```

Row 1 Cell 1	Row 1 Cell 2
Row 2 Cell 1	Row 2 Cell 2

- Content is placed within tables cells. A table cell is defined by <td> and </td>.The border attribute defines how wide the table's border will be.

Spanning Multiple Rows and Cells

- Use rowspan to span multiple rows and colspan to span multiple columns.
- Note: if you would like to place headers at the top of your columns, use the <th> tag as shown below. By default these headers are bold to set them apart from the rest of your table's content.

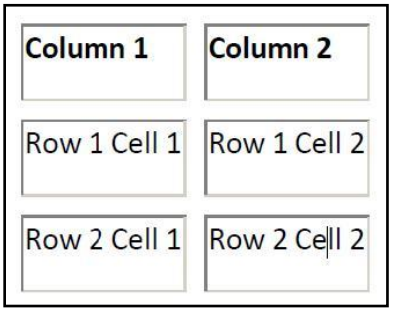
```
<table border="1"><tr><th>Column 1</th><th>Column 2</th><th>Column 3</th></tr>
<tr><td rowspan="2">Row 1 Cell 1</td><td>Row 1 Cell 2</td><td>Row 1 Cell 3</td></tr>
<tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>
<tr><td colspan="3">Row 3 Cell 1</td></tr>
</table>
```

Column 1	**Column 2**	**Column 3**
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

Cell Padding and Spacing

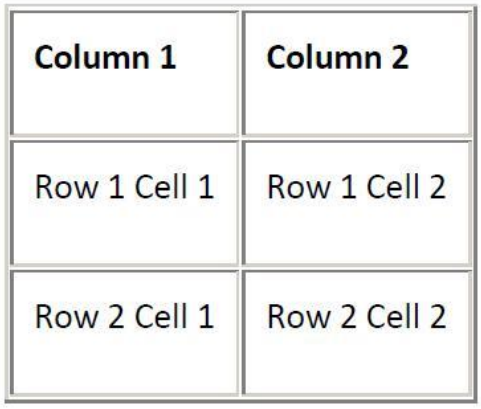
- With the cellpadding and cellspacing attributes you will be able to adjust the white space on your tables. Spacing defines the width of the border, while padding represents the distance between cell borders and the content within. Color has been added to the table to emphasize these attributes.

```
<table border="1" cellspacing="10" bgcolor="rgb(0,255,0)">
<tr><th>Column 1</th><th>Column 2</th></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>
```



- And now we will change the cellpadding of the table and remove the cellspacing from the previous example.

```
<table border="1" cellpadding="10" bgcolor="rgb(0,255,0)">
<tr><th>Column 1</th><th>Column 2</th></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 - <!-- Comments -->

- A comment is a way for you as the web page developer to control what lines of code are to be ignored by the web browser.
- Comment syntax may be a little complicated, there is an opening and a closing much like tags.
    1. <!-- Opening Comment
    2. -- > Closing Comment

<!--Note to self: This is my banner image! Don't forget -->
<imgsrc="http://www.website.com/pics/anyimage.jpg" height="100" width="200"/>

What is an HTML Form? Discuss different form attributes

- Form is a data collection mechanism within HTML that allows the design of various styles of input to suit most types of information.
- An input element can vary in many ways, depending on the type attribute. An input element can be of type textfield, checkbox, password, radiobutton, submit button, and more.
- Following are **attributes of <form>.**

1. **Name:**
   - Thename attribute specifies thenameofaformwhichisusedtoreferenceelementsinaJavaScript.

     <form action="URL"> Value : URL
     Description : Where to send the form data.

2. **Action:**
   - The required action attribute specifies where to send the form-data when a form issubmitted.

     <form action="URL"> Value : URL
     Description : Where to send the form data.

3. **Method :**
   - The method attribute specifies how to send form-data (the form-data is sent to the pagespecified in the action attribute).

     <form method="get|post">
     Value : get
     Description : Default. Appends the form-data to the URL in name/value pairs:
     URL?name=value&name=value
     Value : post
     Description : Sends the form-data as an HTTP post transaction.

4. **Target**
   - The target attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.

     <form target="_blank|_self|_parent|_top|framename">

Value	Description
_blank	Theresponseisdisplayedinanewwindowortab
_self	Theresponseisdisplayedinthesameframe(thisisdefault)
_parent	Theresponseisdisplayedintheparentframe
_top	Theresponseisdisplayedinthefullbodyofthewindow
framename	Theresponseisdisplayedinanamedframe

Explain following terms with example.

(1)<optgroup> :

- The <optgroup> is used to group related options in a drop-down list. If you have a long list of options, groups of related options are easier to handle for a user.

```
<select>
        <optgroup label="Swedish Cars">
                <option value="volvo">Volvo</option>
                <option value="saab">Saab</option>
        </optgroup>
        <optgroup label="German Cars">
                <option value="mercedes">Mercedes</option>
                <option value="audi">Audi</option>
        </optgroup>
</select>
```

(2):

- The tag is used to group inline-elements in a document.
- The tag provides no visual change by itself.
- The tag provides a way to add a hook to a part of a text or a part of a document.
- When the text is hooked in a element you can add styles to the content, or manipulate thecontent with for example JavaScript.

```
<p>Mymotherhas<spanstyle="color:lightblue">lightblue</span>eyes.</p>
```

Introduction to HTML5

- The DOCTYPE declaration for HTML5 is very simple:

  <!DOCTYPE html>

- The character encoding (charset) declaration is also very simple:

  <meta charset="UTF-8">

- New HTML5 Elements:
  - New semantic elements like <header>, <footer>, <article>, and <section>.
  - New form control attributes like number, date, time, calendar, and range.
  - New graphic elements: <svg> and <canvas>.
  - New multimedia elements: <audio> and <video>.
  - Elements Removed in HTML5

    The following HTML4 elements have been removed from HTML5:

Element	**Use instead**
<acronym>	<abbr>
<applet>	<object>
<basefont>	CSS
<big>	CSS
	CSS
<dir>	
<font>	CSS
<frame>	
<frameset>	
<noframes>	
<strike>	CSS
<tt>	CSS

What is CSS?

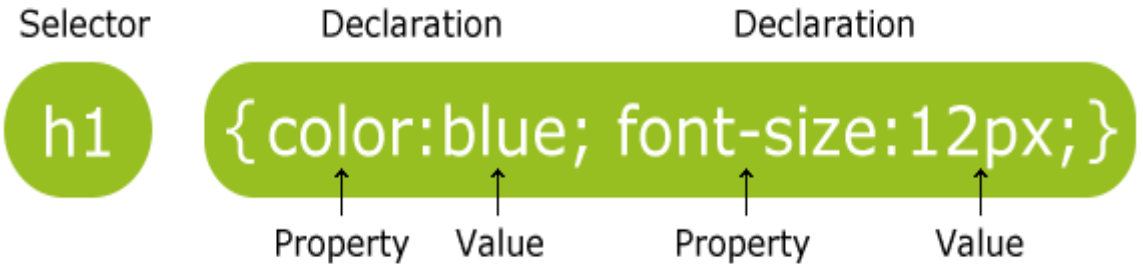
- CSS stands for Cascading Style Sheets
- Styles define how to display HTML elements
- External Style Sheets can save a lot of work
- External Style Sheets are stored in CSS files

Importance of CSS

- CSS defines HOW HTML elements are to be displayed.
- Styles are normally saved in external .css files. External style sheets enable you to change the appearance and layout of all the pages in a Web site, just by editing one single file.

CSS Syntax

- A CSS rule has two main parts: a selector, and one or more declarations:



- The selector is normally the HTML element you want to style.
- Each declaration consists of a property and a value.
- The property is the style attribute you want to change. Each property has a value.

What is the difference between class and id?

The id Selector

- The id selector is used to specify a style for a single, unique element.
- The id selector uses the id attribute of the HTML element, and is defined with a "#".
- The style rule below will be applied to the element with id="para1":

```
#para1
{ text-alian:center: color:red: }
```

The class Selector

- The class selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements.
- This allows you to set a particular style for many HTML elements with the same class.
- The class selector uses the HTML class attribute, and is defined with a "."
- In the example below, all HTML elements with class="center" will be center-aligned:

```
.center {text-align:center;}
```

- We can use more than one class in a single element

```
<a class=”Center bold italic”>
```

Explain different ways to write the CSS. / Explain CSS with all types. / Enlist and explain methods of using CSS in web page.

- There are three ways of inserting a style sheet:
  - External style sheet
  - Internal/Embedded style sheet
  - Inline style

1. External Style Sheet

- When using CSS it is preferable to keep the CSS separate from your HTML.
- Placing CSS in a separate file allows the web designer to completely differentiate between content (HTML) and design (CSS).
- External CSS is a file that contains only CSS code and is saved with a ".css" file extension.
- This CSS file is then referenced in your HTML using the <link> instead of <style>.

File Creation

- Open up notepad.exe, or any other plain text editor and type the following CSS code.

```
body{ background-color: gray;} p { color: blue; }h3{ color: white; }
```

- Save the file as a CSS (.css) file.
- Name the file "test.css" (without the quotes). Now create a new HTML file and fill it with the following code.

```
<html><head>
<link rel="stylesheet" type="text/css" href="test.css" /></head>
<body>
<h3> A White Header </h3>
<p> This paragraph has a blue font.
The background color of this page is gray because we changed it with CSS! </p>
</body></html>
```

Why Use External CSS?

- It keeps your website design and content separate.

- It's much easier to reuse your CSS code if you have it in a separate file. Instead of typing the same CSS code on every web page you have, simply have many pages refer to a single CSS file with the "link" tag.
- You can make drastic changes to your web pages with just a few changes in a single CSS file.

2. Internal/Embedded CSS

- This type of CSS is only for Single Page.
- When using internal CSS, we must add a new tag, <style>, inside the <head> tag. The HTML code below contains an example of <style>'s usage.

```
<html><head>
<style type="text/css"></style>
</head><body>
<p>Your page's content!</p></body>
</html>
```

Creating Internal CSS Code

- Below is an example of simple CSS code.

```
<html><head>
<style type="text/css">
p {color: white; }
body {background-color: black; }
</style></head><body>
<p>White text on a black background!</p></body>
</html>
```

3. Inline CSS

- It is possible to place CSS right in your HTML code, and this method of CSS usage is referred to as inline css.
- Inline CSS has the highest priority out of external, internal, and inline CSS.
- This means that you can override styles that are defined in external or internal by using inline CSS.
- If you want to add a style inside an HTML element all you have to do is specify the desired CSS properties with the style HTML attribute.

```
<html><head>
<link rel="stylesheet" type="text/css" href="test.css" /></head>
<body>
<p style="background: blue; color: white;">A new background andfont color with
inline CSS</p></body>
</html>
```

Explain CSS Background with all its attributes

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

1. CSS Background Color

- The background-color property specifies the background color of an element.

- The background color of a page is defined in the body selector:
- Below is example of CSS backgrounds

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

2. CSS Background Image

- The background-image property specifies an image to use as the background of an element.

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

3. Background Image Repeat

- You can have a background image repeat vertically (y-axis), horizontally (x-axis), in both directions, or in neither direction.

```
p {background-image: url(smallPic.jpg); background-repeat: repeat; }
h4 {background-image: url(smallPic.jpg); background-repeat: repeat-y; }
ol {background-image: url(smallPic.jpg); background-repeat: repeat-x; }
ul {background-image: url(smallPic.jpg);background-repeat: no-repeat;     }
```

4. CSS Fixed Background Image

- The background-attachment property sets whether a background image is fixed or scrolls with the rest of the page.

```
textarea.noScroll { background-image: url(smallPic.jpg); background-attachment: fixed;}
textarea {
background-image: url(smallPic.jpg);
background-attachment: scroll;     }
```

5. CSS Background Image Positioning

- The background-position property sets the starting position of a background image.

```
p {background-image: url(smallPic.jpg); background-position: 20px 10px;}
h4 {background-image: url(smallPic.jpg); background-position: 30% 30%;}
ol {background-image: url(smallPic.jpg); background-position: top center;}
```

Explain CSS Font with all its attributes

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

1. CSS Font Color

- Set the text-color for different elements:

```
h4 { color: red; }
h5 { color: #9000A1; }
h6 { color: rgb(0, 220, 98); }
```

2. CSS Font Family

- The font family of a text is set with the font-family property.

```
h4 { font-family: sans-serif; }h5 { font-family: serif; }
h6 { font-family: arial; }
```

3. CSS Font Size

- The font-size property sets the size of the text.

p { font-size: 120%; } ol{ font-size: 10px; } ul{ font-size: x-large; }

4. CSS Font Style

- The font-style property is mostly used to specify italic text.
- This property has three values:
  - normal - The text is shown normally
  - italic - The text is shown in italics
  - oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

p { font-style: italic; }h4{ font-style: oblique; }

5. CSS Font Weight

- The font-weight property sets how thick or thin characters in text should be displayed.

p { font-weight: 100; } ul{ font-weight: bolder; }

6. CSS Font Variant

- The font-variant property specifies whether or not a text should be displayed in a small-caps font.

p { font-variant: small-caps; }}

Explain CSS Text with all its attributes.

- While CSS Font covers most of the traditional ways to format your text, CSS Text allows you to control the spacing, decoration, and alignment of your text.

1. Text Decoration

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

h4{ text-decoration: line-through; }
h5{ text-decoration: overline; }
h6{ text-decoration: underline; }
a { text-decoration: none; }

2. Text Indent

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

p { text-indent: 20px; } h5 { text-indent: 30%; }

3. Text Align

- The text-align property is used to set the horizontal alignment of a text.

p { text-align: right; }
h5{ text-align: justify; }

4. Text Transform

- The text-transform property is used to specify uppercase and lowercase letters in a text.

p { text-transform: capitalize; } h5{ text-transform: uppercase; }

5. CSS White Space

- The white-space attribute allows you to prevent text from wrapping until you place a break
 into your text.

p { white-space: nowrap; }

6. CSS Word Spacing

- With the CSS attribute word-spacing you are able to specify the exact value of the spacing between your words. Word-spacing should be defined with exact values.

p { word-spacing: 10px; }

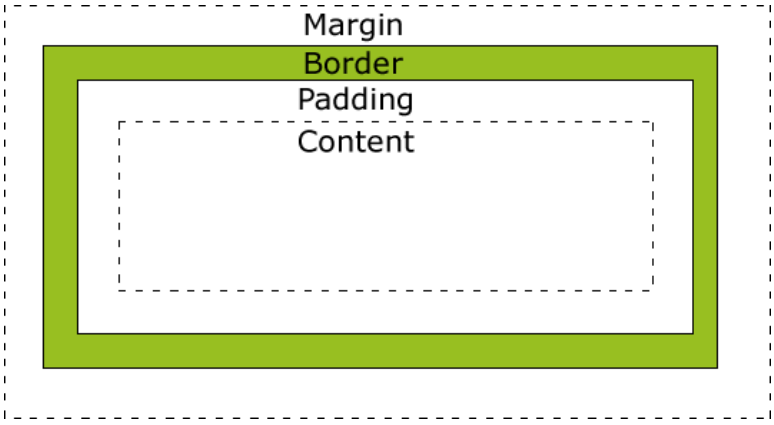
7. CSS Letter Spacing

- With the CSS attribute letter-spacing you are able to specify the exact value of the spacing between your letters. Letter-spacing should be defined with exact values.

p { letter-spacing: 3px; }

Explain BOX MODEL.

- All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content.
- The box model allows us to place a border around elements and space elements in relation to other elements.
- The image below illustrates the box model:



- Explanation of the different parts:
  - **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent
  - **Border** - A border that goes around the padding and content. The border is affected by the background color of the box
  - **Padding** - Clears an area around the content. The padding is affected by the background color of the box
  - **Content** - The content of the box, where text and images appear

Explain CSS Padding.

- The CSS padding properties define the space between the element border and the element content.

p {padding: 15px; border: 1px solid black;

- The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all puddings at once.

1. Possible Values

Value	Descriptions
length	Defines a fixed padding (in pixels, pt, em, etc.)
%	Defines a padding in % of the containing element.

padding-top:25px;
padding-bottom:25px;
padding-right:50px;
padding-left:50px;

2. Padding - Shorthand property

- To shorten the code, it is possible to specify all the padding properties in one property. This is called a shorthand property.

padding:25px 50px;

Explain CSS Margin.

- The CSS margin properties define the space around elements.

p {margin: 5px; border: 1px solid black; }

- The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

Value	Descriptions
auto	The browser calculates a margin
length	Specifies a margin in px, pt, cm, etc. Default value is 0px
%	Specifies a margin in percent of the width of the containing element
inherit	Specifies that the margin should be inherited from the parent element

1. Margin - Individual sides

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

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

2. Margin - Shorthand property

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

margin:100px 50px;

Explain CSS Border with all its attributes.

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

1. Border Style Types

- The border-style property specifies what kind of border to display.

p.solid {border-style: solid; } p.double {border-style: double; } p.groove {border-style: groove; }
p.dotted {border-style: dotted; } p.dashed {border-style: dashed; } p.inset {border-style: inset; }
p.outset {border-style: outset; } p.ridge {border-style: ridge; } p.hidden {border-style: hidden; }

2. Border Width

- The border-width property is used to set the width of the border.

table { border-width: 7px; border-style: outset; }
td { border-width: medium; border-style: outset; }
p { border-width: thick; border-style: solid; }

3. Border Color

- The border-color property is used to set the color of the border.
- Border colors can be any color defined by RGB, hexadecimal, or key terms. Below is an example of each of these types.

table { border-color: rgb(100, 100, 255); border-style: dashed; }
td { border-color: #FFBD32; border-style: ridge; }
p { border-color: blue; border-style: solid; }

4. Border: border-(direction)

- If you would like to place a border on only one side of an HTML element, or maybe have a unique look for each side of the border, then use border-(direction).
- The direction choices are of course: top, right, bottom, and left. CSS allows you to treat each side of a border separately from the other three sides.
- Each side can have its own color, width, and style set, as shown below.

p { border-bottom-style: dashed ; border-bottom-color: yellow; border-bottom-width: 5px; }
h4 { border-top-style: double; border-top-color: purple; border-top-width: thick; }

Explain CSS Lists with all its attributes.

- The CSS list properties allow you to:
  - Set different list item markers for ordered lists
  - Set different list item markers for unordered lists
  - Set an image as the list item marker

1. CSS List Style Type

- Specify all the list properties in one declaration.
  - Unordered list styles: square, circle, disc (default), and none
  - Ordered list styles: upper-alpha, lower-alpha, upper-roman, lower-roman, decimal (default), and none

```
ol { list-style-type: upper-roman; }
ul { list-style-type: circle; }
```

2. CSS Lists with Images

- Specify an image as the list-item marker in a list:

```
ul { list-style-image: url("listArrow.gif"); }
ol { list-style-image: url("listArrow2.gif"); }
```

3. CSS List Position

- With Specify that the the list-item markers should appear inside the content flow (results in an extra indentation)

```
ul { list-style-position: inside; }
ol { list-style-position: outside; }
```

- **Note**: "Outside" is actually the default setting for indentation.

Explain CSS Links

1. CSS Anchor/Link States

- The four links states are:
  - a:link - a normal, unvisited link
  - a:visited - a link the user has visited
  - a:hover - a link when the user mouse over it
  - a:active - a link the moment it is clicked

```
a:link{color:#FF0000;}    /*unvisited link*/
a:visited{color:#00FF00;} /* visited link */
a:hover{color:#FF00FF;} /* mouse over link */
a:active {color:#0000FF;} /* selected link */
```

2. Text Decoration

- The text-decoration property is mostly used to remove underlines from links.

```
a:link {text-decoration:none;}
a:visited {text-decoration:none;}
a:hover {text-decoration:underline;}
a:active {text-decoration:underline;}
```

3. Background Color

- The background-color property specifies the background color for links.

```
a:link {background-color:#B2FF99;}
a:visited {background-color:#FFFF85;}
a:hover {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

Explain CSS Position with example.

- With the knowledge of CSS Positioning we will be able to manipulate the exact position of your HTML elements.

1. Position Relative

- Relative positioning changes the position of the HTML element relative to where it normally appears.
- If we had a header that appears at the top of our page, we could use relative positioning to move it a bit to the right and down a couple of pixels. Below is an example.

```
h3 {position: relative; top: 15px;left: 150px;}
p {position: relative; left: -10px;}
```

2. Position Absolute

- With absolute positioning, you define the exact pixel value where the specified HTML element will appear.
- The point of origin is the top-left of the browser's viewable area, so be sure you are measuring from that point.

```
h3 {position: absolute; top: 50px;left: 45px;}
p{position: absolute; top: 75px;left: 75px;}
```

Explain CSS Layers. / z-index property

- CSS allows you to control which item will appear on top with the use of layers.
- In CSS, each element is given a priority.
- If there are two overlapping CSS positioned elements, the element with the higher priority will appear on top of the other.
- To manually define a priority, set the z-index value. The larger the value, the higher the priority the element will have.

```
h4{position: relative; top: 30px;left: 50px; z-index: 2;}
p {position: relative; z-index: 1;background-color: #FFCCCC;}
```

- This paragraph has a z-index of 1, which is less than the header.
- If we had not defined the z-index, by default the paragraph would have been on top of the header because it appears later in our HTML code.

Explain CSS Float property.

- With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.
- Wrapping text around an image is easy when using the CSS Float attribute.
- You have a choice to either float the picture to the left or to the right and the rest is done for you.

```
img.floatLeft { float: left; margin: 4px;}
img.floatRight { float: right; margin: 4px;}
```

```
<body>
<img src="sunset.gif" class="floatLeft"><p>The images are contained with...</p>
<img src="sunset.gif" class="floatRight"><p>This second paragraph has an...</p>
</body>
```

Introduction to CSS3

- CSS3 is the latest standard for CSS.
- CSS3 is completely backwards-compatible with earlier versions of CSS.
- CSS3 has been split into "modules". It contains the "old CSS specification" (which has been split into smaller pieces). In addition, new modules are added.
- CSS3 Transitions are a presentational effect which allow property changes in CSS values, such as those that may be defined to occur on :hover or :focus, to occur smoothly over a specified duration – rather than happening instantaneously as is the normal behaviour.
- Transition effects can be applied to a wide variety of CSS properties, including background-color, width, height, opacity, and many more.
- Some of the most important CSS3 modules are:
    - Selectors
    - Box Model
    - Backgrounds and Borders
    - Image Values and Replaced Content
    - Text Effects
    - 2D/3D Transformations
    - Animations
    - Multiple Column Layout
    - User Interface

What is JavaScript?

- HTML and CSS concentrate on a static rendering of a page; things do not change on the page over time, or because of events.
- To do these things, we use scripting languages, which allow content to change dynamically.
- Not only this, but it is possible to interact with the user beyond what is possible with HTML.
- Scripts are programs just like any other programming language; they can execute on the client side or the server.

Advantages of client side scripting

- The web browser uses its own resources, and eases the burden on the server.
- It has fewer features than server side scripting.
- It saves network bandwidth.

Disadvantages of client side scripting

- Code is usually visible.
- Code is probably modifiable.
- Local files and databases cannot be accessed.
- User is able to disable client side scripting.

Differentiate between server side and client side scripting languages

Client-side scripting languages

- The client-side environment used to run scripts is usually a browser.
- The processing takes place on the end users computer.
- The source code is transferred from the web server to the user's computer over the internet and run directly in the browser.
- The scripting language needs to be enabled on the client computer.
- Sometimes if a user is conscious of security risks they may switch the scripting facility off.
- When this is the case a message usually pops up to alert the user when script is attempting to run.

Server-side scripting languages

- The server-side environment that runs a scripting language is a web server.
- A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages.
- This HTML is then sent to the client browser.
- It is usually used to provide interactive web sites that interface to databases or other data stores on the server.

- This is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript.
- The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.

What is difference between Java script and JAVA?

- Java is a statically typed language; JavaScript is dynamic.
- Java is class-based; JavaScript is prototype-based.
- Java constructors are special functions that can only be called at object creation; JavaScript "constructors" are just standard functions.
- Java requires all non-block statements to end with a semicolon; JavaScript inserts semicolons at the ends of certain lines.
- Java uses block-based scoping; JavaScript uses function-based scoping.
- Java has an implicit this scope for non-static methods, and implicit class scope; JavaScript has implicit global scope.

Embedded JavaScript

- JavaScript can be embedded in an HTML document.
- To embed it in HTML you must write:

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

- The script tag has effect of the stopping the JavaScript being printed out as well as indentifying the code enclosed.
- The JavaScript can be placed in the head section of your HTML or the body.

```
<html>
<body>
<script type="text/javascript">
        document.write("<h1>This is a heading</h1>");
</script>
</body>
</html>
```

- The Scripts placed in the body section are executed as the page loads and can be used to generate the content of the page.
- As well as the body section, JavaScript can also be placed in the head part.
- The advantages of putting a script in there are that it loads before the main body.

External JavaScript

- If you want to use the same script on several pages it could be a good idea to place the code in a separate file, rather than writing it on each.
- That way if you want to update the code, or change it, you only need to do it once.

- Simply take the code you want in a separate file out of your program and save it with the extension .js.

```
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

JavaScript Variables

- Variables in JavaScript behave the same as variables in most popular programming languages (C, C++, etc) do, but in JavaScript you don't have to declare variables before you use them.
- A variable's purpose is to store information so that it can be used later. A variable is a symbolic name that represents some data that you set.
- When using a variable for the first time it is not necessary to use "**var**" before the variable name.
- Variable names must begin with a letter.
- Variable names are case sensitive (y and Y are different variables).

```
var x=5;
var y=6;
var z=x+y;
```

- You can declare many variables in one statement. Just start the statement with var and separate the variables by comma:

```
var name="Doe", age=30, job="carpenter";
var name="Doe",
age=30,
job="carpenter";
```

- Variable declared without a value will have the value **undefined**.
- If you re-declare a JavaScript variable, it will not lose its value.
- The value of the variable *carname* will still have the value "Volvo" after the execution of the following two statements.

```
varcarname="Volvo";
varcarname;
```

JavaScript Operators

- Operators in JavaScript are very similar to operators that appear in other programming languages.
- The definition of an operator is a symbol that is used to perform an operation.
- Most often these operations are arithmetic (addition, subtraction, etc), but not always.

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
=	Assignment

```
<body>
<script type="text/JavaScript">
<!--
var two = 2
var ten = 10
varlinebreak = "<br />"
document.write("two plus ten = ")
var result = two + ten
document.write(result)
//-->
</script>
</body>
```

Assignment	Equivalent to
X+=Y	X=X+Y
X-=Y	X=X-Y
X*=Y	X=X*Y
X/=Y	X=X/Y
X%=Y	X=X%Y

JavaScript Array

- An array is a special variable, which can hold more than one value at a time.
- The Array object is used to store multiple values in a single variable.
- An array can be created in three ways.
- The following code creates an Array object called myCars.

1. **Regular**

```
varmyCars=new Array();
myCars[0]="Saab";
myCars[1]="Volvo";
myCars[2]="BMW";
```

2. **Condensed**

```
varmyCars=new Array("Saab","Volvo","BMW");
```

3. **Literal**

```
varmyCars=["Saab","Volvo","BMW"];
```

Access an Array

- You refer to an element in an array by referring to the **index** number.
- This statement access the value of the first element in myCars.

```
var name=myCars[0];
```

- This statement modifies the first element in myCars:

```
myCars[0]="Opel";
```

JavaScript Functions

- A function is a section of code that is separate from the main program.
- It is defined once but can be invoked many times.
- A function can be passed as parameters so that they can be used and a value can be returned back.
- There are some functions already built in to JavaScript, such as the Math.cos() function, which calculates the cosine of an angle.
- An example function could be:

```
functionmultByTen(x)
{
        return x*10;
}
```

- This can then be invoked by using the function's name complete with any parameters you want to pass:

```
mysum=multByTen(3)
```

- Below is an example of JavaScript function.

```
<html><body>
<script  type="text/javascript">
var z= multXbyY(10,15);
document.write("The result is" +z);
functionmultXbyY(x,y) {
        document.write("x is " +x);
        document.write("y is "+y);
        return x*y;
}
</script>
</body></html>
```

JavaScript Conditions

- Conditional statements are used to perform different actions based on different conditions.
- In JavaScript we have the following conditional statements:
  - **if statement** - use this statement to execute some code only if a specified condition is true
  - **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false
  - **if...else if....else statement** - use this statement to select one of many blocks of code to be executed
  - **switch statement** - use this statement to select one of many blocks of code to be executed

If Statement

- Use the if statement to execute some code only if a specified condition is true.

```
if (condition)
 {
        code to be executed if condition is true
 }
```

If…else Statement

- Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

```
if (condition)
 {
 code to be executed if condition is true
 }
else
 {
 code to be executed if condition is not true
 }
```

If…else if…else Statement

- Use the if....else if...else statement to select one of several blocks of code to be executed.

```
if (condition1)  {
 code to be executed if condition1 is true
 }
else if (condition2)   {
 code to be executed if condition2 is true
 }
else  {
 code to be executed if neither condition1 nor condition2 is true
 }
```

Switch Statement

- Use the switch statement to select one of many blocks of code to be executed.

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is different from case 1 and 2
}
```

The *default* Keyword

- Use the *default* keyword to specify what to do if there is no match.

Conditional Operator

- JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

```
variablename=(condition)?value1:value2
voteable=(age<18)?"Too young":"Old enough";
```

JavaScript Loops and Repetition

- Loops can execute a block of code a number of times.
- Loops are handy, if you want to run the same code over and over again, each time with a different value.

Different Kinds of Loops

- JavaScript supports different kinds of loops:
  - **for** - loops through a block of code a number of times
  - **for/in**- loops through the properties of an object
  - **while** - loops through a block of code while a specified condition is true
  - **do/while** - also loops through a block of code while a specified condition is true

The For Loop

- The for loop is often the tool you will use when you want to create a loop.
- The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3)
 {
 the code block to be executed
 }
```

- **Statement 1** is executed before the loop (the code block) starts.
- **Statement 2** defines the condition for running the loop (the code block).
- **Statement 3** is executed each time after the loop (the code block) has been executed.

```
for (var i=0; i<5; i++)
 {
 x=x + "The number is " + i + "<br>";
 }
```

The For/In Loop

- The JavaScript for/in statement loops through the properties of an object.

```
var person={fname:"John",lname:"Doe",age:25};
for (x in person)
 {
 txt=txt + person[x];
 }
```

The While Loop

- The while loop loops through a block of code as long as a specified condition is true.

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

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, then it will repeat the loop as long as the condition is true.

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

JavaScript Objects

- JavaScript has several built-in objects, like String, Date, Array, and more.
- An object is just a special kind of data, with **properties** and **methods**.

Accessing Object Properties

- Properties are the values associated with an object.
- The syntax for accessing the property of an object is below

objectName.propertyName

- This example uses the length property of the String object to find the length of a string:

```
var message="Hello World!";
var x=message.length;
```

Accessing Objects Methods

- Methods are the actions that can be performed on objects.
- You can call a method with the following syntax.

```
objectName.methodName()
```

- This example uses the toUpperCase() method of the String object, to convert a text to uppercase.

```
var message="Hello world!";
var x=message.toUpperCase();
```

Creating JavaScript Objects

- With JavaScript you can define and create your own objects.
- There are 2 different ways to create a new object:
  - Define and create a direct instance of an object.
  - Use a function to define an object, then create new object instances.

Creating a Direct Instance

- This example creates a new instance of an object, and adds four properties to it:

```
person=new Object();
person.firstname="Narendra";
person.lastname="Modi";
person.age=24;
person.eyecolor="blue";
```

User- Defined Objects/How user defined objects are created in JavaScript?

- JavaScript allows you to create your own objects.

- The first step is to use the *new* operator.

```
VarmyObj= new Object();
```

- This creates an empty object.
- This can then be used to start a new object that you can then give new properties and methods.
- In object- oriented programming such a new object is usually given a constructor to initialize values when it is first created.
- However, it is also possible to assign values when it is made with literal values.

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

<script language="JavaScript" type="text/JavaScript">
person={
                firstname: "Ketan",
                lastname: "Chavda",
                age: 24,
                eyecolor: "blue"
        }
document.write(person.firstname + " is " + person.age + " years old.");
</script>

</body>
</html>
```

How a constructor can be used to populate data in the object?

- A constructor is pre defined method that will initialize your object.
- To do this in JavaScript a function is used that is invoked through the *new* operator.
- Any properties inside the newly created object are assigned using *this* keyword, referring to the current object being created.

```
<!DOCTYPE html>
<html>
<body>
<script>
function person(firstname, lastname, age)
{
        this.firstname = firstname;
        this.lastname = lastname;
        this. age = age;
}
var person1=new person("Narendra","Modi",24);
document.write(person1.firstname + " "+ person1.lastname +" "+ person1.age);
</script>
</body>
</html>
```

- In above the function *person*becomes the constructor invoked through the *new* keyword on assignment to the *person1* variable.
- Here the values are passed as parameters to the constructor.
- Inside the constructor the *this* keyword takes on the value of the newly created object and therefore applies properties to it.

Explain the event handling in JavaScript.

- Event handlers are attributes that force an element to "listen" for a specific event to occur.
- Event handlers all begin with the letters "on".
- There are two types of events in Javascript
    - Interactive i.g. onClick
    - Non-interactive i.g. onLoad
- The table below lists the HTML event handlers with descriptions.

Event Handler	Elements Supported	Description
onblur	a, area, button, input, label, select, textarea	the element lost the focus
onchange	input, select, textarea	the element value was changed
onclick	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer button was clicked
ondblclick	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer button was double clicked
onfocus	a, area, button, input, label, select, textarea	the element received the focus
onkeydown	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a key was pressed down
onkeypress	All elements except br, font, frame, frameset,	a key was pressed and released

	head, html, iframe, isindex, meta, param, script, style, title	
onkeyup	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a key was released
onload	frameset	all the frames have been loaded
onload	body	the document has been loaded
onmousedown	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer button was pressed down
onmousemove	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer was moved within.
onmouseout	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer was moved away
onmouseover	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer was moved onto
onmouseup	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer button was released
onreset	form	the form was reset
onselect	input, textarea	some text was selected
onsubmit	form	the form was submitted
onunload	frameset	all the frames have been removed
onunload	body	the document has been removed

Explain document object in JavaScript.

- There are lots of objects that descend from the *document* object forming a large sub-tree known as the Document Object Model (DOM), which has become standardized.
- The *document* object represents the HTML displayed in window.
- Using this object it is possible to access information about the document itself and also about the information content.
- Using this object it is possible to control certain parameters of the current document like background, foreground and link colors.

```
VarmyObj= new Object();
document.bgColor = "#9F2020";
documet.fgcolor = "#FAF519";
```

- It is possible to access any form information in document by using the *form[]* array.
- This contains all the *form* objects that are in the document.
- For example, a form can be constructed with :

```
<form name=”userDetails”>
<input type=”text” name= “ fname“/>
<input type=”text” name= “lname“/>
<input type=”submit” name= “ Submit“/>
```

- The form data can then be accessed with various DOM syntax constructions. The form itself can be accessed through:

  document.forms[0];

- It can also referred to by

  document.userDetails

- An individual element can then be accessed:

  document.userDetails.fname

- Below example shows the use of *document* object.

```
<html><head>
<title>Document Object Example</title>

<script type="text/javascript">
functionincrementCurrent() {
current = parseInt(document.forms["noteForm"].total.value);
document.forms["noteForm"].total.value = current + 1;
    }
</script>

</head><body>
<div id="mainDiv">
<h1>Document Object Example</h1>
<form id="noteForm">
            Current number of notes:
        <input type="text" name="total" value="0" size="3"/>
        <input type="button" value="Add a new note"
onclick="incrementCurrent()"/>
</form>
</div>
</body></html>
```

Why do you need validation? Show the use of regular expression in JavaScript to validate the email address with example.

- The idea behind JavaScript form validation is to provide a method to check the user enteredinformation before they can even submit it.
- JavaScript also lets you display helpful alerts to inform the user what information they have entered incorrectly and how they can fix it.

JavaScript Email Address validation using Regular Expression

- Using Regular expression we do checking for valid information.

```
<html>
<head>
<title>JavaScript Forms</title>
</head>

<body>
<form method="post" name="getinfo" onSubmit="return processForm()">
<input type="text" name="email"/>
<input type="submit" value="log in" name="Login"/>
</form>

<script language="JavaScript" type="text/JavaScript">
functionprocessForm()
{
varmyform= document.getinfo;
var check= myform.email.value;
Document.write(testEmail(check));
}
Function testEmail(chkMail)
{
varemailpattern = "^[\\w-_\.]*[\\w-_\.]\@[\\w]\.+[\\w]+[\\w]$";
var regex = new RegExp(emailpattern);
returnregex.test(chkMail);
}
</script>
</body>
</html>
```

- This will check for a valid email as describe.
- The testEmail() function returns true or false.

- The way it determines this end result is built on the template/pattern in the string *emailpattern.*
- This is used to work out the order of expected characters, how many times they repeat and specially occurring punctuation.
- The string in this case that is used as template is:

  "^[\\w- \.][\\w- \.]\@[\\w]\.+[\\w]+[\\w]$"*

- The first section is:

  ^[\\w- \.]

- This sequence, beginning with ^, means check the first character is a word character represented by \\w.
- The next part is :

  **[\\w-_\.]*

- The * means that the next series of characters described can be represented many times or not at all.
- The characters themselves are the same as before; that is word characters, underscore, hyphen or period.

  \@[\\w]\.+

- This section begins by checking for the @ character.
- Following this should ne the word characters and then at least one 'dot'.
- In other words it would not accept a dot straight after the @ character.
- The last part is :

  [\\w]+[\\w]$

- The first set in square brackets makes sure that there are some characters after the dot and the last part checks that the last character is a word character.
- In this program after the string is declared, a regular expression object is created with the pattern:

  var regex = new RegExp(emailpattern);

- The pattern can then be tested against the incoming parameter with object's test method:

  returnregex.test(chkMail);

- This will return true or false depending on whether there is a match or not.

List the important built-in objects.

Built-in Objects:

- JS String
- JS Date

- JS Array
- JS Boolean
- JS Math
- JS RegExp

Example of inbuilt object

Math Object:

- The Math object allows you to perform mathematical tasks.
- The Math object includes several mathematical constants and methods.
- Syntax for using properties/methods of Math:

```
var x=Math.PI;
var y=Math.sqrt(16);
```

Math Object Properties

Property	Description
E	Returns Euler's number(approx.2.718)
LN2	Returns the natural logarithm of 2 (approx.0.693)
LN10	Returns the natural logarithm of 10 (approx.2.302)
LOG2E	Returns the base-2 logarithm of E (approx.1.442)
LOG10E	Returns the base-10 logarithm of E (approx.0.434)
PI	Returns PI(approx.3.14)
SQRT1_2	Returns the squareroot of 1/2(approx.0.707)
SQRT2	Returns the squareroot of 2(approx.1.414)

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between-PI/2 and PI/2radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of Ex
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm(base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x

XML

What is XML?

- XML is a meta-language, which can be used to store data & act as a mechanism to transfer information between dissimilar systems.
- XML stands for E**X**tensible **M**arkup **L**anguage.
- XML is a **markup language** much like HTML.
- XML was designed to **describe data**.
- XML tags are not predefined in XML. You must **define your own tags**.
- XML is **self describing**.
- XML uses a **DTD (Document Type Definition)** to formally describe the data.

```
<?xml version="1.0"?>
<Person>
<Firstname>Ralph</Firstname>
<Lastname>Mosely</Lastname>
</Person>
```

Difference between XML and HTML

XML	HTML
XML was designed to store data and transferthe data.	HTML was designed to display data.
XML focuses on what data is.	HTML focus on how data looks.
In XML you can design your own tag.	HTML has predefined tags.
XML uses parser to check & read xml fileseg. DOM, SAX	HTML don't use any kind of parser

---

Use of XML

- Used to exchange data between dissimilar systems.
- Used to describe content of document.
- XML can be used as database to store data.

Features of XML

- XML has its own tag so it's **self describing**.
- **Language Independent:**Any language is able to read & write XML.
- **OS Independent:** can be work on any platform.
- **Readability:** It's a plain text file in user readable format so you can edit or viewin simple editor.

- **Hierarchical:** It has hierarchical structure which is powerful to express complex data and simple to understand.

XML Key Component

1.) XML Root Element

- XML must have root element. The first element after xml version declaration in file is a root element.

```
<bookstore>
    <book category="CHILDREN">
        <title>Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
</bookstore>
```

- In above example <bookstore> is root element.

2.) XML Element

- An XML element is everything from (including) the element's start tag to (including) the element's end tag.
- An element can contain:
  - other elements
  - text
  - attributes
  - or a mix of all of the above...
- In above example <title>, <author>, <year> and <price> are elements.

3.) XML Attribute

- Attributes provide additional information about an element.
- Attributes often provide information that is not a part of the data. In the example below,the file type is irrelevant to the data, but can be important to the software that wants to manipulate the element

```
<file type="gif">computer.gif</file>
```

- **XML Attributes Must be Quoted**
- Attribute values must always be quoted. Either single or double quotes can be used. For a person's sex, the person element can be written like this:

```
<person sex="female"> or <person sex='female'>
```

4.) XML Namespace

- The XML namespace is a special type of reserved XML attribute that you place in an XML tag.
- The reserved attribute is actually more like a prefix that you attach to any namespace you create.
- This attribute prefix is "**xmlns:**" which stands for XML Namespace.
- The colon is used to separate the prefix from your namespace that you are creating.
- **xmlns** must have a unique value that no other namespace in the document has. What is commonly used is the URI (Uniform Resource Identifier) or the more commonly used URL.

Xmlns="http://www.mydomian.com/ns/animals/1.1"

Create a XML file that contains Book Information.

```
<xml version="1.0"?>
<bookstore>
    <book>
        <title>Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
    </book>
    <book>
        <title>WAD</title>
        <author>Ralph Mosely</author>
        <year>2001</year>
        <price>395</price>
    </book>
</bookstore>
```

DTD (Document Type Definition)

- The purpose of a DTD is to define the legal building blocks of an XML document. Itdefines the document structure with a list of legal elements. A DTD can be declaredinline in your XML document, or as an external reference.
- **Internal DTD**
    - This is an XML document with a Document Type Definition:

```
<?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>Ravi</to>
<from>Ketan</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

- The DTD is interpreted like this:
  - **!ELEMENT note (in line 2)** defines the element "note" as having four elements: "to,from,heading,body".
  - **!ELEMENT to (in line 3)** defines the "to" element to be of the type "CDATA".
  - **!ELEMENT from (in line 4)** defines the "from" element to be of the type "CDATA"

- **External DTD**
  - This is the same XML document with an external DTD

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to>Ravi</to>
<from>Narendra</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

  - This is a copy of the file "note.dtd" containing the Document Type Definition:

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

DTD - Elements

- In the DTD, XML elements are declared with an element declaration. An element declaration has the following syntax:

<!ELEMENT element-name (element-content)>

- **Empty elements**
  - Empty elements are declared with the keyword EMPTY inside the parentheses:

    <!ELEMENT img (EMPTY)>
- **Elements with data**
  - Elements with data are declared with the data type inside parentheses:

    <!ELEMENT element-name (#CDATA)>

    or

    <!ELEMENT element-name (#PCDATA)>

    or

    <!ELEMENT element-name (ANY)>

    example:

    <!ELEMENT note (#PCDATA)>

  - #CDATA means the element contains character data that is not supposed to be parsed bya parser.
  - #PCDATA means that the element contains data that IS going to be parsed by a parser.
  - The keyword ANY declares an element with any content.
  - If a #PCDATA section contains elements, these elements must also be declared.
- **Elements with children (sequences)**
  - Elements with one or more children are defined with the name of the children elementsinside the parentheses:

<!ELEMENT element-name (child-element-name)>

or

<!ELEMENT element-name (child-element-name,child-element-name,.....)>

example:

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

- When children are declared in a sequence separated by commas, the children must appearin the same sequence in the document. In a full declaration, the children must also bedeclared, and the children can also have children. The full declaration of the notedocument will be:

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

<!ELEMENT to (#CDATA)>

<!ELEMENT from (#CDATA)>

<!ELEMENT heading (#CDATA)>

<!ELEMENT body (#CDATA)>

- **Wrapping**
  - If the DTD is to be included in your XML source file, it should be wrapped in aDOCTYPE definition with the following syntax:

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

example:

<?xml version="1.0"?>

<!DOCTYPE note [

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

<!ELEMENT to (#CDATA)>

<!ELEMENT from (#CDATA)>

<!ELEMENT heading (#CDATA)>

<!ELEMENT body (#CDATA)>

]>

<note>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

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

- **Declaring only one occurrence of the same element**

<!ELEMENT element-name (child-name)>

example

<!ELEMENT note (message)>

- The example declaration above declares that the child element message can only occurone time inside the note element.

- **Declaring minimum one occurrence of the same element**

<!ELEMENT element-name (child-name+)>

example

<!ELEMENT note (message+)>

- The + sign in the example above declares that the child element message must occur oneor more times inside the note element.

- **Declaring zero or more occurrences of the same element**

<!ELEMENT element-name (child-name)>*

example

<!ELEMENT note (message)>*

- The * sign in the example above declares that the child element message can occur zeroor more times inside the note element.

- **Declaring zero or one occurrences of the same element**

<!ELEMENT element-name (child-name?)>

example

<!ELEMENT note (message?)>

- The? Sign in the example above declares that the child element message can occur zeroor one times inside the note element.

DTD – Attributes

- Attributes provide extra information about elements.
- Attributes are placed inside the start tag of an element.
- **Declaring Attributes**
  - In the DTD, XML element attributes are declared with an ATTLIST declaration. Anattribute declaration has the following syntax:

<!ATTLIST element-name attribute-name attribute-type default-value>

- As you can see from the syntax above, the ATTLIST declaration defines the elementwhich can have the attribute, the name of the attribute, the type of the attribute, and thedefault attribute value.
- The **attribute-type** can have the following values:

Value	Explanation
CDATA	The value is character data
(eval|eval|..)	The value must be an enumerated value
ID	The value is an unique id
IDREF	The value is the id of another element
IDREFS	The value is a list of other ids
NMTOKEN	The value is a valid XML name
NMTOKENS	The value is a list of valid XML names
ENTITY	The value is an entity
ENTITIES	The value is a list of entities
NOTATION	The value is a name of a notation
xml:	The value is predefined

- The **attribute-default-value** can have the following values:

Value	Explanation
#DEFAULT value	The attribute has a default value
#REQUIRED	The attribute value must be included in the element
#IMPLIED	The attribute does not have to be included
#FIXED value	The attribute value is fixed

- **Attribute declaration example**

DTD example:

<!ELEMENT square EMPTY>

<!ATTLIST square width CDATA "0">

XML example:

<square width="100"></square>

- In the above example the element square is defined to be an empty element with theattributes width of type CDATA. The width attribute has a default value of 0.

XML Schema

- An XML Schema describes the structure of an XML document.
- XML Schema is an XML-based alternative to DTD.
- The XML Schema language is also referred to as XML Schema Definition (XSD).
- XML Schema is a W3C Recommendation.

XSD Elements

- XML Schemas define the elements of your XML files. It's of two types:
  - **Simple**
  - **Complex Type**

XSD Simple Elements

- A simple element is an XML element that can contain only text. It cannot contain any other elements or attributes.
- **Defining a Simple Element**
  - The syntax for defining a simple element is:

    <xs:element name="aaa" type="bbb"/>

  - Where aaa is the name of the element and bbb is the data type of the element.
  - XML Schema has a lot of built-in data types. The most common types are:
    - xs:string
    - xs:decimal
    - xs:integer
    - xs:Boolean
    - xs:date
    - xs:time
- **Example**
  - Here are some XML elements:

    <lastname>Refsnes</lastname>

    <age>36</age>

    <dateborn>1970-03-27</dateborn>

  - And here are the corresponding simple element definitions:

    <xs:element name="lastname" type="xs:string"/>

    <xs:element name="age" type="xs:integer"/>

    <xs:element name="dateborn" type="xs:date"/>

XSD Complex Elements

- A complex element is an XML element that contains other elements and/or attributes.
- There are four kinds of complex elements:
  - empty elements
  - elements that contain only other elements
  - elements that contain only text
  - elements that contain both other elements and text
- **Examples of Complex Elements**
  - A complex XML element, "product", which is empty:

<product pid="1345"/>

  - A complex XML element, "employee", which contains only other elements:

<employee>

<firstname>John</firstname>

<lastname>Smith</lastname>

</employee>

- A complex XML element, "food", which contains only text:

  <food type="dessert">Ice cream</food>

- A complex XML element, "description", which contains both elements and text:

  <description>

  It happened on <date lang="norwegian">03.03.99</date> ....

  </description>

XSD Attributes

- Simple elements cannot have attributes. If an element has attributes, it is considered to be of a complex type. But the attribute itself is always declared as a simple type.
- **How to Define an Attribute?**
  - The syntax for defining an attribute is:

<xs:attribute name="aaa" type="bbb"/>

  - Where aaa is the name of the attribute and bbb specifies the data type of the attribute.
- **Default and Fixed Values for Attributes**
  - Attributes may have a default value OR a fixed value specified.
  - A default value is automatically assigned to the attribute when no other value is specified.
  - In the following example the default value is "EN":

<xs:attribute name="lang" type="xs:string" default="EN"/>

  - A fixed value is also automatically assigned to the attribute, and you cannot specifyanother value.
  - In the following example the fixed value is "EN":

<xs:attribute name="lang" type="xs:string" fixed="EN"/>

- **Optional and Required Attributes**
  - Attributes are optional by default. To specify that the attribute is required, use the "use"attribute:

    <xs:attribute name="lang" type="xs:string" use="required"/>

Difference between DTD and XML Schema

	XML Schema	DTD
Markup	Any global element can be root.	Can specify only the root

validation	No ambiguous content support.	element in the instance document. No ambiguous content support.
Namespace support	Yes. Declarations only where multiple namespaces are used.	No.
Code reuse	Can reuse definitions using named types.	Poorly supported. Can use parameter entities.
Data type Validation	Provides flexible set of data types. Provides multi-field key cross references. No co-occurrence constraints.	No real data type support.

XSL

What is XSL?

- XSL stands for E**X**tensible **S**tylesheet **L**anguage.
- XSL = Style Sheets for XML
- XSL describes how the XML document should be displayed!
- XSL - More Than a Style Sheet Language
- XSL consists of three parts:
  - XSLT-a language for transforming XML documents
  - XPath-a language for navigating in XML documents
  - XSL-FO-a language for formatting XML documents

What is XSLT?

- XSLT stands for XSL Transformations.
- XSLT is the most important part of XSL.
- XSLT transforms an XML document into another XML document.
- XSLT uses XPath to navigate in XML documents.

Explain XSL Transformation and XSL Elements

- The style sheet provides the template that transforms the document from one structure to another.
- In this case <xsl:template> starts the definition of the actual template, as the root of the source XML document.
- The match = "/" attribute makes sure this begins applying the template to the root of the source XML document.

Linking

- The style sheet is linked into the XML by adding the connecting statement to the XML document: <?xml-stylesheet type="text/xsl" href="libstyle.xsl" ?>

XSL Transformations

- XSLT is the most important part of XSL.
- XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.
- With XSLT you can add/remove elements and attributes to or from the output file. You can also

rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.

- A common way to describe the transformation process is to say that XSLT transforms an XML source-tree in to an XML result-tree.
- **XSLT Uses XPath:**
  - XSLT uses XPath to find information in an XML document.
  - XPath is used to navigate through elements and attributes in XML documents.
- **XSLT Works as:**
  - In the transformation process, XSLT uses XPath to define parts of the source document that should match one or more predefined templates.
  - When a match Is found, XSLT will transform the matching part of the source document into the result document.

XSL Elements

- XSL contains many elements that can be used to manipulate, iterate and select XML, for output.
  - value-of
  - for-each
  - sort
  - if
  - choose

<xsl:value-of> Element

- The <xsl:value-of> element extracts the value of a selected node.
- The <xsl:value-of> element can be used to select the value of an XML element and add it to the output.
- **Syntax**

```
<xsl:value-of select="expression" />
```

  - **expression**: This is Required. An XPath expression that specifies which node/attribute to extract the value from. It works like navigating a file system where a forward slash (/) selects subdirectories.

<xsl:for-each> Element

- The XSL <xsl:for-each> element can be used to select every XML element of a specified node-set.

<xsl:if> Element

- To put a conditional if test against the content of the XML file, add an <xsl:if> element to the XSL document.
- **Syntax**

```
<xsl:if test="expression">
...some output if the expression is
true...
</xsl:if>
```

<xsl:sort> Element

- The <xsl:sort> element is used to sort the output.
- <xsl:sort select="artist"/>
- The select attribute indicates what XML element to sort on.

Example using value-of, for-each and if

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
        <tr bgcolor="#9acd32">
                <th>Title</th>
                <th>Artist</th>
        </tr>
        <xsl:for-each select="catalog/cd">
                <xsl:if test="price &gt; 10">
                <tr>
                        <td>
                                <xsl:value-of select="title"/>
                        </td>
                        <td><
                                xsl:value-of select="artist"/>
                        </td>
                </tr>
                </xsl:if>
        </xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

<xsl:choose> Element

- The <xsl:choose> element is used in conjunction with <xsl:when> and <xsl:otherwise> to express multiple conditional tests.

- **Syntax**

```
<xsl:choose>
        <xsl:when test="expression">
                ... some output ...
        </xsl:when>
        <xsl:otherwise>
                ... some output ....
        </xsl:otherwise>
</xsl:choose>
```

<xsl:apply-templates> Element

- The <xsl:apply-templates> element applies a template to the current element or to the current element's child nodes.
- If we add a select attribute to the <xsl:apply-templates> element it will process only the child element that matches the value of the attribute. We can use the select attribute to specify the order in which the child nodes are processed.
- Look at the following XSL style sheet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
 <html>
 <body>
 <h2>My CD Collection</h2>
 <xsl:apply-templates/>
 </body>
 </html>
</xsl:template>

<xsl:template match="cd">
 <p>
 <xsl:apply-templates select="title"/>
 <xsl:apply-templates select="artist"/>
 </p>
</xsl:template>
</xsl:stylesheet>
```

Explain transforming with XSLT.

- It is possible to convert an XML document to XHTML using the browser's own parser. However, this is not always possible:
    - The browser at the client end may not be suitable or equipped to do the transformation.
    - It may not be a good idea to include the reference to the style sheet or even have the style

sheet available!

- The answer to this process the document and style sheet outside of the browser's own mechanism for doing this task.
- This task can be done either on the client side or the server side.

Using JavaScript

- One way to process and transform XML on the client side is using JavaScript, which has several features for doing the task very well.

```
<html>
<body>
<script type="text/javascript">
//Load the XML document
var xml= new ActiveXObject("Microsoft.XMLDOM")
xml.async=false
xml.load("lib.xml")
//Load the XSL document
var xsl= new ActiveXObject("Microsoft.XMLDOM")
xsl.async= false
xsl.load("libstyle.xsl")
//Do the actual transform
document.write(xml.transformNode(xsl))
</script>
</body>
</html>
```

- Above example shows one way to transform with JavaScript using Microsoft's proprietary Application Programming Interface (API) for the Internet Explorer browser.
- It is also possible to process XML using the DOM.
- Using both the of these mechanisms it is possible to also traverse an XML document and process either according to a style sheet or simply using the JavaScript to make the stylistic decisions.
- Apart from JavaScript, it is also possible to use other programming languages (such as Java and .Net) to process and then output a transformed document.

History of PHP, Apache Web Server, MySQL and Open Source

Open Source

- In general, open source refers to any program whose source code is made available for use or modification.
  Open source software is usually developed as a public collaboration and made freely available. It means can be used without purchasing any license.
- Open Source is a certification mark owned by the Open Source Initiative (OSI). Developers of software that is intended to be freely shared and possibly improved and redistributed by others can use the Open Source trademark if their distribution terms conform to the OSI's Open Source Definition. To summarize, the Definition model of distribution terms require that:
  - The software being distributed must be redistributed to anyone else without any restriction.
  - The source code must be made available (so that the receiving party will be able to improve or modify it).
- Example of Open Source: Linux, Apache, MySQL, PHP.

PHP

- **PHP** is a general-purpose server-side scripting language originally designed for web development to produce dynamic websites.
- PHP scripts execute on web server and serve WebPages to user on request.
- PHP was originally created by RasmusLerdorf in 1994. Programmer RasmusLerdorf initially created a set of C scripts he called "Personal Home Page Tools" to maintain his personal homepage. The scripts performed tasks such as displaying his résumé and recording his web-page traffic.
- These were released and extended to include a package called the Form Interpreter (PHP/FI). While PHP originally stood for "Personal Home Page", it is now said to stand for "PHP: Hypertext Preprocessor", a recursive acronym.
- PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. It also has evolved to include a command-line interface capability and can be used in standalone graphical applications.PHP can be deployed on most web servers and as a standalone interpreter, on almost every operating system and platform free of charge.
- In 1997 ZeevSuraski and AndiGutmans along with Rasmus rewrite PHP and released PHP version 3.0 in June 1998.After this release PHP becomes so much popular.
- The PHP version 4.0 was launched in May 2000.This version includes session handling, output buffering, a richer cire language and support for wide variety of web server platforms.
- The PHP 5.0 version released in 2004 with object oriented programming concept.

Web Server

- A Web Server is computer and the program installed on it. Web Server interacts with the client through the browser. It delivers the web pages to the client and to an application by using the web browser and HTTP protocol respectively.
- We can also define the web server as the package of larger number of programs installed on a computer connected to internet or intranet for downloading the requested files using File Transfer Protocol, serving e- mail and building and publishing web pages.

- A web server works on client server model. A computer connected to internet or intranet must have a server program.
- A computer connected to the internet for providing the services to a small company or a department store may contain the HTTP server to access and store WebPages and files, SMTP server to support mail services, FTP server for files downloading and NNTP server for newsgroup.
- The computer containing all the above servers is called the web server.

Apache Web Server

- The Apache web Server, commonly referred to as Apache is web server software notable for playing a key role in the initial growth of the World Wide Web.
- The first version of Apache, based on the NCSA httpd Web server, was developed in 1995. The Apache server has been developed by an open source community - Apache Software Foundation, whose members are constantly adding new useful functionalities, with the sole purpose of providing a secure and extensible server platform that ensures HTTP service delivery in accordance with the current HTTP standards.
- The original version of Apache was written for UNIX, but there are now versions that run under OS/2, Windows and other platforms.
- The Apache Server provides full range of Web Server features, including CGI, SSL and virtual domains. Apache also supports plug-in modules for extensibility.
- It was called Apache because it was developed from existing NCSA code plus various patches, hence the name a patchy server, or Apache server.
- Apache is open source free software distributed by the Apache Software Foundation.
- Apache is reliable, free and relatively easy to configure.

MySQL

- MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases.
- It is named after developer Michael Widenius' daughter, My. The SQL phrase stands for Structured Query Language.
- The data in MySQL is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.
- The MySQL development project has made its source code available under the terms of the License. The license can require improved versions of the software to carry a different name or version from the original software.
- First released in January,1998, MySQL was owned and sponsored by the Swedish company MySQL AB, now owned by Oracle Corporation.
- MySQL is fully multithreaded using kernel threads, and provides application programming interfaces (APIs) for many programming languages, including C,C++, Java, Perl, PHP, Python.
- MySQL is used in a wide range of applications, including data warehousing, e-commerce, Web databases, logging applications and distributed applications.

Relationship between Apache, MySQL and PHP (AMP Module)

- AMP stands for Apache MySQL PHP

PHP

- PHP is a server side scripting that was designed for creating dynamic websites. It slots into your Apache web server and processes instructions contained in a web page before that page is sent through to your web browser.
- PHP is a powerful scripting language that can be run in the command line of any computer with PHPinstalled. However, PHP alone isn't enough in order to build dynamic web sites.

Apache

- To use PHP on a web site, you need a server that can process PHP scripts. Apache is a free web Server that, once installed on a computer, allows developers to test PHP scripts locally; this makes it an invaluable piece of your local development environment.
- Like all web servers, Apache accepts an HTTP request and serves an HTTP response.

MySQL

- Additionally, dynamic websites are dependent on stored information that can be modified quickly and easily; this is the main difference between a dynamic site and a static HTML site. However, PHP doesn't provide a simple, efficient way to store data. This is where a relational database management system like MySQL comes into play. PHP provides native support for it and the database is free, open-source project.
- MySQL is a relational database management system (DBMS). Essentially, this means that MySQL allows users to store information in a table-based structure, using rows and columns to organize different pieces of data.

Explain Array in PHP

Array is a group of variable that can store multiple values under a single name.

In PHP, there are two types of array.

- Numeric Array
- Associative Array
- Multidimensional Array

Numeric Array

- In numeric array each element having numeric key associated with it that is starting from 0;
- You can use array() function to create array.
- The general syntax is given below:
  $array_name=array (value1, value2 …valueN);
- For Example:

```
<?php
        $myarray=array('A','B','C');
        print_r($myarray);
?>
```

Output:

```
Array( [0]=>A [1]=>B [2]=>C)
```

- You can refer to individual element of an array in PHP script using its key value as shown below:

```
<?php
        $myarray=array('A','B','C');
        echo $myarray[1];
?>
```

It will display

B

- In Numeric Array you can use for, while or do while loop to iterate through each element in array because in numeric array key values are consecutive.

```
<?php
        $myarray=array("Apache", "MySQL", "PHP");
        for($i=0;$i<3;$i++)
        {
            echo $myarray[$i]."<br>";
        }
?>
```

Output:

Apache
MySQL
PHP

Associative Array

- The associative part means that arrays store element values in association with key values rather than in a strict linear index order.
- If you store an element in an array, in association with a key, all you need to retrieve it later from that array is the key value.
- Key may be either numeric or string.
- You can use array() function to create associative array.
- The general syntax is given below:

```
$array_name=array(key1=>value1, key1=>value1,….. keyN=>valueN);
```

For Example 1:

```
<?php
        $myarray=array(5=>”Apple”, 10=>”Mango”, 20=>”Grapes”);
        print_r($my_array);
?>
```

Output:

Array([5]=>Apple [10]=>Mango [20]=>Grapes)

Example 2:

```
<?php
        $myarray=array(“Name”=>”James” , “Age”=>25, “Gender”=>”Male” );
        print_r($myarray);
?>
```

Output:

Array([Name]=>James [Age]=>25 [Gender]=>Male)

- You can refer to individual element of an array in PHP using its key value.

Example:

```
<?php
        $myarray=array(“Name”=>”James” , “Age”=>25, “Gender”=>”Male” );
        echo “Name:”.$myarray[‘Name’];
?>
```

Output:

Name:James

- In associative array you cannot use for, while r do..while loop to iterate through each element in array because in Associative array key value are not consecutive.
- So you have to use foreach loop.

For Example:

```
<?php
    $myarray=array(“Name”=>”James” , “Age”=>25, “Gender”=>”Male” );
    foreach($myarray as $item)
    {
        echo $item;
    }
?>
```

Output:

James 25 Male

Multidimensional Array

PHP can easily support multidimensional arrays, with arbitrary numbers of key.And just as in one dimensional arrays, there is no need to declare out intentions in advance. Assignment can be like:

```
$multi_array[0][1][2][3]=”Good Morning”;
```

That is four dimensional array with numerical index.

The values those are stored in array can themselves be arrays, just as legitimately as they can be string or numbers.

For example:

```
$multi_level_array[0]=”a simple string”;
$multi_level_array[1][‘contains’]=”a string stored deeper”;
```

The integer key of 0 stores a string, and the key 1 stores an array that ,in turn, has a string in it.

Creating array within another array using array() construct is as follows:

```
$basket=array( ‘fruit’ => array(‘red’ => ‘apple’,
                                ‘yellow’ => ‘banana’,
                                ‘black’ => ‘grapes’),
               ‘flower’ =>array(‘red’ => ‘rose’,
                                ‘yellow’ => ‘sunflower’,
                                ‘purple’ => ‘iris’)
               );
```

It is simply an array with two values stored in association with keys.Each of them values is an array itself. We can reference it like this

```
echo $basket[‘fruit’][‘black’];
```

will print grapes. And

```
$kind=”flower”;
$color=”yellow”;
print(“ $basket [$kind] [$color]”);
```

will print sunflower.

How do you create session & cookie in PHP? Give difference between session and cookie with example. OR

What are cookies? Explain the cookies handling in PHP with proper example.

PHP Sessions

- A PHP session variable is used to store information about, or change settings for a user session.
- Session variables hold information about one single user, and are available to all pages in one application.

PHP Session Variables

- When you are working with an application, you open it, do some changes and then you close it.
- This is much like a Session. The computer knows who you are. It knows when you start the application and when you end.
- But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.
- A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.
- Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

Starting a PHP Session

- Before you can store user information in your PHP session, you must first start up the session.
- The session_start() function must appear BEFORE the <html> tag:

```
<?phpsession_start(); ?>
<html>
<body>
</body>
</html>
```

- The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

Storing a Session Variable

- The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:

```
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>
<html>
<body>
<?php
//retrieve session data
echo "Pageviews=". $_SESSION['views'];
?>
</body>
</html>
```

- **Output:**Pageviews=1
- In the example below, we create a simple page-views counter. The isset() function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable, and set it to 1:

```
<?php
session_start();

if(isset($_SESSION['views']))
$_SESSION['views']=$_SESSION['views']+1;
else
$_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

Destroying a Session

- If you wish to delete some session data, you can use the unset() or the session_destroy() function.
- The unset() function is used to free the specified session variable:

```
<?php
unset($_SESSION['views']);
?>
```

- You can also completely destroy the session by calling the session_destroy() function:

```
<?php
session_destroy();
?>
```

- **Note:**session_destroy() will reset your session and you will lose all your stored session data.

PHP Cookie

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer.
- Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

How to Create a Cookie?

- The setcookie() function is used to set a cookie.
- **Note:** The setcookie() function must appear BEFORE the <html> tag.

```
setcookie(name, value, expire, path, domain);
```

- In the example below, we will create a cookie named "user" and assign the value "Alex Porter" to it. We also specify that the cookie should expire after one hour:

```
<?php
setcookie("user", "Alex Porter", time()+3600);
?>
<html>
.....
```

- **Note:** The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).
- You can also set the expiration time of the cookie in another way. It may be easier than using seconds.

```
<?php
$expire=time()+60*60*24*30;
setcookie("user", "Alex Porter", $expire);
?>
<html>
.....
```

- *In the example above the expiration time is set to a month (60 sec * 60 min * 24 hours * 30 days).*

How to Retrieve a Cookie Value?

- The PHP $_COOKIE variable is used to retrieve a cookie value.
- In the example below, we retrieve the value of the cookie named "user" and display it on a page:

```
<?php
// Print a cookie
echo $_COOKIE["user"];

// A way to view all cookies
print_r($_COOKIE);
?>
```

In the following example we use the isset() function to find out if a cookie has been set:

```
<html>
<body>

<?php
if (isset($_COOKIE["user"]))
  echo "Welcome " . $_COOKIE["user"] . "!<br />";
else
  echo "Welcome guest!<br />";
?>

</body>
</html>
```

How to Delete a Cookie?

- When deleting a cookie you should assure that the expiration date is in the past.

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time()-3600);
?>
```

What if a Browser Does NOT Support Cookies?

- If your application deals with browsers that do not support cookies, you will have to use other methods to pass information from one page to another in your application. One method is to pass the data through forms (forms and user input are described earlier in this tutorial).\
- The form below passes the user input to "welcome.php" when the user clicks on the "Submit" button:

```
<html>
<body>
<form action="welcome.php" method="post"> Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

- Retrieve the values in the "welcome.php" file like this:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
```

How can you connect to database in PHP? Show the simple database operation using PHP with proper example.

Integration of PHP with MySQL

- It is possible to execute various commands of MySQL from PHP.PHP provides various built-in functions which you allows you to use MySQL commands from PHP page. Thus you can integrate PHP with MySQL.
- Following are the various PHP functions that allows you the facility of integrating PHP with MySQL:

mysql_connect()

- Before you can access data in a database, you must create a connection to the database.
- This function allows you to establish connection of PHP application with MySQL server.
- **Syntax:** mysql_connect(servername,username,password);

servername	• Indicates the name of MySQL server with which you want to establish connection. • It is optional .Default value is localhost:3306
UserName	• Indicates the name of user using which you can logs on to MySQL Server. • Optional. Default value is the name of the user that owns the server process.
Password	• Indicates password of the user using which you can logs on to MySQL Server. • It is optional. Default is "".

- If connection establish successfully with MySQL Server then this function returns TRUE value otherwise it returns FALSE.
- Example

```
<?php
        $conn=mysql_connect("localhost","root","");
        if($conn)
        {
                echo "Connected with MySQL";
        }
        else
        {
                echo "Could not connect to database";
        }
?>
```

mysql_select_db()

- This function allows you to select database from the list of MySQL server databases.
- **Syntax:** mysql_select_db (DatabaseName, ConnectionName);

DatabaseName	• Indicates the name of the database that you want to select.
ConnectionName	• Indicates the name of the variable that is used at the time of establish connection with MySQL server using mysql_connect() function.

- If function successfully executed then it returns TRUE otherwise it returns FALSE.
- **Example:**

```
<?php
        $conn=mysql_connect("localhost","root","");
        $db=mysql_select_db("Mydatabse");
        if($db)
        {
                echo "Database Selected Successfully ";
        }
        else
        {
                echo "Error in selecting Database.";
        }
?>
```

mysql_query()

- This function allows you to specify and execute the MySQL command on MySQL Server.
- **Syntax:**

  mysql_query(Query, ConnectionName);

Query	• Indicates the MySQL command to be executed.
ConnectionName	• Indicates the name of the variable that is used at the time of establish connection with MySQL server using mysql_connect() function.

- **Example**

```
<?php
        $conn=mysql_connect("localhost","root","");
        $db=mysql_select_db("Mydatabse", $conn);
        $cmd=mysql_query("create table Test(ID integer, Name varchar(20))");
        if($cmd)
        {
                echo "Table created Successfully ";
        }
        else
        {
                echo "Error in executing query.";
        }
?>
```

mysql_fetch_row()

- This function allows you to retrieve a record from the recordset that is returned from executing the MySQL query.

- The record that is returned by this function is in the form of numeric array. Numeric array contains index and value associated with that index.
- If there is no record in the record set then it returns false value.
- **Syntax:**

  mysql_fetch_row(VariableName);
- **VariableName** indicates the record set that is returned from executing the MySQL command using mysql_query() function.
- **Example:**

```
<?php
        $conn=mysql_connect("localhost","root","");
        $db=mysql_select_db("Mydatabse", $conn);
        $query="select * from product_master";
        $result=mysql_query($query,$conn);
        $ans=mysql_fetch_row($result);
        print_r($ans);
        mysql_close($con);
?>
```

mysql_fetch_array()

- This function allows you to retrieve a record from the recordset that is returned from executing the MySQL query.
- The record that is returned by this function is in the form of either numeric array, associative array or both.
- If there is no record in record set then it will returns false value.
- **Syntax:**

  mysql_fetch_array(VariableName, ResultArrayType)
- **VariableName:-** indicates the record set that is returned from executing the MySQL command using mysql_query() function.
- **ResultArrayType:-**indicates the type of array to be returned. It can have one of the following values:

MYSQL_ASSOC	This type of array contains name of the field and the value associated with that field for current record.
MYSQL_NUM	This type of array contains index of the field and the value associcated with that index for current record.
MYSQL_BOTH	It is combination of both Associative array and Numeric array. It is the default type to be returned by this function.

- **Example:**

```
<?php
        $conn=mysql_connect("localhost","root","");
        $db=mysql_select_db("Mydatabse", $conn);
        $query="select * from product_master";
        $result=mysql_query($query, $conn);
        $ans=mysql_fetch_array($result,MYSQL_ASSOC);
        print_r($ans);
        mysql_close($con);
?>
```

mysql_fetch_assoc()

- This function allows you to retrieve a record from the recordset that is returned from executing the MySQL query in the form of associative array.
- **Syntax:**

  mysql_fetch_assoc(VariableName)
- Returns an associative array that corresponds to the fetched row, or `FALSE` if there are no more rows.
- **Example:**

```
<?php
        $conn=mysql_connect("localhost","root","");
        $db=mysql_select_db("Mydatabse", $conn);
        $query="select * from product_master";
        $result=mysql_query($query, $conn);
        $ans=mysql_fetch_assoc($result);
        print_r($ans);
        mysql_close($con);
?>
```

To go through all the records fetched in record set.

```
<?php
        $conn=mysql_connect("localhost","root","");
        $db=mysql_select_db("Mydatabse", $conn);
        $query="select * from product_master";
        $result=mysql_query($query, $conn);
        while($ans=mysql_fetch_assoc($result))
        {
                echo $ans['field1']."<br>";
                echo $ans['field2']."<br>";
        }
        mysql_close($con);
?>
```

mysql_num_rows()

- This function allows you to retrieve number of records available in the record set.
- **Syntax:**

  mysql_num_rows(ResultVariable);
- ResultVariable is the variable that holds result returned by mysql_query() function.
- **Example:**

```
<?php
        $conn=mysql_connect("localhost","root","");
        $db=mysql_select_db("Mydatabse", $conn);
        $query="select * from product_master";
        $result=mysql_query($query, $conn);
        $total_records=mysql_fetch_rows($result);
        echo "Total Records:".$total_records;
        mysql_close($con);
?>
```

mysql_error()

- This function allows you to retrieve theerror text from the most recently executed MySQL function.
- If no error encountered while executing the script then it will returns blank string.
- If the mysql operation contains more than one error then it will returns error description of the last statement in which error is encountered.
- **Syntax:**

  mysql_error();

- **Example:**

```
<?php
        $conn=mysql_connect("localhost","root","");
        if($conn)
        {
                echo "Connected with MySQL";
        }
        else
        {
                echo "Error:".mysql_error();
        }
?>
```

mysql_close()

- This function allows you to close the connection that is established using mysql_connect() function.
- **Syntax:**mysql_close(ConnectionName);
- **ConnectionName** :- Indicates the name of the variable that is used at the time of establish connection with MySQL server using mysql_connect() function.
- It returns true is connection is closed successfully otherwise it returns false.

- **Example:**

```
<?php
        $conn=mysql_connect("localhost","root","");
        mysql_close($con);
?>
```