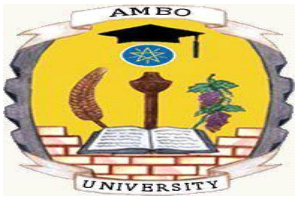


AMBO UNIVERISTY

WOLISO CAMPUS

COLLEGE OF TECHNOLOGY AND INFORMATICS

DEPARTMENT OF INFORMATION TECHNOLOGY

Fundamentals of Internet Programming Handout

By

AYANTU GUYE BERISA

Woliso, Oromia, Ethiopia

June, 2020

Chapter 1

Internet Technologies and Protocols

What is Internet?

- It is a world-wide global system of interconnected computer networks. A ***computer network*** is the interconnection of many individual computers to exchange message. ***Network*** is a group of two or more computer connected together.
- It uses the standard Internet Protocol ***TCP/IP***. Every computer in internet is identified by a unique ***IP address***. IP Address is a unique set of numbers which identifies a computer location. Domain Name server (DNS) is used to give *name to the IP Address* so that user can locate a computer by a name.
  - *For example, a DNS server will resolve a name http://www.google.com to a particular IP address to uniquely identify the computer on which this website is hosted.*
- Internet is accessible to every user all over the world. So, Internet is a network of networks.

Advantage of Internet

- Information sharing
- Communication i.e social networking
- Sharing of resource

Disadvantage of Internet

- Threat to personal information
- Virus attack
- Spamming
- Cyber crime

Terminology

- Learning about the Internet can be a bit confusing at first, but it becomes a lot simpler if you can become familiar with some of the terminology used when talking about the Internet. Here is a list of common words and phrases that you might hear.

A. ***On-line***

- ☞ You may sometimes hear people talk about "***being on-line***". This is just another way of saying that ***they are using the Internet***.

B. ***World-Wide-Web* (WWW)**

- ☞ Tim Berners-Lee, a physicist in Switzerland, invented the World Wide Web in 1992 as a ***way to organize and access information on the Internet***.

C. ***Web browser***

- ☞ A web browser is a program that runs on users' computers and allows them to *view and interact with the web pages on the World Wide Web*. The most common web browsers are called ***Internet Explorer and Google Chrome***.

D. ***Hypertext***

- ➲ Hypertext is a text document that contains links to other text document.
- ➲ It allows a user to ***move from one web page to another*** by using a mouse to click on special ***hypertext links***.
  - ☞ *For example, a user viewing web pages that describe airplanes might encounter a link to jet engines from one of those pages. By clicking on that link, the user automatically jumps to a page that describes jet engines.*

E. ***Webpage***

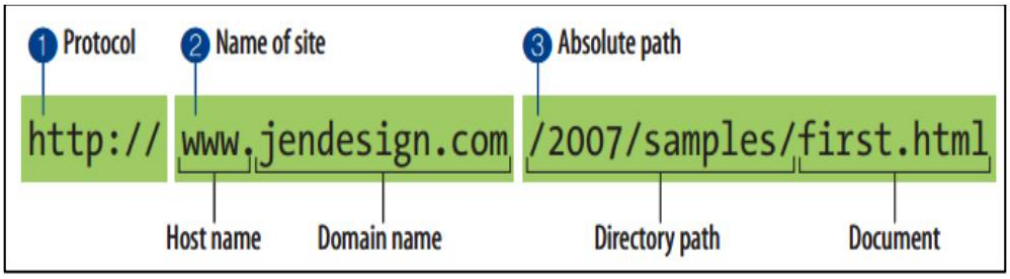
- ➲ Webpage is a single document that may contain text, graphics, and icon.
- ➲ They are created using HTML.

F. ***Web site***

- ➲ It is a group of well-structured and interlinked webpages.

G. ***Uniform Resource Locator* (URL)**

- ➲ To visit a Web site, users type the URL, which is the site's address, into the web browser. An example of a URL is ***www.yahoo.com*.**
- ➲ A complete URL is generally made up of three components: the ***protocol***, the ***site name***, and the ***absolute path*** to the document or resource as shown in the figure below:



H. ***Web server***

- A *web server* is a ***computer that stores a web site***, and is responsible for checking requests for viewing that web site.
- Client computers ***send requests*** for particular URLs to the web server, which then finds the appropriate web page, and sends it back to the client computer.
- A web server on the Internet must have a ***permanent Internet connection***, so that whenever a client computer requests a URL, the web server can respond straight away.

I. ***Internet Service Provider* (ISP)**

- A company that provides Internet connections to customers.

J. ***Protocol***

- It is a set of rules that govern the communication.

K. ***Hypertext Transfer Protocol (HTTP/HTTPS)***

- It is a communications protocol.
- It defines mechanism for communication between ***browser and the web server***. It is also called ***request and response protocol*** because the communication between browser and server takes place in request and response pairs. Simply, it is the means by which ***computers on the WWW communicate***.
- HTTPS is the secure version of HTTP. It is used on web sites where ***sensitive information*** such as bank details is exchanged.

L. ***Hypertext Markup Language (HTML)***

- It is the language used to write web pages on the WWW.

M. ***Extensible Markup Language (XML)***

- It is an alternative language for writing web pages. Whereas HTML pages describe the format of the data's presentation, pages written in XML describe only how the data is structured.

- XML provides a standard format for the movement of data in and between applications.
- The data in an XML file usually requires some other application to interpret the data and display it in a useful format.

N. ***World Wide Web Consortium (W3C)***

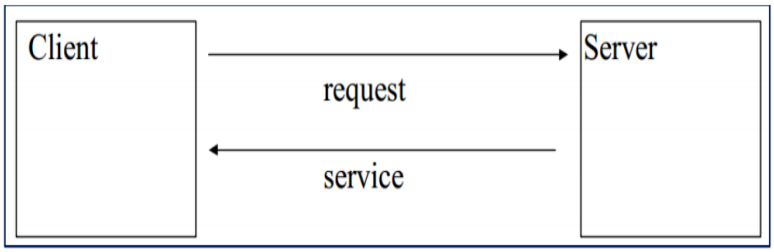
- It is a group of experts who meet regularly to develop common protocols for the evolution of the WWW.
- The W3C agrees on standards for HTML, XML and other web technologies, and for how web browsers should interpret them.

Web development tools

- A number of tools exist for use by web authors (i.e. people who write web sites).
  - ***HTML/XML:*** HTML and XML are the two main languages used for writing web pages. Web authors can use a simple text editor such as ***Notepad*** to enter the *HTML/XML* commands. The final page can then be viewed using a web browser.

Client-server architecture

The data processing is split into distinct parts. A part is either requester (client) or provider (server). The client sends during the data processing one or more requests to the servers to perform specified tasks. The server part provides services for the clients.



How the Web works?

- What happens when a browser requests an HTML document over the Internet?
  1. *The user types a URL into the Web browser to identify which Web page they would like to view.*
  2. *The browser parses the URL and requests a DNS server using broadcast IP. It then sends the URL to the DNS to resolve the IP address. In other words, it converts appdev.com to 206.191.222.239.*

3. *The browser then uses the IP address to send the HTTP packet to the browser's ISP connection, which passes it to the next server, routing it from server to server until it reaches the destination Web server.*

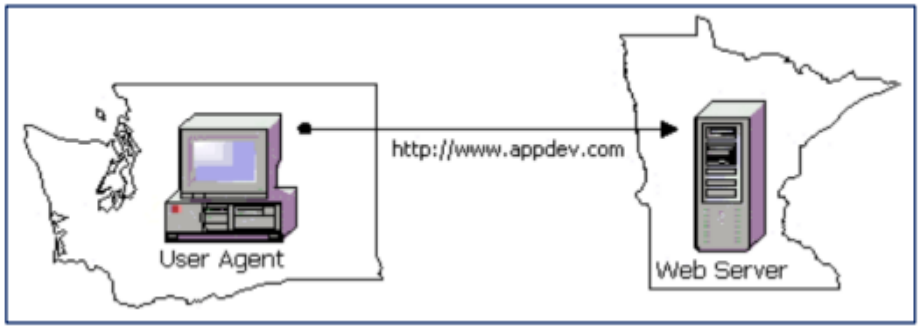


Figure 1: A request is made to the server.

4. *The Web server locates the request page either on its hard drive or cached in memory.*
5. *The Web server sends its contents back to the requested browser.*

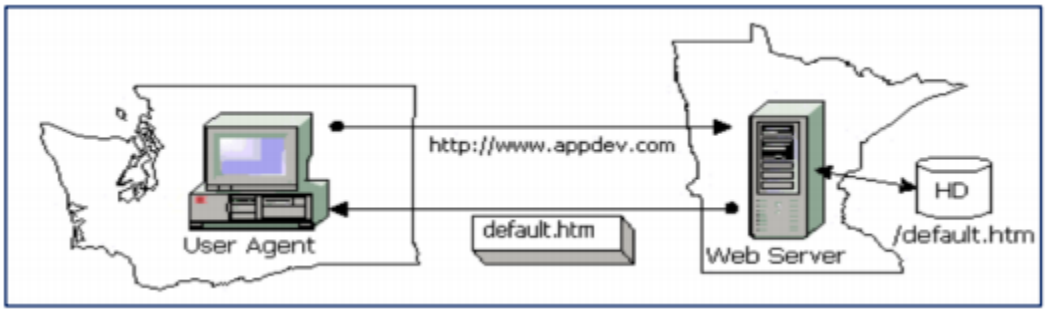


Figure 2: -The page is located and returned to the client machine.

6. *The browser interprets the HTML formatting instructions and displays the content to the user.*

Website validation

1. **Authorship:** Who put up the site?
   - ☞ The name of the individual or organization creating the site should be clearly stated.
   - ☞ The site should list the credentials of the author, whether it is an individual or an organization.
   - ☞ The site should provide a way for users to contact the author and to make comments or ask questions.

2. **Purpose:** Every site has a reason for being on the web
    - ☞ A site's purpose should be clear and its content should reflect its purpose, be it to inform, entertain, persuade, educate or sell.
    - ☞ Bias (if any) should be clearly stated through a mission statement or "about us" section or elsewhere on the site.
3. **Content & Currency:** Is the information authoritative and up to date?
    - ☞ The information should be accurate and the site should be updated regularly, especially if the topic is time-sensitive.
    - ☞ The site's content should be easy to read and easy to understand by its intended audience.
    - ☞ The site should offer enough information to make it worth visiting.
4. **Technical Aspects**
    - ☞ A search function should be provided for sites with large amounts of information.
    - ☞ You should not have to pay to view the information on the site.
    - ☞ Spelling and grammar should always be correct.
    - ☞ Links to more information should be provided.
    - ☞ Graphics on the site should be relevant and appropriate to the content.
    - ☞ Advertising should be limited.

Putting it all together: If the website you found provides:

- ☞ Author name, acceptable author credentials and a way to contact the author.
- ☞ A clear statement of purpose or mission.
- ☞ accurate information (as measured by the citations for information on the site OR by what you already know about the topic OR by comparing it to information from an authoritative source)
- ☞ up-to-date information

Chapter 2

HTML

What is Markup language?

- A markup language is a programming language that is used to make text more ***interactive and dynamic***. It can turn a text into images, tables, links etc.

What is Hyper Text Markup Language (HTML)?

- *It is the standard markup language for creating Web pages.*
- *It is a language for describing web pages.*
- *HTML documents contain **HTML tags and plain text***
- *HTML documents are also called **web pages***
- *It consists of a series of elements*
- *HTML elements tell the browser how to display the content*
- *HTML elements are represented by tags*
- *HTML tags label pieces of content such as heading, paragraph, table, and so on*
- *Browsers do not display the HTML tags, but use them to render the content of the page*

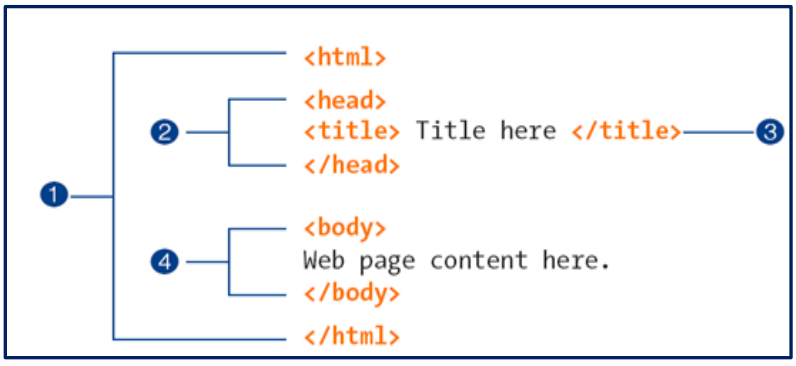
What is HTML Tag?

- HTML tags are element names surrounded by angle brackets:

  <tagname>** content goes here...**</tagname>

- HTML tags normally come in pairs like and
- The first tag in a pair is the ***start tag***, the second tag is the ***end tag***
- The end tag is written like the start tag, but with a ***forward slash inserted before the tag name***

The structure of an HTML element



The minimal structure of HTML documents:

1. Identifies the document as written in HTML.
2. The head provides information about the document.
3. A descriptive title is required.
4. The body contains the content that displays in the browser.

HTML Element

- The **<!DOCTYPE html>** declaration defines this document to be ***HTML version 5***
- The **<html>** element is the ***root element of an HTML page***
- The **<head>** element contains ***meta information about the document***
- The **<title>** element specifies a ***title for the document***
- The **<body>** element contains the ***visible page content***
- The **<h1>** element defines a ***large heading***
- The **<p>** element defines a ***paragraph***

Heading Tags

- Any document starts with a heading. You can use different sizes for your headings.
- HTML also has six levels of headings, which use the elements <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>. While displaying any heading, *browser adds one line before and one line after that heading.*

Example

```
<!DOCTYPE html>
<html>
<head> <title>Heading Example</title> </head>
<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>
```

Paragraph Tag

- The tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening and a closing tag as shown below in the example:

```
<!DOCTYPE html>
<html>
<head> <title>Paragraph Example</title> </head>
<body>
<p>Here is a first paragraph of text.</p>
<p>Here is a second paragraph of text.</p>
<p>Here is a third paragraph of text.</p>
</body>
</html>
```

Line Break Tag

- Whenever you use the ***
 element***, anything following it starts from the next line. This tag is an example of an ***empty element***, where you do not need opening and closing tags, as there is nothing to go in between them. The
 tag has a space between the characters br and the forward slash. If you omit this space, older browsers will have trouble rendering the line break.

Example

```
<!DOCTYPE html>
<html>
<head> <title>Line Break  Example</title> </head>
<body>
<p>Hello<br />
You delivered your assignment on time. <br />
Thanks<br />
Mahnaz</p>
</body>
</html>
```

Centering Content

- You can use * tag* to put any content in the center of the page or any table cell.

```
<!DOCTYPE html>
<html>
<head>
<title>Centring Content Example</title> </head>
<body> <p>This text is not in the center.</p>
<center> <p>This text is in the center.</p> </center>
</body>
</html>
```

Horizontal Lines

- Horizontal lines are used to visually ***break-up sections of a document***.
- The ***<hr> tag*** creates a line from the current position in the document to the right margin and breaks the line accordingly. For example, you may want to give a line between two paragraphs as in the given example below:

```
<!DOCTYPE html>
<html>
<head> <title>Horizontal Line Example</title> </head>
<body>
<p>This is paragraph one and should be on top</p>
<hr />
<p>This is paragraph two and should be at bottom</p>
</body>
</html>
```

- Again ***<hr /> tag*** is an example of the ***empty element***, where you do not need opening and closing tags, as there is nothing to go in between them. The ***<hr /> element*** has a space between the characters ***hr and the forward slash***. If you omit this space, older browsers will have trouble rendering the horizontal line.

Nonbreaking Spaces

- In cases, where you do not want the client browser to break text, you should use a nonbreaking space entity *** ***; instead of a normal space.

- For example, when coding the "12 Angry Men" in a paragraph, you should use something similar to the following code:

```
<!DOCTYPE html>
<html>
<head> <title>Nonbreaking Spaces Example</title> </head>
<body>
<p>An example of this technique appears in the movie "12 Angry Men."</p>
</body>
</html>
```

HTML Links

- HTML links are defined with the ***<a> tag***: *link text*. The link's destination is specified in the **href** attribute. **Attributes** are used to provide **additional information about HTML elements**.

Example: -Visit our HTML tutorial

HTML image

- HTML images are defined with the **<img>** tag.
- The source file (src), alternative text (alt), width, and height are provided as attributes.

Example: -<img src="smiley.gif" alt="Smiley face" height="42" width="42">

HTML Lists

- HTML lists are defined with the * (unordered/bullet list)* or the * (ordered/numbered list) tag*, followed by * tags* (list items):

Example

```
<!DOCTYPE html>
<html>
<head> <title>HTML list Tag</title>  </head>
<body>
 <h1>this is first group</h1>
 <p>Following is a list of vegetables</p>
 <ol>
        <li>Beetroot</li>
```

```
        <li>Ginger</li>
        <li>Potato</li>
        <li>Radish</li>
    </ol>
    <h2>this is second group</h2>
    <p >Following is a list of fruits</p>
    <ul>
        <li>Apple</li>
        <li>Banana</li>
        <li>Mango</li>
        <li>Strawberry</li>
    </ul>
    </body>
    </html>
```

HTML Comments

- Comment is a piece of code which is ignored by any web browser. It is a good practice to add comments into your HTML code, especially in complex documents, to indicate sections of a document, and any other notes to anyone looking at the code. Comments help you and others understand your code and increases code readability.
- HTML comments are placed in between ***<!-- ... --> tags***. So, any content placed with-in **<!-- ... -->** tags will be treated as comment and will be completely ignored by the browser. In HTML there are three types of comments those are described as follows:

1. *Single comment*

Example

```
<!DOCTYPE html>
<html>
<head> <!-- Document Header Starts --> <title>This is document title</title>
</head> <!-- Document Header Ends -->
<body>
  <p>Document content goes here.....</p>
</body>
```

```
</html>
```

2. Multiline Comments

➲ So far we have seen single line comments, but HTML supports multi-line comments as well. You can comment multiple lines by the special beginning tag ***<!-- and ending tag --*** > placed before the first line and end of the last line as shown in the given example below. ***Example***

```
<!DOCTYPE html>
<html>
<head> <title>Multiline Comments</title> </head>
 <body>
    <!--
      This is a multiline comment and it can
      span through as many as lines you like.
    -->
 <p>Document content goes here.....</p>
</body> </html>
```

3. Conditional Comments

➲ Conditional comments only work in Internet Explorer (IE) on Windows but they are ignored by other browsers. They are supported from Explorer 5 onwards, and you can use them to give conditional instructions to different versions of IE. ***Example***

```
<!DOCTYPE html>
<html>
 <head>   <title>Conditional Comments</title>
    <!--[if IE 6]>
      Special instructions for IE 6 here
    <![endif]-->
  </head>
  <body>
  <p>Document content goes here.....</p>
  </body>
 </html>
```

HTML Frames

- HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

Disadvantages of Frames

- There are few drawbacks with using frames, so it's never recommended to use frames in your webpages. Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up. Sometimes your page will be displayed differently on different computers due to different screen resolution.
- The browser's back button might not work as the user hopes. There are still few browsers that do not support frame technology.

Creating Frames

- To use frames on a page we use <frameset> tag instead of <body> tag.
- The <frameset> tag defines how to divide the window into frames.
- The rows attribute of <frameset> tag defines horizontal frames and cols attribute defines vertical frames.
- Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

Following is the example to create three horizontal frames:

```
<!DOCTYPE html>
<html>
 <head>  <title>HTML Frames</title>  </head>
 <frameset rows = "10%,80%,10%">
 <frame name = "top" src = "/html/top_frame.htm" />
 <frame name = "main" src = "/html/main_frame.htm" />
 <frame name = "bottom" src = "/html/bottom_frame.htm" />
 <noframes>
 <body>Your browser does not support frames.</body>
 </noframes>
 </frameset>
```

```
</html>
```

Let's put the above example as follows, here we replaced rows attribute by ***cols*** and changed their width. This will create all the three frames vertically:

```
<!DOCTYPE html>
<html>
 <head>  <title>HTML Frames</title>  </head>
 <frameset cols = "25%, 50%, 25 %">
 <frame name = "left" src = "/html/top_frame.htm" />
 <frame name = "center" src = "/html/main_frame.htm" />
 <frame name = "right" src = "/html/bottom_frame.htm" />
 <noframes>
 <body>Your browser does not support frames.</body>
 </noframes>
</frameset>
</html>
```

HTML Tables

- HTML Tables is used to preset data in rows and columns format. A Table is the collection of rows and rows is the collection of columns. <tr> stands for table rows. To add a row in a table ***<table row> tags*** are used. <td> or <th> is used to put the column inside the row. Whereas <td> means table data and <th> means table head.

Syntax:

```
<table>
  <tr>
    <td>row1col1</td>
    <td>row1col2</td>
  </tr>
</table>
```

Example

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Email</th>
```

```
</tr>
<tr>
   <td>phptpoint</td>
  <td>phptpoint@gmail.com</td>
</tr>
<tr>
  <td>phptpoint blog</td>
 <td>blog@gmail.com</td>
</tr>
</table>
```

- In the above example a table is created have 3 rows and 6 columns where each row contains 2 column. <tr> tag is used to create a row while <td> or <th> is used to create column. <tr> comes in between <table> tag while <td> or <th> comes in between <tr>.

How to merge table column

- When you want to merge 2 or more than 2 column (cell), use ***colspan*** property of <td colspan="2″> or <th colspan=”2″>. *Example*

```
<table border="1">
<tr>
   <th colspan="2"> User Details</th>
</tr>
<tr>
    <th>Name</th>
   <th>Email</th>
</tr>
<tr>
     <td>phptpoint</td>
    <td>phptpoint@gmail.com</td>
</tr>
<tr>
     <td>phptpoint blog</td>
    <td>blog@gmail.com</td>
```

```
</tr> </table>
```

How to merge table rows

- When you want to merge 2 or more than 2 rows in a single row, use ***rowspan*** property of <td rowspan=”2″> or <th rowspan=”2″>
- ***Example***

```
<table  border="1">
    <tr>
        <td rowspan="2"> </td>
        <td> </td>
    </tr>
    <tr>
        <td> </td>
    </tr>
</table>
```

Nested Table

- Nested table means how to use table inside a table. Multiple times you need to use table inside a table. When you want to use a table inside a table write the syntax of table in between your cell i.e either <td> or <th>.
- ***Example***

```
<html>
<body>
<table border="1" bgcolor="gray" width="200" height="200">
<tr>
<th>
<table align="center" border="1" bgcolor="#F8F8F8" width="100"
height="100">
<tr>
<th>Inner Table</th>
</tr>
</table>
</th>
```

```
</tr>
</table>
</body>
</html>
```

Definition and Usage

- The <table> tag defines an HTML table. An HTML table consists of the <table> element and one or more <tr>, <th>, and <td> elements.
- The <tr> element defines a table row, the <th> element defines a table header, and the <td> element defines a table cell. A more complex HTML table may also include <caption>, <col>, <colgroup>, <thead>, <tfoot>, and <tbody> elements.

HTML Buttons

- HTML buttons are defined with the **<button> tag**. The <button> tag defines a ***clickable button***. Inside a <button> element you can put content, like text or images. This is the difference between this element and buttons created with the <input> element.
- ***Example: -***Two button elements that act as one submit button and one reset button (in a form):

```
<form action="/action_page.php" method="get">
        First name: <input type="text" name="fname"><br>
        Last name: <input type="text" name="lname"><br>
  <button type="submit" value="Submit">Submit</button>
  <button type="reset" value="Reset">Reset</button>
</form>
```

HTML Forms

- HTML Forms are required, when you want to collect some data from the site visitor.
- For example, during user registration you would like to collect information such as name, email address, credit card, etc.
- A form will take input from the site visitor and then will post it to a back-end application such as CGI, ASP Script or PHP script etc.
- The back-end application will perform required processing on the passed data based on defined business logic inside the application.

- There are various form elements available like text fields, text-area fields, drop-down menus, radio buttons, checkboxes, etc.
- The HTML ***<form> tag*** is used to create an HTML form and it has following syntax:

```
<form action = "Script URL" method = "GET|POST">
  form elements like input, textarea etc.
</form>
```

Form Attributes

- Apart from common attributes, following is a list of the most frequently used form attributes

S.No	***Attribute***	***Description***
1	*Action*	*Backend script ready to process your passed data.*
2	*Method*	*Method to be used to upload data. The most frequently used are GET and POST methods.*
3	*Target*	*Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.*
4	*Enctype*	*You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are:* ➲ ***application/x-www-form-urlencoded*** *− This is the standard method most forms use in simple scenarios.* ➲ ***mutlipart/form-data*** *− This is used when you want to upload binary data in the form of files like image, word file etc.*

HTML Form Controls

- There are different types of form controls that you can use to collect data using HTML form:

1. Text Input Controls

- There are three types of text input used on forms:

A. ***Single-line text input controls***: is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag. Here is a basic example of a single-line text input used to take first name and last name

```
<!DOCTYPE html>
<html>
<head> <title>Text Input Control</title>  </head>
<body>
<form >
First name: <input type = "text" name = "first_name" />     <br>
 Last name: <input type = "text" name = "last_name" /
</form>
 </body>
</html>
```

Attributes: -*Following is the list of attributes for <input> tag for creating text field.*

S.No	***Attribute***	***Description***
1	*Type*	*Indicates the type of input control and for text input control it will be set to text.*
2	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
3	*Value*	*This can be used to provide an initial value inside the control.*
4	*Size*	*Allows specifying the width of the text-input control in terms of characters.*
5	*maxlength*	*Allows specifying the maximum number of characters a user can enter into the text box.*

B. ***Password input controls***: This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input> tag. Here is a basic example of a single-line password input used to take user password:

```
<!DOCTYPE html>
<html>
<head>  <title>Password Input Control</title>  </head>
 <body>
<form >
 User ID : <input type = "text" name = "user_id" />      <br>
 Password: <input type = "password" name = "password" />
```

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

Attributes: -*Following is the list of attributes for <input> tag for creating password field.*

S.No	***Attribute***	***Description***
1	*Type*	*Indicates the type of input control and for password input control it will be set to password.*
2	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
3	*Value*	*This can be used to provide an initial value inside the control.*
4	*Size*	*Allows specifying the width of the text-input control in terms of characters.*
5	*maxlength*	*Allows specifying the maximum number of characters a user can enter into the text box.*

C. ***Multi-line text input controls***: This is used when the user is required to give details that may be longer than a single sentence. They are created using HTML <textarea> tag. Here is a basic example of a multi-line text input used to take item description:

```
<!DOCTYPE html>
<html>
<head> <title>Multiple-Line Input Control</title>  </head>
<body>
  <form>
      Description : <br />
    <textarea rows = "5" cols = "50" name = "description">
        Enter description here...
   </textarea>
 </form>
</body>
</html>
```

Attributes: -*Following is the list of attributes for <textarea> tag.*

S.No	***Attribute***	***Description***
1	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
2	*Rows*	*Indicates the number of rows of text area box.*
3	*Cols*	*Indicates the number of columns of text area box*

2. Checkbox Control

➲ *Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to checkbox.Here is an example HTML code for a form with two checkboxes:*

```
<!DOCTYPE html>
<html>
<head> <title>Checkbox Control</title> </head>
<body>
<form>
    <input type = "checkbox" name = "maths" value = "on"> Maths
    <input type = "checkbox" name = "physics" value = "on"> Physics
</form>
</body>
</html>
```

Attributes: -*Following is the list of attributes for <checkbox> tag.*

S.No	***Attribute***	***Description***
1	*Type*	*Indicates the type of input control and for checkbox input control it will be set to checkbox.*
2	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
3	*Value*	*The value that will be used if the checkbox is selected.*
4	*Checked*	*Set to checked if you want to select it by default.*

3. Radio Button Control

- *Radio buttons are used when out of many options; just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to radio. Here is example HTML code for a form with two radio buttons*

```
<!DOCTYPE html>
<html>
<head> <title>Radio Box Control</title> </head>
<body>
   <form>
     <input type = "radio" name = "subject" value = "maths"> Maths
     <input type = "radio" name = "subject" value = "physics"> Physics
   </form>
</body>
</html>
```

Attributes: *-Following is the list of attributes for radio button.*

S.No	***Attribute***	***Description***
1	*Type*	*Indicates the type of input control and for checkbox input control it will be set to radio.*
2	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
3	*Value*	*The value that will be used if the radio box is selected.*
4	*Checked*	*Set to checked if you want to select it by default.*

4. Select Box Control

- *A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options. Here is example HTML code for a form with one drop down box*

```
<!DOCTYPE html>
<html>
<head>  <title>Select Box Control</title>  </head>
<body>
    <form>
        <select name = "dropdown">
```

```
                    <option value = "Maths" selected>Maths</option>
                    <option value = "Physics">Physics</option>
                </select>
            </form>
        </body>
        </html>
```

Attributes: -*Following is the list of important attributes of* ***<select>*** *tag*

S.No	***Attribute***	***Description***
1	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
2	*Size*	*This can be used to present a scrolling list box.*
3	*Multiple*	*If set to "multiple" then allows a user to select multiple items from the menu.*

➲ *Following is the list of important attributes of* ***<option>*** *tag*

S.No	***Attribute***	***Description***
1	*Value*	*The value that will be used if an option in the select box box is selected.*
2	*Selected*	*Specifies that this option should be the initially selected value when the page loads.*
3	*Label*	*An alternative way of labeling options*

5. File Upload Box

➲ *If you want to allow a user to upload a file to your web site, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element but type attribute is set to file. Here is example HTML code for a form with one file upload box*

```
<!DOCTYPE html>
<html>
<head><title>File Upload Box</title></head>
<body>
    <form>
      <input type = "file" name = "file upload" accept = "image/*" />
```

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

Attributes: -*Following is the list of important attributes of file upload box*

S.No	***Attribute***	***Description***
1	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
2	*Accept*	*Specifies the types of files that the server accepts.*

6. Button Controls

➲ *There are various ways in HTML to create clickable buttons. You can also create a clickable button using <input>tag by setting its type attribute to button. The type attribute can take the following values*

S.No	***Attribute***	***Description***
1	*Submit*	*This creates a button that automatically submits a form.*
2	*Reset*	*This creates a button that automatically resets form controls to their initial values.*
3	*Button*	*This creates a button that is used to trigger a client-side script when the user clicks that button.*
4	*Image*	*This creates a clickable button but we can use an image as background of the button.*

Here is example HTML code for a form with three types of buttons

```
<!DOCTYPE html>
<html>
<head> <title>File Upload Box</title> </head>
<body>
<form>
    <input type = "submit" name = "submit" value = "Submit" />
    <input type = "reset" name = "reset"  value = "Reset" />
    <input type = "button" name = "ok" value = "OK" />
    <input type = "image" name = "imagebutton" src = "/html/images/logo.png" />
</form>
```

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

7. *Hidden Form Controls*

➲ Hidden form controls are used to hide data inside the page which later on can be pushed to the server. This control hides inside the code and does not appear on the actual page. For example, following hidden form is being used to keep current page number. When a user will click next page then the value of hidden control will be sent to the web server and there it will decide which page will be displayed next based on the passed current page. *Here is example HTML code to show the usage of hidden control*

```
<!DOCTYPE html>
<html>
 <head> <title>File Upload Box</title> </head>
<body>
   <form>
     <p>This is page 10</p>
     <input type = "hidden" name = "pagename" value = "10" />
     <input type = "submit" name = "submit" value = "Submit" />
     <input type = "reset" name = "reset" value = "Reset" />
   </form>
</body>
</html>
```

Chapter 3

Styling HTML with CSS

CSS was introduced together with HTML 4, to provide a better way to style HTML elements. Styles were added to HTML 4.0 to solve a problem

What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- CSS is a language that describes the style of an HTML document.
- CSS describes how HTML elements should be displayed.
- CSS is used to control the style of a web document in a simple and easy way.

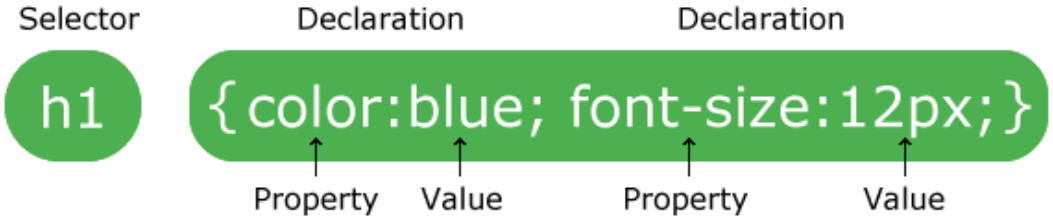
Why Use CSS?

- CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS Syntax and Selectors

CSS Syntax

- A CSS rule-set consists of a selector and a declaration block:



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

Example

In this example all <p> elements will be center-aligned, with a red text color:

```
p {
  color: red;//declaration
  text-align: center;
}
```

CSS Selectors

- CSS selectors are used to "find" (or select) HTML elements based on their element ***name, id, class, attribute***, and more.

The element Selector

- The element selector selects elements based on the element name.

Example

- You can select all <p> elements on a page like this (here, all <p> elements will be center-aligned, with a red text color):

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

The id Selector

- The id selector uses the ***id attribute*** of an HTML element to select a specific element.
- The id of an element should be unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example

The style rule below will be applied to the HTML element with id="para1":

```
#para1 {
 text-align: center;
 color: red;
}
```

Note: An id name cannot start with a number!

The class Selector

- The class selector selects elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the name of the class.

Example

- In this example all HTML elements with class="center" will be red and center-aligned:

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

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

Example

- In this example only <p> elements with class="center" will be center-aligned:

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

- HTML elements can also refer to more than one class.
- Example
- In this example the <p> element will be styled according to class="center" and to class="large":
- `<p class="center large">This paragraph refers to two classes.</p>`

Note: A class name cannot start with a number!

Grouping Selectors

- If you have elements with the same style definitions, like this:

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

h2 {
  text-align: center;
  color: red;
}

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

- It will be better to group the selectors, to minimize the code.

- To group selectors, separate each selector with a comma.
- Example
- In this example we have grouped the selectors from the code above:
- h1, h2, p {
  text-align: center;
  color: red;
  }

CSS Comments

- Comments are used to explain the code, and may help when you edit the source code at a later date.
- Comments are ignored by browsers.

Example

- A CSS comment starts with /* and ends with */. Comments can also span multiple lines:
- p {
  color: red;
  /* This is a single-line comment */
  text-align: center;
  }

CSS can be added to HTML in the following ways:

- Inline - using the style **attribute** in HTML elements
- Internal - using the <style> **element** in the <head> section
- External - using an external CSS **file**
- The preferred way to add CSS to HTML is to put CSS syntax in separate CSS files.

Inline Styles

- An inline style can be used if a ***unique style is to be applied to one single occurrence of an element***.
- To use inline styles, use the style attribute in the relevant tag. The style attribute can contain any CSS property.
- The example below shows how to change the text color and the left margin of a paragraph:

  ** This is a paragraph. **

HTML Style Example - Background Color

- The background-color property defines the background color for an element: **Example**

```
<!DOCTYPE html>
<html>
<body style="background-color:yellow;">
<h2 style="background-color:red;">This is a heading</h2>
<p style="background-color:green;"> This is a paragraph.</p>
</body>
</html>
```

Internal Style Sheet

- An internal style sheet can be used if ***one single document has a unique style***. Internal styles are defined in the <head> section of an HTML page, by using the <style> tag, like this:

```
<head>
    <style>
    body {background-color:yellow;}
    p {color:blue;}
    </style>
</head>
```

External Style Sheet

- An external style sheet is ideal when the style is applied to many pages.
- External Style Sheets can save a lot of work
- External Style Sheets are stored in CSS files
- With an external style sheet, you can change the look of an entire Web site by changing one file.
- Each page must link to the style sheet using the ***<link> tag***. The <link> tag goes inside the <head> section:

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

Example mystle.css

```
body {
  background-color: lightblue;
}
h1 {
  color: navy;
  margin-left: 20px;
}
```

Chapter 4

JavaScript

Introduction

- JavaScript is a popular web-page scripting language, and is supported by almost every browser.
- It adds interactivity to web technology pages.
- JavaScript is usually embedded directly into HTML pages.

Features of JavaScript

- It is lightweight, whose implementations allow client-side script to interact with the user and make dynamic pages.
- It is interpreted programming language with object-oriented capabilities.
- Open and cross-platform (machine independent).
- JavaScript is a case-sensitive language. So the identifiers ***Time and TIME*** will convey different meanings in JavaScript.

So, JavaScript is a ***client-side scripting language*** that runs entirely inside the web browser.

Benefits of JavaScript

Following are some of the benefits that JavaScript language possesses to make the website dynamic.

- It is widely supported in browser.
- It gives easy access to document object and can manipulate most of them.
- It can give interesting animations with many multimedia data types.
- It is secure language because it can detect the visitor's browser.
- It can react to events, read and write HTML elements
- It can be used to validate data

JavaScript Rules

- JavaScript program contains variables, objects and functions.
- Each line is terminated by a semicolon. Blocks of code must be surrounded by curly brackets.
- Functions have parameters which are passed inside parenthesis.
- Variables are declared using the keyword **var**.

- Script does not require ***main function and exit condition***.

Language Format

- JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.
- You can place the <script> tags, containing your JavaScript, anywhere within you web page, but it is normally *recommended that you should keep it within the <head> tags*.
- The <script> tag alerts the browser program to start interpreting all the text between these tags as a script.

So, syntax will look as follows:

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

JavaScript code

</script>

The script tag takes two important attributes:

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

Take a look at the following code.

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

This code will produce the following result: Hello World!

Description:

- The <script type="text/javascript"> tag tells the browser that whatever comes between that tag and the coming </script> tag is **script**.
- The type="text/javascript" tells it that it is JavaScript.

- The **document.write()** is the JavaScript standard for writing text to the browser window.
- The **'document'** clause refers to the HTML webpage (termed a document) as a whole; what follows ('.write()') is a command for the document object to carry out.

JavaScript variables

- JavaScript is used for the manipulation of data. The pieces of data can be stored either in variables or arrays.
- JavaScript Variables are declared using the "**var**" keyword. ***Syntax: var num;***
- JavaScript variables can be declared with initial value. ***Syntax: var num=1;***
- Multiple JavaScript variables can even be declared at the same time.
    - ***Syntax: var num=1; var name= "Bikila";***
- Every JavaScript variables ends with semicolon (;).
- Variables in JavaScript are weakly typed, meaning that the types of individual variables are not determined by the programmer. Unlike many languages, JavaScript only provides a generic "var" rather than separate types for integers, floating point numbers, characters, and strings.
    - Example: var x = 0; → x is a number

Data types and primitives

- There are six data types in JavaScript:

1. Numbers: – Integer or floating point numbers
2. Booleans: – Either true/false or a number (0 being false) can be used for boolean values
3. Strings: – Sequence of characters enclosed in a set of single or double quotes
4. Objects: – Entities that typically represents elements of a HTML page
5. Null: – No value assigned which is different from 0
6. Undefined: – Is a special value assigned to an identifier after it has been declared but before a value has been assigned to it

JavaScript Popup Boxes

- JavaScript has three kinds of popup boxes:

1. **Alert Box**: - is used if you want to make sure information comes through to the user.
    - Syntax: **alert("sometext");**
2. **Confirm Box**: - is used if you want the user to ***verify or accept*** something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

☞ Syntax: **confirm("sometext");**

```
if (confirm("Press a button!")) {
txt = "You pressed OK!";
} else {
txt = "You pressed Cancel!";
}
```

3. **Prompt Box**: - is used if you want the *user to input a value before entering a page*. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

☞ Syntax: **prompt("sometext","defaultvalue");**

```
var person = prompt("Please enter your name", "Nahil");
if (person == null || person == "") {
txt = "User cancelled the prompt.";
} else {
txt = "Hello " + person + "! How are you today?";
}
```

4.9. Operators in JavaScript

1. Arithmetic Operators

Operator	Description	Example	Result
+	Addition	var x = 2; var y = 2; var sum=x + y;	4
-	Subtraction	var=x = 5; var y = 2; var diff=x – y;	3
*	Multiplication	var x = 5; var y = 4; var prod=x * y;	20
/	Division	var x= 15; var y= 5; var quotient=x/y;	3
%	Modulus	var x= 15; var y= 5; var mod=x%y;	0
++	Increment	var x = 5; x++;	6
--	Decrement	var x = 5; x--;	4

2. **Logical Operators**

Operator	Description	Example
&&	And	var x = 6; var y = 3; (x < 10 && y > 1) returns true
||	Or	var x = 6; var y = 3; (x == 5 || y == 5) returns false
!	Not	var x = 6; var y = 3; !(x == y) returns true

3. **Assignment Operators: used to assign value to variable**

Operator	Example	Is The Same As
=	x = y	x = y
+=	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

4. **Comparison Operators**

Operator	Description	Example
==	is equal to	5 == 8 returns false
===	is equal to (checks for both value and type)	var x = 5; var y = "5"; x == y returns true x === y returns false
!=	is not equal	5 != 8 returns true
>	is greater than	5 > 8 returns false
<	is less than	5 < 8 returns true
>=	is greater than or equal to	5 >= 8 returns false
<=	is less than or equal to	5 <= 8 returns true

5. **String Operator: -** is used to concatenate two strings together.

Example: var x = "Hello " + "world!"

Functions in JavaScript

- A function consists of the “function” keyword followed by the name of the function, a set of open and close parentheses enclosing an optional parameter list and a body enclosed in a set of curly braces.

Syntax: function functionName(parameterList) {

```
        // body
    }
```

- Function parameters are separated by commas in the function declaration.
- A function uses the return keyword to return a value from a function.
- JavaScript code found in a function is not executed until the function is called.

Assigns function to event: - many elements of DOM support events. These events are normally the result of some user actions.

Event	Meaning
onload	Occurs when a window or frame has loaded
onunload	Occurs when a document is removed from a window or frame
onclick	The mouse is clicked on an element
ondblclick	The double click event
onmousedown	Mouse down event
onmouseup	Mouse up event
onmouseover	Mouse moves onto an element
onmousemove	Mouse moves over an element
onmouseout	Mouse leaves an element
onfocus	Element receives focus
onblur	Element loses focus
onkeypress	Key press event
onkeydown	Key is pressed down
onkeyup	Key is released
onsubmit	Submit button is pressed
onreset	Form reset event occurs
onselect	Some text in an element is selected
onchange	Element loses focus and its value changes

Example:

```
<html>
<body>
<script type="text/javascript">
function addition(){
var x=7;
var y=10;
var sum=x+y;
```

```
alert(sum);}
</script>
<form>
<input type="button" onclick="addition()" value="show">
</form> </body></html>
```

Objects in JavaScript

- JavaScript is not a pure object oriented programming language, but uses the concept of objects. The ***new*** keyword used here is to create an object, it allocates memory and storage.
- Objects can have functions and variables. To differentiate between global variables and those which are part of an object but may have the same name,
- JavaScript uses this keyword. When referring to a property of an object, whether a method or a variable, a dot is placed between the object name and the property.

Regular Expression:

- A script language may take name data from a user and have to search through the string one character at a time. The usual approach in scripting language is to create a pattern called a regular expression which describes a set of characters that may be present in a string. The regular expression can be shown below.

```
var pattern = new RegExp("target");
var string = "can you find the target";
pattern.test(string);
```

Regular expression is a javascript object. Dynamic patterns are created using the keyword *new*.

regex = new RegExp("feroz | amer");

Example 5: JavaScript code to implement RegExp

```
<html>
<head> <title> registration form</title>
<script type="text/javascript">
function v()
{
var a=document.f.un.value;
var b=document.f.pw.value;
```

```
var r1=new RegExp("^[a-zA-Z]{6,}$");
var r2=new RegExp("^[a-zA-Z0-9]{6,}$");
if(r1.test(a))
{
if(r2.test(b))
{
 window.alert("ur successfully login");
}
else
window.alert("enter password atleast 6 characters ");
}
else
{
window.alert("enter uname atleast 6 characters and should contain only alphabets");
}
}
</script> </head>
<body bgcolor="#5555" text="white"onSubmit="return v()">
<form name="f">
<center>
<table border="0"><br><br><br><br><br><br>
<tr>
<td align="center"> username:</td>
<td align="center"><input type="text" value="" name="un"/></td>
</tr>
<tr>
<td align="center">password:</td>
<td align="center"><input type="password" value="" name="pw"/></td></tr>
<tr> <td align="center" colspan="2"><input type="submit" value="Submit">
<input type="reset" value="reset"/></td></tr>
</table></center></form> </body> </html>
```

Cookies in JavaScript

What are cookies?

- Cookies are data, stored in small text files, on our computer.
- Cookies were invented to remember information about the user.
- JavaScript can create, read, and delete cookies with the **document.cookie** property.
  - With JavaScript, a cookie can be created like: *document.cookie = "username=Ayantu";*
  - We can also add an expiry date (in UTC time). By default, the cookie is deleted when the browser is closed. *document.cookie = "username=Ayantu; expires=Thu, 18 Dec 2019 12:00:00 UTC";*
  - With JavaScript, we can read cookies like:

```
<script>
{
 document.write(document.cookie);
}
</script>
```

  - Deleting a cookie is very simple. Just set the "expires" parameter to a passed date. *document.cookie = "username=Ayantu; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;" ;*
  - **Session:** - A session uniquely identifies and provides continuity for a user's website usage to a better degree than an IP address or cookie could (essentially, by using both together)
  - **Session Cookie:** - A cookie that is deleted when the web browser is closed.

Using JavaScript on HTML forms

- JavaScript makes HTML pages more dynamic and interactive.
- JavaScript can be used to validate HTML forms.
- Validation can be defined by many different methods, and deployed in many different ways.
  - ***Server side validation*** *is performed by a web server, after input has been sent to the server.*
  - ***Client side validation*** *is performed by a web browser, before input is sent to a web server.*
- JavaScript is used for client side form validation.

- If a form field (fieldname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted:

```
function validateForm() {
var x = document.forms["myForm"]["fieldname"].value;
if (x == "") {
alert("Name must be filled out");
return false;
}
}
```

The function can be called when the form is submitted.

```
<form name="myForm" action="/action_page.php" onsubmit="return
validateForm()" method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

What is the DOM?

- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:
- "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

What is the HTML DOM?

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- *The HTML elements as objects*
- *The properties of all HTML elements*
- *The methods to access all HTML elements*
- *The events for all HTML elements*

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

1. JavaScript HTML DOM Window Object

- The window object represents an open window in a browser.

- If a document contain frames (<iframe> tags), the browser creates one window object for the HTML document, and one additional window object for each frame.
- Note: There is no public standard that applies to the Window object, but all major browsers support it.
- Window object ***properties and methods***:
  1. *self :- it returns the current window*
  2. *status:- it sets or returns the text in the status bar of a window*
  3. *top:- it returns the topmost browser window*

2. JavaScript HTML DOM Document:

- The HTML DOM document object is the owner of all other objects in our web page.
- The document object represents our web page. When we want to access any element in an HTML page, we always start with accessing the document object.
- Below are some examples of how we can use the document object to access and manipulate HTML.
  - DOM Document methods for ***finding HTML elements***
    - ✓ *document.getElementById(id):- Find an element by element id.*
    - ✓ *document.getElementsByTagName(name):- Find elements by tag name*
    - ✓ *document.getElementsByClassName(name):- Find elements by class name*
  - DOM Document methods for ***changing HTML elements***
    - ✓ *element.innerHTML = new html content:- Change the inner HTML of an element*
    - ✓ *element.attribute = new value: - Change the attribute value of an HTML element.*
    - ✓ *element.setAttribute(attribute, value):- Change the attribute value of an HTML element*
    - ✓ *element.style.property = new style:- Change the style of an HTML element*
  - DOM Document methods for ***Adding and Deleting Elements***
    - ✓ *document.createElement(element):- Create an HTML element*
    - ✓ *document.removeChild(element):- Remove an HTML element*
    - ✓ *document.appendChild(element):- Add an HTML element*
    - ✓ *document.replaceChild(element):- Replace an HTML element*

- ✓ *document.write(text):- Write into the HTML output stream*

JavaScript HTML DOM Elements

- Used to find and access HTML elements in an HTML page. There are a couple of ways to do this:
  - *Finding HTML elements by id*
  - *Finding HTML elements by tag name*
  - *Finding HTML elements by class name*
  - *Finding HTML elements by CSS selectors*
  - *Finding HTML elements by HTML object collections*
- The easiest way to find an HTML element in the DOM, is by using the element id.
  - example to find the element with id="intro":
    - *var myElement = document.getElementById("intro");*
  - example to find all <p> elements:
    - *var x = document.getElementsByTagName("p");*
- *To find all HTML elements with the same class name, use getElementsByClassName().*
- This example returns a list of all elements with class="intro".
  - *var x = document.getElementsByClassName("intro");*
- Finding elements by class name does not work in Internet Explorer 8 and earlier versions.

Form methods and properties

- Document.forms:- returns an array of HTML FormElement objects listing each of the forms on the page. We can then use any of the following syntaxes to get an individual form:
  - *document.forms[index]:- used to returns the form at the specified index into the array of forms.*
  - *document.forms[id]:- used to returns the form whose ID is id.*
  - *document.forms[name]:- used to returns the form whose name attribute's value is name.*

Chapter 5

Server-side programming

Introduction to server-side scripting

- Server-side scripting is a method of designing websites so that the process or user request is run on the originating server.
- It is a technique used in **web development** which involves employing scripts on a **web server** which produce a response for each user's request to the website.
- It provides an interface to the user and is used to limit access to proprietary data and help keep control of the script source code.
- Examples: Java, JavaScript using Server-side JavaScript (SSJS), Perl, PHP.
- All of the code is executed on the server before the data is passed to the user's browser. In the case of PHP this means that no PHP code ever reaches the user, it is instead executed and only the information it outputs is sent to the web browser.
- Server-side processing is used to interact with permanent storage like databases or files.
- Server-side processing happens when a page is first requested and when pages are posted back to the server.
- Examples of server-side processing are:
  - *User validation*,
  - *Saving and retrieving data*, **and**
  - *Navigating to other pages*.

Differences between Client-side and Server-side Scripting

- The **client** refers to a web browser running on the user's local machine.
- The **server** is the web server software (e.g. Apache) that runs on server hardware.

Client-Side Scripting

- It is used to run scripts which are 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**.

Server-Side Scripting

- It is used to runs a scripting language in 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.
- Its advantage is the ability **to highly customize the response** based on the user's requirements, access rights, or queries into data stores.

What is PHP?

- It is an acronym for "**PHP Hypertext Preprocessor**"
- It is used for creating dynamic and interactive websites.
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension "**.php**"
- It can create, open, read, write, delete, and close files on the server
- It can collect data from form, and it can send and receive cookies
- It can add, delete, modify data in your database and supports a wide range of databases
- It can restrict users to access some pages on your website and it can encrypt data
- It runs on various platforms (Windows, Linux, UNIX, Mac OS X, etc.)
- It is compatible with almost all servers used today (Apache, IIS, etc.)
- It is free. Download it from the official PHP resource: www.php.net
- It is easy to learn and runs efficiently on the server side

To start using PHP, you can:

- Install a web server on your own PC
- Install PHP
- Install a database, such as MySQL
- Or you can have and install service that comprises all things together such as WAMP, XAMPP.

Use Basic Syntax

There are three main differences between a standard **HTML document** and a **PHP document**.

- **First**, PHP scripts should be saved with the .php file extension (for example, index.php). Just as a file's extension on your computer tells the operating system in what application to open the file, a Web page's extension tells the server **how to process the file**.
- **Second**, you place PHP code within <?php and ?> tags, normally within the context of some HTML.

Example

```
<html><body><h1>This is HTML.</h1>
<?php PHP code! ?>
<p>More HTML</p>
</body></html>
```

PHP tags indicate the parts of the page to be run through the PHP processor on the server.

- **Third**, PHP scripts must be run on a **PHP-enabled Web server** (whereas HTML pages can be viewed on any computer, directly in a browser). This means that PHP scripts must always be run through a **URL**.

Note

- A PHP script can be placed anywhere in the document.
- PHP statements are terminated by **semicolon** (;).
- The closing tag of a block of PHP code also automatically implies a semicolon (so you do not have to have a semicolon terminating the last line of a PHP block).

Example

```
<!DOCTYPE html>
<html><body><h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body></html>
```

Sending Data to the Web Browser

1. You'll use PHP most frequently to send information to the browser in the form of **plain text and HTML tags**. To do so, there are two basic ways to get output: **echo and print**.

There are some differences between echo and print:

1. **Echo:** - can output one or more strings
   - ☞ It is a language construct, and used with or without parentheses: echo or echo().
   - ☞ It is faster compared to print as echo **does not return any value**.
2. **Print:** - can only output one string, and returns always 1
   - ☞ It is can be used with or without parentheses: print or print().
   - ☞ Just type the word print, followed by what you want to display: a simple message, the value of a variable, the result of a calculation, and so forth.
   - ☞ To be clear, **print doesn't actually print anything; it just outputs data**. When a PHP script is run through a Web browser, that PHP output is received by the browser itself.

Example using Echo

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

Example using Print

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

Comments in PHP

- Comments are integral to programming, not because they do anything but because they help you remember why you did something.
- The computer ignores these comments when it processes the script. Furthermore, PHP comments are never sent to the Web browser and therefore remain your secret.

Comments are useful for:

- **To let others understand what you are doing** - Comments let other programmers understand what you were doing in each step.
- **To remind yourself what you did** - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code.

PHP supports three ways of commenting:

1. `//` This is a single line comment
2. `#` This is also a single line comment
3. `/*` This is a multiple lines comment block that spans over more than one line `*/`

PHP Case Sensitivity

- In PHP, all user-defined functions, classes, and keywords (e.g. if, else, while, echo, etc.) are NOT case-sensitive.
- However; in PHP, all **variables are case-sensitive**.
- In the example below, only the first statement will display the value of the $color variable (this is because $color, $COLOR, and $coLOR are treated as three different variables):

Example

```
<!DOCTYPE html>
<html><body>
<?php
$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
</body></html
```

Utilize Variables

- A variable is a **container for data**.
- Once data has been stored in a variable that data can be altered, printed to the Web browser, saved to a database, emailed, and so forth.
- Variables in PHP are, by their nature, **flexible**: You can put data into a variable, retrieve that data from it (without affecting the value of the variable), put new data in, and continue this cycle as long as necessary.
- Variables in PHP are **largely temporary**: - they only have a value for the duration of the script's execution on the server. Once the execution passes the final closing PHP tag, those variables cease to exist.

Example

```
<?php
$x=5; $y=6;$z=$x+$y;
echo $z;
?>
```

Note

- A variable can have **a short name** (like x and y) or a more **descriptive name** (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the **$ sign**, followed by the name of the variable
- A variable name must start with a **letter or the underscore character**
- A variable name cannot start with a **number**
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, _)
- Variable names are **case sensitive** ($y and $Y are two different variables)
- PHP has no command for declaring a variable.

A variable is created the moment you first assign a value to it:

```
<?php
$txt="Hello world!";
$x=5;
$y=10.5;
?>
```

After the execution of the statements above, the variable **txt** will hold the value Hello world!, the variable **x** will hold the value 5, and the variable **y** will hold the value 10.5.

Note

- When you assign a text value to a variable, put quotes around the value.
- PHP is a Loosely Typed Language
- PHP automatically converts the variable to the correct data type, depending on its value.
- In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/ used.

PHP has three different variable scopes:

1. local
2. global
3. static

Local and Global Scope

- A variable **declared outside a function** has a GLOBAL SCOPE and can **only be accessed outside a function**.
- A variable **declared within a function** has a LOCAL SCOPE and can **only be accessed within that function**.

The following example tests variables with local and global scope:

Example

```
<?php
$x=5; // global scope
function myTest() {
$y=10; // local scope
echo "<p>Test variables inside the function:</p>";
echo "Variable x is: $x";
echo "<br>";
echo "Variable y is: $y"; }
myTest();
```

```
echo "<p>Test variables outside the function:</p>";
echo "Variable x is: $x";
echo "<br>";
echo "Variable y is: $y";
?>
```

- In the example above there are two variables $x and $y and a function myTest().
- **$x** is a **global variable** since it is declared outside the function and **$y** is **a local variable** since it is created inside the function.
- When we output the values of the two variables **inside** the myTest() function, it prints the value of $y as it is the locally declared, but cannot print the value of $x since it is created outside the function.
- Then, when we output the values of the two variables **outside** the myTest() function, it prints the value of $x, but cannot print the value of $y since it is a local variable and it is created inside the myTest() function.
- You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

PHP the global Keyword

- The global keyword is used to access a global variable from within a function. To do this, use the global keyword before the variables (inside the function):

Example

```
<?php
$x=5;$y=10;
function myTest() {
global $x,$y;
$y=$x+$y; }
myTest();
echo $y; // outputs 15
?>
```

- PHP also stores all global variables in an array called $GLOBALS[index].
- The index holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten like this:

Example

```
<?php
$x=5; $y=10;
function myTest() {
```

```
$GLOBALS['y']=$GLOBALS['x']+$GLOBALS['y']; }
myTest();
echo $y; // outputs 15
?>
```

PHP the static Keyword

- Normally, when a function is completed or executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job. To do this, use the **static** keyword when you first declare the variable:

Example

```
<?php
function myTest() {
static $x=0;
echo $x;
$x++; }
myTest();
myTest();
myTest();
?>
```

- Then, each time the function is called, that variable will still have the information it contained from the last time the function was called. The variable is still local to the function.

Manipulate Strings

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:

Example

```
<?php
$x = "Hello world!";
echo $x;
echo "<br>";
$x = 'Hello world!';
echo $x;
?>
```

Concatenating Strings

- It refers to the appending of one item onto another.
- The period (.) is the operator for performing this action, and it's used like so:

```
$s1 = 'Hello, ';
$s2 = 'world!';
```

```
$greeting = $s1 . $s2;
```

- The end result of this concatenation is that the $greeting variable has a value of Hello, world!. Because of the way PHP deals with variables, the same effect could be accomplished using $***greeting = "$s1$s2";***.
- This code works because PHP replaces variables within double quotation marks with their value. However, the formal method of using the period to concatenate strings is more commonly used and is recommended (it will be more obvious what's occurring in your code).

Adjusting String Case

A handful of PHP functions are used to change the case of a string's letters:

- . ucfirst() capitalizes the first letter of the string.
- . ucwords() capitalizes the first letter of words in a string.
- . strtoupper() makes an entire string uppercase.
- . strtolower() makes an entire string lowercase.

Manipulate Numbers

- An integer is a number without decimals.

Rules for integers:

- An integer must have at least one digit (0-9)
- An integer cannot contain comma or blanks
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based – prefixed with 0x) or octal (8-based - prefixed with 0)

In the following example we will test different numbers. The PHP var_dump() function returns the data type and value of variables:

Example

```
<?php
$x = 5985;
var_dump($x);
echo "<br>";
$x = -345; // negative number
var_dump($x);
echo "<br>";
```

```
$x = 0x8C; // hexadecimal number
var_dump($x);
echo "<br>";
$x = 047; // octal number
var_dump($x);
?>
```

PHP Floating Point Numbers

- A floating point number is a number with a decimal point or a number in exponential form.
- In the following example we will test different numbers. The PHP var_dump() function returns the data type and value of variables:

Example

```
<?php
$x = 10.365;
var_dump($x);
echo "<br>";
$x = 2.4e3;
var_dump($x);
echo "<br>";
$x = 8E-5;
var_dump($x);
?>
```

PHP Booleans

- Booleans can be either TRUE or FALSE. Booleans are often used in conditional testing.

```
$x=true;
$y=false;
```

Operators in PHP

1. Arithmetic Operators

There are following arithmetic operators supported by PHP language.

- *Addition (+), Subtraction (-), Multiplication (*), Division (/)*

Assume variable **A** holds **10** and variable **B** holds **20** then:

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A –B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2

%	Modulus operator and remainder of after an integer division	B% A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decrease integer value by one	A—will give 9

2. Comparison Operators

There are following comparison operators supported by PHP language. Assume variable A holds 10 and variable B holds 20 the**n**

Operator	Description	Example
==	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(A! = B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

3. Logical Operators

There are following logical operators supported by PHP language. Assume variable A holds 10 and variable B holds 20 then

Operator	Description	Example
And	Called Logical AND operator. If both the operands are true then condition becomes true	(A and B) is true.
Or	Called Logical OR Operator. If any of the two operands are non-zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non-zero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are	(A || B) is true.

	non-zero then condition becomes true.	
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!!(A && B) is false.

4. Assignment Operators

There are following assignment operators supported by PHP language:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

5. Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax:

Operator	Description	Example
?:	Conditional Expression	If condition is true? Then value x: otherwise value y

All the operators we have discussed above can be categorized into following categories

1. **Unary prefix operators: -** which precede a single operand.

2. **Binary operators: -** which take two operands and perform a variety of arithmetic and logical operations.
3. **The conditional operator** (a ternary operator): - which takes three operands and evaluates either the second or third expression, depending on the evaluation of the first expression.
4. **Assignment operators: -** which assign a value to a variable.

What are constants?

- Constants are like variables except that once they are defined they **cannot be changed** or **undefined**.
- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. A constant is **case-sensitive by default**. By convention, constant identifiers are **always uppercase**.

Note: Unlike variables, constants are automatically **global** across the entire script.

Set a PHP Constant

- To set a constant, use the **define()** function - it takes three parameters:
  - The first parameter defines the **name of the constant**,
  - The second parameter defines the **value of the constant**, and
  - The optional third parameter specifies **whether the constant name should be case-insensitive**. **Default is false**.

The example below creates a **case-sensitive constant**, with the value of "Welcome to Ambo University!":

Example

```
<?php
define("GREETING", "Welcome to Ambo University!");
echo GREETING;
?>
```

The example below creates a **case-insensitive constant**, with the value of "Welcome to Ambo

University!":

Example

```
<?php
```

```
define("GREETING", "Welcome to Ambo University!", true);
echo greeting;
?>
```

- To define a constant you have to use **define**() function and to retrieve the value of a constant, you have to simply specifying its name.
- You can also use the function **constant**() to read a constant's value if you wish to obtain the constant's name dynamically.

constant() function

- As indicated by the name, this function will return the value of the constant.
- This is useful when you want to retrieve value of a constant, but you do not know its name, i.e. it is stored in a variable or returned by a function.

constant() example

```
<?php
define("MINSIZE", 50);
echo MINSIZE;
echo constant("MINSIZE"); // same thing as the previous line
?>
```

- Only scalar data (boolean, integer, float and string) can be contained in constants.

Differences between constants and variables are

- There is no need to write a dollar sign ($) before a constant, where as in Variable one has to write a dollar sign.
- Constants cannot be defined by simple assignment, they may only be defined using the **define()** function.
- Constants may be defined and accessed anywhere ***without regard to variable scoping rules***.
- Once the Constants have been set, may not be redefined or undefined.

Valid and invalid constant names

```
// Valid constant names
define("ONE", "first thing");
define("TWO2", "second thing");
define("THREE_3", "third thing");
// Invalid constant names
define("2TWO", "second thing");
define("__THREE__", "third value");
```