UNIT 1
HTML AND COMMON TAGS

INTRODUCTION :-

HTML stands for Hyper Text Markup Language, developed by scientist Tim Berners-Lee in 1990. HTML is a language for describing the structure of web-page (documents) using ordinary text for retrieval across the Internet using browser programs of the World Wide Web(WWW). HTML is not a complex programming language but it is an application of the Standard Generalized Markup Language (SGML) which is the International Standard (ISO 8879) for text markup. Every web page is actually a HTML file. Each HTML file is just a plain-text file, but with a "**.html"** file extension instead of "**.txt"**, and is made up of many HTML tags as well as the content for a web page. The backbone of the World Wide Web is made of HTML files, which are specially formatted documents that can contain links, as well as images and other media. All web browsers can read HTML files. In addition to HTML, it's common for websites to use technologies like CSS (Cascading Style Sheets) and JavaScript to do more advanced things.

WORLD WIDE WEB (WWW) :-

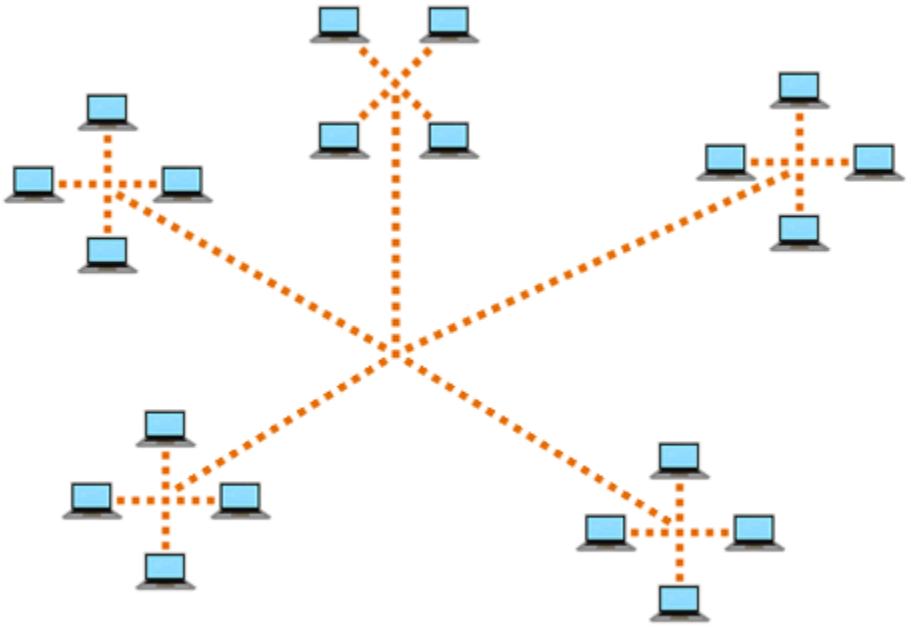
The World Wide Web (abbreviated as WWW or W3 commonly known as the Web) is a system of interlinked hypertext documents that are accessed via the Internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them via hyperlinks. The World Wide Web is a virtual network of websites connected by hyperlinks (or "links"). Websites are stored on servers on the Internet, so the World Wide Web is a part of the Internet.

INTERNET :-

Internet is a network of networks. The Internet is the largest computer network in the world, connecting millions of computers. A network is a group of two or more computer systems linked together. There are two main types of computer networks:

Local Area Network (LAN): A LAN is two or more connected computers sharing certain resources in a relatively small geographic location, often in the same building. Examples include home networks and office networks.

Wide Area Network (WAN): A WAN typically consists of two or more LANs. The computers are farther apart and are linked by telephone lines, dedicated telephone lines, or radio waves. The Internet is the largest Wide Area Network (WAN) in existence.



The Internet is used to send information quickly between computers around the world. It has millions of smaller domestic, academic, business, and government networks and websites, which together carry many different kinds of information (facts and details) and services. In the early days, most people just used the Internet to search for information. Today's Internet is a constantly evolving tool that not only contains an

amazing variety of information but that also provides new ways of accessing, interacting, and connecting with people and content.

URL :-

A URL stands for Uniform Resource Locator (previously Universal Resource Locator) is the unique address for a file that is accessible on the Internet. The URL, also known as the web address, tells the browser exactly where to find the page. A common way to get to a Web site is to enter the URL of its home page file in your Web browser's address line. However, any file within that Web site can also be specified with a URL. Such a file might be any Web (HTML) page other than the home page, an image file, or a program such as a common gateway interface application or Java applet. The URL contains the name of the protocol to be used to access the file resource, a domain name that identifies a specific computer on the Internet, and a pathname, a hierarchical description that specifies the location of a file in that computer.

Format of a URL :-

scheme://host[:port]/path/filename{#location|?indexterm}

Examples :-

ftp://www.urlofftpexampl.in/pub/sgml/ftpexample.ps
http://www.urlofhttpexampl.in/cgi-bin/acronym?url
gopher://ds.urlofgopherexampl.net/

where :-

scheme -	is one of http, ftp, gopher, wais or file, depending on the kind of service you want to access
[square brackets] -	mean the :port is optional (the default port is the one for the scheme being used, so leave it out unless told otherwise)
host -	is the Internet hostname of the machine where the server runs which provides the resource you are looking for, eg www.urlofftpexampl.in
path -	is the directory path to the resource
filename.type -	is the filename (including the filetype, if needed)
location -	is the identity of a known location in a .html file being retrieved (eg you know it has already been marked in the file by the author

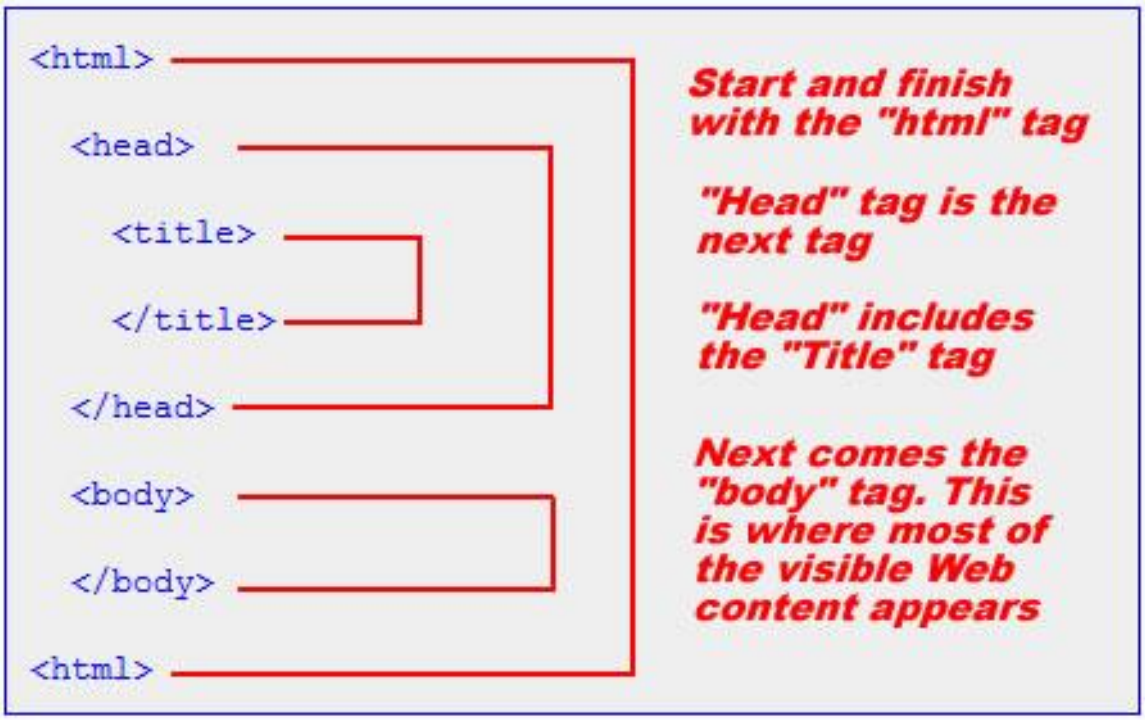
using the nameattribute to an anchor element

indexterm - is a search term or argument to pass to the resource.

COMMON TAGS :-

HTML tags are the hidden keywords surrounded by angle brackets within a web page that define how the browser must format and display the content. Most tags must have two parts, an opening and a closing part. For example, <tagname> is the opening tag and </tagname> is the closing tag. Note that the closing tag has the same text as the opening tag, but has an additional forward-slash (/) character, interpret this as the "end" or "close" character.

There are some tags that are an exception to this rule, and where a closing tag is not required. The <img>tag for showing images is one example of this. Each HTML file must have the essential tags for it to be valid, so that web browsers can understand it and display it correctly. The rest of the HTML file can contain as little or as many tags as you want to display your content. Following figure shows the structure of HTML document with minimum required tags.



HTML Section Tags :-

To organize the various parts of the HTML document that describes a Web page, you use a set of *section* tags. The types of HTML tags in each section of the Web page definition have a specific purpose:

- **<html>...</html>**

The <html> tag must be the VERY FIRST tag that appears in any HTML file. <html> tells the Web browser that this is the beginning of an HTML document and the </html> tag tells the browser that the HTML file is finished. The tags with slash "/" respresent a closing tag.

- **<head> ... </head>**

The <head> and </head> tags defines that part of the document containing information about the page. This section includes the TITLE of the document.

- **<title> ... </title>**

This tag surrounds the TITLE of your document. Note: The TITLE appears at the top of your browser window.

- **<body> ... </body>**

The <body> tag usually comes immediately after the </head> tag that ended your HEAD section. The </body> tag comes before the </html> tag at the end of the document.

HTML Container Tags :-

Most of the tags require an opening and a closing command. These are called container tags. The reason for the name is simple - the text or object 'contained' within the opening and closing command is acted upon by the function of the command, such as the following:

<B>this text is bold</B>

Tag Syntax

Think of tags as having the following structure:

<COMMAND property="value">"contained object</COMMAND>.

If there are no properties to define, then you only require <COMMAND>object</COMMAND>. There are also occasions where you may want to define several properties using the one command:

<BODY background="tiles.gif" bgcolor="black">

HTML Empty Tags :-

There are, however, exceptions to every rule. Some tags, such as
, do not follow this format because there is nothing to contain - the line break(
) is the object. Two more examples are<HR> and . <HR> inserts a horizontal rule on your page, and   inserts a space. Pressing the spacebar several times in your code will NOT create several spaces in your Web page. There is always a default of one space on either side of a character. Therefore, if you wanted three spaces between two words, you would code it like this:

two   words

There is one space (by default) after the word two, then the   character, followed by another default space. Or, you could code it like this:

two   words

In this instance, all spaces are replaced by code but the result is the same.

Attributes of HTML Tags :-

Most of the HTML tags can also have attributes, which are extra bits of information. An attribute is used to define the characteristics of an HTML element and is placed inside the element's opening tag. All attributes are made up of two parts:

a name and a value:

The name is the property you want to set. For example, the paragraph <p> element in the example carries an attribute whose name is align, which you can use to indicate the alignment of paragraph on the page.

The value is what you want the value of the property to be set and always put within quotations. The below example shows three possible values of align attribute: left, center and right.

Attribute names and attribute values are case-insensitive. However, the World Wide Web Consortium (W3C) recommends lowercase attributes/attribute values in their HTML 4 recommendation.

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>Align Attribute  Example</title>
    </head>
    <body>
        <p align="left">This is left aligned</p>
        <p align="center">This is center aligned</p>
        <p align="right">This is right aligned</p>
    </body>
</html>
```

This will display following result:

This is left aligned

This is center aligned

This is right aligned

Core Attributes :-

The four core attributes that can be used on the majority of HTML elements (although not all) are:

- **id**
- **title**
- **class**
- **style**

The id Attribute:-

The id attribute of an HTML tag can be used to uniquely identify any element within an HTML page. There are two primary reasons that you might want to use an id attribute on an element:

1. If an element carries an id attribute as a unique identifier it is possible to identify just that element and its content.
2. If you have two elements of the same name within a Web page (or style sheet), you can use the id attribute to distinguish between elements that have the same name.

Let's use the id attribute to distinguish between two paragraph elements as shown below.

Example:-

```
<p id="html">This para explains what is HTML</p>
<p id="css">This para explains what is Cascading Style Sheet</p>
```

The title Attribute:-

The title attribute gives a suggested title for the element. They syntax for the title attribute is similar as explained for id attribute:

The behavior of this attribute will depend upon the element that carries it, although it is often displayed as a tooltip when cursor comes over the element or while the element is loading.

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>The title Attribute Example</title>
    </head>
    <body>
        <h3 title="Hello HTML!">Titled Heading Tag Example</h3>
    </body>
</html>
```

This will produce following result:

Titled Heading Tag Example

Now try to bring your cursor over "Titled Heading Tag Example" and you will see that whatever title you used in your code is coming out as a tooltip of the cursor.

The class Attribute :-

The class attribute is used to associate an element with a style sheet, and specifies the class of element. The value of the attribute may also be a space-separated list of class names.

Example :-

```
class="className1 className2 className3"
```

The style Attribute :-

The style attribute allows you to specify Casecading Style Sheet (CSS) rules within the element.

```
<!DOCTYPE html>
<html>
    <head>
        <title>The style Attribute</title>
    </head>
    <body>
        <p style="font-family:arial; color:#FF0000;">Some text...</p>
    </body>
</html>
```

This will produce following result:

Some text...

TEXT FORMATTING TAGS :-

The following HTML tags are used to format the appearance of the text on your web page. This can jazz up the look of the web page, *however*, too much variety in the text formatting can also look displeasing.

Header - <h?> </h?>

There are 6 levels of headings available, from h1 for the largest and most important heading, down to h6 for the smallest heading. (? – number from 1 to 6)

Bold -

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

Italic -

Also working the same way as a word processor, italics displays the text at a slight angle.

Underline - <u> </u>

Again, the same as underline in a word processor. Note that html links are already underlined and don't need the extra tag.

Strike-out - <strike> </strike>

Puts a line right through the centre of the text, crossing it out. Often used to show that text is old and no longer relevant. Also works by using <s> </s> instead.

Preformatted Text - <pre> </pre>

Any text between the pre tags, including spaces, carriage returns and punctuation, will appear in the browser as it would in a text editor (normally browsers ignore multiple spaces)

Source Code - <code> </code>

Similar to tt the text is displayed in a fixed-width font, and is commonly used to show source code.

Typewriter Text - <tt> </tt>

The text appears to have been typed by a typewriter, in a fixed-width font. For example: This text is written using the <tt></tt> tags.

Block Quote - <blockquote> </blockquote>

Defines a long quotation, and the quote is displayed with an extra wide margin on the left hand side of the block quote.

Small - <small> </small>

Instead of having to set a font size, you can use the small tag to render text slightly smaller than the text around it. Useful for displaying the 'fine-print'.

Font Colour -

Change the colour of a few words or a section of text. The 6 question marks represent the hex color code,

Font Size -

Replace the ? with a number from 1 to 7 to change the size of the font. One being the smallest and seven the largest.

Font Size Change -

For an immediate change of font size with respect to the font size preceding it, this tag increase or decreases the size of the font by the number you specify. Eg: Some Text

Change Font Face -

To show text in a particular font, use the font name such "Helvetica" or "Arial" or "Courier". Be aware that using some fancy font from your computer means that the person viewing that page must also have that font installed on their computer too, otherwise it will look totally different to them.

Centre -

A useful tag, as it says, it makes everything in between the tags centred (in the middle of the page).

Emphasis -

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

Strong Emphasis -

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

LINE AND PARAGRAPH TAGS :-

Authors traditionally divide their thoughts and arguments into sequences of paragraphs. The organization of information into paragraphs is not affected by how the paragraphs are presented: paragraphs that are double-justified contain the same thoughts as those that are left-justified.

The HTML markup for defining a paragraph is straightforward: the P element defines a paragraph. The visual presentation of paragraphs is not so simple. A number of issues, both stylistic and technical, must be addressed:

- Treatment of white space
- Line breaking and word wrapping
- Justification
- Hyphenation
- Written language conventions and text directionality
- Formatting of paragraphs with respect to surrounding content

Paragraphs: the "P" element :-

The HTML <p> element defines a paragraph.

<!ELEMENT P - O (%inline;) -- paragraph -->*

<!ATTLIST P

%attrs; *-- %coreattrs, %i18n, %events -->*

Start tag: **required**, End tag: **optional**

Attributes defined elsewhere

- id, class (document-wide identifiers)
- lang (language information), dir (text direction)
- title (element title)
- style (inline style information)
- align (alignment)
- onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, on mouseout, onkeypress, onkeydown, onkeyup(intrinsic events)

The P element represents a paragraph. It cannot contain block-level elements (including P itself).

Example :-

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

Browsers automatically add an empty line before and after a paragraph. You cannot be sure how HTML will be displayed. Large or small screens, and resized

windows will create different results. With HTML, you cannot change the output by adding extra spaces or extra lines in your HTML code. The browser will remove extra spaces and extra lines when the page is displayed. Any number of spaces, and any number of new lines, counts as only one space.

Example:-

```
<p>
        This paragraph
        contains a lot of lines
        in the source code,
        but the browser
        ignores it.
</p>

<p>
        This paragraph
        contains        a lot of spaces
        in the source        code,
        but the        browser
        ignores it.
</p>
```

Controlling line breaks :-

The HTML
 element defines a line break. Use
 if you want a line break (a new line) without starting a new paragraph. The
 element is an empty HTML element. It has no end tag.

A line break is defined to be a carriage return (
), a line feed (
), or a carriage return/line feed pair. All line breaks constitute white space. For more information about SGML's specification of line breaks.

Forcing a line break: the BR element

<!ELEMENT BR - O EMPTY -- forced line break -->

<!ATTLIST BR

%coreattrs; -- id, class, style, title -->

Start tag: **required**, End tag: **forbidden**

Attributes defined elsewhere

- id, class (document-wide identifiers)
- title (element title)
- style (inline style information)
- clear (alignment and floating objects)

The BR element forcibly breaks (ends) the current line of text.

Example :-

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

For visual user, the clear attribute can be used to determine whether markup following the BR element flows around images and other objects floated to the left or right margin, or whether it starts after the bottom of such objects. Further details are given in the section on alignment and floating objects. Authors are advised to use style sheets to control text flow around floating images and other objects.

With respect to bidirectional formatting, the BR element should behave the same way the [ISO10646] LINE SEPARATOR character behaves in the bidirectional algorithm.

Prohibiting a line break:-

Sometimes authors may want to prevent a line break from occurring between two words. The   entity (or) acts as a space where user should not cause a line break.

Hyphenation :-

In HTML, there are two types of hyphens: the plain hyphen and the soft hyphen. The plain hyphen should be interpreted by a user agent as just another character. The soft hyphen tells the user agent where a line break can occur.

Those browsers that interpret soft hyphens must observe the following semantics: If a line is broken at a soft hyphen, a hyphen character must be displayed at the end of the first line. If a line is not broken at a soft hyphen, the user agent must not display a hyphen

character. For operations such as searching and sorting, the soft hyphen should always be ignored.

In HTML, the plain hyphen is represented by the "-" character (- or -). The soft hyphen is represented by the character entity reference (or)

Preformatted text: The PRE element

<!ENTITY %pre.exclusion "IMG|OBJECT|BIG|SMALL|SUB|SUP">

<!ELEMENT PRE - - (%inline;)* -(%pre.exclusion;) -- preformatted text -->

<!ATTLIST PRE

%attrs; -- %coreattrs, %i18n, %events --

>

Start tag: required, End tag: required

Attribute definitions

width = number [CN]

Deprecated. This attribute provides a hint to visual user agents about the desired width of the formatted block. The user agent can use this information to select an appropriate font size or to indent the content appropriately. The desired width is expressed in number of characters. This attribute is not widely supported currently.

Attributes defined elsewhere

id, class (document-wide identifiers)

lang (language information), dir (text direction)

title (element title)

style (inline style information)

onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup(intrinsic events)

The PRE element tells visual user that the enclosed text is "preformatted". When handling preformatted text, visual user:

- May leave white space intact.
- May render text with a fixed-pitch font.

- May disable automatic word wrap.
- Must not disable bidirectional processing.
- Non-visual user are not required to respect extra white space in the content of a PRE element.

The DTD fragment above indicates which elements may not appear within a PRE declaration. This is the same as in HTML 3.2, and is intended to preserve constant line spacing and column alignment for text rendered in a fixed pitch font. Authors are discouraged from altering this behavior through style sheets.

The following example shows a preformatted verse:

<PRE>

Higher still and higher

From the earth thou springest

Like a cloud of fire;

The blue deep thou wingest,

And singing still dost soar, and soaring ever singest.

</PRE>

Here is how this is typically rendered:-

Higher still and higher

From the earth thou springiest

Like a cloud of fire;

The blue deep thou winglets,

And singing still does soar, and soaring ever signets.

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:

Example :-

<!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>
```

This will produce following result:

This is paragraph one and should be on top

This is paragraph two and should be at bottom

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, while if you miss the forward slash character and just use <hr> it is not valid in XHTML.

HTML LISTS :-

HTML offers web authors three ways for specifying lists of information. All lists must contain one or more list elements. Lists may contain:

- - An unordered list. This will list items using plain bullets.
- - An ordered list. This will use different schemes of numbers to list your items.
- <dl> - A definition list. This arranges your items in the same way as they are arranged in a dictionary.

HTML UNORDERED LISTS :-

An unordered list is a collection of related items that have no special order or sequence. This list is created by using HTML <ul> tag. Each item in the list is marked with a bullet.

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Unordered List</title>
    </head>
    <body>
        <ul>
            <li>Beetroot</li>
            <li>Ginger</li>
            <li>Potato</li>
            <li>Radish</li>
        </ul>
    </body>
</html>
```

This will produce following result:

- Beetroot
- Ginger
- Potato
- Radish

The type Attribute :-

You can use type attribute for <ul> tag to specify the type of bullet you like. By default it is a disc. Following are the possible options:

```
<ul type="square">
<ul type="disc">
<ul type="circle">
```

Example :-

Following is an example where we used <ul type="square">

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Unordered List</title>
    </head>
    <body>
        <ul type="square">
            <li>Beetroot</li>
            <li>Ginger</li>
            <li>Potato</li>
            <li>Radish</li>
        </ul>
    </body>
</html>
```

This will produce following result:

- Beetroot
- Ginger
- Potato
- Radish

HTML ORDERED LISTS :-

If you are required to put your items in a numbered list instead of bulleted list then HTML ordered list will be used. This list is created by using <ol> tag. The numbering starts at one and is incremented by one for each successive ordered list element tagged with <li>.

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Ordered List</title>
    </head>
```

```
<body>
    <ol>
        <li>Beetroot</li>
        <li>Ginger</li>
        <li>Potato</li>
        <li>Radish</li>
    </ol>
</body>
</html>
```

This will produce following result:

1. Beetroot
2. Ginger
3. Potato
4. Radish

The type Attribute :-

You can use type attribute for <ol> tag to specify the type of numbering you like. By default it is a number. Following are the possible options:

<ol type="1"> - Default-Case Numerals.

<ol type="I"> - Upper-Case Roman Numerals.

<ol type="i"> - Lower-Case Roman Numerals.

<ol type="a"> - Lower-Case Letters.

<ol type="A"> - Upper-Case Letters.

Example :-

Following is an example where we used <ol type="I">

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Ordered List</title>
    </head>
    <body>
```

```
        <ol type="I">
            <li>Beetroot</li>
            <li>Ginger</li>
            <li>Potato</li>
            <li>Radish</li>
        </ol>
    </body>
</html>
```

This will produce following result:

I. Beetroot
II. Ginger
III. Potato
IV. Radish

HTML DEFINITION LISTS :-

HTML and XHTML support a list style which is called definition lists where entries are listed like in a dictionary or encyclopedia. The definition list is the ideal way to present a glossary, list of terms, or other name/value list. Definition List makes use of following three tags :-

- <dl> - Defines the start of the list
- <dt> - A term
- <dd> - Term definition
- </dl> - Defines the end of the list

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Definition List</title>
    </head>
    <body>
        <dl>
            <dt><b>HTML</b></dt>
```

```
            <dd>This stands for Hyper Text Markup Language</dd>
            <dt><b>HTTP</b></dt>
            <dd>This stands for Hyper Text Transfer Protocol</dd>
        </dl>
    </body>
</html>
```

This will produce following result:

HTML

This stands for Hyper Text Markup Language

HTTP

This stands for Hyper Text Transfer Protocol

HTML Text Links - Hyperlink :-

A webpage can contain various links that take you directly to other pages and even specific parts of a given page. These links are known as hyperlinks. Hyperlinks allow visitors to navigate between Web sites by clicking on words, phrases, and images. Thus you can create hyperlinks using text or images available on a webpage.

A link is specified using HTML tag <a>. This tag is called **anchor tag** and anything between the opening <a> tag and the closing tag becomes part of the link and a user can click that part to reach to the linked document. Following is the simple syntax to use <a> tag.

```
<a href="Document URL" ... attributes-list>Link Text</a>
```

Example

Let's try following example which links http://www.google.com at your page:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Hyperlink Example</title>
    </head>
    <body>
        <p>Click following link</p>
```

```
        <a href="http://www.google.com" target="_self">Google Search</a>
    </body>
</html>
```

This will produce following result, where you can click on the link generated "Google Search" to reach to the home page of "www.google.com".

Creating Links :-

When you're creating links to documents and images on the web, you need to think about how you're going to link to them. There are two standard ways to create links:

- Absolute paths
- Relative paths

ABSOLUTE PATH URLS :-

Absolute paths are called that because they refer to the very specific location, including the domain name. The absolute path to a web element is also often referred to as the URL. For example, the absolute path to Rediff web page is:

http://www.rediff.com/

You typically use the absolute path with the domain to point to Web elements that are on another domain than your own.

If you're referring to a web element that is on the same domain that you're on, you don't need to use the domain name in the path of your link. Simply leave off the domain, but be sure to include the first slash (/) after the domain name.

It is a good idea to use absolute paths, without the domain name, on most websites. This format insures that the link or image will be usable no matter where you place the page. This may seem like a silly reason to use longer links, but if you share code across multiple pages and directories on your site, using absolute paths will speed up your maintenance.

Absolute URLs tie your code to the protocol and domain. This can be overcome with dynamic URLs.

https://dev.example.com/a.html?q=

ABSOLUTE PROS:-

Control - The subdomain and protocol can be controlled. People that enter through an obscure subdomain will be funneled into the proper subdomain. You can hop back and forth between secure and non-secure as appropriate.

Configurable - Developers love things to be absolute. You can design neat algorithms when using absolute URLs. URLs can be made configurable so that a URL can be updated site-wide with a single change in a single configuration file.

Clairvoyance - You can search for the people scraping your site or maybe pick up some extra external links.

RELATIVE PATH URLS :-

Relative paths change depending upon the page the links are on. There are several rules to creating a link using the relative path:

- links in the same directory as the current page have no path information listed
  filename
- sub-directories are listed without any preceding slashes
  weekly/filename
- links up one directory are listed as
  ../filename

Root Relative URLs tie your code to the base url. This can be overcome with dynamic URLs and/or base tags.

.example.com/index.php?q=

Root Relative Pros:

Configurable - The base tag makes them relative to any root you choose making switching domains and implementing templates easy.

Relative URLs tie your code to the directory structure. There is no way to overcome this. Relative URLs are only useful in file systems for traversing directories or as a shortcut for a menial task.

index.php?q=

<link src="../../../../css/default.css" />

RELATIVE CONS :-

Confusing - How many dots is that? how many folders is that? Where is the file? Why isn't it working?

Maintenance - If a file is accidentally moved resources quit loading, links send the user to the wrong pages, form data might be sent to the incorrect page. If a file NEEDS to be moved all the resources that are going to quit loading and all the links that are going to be incorrect need to be updated.

Does not scale - When webpages become more complex and views start getting reused across multiple pages the relative links will be relative to the file that they were included into. If you have a navigation snippet of HTML that is going to be on every page then relative will be relative to a lot of different places. The first thing people realize when they start creating a template is that they need a way to manage the URLs.

Computed - They are implemented by your browser (hopefully according to RFC).

TABLE TAG :-

Across the worldwide web, HTML tables are used and abused to layout pages. The correct use for tables is to do exactly what you would expect a table to do - to layout tabular data. There are a number of tags used in tables as shown in the following example.

```
<table>
    <tr>
        <td>Row 1, cell 1</td>
        <td>Row 1, cell 2</td>
        <td>Row 1, cell 3</td>
    </tr>
    <tr>
```

```
            <td>Row 2, cell 1</td>
            <td>Row 2, cell 2</td>
            <td>Row 2, cell 3</td>
        </tr>
        <tr>
            <td>Row 3, cell 1</td>
            <td>Row 3, cell 2</td>
            <td>Row 3, cell 3</td>
        </tr>
        <tr>
            <td>Row 4, cell 1</td>
            <td>Row 4, cell 2</td>
            <td>Row 4, cell 3</td>
        </tr>
    </table>
```

The **table** element defines the table.

The **tr** element defines a table row.

The **td** element defines a data cell. These must be enclosed in tr tags, as shown above.

The above code wil generate the following result:-

Row 1, cell 1	Row 1, cell 2	Row 1, cell 3
Row 2, cell 1	Row 2, cell 2	Row 2, cell 3
Row 3, cell 1	Row 3, cell 2	Row 3, cell 3
Row 4, cell 1	Row 4, cell 2	Row 4, cell 3

If you imagine a 3x4 table, which is 12 cells, there should be four tr elements to define the rows and three td elements within each of the rows, making a total of 12 td elements.

CELLPADDING AND CELLSPACING ATTRIBUTES :-

There are two attribiutes called cellpadding and cellspacing which you will use to adjust the white space in your table cells. The cellspacing attribute defines the width of the border, while cellpadding represents the distance between cell borders and the content within a cell.

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Table Cellpadding</title>
    </head>
    <body>
        <table border="1" cellpadding="5" cellspacing="5">
            <tr>
                <th>Name</th>
                <th>Salary</th>
            </tr>
            <tr>
                <td>Ramesh Raman</td>
                <td>5000</td>
            </tr>
            <tr>
                <td>Shabbir Hussein</td>
                <td>7000</td>
            </tr>
        </table>
    </body>
</html>
```

This will produce following result: -

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

COLSPAN AND ROWSPAN ATTRIBUTES :-

You will use colspan attribute if you want to merge two or more columns into a single column. Similar way you will use rowspan if you want to merge two or more rows.

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Table Colspan/Rowspan</title>
    </head>
    <body>
        <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>
    </body>
```

</html>

This will produce following result:-

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		

TABLES BACKGROUNDS :-

You can set table background using one of the following two ways:

- **bgcolor** attribute - You can set background color for whole table or just for one cell.
- **background** attribute - You can set background image for whole table or just for one cell.
- You can also set border color also using **bordercolor** attribute.

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Table Background</title>
    </head>
    <body>
        <table border="1" bordercolor="green" bgcolor="yellow">
            <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>
        </body>
</html>
```

This will produce following result:

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		

Here is an example of using background attribute. Here we will use an image available in /images directory.

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Table Background</title>
    </head>
    <body>
        <table border="1" bordercolor="green" background="/images/test.png">
            <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>
    </body>
</html>
```

Here background image did not apply to table's header.

TABLE HEIGHT AND WIDTH :-

You can set a table width and height using width and height attributes. You can specify table width or height in terms of pixels or in terms of percentage of available screen area.

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Table Width/Height</title>
    </head>
    <body>
        <table border="1" width="400" height="150">
            <tr>
                <td>Row 1, Column 1</td>
                <td>Row 1, Column 2</td>
            </tr>
            <tr>
                <td>Row 2, Column 1</td>
                <td>Row 2, Column 2</td>
            </tr>
        </table>
    </body>
</html>
```

This will produce following result:

Row 1, Column 1	Row 1, Column 2
Row 2, Column 1	Row 2, Column 2

TABLE CAPTION :-

The caption tag will serve as a title or explanation for the table and it shows up at the top of the table. This tag is deprecated in newer version of HTML/XHTML.

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Table Caption</title>
    </head>
    <body>
        <table border="1" width="100%">
            <caption>This is the caption</caption>
            <tr>
                <td>row 1, column 1</td><td>row 1, columnn 2</td>
            </tr>
            <tr>
                <td>row 2, column 1</td><td>row 2, columnn 2</td>
            </tr>
        </table>
    </body>
</html>
```

This will produce following result:

This is the caption

row 1, column 1	row 1, columnn 2
row 2, column 1	row 2, columnn 2

TABLE HEADER, BODY, AND FOOTER :-

Tables can be divided into three portions: a header, a body, and a footer. The head and foot are rather similar to headers and footers in a word-processed document that remain the same for every page, while the body is the main content holder of the table. The three elements for separating the head, body, and foot of a table are:

- **<thead>** - to create a separate table header.
- **<tbody>** - to indicate the main body of the table.
- **<tfoot>** - to create a separate table footer.

A table may contain several **<tbody>** elements to indicate different pages or groups of data. But it is notable that **<thead>** and **<tfoot>** tags should appear before **<tbody>.**

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Table</title>
    </head>
    <body>
        <table border="1" width="100%">
        <thead>
            <tr>
                <td colspan="4">This is the head of the table</td>
            </tr>
        </thead>
            <tfoot>
                <tr>
                    <td colspan="4">This is the foot of the table</td>
                </tr>
```

```
            </tfoot>
            <tbody>
                <tr>
                    <td>Cell 1</td>
                    <td>Cell 2</td>
                    <td>Cell 3</td>
                    <td>Cell 4</td>
                </tr>
            </tbody>
        </table>
        </body>
</html>
```

This will produce following result:

This is the head of the table			
Cell 1	Cell 2	Cell 3	Cell 4
This is the foot of the table			

NESTED TABLES :-

You can use one table inside another table. Not only tables you can use almost all the tags inside table data tag <td>.

Example :-

Following is the example of using another table and other tags inside a table cell.

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Table</title>
    </head>
    <body>
        <table border="1" width="100%">
            <tr>
                <td>
```

```
<table border="1" width="100%">
    <tr>
        <th>Name</th>
        <th>Salary</th>
    </tr>
    <tr>
        <td>Ramesh Raman</td>
        <td>5000</td>
    </tr>
    <tr>
        <td>Shabbir Hussein</td>
        <td>7000</td>
    </tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

This will produce following result:

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

HTML IMAGES :-

Images are very important to beautify as well as to depict many complex concepts in simple way on your web page. This will take you through simple steps to use images in your web pages.

Insert Image :-

You can insert any image in your web page by using <img> tag. Following is the simple syntax to use this tag.

****

The <img> tag is an empty tag, which means that it can contain only list of attributes and it has no closing tag.

Example :-

To try following example, let's keep our HTML file test.htm and image file test.png in the same directory:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Using Image in Webpage</title>
    </head>
    <body>
        <p>Simple Image Insert</p>
        <img src="test.png" alt="Test Image" />
    </body>
</html>
```

This will produce following result:-

Simple Image Insert

You can use PNG, JPEG or GIF image file based on your comfort but make sure you specify correct image file name in src attribute. Image name is always case sensitive. Set Image Location Usually we keep our all the images in a separate directory. So let's keep HTML file test.htm in our home directory and create a subdirectory images inside the home directory where we will keep our image test.png. **Example :-**

Assuming our image location is "image/test.png", try the following example:

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

```
<head>
        <title>Using Image in Webpage</title>
</head>
<body>
        <p>Simple Image Insert</p>
        <img src="images/test.png" alt="Test Image" />
</body>
</html>
```

This will produce following result:-

Simple Image Insert

THE ALT ATTRIBUTE :-

The **alt** attribute is used to define an **alternate text** for an image. The value of the alt attribute is author-defined text:

```
<img src="images/test.png" alt="Test Image" />
```

The alt attribute is a mandatory attribute which specifies an alternate text for an image, if the image cannot be displayed. The alt attribute tells the reader what he or she is missing on a page if the browser can't load images. The browser will then display the alternate text instead of the image. It is a good practice to include the alt attribute for each image on a page, to improve the display and usefulness of your document for people who have text-only browsers or use screen readers.

SET IMAGE LOCATION :-

Usually we keep our all the images in a separate directory. So let's keep HTML file test.htm in our home directory and create a subdirectory images inside the home directory where we will keep our image test.png.

Example :-

Assuming our image location is "image/test.png", try the following example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Using Image in Webpage</title>
    </head>
    <body>
        <p>Simple Image Insert</p>
        <img src="images/test.png" alt="Test Image" />
    </body>
</html>
```

This will produce following result:-

Simple Image Insert

SET IMAGE WIDTH/HEIGHT :-

You can set image width and height based on your requirement using width and height attributes. You can specify width and height of the image in terms of either pixels or percentage of its actual size.

Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>Set Image Width and Height</title>
    </head>
        <body>
        <p>Setting image width and height</p>
        <img src="test.png" alt="Test Image" width="150" height="100"/>
```

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

This will produce following result:

Simple Image Insert

SET IMAGE BORDER:-

By default image will have a border around it, you can specify border thickness in terms of pixels using border attribute. A thickness of 0 means, no border around the picture.

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>Set Image Border</title>
    </head>
        <body>
        <p>Setting image Border</p>
        <img src="test.png" alt="Test Image" border="3"/>
    </body>
</html>
```

This will produce following result:-

Setting image Border

SET IMAGE ALIGNMENT :-

By default image will align at the left side of the page, but you can use align attribute to set it in the center or right.

Example :-

```
<!DOCTYPE html>
<html>
    <head>
        <title>Set Image Alignment</title>
    </head>
        <body>
        <p>Setting image Alignment</p>
        <img src="test.png" alt="Test Image" border="3" align="right"/>
    </body>
</html>
```

This will produce following result: -

HTML FRAMES :-

HTML frames allow users to present documents in multiple views, which may be independent windows or sub windows. Multiple views offer designers a way to keep certain information visible, while other views are scrolled or replaced. For example,

within the same window, one frame might display a static banner, a second a navigation menu, and a third the main document that can be scrolled through or replaced by navigating in the second frame.

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.

Example

Following is the example to create three horizontal frames:

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Frames</title>
    </head>
```

```
<frameset rows="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>
```

This will produce following result:

Example :-

Let's put 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>
```

This will produce following result: -

THE <FRAMESET> TAG ATTRIBUTES :-

Following are important attributes of the <frameset> tag:

Attribute	Description
cols	Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of four ways: ➢ Absolute values in pixels. For example to create three vertical frames, use ***cols="100, 500,100".*** ➢ A percentage of the browser window. For example tocreate three vertical frames, use ***cols="10%, 80%,10%".*** ➢ Using a wildcard symbol. For example to create three vertical frames, use ***cols="10%, *,10%".*** In this case wildcard takes remainder of the window. ➢ As relative widths of the browser window. For example to create three vertical frames, use ***cols="3*,2*,1*".*** This is an

	alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.
rows	This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example to create two horizontal frames, use ***rows="10%, 90%".*** You can specify the height of each row in the same way as explained above for columns.
border	This attribute specifies the width of the border of each frame in pixels. For example ***border="5".*** A value of zero means no border.
frameborder	This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example ***frameborder="0"*** specifies no border.
framespacing	This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example ***framespacing="10***" means there should be 10 pixels spacing between each frames.

THE <FRAME> TAG ATTRIBUTES :-

Following are important attributes of <frame> tag:

Attribute	Description
src	This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, ***src="/html/top_frame.htm"*** will load an HTML file available in html directory.
name	This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into another frame, in which case the second frame needs a name to identify itself as the target of the link.
frameborder	This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the

	<frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
marginwidth	This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example ***marginwidth="10".***
marginheight	This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example ***marginheight="10".***
noresize	By default you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example ***noresize="noresize".***
scrolling	This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example ***scrolling="no"*** means it should not have scroll bars.
longdesc	This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example ***longdesc="framedescription.htm"***

BROWSER SUPPORT FOR FRAMES :-

If a user is using any old browser or any browser which does not support frames then <noframes> element should be displayed to the user. So you must place a <body> element inside the <noframes> element because the <frameset> element is supposed to replace the <body> element, but if a browser does not understand <frameset> element then it should understand what is inside the <body> element which is contained in a <noframes> element.

You can put some nice message for your user having old browsers. For example Sorry!! your browser does not support frames. as shown in the above example.

FRAME'S NAME AND TARGET ATTRIBUTES :-

One of the most popular uses of frames is to place navigation bars in one frame and then load main pages into a separate frame. Let's see following example where a test.htm file has following code:

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Target Frames</title>
    </head>
    <frameset cols="200, *">
        <frame src="/html/menu.htm" name="menu_page" />
        <frame src="/html/main.htm" name="main_page" />
        <noframes>
            <body>
                Your browser does not support frames.
            </body>
        </noframes>
    </frameset>
</html>
```

Here we have created two columns to fill with two frames. The first frame is 200 pixels wide and will contain the navigation menubar implemented by menu.htm file. The second column fills in remaining space and will contain the main part of the page and it is implemented by main.htm file. For all the three links available in menubar, we have mentioned target frame as main_page, so whenever you click any of the links in menubar, available link will open in main_page. Following is the content of menu.htm file

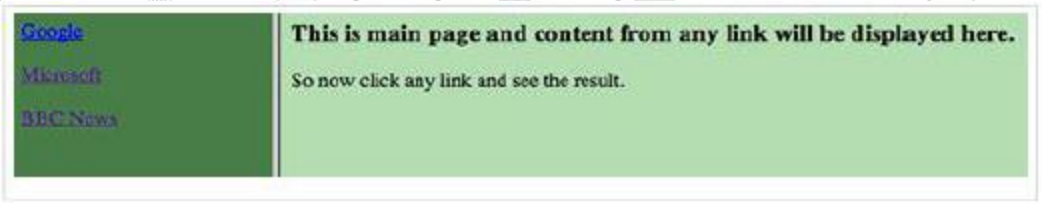
```
<!DOCTYPE html>
<html>
    <body bgcolor="#4a7d49">
        <a href="http://www.google.com" target="main_page">Google</a>
        <br /><br />
        <a href="http://www.microsoft.com" target="main_page">Microsoft</a>
```

```
        <br /><br />
        <a href="http://news.bbc.co.uk" target="main_page">BBC News</a>
    </body>
</html>
```

Following is the content of main.htm file:

```
<!DOCTYPE html>
<html>
    <body bgcolor="#b5dcb3">
        <h3>This is main page and content from any link will be displayed here.</h3>
        <p>So now click any link and see the result.</p>
    </body>
</html>
```

When we load test.htm file, it produces following result:



Now you can try to click links available in the left panel and see the result. The target attribute can also take one of the following values:

Option	Description
_self	Loads the page into the current frame.
_blank	Loads a page into a new browser window.
_parent	Loads the page into the parent window, which in the case of a single frameset, is the main browser window.
_top	Loads the page into the browser window, replacing any current frames.
_targetframe	Loads the page into a named targetframe.

HTML IFRAMES :-

You can define an inline frame with HTML tag <iframe>. The <iframe> tag is not somehow related to <frameset> tag, instead, it can appear anywhere in your document. The <iframe> tag defines a rectangular region within the document in which the browser can display a separate document, including scrollbars and borders. The src attribute is used to specify the URL of the document that occupies the inline frame.

Example :-

Following is the example to show how to use the <iframe>

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Iframes</title>
    </head>
    <body>
        <p>Document content goes here...</p>
        <iframe src="/html/menu.htm" width="555" height="200">
            Sorry your browser does not support inline frames.
        </iframe>
        <p>Document content also go here...</p>
    </body>
</html>
```

This will produce following result:

THE <IFRAME> TAG ATTRIBUTES :-

Most of the attributes of the <iframe> tag, including name, class, frameborder, id, longdesc, marginheight, marginwidth, name, scrolling, style, and title behave exactly like the corresponding attributes for the <frame> tag.

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, textarea fields, dropdown 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:

Attribute	Description
action	Backend script ready to process your passed data.
method	Method to be used to upload data. The most frequently used are **GET** and **POST** methods.
target	Specify the target window or frame where the result of the script will be displayed. It takes values like **_blank**, **_self**, **_parent** etc.
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:

- Text Input Controls
- Checkboxes Controls
- Radio Box Controls
- Select Box Controls
- File Select boxes
- Hidden Controls
- Clickable Buttons
- Submit and Reset Button

TEXT INPUT CONTROLS :-

There are three types of text input used on forms:

- Single-line text input controls - This control 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.
- 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.
- Multi-line text input controls - This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML **<textarea>** tag.

Single-line text input controls :-

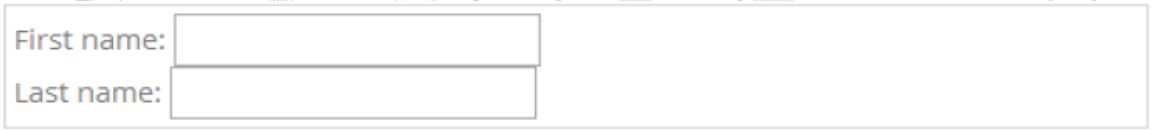
This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML **<input>** tag.

Example :-

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>
```

This will produce following result:-



Attributes :-

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

Attribute	Description
type	Indicates the type of input control and for text input control it will be set to text.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	This can be used to provide an initial value inside the control.
size	Allows to specify the width of the text-input control in terms of characters.
maxlength	Allows to specify the maximum number of characters a user can enter into the text box.

PASSWORD INPUT CONTROLS :-

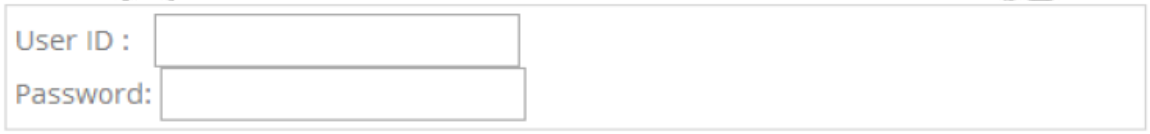
This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input> tag but type attribute is set to password.

Example:-

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>
```

This will produce following result:-



Attributes

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

Attribute	Description
type	Indicates the type of input control and for password input control it will be set to password.
name	Used to give a name to the control which is sent to the server to be

	recognized and get the value.
value	This can be used to provide an initial value inside the control.
size	Allows to specify the width of the text-input control in terms of characters.
maxlength	Allows to specify the maximum number of characters a user can enter into the text box.

MULTIPLE-LINE TEXT INPUT CONTROLS :-

This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML **<textarea>** tag.

Example:-

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>
```

This will produce following result:-

Attributes :-

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

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
rows	Indicates the number of rows of text area box.
cols	Indicates the number of columns of text area box

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.

Example:-

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>
```

This will produce following result:-

Attributes :-

Following is the list of attributes for **checkbox**.

Attribute	Description
type	Indicates the type of input control and for checkbox input control it will be set to checkbox.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	The value that will be used if the checkbox is selected.
checked	Set to checked if you want to select it by default.

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.

Example :-

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>
```

This will produce following result:-

Attributes :-

Following is the list of attributes for **radio button**.

Attribute	Description
type	Indicates the type of input control and for checkbox input control it will be set to radio.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	The value that will be used if the radio box is selected.
checked	Set to checked if you want to select it by default.

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.

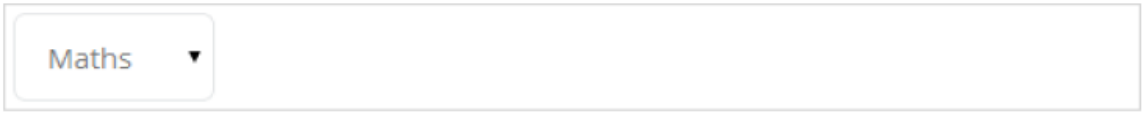
Example :-

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>
```

This will produce following result:-



Attributes :-

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

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
size	This can be used to present a scrolling list box.
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:

Attribute	Description
value	The value that will be used if an option in the select box is selected.
selected	Specifies that this option should be the initially selected value when the page loads.
label	An alternative way of labeling options.

BUTTONS :-

Type	Description
submit	This creates a button that automatically submits a form.
reset	This creates a button that automatically resets form controls to their

	initial values.
button	This creates a button that is used to trigger a client-side script when the user clicks that button.
image	This creates a clickable button but we can use an image as background of the button.

Example :-

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>
```

This will produce the following result :-

CASCADING STYLE SHEETS (CSS) :-

Cascading Style Sheets (CSS) describe how documents are presented on screens, in print, or perhaps how they are pronounced. W3C has actively promoted the use of style sheets on the Web since the Consortium was founded in 1994.

Cascading Style Sheets (CSS) provide easy and effective alternatives to specify various attributes for the HTML tags. Using CSS, you can specify a number of style properties for a given HTML element. Each property has a name and a value, separated by a colon (:). Each property declaration is separated by a semi-colon (;).

Example :-

First let's consider an example of HTML document which makes use of <font> tag and associated attributes to specify text color and font size:

```
<!DOCTYPE html>
<html>
	<head>
		<title>HTML CSS</title>
	</head>
	<body>
		<p><font color="green" size="5">Hello, World!</font></p>
	</body>
</html>
```

We can re-write above example with the help of Style Sheet as follows:

```
<!DOCTYPE html>
<html>
	<head>
		<title>HTML CSS</title>
	</head>
	<body>
		<p style="color:green;font-size:24px;">Hello, World!</p>
	</body>
```

```
</html>
```

This will produce following result:-

Hello, World!

You can use **CSS** in three ways in your HTML document:

- External Style Sheet - Define style sheet rules in a separate .css file and then include that file in your HTML document using HTML <link> tag.
- Internal Style Sheet - Define style sheet rules in header section of the HTML document using <style> tag.
- Inline Style Sheet - Define style sheet rules directly along with the HTML elements using style attribute.

Let's see all the three cases one by one with the help of suitable examples.

External Style Sheet :-

If you need to use your style sheet to various pages, then it's always recommended to define a common style sheet in a separate file. A cascading style sheet file will have extension as .css and it will be included in HTML files using <link> tag.

Example:-

Consider we define a style sheet file style.css which has following rules:

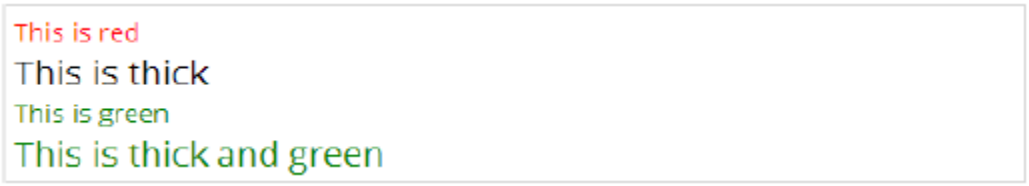
```
.red    {
color: red;
}
.thick  {
font-size:20px;
}
.green  {
color:green;
}
```

Here we defined three CSS rules which will be applicable to three different classes defined for the HTML tags. I suggest you should not bother about how these rules are being defined because you will learn them while studying CSS.

Now let's make use of the above external CSS file in our following HTML document:

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML External CSS</title>
        <link rel="stylesheet" type="text/css" href="/html/style.css">
    </head>
    <body>
        <p class="red">This is red</p>
        <p class="thick">This is thick</p>
        <p class="green">This is green</p>
        <p class="thick green">This is thick and green</p>
    </body>
</html>
```

This will produce following result:-



Internal Style Sheet:-

If you want to apply Style Sheet rules to a single document only then you can include those rules in header section of the HTML document using **<style>** tag. Rules defined in internal style sheet overrides the rules defined in an external CSS file.

Example :-

Let's re-write above example once again, but here we will write style sheet rules in the same HTML document using <style> tag:

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

```
<head>
        <title>HTML Internal CSS</title>
        <style type="text/css">
        .red{
        color: red;
        }
        .thick{
        font-size:20px;
        }
        .green{
        color:green;
        }
        </style>
</head>
<body>
        <p class="red">This is red</p>
        <p class="thick">This is thick</p>
        <p class="green">This is green</p>
        <p class="thick green">This is thick and green</p>
</body>
</html>
```

This will produce following result:-

This is red
This is thick
This is green
This is thick and green

Inline Style Sheet :-

You can apply style sheet rules directly to any HTML element using style attribute of the relevant tag. This should be done only when you are interested to make a particular change in any HTML element only. Rules defined inline with the element

overrides the rules defined in an external CSS file as well as the rules defined in <style> element.

Example :-

Let's re-write above example once again, but here we will write style sheet rules along with the HTML elements using style attribute of those elements.

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Inline CSS</title>
    </head>
    <body>
        <p style="color:red;">This is red</p>
        <p style="font-size:20px;">This is thick</p>
        <p style="color:green;">This is green</p>
        <p style="color:green;font-size:20px;">This is thick and green</p>
    </body>
</html>
```

This will produce following result:-