Chapter – 2

HTML and Graphics

INTRODUCTION:

1. HTML:

HTML stands for Hyper Text Markup Language, which is the most widely used language on web to develop web pages. A markup language is a set of markups tags. HTML documents are described by HTML tags. Each HTML tag describes different document content.

HTML tags are keywords (tag names) surrounded by angle brackets. For example: <tagname> contents </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 slash before the tag name.

HTML was created by Tim Berners Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01version is widely used but currently we are having HTML 5 version which is an extension to HTML 4.01 and this version was published in 2012.

2. Structure of HTML:

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

3. < ! DOCTYPE>:

The <! DOCTYPE> declaration must be the very first thing in our HTML document, before the <html> tag. The <! DOCTYPE> declaration is not an HTML tag, it is an instruction to the web browser about what version of HTML the page is written. In HTML 4.01, the <! DOCTYPE> declaration refers to a Document Type Definition (DTD), because HTML 4.01 was based on Standard Generalized Markup Language (SGML). The DTD specifies the rules for the markup language, so that the browsers render the content correctly. HTML 5 is not based on SGML, and therefore does not require a reference to a DTD.

4. Elements:

An HTML element usually consists of a start tag and end tag, with the content inserted in between, for example: <tagname>Content goes here...</tagname>. The HTML element is everything from the start tag to the end tag, for example: My first paragraph .

HTML elements can be nested (elements can contain elements). All HTML documents consist of nested HTML elements.

HTML elements with no content are called empty elements.
 is an empty element without a closing tag (the <br> tag defines a line break). Empty elements can be "closed" in the opening tag like this: <br />. HTML5 does not require empty elements to be closed. But if we want stricter validation, or if we need to make our document readable by XML parsers, we must close all HTML elements properly.

HTML tags are not case sensitive: <P> means the same as <p>. The HTML5 standard does not require lowercase tags, but W3C recommends lowercase in HTML, and demands lowercase for stricter document types like XHTML.

```
<! DOCTYPE HTML>
<html>
<head>
<title> Elements Example</title>
</head>
<body>
<h1>This is <i>italic</i> heading</h1>
<p >This is  <u>underlined</u> paragraph </p>
</body>
</html>
```

5. Attributes:

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

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

```
<! 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>
```

6. Comment Tag:

Comment is a piece of code which is ignored by web browser. It is a good practice to add comment into HTML code, especially in complex documents, to indicate sections of a document and any other notes to anyone looking at the code. Comments help us and others understand our code and increases code readability.

HTML comments are placed in between <! -- Comment --> tags. So any content placed with in <! -- Comment -- > tags will be treated as comment and will be completely ignored by the browser.

7. Meta Tag:

HTML lets us specify metadata: additional important information about a document is a variety of ways. The META elements can be used to include name/value pairs describing properties of the HTML document, such as author, expiry data, a list of keywords, document author, etc.

The <meta> tag is used to provide such additional information. This tag is an empty element and so does not have a closing tab but it carries information within its attributes.

We can include one or more meta tags in our document based on what information we want to keep in our document but in general, meta tag do not impact physical appearance of the document so from appearance point of view, it does not matter if we include them or not.

We can add metadata to our web pages by placing <meta> tags inside the header of the document which is represented by <head> and </head> tags.

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML,CSS,XML,JavaScript">
  <meta name="author" content="John Doe">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

HTML BASIS:

1. Text Formatting:

HTML also defines special elements for defining text with a special meaning. HTML uses elements like <b> and <i> for formatting output, like bold or italic text. Formatting elements were designed to display special types of text:

- <b> - Bold text
- - Important text
- <i> - Italic text
- - Emphasized text
- <mark> - Marked text
- <small> - Small text
- <del> - Deleted text
- <ins> - Inserted text

- <sub> - Subscript text
- <sup> - Superscript text

2. Phrase Element:

The phrase elements have been designed for specific purpose, though they are displayed in a similar way as other basic tags like <b>, <i>, <pre> and <tt>.

- **Emphasized Text:** Anything that appears within   element is displayed as emphasized text.
- **Marked Text:** Anything that appears within <mark> </mark> element is displayed as marked with yellow ink.
- **Strong Text:** Anything that appears within   element is displayed as important text.
- **Short Quotations:** The <q> </q> element is used when we want to add a double quote within a sentence.

3. Listing:

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

a. 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.

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

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

Example:

```
<!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>
```

b. Ordered Lists:

If we are required to put our items in a numbered list instead of bulleted 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>.

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

<ol type="1"> - Default-Case Numerals.
<ol type="I"> - Upper-Case Numerals.
<ol type="i"> - Lower-Case Numerals.
<ol type="a"> - Lower-Case Letters.
<ol type="A"> - Upper-Case Letters.

We can use start attribute for <ol> tag to specify the starting point of numbering we need. Following are the possible options:

<ol type="1" start="4"> - Numerals starts with 4.
<ol type="I" start="4"> - Numerals starts with IV.
<ol type="i" start="4"> - Numerals starts with iv.
<ol type="a" start="4"> - Letters starts with d.
<ol type="A" start="4"> - Letters starts with D.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
  <ol type="i" start="4">
  <li>Beetroot</li>
  <li>Ginger</li>
  <li>Potato</li>
  <li>Radish</li>
  </ol>
</body>
</html>
```

c. 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>
```

Nested List:

When we create a list with in another list then it is called nested list. For example:

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

4. Using Character Entities for Special Character:

There are few special characters or symbols which are not available to be typed directly from keyboard. Character Entities can be used to display those symbols/special characters also. Browsers may not support all entity names, but the support for numbers is good.

Some Useful HTML Character Entities

Result	Description	Entity Name	Entity Number
	Non-Breaking Space		
<	Less Than	&lt;	&#60;
>	Greater Than	&gt;	&#62;
&	Ampersand	&amp;	&#38;
"	Double Quotation Mark	&quot;	&#34;
'	Single Quotation Mark (Apostrophe)	&apos;	&#39;
¢	Cent	&cent;	&#162;
£	Pound	&pound;	&#163;
¥	Yen	&yen;	&#165;
€	Euro	&euro;	&#8364;
©	Copyright	&copy;	&#169;
®	Registered Trademark	&reg;	&#174;

5. <Font> Element and Attributes:

Fonts play very important role in making a website more user friendly and increasing content readability. Font face and color depends entirely on the computer and browser that is being used to view our page but we can use HTML tag to add style, size, and color to the text on our website.

The font tag is having three attributes called size, color, and face to customize our fonts. To change any of the font attributes at any time within our webpage, simply use the tag. The text that follows will remain changed until we close with the tag. We can change one or all of the font attributes within one <font> tag.

```
<!DOCTYPE html>
<html>
<head>
<title> Font Exmple </title>
</head>
<body>
<font face="Times New Roman" size="5" color = "Green">Use of Font element and attribute</font><br />
<font face="Impact" size="5" color = "red">Use of Font element and attribute</font><br />
</body>
</html>
```

6. Grouping Element:

There are two important tags which we use vary frequently to group various other HTML tags and they are:

a. ** tag:**

This is the very important block level tag which plays a big role in grouping various other HTML tags and applying CSS on group of elements. Even now tag can be used to create webpage layout where we define different parts (Left, Right, Top, and Bottom) of the page using tag. This tag does not provide any visual change on the block but this has more meaning when it is used with CSS.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML div Tag</title>
<style type = "text/css" media = "all">
#contentinfo p {
   line-height: 20px;
         margin: 30px;
         padding-bottom: 20px;
         text-align: justify;
         width: 140px;
         color: red;
}
</style>
</head>
<body>
<div id="contentinfo">
<p>Welcome to our website. We provide tutorials on various subjects.</p>
</div>
</body>
</html>
```

b. ** tag:**

The HTML is an inline element and it can be used to group inline elements in an HTML document. This tag also does not provide any visual change on the block but has more meaning when it is used with CSS.

The main difference between the tag and the tag is that the tag is used with inline elements whereas the tag is used with block level elements.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML span Tag</title>
</head>
<body>
<p>This is a paragraph <span style="color:#FF0000;">
This is a paragraph</span>
This is a paragraph</p>
<p><span style="color:#8866ff;">
```

```
This is another paragraph</span></p>
</body>
</html>
```

- **Block Elements:**

They appear on the screen as if they have a line break before and after them. For example: , <h1> </h1>, <ul> , <ol> <hr/>, <blockquote>, etc. They all start on their own new line, and anything that follows them appears on its own new line.

- **Inline Elements:**

They appear within sentence and do not have to appear on a new line of their own. For example: <b>, <i>, <u>, , , <sup>, <sub>, <big>, <small>, <li>, etc.

DIFFERENCE BETWEEN <DIV> TAG AND <SPAN> TAG:

<Div>	<Span>
is a block-level-element	is an inline element.
If we need to modify a large division, the height, move an element, or contain other elements we should use a	if we want to adjust a small portion of text and not break it out of the current line we should use a
Example: This is an example of a div tag that has a maximum width of 200 pixels and a silver background. 	Example: This text is red text and an example of a span tag.

LINKS AND NAVIGATION:

1. LINKING TO OTHER WEB PAGES:

A webpage can contain various links that take us 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 we can create hyperlinks using text or images available on a webpage.

- **Linking Documents:**

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.

Link Text

Example:

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

```
<head>
<title>Hyperlink Example</title>
</head>
<body>
<p>Click following link</p>
<a href="https://www.facebook.com" target="_self">Tutorials Point</a>
</body>
</html>
```

- **The Target Attribute:**

We have used target attribute in our previous example. This attribute is used to specify the location where linked document is opened. Following are possible options:

Option	Description
_blank	Opens the linked document in a new window or tab.
_self	Opens the linked document in the same frame.
_parent	Opens the linked document in the parent frame.
_top	Opens the linked document in the full body of the window.
targetframe	Opens the linked document in a named *targetframe*.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Hyperlink Example</title>
<base href="https://www.facebook.com/">
</head>
<body>
<p>Click any of the following links</p>
<a href="/html/index.htm" target="_blank">Opens in New</a> |
<a href="/html/index.htm" target="_self">Opens in Self</a> |
<a href="/html/index.htm" target="_parent">Opens in Parent</a> |
<a href="/html/index.htm" target="_top">Opens in Body</a>
</body>
</html>
```

- **Use of Base Path:**

When we link HTML documents related to the same website, it is not required to give a complete URL for every link. We can get rid of it if we use <base> tag in our HTML document header. This tag is used to give a base path for all the links. So our browser will concatenate given relative path to this base path and will make a complete URL.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Hyperlink Example</title>
<base href="https://www.google.com/">
</head>
<body>
<p>Click following link</p>
<a href="/html/index.htm" target="_blank">Google</a>
</body>
</html>
```

- **Linking to a Page Section:**

We can create a link to a particular section of a given webpage by using name attribute. This is a two-step process:

- First create a link to the place where we want to reach with-in a webpage and name it using <a...> tag as follows:
  <h1>HTML Text Links </h1>
- Second step is to create a hyperlink to link the document and place where we want to reach:
  Go to the Top

This will produce following link, where we can click on the link generated **Go to the Top** to reach to the top of the HTML Text Link tutorial.

- **Setting Link Colors:**

We can set colors of our links, active links and visited links using link, alink and vlink attributes of <body> tag.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Hyperlink Example</title>
</head>
<body alink="#54A250" link="#040404" vlink="#F40633">
<p>Click following link</p>
<a href="https://www.facebook.com" target="_blank" >Facebook</a>
</body>
</html>
```

- **Download Links:**

We can create text link to make our PDF, or DOC or ZIP files downloadable. This is very simple, we just need to give complete URL of the downloadable file as follows:

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Hyperlink Example</title>
</head>
<a href="https://www.tutorialspoint.com/page.pdf">Download PDF File</a>
</body>
</html>
```

2. URL (Uniform Resource Locator):

A **URL** (**Uniform Resource Locator**) is a form of URI (Uniform Resource Identifier) and is a standardized naming convention for addressing documents accessible over the Internet and Intranet. An example of a URL is http://www.computerhope.com, which is the URL for the Computer Hope website.

- **Overview of a URL:**

- **http://**

The "http" stands for HyperText Transfer Protocol and is what enables the browser to know what protocol it is going to use to access the information specified in the domain. After the http is the colon (:) and two forward slashes (//) that separate the protocol from the remainder of the URL.

Tip: A URL is not explicit to HTTP addresses; HTTPS, FTP, TFTP, Telnet, and other addresses are also considered URLs and may not follow the same syntax as above example.

- **www.**

Next, www. Stands for World Wide Web and is used to distinguish the content. This portion of the URL is not required and many times can be left out. For example, typing "http://computerhope.com" would still get us to the Computer Hope web page. This portion of the address can also be substituted for an important subpage known as a subdomain. For example, http://support.computerhope.com forwards us to the main help section of Computer Hope.

- **computerhope.com**

Next, computerhope.com is the domain name for the website. The last portion of the domain is known as the "domain suffix", or TLD, and is used to identify the type or location of the website. For example, **.com** is short for commercial, .org is short for an organization, and .co.uk is the United Kingdom. There are dozens of other domain suffixes available. To get a domain, we would register the name through a domain registrar.

- **/jargon/u/**

Next, the "jargon" and "u" portions of the above URL are the directories of where on the server the web page is located. In this example, the web page is two directories deep, so if we were trying to find the file on the server, it would be in the */public_html/jargon/u* directory. With most servers, the public_html directory is the default directory containing the HTML files.

- **url.htm**

Finally, url.htm is the actual web page on the domain we're viewing. The trailing .htm is the file extension of the web page that indicates the file is an HTML file. Other common file extensions on the Internet include .html, .php, .asp, .cgi, .xml, .jpg, and .gif. Each of these file extensions performs a different function, just like all the different types of files on your computer.

- **What characters are not allowed in a URL?**

Most people realize that a space is not allowed in a URL. However, it is also important to realize, as documented in RFC 1738, the URL string can only contain alphanumeric characters and the ! $ -_ + * ' ( ) , characters. Any other characters that are needed in the URL must be encoded.

- **Understanding more complex URLs and parameters**

When a URL points to a script that performs additional functions, such as a search engine pointing to a search results page, additional information (parameters) is added to the end of the URL. Below is additional information about a URL that points to the Computer Hope Search page, with the search query of "example search".

http://www.computerhope.com/cgi-bin/search.cgi?q=example%20search

In this URL, the script file being pointed to is **search.cgi** in the cgi-bin directory. Because this file ends with .cgi, it is assumed to be a Perl script.

After the script file name there is a **?** (question mark). The question mark in a URL separates the URL from all the parameters or variables that are being sent to the script. In the above example, the parameter being sent is **q=example%20search**. The "q" is a variable name, and the "example%20search" is the value being sent to that variable. Because no spaces are allowed in a URL, the space has been encoded as %20. In many scripts, a **+** (plus) is also used to represent a space.

In our example, because there is a variable the script would use it as it is executed. Scripts are also not limited to only one variable. If the script needs multiple variables, each variable can be separated with an **&** (ampersand) as shown in the example below.

http://www.computerhope.com/cgi-bin/search.cgi?q=example%20search&example=test

In the above example, there are two different variables. The "q" variable equals "example search" and the "example" variable equals "test". If the script was looking for an *example* variable, it could be processed and perform an additional feature.

- **URI (Uniform Resource Identifier):**

Short for Uniform Resource Identifier, URI is defined in RFC (Remote Function Call) 1630 as a reference to addresses, names, or objects that apply to registered protocols or name spaces on the Internet. For example, URL and URN are forms of Uniform Resource Identifiers.

3. Types of URL:

- **Absolute URL:**

In addition to several other meanings, the word **absolute**, in English, means "*not dependent on anything else*". It also means "*free from doubt*". An **Absolute URL** is, thus, something that is *independent* or *free from any relationship*. When we use an absolute URL, we point directly to a file. Hence, an absolute URL specifies the *exact* location of a file/directory on the internet. It also follows that each absolute URL is unique, which means that if two URLs are identical, they point to the same file.

For example:

- **http://www.webdevelopersnotes.com/images/email.gif** specifies an image file *email.gif* located in the *images* directory, under www.webdevelopersnotes.com domain name.
- Similarly, the absolute URL of the document we are viewing is **http://www.webdevelopersnotes.com/design/relative_and_absolute_urls.php3** which is a page in the directory called *design* on this web site.

- **Relative URL:**

A relative URL points to a file/directory in relation to the present file/directory. Let us understand relative URLs through a small exercise.
Look at the two URL above. We want to include (display) the image file **email.gif** stored in the images directory of **www.webdevelopersnotes.com** domain on this (**relative_and_absolute_urls.php3** stored in the **design** directory) page.

There are two ways to do this. We can either refer to it using an absolute URL or use a relative URL. The tag for this image display will be as follows:

- **Using an Absolute URL in an tag:**

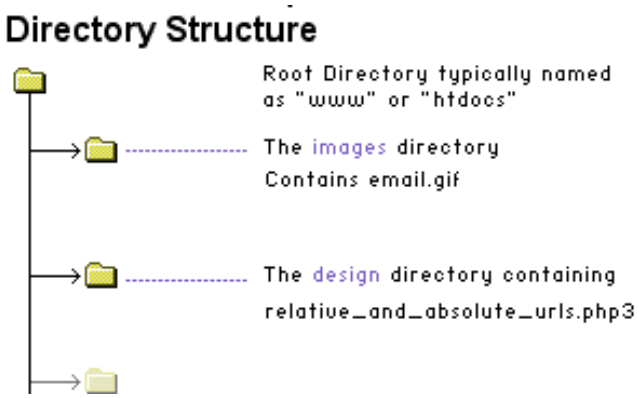
```
<img src="http://www.webdevelopersnotes.com/images/email.gif"width="..." height="..." />
```

- **Using a Relative URL in an tag**

```
<img src="../images/email.gif" width="..." height="..." />
```

The absolute URL is straight forward but in the relative URL we'll notice that we have referred to the image with **../images/email.gif**. In order to understand the relative URL, we need to know about the directory structure of this web site.
This web site has several sections and the files and web pages for each section have been segregated into different directories. This helps us to keep things organized and uncluttered on the web site. Under the document or server root directory (the main directory of the web site),

we have a directory called images which stores all common images used on the pages of this web site. The **image** email.gif resides in this directory. We have another directory called **design** which is at the same level as images i.e. it is also in the document root directory. This design directory contains the files and web pages for the "Web Page Design" section of this web site. Diagrammatically, the scenario can be represented as:



Now to access email.gif file from relative_and_absolute_urls.php3 page using a relative URL we put ../images/email.gif in the SRC attribute. We, thus, instruct the browser to first go one level up (i.e. to the document root) and then move to the images directory and pick up the file email.gif. The two periods (..) instruct the server to move up one directory (which is the root directory), then enter images directory (/images) and finally point at email.gif.

4. HTML Email Tag:

HTML <a> tag provides us option to specify an email address to send an email. While using <a> tag as an email tag, we will use **mailto:email address** along with *href* attribute. Following is the syntax of using **mailto** instead of using http.

Send Email

This code will generate following link which we can use to send email: Send Email

Now if a user clicks this link, it launches one Email Client (like Lotus Notes, Outlook Express etc.) installed on our user's computer. There is another risk to use this option to send email because if user do not have email client installed on their computer then it would not be possible to send email.

Default Settings:

We can specify a default *email subject* and *email body* along with our email address. Following is the example to use default subject and body.

**Send Feedback **

This code will generate following link which we can use to send email: Send Feedback

- **Host Address:**

The host address is where a website can be found, either IP address (four sets of numbers from 0 to 258, for example 68.178.157.132) or more commonly the domain name for a site such as www.computerhope.com. Note that "www" is not actually part of the domain name though it is often used in the host address.

- **File Path:**

The file path always begins with a forward slash character, and may consist of one or more directory or folder names. Each directory name is separated by forward slash characters and the file path may end with a filename at the end. Here index.html is the filename which is available in html directory: https://www.computerhope.com/html/index.htm.

- **Local Links:**

A local link (link to the same web site) is specified with a relative URL (without http://www....).

Example

HTML Images

- **External Paths:**

External pages can be referenced with a full URL or with a path relative to the current web page. This example uses a full URL to link to a web page:

Example

HTML tutorial

5. **Advance Email Links:**

Now a days due to improvement in technology and different approaches of programming we have seen different methods of sending emails. These all methods used are especially for security purpose. They are used with anti-spam, certain formats, anti-spam sounds.

Images, Audio and Video:

1. **Image:**

- **Inserting Image:**

Images are very important to beautify as well as to depict many complex concepts in simple way on our web page. We can insert any image in our 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:

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

We can use PNG, JPEG or GIF image file based on our comfort but make sure we specify correct image file name in **src** attribute. Image name is always case sensitive. The **alt** attribute is a mandatory attribute which specifies an alternate text for an image, if the image cannot be displayed.

- **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:*** "/html/image/test.png"

- **Set Image Width/Height:**

We can set image width and height based on our requirement using **width** and **height** attributes. We 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="/html/images/test.png" alt="Test Image" width="150" height="100"/>
</body>
</html>
```

- **Set Image Border:**

By default image will have a border around it, we 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="/html/images/test.png" alt="Test Image" border="3"/>
</body>
</html>
```

- **Set Image Alignment:**

By default image will align at the left side of the page, but we 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="/html/images/test.png" alt="Test Image" border="3" align="right"/>
</body>
</html>
```

2. Images as Link:

We have seen how to create hypertext link using text and we also learnt how to use images in our webpages. Now we will learn how to use images to create hyperlinks. It's simple to use an image as hyperlink. We just need to use an image inside hyperlink at the place of text as shown below:

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Image Hyperlink Example</title>
</head>
<body>
<p>Click Image for Facbook</p>
<a href="https://www.facebook.com" target="_self">
<img src="\HTML Programs\on.jpg" alt="Facbook" border="0"/>
</a>
</body>
</html>
```

Mouse-Sensitive Images:

The HTML and XHTML standards provide a feature that lets us embed many different links inside a single image. We can create different links on the single image based on different coordinates available on the image. Once different are links attached to different coordinates, we can click different parts of the image to open target documents. Such mouse-sensitive images are known as image maps.

There are two ways to create image maps:

- **Server-side image maps -** This is enabled by the **ismap** attribute of the <img> tag and requires access to a server and related image-map processing applications.
- **Client-side image maps -** This is created with the **usemap** attribute of the <img> tag, along with corresponding <map> and <area> tags.

Server-Side Image Maps:

Here we simply put our image inside a hyper link and use **ismap** attribute which makes it special image and when the user clicks some place within the image, the browser passes the coordinates of the mouse pointer along with the URL specified in the <a> tag to the web server. The server uses the mouse-pointer coordinates to determine which document to deliver back to the browser.

When *ismap* is used, the href attribute of the containing <a> tag must contain the URL of a server application like a cgi or PHP script etc. to process the incoming request based on the passed coordinates.

The coordinates of the mouse position are screen pixels counted from the upper-left corner of the image, beginning with (0, 0). The coordinates, preceded by a question mark, are added to the end of the URL.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>ISMAP Hyperlink Example</title>
</head>
<body>
<p>Click following link</p>
<a href="/cgi-bin/ismap.cgi" target="_self">
  <img ismap src="/images/logo.png" alt="Tutorials Point" border="0"/>
</a>
</body>
</html>
```

Then the browser sends the following search parameters to the web server which can be processed by **ismap.cgi** script or **map file** and we can link whatever documents we like to these coordinates: ***/cgi-bin/ismap.cgi?20,30***

This way we can assign different links to different coordinates of the image and when those coordinates are clicked, we can open corresponding linked document.

- **Client-Side Image Maps:**

Client side image maps are enabled by the **usemap** attribute of the <img /> tag and defined by special <map> and <area> extension tags.

The image that is going to form the map is inserted into the page using the <img /> tag as a normal image, except it carries an extra attribute called **usemap**. The value of the usemap attribute is the value which will be used in a <map> tag to link map and image tags. The <map> along with <area> tags define all the image coordinates and corresponding links.

The <area> tag inside the map tag, specifies the shape and the coordinates to define the boundaries of each clickable hotspot available on the image. Here's an example from the image map:

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>USEMAP Hyperlink Example</title>
</head>
<body>
<p>Search and click the hotspot</p>
<img src=/images/html.gif alt="HTML Map" border="0" usemap="#html"/>
<!-- Create  Mappings -->
<map name="html">
  <area shape="circle"
  coords="80,80,20" href="/css/index.htm" alt="CSS Link" target="_self" />
  <area shape="rect"
  coords="5,5,40,40" alt="jQuery Link" href="/jquery/index.htm" target="_self" />
</map>
</body>
</html>
```

- **Coordinate System:**

The actual value of coords is totally dependent on the shape in question. Here is a summary, to be followed by detailed examples:

rect = x_1 , y_1 , x_2 , y_2

x_1 and y_1 are the coordinates of the upper left corner of the rectangle; x_2 and y_2 are the coordinates of the lower right corner.

circle = x_c , y_c , radius

x_c and y_c are the coordinates of the center of the circle, and radius is the circle's radius. A circle centered at 200, 50 with a radius of 25 would have the attribute *coords="200, 50,25"*

poly = x_1 , y_1 , x_2 , y_2 , x_3 , y_3 , … x_n , y_n

The various x-y pairs define vertices (points) of the polygon, with a "line" being drawn from one point to the next point. A diamond-shaped polygon with its top point at 20, 20 and 40 pixels across at its widest points would have the attribute *coords="20,20,40,40,20,60,0,40"*.

All coordinates are relative to the upper-left corner of the image (0, 0). Each shape has a related URL. We can use any image software to know the coordinates of different positions.

3. Image Format:

Each of image file types has their own pros and cons. They were created for specific, yet different, purposes. What's the difference, and when is each format appropriate to use?

- **JPEG:**

JPEG is short for Joint Photographic Experts Group, and is the most popular among the image formats used on the web. JPEG files are very 'lossy', meaning so much information is lost from the original image when you save it in a JPEG file.

This is because JPEG discards most of the information to keep the image file size small; which means some degree of quality is also lost.

Almost every digital camera can shoot and save in the JPEG format. JPEG is very web friendly because the file is smaller, which means it takes up less room, and requires less time to transfer to a site. Moreover it is less grainy then GIF, the old king of the internet roost. Since 1994, JPEG has been considered the standard.

Pros of JPEG:

- 24-bit color, with up to 16 million colors
- Rich colors, great for photographs that need fine attention to color detail
- Most used and most widely accepted image format
- Compatible in most OS (Mac, PC, Linux)

Cons of JPEG:

- They tend to discard a lot of data
- After compression, JPEG tends to create artifacts
- Cannot be animated
- Does not support transparency

- **GIF:**

GIF, short for Graphics Interchange Format, is limited to the 8 bit palette with only 256 colors. GIF is still a popular image format on the internet because image size is relatively small compared to other image compression types.

GIF is most suitable for graphics, diagrams, cartoons and logos with relatively few colors. GIF is still the chosen format for animation effects.

Compared to JPEG, it is lossless and thus more effective with compressing images with a single color, but pales in detailed or dithered pictures. In other words, GIF is lossless for images with 256 colors and below. So for a full color image, it may lose up to 99.998% of its colors.

One edge of the GIF image format is the interlacing feature, giving the illusion of fast loading graphics. When it loads in a browser, the GIF first appears to be blurry and fuzzy, but as soon as more data is downloaded, the image becomes more defined until all the date has been downloaded.

Pros of GIF:

- Can support transparency
- Can do small animation effects
- 'Lossless' quality–they contain the same amount of quality as the original, except of course it now only has 256 colors
- Great for images with limited colors, or with flat regions of color

Cons of GIF:

- Only supports 256 colors
- It's the oldest format in the web, having existed since 1989. It hasn't been updated since, and sometimes, the file size is larger than PNG.

BMP:

The Windows Bitmap or BMP files are image files within the Microsoft Windows operating system. In fact, it was at one point one of the few image formats. These files are large and uncompressed, but the images are rich in color, high in quality, simple and compatible in all Windows OS and programs. BMP files are also called raster or paint images.

BMP files are made of millions and millions of dots called 'pixels,' with different colors and arrangements to come up with an image or pattern. It might be an 8-bit, 16-bit or 24-bit image. Thus when we make a BMP image larger or smaller, we are making the individual pixels larger, and thus making the shapes look fuzzy and jagged.

BMP files are not great and not very popular. Being oversized, bitmap files are not what we call 'web friendly', nor are they compatible in all platforms and they do not scale well.

Pros of BMP:

- Works well with most Windows programs and OS, you can use it as a Windows wallpaper

Cons of BMP:

- Does not scale or compress well
- Again, very huge image files making it not web friendly
- No real advantage over other image formats

TIFF:

TIFF was created by Aldus for 'desktop publishing', and by 2009 it was transferred to the control of Adobe Systems. TIFF is popular among common users, but has gained recognition in the graphic design, publishing and photography industry. It is also popular among Apple users.

The TIFF image format is easy to use with software that deals with page layout, publishing and photo manipulation via fax, scanning, word processing, etc. TIFF is very flexible, it can be lossy or lossless. TIFF is a rich format and supported by many imaging programs.

It is capable of recording halftone image data with different pixel intensities, thus is the perfect format for graphic storage, processing and printing. This makes TIFF the superior raster image format.

Pros of TIFF:

- Very flexible format, it supports several types of compression like JPEG, LZW, ZIP or no compression at all.
- High quality image format, all color and data information are stored
- TIFF format can now be saved with layers

Cons of TIFF:

- Very large file size–long transfer time, huge disk space consumption, and slow loading time.

PNG

PNG or (Portable Network Graphics) is a recently introduced format, so not everyone is familiar with it. But PNG has been approved as a standard since 1996. It is an image format specifically designed for the web. PNG is, in all aspects, the superior version of the GIF.

Just like the GIF format, the PNG is saved with 256 colors maximum but it saves the color information more efficiently. It also supports an 8 bit transparency.

PNG was actually created for the intent to replace the GIF as an image format that doesn't require a patent license. PNG can support 24 bit RGB color images, grayscale images, both with and without alpha channels. RGB cannot support CMYK color spaces, and is not designed for print graphics.

Pros of PNG:

- Lossless, so it does not lose quality and detail after image compression
- In a lot ways better then GIF. To start, PNG often creates smaller file sizes than GIF
- Supports transparency better than GIF

Cons of PNG:

- Not good for large images because they tend to generate a very large file, sometimes creating larger files than JPEG.
- Unlike GIF however, it cannot be animated.
- Not all web browsers can support PNG.

4. Working with Multimedia:

Sometimes we need to add music or video into our web page. The easiest way to add video or sound to our web site is to include the special HTML tag called **<embed>**. This tag causes the browser itself to include controls for the multimedia automatically provided browser supports <embed> tag and given media type.

We can also include a **<noembed>** tag for the browsers which don't recognize the <embed> tag. We could, for example, use <embed> to display a movie of our choice, and <noembed> to display a single JPG image if browser does not support <embed> tag.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML embed Tag</title>
</head>
<body>
<embed src="/html/yourfile.mid" width="100%" height="60" >
  <noembed><img src="yourimage.gif" alt="Alternative Media" ></noembed>
</embed>
</body>
</html>
```

- **The <embed> Tag Attributes:**

Following is the list of important attributes which can be used with <embed> tag.

Attribute	Description
align	Determines how to align the object. It can be set to either *center, left or right*.
autostart	This Boolean attribute indicates if the media should start automatically. We can set it either true or false.
loop	Specifies if the sound should be played continuously (set loop to true), a certain number of times (a positive value) or not at all (false)
playcount	Specifies the number of times to play the sound. This is alternate option for *loop* if we are using IE.
hidden	Specifies if the multimedia object should be shown on the page. A false value means no and true values means yes.
width	Width of the object in pixels
height	Height of the object in pixels
name	A name used to reference the object.
src	URL of the object to be embedded.
volume	Controls volume of the sound. Can be from 0 (off) to 100 (full volume).

- **Multimedia Formats:**

Multimedia elements (like audio or video) are stored in media files. The most common way to discover the type of a file, is to look at the file extension. Multimedia files have formats and different extensions like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

- **Common Video Formats:**

MP4 is the new and upcoming format for internet video.

- MP4 is recommended by YouTube.

- MP4 is supported by Flash Players.
- MP4 is supported by HTML5.

Format	File	Description
MPEG	.mpg .mpeg	MPEG. Developed by the Moving Pictures Expert Group. The first popular video format on the web. Used to be supported by all browsers, but it is not supported in HTML5 (See MP4).
AVI	.avi	AVI (Audio Video Interleave). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
WMV	.wmv	WMV (Windows Media Video). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
QuickTime	.mov	QuickTime. Developed by Apple. Commonly used in video cameras and TV hardware. Plays well on Apple computers, but not in web browsers. (See MP4)
RealVideo	.rm .ram	RealVideo. Developed by Real Media to allow video streaming with low bandwidths. It is still used for online video and Internet TV, but does not play in web browsers.
Flash	.swf .flv	Flash. Developed by Macromedia. Often requires an extra component (plug-in) to play in web browsers.
Ogg	.ogg	Theora Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5.
WebM	.webm	WebM. Developed by the web giants, Mozilla, Opera, Adobe, and Google. Supported by HTML5.
MPEG-4 or MP4	.mp4	MP4. Developed by the Moving Pictures Expert Group. Based on QuickTime. Commonly used in newer video cameras and TV hardware. Supported by all HTML5 browsers. Recommended by YouTube.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML embed Tag</title>
</head>
<body>
<embed src="/html/yourfile.swf" width="200" height="200" >
  <noembed><img src="yourimage.gif" alt="Alternative Media" ></noembed>
</embed>
</body>
</html>
```

Audio Formats:

MP3 is the newest format for compressed recorded music. The term MP3 has become synonymous with digital music. If our website is about recorded music, MP3 is the choice.

Format	File	Description
MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface). Main format for all electronic music devices like synthesizers and PC sound cards. MIDI files do not contain sound, but digital notes that can be played by electronics. Plays well on all computers and music hardware, but not in web browsers.
RealAudio	.rm .ram	RealAudio. Developed by Real Media to allow streaming of audio with low bandwidths. Does not play in web browsers.
WMA	.wma	WMA (Windows Media Audio). Developed by Microsoft. Commonly used in music players. Plays well on Windows computers, but not in web browsers.
AAC	.aac	AAC (Advanced Audio Coding). Developed by Apple as the default format for iTunes. Plays well on Apple computers, but not in web browsers.
WAV	.wav	WAV. Developed by IBM and Microsoft. Plays well on Windows, Macintosh, and Linux operating systems. Supported by HTML5.
Ogg	.ogg	Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5.
MP3	.mp3	MP3 files are actually the sound part of MPEG files. MP3 is the most popular format for music players. Combines good compression (small files) with high quality. Supported by all browsers.
MP4	.mp4	MP4 is a video format, but can also be used for audio. MP4 video is the upcoming video format on the internet. This leads to automatic support for MP4 audio by all browsers.

Background Audio:

We can use HTML **<bgsound>** tag to play a soundtrack in the background of our webpage. This tag is supported by Internet Explorer only and most of the other browsers ignore this tag. It downloads and plays an audio file when the host document is first downloaded by the user and displayed. The background sound file also will replay whenever the user refreshes the browser.

This tag is having only two attributes *loop* and *src*. Both these attributes have same meaning as explained above.

Here is a simple example to play a small midi file:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML embed Tag</title>
</head>
<body>
<bgsound src="/html/yourfile.mid">
  <noembed><img src="yourimage.gif" ></noembed>
</bgsound>
</body>
```

```
</html>
```

- **HTML Object tag:**

HTML 4 introduces the **<object>** element, which offers an all-purpose solution to generic object inclusion. The <object> element allows HTML authors to specify everything required by an object for its presentation by a user agent

Here are few examples:

a. ***Example - 1***

We can embed an HTML document in an HTML document itself as follows:

```
<object data="data/test.htm" type="text/html" width="300" height="200">
 alt : <a href="data/test.htm">test.htm</a>
</object>
```

Here *alt* attribute will come into picture if browser does not support *object* tag.

b. ***Example - 2***

We can embed a PDF document in an HTML document as follows:

```
<object data="data/test.pdf" type="application/pdf" width="300" height="200">
 alt : <a href="data/test.pdf">test.htm</a>
</object>
```

c. ***Example - 3***

We can specify some parameters related to the document with the **<param>** tag. Here is an example to embed a wav file:

```
<object data="data/test.wav" type="audio/x-wav" width="200" height="20">
 <param name="src" value="data/test.wav">
 <param name="autoplay" value="false">
 <param name="autoStart" value="0">
 alt : <a href="data/test.wav">test.wav</a>
</object>
```

d. ***Example - 4***

We can add a flash document as follows:

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" id="penguin"
  codebase="someplace/swflash.cab" width="200" height="300">
  <param name="movie" value="flash/penguin.swf" />
  <param name="quality" value="high" />
  <img src="penguin.jpg" width="200" height="300" alt="Penguin" />
</object>
```

e. *Example - 5*

We can add a java applet into HTML document as follows:

```
<object classid="clsid:8ad9c840-044e-11d1-b3e9-00805f499d93"
  width="200" height="200">
  <param name="code" value="applet.class">
</object>
```

The **classid** attribute identifies which version of Java Plug-in to use. We can use the optional *codebase* attribute to specify if and how to download the JRE.

TABLE:

The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells. The HTML tables are created using the **<table>** tag in which the **<tr>** tag is used to create table rows and **<td>** tag is used to create data cells.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Tables</title>
</head>
<body>
<table border="1">
<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>
```

1. Table Heading:

Table heading can be defined using **<th>** tag. This tag will be put to replace <td> tag, which is used to represent actual data cell. Normally we will put our top row as table heading as shown below, otherwise we can use <th> element in any row.

Example

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

```
<head>
<title>HTML Table Header</title>
</head>
<body>
<table border="1">
<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>
```

2. Cellpadding and Cellspacing Attributes:

There are two attribiutes called *cellpadding* and *cellspacing* which we will use to adjust the white space in our 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>
```

3. Colspan and Rowspan Attributes:

We will use **colspan** attribute if we want to merge two or more columns into a single column. Similar way we will use **rowspan** if we 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>
```

4. Tables Backgrounds:

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

- **bgcolor** attribute - We can set background color for whole table or just for one cell.
- **background** attribute - We can set background image for whole table or just for one cell.

We 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>
```

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>
```

5. Table Height and Width:

We can set a table width and height using **width** and **height** attrubutes. We 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>
```

6. 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>
```

7. Table Header, Body, and Footer:

Tables can be divided into three portions: a header, a body, and a foot. 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>
```

8. Nested Tables:

We can use one table inside another table. Not only tables we 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>
```

9. Grouping Section of Table:

- **HTML <colgroup> Tag:**

The <colgroup> tag specifies a group of one or more columns in a table for formatting. The <colgroup> tag is useful for applying styles to entire columns, instead of repeating the styles for each cell, for each row.

Note: The <colgroup> tag must be a child of a <table> element, after any <caption> elements and before any <thead>, <tbody>, <tfoot>, and <tr> elements.

Tip: To define different properties to a column within a <colgroup>, use the <col> tag within the <colgroup> tag.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
</head>
<body>
<table border = "1" cellpadding = "3" cellspacing = "0">
 <colgroup>
  <col span="2" style="background-color:cyan">
   <col style="background-color:yellow">
```

ISBN	Title	Quantity	Price
3476896	Motorola	15	$53
3476897	Samsung	20	$55

```
        <col style="background-color:red">

  </colgroup>
  <tr>
   <th>ISBN</th>
   <th>Title</th>
        <th>Quantity</th>
   <th>Price</th>
  </tr>
  <tr>
   <td>3476896</td>
   <td>Motorola</td>
        <td>15</td>
   <td>$53</td>
  </tr>
   <tr>
   <td>3476897</td>
   <td>Samsung</td>
        <td>20</td>
   <td>$55</td>
  </tr>
 </table>
 </body>
 </html>
```

- **HTML <thead> Tag:**

The <thead> tag is used to group header content in an HTML table. The <thead> element is used in conjunction with the <tbody> and <tfoot>elements to specify each part of a table (header, body, footer).

Browsers can use these elements to enable scrolling of the table body independently of the header and footer. Also, when printing a large table that spans multiple pages, these elements can enable the table header and footer to be printed at the top and bottom of each page.

The <thead> tag must be used in the following context: As a child of a <table> element, after any <caption>, and <colgroup> elements, and before any <tbody>, <tfoot>, and <tr> elements.

Example:

Month	Savings
January	$100
February	$80
Sum	$180

```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
</head>
<body>
<table border = "1" cellpadding = "3" cellspacing = "0">
 <colgroup>
  <col style="background-color:green">
```

```
        <col style="background-color:red">
  </colgroup>
  <thead>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
 </thead>
 <tfoot>
  <tr>
    <td>Sum</td>
    <td>$180</td>
  </tr>
 </tfoot>
 <tbody>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$80</td>
  </tr>
 </tbody>
</table>
</body>
</html>
```

10. Accessible Table:

Data tables are used to organize data with a logical relationship in grids. Accessible tables need HTML markup that indicates header cells and data cells, and defines their relationship. Assistive technologies use this information to provide context to users.

To make tables accessible, header cells must be marked up with <th>, and data cells with <td>. For more complex tables, explicit associations may be needed using scope, id, and headers attributes.

Using id and headers attributes to associate data cells with header cells in data tables

The objective of this technique is to associate each data cell (in a data table) with the appropriate headers. This technique adds a headers attribute to each data cell (td element). It also adds an **id** attribute to any cell used as a header for other cells. The headers attribute of a cell contains a list of the id attributes of the associated header cells. If there is more than one id, they are separated by **spaces**.

This technique is used when data cells are associated with more than one row and/or one column header. This allows screen readers to speak the headers associated with each data cell when the relationships are too complex to be identified using the **th** element alone or

the th element with the scope attribute. Using this technique also makes these complex relationships perceivable when the presentation format changes.

This technique is not recommended for layout tables since its use implies a relationship between cells that is not meaningful when tables are used for layout.

Example:

Homework	Exams			Projects		
	1	2	Final	1	2	Final
15%	15%	15%	20%	10%	10%	15%

```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
<style>
th{
        background-color:lime;
}
td{
        background-color:cyan;
}
</style>
</head>
<body>
<table border = "1" cellspacing = "0">
<table>
  <tr>
   <th rowspan="2" id="h">Homework</th>
   <th colspan="3" id="e">Exams</th>
   <th colspan="3" id="p">Projects</th>
  </tr>
  <tr>
   <th id="e1" headers="e">1</th>
   <th id="e2" headers="e">2</th>
   <th id="ef" headers="e">Final</th>
   <th id="p1" headers="p">1</th>
   <th id="p2" headers="p">2</th>
   <th id="pf" headers="p">Final</th>
  </tr>
  <tr>
   <td headers="h">15%</td>
   <td headers="e e1">15%</td>
   <td headers="e e2">15%</td>
   <td headers="e ef">20%</td>
   <td headers="p p1">10%</td>
   <td headers="p p2">10%</td>
   <td headers="p pf">15%</td>
  </tr>
 </table>
</body>
```

```
</html>
```

Using the scope attribute to associate header cells and data cells in data tables

The objective of this technique is to associate header cells with data cells in complex tables using the scope attribute. The scope attribute may be used to clarify the scope of any cell used as a header. The scope identifies whether the cell is a header for a row, column, or group of rows or columns. The values row, col, rowgroup, and colgroup identify these possible scopes respectively.

For simple data tables where the header is not in the first row or column, like the one in Example 1, this technique can be used. Based on screen reader support today, its use is suggested in two situations both relating to simple tables:

- data cells marked up with td that also function as row header or column header
- header cells marked up with td instead of th. Sometimes, authors use this to avoid the display characteristics associated with th and also do not choose to use CSS to control the display for th.

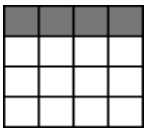
Example:

Contact Information

	Name	Phone	Fax	City
1.	Joel Garner	412-212-5421	412-212-5400	Pittsburgh
2.	Clive Lloyd	410-306-1420	410-306-5400	Baltimore
3.	Gordon Greenidge	281-564-6720	281-511-6600	Houston

```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
<style>
th{
        background-color:lime;
}
td{
        background-color:cyan;
}
</style>
</head>
<body>
<table border = "1" cellspacing = "0">
<caption>Contact Information</caption>
 <tr>
  <td></td>
  <th scope="col">Name</th>
  <th scope="col">Phone</th>
  <th scope="col">Fax</th>
  <th scope="col">City</th>
 </tr><tr>
  <td>1.</td>
  <th scope="row">Joel Garner</th>
  <td>412-212-5421</td>
  <td>412-212-5400</td>
  <td>Pittsburgh</td>
```

```
 </tr><tr>
  <td>2.</td>
  <th scope="row">Clive Lloyd</th>
  <td>410-306-1420</td>
  <td>410-306-5400</td>
  <td>Baltimore</td>
 </tr><tr>
  <td>3.</td>
  <th scope="row">Gordon Greenidge</th>
  <td>281-564-6720</td>
  <td>281-511-6600</td>
  <td>Houston</td>
 </tr>
</table>
</body>
</html>
```



Tables with one header for rows or columns:

For tables with content that is easy to distinguish, mark up header cells with <th> and data cells with <td> elements.

Example 1: Table with header cells in the top row only

```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
<style>
.head{
        background-color:wheat;
}
</style>
</head>
<body>
<table border = "1" cellpadding = "3" cellspacing = "0">
<caption>
<p>Phone Detail</p>
</caption>
 <colgroup>
  <col span="2" style="background-color:cyan">
   <col style="background-color:yellow">
        <col style="background-color:red">
 </colgroup>
 <tr class = "head">
  <th>ISBN</th>
```

Phone Detail

ISBN	Title	Quantity	Price
3476896	Motorola	15	$53
3476897	Samsung	20	$55

```
  <th>Title</th>
        <th>Quantity</th>
  <th>Price</th>
 </tr>
 <tr>
  <td>3476896</td>
  <td>Motorola</td>
        <td>15</td>
  <td>$53</td>
 </tr>
  <tr>
  <td>3476897</td>
  <td>Samsung</td>
        <td>20</td>
  <td>$55</td>
 </tr>
</table>
</body>
</html>
```

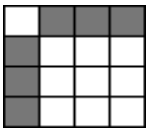
Example 2: Table with header cells in the first column only

```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
</head>
<body>
<table border = "1" cellpadding = "3" cellspacing = "1">
<caption>
<p>Phone Detail</p>
</caption>
 <colgroup>
  <col style="background-color:cyan">
  <col span = "2" style="background-color:lime">
 </colgroup>
 <tr>
  <th>ISBN</th>
        <td>3476896</td>
        <td>3476897</td>
</tr>
<tr>
  <th>Title</th>
        <td>Motorola</td>
        <td>Samsung</td>
</tr>
<tr>
```

Phone Detail

ISBN	3476896	3476897
Title	Motorola	Samsung
Quantity	15	20
Price	$53	$55

```
        <th>Quantity</th>
        <td>15</td>
        <td>20</td>
</tr>
 <tr>
  <th>Price</th>
  <td>$53</td>
        <td>$55</td>
 </tr>
</table>
</body>
</html>
```



Tables with two headers have a simple row header and a simple column header:

For tables with unclear header directions, define the direction of each header by setting the scope attribute to col or row.

Example 1: Table with header cells in the top row and first column

```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
<style>
.head{
        background-color:red;
}
</style>
</head>
<body>
<table border = "1" cellpadding = "3" cellspacing = "0">
<colgroup>
<col span = "6" style = "background-color:cyan"
</colgroup>
<caption>
<p><mark>Time Table</mark></p>
</caption>
<tr class = "head">
                <th>Days/Time</th>
                <th>Monday</th>
                <th>Tuesday</th>
                <th>Wednesday</th>
                <th>Thursday</th>
                <th>Friday</th>
</tr>
        <tr>
```

Time Table

Days/Time	Monday	Tuesday	Wednesday	Thursday	Friday
09:00 - 11:00	Closed	Open	Open	Closed	Closed
11:00 - 13:00	Open	Open	Closed	Closed	Closed
13:00 - 15:00	Open	Open	Open	Closed	Closed
15:00 - 17:00	Closed	Closed	Closed	Open	Open

```
            <th class = "head">09:00 - 11:00</th>
            <td>Closed</td>
            <td>Open</td>
            <td>Open</td>
            <td>Closed</td>
            <td>Closed</td>
        </tr>
        <tr>
            <th class = "head">11:00 - 13:00</th>
            <td>Open</td>
            <td>Open</td>
            <td>Closed</td>
            <td>Closed</td>
            <td>Closed</td>
        </tr>
        <tr>
            <th class = "head">13:00 - 15:00</th>
            <td>Open</td>
            <td>Open</td>
            <td>Open</td>
            <td>Closed</td>
            <td>Closed</td>
        </tr>
        <tr>
            <th class = "head">15:00 - 17:00</th>
            <td>Closed</td>
            <td>Closed</td>
            <td>Closed</td>
            <td>Open</td>
            <td>Open</td>
        </tr>
</table>
</body>
</html>
```

Example 2: Table with an offset column of header cells

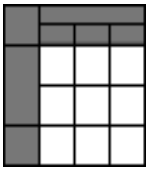
```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
<style>
th{
        background-color:yellow;
}
td{
        background-color:cyan;
```

Holidays taken in the last six months

ID	Name	July	August	September	October	November	December
215	Abel	5	2	0	0	0	3
231	Annette	0	5	3	0	0	6
173	Bernard	2	0	0	5	0	0
141	Gerald	0	10	0	0	0	8
99	Michael	8	8	8	8	0	4

```
}
</style>
</head>
<body>
<table border = "1" cellspacing = "0">
 <caption>
  Holidays taken in the last six months
 </caption >
 <thead>
  <tr>
   <th scope="col"><abbr title="Identification Number">ID</abbr></th>
   <th scope="col">Name</th>
   <th scope="col">July</th>
   <th scope="col">August</th>
   <th scope="col">September</th>
   <th scope="col">October</th>
   <th scope="col">November</th>
   <th scope="col">December</th>
  </tr>
 </thead>

 <tbody>
  <tr>
   <td>215</td>
   <th scope="row">Abel</th>
   <td>5</td>
   <td>2</td>
   <td>0</td>
   <td>0</td>
   <td>0</td>
   <td>3</td>
  </tr>

  <tr>
   <td>231</td>
   <th scope="row">Annette </th>
   <td>0</td>
   <td>5</td>
   <td>3</td>
   <td>0</td>
   <td>0</td>
   <td>6</td>
  </tr>

  <tr>
   <td>173</td>
```

```
    <th scope="row">Bernard</th>
    <td>2</td>
    <td>0</td>
    <td>0</td>
    <td>5</td>
    <td>0</td>
    <td>0</td>
   </tr>

   <tr>
    <td>141</td>
    <th scope="row">Gerald</th>
    <td>0</td>
    <td>10</td>
    <td>0</td>
    <td>0</td>
    <td>0</td>
    <td>8</td>
   </tr>

   <tr>
    <td>99</td>
    <th scope="row">Michael</th>
    <td>8</td>
    <td>8</td>
    <td>8</td>
    <td>8</td>
    <td>0</td>
    <td>4</td>
   </tr>
  </tbody>
 </table>
 </body>
 </html>
```



Tables with irregular headers have header cells that span multiple columns and/or rows:

For these tables, define column and row groups and set the range of the header cells using the colgroup and rowgroup values of the scope attribute.

Example 1: Table with two tier headers

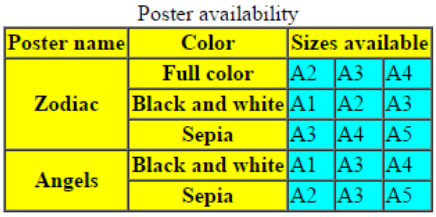
```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
```

Inventory

	Mars		Venus	
	Produced	Sold	Produced	Sold
Teddy Bears	50,000	30,000	100,000	80,000
Board Games	10,000	5,000	12,000	9,000

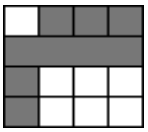
```
<style>
th{
	background-color:yellow;
}
td{
	background-color:cyan;
}
</style>
</head>
<body>
<table border = "1" cellspacing = "0">
 <caption>  Inventory  </caption >
 <colgroup span="2"></colgroup>
 <colgroup span="2"></colgroup>
 <tr>
  <td rowspan="2"></td>
  <th colspan="2" scope="colgroup">Mars</th>
  <th colspan="2" scope="colgroup">Venus</th>
 </tr>
 <tr>
  <th scope="col">Produced</th>
  <th scope="col">Sold</th>
  <th scope="col">Produced</th>
  <th scope="col">Sold</th>
 </tr>
 <tr>
  <th scope="row">Teddy Bears</th>
  <td>50,000</td>
  <td>30,000</td>
  <td>100,000</td>
  <td>80,000</td>
 </tr>
 <tr>
  <th scope="row">Board Games</th>
  <td>10,000</td>
  <td>5,000</td>
  <td>12,000</td>
  <td>9,000</td>
 </tr>
</table>
</body>
</html>
```

Example 2: Table with headers spanning multiple rows or columns




```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
<style>
th{
        background-color:yellow;
}
td{
        background-color:cyan;
}
</style>
</head>
<body>
<table border = "1" cellspacing = "0">
 <caption>
  Poster availability
 </caption>
 <col>
 <col>
 <colgroup span="3"></colgroup>
 <thead>
  <tr>
   <th scope="col">Poster name</th>
   <th scope="col">Color</th>
   <th colspan="3" scope="colgroup">Sizes available</th>
  </tr>
 </thead>
 <tbody>
  <tr>
   <th rowspan="3" scope="rowgroup">Zodiac</th>
   <th scope="row">Full color</th>
   <td>A2</td>
   <td>A3</td>
   <td>A4</td>
  </tr>
  <tr>
   <th scope="row">Black and white</th>
   <td>A1</td>
   <td>A2</td>
   <td>A3</td>
  </tr>
  <tr>
   <th scope="row">Sepia</th>
   <td>A3</td>
```

```
    <td>A4</td>
    <td>A5</td>
   </tr>
  </tbody>
  <tbody>
   <tr>
    <th rowspan="2" scope="rowgroup">Angels</th>
    <th scope="row">Black and white</th>
    <td>A1</td>
    <td>A3</td>
    <td>A4</td>
   </tr>
   <tr>
    <th scope="row">Sepia</th>
    <td>A2</td>
    <td>A3</td>
    <td>A5</td>
   </tr>
  </tbody>
</table>
</body>
</html>
```



Tables with multi-level headers have multiple header cells associated per data cell: For tables that are so complex that header cells can't be associated in a strictly horizontal or vertical way, use id and headers attributes to explicitly associate header and data cells.

Example 1: Table with multiple column headers in each column

```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
<style>
th{
        background-color:yellow;
}
td{
        background-color:cyan;
}
</style>
</head>
<body>
<table border = "1" cellspacing = "0">
 <caption>
   Supplier contacts
```

Supplier contacts

	Example 1 Ltd	Example 2 Co
Contact	James Phillips	Marie Beauchamp
Position	Sales Director	Sales Manager
Email	jp@1ltd.example.com	marie@2co.example.com
	Example 3 Ltd	Example 4 Inc
Contact	Suzette Jones	Alex Howe
Position	Sales Officer	Sales Director
Email	Suz@ltd3.example.com	howe@4inc.example.com

```
 </caption>
 <tr>
  <td id="blank">&amp;nbsp;</td>
  <th id="co1" headers="blank">Example 1 Ltd</th>
  <th id="co2" headers="blank">Example 2 Co</th>
 </tr>
 <tr>
  <th id="c1" headers="blank">Contact</th>
  <td headers="co1 c1">James Phillips</td>
  <td headers="co2 c1">Marie Beauchamp</td>
 </tr>
 <tr>
  <th id="p1"  headers="blank">Position</th>
  <td headers="co1 p1">Sales Director</td>
  <td headers="co2 p1">Sales Manager</td>
 </tr>
 <tr>
  <th id="e1"  headers="blank">Email</th>
  <td headers="co1 e1">jp@1ltd.example.com</td>
  <td headers="co2 e1">marie@2co.example.com</td>
 </tr>
 <tr>
  <td>&amp;nbsp;</td>
  <th id="co3" headers="blank">Example 3 Ltd</th>
  <th id="co4" headers="blank">Example 4 Inc</th>
 </tr>
 <tr>
  <th id="c2"  headers="blank">Contact</th>
  <td headers="co3 c2">Suzette Jones</td>
  <td headers="co4 c2">Alex Howe</td>
 </tr>
 <tr>
  <th id="p2" headers="blank">Position</th>
  <td headers="co3 p2">Sales Officer</td>
  <td headers="co4 p2">Sales Director</td>
 </tr>
 <tr>
  <th id="e2" headers="blank">Email</th>
  <td headers="co3 e2">Suz@ltd3.example.com</td>
  <td headers="co4 e2">howe@4inc.example.com</td>
 </tr>
</table>
</body>
</html>
```

Example 2: Table with three headers related to each data cell

```
<!DOCTYPE html>
<html>
<head>
<title>Table Example</title>
<style>
th{
        background-color:yellow;
}
td{
        background-color:cyan;
}
</style>
</head>
<body>
<table border = "1" cellspacing = "0">
<caption>
  Availability of holiday accommodation
</caption>
<thead>
  <tr>
    <td></td>
    <th id="stud" scope="col">
      Studio
    </th>
    <th id="apt" scope="col">
      <abbr title="Apartment">Apt</abbr>
    </th>
    <th id="chal" scope="col">
      Chalet
    </th>
    <th id="villa" scope="col">
      Villa
    </th>
  </tr>
</thead>
<tbody>
  <tr>
    <th id="par" class="span" colspan="5" scope="colgroup">
      Paris
    </th>
  </tr>
  <tr>
    <th headers="par" id="pbed1">
      1 bedroom
    </th>
```

Availability of holiday accommodation

	Studio	Apt	Chalet	Villa
Paris				
1 bedroom	11	20	25	23
2 bedroom	-	43	52	32
3 bedroom	-	13	15	40
Rome				
1 bedroom	13	21	22	3
2 bedroom	-	23	43	30
3 bedroom	-	16	32	40

```
    <td headers="par pbed1 stud">
      11
    </td>
    <td headers="par pbed1 apt">
      20
    </td>
    <td headers="par pbed1 chal">
      25
    </td>
    <td headers="par pbed1 villa">
      23
    </td>
  </tr>
  <tr>
    <th headers="par" id="pbed2">
      2 bedroom
    </th>
    <td headers="par pbed2 stud">
      -
    </td>
    <td headers="par pbed2 apt">
      43
    </td>
    <td headers="par pbed2 chal">
      52
    </td>
    <td headers="par pbed2 villa">
      32
    </td>
  </tr>
  <tr>
    <th headers="par" id="pbed3">
      3 bedroom
    </th>
    <td headers="par pbed3 stud">
      -
    </td>
    <td headers="par pbed3 apt">
      13
    </td>
    <td headers="par pbed3 chal">
      15
    </td>
    <td headers="par pbed3 villa">
      40
    </td>
```

```
</tr>
<tr>
  <th id="rome" class="span" colspan="5" scope="colgroup">
    Rome
  </th>
</tr>
<tr>
  <th id="rbed1" headers="rome">
    1 bedroom
  </th>
  <td headers="rome rbed1 stud">
    13
  </td>
  <td headers="rome rbed1 apt">
    21
  </td>
  <td headers="rome rbed1 chal">
    22
  </td>
  <td headers="rome rbed1 villa">
    3
  </td>
</tr>
<tr>
  <th id="rbed2" headers="rome">
    2 bedroom
  </th>
  <td headers="rome rbed2 stud">
    -
  </td>
  <td headers="rome rbed2 apt">
    23
  </td>
  <td headers="rome rbed2 chal">
    43
  </td>
  <td headers="rome rbed2 villa">
    30
  </td>
</tr>
<tr>
  <th id="rbed3" headers="rome">
    3 bedroom
  </th>
  <td headers="rome rbed3 stud">
    -
```

```
        </td>
        <td headers="rome rbed3 apt">
          16
        </td>
        <td headers="rome rbed3 chal">
          32
        </td>
        <td headers="rome rbed3 villa">
          40
        </td>
      </tr>
    </tbody>
    </table>
    </body>
    </html>
```

Caption & Summary: A caption identifies the overall topic of a table and is useful in most situations. A summary provides orientation or navigation hints in complex tables.

Some document formats other than HTML, such as PDF, provide similar mechanisms to markup table structures. Word processing applications may also provide mechanisms to markup tables. Table's markup is often lost when converting from one format to another, though some programs may provide functionality to assist converting table markup.

Many web authoring tools and content management systems (CMS) provide functions to define header cells during table creation without having to manually edit the code.

- **Why is this important?**

Tables without structural markup to differentiate and properly link between header and data cells, create accessibility barriers. Relying on visual cues alone is not sufficient to create an accessible table. With structural markup, headers and data cells can be programmatically determined by software, which means that:

- **People using screen readers** can have the row and column headers read aloud as they navigate through the table. Screen readers speak one cell at a time and reference the associated header cells, so the reader doesn't lose context.
- **Some people use alternative ways to render the data**, for example by using custom stylesheets to display header cells more prominently. Techniques like this enable them to change text size and colors, and display the information as lists rather than grids. In order to allow alternative renderings, table code needs to be properly structured.

FORM:

1. Introduction Form:

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, 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>
```

2. 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 our 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	We 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.

3. HTML Form Controls:

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

Text Input Controls:

There are three types of text input used on forms:

a. 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.

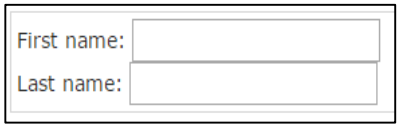
Attributes:

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.

Example:

```
<!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>
```



b. Password Input Controls:

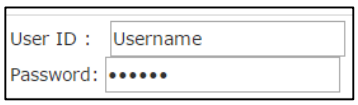
This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input> tag but type attribute is set to **password**.

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.

Example:




```
<!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>
```

c. Multiple-Line Text Input Controls:

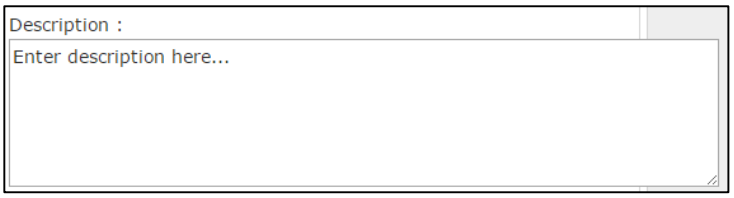
This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

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

Example:




```
<!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>
```

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**.

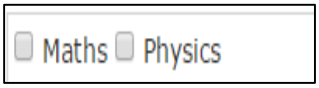
Attributes

Following is the list of attributes for <checkbox> tag.

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.
Id	provide a unique identifier for the form so it can be more easily styled using CSS

Example:

```
<!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>
```



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**.

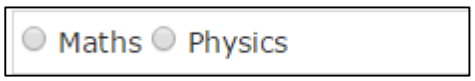
Attributes

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.

id	provide a unique identifier for the form so it can be more easily styled using CSS

Example:




```
<!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>
```

- **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.

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 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

Example:


```
<!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>
```

- **File Upload Box:**

If we want to upload a file to our web site, we 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**.

Attributes

Following is the list of important attributes of file upload box:

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
accept	Specifies the types of files that the server accepts.

Example:

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

- **Button Controls:**

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

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.

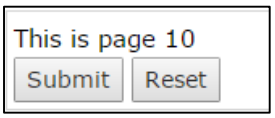
Example:


```
<!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>
```

- **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 has be displayed next based on the passed current page.

Example:




```
<!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>
```

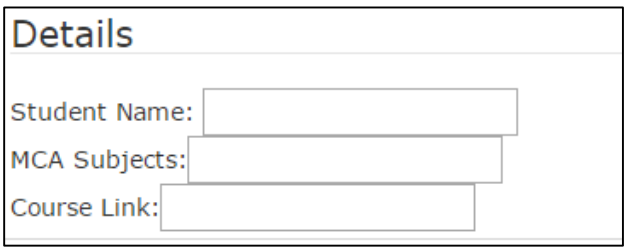
4. Fieldset and Legend:

The HTML <fieldset> tag is used for grouping related form elements. By using the fieldset tag and the legend tag, we can make our forms much easier to understand for our users.

The HTML <legend> tag s used to define a caption for <fieldset> tag.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML fieldset Tag</title>
</head>
<body>
<form>
 <fieldset>
  <legend>Details</legend>
  Student Name: <input type="text"><br />
  MCA Subjects:<input type="text"><br />
  Course Link:<input type="url" name="websitelink">
 </fieldset>
</form>
</body>
</html>
```



- **Global Attributes:**

Not valid in base, head, html, meta, param, script, style, and title elements.

Attribute	HTML-5	Description
accesskey		Specifies a shortcut key for an element to be used in place of keyboard.
class		The class of the element
contenteditable	Yes	Specify whether the element is editable or not.
contextmenu	Yes	Specifies a context menu for an element.
data-*	Yes	Used to store custom data associated with the element.
draggable	Yes	Boolean attribute to specify whether the element can be dragged or not.
dropzone	Yes	Specifies whether the dragged data is copied, moved, or linked, when dropped.
hidden	Yes	Specifies whether element should be visible or not.
id		A unique id for the element
spellcheck	Yes	Specifies if the element must have it's spelling or grammar checked.
style		An inline style definition
tabindex		Specifies the tab order of an element.
title		A text to display in a tool tip
translate	Yes	Boolean attribute specifies whether the content of an element should be translated or not

Language Attributes:

The **lang** attribute indicates the language being used for the enclosed content. The language is identified using the ISO standard language abbreviations, such as **fr** for **French**, **en** for **English**, and so on. RFC 1766 (https://www.ietf.org/rfc/rfc1766.txt) describes these codes and their formats.

Not valid in base, br, frame, frameset, hr, iframe, param, and script elements.

Attribute	Value	Description
dir	ltr | rtl	Sets the text direction
lang	language_code	Sets the language code

Window Events Attributes:

Following events have been introduced in older versions of HTML but all the tags marked with  are part of HTML-5.

Events	HTML-5	Description
onafterprint		Triggers after a document is printed
onbeforeprint		Triggers before a document is printed
onbeforeonload		Triggers before a document loads
onerror		Triggers when an error occurs
onhaschange		Triggers when a document has changed
onload		Triggers when a document loads
onmessage		Triggers when a message is triggered
onoffline		Triggers when a document goes offline
ononline		Triggers when a document comes online
onpagehide		Triggers when a window is hidden
onpageshow		Triggers when a window becomes visible
onpopstate		Triggers when a window's history changes
onredo		Triggers when a document performs a redo
onresize		Triggers when a window is resized
onstorage		Triggers when a document loads
onundo		Triggers when a document performs an undo
onunload		Triggers when a user leaves the document

Form Events:

Following tags have been introduced in older versions of HTML but all the tags marked with  are part of HTML-5.

Events	HTML-5	Description
onblur		Triggers when a window loses focus
onchange		Triggers when an element changes
oncontextmenu		Triggers when a context menu is triggered
onfocus		Triggers when a window gets focus
onformchange		Triggers when a form changes
onforminput		Triggers when a form gets user input

oninput		Triggers when an element gets user input
oninvalid		Triggers when an element is invalid
onreset		Triggers when a form is reset
onselect		Triggers when an element is selected
onsubmit		Triggers when a form is submitted

- **Keyboard Events:**

Events	HTML-5	Description
onkeydown		Triggers when a key is pressed
onkeypress		Triggers when a key is pressed and released
onkeyup		Triggers when a key is released

- **Mouse Events:**

Following tags have been introduced in older versions of HTML but all the tags marked with  are part of HTML-5.

Events	HTML-5	Description
onclick		Triggers on a mouse click
ondblclick		Triggers on a mouse double-click
ondrag		Triggers when an element is dragged
ondragend		Triggers at the end of a drag operation
ondragenter		Triggers when an element has been dragged to a valid drop target
ondragleave		Triggers when an element leaves a valid drop target
ondragover		Triggers when an element is being dragged over a valid drop target
ondragstart		Triggers at the start of a drag operation
ondrop		Triggers when a dragged element is being dropped
onmousedown		Triggers when a mouse button is pressed
onmousemove		Triggers when the mouse pointer moves
onmouseout		Triggers when the mouse pointer moves out of an element
onmouseover		Triggers when the mouse pointer moves over an element
onmouseup		Triggers when a mouse button is released
onmousewheel		Triggers when the mouse wheel is being rotated
onscroll		Triggers when an element's scrollbar is being scrolled

- **Media Events:**

Following tags have been introduced in older versions of HTML but all the tags marked with  are part of HTML-5.

Events	HTML-5	Description
onabort		Triggers on an abort event
oncanplay		Triggers when a media can start play, but might has to stop for buffering

oncanplaythrough		Triggers when a media can be played to the end, without stopping for buffering
ondurationchange		Triggers when the length of a media is changed
onemptied		Triggers when a media resource element suddenly becomes empty.
onended		Triggers when a media has reached the end
onerror		Triggers when an error occurs
onloadeddata		Triggers when media data is loaded
onloadedmetadata		Triggers when the duration and other media data of a media element is loaded
onloadstart		Triggers when the browser starts loading the media data
onpause		Triggers when media data is paused
onplay		Triggers when media data is going to start playing
onplaying		Triggers when media data has started playing
onprogress		Triggers when the browser is fetching the media data
onratechange		Triggers when the playing rate of media data has changed
onreadystatechange		Triggers when the ready-state changes
onseeked		Triggers when the seeking attribute of a media element is no longer true, and the seeking has ended
onseeking		Triggers when the seeking attribute of a media element is true, and the seeking has begun
onstalled		Triggers when there is an error in fetching media data
onsuspend		Triggers when the browser has been fetching media data, but stopped before the entire media file was fetched
ontimeupdate		Triggers when media changes its playing position
onvolumechange		Triggers when a media changes the volume, also when volume is set to "mute"
onwaiting		Triggers when media has stopped playing, but is expected to resume

5. Tab Index Attribute:

The tabindex attribute specifies the tab order of an element (when the “tab” button is used for navigating).

For example:

```
<a href = “http://www.w3school.com/” tabindex = “2”>W3School</a>
<a href = “http://www.google.com/” tabindex = “1”>Google</a>
<a href = “http://www.microsoft.com/” tabindex = “3”>Microsoft</a>
```

6. Sending Form Data to the Server:

There are two ways the browser client can send information to the web server.

1. The GET Method
2. The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

name1=value1&name2=value2&name3=value3

Spaces are removed and replaced with the + character and any other nonalphanumeric characters are replaced with a hexadecimal values. After the information is encoded it is sent to the server.

1. **The GET Method:**

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.

http://www.test.com/index.htm?name1=value1&name2=value2

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides **$_GET** associative array to access all the sent information using GET method.

Example:

```
<?php
  if( $_GET["name"] || $_GET["age"] ) {
     echo "Welcome ". $_GET['name']. "<br />";
     echo "You are ". $_GET['age']. " years old.";

     exit();
  }
?>
<html>
  <body>
     <form action = "<?php $_PHP_SELF ?>" method = "GET">
        Name: <input type = "text" name = "name" />
        Age: <input type = "text" name = "age" />
        <input type = "submit" />
     </form>
  </body>
</html>
```

2. The POST Method:

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using POST method.

Example:

```
<?php
  if( $_POST["name"] || $_POST["age"] ) {
    if (preg_match("/[^A-Za-z'-]/",$_POST['name'] )) {
      die ("invalid name and name should be alpha");
    }
    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['age']. " years old.";
    exit();
  }
?>
<html>
  <body>
    <form action = "<?php $_PHP_SELF ?>" method = "POST">
      Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" />
    </form>
  </body>
</html>
```

FRAMES:

HTML frames are used to divide our 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.

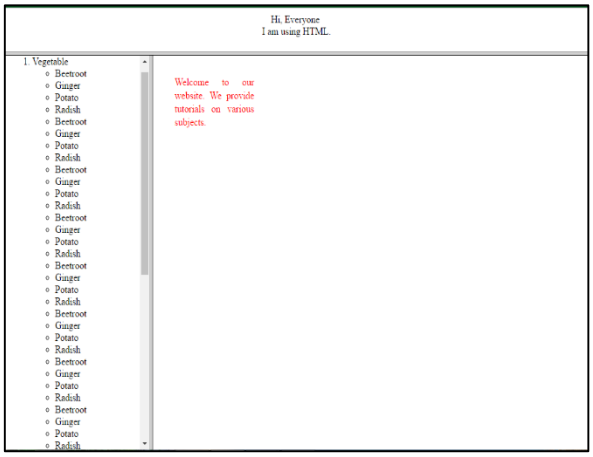
- The browser's *back button* might not work as the user hopes.
- There are still few browsers that do not support frame technology.

1. 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:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Frames</title>
</head>
<frameset rows="10%,*">
  <frame name="top" src="/HTML Programs/FirstProgram.html" />
  <frameset cols = "25%,*">
  <frame name="main" src="/HTML Programs/NestedList.html" />
  <frame name="bottom" src="/HTML Programs/Font.html" />
  </frameset>
  <noframes>
  <body>
    Your browser does not support frames.
  </body>
  </noframes>
</frameset>
</html>
```



2. 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. We 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 to create 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. We 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%"*. We 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.

3. 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 us 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 we want to create links in one frame that load pages into an 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 we 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"

4. 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 we 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.

We can put some nice message for our user having old browsers. For example *Sorry!! your browser does not support frames.*

5. 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 we click any of the links in menubar, available link will open in main_page.

Following is the content of menu.htm file

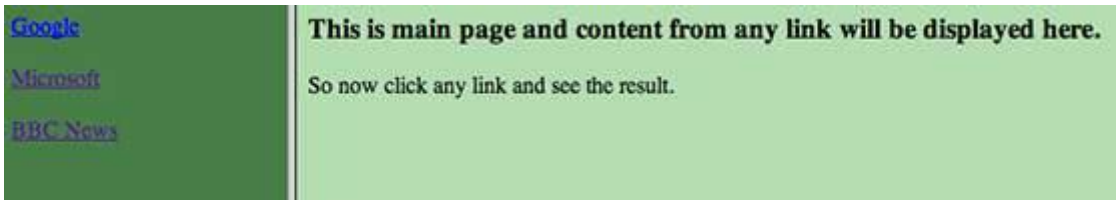
```
<!DOCTYPE html>
```

```
<html>
<body bgcolor="#4a7d49">
<a href="https://www.google.com" target="main_page">Google</a>
<br /><br />
<a href="https://www.microsoft.com" target="main_page">Microsoft</a>
<br /><br />
<a href="https://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 we 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.opening a new 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.

6. IFRAME:

We 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:


```
<!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>
```

7. 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.

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 avalaible 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 an 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"

EXPLORING NEW ELEMENTS OF HTML5:

HTML5 is the latest and most enhanced version of HTML. Technically, HTML is not a programming language, but rather a markup language.

HTML5 is the next major revision of the HTML standard superseding HTML 4.01, XHTML 1.0, and XHTML 1.1. HTML5 is a standard for structuring and presenting content on the World Wide Web.

HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).

The new standard incorporates features like video playback and drag-and-drop that have been previously dependent on third-party browser plug-ins such as Adobe Flash, Microsoft Silverlight, and Google Gears.

1. Browser Support:

The latest versions of Apple Safari, Google Chrome, Mozilla Firefox, and Opera all support many HTML5 features and Internet Explorer 9.0 will also have support for some HTML5 functionality.

The mobile web browsers that come pre-installed on iPhones, iPads, and Android phones all have excellent support for HTML5.

2. New Features:

HTML5 introduces a number of new elements and attributes that helps in building a modern website. Following are great features introduced in HTML5.

- **New Semantic Elements**: These are like <header>, <footer>, and <section>.
- **Forms 2.0**: Improvements to HTML web forms where new attributes have been introduced for <input> tag.
- **Persistent Local Storage**: To achieve without resorting to third-party plugins.
- **WebSocket**: A next-generation bidirectional communication technology for web applications.
- **Server-Sent Events**: HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).
- **Canvas**: This supports a two-dimensional drawing surface that you can program with JavaScript.
- **Audio & Video**: You can embed audio or video on your web pages without resorting to third-party plugins.
- **Geolocation**: Now visitors can choose to share their physical location with your web application.
- **Microdata**: This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.

- **Drag and drop**: Drag and drop the items from one location to another location on a the same webpage.

3. Backward Compatibility:

HTML5 is designed, as much as possible, to be backward compatible with existing web browsers. New features build on existing features and allow us to provide fallback content for older browsers.

It is suggested to detect support for individual HTML5 features using a few lines of JavaScript.

4. HTML5 Document:

The following tags have been introduced for better structure –

- **Section**: This tag represents a generic document or application section. It can be used together with h1-h6 to indicate the document structure.
- **Article**: This tag represents an independent piece of content of a document, such as a blog entry or newspaper article.
- **Aside**: This tag represents a piece of content that is only slightly related to the rest of the page.
- **Header**: This tag represents the header of a section.
- **Footer:** This tag represents a footer for a section and can contain information about the author, copyright information, etc.
- **Nav**: This tag represents a section of the document intended for navigation.
- **Dialog**: This tag can be used to mark up a conversation.
- **Figure**: This tag can be used to associate a caption together with some embedded content, such as a graphic or video.

5. Deprecated Tags:

The following elements are not available in HTML5 anymore and their function is better handled by CSS:

Tags (Elements)	Description
<acronym>	Defines an acronym
<applet>	Defines an applet
<basefont>	Defines an base font for the page.
<big>	Defines big text
** **	Defines centered text
<dir>	Defines a directory list
<font>	Defines text font, size, and color
<frame>	Defines a frame
<frameset>	Defines a set of frames
<isindex>	Defines a single-line input field
<noframes>	Defines a noframe section
<s>	Defines strikethrough text
<strike>	Defines strikethrough text
<tt>	Defines teletype text
<u>	Defines underlined text

6. Deprecated Attributes:

HTML5 has none of the presentational attributes that were in HTML4 as their functions are better handled by CSS. Some attributes from HTML4 are no longer allowed in HTML5 at all and they have been removed completely.

Following is the table having removed attributed and their corresponding impacted tags (elements) ie. elements from which those attributes have been removed permanently

Removed Attributes	From the Elements
rev	link, a
charset	link and a
shape	a
coords	a
longdesc	img and iframe.
target	link
nohref	area
profile	head
version	html
name	img
scheme	meta
archive	object
classid	object
codebase	object
codetype	object
declare	object
standby	object
valuetype	param
type	param
axis	td and t
abbr	td and t
scope	td
align	caption, iframe, img, input, object, legend, table, hr, div, h1, h2, h3, h4, h5, h6, p, col, colgroup, tbody, td, tfoot, th, thead and tr.
alink	body
link	body
vlink	body
text	body
background	body
bgcolor	table, tr, td, th and body.
border	table and object.
cellpadding	table
cellspacing	table
char	col, colgroup, tbody, td, tfoot, th, thead and tr.
charoff	col, colgroup, tbody, td, tfoot, th, thead and tr.
clear	br
compact	dl, menu, ol and ul.
frame	table

compact	dl, menu, ol and ul.
frame	table
frameborder	iframe
hspace	img and object.
vspace	img and object.
marginheight	iframe
marginwidth	iframe
noshade	hr
nowrap	td and th
rules	table
scrolling	iframe
size	hr
type	li, ol and ul.
valign	col, colgroup, tbody, td, tfoot, th, thead and tr
width	hr, table, td, th, col, colgroup and pre.

7. HTML5 - New Tags (Elements):

The following tags (elements) have been introduced in HTML5.

Tags (Elements)	Description
<article>	Represents an independent piece of content of a document, such as a blog entry or newspaper article
<aside >	Represents a piece of content that is only slightly related to the rest of the page.
<audio>	Defines an audio file.
<canvas>	This is used for rendering dynamic bitmap graphics on the fly, such as graphs or games.
<command>	Represents a command the user can invoke.
<datalist>	Together with the a new list attribute for input can be used to make comboboxes
<details>	Represents additional information or controls which the user can obtain on demand
<embed>	Defines external interactive content or plugin.
<figure>	Represents a piece of self-contained flow content, typically referenced as a single unit from the main flow of the document.
<footer>	Represents a footer for a section and can contain information about the author, copyright information, et cetera.
<header>	Represents a group of introductory or navigational aids.
<hgroup>	Represents the header of a section.
<keygen>	Represents control for key pair generation.
<mark>	Represents a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.
<meter>	Represents a measurement, such as disk usage.
<nav>	Represents a section of the document intended for navigation.
<output>	Represents some type of output, such as from a calculation done through scripting.

<progress>	Represents a completion of a task, such as downloading or when performing a series of expensive operations.
<ruby>	Together with <rt> and <rp> allow for marking up ruby annotations.
<section>	Represents a generic document or application section
<time>	Represents a date and/or time.
<video>	Defines a video file.
<wbr>	Represents a line break opportunity.

8. New types for <input> tag:

The input element's type attribute now has the following new values.

Type	Description
color	Color selector, which could be represented by a wheel or swatch picker
date	Selector for calendar date
datetime-local	Date and time display, with no setting or indication for time zones
datetime	Full date and time display, including a time zone.
email	Input type should be an email.
month	Selector for a month within a given year
number	A field containing a numeric value only
range	Numeric selector within a range of values, typically visualized as a slider
search	Term to supply to a search engine. For example, the search bar atop a browser.
tel	Input type should be telephone number.
time	Time indicator and selector, with no time zone information
url	Input type should be URL type.
week	Selector for a week within a given year

Example:

```
<!DOCTYPE html>
<html>
<body bgcolor="#b5dcb3">
<form>
    Email: <input type = "email" name = "email" id = "01"/>
    Date: <input type = "date" name = "data" id = "001"/>
    <input type = "submit" name = "submit" value = "submit"/>
</form>
</body>
</html>
```

9. KEYGEN:

The <keygen> tag specifies a key pair generator field used for forms. When the for is submitted, the private key is stored locally, and the public key is sent to the server.

Example:

```
<!DOCTYPE html>
<html>
<body bgcolor="#b5dcb3">
<form action = "mailto:go@gmail.com" method = "get">
Username: <input type = "text" name = "username"/><br/>
Password: <input type = "password" name = "password"/><br/>
Encryption:<keygen name = "security"><br/>
<input type = "submit" name = "submit" value = "submit"/>
</form>
</body>
</html>
```

10. Progress:

The progress element represents the completion progress of a task. A progress must have both a start tag and an end tag, even though it looks like a replaced element like an input. This is good though, as it helps with fallback content as we will cover later.

Apart from the global attributes, it can have two more attributes:

1. **Max:**

Indicates how much task needs to be done before it can be considered as complete. If not specified the default value is 1.0.

2. **Value:**

Indicates the current status of the progress bar. It must be greater than or equal to 0.0 and less than or equal to 1.0 or the value of the max attribute (if present)

Example:

```
<!DOCTYPE html>
<html>
<body bgcolor="#b5dcb3">
<form>
        <progress max = “100” value = “80”></progress>
</form>
</body>
</html>
```

11. Meter:

The html <meter> element represents either a scalar value within a known range or a fractional value. The <meter> tag should not be used to indicate progress (as in progress bar). For progress bars, we use the <progress> tag.

Example:

```
<!DOCTYPE html>
```

```
<html>
<body bgcolor="#b5dcb3">
<form>
        <meter value = “5” min = “0” max = “10”> 2 out of 10 </meter>
</form>
</body>
</html>
```

12. Command:

The command element represents a command which the user can invoke. The type attribute specifies the type of command.

Syntax: <command type = “command or checkbox or radio”>

Attribute Values

Value	Description
command	Default. Specifies a normal command with an action
checkbox	Specifies a command that can be toggled using a checkbox
radio	Specifies a command that can be toggled using a radio button

Example:

```
<!DOCTYPE html>
<html>
<body bgcolor="#b5dcb3">
<form>
<menu>
        <command type="command" label="Save" onclick="save()">Save</command>
</menu>
</form>
</body>
</html>
```

13. Menu:

The <menu> tag defines a list or menu of commands. The <menu> tag is used for context menus, toolbars and for listing form controls and commands.

Example:

```
<!DOCTYPE html>
<html>
<body bgcolor="#b5dcb3">
<form>
<menu>
        <command type="command" label="Save" onclick="save()">Save</command>
</menu>
```

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

14. Spellcheck:

The spellcheck attributes specifies whether the element is to have its spelling and grammar checked or not. The following can be spellchecked:

- Text values in input elements except passwords.
- Text in <textarea> element
- Text in editable elements

Example:

```
<!DOCTYPE html>
<html>
<body bgcolor="#b5dcb3">
<form>
<p contenteditable = "true" spellcheck = "true"> this is a paragraph. </p>
</form>
</body>
</html>
```

15. Header:

The <header> element represents a container for introductory content or a set of navigational links. A <header> element typically contains:

- One Or More Heading Elements (<H1> - <H6>)
- Logo Or Icon
- Authorship Information

We can have several <header> elements in one document.

Note: A <header> tag cannot be placed within a <footer>, <address> or another <header> element.

Example:

```
<!DOCTYPE html>
<html>
<body bgcolor="#b5dcb3">
<form>
<article>
 <header>
  <h1>Most important heading here</h1>
  <h3>Less important heading here</h3>
  <p>Some additional information here</p>
 </header>
 <p>Lorem Ipsum dolor set amet....</p>
```

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

16. Footer:

The <footer> tag defines a footer for a document or section. A <footer> element should contain information about its containing element. A <footer> element typically contains:

- Authorship Information
- Copyright Information
- Contact Information
- Sitemap
- Back To Top Links
- Related Documents

We can have several <footer> elements in one document.

Example:

```
<!DOCTYPE html>
<html>
<body bgcolor="#b5dcb3">
<form>
<article>
<footer>
 <p>Posted by: Hege Refsnes</p>
 <p>Contact information: <a href="mailto:someone@example.com">
 someone@example.com</a>.</p>
</footer>
</article>
</form>
</body>
</html>
```