PHP MODULE I PART 1

Introduction To E-Documents

An electronic document is any electronic media content (other than computer programs or system files) that is intended to be used in either an electronic form or as printed output. Examples of documents subject to e-discovery are e-mails, voicemails, instant messages, e-calendars, audio files, and data on handheld devices, animation, metadata, graphics, photographs, spread sheets, websites, drawings and other types of digital data.

Static

“Static" means unchanged or constant. Static Web pages contain the same prebuilt content each time the page is loaded. Each page is coded in HTML and displays the same information to every visitor. Static sites are the most basic type of website and are the easiest to create. A static site can be built by simply creating a few HTML pages and publishing them to a Web server. Since static Web pages contain fixed code, the content of each page does not change unless it is manually updated by the webmaster.

Examples:

www.google.com, www.gemseducation.org, www.keralapsc.gov.in

Dynamic

"Dynamic" means changing or lively. The content of dynamic Web pages can be generated on-the-fly. PHP, ASP, and JSP pages are dynamic Web pages. These pages contain "server-side" code, which allows the server to generate unique content each time the page is loaded. For example, the server may display the current time and date on the Web page. Many dynamic pages use server-side code to access database information, which enables the page's content to be generated from information stored in the database. Websites that generate Web pages from database information are often called database-driven websites.

We can often tell if a page is static or dynamic simply by looking at the page's file extension in the URL, located in the address field of the Web browser. If it is ".htm" or ".html," the page is probably static. If the extension is ".php," ".asp," or ".jsp," the page is most likely dynamic.

Active

Active pages more dynamic than "dynamic" web pages. Active web pages contain constantly changing data. Every time a dynamic web page is requested by a web browser the HTML that is returned is created programmatically by the web server exactly as specified in scripts or programs. It can show different content for different customers. Internal interaction at client - not dependent on server for interaction.

Active Web pages - advantages

- Decreased server load - in requests and in CPU work
- Decreased network traffic
- Faster response to user
- An active document may be smaller to download than its resulting display (whether essentially static in content, or active)
- The page itself does not change – cacheable

WEB PROGRAMMING

Two types of web programming Client side programming and Server side programming

Client-side

Client-side programming is run on the user's computer. The processing takes place on the end users computer. The source code is transferred from the web server to the users' computer over the internet and run directly in the browser. An example of client-side programming is JavaScript. JavaScript can be used to run checks on form values and send alerts to the user's browser. The problem with client-side scripts is the limit of control and problems with operating systems and web browsers.

Server-side

Server-side scripts are run on the server. A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser. It is usually used to provide interactive web sites that interface to databases or other data stores on the server. This reduces the amount of bugs or compatibility issues since the code is run on one server using one language and hosting software. Server-side programming can also be encrypted when users send form variables, protecting users against any hack attempts. Some examples of server-side programming languages are C#, VB.NET, and PHP.

HTML5

HTML5 is the fifth revision and newest version of the HTML standard. It offers new features that provide not only rich media support but also enhance support for creating web applications that can interact with users, their local data, and servers more easily and effectively than was previously possible. Firefox, has very good support for HTML5.

HTML stands for Hyper Text Markup Language. It is used to design web pages using markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. Markup language is used to define the text document within tag which defines the structure of web pages. HTML 5 is the fifth and current version of HTML. It has

improved the markup available for documents and has introduced application programming interfaces (API) and Document Object Model (DOM).

Features:

It has introduced new multimedia features which supports audio and video controls by using <audio> and <video> tags. There are new graphics elements including vector graphics and tags. Enrich semantic content by including <header> <footer>, <article>, <section> and <figure> are added. Drag and Drop- The user can grab an object and drag it further dropping it on a new location. Geo-location services- It helps to locate the geographical location of a client. Web storage facility which provides web application methods to store data on web browser.

- Uses SQL database to store data offline.
- Allows to draw various shapes like triangle, rectangle, circle, etc.
- Capable of handling incorrect syntax.
- Easy DOCTYPE declaration i.e. <!doctype html>
- Easy character encoding i.e. <meta charset="UTF-8">

DOCUMENT STRUCTURE

```
<! DOCTYPE html>
<html> <head>
            <meta charset=UTF-8" />
            <title>.....</title>
      </head>
     <body>
          <header>....</header>
          <h1>....</h1>
          <nav>.....</nav>
          <article>
               <section>
                    .....
               <section>
          </article>
          <aside>.....</aside>
          <footer>....</footer>
     </body>
</html>
```

The DOCTYPE

It describes what type of the document is. This is a separate file that contains a formal definition of a grammar. That supported elements attributes and the versions of markup language. The DOCTYPE is not a case sensitive, so we can write in lower case, upper case or in mixed case letters.

```
<! DOCTYPE  html>
```

The <HTML> tag

It is a container tag. The DIR attribute of the tag is used to indicate the document direction. It can take values rtl or ltr. It describes language direction is left to right or right to left.

Eg:- <HTML DIR =”rtl”>

The <HEAD> tag

It is a container tag. The head section includes document title, general style definition, script programs, etc.

Supporting Elements of <head>

1. The <**title**> element
   - Contained the title of the HTML document
   - Which appears in the title bar of the web browser

2. The <**base**> element
   - To specifies the default URL and target for all the links in the HTML document
   - It is the first element in the head element
   - The href attribute contains the url of the HTML document.
   - The target attribute specifies where the link opens.
   - The values are _blank, _parent, _self, _top

   base href=“url” target=“value”/>

3. The <**link**> element
   - Used to link other HTML documents
   - Defines the relationship between two different HTML documents
   - It have two attributes href and rel.
   - rel used to define the relationship of the linked documents

   <link href=“file_name.html” rel=“stylesheet”/>

4. The <**command**> element
   - To execute a command when an event is fired by a form control such as radio button or checkbox
   - Can use the command element with the context menu or toolbar.

   ```
   <menu>
       <command type=“command” label=“Save”
                onclick=“save()”>Save</command>
   </menu>
   ```

5. The <**meta**> element
   - To provide the information about an HTML document such as page description and keywords
   - Attributes are charset, content, http-equiv, name and scheme
   - Only the content attribute is required.

```
<head>
   <meta charset= "UTF-8">
   <meta name="description" content="free online classes">
   <meta name="keyword" content="HTML, CSS, PHP, JavaScript">
   <meta name="author" content="Kiran Babu">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
</head>
```

6. The <**script**> element
   - To define a client-side script, such as a JavaScript
   - The <script> element either contains scripting statements, or it points to an external script file through the src attribute

```
<script type="text/javascript">
        document.write("Hello World!");
</script>
<script type="text/javascript" src="first.js">
</script>
```

7. The <**noscript**> element
   - Used to display the alternate content of script not working in the browser

```
<script type="text/javascript">
        document.write("Hello World!");
</script>
<noscript>Your browser does not support JavaScript !</noscript>
```

8. The <**style**> element
   - To declare the style sheet within the HTML document
   - Main attributes are type, media and scoped
   - If we use scoped attribute the <style> tag is placed within <body> tag

The <BODY> tag

It is a container tag. Body section forms the content displayed in the browser window. Hence all other tags, which define the document content, are given in the body section.

Supporting Elements of <body>

1. The <**section**> element
   - To group **stand alone** sections within a web page

```
<section>
        <h1> Header 1 </h1>
        <p>Some paragraph for example</p>
</section>
```

2. The <**nav**> element
   - Defines a section of navigation links in a documents

- Internal links and external links

```
<nav>
    <a href="/html/">HTML</a>
    <a href="/css/">CSS</a>
    <a href="/jquery/">jQuery</a>
</nav>
```

3. The <**article**> element
   - To define an independent, self-contained content
   - Article, blog posts, comments, etc
   - We can display the information in this element in various format

```
<body>
    <article>
        <p>Text of the article</p>
    </article>
```

4. The <**aside**> element
   - To create a section that used to display information about the content of other element Such as time, date, current news, weather report, etc.
   - Also used for typographical effect in the documents
   - Such as sidebars for advertisement, links, notes and tips

```
<aside>
    Enter some content related to the article
</aside>
```

5. The <**header**> element
   - Provide the introductory content on a HTML page
   - It contained headings, paragraphs, links, etc.
   - We can insert several header elements in one document
- But header element cannot placed any other <header>, <footer> or <address>

```
<header>

    <h1>Most important heading here</h1>
    <h3>Less important heading here</h3>
    <p>Some additional information here</p>
</header>
```

6. The <**footer**> element
   - Defines the foot note of the document or a section
   - It contained typically information about the author of the document, copyright information, links to related documents

```
<footer>
    <p>Posted by: Anoos Babu P K</p>
    <p>Contact Information:
    <a href="mailto:pkandoubleos@gmail.com"> Send Email </a> </p>
</footer>
```

7. The <**address**> element
   - Defines the contact information for the author/owner of the document or article
   - If <address> element is inside the <body> element, it represents contact information for the document
   - If <address> element is inside the <article> element, it represents contact information for the article

```
<address>
        written by
    <a href="mailto:pkandoubleos@gmail.com"> Raj Kumar MK</a> <br>
            Visit us at:<br>
            gemscollege.in<br>
            Ramapuram, Malappuram <br>
            Kerala, India<br>
    </address>
```

HTML ELEMENTS

An HTML element is defined by a starting tag. If the element contains other content, it ends with a closing tag, where the element name is preceded by a forward slash. The following are the types of elements.

1. Structural Elements

Element	Description
<a>	Defines a hyperlink.
<article>	Defines an article.
<aside>	Defines some content loosely related to the page content.
<body>	Defines the document's body.
 	Produces a single line break.
<details>	Represents a widget from which the user can obtain additional information or controls on-demand.
	Specifies a division or a section in a document.
<h1> to <h6>	Defines HTML headings.
<head>	Defines the head portion of the document that contains information about the document.
<header>	Represents the header of a document or a section.
<hgroup>	Defines a group of headings.
<hr>	Produce a horizontal line.
<html>	Defines the root of an HTML document.
<footer>	Represents the footer of a document or a section.
<nav>	Defines a section of navigation links.
<p>	Defines a paragraph.
<section>	Defines a section of a document, such as header, footer etc.
	Defines an inline styleless section in a document.
<summary>	Defines a summary for the <details> element.

2. Metadata Elements

Element	Description
<base>	Defines the base URL for all linked objects on a page.
<link>	Defines the relationship between the current document and an external resource.
<meta>	Provides structured metadata about the document content.
<style>	Inserts style information (commonly CSS) into the head of a document.
<title>	Defines a title for the document.

3. Form Elements

Element	Description
<button>	Creates a clickable button.
<datalist>	Represents a set of pre-defined options for an <input> element.
<fieldset>	Specifies a set of related form fields.
<form>	Defines an HTML form for user input.
<input>	Defines an input control.
<keygen>	Represents a control for generating a public-private key pair.
<label>	Defines a label for an <input> control.
<legend>	Defines a caption for a <fieldset> element.
<meter>	Represents a scalar measurement within a known range.
<optgroup>	Defines a group of related options in a selection list.
<option>	Defines an option in a selection list.
<select>	Defines a selection list within a form.
<textarea>	Defines a multi-line text input control (text area).

4. Script Elements

Element	Description
<noscript>	Defines alternative content to display when the browser doesn't support scripting.
<script>	Places script in the document for client-side processing.

5. Table Elements

Tag	Description
<caption>	Defines the title of a table.
<col>	Defines attribute values for one or more columns in a table.
<colgroup>	Specifies attributes for multiple columns in a table.
<table>	Defines a data table.
<tbody>	Groups a set of rows defining the main body of the table data.
<td>	Defines a cell in a table.
<tfoot>	Groups a set of rows summarizing the columns of the table.
<thead>	Groups a set of rows that describes the column labels of a table.
<th>	Defines a header cell in a table.
<tr>	Defines a row of cells in a table.

6. Formatting Elements

Element	Description
<abbr>	Defines an abbreviated form of a longer word or phrase.
<acronym>	Defines an acronym.
<address>	Specifies the author's contact information.
<b>	Displays text in a bold style.
<bdi>	Represents text that is isolated from its surrounding for the purposes of bidirectional text formatting.
<bdo>	Overrides the current text direction.
<big>	Displays text in a large size.
<blockquote>	Defines a long quotation.
<cite>	Indicates a citation or reference to another source.
<code>	Specifies text as computer code.
<del>	Specifies a block of deleted text.
<dfn>	Specifies a definition.
	Specifies emphasized text.
<i>	Displays text in an italic style.
<ins>	Defines a block of text that has been inserted into a document.
<kbd>	Specifies text as keyboard input.
<mark>	Represents text highlighted for reference purposes.
<output>	Represents the result of a calculation.
<pre>	Defines a block of preformatted text.
<progress>	Represents the completion progress of a task.

Element	Description
<q>	Defines a short inline quotation.
<rp>	Provides fall-back parenthesis for browsers that that don't support ruby annotations.
<rt>	Defines the pronunciation of character presented in a ruby annotations.
<ruby>	Represents a ruby annotation.
<samp>	Specifies text as sample output from a computer program.
<small>	Displays text in a smaller size.
	Indicate strongly emphasized text.
<sub>	Defines subscripted text.
<sup>	Defines superscripted text.
<tt>	Displays text in a teletype style.
<var>	Defines a variable.
<wbr>	Represents a line break opportunity.

7. List Elements

Element	Description
<dd>	Specifies a definition for a term in a definition list.
<dl>	Defines a definition list.
<dt>	Defines a term (an item) in a definition list.
	Defines a list item.
	Defines an ordered list.
<menu>	Represents a list of commands.
	Defines an unordered list.

8. Embedded Content Elements

Element	Description
<area>	Defines a specific area within an image map.
<audio>	Embeds a sound, or an audio stream in an HTML document.
<canvas>	Defines a region in the document, which can be used to draw graphics on the fly via scripting (usually JavaScript).
<embed>	Embeds external application, typically multimedia content like audio or video into an HTML document.
<figcaption>	Defines a caption or legend for a figure.
<figure>	Represents a figure illustrated as part of the document.
<frame>	Defines a single frame within a frameset.
<frameset>	Defines a collection of frames or other frameset.
<iframe>	Displays a URL in an inline frame.
<img>	Displays an inline image.
<map>	Defines a client-side image-map.
<noframes>	Defines an alternate content that displays in browsers that do not support frames.
<object>	Defines an embedded object.
<param>	Defines a parameter for an object or applet element.
<source>	Defines alternative media resources for the media elements like <audio> or <video>.
<time>	Represents a time and/or date.
<video>	Embeds video content in an HTML document.

HTML ATTRIBUTES

- All HTML elements can have attributes
- Attributes provide additional information about elements
- Attributes are always specified in the start tag
- Attributes usually come in name/value pairs like: name="value"

Types of Attributes

1. Required and Optional Attribute

Attribute	Belongs to	Description
accept	<input>	Specifies the types of files that the server accepts (only for type="file")
accept-charset	<form>	Specifies the character encodings that are to be used for the form submission
action	<form>	Specifies where to send the form-data when a form is submitted
alt	<area>, <img>, <input>	Specifies an alternate text when the original element fails to display
async	<script>	Specifies that the script is executed asynchronously (only for external scripts)
autocomplete	<form>, <input>	Specifies whether the <form> or the <input> element should have autocomplete enabled
autofocus	<button>, <input>, <select>, <textarea>	Specifies that the element should automatically get focus when the page loads
autoplay	<audio>, <video>	Specifies that the audio/video will start playing as soon as it is ready
autoplay	<audio>, <video>	Specifies that the audio/video will start playing as soon as it is ready
charset	<meta>, <script>	Specifies the character encoding
checked	<input>	Specifies that an <input> element should be pre-selected when the page loads (for type="checkbox" or type="radio")
cite	<blockquote>, <del>, <ins>, <q>	Specifies a URL which explains the quote/deleted/inserted text
cols	<textarea>	Specifies the visible width of a text area
colspan	<td>, <th>	Specifies the number of columns a table cell should span
content	<meta>	Gives the value associated with the http-equiv or name attribute
controls	<audio>, <video>	Specifies that audio/video controls

Attribute	Belongs to	Description
		should be displayed (such as a play/pause button, etc)
coords	<area>	Specifies the coordinates of the area
datetime	<del>, <ins>, <time>	Specifies the date and time
default	<track>	Specifies that the track is to be enabled if the user's preferences do not indicate that another track would be more appropriate
defer	<script>	Specifies that the script is executed when the page has finished parsing (only for external scripts)
dirname	<input>, <textarea>	Specifies that the text direction will be submitted
disabled	<button>, <fieldset>, <input>, <optgroup>, <option>, <select>, <textarea>	Specifies that the specified element/group of elements should be disabled
download	<a>, <area>	Specifies that the target will be downloaded when a user clicks on the hyperlink
enctype	<form>	Specifies how the form-data should be encoded when submitting it to the server (only for method="post")
for	<label>, <output>	Specifies which form element(s) a label/calculation is bound to
form	<button>, <fieldset>, <input>, <label>, <meter>, <object>, <output>, <select>, <textarea>	Specifies the name of the form the element belongs to
formaction	<button>, <input>	Specifies where to send the form-data when a form is submitted. Only for type="submit"
headers	<td>, <th>	Specifies one or more headers cells a

Attribute	Belongs to	Description
		cell is related to
height	<canvas>, <embed>, <iframe>, <img>, <input>, <object>, <video>	Specifies the height of the element
high	<meter>	Specifies the range that is considered to be a high value
href	<a>, <area>, <base>, <link>	Specifies the URL of the page the link goes to
hreflang	<a>, <area>, <link>	Specifies the language of the linked document
http-equiv	<meta>	Provides an HTTP header for the information/value of the content attribute
ismap	<img>	Specifies an image as a server-side image-map
kind	<track>	Specifies the kind of text track
label	<track>, <option>, <optgroup>	Specifies the title of the text track
list	<input>	Refers to a <datalist> element that contains pre-defined options for an <input> element
loop	<audio>, <video>	Specifies that the audio/video will start over again, every time it is finished
low	<meter>	Specifies the range that is considered to be a low value
max	<input>, <meter>, <progress>	Specifies the maximum value
maxlength	<input>, <textarea>	Specifies the maximum number of characters allowed in an element
media	<a>, <area>, <link>, <source>, <style>	Specifies what media/device the linked document is optimized for
method	<form>	Specifies the HTTP method to use when sending form-data
min	<input>, <meter>	Specifies a minimum value
multiple	<input>, <select>	Specifies that a user can enter more

Attribute	Belongs to	Description
		than one value
muted	<video>, <audio>	Specifies that the audio output of the video should be muted
name	<button>, <fieldset>, <form>, <iframe>, <input>, <map>, <meta>, <object>, <output>, <param>, <select>, <textarea>	Specifies the name of the element
novalidate	<form>	Specifies that the form should not be validated when submitted
open	<details>	Specifies that the details should be visible (open) to the user
optimum	<meter>	Specifies what value is the optimal value for the gauge
pattern	<input>	Specifies a regular expression that an <input> element's value is checked against
placeholder	<input>, <textarea>	Specifies a short hint that describes the expected value of the element

poster	<video>	Specifies an image to be shown while the video is downloading, or until the user hits the play button
preload	<audio>, <video>	Specifies if and how the author thinks the audio/video should be loaded when the page loads
readonly	<input>, <textarea>	Specifies that the element is read-only
rel	<a>, <area>, <link>	Specifies the relationship between the current document and the linked document
required	<input>, <select>, <textarea>	Specifies that the element must be filled out before submitting the form
reversed		Specifies that the list order should be descending (9,8,7...)

Attribute	Belongs to	Description
rows	<textarea>	Specifies the visible number of lines in a text area
rowspan	<td>, <th>	Specifies the number of rows a table cell should span
sandbox	<iframe>	Enables an extra set of restrictions for the content in an <iframe>
scope	<th>	Specifies whether a header cell is a header for a column, row, or group of columns or rows
selected	<option>	Specifies that an option should be pre-selected when the page loads
shape	<area>	Specifies the shape of the area
size	<input>, <select>	Specifies the width, in characters (for <input>) or specifies the number of visible options (for <select>)
sizes	<img>, <link>, <source>	Specifies the size of the linked resource
span	<col>, <colgroup>	Specifies the number of columns to span

src	<audio>, <embed>, <iframe>, <img>, <input>, <script>, <source>, <track>, <video>	Specifies the URL of the media file
srcdoc	<iframe>	Specifies the HTML content of the page to show in the <iframe>
srclang	<track>	Specifies the language of the track text data (required if kind="subtitles")
srcset	<img>, <source>	Specifies the URL of the image to use in different situations
start		Specifies the start value of an ordered list
step	<input>	Specifies the legal number intervals for an input field
target	<a>, <area>, <base>, <form>	Specifies the target for where to open the linked document or where to submit the form

Attribute	Belongs to	Description
type	<a>, <button>, <embed>, <input>, <link>, <menu>, <object>, <script>, <source>, <style>	Specifies the type of element
usemap	<img>, <object>	Specifies an image as a client-side image-map
value	<button>, <input>, , <option>, <meter>, <progress>, <param>	Specifies the value of the element
width	<canvas>, <embed>, <iframe>, <img>, <input>, <object>, <video>	Specifies the width of the element
wrap	<textarea>	Specifies how the text in a text area is to be wrapped when submitted in a form

2. Standard or Global Attribute

Attribute	Value	Description
accesskey	*shortcut key*	Specifies a keyboard shortcut to activate or focus the element.
class	*classname*	Assigns a class name or space-separated list of class names to an element.
contenteditable	true	Indicates whether the content of an
	false	element is editable by the user or not.
contextmenu	*menu-id*	Specifies a context menu for an element. A context menu is a menu that appears when the user clicks the right mouse button on the element.
data-*	*data*	Specified on any HTML element, to store custom data specific to the page.
dir	ltr rtl	Specifies the base direction of directionality of the element's text.
draggable	true false	Specifies whether an element is draggable or not.
dropzone	copy move link	Specifies whether the dragged data is copied, moved, or linked, when dropped.
hidden	hidden	Indicates that the element is not yet, or is no longer, relevant.
id	*name*	Specifies a unique identifier (ID) for an element which must be unique in the whole document.
lang	*language-code*	Specifies the primary language for the element's text content.
spellcheck	true false	Specifies whether the element may be checked for spelling errors or not.
style	*style*	Specifies inline style information for an element.
tabindex	*number*	Specifies the tabbing order of an element.
title	*text*	Provides advisory information related to the element. It would be appropriate for a tooltip.
translate	yes no	Specifies whether the text content of an element should be translated or not.
xml:lang	*language-code*	Specifies the primary language for the element's text content, in XHTML documents.

3. Event Attribute

a) Window Events

Attribute	Value	Description
onafterprint	*script*	Fires after the associated document is printed.
onbeforeprint	*script*	Fires before the associated document is printed.
onbeforeunload	*script*	Fires before a document being unloaded.
onerror	*script*	Fires when document errors occur.
onhashchange	*script*	Fires when the fragment identifier part of the document's URL i.e. the portion of a URL that follows the sign (#) changes.
onload	*script*	Fires when the document has finished loading.
onmessage	*script*	Fires when the message event occurs i.e. when user sends a cross-document message or a message is sent from a worker with `postMessage()` method.
onoffline	*script*	Fires when the network connection fails and the browser starts working offline.
ononline	*script*	Fires when the network connections returns and the browser starts working online.
onpagehide	*script*	Fires when the page is hidden, such as when a user is moving to another webpage.
onpageshow	*script*	Fires when the page is shown, such as when a user navigates to a webpage.
onpopstate	*script*	Fires when changes are made to the active history.
onresize	*script*	Fires when the browser window is resized.
onstorage	*script*	Fires when a Web Storage area is updated.
onunload	*script*	Fires immediately before the document is unloaded or the browser window is closed.

b) Form Events

Attribute	Value	Description
onblur	*script*	Fires when an element loses focus.
onchange	*script*	Fires when the value or state of the element is changed.
onfocus	*script*	Fires when the element receives focus.
oninput	*script*	Fires when the value of an element is changed by the user.
oninvalid	*script*	Fires when a submittable element do not satisfy their constraints during form validation.
onreset	*script*	Fires when the user resets a form.
onselect	*script*	Fires when some text is being selected or the current selection is changed by the user.
onsearch	*script*	Fires when the user writes something in a search input field.
onsubmit	*script*	Fires when a form is submitted.

c) Mouse Events

Attribute	Value	Description
onclick	*script*	Fires when the user clicks the left mouse button on the element.
ondblclick	*script*	Fires when the user double-clicks on the element.
oncontextmenu	*script*	Fires when a context menu is triggered by the user through right-click on the element.
ondrag	*script*	Fires when the user drags an element. The ondrag event fires throughout the drag operation.
ondragend	*script*	Fires when the user releases the mouse button at the end of a drag operation.
ondragenter	*script*	Fires when the user drags an element to a valid drop target.
ondragleave	*script*	Fires when an element leaves a valid drop

Attribute	Value	Description
		target during a drag operation.
ondragover	*script*	Fires when an element is being dragged over a valid drop target.
ondragstart	*script*	Fires when the user starts to drag a text selection or selected element.
ondrop	*script*	Fires when the mouse button is released during a drag-and-drop operation i.e. when dragged element is being dropped.
onmousedown	*script*	Fires when the mouse button is pressed over an element.
onmousemove	*script*	Fires when the user moves the mouse pointer over an element.
onmouseout	*script*	Fires when the user moves the mouse pointer outside the boundaries of an element.
onmouseover	*script*	Fires when the user moves the mouse pointer onto an element.
onmouseup	*script*	Fires when the user releases the mouse button while the mouse is over an element.
onscroll	*script*	Fires when the user scrolls the contents of an element by scrolling the element's scrollbar.
onshow	*script*	Fires when a contextmenu event was fired onto an element that has a contextmenu attribute.
ontoggle	*script*	Fires when the user opens or closes the <details> element.
onwheel	*script*	Fires when the user scrolls the contents of an element by rolling the mouse wheel up or down over an element.

d) Keyboard Events

Attribute	Value	Description
onkeydown	*script*	Fires when the user presses a key.
onkeypress	*script*	Fires when the user presses an alphanumeric key.
onkeyup	*script*	Fires when the user releases a key.

e) Clipboard Events

Attribute	Value	Description
oncopy	*script*	Fires when the user copies the element or selection, adding it to the system clipboard.
oncut	*script*	Fires when the element or selection is removed from the document and added to the system clipboard.
onpaste	*script*	Fires when the user pastes data, transferring the data from the system clipboard to the document.

f) Media Events

Attribute	Value	Description
onabort	*script*	Fires when playback is aborted, but not due to an error.
oncanplay	*script*	Fires when enough data is available to play the media, at least for a couple of frames, but would require further buffering.
oncanplaythrough	*script*	Fires when entire media can be played to the end without requiring to stop for further buffering.
oncuechange	*script*	Fires when the text track cue in a <track> element changes.
ondurationchange	*script*	Fires when the duration of the media changes.
onemptied	*script*	Fires when the media element is reset to its initial state, either because of a fatal error during load, or because the load() method is called to reload it.
onended	*script*	Fires when the end of playback is reached.
onerror	*script*	Fires when an error occurs while fetching the media data.
onloadeddata	*script*	Fires when media data is loaded at the current playback position.

Attribute	Value	Description
onloadedmetadata	*script*	Fires when metadata of the media (like duration and dimensions) has finished loading.
onloadstart	*script*	Fires when loading of the media begins.
onpause	*script*	Fires when playback is paused, either by the user or programmatically.
onplay	*script*	Fires when playback of the media starts after having been paused i.e. when the play() method is requested.
onplaying	*script*	Fires when the audio or video has started playing.
onprogress	*script*	Fires periodically to indicate the progress while downloading the media data.
onratechange	*script*	Fires when the playback rate or speed is increased or decreased, like slow motion or fast forward mode.
onseeked	*script*	Fires when the seek operation ends.
onseeking	*script*	Fires when the current playback position is moved.
onstalled	*script*	Fires when the download has stopped unexpectedly.
onsuspend	*script*	Fires when the loading of the media is intentionally stopped.
ontimeupdate	*script*	Fires when the playback position changed, like when the user fast forwards to a different playback position.
onvolumechange	*script*	Fires when the volume is changed, or playback is muted or unmuted.
onwaiting	*script*	Fires when playback stops because the next frame of a video resource is not available.

BASIC HTML DATA TYPES

1. Character
   - Single alpha numeric text
   - Letters, numbers, Symbols, Space or Punctuations
2. Text
   - String
3. Name
   - Any particular data element or function
4. Number
   - Store a number (15 digits of +ve or –ve)

RFC and IANA Data Types

1. Uniform Resource identifier
   - url
2. Content Type
   - Text/plain(.txt), image/png(.png), videi/mpeg(.mpg, mpeg)
3. Language Code
   - “en” -English, “hn” -Hindi, “el” -Greek, “de” -German, “ga” -Irish
4. Character Set
   - dollar symbol, delta, omega, exclamation mark, question mark, lowercase letters, uppercase letters

W3C Data Types

1. Date Time
   (YYYY, MM, DD, hh, mm, ss, T, TZD)
2. RGB Triplet
   (#000000, #FFFFFF, #FF0000,#0000FF)
3. Color Names
   (Yellow, Black, Navy, Teal, Maroon, Silver)
4. Link Type
   (Next, Perv, Start, Section, Help, Script, Index)
5. Media Type
   (tv, projection, handled, print, aural, screen)

HTML5 FORM ELEMENTS

1. DATALIST

The HTML <datalist> element contains a set of <option> elements that represent the permissible or recommended options available to choose from within other controls. The <datalist> tag specifies a list of pre-defined options for an <input> element. The <datalist> tag is used to provide an "autocomplete" feature for <input> elements. Users will see a drop-down list of pre-defined options as they input data. The <datalist> element's id attribute must be equal to the <input> element's list attribute

```
<input list="ice-cream-flavors" />
<datalist id="ice-cream-flavors">
   <option value="Chocolate">
   <option value="Coconut">
   <option value="Mint">
   <option value="Strawberry">
   <option value="Vanilla">
</datalist>
```

2. KEYGEN

The <keygen> element generates an encryption key for passing encrypted data to a server. When an HTML form is submitted, the browser will generate a key pair and store the private key in the browser's local key storage and send the public key to the server.

```
<body>
    <form action="demo_keygen.php" method="POST">
    <label>Username: <input type="text" name="username"></label>
    <label>Encryption: <keygen name="key"></label>
    <input type="submit" value="Submit">    <br><br>
    </form>
</body>
```

3. OUTPUT

The <output> tag is used to represent the result of a calculation. The <output> tag in HTML is used to represent the result of a calculation performed by the client-side script such as JavaScript.

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">
    <input type="range" id="a" value="50">
    +<input type="number" id="b" value="25">
    =<output name="x" for="a b"></output>
</form>
```

4. PROGRESS

The HTML <progress> element displays an indicator showing the completion progress of a task, typically displayed as a progress bar.
HTML <progress> tag is used to display the progress of a task. It provides an easy way for web developers to create progress bar on the website. It is mostly used to show the progress of a file uploading on the web page.

```
<body>
    <h3>The progess element</h3>
    <label for="file">Downloading progress:</label>
    <progress id="file" value="40" max="100"> 32% </progress>
</body>
```

5. METER

The HTML <meter> element represents either a scalar value within a known range or a fractional value.

```
<body>
<label for="fuel">Fuel level:</label>
<meter id="fuel"
    min="0" max="100"
    low="33" high="66" optimum="80"
```

```
    value="70">
  at 50/100
</meter>
</body>
```

6. FILE UPLOADING USING FORMS

The <input type="file"> defines a file-select field and a "Browse" button for file uploads. To define a file-select field that allows multiple files to be selected, add the "multiple" attribute.

```
<body>
<form action="/action_page.php">
      <label for="img">Select image:</label>
       <input type="file" id="img" name="img" accept="image/*">
       <input type="submit">
</form>
</body>
```

7. FRAMESET AND FRAMES

The frameset page divides the browser window into number of section so that different pages can be loaded in each frame. The individual section of a frameset window is known as frame. The <frameset> and <frame> tags are needed to create a frameset page. The <frameset> tag is a container tag. But the <frame> tag is an empty tag. No <body> tag is needed for a frameset page.

```
<html><head><title>Frameset
Page </title></head><body><h2>Left Frame</h2></body></html>
```
And save as ***E:\my webpages\leftframe.html***

```
<html><head><title>Frameset Page </title></head><body><h2>Right Frame 1
</h2></body></html>
```
And save as ***E:\my webpages\rightframe1.html***

```
<html><head><title>Frameset Page </title></head><body><h2>Right Frame 2
</h2></body></html>
```
And save as ***E:\my webpages\rightframe2.html***

```
<html><head><title>Frameset Page </title></head><body><h2>Right Frame 3
</h2></body></html>
```
And save as ***E:\my webpages\rightframe3.html***

Then we create frameset window and save as ***frameset.html***

```
<html><head><title>Frameset Page
</title>.</head>
<frameset cols="30%,*">
<frame src="E:\my
webpages\leftframe.html">
<frameset
rows="28%,20%,30%,*">
<frame src="E:\my
webpages\rightframe1.html">
```

```
<frame src="E:\my
webpages\rightframe2.html">
<frame src="E:\my
webpages\rightframe3.html">
</frameset>
</html>
```