HTML BASICS

HTML stands for Hypertext Markup Language, and it is the most widely used language to write Web Pages.

• Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.

• As its name suggests, HTML is a Markup Language which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

KEY NOTE:

- HTML describes the structure of Web pages using markup.
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tag.
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on.
- Browsers do not display the HTML tags, but use them to render the content of the page

BASIC HTML DOCUMENT

```
<!DOCTYPE html>
<html>
<head>
<title>This is document title</title>
</head>
<body>
<h1>This is a heading</h1>
<p>Document content goes here.....</p>
</body>
</html>
```

HTML VERSIONS

Version	Year
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2014

HTML TAGS

HTML tags are element names surrounded by angle brackets:

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

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

Tag	Description
<!DOCTYPE...>	This tag defines the document type and HTML version.
<html>	This tag encloses the complete HTML document and mainly comprises of document header which is represented by <head>...</head> and document body which is represented by <body>...</body> tags.
<head>	This tag represents the document's header which can keep other HTML tags like <title>, <link> etc.
<title>	The <title> tag is used inside the <head> tag to mention the document title.
<body>	This tag represents the document's body which keeps other HTML tags like <h1>, , etc.
<h1>	This tag represents the heading.

	This tag represents a paragraph.

Heading tags:

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

Paragraph tag:

The tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening ** ** and a closing ** ** tag.

- LOREM IPSUM : It is the Dummy text written inside the paragraph tag. And it has a Plug in with visual Studios.
- The Library name of Lorem Ipsum is =EMMET ABBREVIATION

Line Break Tag:

Whenever you use the **
** element, anything following it starts from the next line. This tag is an example of an empty element, where you do not need opening and closing tags, as there is nothing to go in between them.

Centering Content:

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

Horizontal Lines:

Horizontal lines are used to visually break-up sections of a document.

- The **<hr>** tag creates a line from the current position in the document to the right margin and breaks the line accordingly.

- The **<hr >** tag is an example of the empty element, where you do not need opening and closing tags, as there is nothing to go in between them.

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 as shown below with few tags:

Start Tag	Content	End Tag
	This is paragraph content.	
<h1>	This is heading content.	</h1>
	This is division content.	

- So here ** ... ** is an HTML element, **<h1>....</h1>** is another HTML element. There are some HTML elements which don't need to be closed, such as **<img>,<meta>,<link>,<hr>,<input> and
** elements. These are known as **void elements**.

- `area` - clickable, defined area in an image
- `base` - specifies a base URL from which all links base
- `br` - line break
- `col` - column in a table [deprecated]
- `hr` - horizontal rule (line)
- `img` - image
- `input` - field where users enter data
- `link` - links an external resource to the document
- `meta` - provides information about the document
- `param` - defines parameters for plugins

HTML Tags v/s Elements

An HTML element is defined by a starting tag. If the element contains other content, it ends with a closing tag.

For example, is starting tag of a paragraph and is closing tag of the same paragraph but ,This is paragraph This is a paragraph is a paragraph element.

HTML 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

name Attribute

- Each input field must have a `name` attribute to be submitted.
- If the `name` attribute is omitted, the data of that input field will not be sent at all.

Example:

```
<form action="/action_page.html">
 First name:<br>
<input type="text" value="ankith"><br>
  Last name:<br>
<input type="text" name="lastname" value="mishra"><br><br>
<input type="submit" value="Submit">
</form>
```

Value Attribute

The **value** is what you want the value of the property to be set and always put within quotations.

- The `value` attribute specifies the initial value for an input field.
- Attribute names and attribute values are case-insensitive.
- Example:

```
<form action="#">
        First name:<br>
    <input type="text" name="firstname" value="John">
        </form>
```

Core Attributes

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

- Id
- Title

- Class
- Style

To make a HTML Structure :

We use 2 tags of Structural Elements: they are

1.<Div>

2.<Span>

Block Level Elements	Inline Elements
It is used to divide the blocks and called as "Block Level Elements"	It is also used to divide the blocks and called as "Inline Elements"
It takes the Entire width	It only takes the Element width
Examples: ▪ <section> ▪ <paragraph> ▪ <Div> ▪ <table> ▪ <form>	Examples: ▪ <a> ▪ ▪ <input> ▪ <bold> ▪ <italic> ▪ <Superset> ▪ ▪ <subset>

Empty Elements:

-
- <hr>
- <img>

HTML Links

Links are found in nearly all web pages. Links allow users to click their way from page to page.

HTML Links - Hyperlinks

HTML links are hyperlinks. You can click on a link and jump to another document. When you move the mouse over a link, the mouse arrow will turn into a little hand.

Note: A link does not have to be text. It can be an image or any other HTML element.

ANCHOR TAG: <a> tag is used for linking purpose. It has a href (Hyper reference) attribute.

- Anchor tags are commonly used to link separate webpages, but they can also be used to link between different places in a single document, often within table of contents or even launch external applications .

```
<a href="https://www.facebook.com">This is a link</a>
```

- To denote that this link leads to an external website, you can use the external link type:

```
<a href="http://example.com/" rel="external">example site</a>
```

➢ **Link to a page on the same site:** You can use a relative path to link to pages on the same website.

Text Here

The above example would go to the file example at the root directory (/) of the server.

LINK TAG : This tag helps us to Connect one page to another page. And it uses the Anchor tag. We have the External link which uses the **target attribute**.

< a href="www.gmail.com" target="_blank">Gmail

- Link tag is used for getting the External Style sheet. And it should always be written in the <Head> tag.
- Link tag is always used to link the External SS to HTML Page.

< link rel =”Stylesheet” href =”path”/>

- rel : Specifies the relationship between the current document and the linked document. For HTML5, the values must be defined in the specification.

target Attribute

- The target attribute specifies if the submitted result will open in a new browser tab, a frame, or in the current window.
- The default value is "_self" which means the form will be submitted in the current window.

The target attribute can have one of the following values:

- _blank - Opens the linked document in a new window or tab
- _self - Opens the linked document in the same window/tab as it was clicked (this is default)
- _parent - Opens the linked document in the parent frame
- _top - Opens the linked document in the full body of the window
- *framename* - Opens the linked document in a named frame
- Other legal values are "_parent", "_top", or a name representing the name of an iframe.

HTML Images

- HTML images are defined with the <img> tag. Image tags (img) do not have closing tags. The two main attributes you give to the img tag are src, the image source and alt, which is alternative text describing the image.

```
<img src=" images/hello.png " alt=" Hello World " width="104" height="142">
```

- <img> tag is the EMPTY TAG i.e with no Ending tag.

- You can also get images from a web URL:

<img src="https://i.stack.imgur.com/ALgZi.jpg?s=48&g=1" alt="StackOverflow user Caleb Kleveter">

Note:

- Images are not technically inserted into an HTML page, images are linked to HTML pages. The <img> tag creates a holding space for the referenced image.
- To link an image to another document, simply nest the <img> tag inside <a> tags.

`<a>` element to define a link
`href` attribute to define the link address
`target` attribute to define where to open the linked document
`<img>` element (inside <a>) to use an image as a link
`id` attribute (id="*value*") to define bookmarks in a page
`href` attribute (href="#*value*") to link to the bookmark

HTML Lists

Unordered / bullet list	Ordered / numbered list
It is defined with the	It is defined with the
  - Coffee - Tea - Milk 	 - java - ruby - sql
▪ Coffee ▪ Tea ▪ Milk	1. Java 2. Ruby 3. sql

HTML offers two ways for specifying lists: ordered lists, unordered lists.

Ordered lists use ordinal sequences to indicate the order of list elements,

unordered lists use a defined symbol such as a bullet to list elements in no designated order,

HTML TABLES

<table> table tag:

- Each **table row** is defined with the `<tr>` tag.
- A **table header** is defined with the `<th>` tag. By default, table headings are bold and centered.
- A **table data/cell** is defined with the `<td>` tag. The `<td>` elements are the data containers of the table. They can contain all sorts of HTML elements; text, images, lists, other tables, etc.

```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>jack</td>
    <td>Smith</td>
    <td>45</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>ronald</td>
    <td>49</td>
  </tr>
</table>
```

HTML Table - Adding a Border A border is set using the CSS `border` property:

```
table, th, td {
  border: 1px solid black;
}
```

HTML Table - Collapsed Borders :If you want the borders to collapse into one border, add the CSS `border-collapse` property:

```
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
```

CELL PADDING ATTRIBUTE	**CELL SPACING ATTRIBUTE**
The cellpadding attribute specifies the space, in pixels, between the cell wall and the cell content	The cellspacing attribute specifies the space, in pixels, between cells
(Attribute) ***pixels*** = **The space between the cell wall and the cell content** (value)	(Attribute) Pixels = Space Between Cells (value)
SYNTAX: <table cellpadding="*pixels*">	SYNTAX: <table cellspacing="*pixels*">
the cellpadding attribute, which specifies the space between the cell wall and the cell content	the cellspacing attribute, which specifies the space between cells.
Table without cellpadding: **Month** | **Savings** January | $100 Table with cellpadding: **Month** | **Savings** January | $100	Table without cellspacing: **Month** | **Savings** January | $100 Table with cellspacing: **Month** | **Savings** January | $100

NOTE: The cellpadding attribute of <table> is not supported in HTML5. Use CSS instead

Cell Spanning

Cell spanning means joining cells together to make a larger cell. You can join cells horizontally, vertically, or both.

Spanning columns or rows

Table cells can span multiple columns or rows using the colspan and rowspan attributes. These attributes can be applied to <th> and <td> elements.

```
<table>
        <tr>
         <td>row 1 col 1</td>
         <td>row 1 col 2</td>
         <td>row 1 col 3</td>
        </tr>
  <tr>
    <td colspan="3">This second row spans all three columns</td>
  </tr>

     <tr>
        <td rowspan="2">This cell spans two rows</td>
        <td>row 3 col 2</td>
        <td>row 3 col 3</td>
      </tr>
    <tr>
       <td>row 4 col 2</td>
       <td>row 4 col 3</td>
   </tr>

</table>
```

Will result in

row 1 col 1	row 1 col 2	row 1 col 3
This second row spans all three columns		
This cell spans two rows	row 3 col 2	row 3 col 3
	row 4 col 2	row 4 col 3

rowspan = A non-negative integer that specifies the number of rows spanned by a cell. The default value of this attribute is one (1). A value of zero (0) means that the cell will extend from the current row until the last row of the table (<thead>, <tbody>, or <tfoot>).

colspan = A non-negative integer that specifies the number of columns spanned by the current cell. The default value of this attribute is one (1). A value of zero (0) means that the cell will extend from the current to the last column of the column group <colgroup> in which the cell is defined.

HTML FRAMES:

- The <frame> tag defines one particular window (frame) within a <frameset>.
- Each <frame> in a <frameset> can have different attributes, such as border, scrolling, the ability to resize, etc.
- When you use frameset you split the visual real estate of a browser window into multiple frames. Each frame has it's own contents and the content in one don't spill into the next.
- An iframe, on the other hand, embeds a frame directly inline with the other elements of a webpage.

While both frames and iframes perform a similar function – embedding a resource into a webpage – they are fundamentally different.

- Frames are layout-defining elements.
- Iframes are a content-adding elements.

How to create a Frames…?

The basic concept behind frames is pretty simple:

- Use the frameset element in place of the body element in an HTML document.
- Use the frame element to create frames for the content of the web page.
- Use the src attribute to identify the resource that should be loaded inside each frame.
- Create a different file with the contents for each frame.

VERTICAL COLUMNS FRAME	HORIZONTAL ROWS FRAME
We use the frameset element with the cols attribute. The cols attribute is used to define the number and size of columns the frameset will contain	We use the frameset element with the rows attribute. The rows attribute is used to define the number and size of columns the frameset will contain

<html> <frameset cols="*,*,*,*"> <frame src="../file_path/frame_1.html"> <frame src="frame_2.html"> <frame src="frame_3.html"> <frame src="frame_4.html"> </frameset> </html>	<html> <frameset rows="*,*,*,*"> <frame src="../file_path/frame_1.html"> <frame src="frame_2.html"> <frame src="frame_3.html"> <frame src="frame_4.html"> </frameset> </html>

Here we have different types of frames:

Grid Frames , Mixed Frames , Nested Frames .

Note:

- If you want to validate a page containing frames, be sure the <!DOCTYPE> is set to either "HTML Frameset DTD" or "XHTML Frameset DTD".
- The <frame> tag is not supported in HTML5.
- In HTML, the <frame> tag has no end tag. In XHTML, the <frame> tag must be properly closed.

HTML FORMS

- The HTML `<form> tag` elements helps to collect the user information/ Data.
- It is a paired tag <form></form> And is a **Block Level Element.**
- An HTML form contains **form elements**.
- Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

HTML INPUT TYPES:

<input> Element

- The `<input>` element is the most important form element.
- The `<input>` element can be displayed in several ways, depending on the **type** attribute.

Type	Description
<input type="text">	Defines a one-line text input field
<input type="radio">	Defines a radio button (for selecting one of many choices)
<input type="submit">	Defines a submit button (for submitting the form)

1.Text Input

- `<input type="text">` defines a one-line input field for **text input**.
- the default width of a text field is 20 characters.
- **Example**

```
<form>
   First name:<br>
  <input type="text" name="firstname"><br>
   Last name:<br>
  <input type="text" name="lastname">
</form>
```

2.Input Type Password

- `<input type="password">` defines a **password field**.
- The characters in a password field are masked (shown as asterisks or circles).

Example:

```
<form>
  User name:<br>
  <input type="text" name="username"><br>
  User password:<br>
  <input type="password" name="psw">
 </form>
```

3.Radio Button Input

- `<input type="radio">` defines a **radio button**.
- Radio buttons let a user select ONE of a limited number of choices:

Example

```
<form>
 <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```

4.Input Submit

- `<input type="submit">` defines a button for **submitting** the form data to a **form-handler**.
- The form-handler is typically a server page with a script for processing input data.
- The form-handler is specified in the form's **action** attribute:
- **Example**

```
<form action="/action_page.html">
 First name:<br>
<input type="text" name="firstname" value="tinga"><br>
Last name:<br>
 <input type="text" name="lastname" value="dinga"><br><br>
<input type="submit" value="Submit">
 </form>
```

5.Input Type Reset

- `<input type="reset">` defines a **reset button** that will reset all form values to their default values.
- `Example:`

```
<form action="/action_page.html">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
  <input type="reset">
</form>
```

6.Input Type Button

- `<input type="button">` defines a **button**.
- `Example:`

```
<input type="button" onclick="alert('Hello World!')" value="Click
Me!">
```

7.Input Type Checkbox

- `<input type="checkbox">` defines a **checkbox**.
- Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<form>
  <input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>
  <input type="checkbox" name="vehicle2" value="Car"> I have a car
</form>
```

8.Input Type File

- The `<input type="file">` defines a file-select field and a "Browse" button for file uploads.

```
Example:   <form>
       Select a file: <input type="file" name="myFile">
         </form>
```

HTML INPUT TYPES:

1.readonly Attribute

- The `readonly` attribute specifies that the input field is read only (cannot be changed).
- ```
  <form action="#">
    First name:<br>
    <input type="text" name="firstname" value="John" readonly>
  </form>
  ```

2.disabled Attribute

- The `disabled` attribute specifies that the input field is disabled.
  ```

- A disabled input field is unusable and un-clickable, and its value will not be sent when submitting the form.
- Example:

```
<form action="#">
 First name:<br>
 <input type="text" name="firstname" value="John" disabled>
</form>
```

3.novalidate Attribute

- The `novalidate` attribute is a `<form>` attribute.
- When present, novalidate specifies that the form data should not be validated when submitted.
- Example:

```
<form action="/action_page.html" novalidate>
 E-mail: <input type="email" name="user_email">
 <input type="submit">
</form>
```

4.form Attribute

- The `form` attribute specifies one or more forms an `<input>` element belongs to.
- Example:

```
<form action="/action_page.html" id="form1">
 First name: <input type="text" name="fname"><br>
 Last name: <input type="text" name="lname" form="form1">

 <input type="submit" value="Submit">
 </form>
```

5.placeholder Attribute

- The `placeholder` attribute specifies a hint that describes the expected value of an input field (a sample value or a short description of the format).
- The hint is displayed in the input field before the user enters a value.
- The `placeholder` attribute works with the following input types: text, search, url, tel, email, and password.
- Ex:

```
<input type="text" name="fname" placeholder="First name">
```

6.required Attribute

- The `required` attribute specifies that an input field must be filled out before submitting the form.
- The `required` attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.
- Example:

```
Username: <input type="text" name="username" required>
```

7.Action Attribute

- The `action` attribute defines the action to be performed when the form is submitted.
- Normally, the form data is sent to a web page on the server when the user clicks on the submit button.
- In the example above, the form data is sent to a page on the server called "/action_page.php". This page contains a server-side script that handles the form data:
- **Example :**

```
<form action="/action_page.php">
```

If the `action` attribute is omitted, the action is set to the current page.

8.Method Attribute

The `method` attribute specifies the HTTP method (**GET** or **POST**) to be used when submitting the form data:

Example: `<form action="#" method="get">`

NOTE:

1.get ():

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)

- Useful for form submissions where a user wants to bookmark the result
- GET is better for non-secure data, like query strings in Google

2.post ():

- Always use POST if the form data contains sensitive or personal information.
- The POST method does not display the submitted form data in the page address field.
- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked

Grouping Form Data with <fieldset>

- The `<fieldset>` element is used to group related data in a form.
- The `<legend>` element defines a caption for the `<fieldset>` element.
- Example:

```
<form action="/action_page.html">
  <fieldset>
    <legend>Personal information:</legend>
    First name:<br>
    <input type="text" name="firstname" value="Mickey"><br>
    Last name:<br>
    <input type="text" name="lastname" value="Mouse"><br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

Attribute	Description
accept-charset	Specifies the charset used in the submitted form (default: the page charset).
action	Specifies an address (url) where to submit the form (default: the submitting page).
autocomplete	Specifies if the browser should autocomplete the form (default: on).
enctype	Specifies the encoding of the submitted data (default: is url-encoded).
method	Specifies the HTTP method used when submitting the form (default: GET).
name	Specifies a name used to identify the form (for DOM usage: document.forms.name).
novalidate	Specifies that the browser should not validate the form.
target	Specifies the target of the address in the action attribute (default: _self).

HTML FORM ELEMENTS:

1.<input> Element

- The most important form element is the `<input>` element.
- The `<input>` element can be displayed in several ways, depending on the `type` attribute.

```
<input name="firstname" type="text">
```

2.<select> Element

- The `<select>` element defines a **drop-down list**.
- The `<option>` elements defines an option that can be selected.
- By default, the first item in the drop-down list is selected.
- To define a pre-selected option, add the `selected` attribute to the option
- **Visible Values:** Use the `size` attribute to specify the number of visible values.
- Use the `multiple` attribute to allow the user to select more than one value:

Example:

```
<select name="languages" size="4" multiple>
  <option value="java">java</option>
  <option value="html">html</option>
```

```
    <option value="sql">sql</option>
    <option value="angularjs">Angularjs</option>
  </select>
```

3.<textarea> Element

- The `<textarea>` element defines a multi-line input field (**a text area**).
- The `rows` attribute specifies the visible number of lines in a text area.
- The `cols` attribute specifies the visible width of a text area.

Example:

```
<textarea name="message" rows="10" cols="30">
    The cat was playing in the garden.
</textarea>
```

4.<button> Element

- The `<button>` element defines a clickable **button**.
- Always specify the **type** attribute for the button element. Different browsers may use different default types for the button element
- **Example:**

```
<button>Click Me</button>
```

```
<button type="button" onclick="alert('Hello World!')">Click Me!</button>
```

5.<label> Element

The <label> element is used to reference a form action element. In the scope of User Interface it's used to ease the target / selection of elements like Type radio or checkbox.

<label> as wrapper

It can enclose the desired action element

<label> <input type="checkbox" name="Cats"> I like Cats! </label>

(Clicking on the text the target input will toggle it's state / value)

<label> as reference

Using the for attribute you don't have to place the control element as descendant of label - but the for value must match it's ID

<input id="cats" type="checkbox" name="Cats"> <label for="cats" >I like Cats!</label>

Note: Don't use more than one Control Element within a <label> element

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<button>	Defines a clickable button

NEW HTML FORM ELEMENTS:

1.<datalist> Element

- The `<datalist>` element specifies a list of pre-defined options for an `<input>` element.
- Users will see a drop-down list of the pre-defined options as they input data.
- The `list` attribute of the `<input>` element, must refer to the `id` attribute of the `<datalist>` element.

Example :

```
<form action="/action_page.html">
<input list="browsers">
<datalist id="browsers">
<option value="Internet Explorer">
<option value="Firefox">
```

```
<option value="Chrome">
<option value="Opera">
<option value="Safari">
</datalist>
</form>
```

2.date Element

- The `<input type="date">` is used for input fields that should contain a date.
- Depending on browser support, a date picker can show up in the input field.
- Example:

```
<form>
Birthday:
<input type="date" name="bday">
</form>
```

- You can also use the `min` and `max` attributes to add restrictions to dates:

Example:

```
<form>
Enter a date before 1980-01-01:
<input type="date" name="bday" max="1979-12-31"><br>
Enter a date after 2000-01-01:
<input type="date" name="bday" min="2000-01-02"><br>
 </form>
```

3.Datetime-local Element

- The `<input type="datetime-local">` specifies a date and time input field, with no time zone.
- Depending on browser support, a date picker can show up in the input field.

Example:

```
<form>
 Birthday (date and time):
 <input type="datetime-local" name="bdaytime">
 </form>
```

4.email Element

- The `<input type="email">` is used for input fields that should contain an e-mail address.

- Depending on browser support, the e-mail address can be automatically validated when submitted.
- Some smartphones recognize the email type, and add ".com" to the keyboard to match email input.

Example:
```
<form>
E-mail:
<input type="email" name="email">
</form>
```

5. Number Element

- The `<input type="number">` defines a **numeric** input field.
- You can also set restrictions on what numbers are accepted.
- The following example displays a numeric input field, where you can enter a value from 1 to 5:
- Example:
```
<form>
  Quantity (between 1 and 5):
<input type="number" name="quantity" min="1" max="5">
  </form>
```

6. Range Element

- The `<input type="range">` defines a control for entering a number whose exact value is not important (like a slider control).
- Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the `min`, `max`, and `step` attributes:
- Example:
```
<form>
<input type="range" name="points" min="0" max="10">
 </form>
```

7.search Element

- The `<input type="search">` is used for search fields (a search field behaves like a regular text field).
- Example:
```
<form>
Search Google:
<input type="search" name="googlesearch">
</form>
```

8.Time Element

- The `<input type="time">` allows the user to select a time (no time zone).
- Depending on browser support, a time picker can show up in the input field.
- Example:

```
<form>
Select a time:
<input type="time" name="usr_time">
</form>
```

9.Url Element

- The `<input type="url">` is used for input fields that should contain a URL address.
- Depending on browser support, the url field can be automatically validated when submitted.
- Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.
- Example:

```
<form>
Add your homepage:
<input type="url" name="homepage">
</form>
```

Text formatting

While most HTML tags are used to create elements, HTML also provides in-text formatting tags to apply specific text-related styles to portions of text. This topic includes examples of HTML text formatting such as highlighting, bolding, underlining, subscript, and stricken text.

Section 5.1: Highlighting

- The <mark> element is new in HTML5 and is used to mark or highlight text in a document "due to its relevance in another context".
- The most common example would be in the results of a search were the user has entered a search query and results are shown highlighting the desired query.

Here is some content from an article that contains the <mark>searched query</mark> that we are looking for. Highlighting the text will make it easier for the user to find what they are looking for.

Output:

Here is content from an article that contains the searched query that we are looking for.

- A common standard formatting is black text on a yellow background, but this can be changed with CSS.

Section 5.2: Bold, Italic, and Underline

1.Bold Text: To bold text, use the or tags:

Bold Text Here

or

Bold Text Here

What's the difference? Semantics.

 is used to indicate that the text is fundamentally or semantically important to the surrounding text, while

 indicates no such importance and simply represents text that should be bolded.

2.Italic Text: To italicize text, use the or tags:

Italicized Text Here

or

Italicized Text Here

What's the difference? Semantics.

 is used to indicate that the text should have extra emphasis that should be stressed, while

<i> simply represents text which should be set off from the normal text around it.

3.Underlined Text

While the <u> element itself was deprecated in HTMl 4, it was reintroduced with alternate semantic meaning in HTML 5 - to represent an unarticulated, non-textual annotation. You might use such a rendering to indicate misspelled text on the page, or for a Chinese proper name mark.

This paragraph contains some <u>misspelled</u> text.

Superscript and Subscript

To offset text either upward or downward you can use the tags <sup> and <sub>.

1.To create superscript: <sup>

^{superscript here}

2. To create subscript: <sub>

_{subscript here}

HTML 5

- HTML5 combines **HTML, CSS3 and Java script** technologies.
- 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.

Offerings of HTML 5

The Rules behind HTML5

- New features should be based on HTML, CSS, DOM, and JavaScript.
- Reduce the need for external plugins (like Flash).
- Better error handling.
- More markup to replace scripting.
- HTML5 should be device independent.
- The development process should be visible to the public

HTML5 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.

HTML5 Syntax

- HTML 5 does not have the same syntax rules as XHTML where we needed lower case tag names, quoting our attributes, an attribute had to have a value and to close all empty elements.

- HTML5 comes with a lot of flexibility and it supports the following features:
- Uppercase tag names.
- Quotes are optional for attributes.
- Attribute values are optional.
- Closing empty elements are optional.

- The DOCTYPE declaration for HTML5 is very simple:

```
<!DOCTYPE html>
```

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

```
<meta charset="UTF-8">
```

- EXAMPLE:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>
<body>
Content of the document......
</body>
</html>
```

New Features of HTML 5

New Features HTML5 introduces a number of new elements and attributes that can help you in building modern websites. Here is a set of some of the most prominent features introduced in HTML5.

- **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 the same webpage.

- **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 webpages without resorting to third-party plugins.

NEW HTML Elements:

The most interesting **new HTML5 elements** are:

1. New **semantic elements** like `<header>`, `<footer>`, `<article>`, and `<section>`.
2. New **attributes of form elements** like number, date, time, calendar, and range.
3. New **graphic elements**: `<svg> scalable vector graphics` and `<canvas>`.
4. New **multimedia elements**: `<audio>` and `<video>`.

New HTML5 API's (Application Programming Interfaces)

The most interesting new API's in HTML5 are:

- HTML Geolocation
- HTML Drag and Drop
- HTML Local Storage
- HTML Application Cache
- HTML Web Workers
- HTML SSE (server sent events)

Tip: HTML Local storage is a powerful replacement for cookies.

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.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Title of the document</title>
    </head>
    <body>
        <header>.....</header>
         <nav>.....</nav>
      <article>
            <section>
             .....
            </section>
       </article>

    <aside> .....</aside>
    <footer>.....</footer>
</body>
</html>
```

NEW HTML5 FORM ELEMENTS:

1.<datalist> Element

- The `<datalist>` element specifies a list of pre-defined options for an `<input>` element.
- Users will see a drop-down list of the pre-defined options as they input data.
- The `list` attribute of the `<input>` element, must refer to the `id` attribute of the `<datalist>` element.

Example :

```
<form action="/action_page.html">
<input list="browsers">
<datalist id="browsers">
<option value="Internet Explorer">
<option value="Firefox">
<option value="Chrome">
<option value="Opera">
<option value="Safari">
</datalist>
</form>
```

NEW INPUT TYPES:

New Input Types	New Input Attributes
• color • date • datetime • datetime-local • email • month • number • range • search • tel • time • url • week	• autocomplete • autofocus • form • formaction • formenctype • formmethod • formnovalidate • formtarget • height and width • list • min and max • multiple • pattern (regexp) • placeholder • required • step

HTML5 - New Attribute Syntax

HTML5 allows four different syntaxes for attributes. We use the <input>

Type	Example
Empty	<input type="text" value="John" **disabled**>
Unquoted	<input type="text" **value=John**>
Double-quoted	<input type="text" **value="John Doe"**>
Single-quoted	<input type="text" **value='John Doe'**>

HTML5 - New Media Elements

1. HTML 5 AUDIO

Audio on the Web:

- Before HTML5, audio files could only be played in a browser with a plug-in (like flash).
- The HTML5 `<audio>` element specifies a standard way to embed audio in a web page.

The HTML5 <audio> Element:

- The <source> tag is used to specify multiple media resources for media elements, such as <video>, <audio>, and <picture>.
- The <source> tag allows you to specify alternative video/audio/image files which the browser may choose from, based on its media type, codec support or media query.
- The <audio> tag defines sound, such as music or other audio streams.
- To play an audio file in HTML, use the `<audio>` element:

Example:

```
<audio controls>
<source src="horse.ogg" type="audio/ogg">
<source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

How HTML5 Audio Works..?

- The `controls` attribute adds audio controls, like play, pause, and volume.
- The `<source>` element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.
- The text between the `<audio>` and `</audio>` tags will only be displayed in browsers that do not support the `<audio>` element.

- In HTML5, there are 3 supported audio formats: **MP3, WAV, and OGG**.(all 3 works with google, firefox ,opera)

HTML5 Audio - Media Types

File Format	Media Type
MP3	audio/mpeg
OGG	audio/ogg
WAV	audio/wav

HTML5 Audio - Methods, Properties, and Events

- HTML5 defines DOM methods, properties, and events for the `<audio>` element.
- This allows you to load, play, and pause audios, as well as set duration and volume.
- There are also DOM events that can notify you when an audio begins to play, is paused, etc.

HTML5 Audio Tags

Tag	Description
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements, such as <video> and <audio>

HTML 5 VIDEO

Playing Videos in HTML

- Before HTML5, a video could only be played in a browser with a plug-in (like flash).
- The HTML5 `<video>` element specifies a standard way to embed a video in a web page.

The HTML <video> Element

- To show a video in HTML, use the `<video>` element:

Example :

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  Your browser does not support the video tag.
  </video>
```

How it Works

- The `controls` attribute adds video controls, like play, pause, and volume.
- It is a good idea to always include `width` and `height` attributes. If height and width are not set, the page might flicker while the video loads.
- The `<source>` element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.
- The text between the `<video>` and `</video>` tags will only be displayed in browsers that do not support the `<video>` element.

HTML5 <video> Autoplay

To start a video automatically use the `autoplay` attribute:

Example:

```
<video width="320" height="240" autoplay>
<source src="movie.mp4" type="video/mp4">
<source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

Note: The autoplay attribute does not work in mobile devices like iPad and iPhone.

HTML Video - Browser Support

In HTML5, there are 3 supported video formats: MP4, WebM, and Ogg. The browsers which support are Chrome, firefox, Opera(from opera 25) for all above video formats.

HTML Video - Methods, Properties, and Events

- HTML5 defines DOM methods, properties, and events for the `<video>` element.
- This allows you to load, play, and pause videos, as well as setting duration and volume.
- There are also DOM events that can notify you when a video begins to play, is paused, etc.

Html5 video tags

Tag	Description
<video>	Defines a video or movie
<source>	Defines multiple media resources for media elements, such as <video> and <audio>
<track>	Defines text tracks in media players

1.<video> tag:

- The <video> tag specifies video, such as a movie clip or other video streams.
- Currently, there are 3 supported video formats for the <video> element: MP4, WebM, and Ogg
- Example:

```
<video width="320" height="240" controls>
<source src="movie.mp4" type="video/mp4">
<source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

2.<source> tag:

- The <source> tag is used to specify multiple media resources for media elements, such as <video>, <audio>, and <picture>.
- The <source> tag allows you to specify alternative video/audio/image files which the browser may choose from, based on its media type, codec support or media query.

Example:

```
<picture>
  <source media="(min-width: 650px)" srcset="img_pink_flowers.jpg">

  <source media="(min-width: 465px)" srcset="img_white_flower.jpg">

<img src="img_orange_flowers.jpg" alt="Flowers" style="width:auto;">
</picture>
```

Attribute	Value	Description
src	*URL*	Required when <source> is used in <audio> and <video>. Specifies the URL of the media file
srcset	*URL*	Required when <source> is used in <picture>. Specifies the URL of the image to use in different situations
media	*media_query*	Accepts any valid media query that would normally be defined in a CSS
sizes		Specifies image sizes for different page layouts
type	*MIME-type*	Specifies the MIME-type of the resource

3.<track> tag:

- The <track> tag specifies text tracks for media elements (<audio> and <video>).
- This element is used to specify subtitles, caption files or other files containing text, that should be visible when the media is playing.

- Example:

```
    <video width="320" height="240" controls>
  <source src="forrest_gump.mp4" type="video/mp4">
  <source src="forrest_gump.ogg" type="video/ogg">
<track src="subtitles_en.vtt" kind="subtitles" srclang="en" label="Engl
ish">
  <track src="subtitles_no.vtt" kind="subtitles" srclang="no" label="No
```

```
rwegian">
</video>
```

kind	captions chapters descriptions metadata subtitles	Specifies the kind of text track
label	*text*	Specifies the title of the text track
src	*URL*	Required. Specifies the URL of the track file
srclang	*language_code*	Specifies the language of the track text data (required if kind="subtitles")

HTML 5 Detection Library : Modernizer

What is modernizr?

- Modernizr is a small **JavaScript Library** that detects the availability of native implementations for next-generation web technologies
- There are several new features which are being introduced through HTML5 and CSS3 but same time many browsers do not support these news features.
- Modernizr provides an easy way to detect any new feature so that you can take corresponding action. For example, if a browser does not support video feature then you would like to display a simple page.
- You can create CSS rules based on the feature availability and these rules would apply automatically on the webpage if browser does not support a new feature.

Why do I need it…?

- All web developers come up against differences between browsers and devices.
- That's largely due to different feature sets the latest versions of the popular browsers can do some awesome things which older browsers can't– but we still have to support the older ones

- Modernizr makes it easy to deliver tiered experiences: make use of the latest and greatest features in browsers which support them, without leaving less fortunate users high and dry.

Syntax:

Before you start using Modernizr, you would have to include its javascript library in your HTML page header as follows –

```
<script src="modernizr.min.js" type="text/javascript"></script>
```

As mentioned above, you can create CSS rules based on the feature availability and these rules would apply automatically on the webpage if browser does not support a new feature.

```
/* In your CSS: */
.no-audio #music {
   display: none; /* Don't show Audio options */
}
.audio #music button {
   /* Style the Play and Pause buttons nicely */
}

<!-- In your HTML: -->
<div id="music">

   <audio>
      <source src="audio.ogg" />
      <source src="audio.mp3" />
   </audio>
```

```
   <button id="play">Play</button>
   <button id="pause">Pause</button>
</div>
```

Here it is notable that you need to prefix "no-" to every feature/property you want to handle for the browser which does not support them.

Following is the syntax to detect a particular feature through Javascript

```
if (Modernizr.audio) {
   /* properties for browsers that
   support audio */
}

else{
   /* properties for browsers that
   does not support audio */
}
```

Features Detected by the Modernizr:

Few of them are listed below

Feature	CSS Property	JavaScript Check
@font-face	.fontface	Modernizr.fontface
Canvas	.canvas	Modernizr.canvas
Canvas Text	.canvastext	Modernizr.canvastext
HTML5 Audio	.audio	Modernizr.audio
HTML5 Audio formats	NA	Modernizr.audio[format]
HTML5 Video	.video	Modernizr.video
HTML5 Video Formats	NA	Modernizr.video[format]

HTML 5 CANVAS

- HTML5 element <canvas> gives you an easy and powerful way to draw graphics using JavaScript.
- It can be used to draw graphs, make photo compositions or do simple (and not so simple) animations.
- The HTML `<canvas>` element is used to draw graphics on a web page.
- The graphic to the left is created with `<canvas>`. The `<canvas>` element is only a container for graphics. You must use JavaScript to actually draw the graphics.
- It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

- Canvas has several methods for drawing paths, boxes, circles, text, and adding images

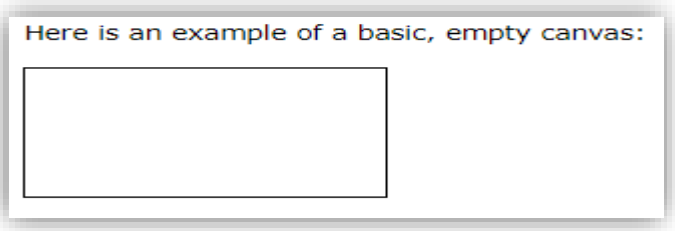
Canvas Examples

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

The markup looks like this:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Note: Always specify an `id` attribute (to be referred to in a script), and a `width` and `height` attribute to define the size of the canvas. To add a border, use the `style` attribute.



For empty canvas:

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#000000;">
</canvas>
```

Draw a Line:

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.</canvas>

<script>
   var c = document.getElementById("myCanvas");
   var ctx = c.getContext("2d");
  ctx.moveTo(0,0);
  ctx.lineTo(200,100);
  ctx.stroke();
</script>
```

Draw a circle:

```
var c = document.getElementById("myCanvas");
 var ctx = c.getContext("2d");
  ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
```

Draw a text :

```
 var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
```

HTML 5 – Client Side Storage:

HTML5 introduces two mechanisms, similar to HTTP session cookies, for storing structured data on the client side and to overcome following drawbacks.

- Cookies are included with every HTTP request, thereby slowing down your web application by transmitting the same data.
- Cookies are included with every HTTP request, thereby sending data unencrypted over the internet.
- Cookies are limited to about 4 KB of data. Not enough to store required data.

TYPES OF CLIENT SIDE STORAGE:

- The two storages are **session storage** and **local storage** and they would be used to handle different situations.
- The latest versions of pretty much every browser supports HTML5 Storage including Internet Explorer.

1.Session Storage

The *Session Storage* is designed for scenarios where the user is carrying out a single transaction, but could be carrying out multiple transactions in different windows at the same time.

For example,

if a user buying plane tickets in two different windows, using the same site. If the site used cookies to keep track of which ticket the user was buying, then as the user clicked from page to page in both windows, the ticket currently being purchased would "leak" from one window to the other, potentially causing the user to buy two tickets for the same flight without really noticing.

- HTML5 introduces the ***session Storage*** attribute which would be used by the sites to add data to the session storage, and it will be accessible to any page from the same site opened in that window, i.e., **session** and as soon as you close the window, the session would be lost.

Following is the code which would set a session variable and access that variable

```
<!DOCTYPE HTML>

<html>

   <body>

      <script type = "text/javascript">


if( sessionStorage.hits ) {
```

```
            sessionStorage.hits = Number(sessionStorage.hits) +1;
         } else {
            sessionStorage.hits = 1;
         }
         document.write("Total Hits :" + sessionStorage.hits );
      </script>
        <p>Refresh the page to increase number of hits.</p>
      <p>Close the window and open it again and check the result.</p>
</body>
</html>
```

This will produce the following result :

Total Hits :1

Refresh the page to increase number of hits.

Close the window and open it again and check the result.

2.Local Storage

- The *Local Storage* is designed for storage that spans multiple windows, and lasts beyond the current session.
- In particular, Web applications may wish to store megabytes of user data, such as entire user-authored documents or a user's mailbox, on the client side for performance reasons.

Again, cookies do not handle this case well, because they are transmitted with every request.

Example

HTML5 introduces the ***local Storage*** attribute which would be used to access a page's local storage area without no time limit and this local storage will be available whenever you would use that page.

Following is the code which would set a local storage variable and access that variable every time this page is accessed, even next time, when you open the window

```
<!DOCTYPE HTML>

<html>

   <body>

      <script type = "text/javascript">


         if( localStorage.hits ) {

            localStorage.hits = Number(localStorage.hits) +1;

         } else {

            localStorage.hits = 1;

         }

         document.write("Total Hits :" + localStorage.hits );

      </script>


      <p>Refresh the page to increase number of hits.</p>

      <p>Close the window and open it again and check the result.</p>


   </body>

</html>
```

This will produce the following result:

Total Hits :2

Refresh the page to increase number of hits.

Close the window and open it again and check the result.

Delete Web Storage

- **Storing sensitive data on local machine could be dangerous and could leave a security hole.**

The *Session Storage Data* would be deleted by the browsers immediately after the session gets terminated.

- To clear a local storage setting you would need to call **localStorage.remove('key')**; where 'key' is the key of the value you want to remove.
- If you want to clear all settings, you need to call **localStorage.clear()** method.

Following is the code which would clear complete local storage :

```
<!DOCTYPE HTML>

<html>

   <body>


      <script type = "text/javascript">

         localStorage.clear();


         // Reset number of hits.

         if( localStorage.hits ) {

            localStorage.hits = Number(localStorage.hits) +1;

         } else {

            localStorage.hits = 1;

         }

         document.write("Total Hits :" + localStorage.hits );
```

```
      </script>

      <p>Refreshing the page would not to increase hit counter.</p>
      <p>Close the window and open it again and check the result.</p>

   </body>
</html>
```

This will produce following result

Total Hits :1

Refreshing the page would not to increase hit counter.

Close the window and open it again and check the result.

.....