



Unit-I

Hyper Text Markup Language (HTML) is a standard markup language to create the structure of a web page. It annotates the content on a web page using HTML elements.

In a web page, all instructions to the browser are given in the form of HTML tags, also known as HTML elements.

The content of the web page will be rendered as per the HTML tags in which they are enclosed.

Observe the below picture.

Input: Hello World!

Output: Hello World!

Plain content

Annotated content

Content enclosed with `<h1>` element

```
<h1> Hello World! </h1>
```

The text content "Hello World!" is annotated to the heading, increasing the font-weight when it is enclosed in the HTML tag `<h1>`.

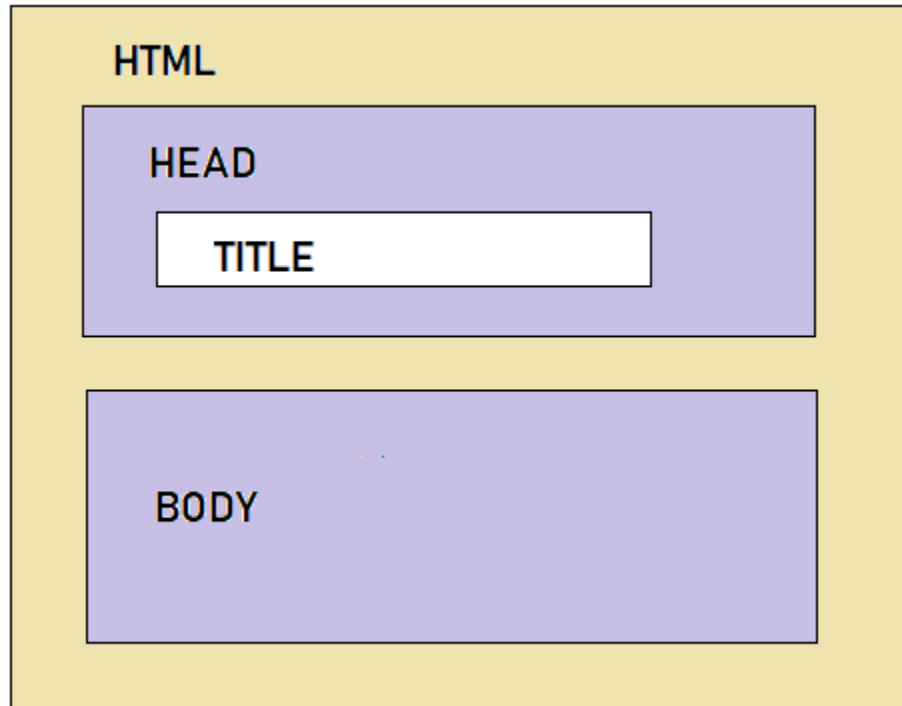
Likewise, the look and feel of the web page can be defined and the content can be organized on the web page with the help of various HTML elements.

Now, let us see the document structure of HTML.

HTML document structure tells the browser how to render the text written in each of the HTML elements of the web page.



Consider that we need to create a web page, the basic HTML document structure will be



as below:

The structure of an HTML document is defined using HTML tags.

Below is the basic structure of a simple HTML document or web page:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title></title>
```

```
</head>
```

```
<body>
```

```
</body>
```

```
</html>
```

The following are some key elements in HTML that form the basic structure of a web page.



- `<!DOCTYPE html>` declaration update the browser about the version of HTML being used. By default, it points to HTML5, the latest version.
- The `<html>` tag encapsulates the complete web page content. All other tags are 'nested' within the `<HTML>` tag.
- Any HTML document or web page consists of two main sections the 'head' and the 'body'.
 - The head section starts with the start tag `<head>` and ends with the end tag `</head>`.

The following elements can be provided within the head tag.

| Tags | Description |
|-----------------------------|--|
| <code><title></code> | Defines the title that should be displayed on the browser tab |
| <code><meta></code> | Metadata is in-general, data about data. Provides metadata about the HTML document. Metadata will not be displayed on the page but will be machine-readable. Used to specify page description, author of the document, last modified, etc. Used by browsers (control how to display content or reload the page), search engines (keywords), or other web services. Post HTML5, meta tag also allows web designers to take control over the viewport by setting the meta viewport tag. |
| <code><style></code> | Defines style information for the web page |
| <code><link></code> | Defines a link to other documents like CSS |
| <code><script></code> | Defines script like JavaScript |

- The body section is denoted within the start tag `<body>` and ends with the end tag `</body>`. This section contains actual web page content like text, images, etc.

Every HTML document/web page will have only one set of

- `<html>...</html>` tag
- `<head>...</head>` tag
- `<body>...</body>` tag

HTML document/web page is saved with .htm or .html extension

Case-insensitivity

HTML elements are case-insensitive. The browser understands the HTML tags irrespective of their cases.



Consider the code snippets below, copy the snippets to your workspace and observe the output

Code 1:

```
<!DOCTYPE html>

<html>

  <head>

    <title>Homepage </title>

  </head>

  <body>

    Hello World!

  </body>

</html>
```

Code2:

```
<!DOCTYPE html>

<html>

  <head>

    <title>Homepage </title>

  </head>

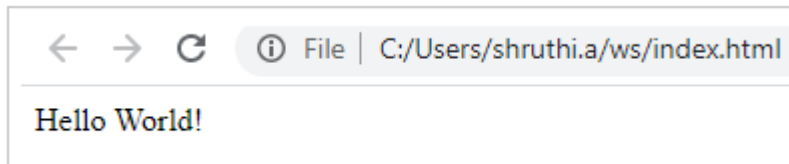
  <body>

    Hello World!

  </boDy>

</html>
```

You can observe that both codes generate the same output as below:



Platform-independency

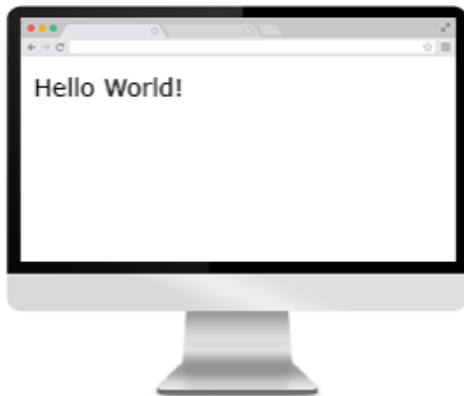
HTML Language is platform-independent. That means the same HTML code can run on different operating systems as shown below.

Sample.html

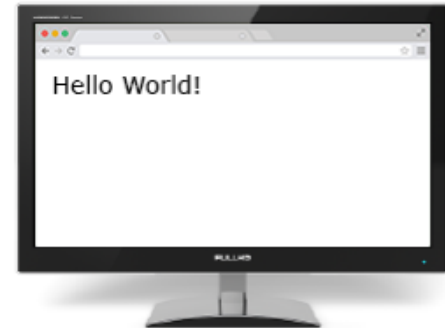
```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.     <title>sample page</title>
5. </head>
6. <body>
7.     <p>Hello World!</p>
8. </body>
9. </html>
10.
11.
```

Output:

On executing the above-mentioned code we can observe the same output in different platforms as shown below:



Machintosh



Windows

Cross-platform support

There are many versions of HTML out there such as - HTML 2.0, HTML 3.0, HTML 3.2, HTML4.0, HTML 4.01 and latest is HTML5.0. In each version, some elements and attributes are either added or depreciated. The appearance of your .html page depends on how the browser renders HTML elements. And how the browser renders HTML elements depends on how the browser understands them.

Thus, to ensure that the browser understands all HTML elements specific to a particular version, as a developer you need to tell the browser what version of HTML you have followed while developing your web page.

This is done by using `<!DOCTYPE>` declaration which stands for Document Type. It tells the browser what version of HTML it should follow for rendering the web page.

Syntax: `<!DOCTYPE html>`

HTML file begins with `<!DOCTYPE>` declaration as below:

Sample.html:

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
```

```
4. <title> Sample page </title>
5. </head>
6. <body>
7.     Hello world!
8. </body>

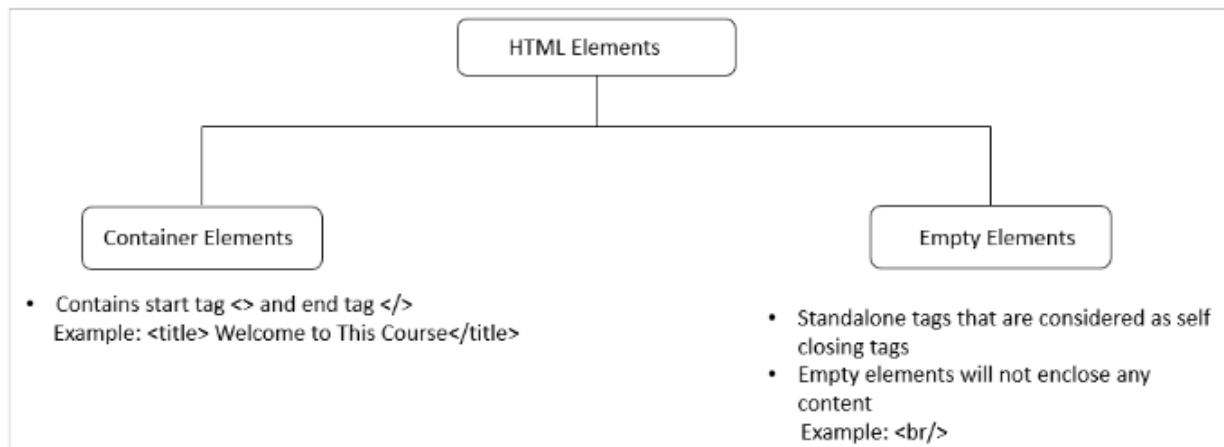
9. </html>
```

In the above code, `<!DOCTYPE html>` signifies that, the code is written in HTML5.

Best Practice:

Provide a proper DOCTYPE declaration while designing an HTML web page, so that browser can understand the version and interpret elements of the web page appropriately.

HTML elements can be broadly categorized into two as below:



HTML elements can be further categorized into two as below:

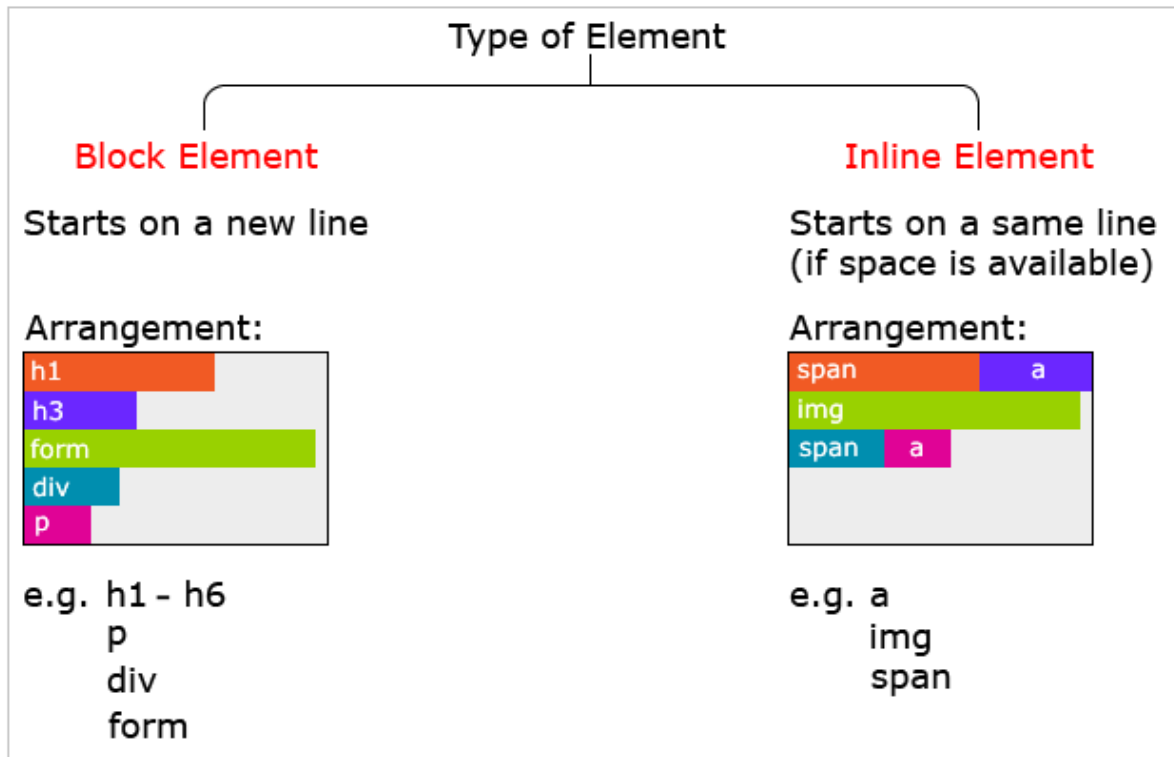
Block Element:

A block element begins on a new line occupying the entire width of the parent tag.

Inline Element:

An inline element occupies the necessary space to accommodate the content in the element. Inline elements can be nested within other inline elements, whereas, block elements cannot be nested within inline elements.

Some of the examples are illustrated below:



HTML elements can contain attributes that can be considered as an additional feature to set various properties and they are optional.

Some of the attributes can be used with any of the HTML elements and there can be referred to as 'global attributes'. Also, some attributes can be used only with particular elements. Following are some features of attributes:

- All the attributes can contain properties like name and value which can be used by a developer to assign respective details for that HTML elements.
- Attributes are to be set only in the start tag of a container HTML element.
- Attributes are case-insensitive, but it is recommended to use lowercase as a best practice.
- The best practice is always to quote attribute value even though we will not get any execution errors if they are not provided in quotes.
- **Example:**
- The lang attribute specifies the language of the content of the HTML page.

Syntax: `<html lang="en-US">`

↑
Specifies that the content of
.html page is written in U.S.
version of English language

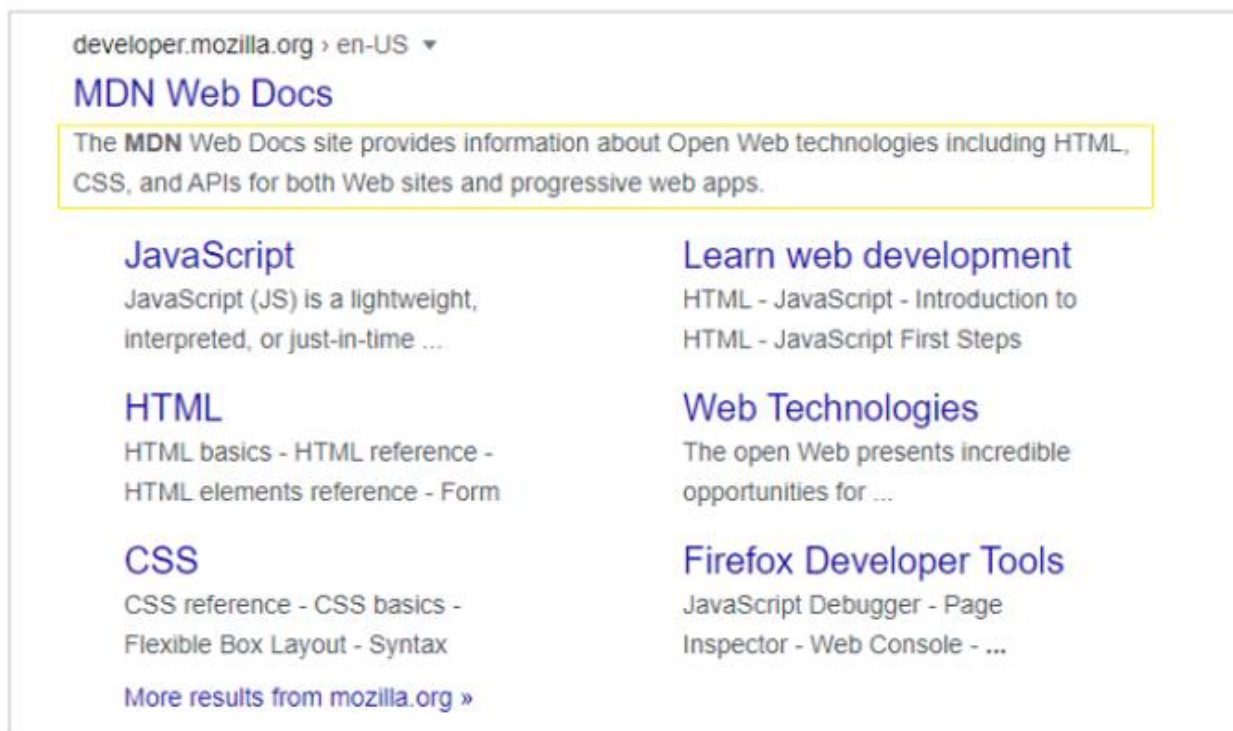
Comment

As a developer, you may want to document your code, so that you can easily refer to it in the future.

For this, comments are used.

Syntax: `<!-- This line is commented -->`

Meta Elements:



The screenshot shows the MDN Web Docs homepage. At the top, it says 'developer.mozilla.org > en-US' and 'MDN Web Docs'. Below this, a yellow box contains the text: 'The MDN Web Docs site provides information about Open Web technologies including HTML, CSS, and APIs for both Web sites and progressive web apps.' The main content area is divided into two columns. The left column has links for 'JavaScript', 'HTML', and 'CSS', each followed by a brief description. The right column has links for 'Learn web development', 'Web Technologies', and 'Firefox Developer Tools', each followed by a brief description. At the bottom of the left column, there is a link 'More results from mozilla.org »'.

The head part of the HTML web page contains the additional information otherwise called meta information for the search engine which will not come as a part of the web page and are provided as `<meta>` tags.



There is an attribute named 'description' that summarizes the contents of your page for the benefit of users and search engines to get to know the content of the web page even without opening it.

Syntax of <meta> tag:

Syntax: `<meta attributes >`

The metadata element is defined within the head element.

```
<head>
  <meta attributes >
</head>
```

The below table discusses some of the attributes and their values related to the <meta> tag.

| Attribute | Value | Description |
|------------|--|--|
| name | application-name author description generator keywords | Specifies name for the metadata |
| http-equiv | content-type default-style refresh | Provides an HTTP reader for information/value of the content attribute |
| content | text | Gives the value associated with http-equiv or name attribute |
| charset | character_set | Specifies character coding for an HTML document |

For example,

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <title>Test page</title>
5.         <meta name="description" content="This is a test
   web page"/>
6.         <meta charset="utf-8">
7.     </head>
8.     <body>
9.         <p>Kindly check head section for meta tags</p>
10.    </body>
```



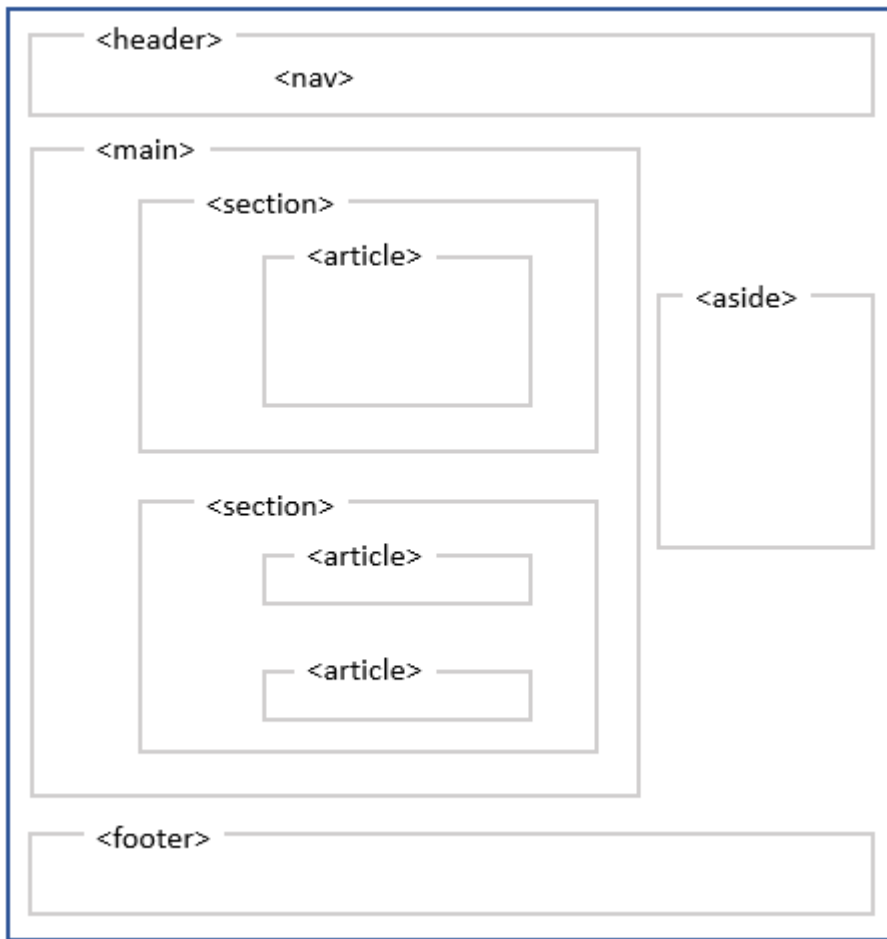
```
11. </html>
```

Web crawlers like Google, Bing, etc. are widely used for searching websites. They lookup each web page code and render the web page as per the HTML tags used and the styling associated.

Any regular user who is accessing any website will notice the below observations in most of the web pages of the website:

1. Right at the beginning of a web page, a header containing the website name is clearly displayed in the form of a logo or text. This helps the user to know which website they are currently referring to.
2. The links to navigate to other web pages of the website are displayed in the header. This makes a website user to figure out easily how to access other web pages of that website.
3. Details like copyright, about us, etc. are usually displayed at the bottom end of the screen, as part of the footer, as these details hold lesser importance, as compared to the actual data that they intend to read in the page.

The below figure illustrates some of the widely used sectioning elements in HTML5.



Below are some of the semantic tags in HTML:

<header>

The <header> element is used to include header content like web page logo, login link, website settings link, etc. Ideally, every web page has one header. However, multiple headers may also be included as per need.

```
1. <header>
2.     <h3>About Us</h3>
3. </header>
```

<footer>

The <footer> element is used to include footer content like copyright, about us, terms and conditions link, etc. One footer is included per page.



```
1. <footer>
2.     Copyright @ WayFar, 2020
3.     <a href="./AboutUs.html">About Us</a>
4. </footer>
```

<main>

The <main> element is used for demarking the main content of the web page. Only one main tag per web page is allowed.

```
1. <main>
2.     <section>
3.         ..
4.     </section>
5.     <section>
6.         <article>
7.             ..
8.         </article>
9.         <article>
10.            ..
11.        </article>
12.    </section>
13. </main>
```

<nav>

The <nav> element is used for navigational content like navigation menu for the website. There is no limit to the number of times <nav> tag can be used on a web page. As long as there are navigation links, links can be wrapped inside <nav>.

```
1. <nav>
2.     <a href="Home.html">Home</a>
3.     <a href="Login.html">Login</a>
4. </nav>
```

<section>

The <section> element is used to organize the web page into different sections.



```
1. <main>
2.   <section>
3.     <p>Section 1</p>
4.   </section>
5.   <section>
6.     <p>Section2</p>
7.   </section>
8. </main>
```

<article>:

The <article> element is used to include self-contained composition on a web page.

```
1. <article>
2.   <h1>MEAN stack</h1>
3.   <p>MEAN stack training includes discussion on MongoDB,
4.   Node,
5.   Express and Angular with the corresponding
6.   certifications</p>
7. </article>
```

<aside>:

The <aside> element is used to include content related to the main content of the web page.

```
1. <article>
2.   <h1>MEAN stack</h1>
3.   <p>MEAN stack training includes discussion on MongoDB,
4.   Node, Express and Angular with the corresponding
5.   certifications</p>
6.   <aside>
7.     <p>Visit our official website to attempt our
8.     certifications</p>
9.   </aside>
10. </article>
```

<address>:

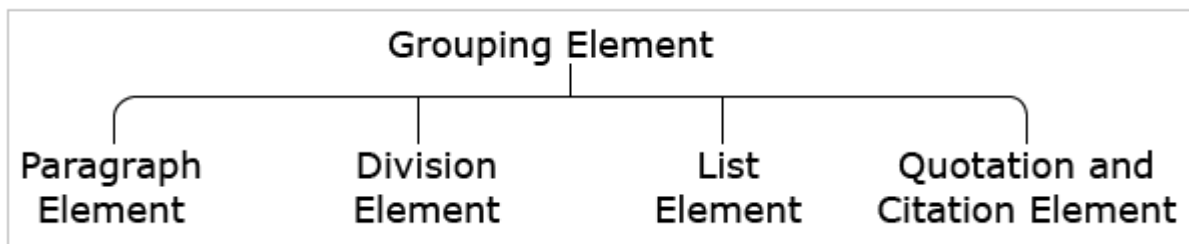


The <address> element helps to semantically organize address details in HTML content.

```
1. <address>
2.   John
3.   #231A
4.   Palace Lane
5.   Bangalore
6. </address>>
```

Group Elements:

Grouping elements in HTML5 can be categorized majorly as below:



The paragraph element is generally used for denoting a paragraph. Any textual content can be mentioned inside this element.

It is defined using <p>...</p> tag.

Let us understand this with an example.

Copy the below code into your Visual Studio Code workspace.

demo.html

```
<!DOCTYPE html>
```

```
<html>
```

```
  <body>
```

```
    <p>This is a Paragraph</p>
```

```
  </body>
```

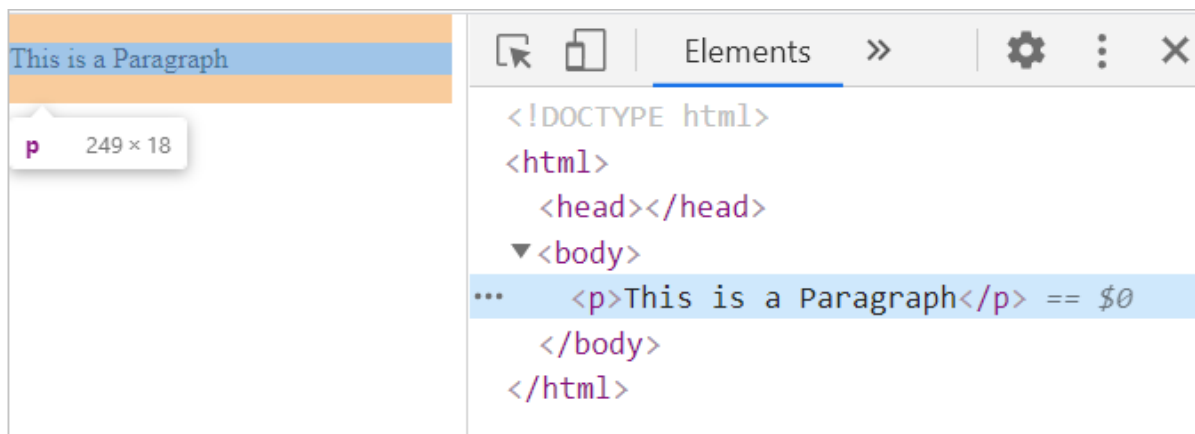


</html>

To execute this code:

- Right-click on the file demo.html
- Select the option "Open with Live Server"
- Right-click on the browser, and select the inspect option.

In the pop-up window, observe the 'Elements' tab as below:



We can observe that the paragraph element is a block-level element and hence adds a new line automatically when a paragraph ends. This ensures visual spacing between consecutive paragraphs of text.

Division and Span Elements:

The division element is used to group various other HTML tags. This element helps us in organizing the web page into different sections.

If any common rule or style needs to be added to a particular section, the same can be applied to the corresponding division. The rule or style gets applied to all the contents of the division thereby.

It is defined using <div>...</div> tag.

Let us understand this with an example

Copy the below code into your Visual Studio Code workspace.

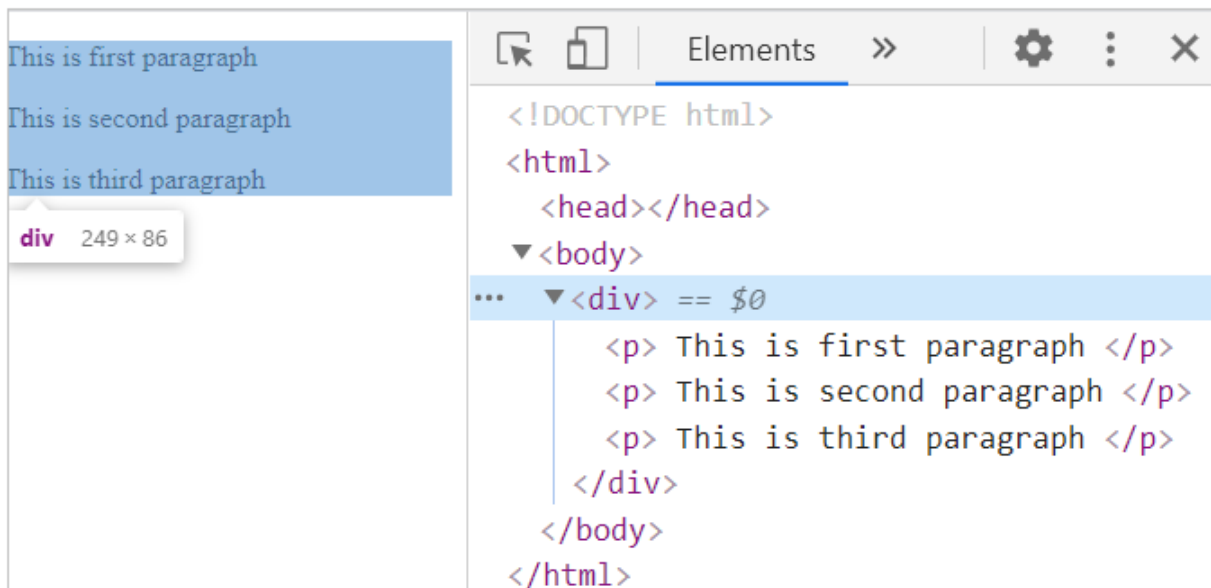
demo.html


```
1. <!DOCTYPE html>
2. <html>
3.     <body>
4.         <div>
5.             <p> This is first paragraph </p>
6.             <p> This is second paragraph </p>
7.             <p> This is third paragraph </p>
8.         </div>
9.     </body>
10. </html>
```

To execute this code:

- Right-click on the file demo.html
- Select the option "Open with Live Server"
- Right-click on the browser, and select the inspect option.

In the pop-up window, observe the 'Elements' tab as below:



We can observe that the division element is a block-level element and hence a new line is automatically added after the division ends.

Similar to the division element, the span element is also used to group various other HTML tags to apply some common styles.



It is defined by using ` ...` tag.

The span element is by default inline in nature, and hence no new line is added after the span ends. This tag is preferred only when we cannot use any other semantic tags.

Copy the below code into your Visual Studio Code workspace.

demo.html

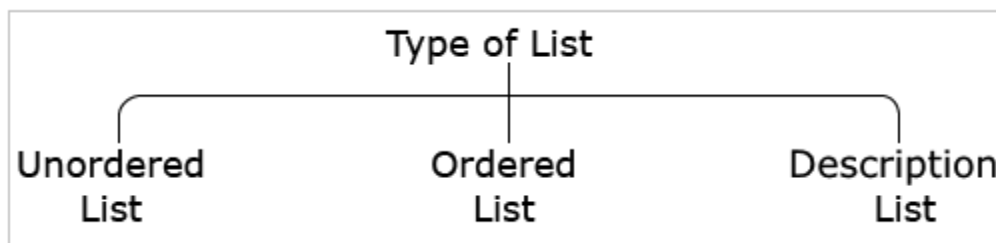
```
1. <!DOCTYPE html>
2. <html>
3.     <body>
4.         <div>
5.             <span>first section of paragraph</span>
6.             <span>second section of paragraph</span>
7.         </div>
8.     </body>
9. </html>
```

To execute this code:

- Right-click on the file demo.html
- Select the option "Open with Live Server"
- Right-click on the browser, and select the inspect option.

List Elements:

HTML lists come in three basic flavors and each one has a specific implementation.



HTML lists come in 3 basic flavors:

1. Unordered list
2. Ordered list
3. Description list



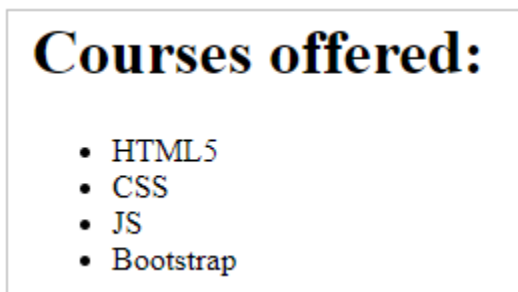
Each one has a specific purpose and meaning.

Let us start with the unordered list.

An unordered list is used to create a list of related items, in no specific order, like in the Terms and Conditions page where there is more focus on ensuring the readability of content by listing out points but not much concern about the specific order of points.

- An unordered list starts with the `` tag.
- Each item within the list technically referred to as 'list-item' enclosed within the `` tag.

For example, to generate an unordered list as seen below :



The following snippet can be used.

```
1. <h1>Courses offered:</h1>
2.   <ul>
3.     <li>HTML5</li>
4.     <li>CSS</li>
5.     <li>JS</li>
6.     <li>Bootstrap</li>
7.   </ul>
```

Let us learn how to customize the unordered list appearance.

The types of bullet points can be customized in an unordered list by using the `list-style-type` property provided by CSS.

For example, if we want our list to be displayed as below:



Courses offered:

- HTML5
- CSS
- JS
- Bootstrap

The corresponding code to achieve this requirement is:

```
1. <h1>Courses offered:</h1>
2.   <ul style="list-style-type: square;">
3.     <li>HTML5</li>
4.     <li>CSS</li>
5.     <li>JS</li>
6.     <li>Bootstrap</li>
7.   </ul>
```

The possible values for the list-style-type property are :

| Possible Values of list-style-type property | Type of Bullet |
|---|----------------|
| disc | • |
| circle | ○ |
| square | ■ |

By default, the value 'disc' will be assigned to the property.

Let us learn how to nest the unordered list while designing a web page.

Note: There is a type attribute to define the types of bullet points, which is not recommended to be used as per the latest HTML version.

In HTML, it is also possible to nest lists into different levels.

For example, if you want to create a nested list as shown below:

Courses Offered:

- Markup
 - Basics of HTML
 - First level course on HTML
 - Adaptive HTML
- Styling
 - CSS3
 - Latest version of CSS

Corresponding HTML code to achieve this requirement is:

```
1. <h1>Courses Offered:</h1>
2.     <ul>
3.         <li>Markup
4.             <ul>
5.                 <li> Basics of HTML
6.                     <ul>
7.                         <li> First level course on
HTML </li>
8.                     </ul>
9.                 </li>
10.                <li> Adaptive HTML </li>
11.            </ul>
12.        </li>
13.        <li>Styling
14.            <ul>
15.                <li> CSS3
16.                    <ul>
17.                        <li> Latest version of CSS
</li>
18.                    </ul>
19.                </li>
20.            </ul>
21.        </li>
22.    </ul>
```



If you observe the above-given code snippet, based on the enclosure of the inner `...` elements on various lines, the default bullet styling of unordered list element has been populated on to the web page.

An ordered list is used when the order of items is important and we want to create a list of related items, in a specific order.

- An ordered list starts with the `` tag.
- Each item within the list technically referred to as 'list-item' enclosed within the `` tag.

For example, to generate an unordered list as seen below :

Courses offered:

1. HTML5
2. CSS
3. JS
4. Bootstrap

The below-given code snippet can be used to achieve this requirement.

```
1. <h1>Courses offered:</h1>
2.   <ol>
3.     <li>HTML5</li>
4.     <li>CSS</li>
5.     <li>JS</li>
6.     <li>Bootstrap</li>
```

Courses offered:

- I. HTML5
- II. CSS
- III. JS
- IV. Bootstrap

The corresponding code to achieve this requirement is:



```
1. <h1>Courses offered:</h1>
2.   <ol style="list-style-type: upper-roman;">
3.     <li>HTML5</li>
4.     <li>CSS</li>
5.     <li>JS</li>
6.     <li>Bootstrap</li>
7.   </ol>
```

Some of the following values for the list-style-type property are:

| Some of the possible values of the list-style-type property | Type of bullet |
|---|----------------|
| 1,2,3,4 ... | decimal |
| I, II,III,IV,.. | upper-roman |
| i,ii,iii,iv,.. | lower-roman |
| A,B,C,D,.. | upper-latin |
| a,b,c,d,.. | lower-latin |

By default, the 'decimal' value will be set to the CSS property. We can also have 'none' value for the list-style-type property if we do not need any bullets to be present in the list.

Let us have a glance over some of the attributes which help us to customize the ordered list while designing a web page.

Some of the attributes which can be used with this element are:



| Name | Description |
|----------|---|
| start | Specifies the initial value of the list. |
| reversed | Specifies the pattern to be rendered in reversed order. |
| type | Specifies the different numbering values like : <ul style="list-style-type: none">• 1 for number (default).• a for lowercase alphabets.• A for uppercase alphabets.• i for lowercase roman numeral value.• I for uppercase roman numeral value. |

For example:

Consider the following code snippet:

```
1. <h1>Courses offered:</h1>
2.   <ol type="a" start="d" reversed>
3.     <li>HTML5</li>
4.     <li>CSS</li>
5.     <li>JS</li>
6.     <li>Bootstrap</li>
7.   </ol>
```

The above code will generate lists of course offered on the web page as shown below:

Courses offered:

- d. HTML5
- c. CSS
- b. JS
- a. Bootstrap



Description List

Now, let us move to the Description lists.

Description lists are used to contain name-value groups, where names can be a list of terms and values can be their related descriptions.

The description list otherwise called definition list arranges items in the same way as the meaning associated with each word is arranged in a dictionary as below:

The God of Small Things

The story is authored by Arunthathi Ry and is set in Kerala and revolves around the lives of two children Rahel and Esthepa and how they weave and imagine their childhood experiences.

Shadow Lines

Shadow Lines is an invigorating story by Amitav Ghosh about the borders that mark and limit our imaginations and memories.

The Lord of The Rings

An epic high fantasy book by the English author and scholar J.R.R.Tolkien

- Description lists are created with the <dl> tag.
- The term is placed within <dt>.. </dt> tag and description is placed between <dd>...</dd> tag.

Let us see the syntax now:

```
1. <dl>
2. <dt> Description Term 1 </dt>
3. <dd> Description Definition 1 </dd>
4. <dt> Description Term 2 </dt>
5. <dd> Description Definition 2 </dd>
6. </dl>
```

Quotation and Citation Elements:

While designing a website, we might include quotations or blocks of text from another source on our web page. For example:



Here is a quote from WHO's website:

WHO began when our Constitution came into force on 7 April 1948 – a date we now celebrate every year as World Health Day. We are now more than 7000 people from more than 150 countries working in 150 country offices, in 6 regional offices and at our headquarters in Geneva.

visually, such quotes if included, should be identifiable. Also, the browser needs to render this appropriately. This is why the quotation and citation element is introduced in HTML.

The quotation element helps to display the quotation texts with an alignment such that it looks unique and different from the remaining textual content.

By default, the quotation element is rendered visually with indentation by the browsers.

The quotation element also helps to include the citation, i.e., the URL from where the quote has been picked. This helps to retain the reference courtesy to the original site.

Quotation element is defined using `<blockquote>...</blockquote>` tag.

The source of the quote is mentioned in the cite attribute of the blockquote element.

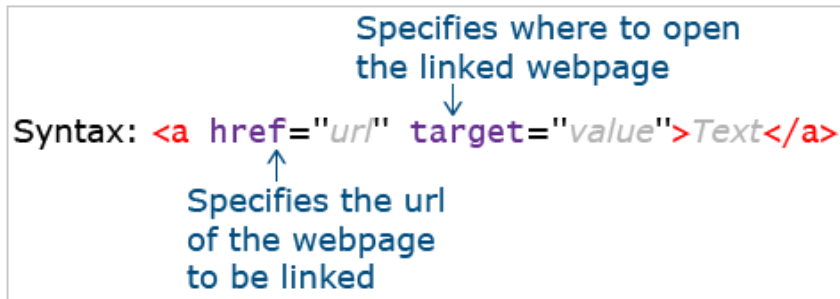
```
1. <!DOCTYPE html>
2. <html>
3.     <body>
4.         Below line has been referred from OWASP website:
5.         <blockquote cite="https://owasp.org/">
6.             The Open Web Application Security Project®
7.             (OWASP) is a nonprofit foundation that works to improve the
8.             security of software.
9.         </blockquote>
10.    </body>
11. </html>
```

The content enclosed inside `<blockquote>...</blockquote>` will appear in Chrome browser with default indentation as observed below.

Note: The output may vary in different browsers.

Link Element :

Link elements are defined using `<a> .. ` tag as below:



Text / Image can be used as link.

Text / Image that provides such a link is called "**hyperlink**".

A hyperlink is a prime way in which users can navigate from one web page to another. A hyperlink can point to another web page, or website, or files, or even specific locations on the same web page.

Hyperlinks can be of any of the below types:

Text hyperlink:

- A clickable text is used to take the user to another web page. Largely, we use text-based hyperlinks.
- This text usually appears with an underline and in a different color.
- This color mapping is automatically done by the browser for all text hyperlinks.

Image hyperlink:

- A clickable image is used to take the user to another web page.

Bookmark hyperlink:

- A clickable text/image is used to take the user to another part of the same web page.

Email hyperlink:

- It allows users to send an email by clicking on that link.

Contact number hyperlink:

- It allows the user to call a number by clicking on that link.

Let us discuss various hyperlinks that can be created in an HTML page.

Text hyperlink:

The text is mentioned within the start tag `<a>` and end tag `` and is displayed on the screen in a clickable manner



1. ` Click here to connect to us
`
2. ` Click here to go to Google website `

Image hyperlink:

We can also have an image-based hyperlink. For this, we need to wrap the image inside an anchor tag.

1. ``
2. ``
3. ``

On click of the image, the user gets redirected and the google.com website gets loaded in the browser tab.

Bookmark hyperlink:

When a web page is lengthy, we commonly come across icons or links that say "Go to Top" or "Go to Bottom". Click on these links does take the user to the top of the page or bottom, as applicable. Sometimes we also observe, on click of a text in the menu bar, the page auto scrolls to that particular section on that page. This is achieved by using the Bookmarking concept and the same is implemented by using hyperlinks.

1. `<h2 id="top">Topic</h2>`
2. `<p>Detail.....</p>`
3. `<p>Detail.....</p>`
4. `<p>Detail.....</p>`
5. `Go to Top`

Email hyperlink:

To send an email on click of a hyperlink, use the below syntax:

1. `Send Mail`

On click of the link, the installed mail client on the computer gets activated for sending the email. An installed email client should necessarily be installed on the computer for this to work.

Contact number hyperlink:

To make a call to a number on click of a hyperlink, use the below syntax:

1. `Call Us`



Link element has an attribute named 'target' which specifies in which window, the hyperlinked content should appear.

The target attribute can hold the following values:

| The possible value of "target" | Description |
|--------------------------------|---|
| _blank | Opens a web page in a new window or tab |
| _self | Opens a web page in the same window (default) |
| _parent | Opens a web page in the parent frame |
| _top | Opens a web page in the full body of the window |
| frame-name | Opens a web page in a named frame |

For example:

```
1. <a href="https://owasp.org/" target="_blank">Link to OWASP web site</a>
```

In the above example, the <https://owasp.org/> website opens in a new window.

Character Entities

Some characters are reserved in HTML.

For example: If you use the less than (<) or greater than (>) sign in your content, the browser may mix them with HTML tags.

Also, some characters are unavailable on the keyboard.

For example: ©

Character entities are used to include such character content on a web page.

Syntax: **&entity_name;**
OR
&#entity_number;

The table below lists widely used character entities supported in HTML5.



| Character | Description | Entity Name | Entity Number |
|-----------|----------------------|-------------|---------------|
| | Non-breaking space | | |
| < | Less than | < | < |
| > | Greater than | > | > |
| & | Ampersand | & | & |
| © | Copyright | © | © |
| € | Euro | € | € |
| £ | Pound | £ | £ |
| ® | Registered trademark | ® | ® |

HTML5 Global Attributes

Attributes that can be used with all HTML elements are called "Global attributes".

The table below lists a few global attributes supported in HTML5.

| Attribute | Description |
|-----------------|--|
| contenteditable | Allows the user to edit content. Possible values are true/false. |
| dir | Specifies text direction. Possible values are ltr/ rtl. |
| title | Displays the string message as a tooltip. |
| spellcheck | Specifies whether the spelling of an element's value should be checked or not. Possible values are true/false. |
| id | Gives a unique id to an element. |

For example:

```
1. <div>
2.   <p contenteditable="true">This is editable</p>
3.   <p dir="rtl">The direction of the text is from right to left</p>
4.   <p title="mydemo">Hover your mouse here to see the title</p>
5.   <p id="id1">ID should be unogue for each element</p>
6. </div>
7.
```

In the above example you can observe that:

- Line number 2 allows the user to edit the text which has been enclosed with the <p> element.
- Line number 3 modifies the text direction of the <p> element from right to left orientation.
- While hovering the mouse on line number 4, the title message will be displayed.
- The id attribute has been used to set a unique id for <p> element in line number 5.

Tables:



The table element is defined in HTML using <table>...</table> tag

It contains table header and table body.

The table header is for adding header information like column headers and the table body is for table contents.

```
Syntax: <table>
        <!-- Table data -->
        </table>
```

The following elements are used within the table element:

| Element | Description |
|----------|--|
| caption | Defines table heading |
| tr | Defines row of the table |
| th | Defines heading of the column |
| td | Defines data of column |
| thead | Defines header part of the table |
| tbody | Defines the content part of the table |
| colgroup | Helps to logically group two or more consecutive columns |

Table Structure:

```
1. <table>
2.     <caption>Table heading</caption>
3.     <thead>
4.         <tr>
5.             <th>Column 1 heading</th>
6.             <th>Column 2 Heading</th>
7.         </tr>
8.     </thead>
9.     <tbody>
10.        <tr>
11.            <td>Column 1 data</td>
12.            <td>Column 2 data</td>
13.        </tr>
14.    </tbody>
15. </table>
```

The above code snippet displays the below table on the web page:

Let us have a glance over how to create a simple table using HTML now.



Consider the below code snippet:

```
1. <html>
2.   <table>
3.     <thead>
4.       <th>Back-end JS </th>
5.       <th>Front-end JS </th>
6.     </thead>
7.     <tbody>
8.       <tr>
9.         <td>Node</td>
10.        <td>React</td>
11.      </tr>
12.      <tr>
13.        <td>Express</td>
14.        <td>Angular</td>
15.      </tr>
16.    </tbody>
17.  </table>
18.</html>
```

The output for the above code snippet will be as shown below:

| Back-end JS | Front-end JS |
|-------------|--------------|
| Node | React |
| Express | Angular |

In the above-mentioned code we can observe that:

- The <table> element encloses all the other table elements in it.
- The optional <thead> and <tbody> elements helps us to logically divide the table content.
- The <th> element is used to define the column heading.
- The <tr> element creates a new row.
- The <td> element creates a new table column.

Table Elements: colspan/rowspan Attributes

The elements <td> and <th> supports the attributes namely colspan and rowspan which helps to merge the table cells accordingly.

The colspan attribute accepts a numeric value and merges specified numeric value of columns together whereas, the rowspan attribute accepts a numeric value and merges specified numeric value of rows together.

Consider the below table with 4 rows and 4 columns.

| | C1 | C2 | C3 | C4 |
|----|----|----|----|----|
| R1 | | | | |
| R2 | | | | |
| R3 | | | | |
| R4 | | | | |

For example, if we provide colspan attribute with a value 2, then 2 columns of the table will be merged as shown below:

| | C1 | C2 | C3 | C4 |
|----|----|----|----|----|
| R1 | A | B | C | D |
| R2 | | | | |
| R3 | | | | |
| R4 | | | | |

```

<tr>
  <td rowspan="2">A</td>
  <td>B</td>
  <td>C</td>
  <td>D</td>
</tr>

```

And, if we provide rowspan attribute with a value 2, then 2 rows of the table will be merged as shown below:

| | C1 | C2 | C3 | C4 |
|----|----|----|----|----|
| R1 | A | | B | C |
| R2 | | | | |
| R3 | | | | |
| R4 | | | | |

```

<tr>
  <td colspan="2">A</td>
  <td>B</td>
  <td>C</td>
</tr>

```

Form Elements:

The form can be created using <form>...</form> tag of HTML.

The <form> tag has the below attributes:

- method: Defaults to HTTP "get" method of submission to the server. To use HTTP "post", use method="post"
- action: The URL to which the form data has to be submitted
- target: Specifies if the submitted result will open in the current window, a new tab, or on a new frame

Used for accessing form data by the scripting language

Specifies the server-side program that will be executed when the form is submitted

Syntax: `<form name="Name of form" action="Link to server-side program" method="HTTP Request method">`
<!-- All form elements will come here -->
`</form>`

Specifies HTTP request method that will be used to submit form data to the server-side program

The form input element is used to collect details from the user.

Specifies type of element

Syntax: `<input type="input type" value="element value">`

Specifies the element's value that will be send to the server program

The table below lists the various types of input elements.

| The possible value of "type" | Description |
|------------------------------|---|
| text | Creates textbox |
| password | Creates textbox that accepts the only password |
| checkbox | Creates checkbox |
| radio | Creates a radio button |
| button | Creates button |
| submit | Creates a button that submits values of all form elements to the server |
| reset | Creates a button that resets values of all form elements to their default value |
| image | Creates a graphical version of a button |
| file | Creates control to upload the file to the server |
| hidden | Creates a hidden text field |
| email | Creates textbox that accepts only valid email id |
| number | Creates spinbox that accepts only whole numbers |



| The possible value of "type" | Description |
|------------------------------|---|
| range | Creates a range slider |
| search | Creates a search bar |
| URL | Creates textbox that accepts only valid URL |
| color | Creates color picker |
| date | Creates date picker to select date |
| month | Creates date picker to select a month |
| week | Creates date picker to select week |

Let us learn some of the basic input types of HTML form and understand their implementation in short.

Input type - text:

A single-line text field. The value attribute defines the value of the input field.

```
1. Name: <input type="text" value="">
```

Input type - password:

An input field can be used to enter a password.

```
1. Password: <input type="password">
```

Input type - email:

- An input field that accepts email addresses.
- It has in-built validation for an email.

```
1. Email-Id: <input type="email">
```



Input type - number:

- Defines an input text box, where the user can enter only numerical input.
- Gives an error on form submission if the value entered goes beyond the min and max limits and includes built-in validation to reject non-numerical values.
- Attributes min and max can be used to define a boundary and step attribute value which can be used for defining the difference between consecutive numbers.

```
1. Age: <input type="number">
```

Input type - checkbox:

- Defines a checkbox.
- The checked attribute checks that particular checkbox value.
- Also, multiple checkboxes can be checked at a time.

```
1. Hobbies: <input type="checkbox" checked> Reading
2.         <input type="checkbox" checked> Singing
3.         <input type="checkbox" > Dancing
```

Input type - radio:

- Defines a radio button.
- The name attribute specifies the associated name of that radio button.
- Radio buttons in a group should have the same name.

```
1. Gender: <input type="radio" name="gender" checked value="Male"> Male
2. <input type="radio" name="gender" value="Female"> Female
```

Input type - file:

Creates a control to upload a file to the server.



```
1. Select a file: <input type="file">
```

<button> element:

- Defines a clickable button that can be used to submit the form.
- The button can be of 3 types:
 - submit (default with <button> tag)
 - reset (to reset the form)
 - button (just a clickable button)

<textarea> element:

- Defines a multi-line text field.
- It is not possible to set a default text using the value attribute. Hence, default text can be placed into <textarea>...</textarea> tag.

```
1. Write your comments: <textarea rows="4" cols="10" >Default value</textarea>
```

Input type: Hidden:

You may want to submit supplementary data (such as users' language of user input) to the server, without any user interaction. This can be done using a hidden element.

```
1. <input type="hidden" name="Language" value="English"/>
```

Note: Output may vary for some input fields according to the browser

```
1. <input type="hidden" name="Language" value="English"/>
```

Note: Output may vary for some input fields according to the browser

Label:

The <label> element is used to associate a text label with a form <input> field. The label is used to tell users the value that should be entered in the associated input field.

Additionally, the "for" attribute of the label can point to the "id" of input control. This ensures the cursor focuses on the respective input control on the click of the label.

It also enhances the usability, by allowing the user to toggle the control by clicking on text written within <label>...</label> tag.

Refers to *id* attribute
of an *input* element



Example: `<label for="Username">Enter Username</label>
<input type="text" id="Username">`

Color and Date Pickers:

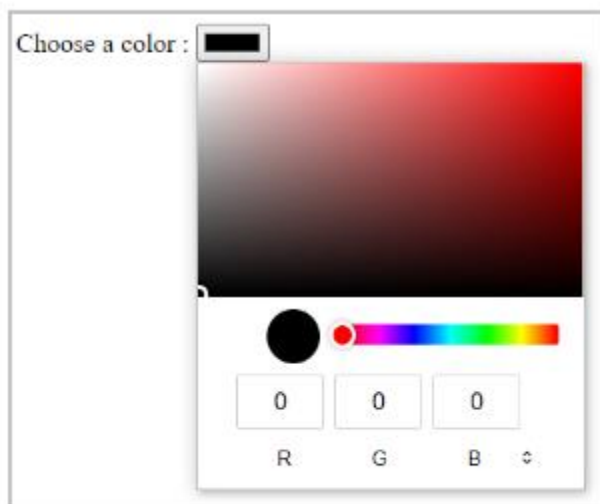
We have various picker-elements in HTML forms such as color-picker and date-picker elements.

Let us see them in detail.

Input type - color:

Defines a color picker.

```
1. Choose a color: <input type="color">
```



Input type - date:

Creates a date-picker which is used to collect dates.

Date of Birth :

June 2020

| Su | Mo | Tu | We | Th | Fr | Sa |
|----|----|----|----|----|----|----|
| 31 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |

Today

Input type – datetime-local:

Defines a date-time picker, where the user can pick a date as well as time

```
1. Choose a date and time: <input type="datetime-local"/>
```

Choose a date and time:

June 2020

| Su | Mo | Tu | We | Th | Fr | Sa |
|----|----|----|----|----|----|----|
| 31 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |

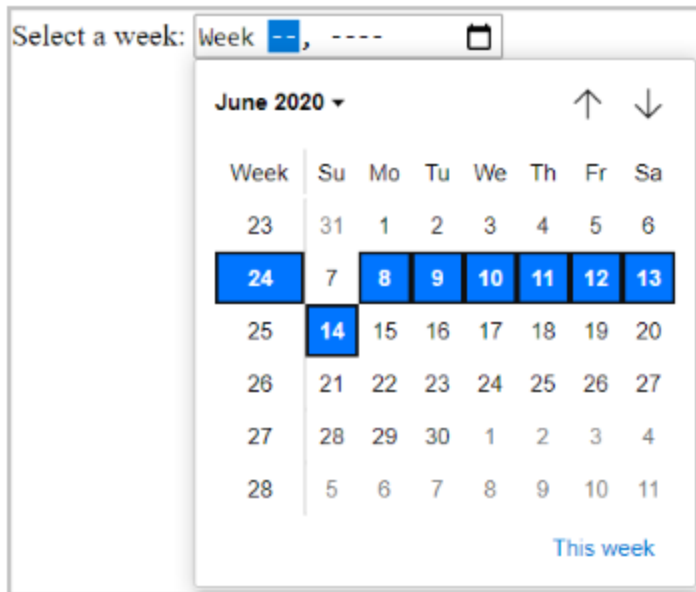
Today

| | | |
|----|----|----|
| 04 | 39 | PM |
| 05 | 40 | AM |
| 06 | 41 | |
| 07 | 42 | |
| 08 | 43 | |
| 09 | 44 | |
| 10 | 45 | |

Input type – week:

Defines a date picker, where the user can pick a week.

```
1. Select a week: <input type="week"/>
```



Select a week: Week --, ----

June 2020

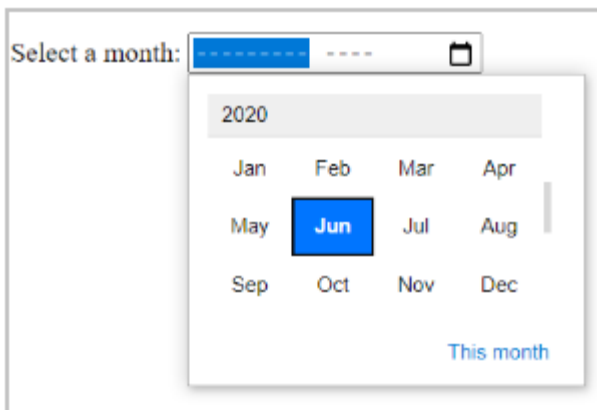
| Week | Su | Mo | Tu | We | Th | Fr | Sa |
|------|----|----|----|----|----|----|----|
| 23 | 31 | 1 | 2 | 3 | 4 | 5 | 6 |
| 24 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 25 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 26 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 27 | 28 | 29 | 30 | 1 | 2 | 3 | 4 |
| 28 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

This week

Input type – month:

Defines a date picker, where the user can pick a month.

1. Select a month: `<input type="month"/>`



Select a month: -----

2020

| | | | |
|-----|-----|-----|-----|
| Jan | Feb | Mar | Apr |
| May | Jun | Jul | Aug |
| Sep | Oct | Nov | Dec |

This month

Select and Datalist Elements

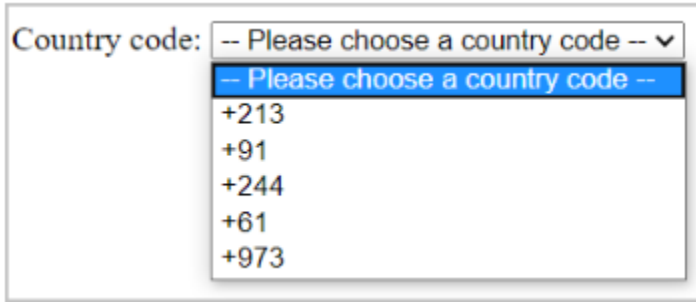
We have `<select>` and `<datalist>` elements in HTML which helps to collect input data from user as a drop-downs.

Let us see them in detail:

`<select>` element :

- Defines a drop-down list.
- The "multiple" attribute can be used for having a multi-select dropdown menu.



```
1. Country code: <select >
2.   <option value="">-- Please choose a country code --</option>
3.   <option value="+213">+213</option>
4.   <option value="+91">+91</option>
5.   <option value="+244">+244</option>
6.   <option value="+61">+61</option>
7.   <option value="+973">+973</option>
8. </select>
```



<datalist> element:

- Defines a set of pre-defined options available to choose for an <input> element.
- In the below example list attribute holds lists of possible options, the value assigned to the list attribute of the input element and id attribute of datalist attribute should be the same and the value sent to the server should be assigned to the option element value attribute.

```
1. Country: <input list="countries">
2. <datalist id="countries">
3.   <option value="India">
4.     <option value="France">
5.       <option value="Singapore">
6.         <option value="Thailand">
7.           <option value="United Arab Emirates">
8.             <option value="United States of America">
9. </datalist>
```





Input Elements-Attributes: The following are some of the attributes which can be used with HTML input elements.

- Placeholder
- Pattern
- Min
- Max
- Step
- Required
- Multiple
- Form-override

Let us see each of them in detail.

Placeholder:

The placeholder attribute specifies a value that appears in the textbox as below:

```
1. First name: <input type="text" placeholder="Enter your first name"/>
```

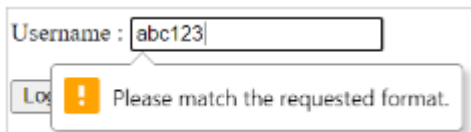
Pattern:

The pattern attribute creates a custom pattern validator.

The value entered by the user is checked for validity against a specified pattern.

If the user input value does not match with a specified pattern, an error message appears.

```
1. Username: <input type="text" pattern="[A-Za-z]"/>
```



Min, Max, and Step:

The following are some of the attributes which are used only with range and number input types

- min: Specifies a minimum acceptable value.
- max: Specifies maximum acceptable value.
- step: Specifies a difference of consecutive values when the user uses the range/number input element.



```
1. Job experience: <input type="number" min="2" max="10" step="1"/>
```

Job experience:

Multiple:

The multiple attribute value allows the user to enter/select/upload more than one value.

```
1. <input type="file" multiple/>
```

Form-Override:

The override attributes can be used to override the form-level built-in attribute functionalities using the submit/image input elements.

| Form-override attribute | Description |
|-------------------------|---|
| formaction | Overrides the form action attribute |
| formnovalidate | Overrides the form novalidate attribute |
| formmethod | Overrides the form method attribute |
| formtarget | Overrides the form target attribute |

In the below example, you can observe that the default form submission method 'GET' has been overridden to the 'POST' method due to the usage of 'formmethod' attribute in the submit input tag.

```
1. <form method="GET" action="">
2.   <input type="submit" formmethod="POST">
3. </form>
```

Editing Elements:

The following are elements used for editing.

- **del:** It defines deleted text by striking on it
- **ins:** It defines inserted text by underlining it

Example:

```
1. <p>HTML is building block of Internet</p>
```

```
2. <p>HTML is building block of <del>Internet</del> <ins>WWW</ins></p>
```

Media:

Image Element:

Embedding images to a web page can be done with the `...` tag. The `` tag has attributes 'src' and 'alt' where src defines the location of the image and alt defines the alternative text if it is failed to load the image inside the web page.

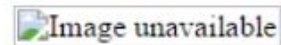
```
1. 
```

Example: ``

Output:



When image is available



When image is not available, text written in alt attribute is displayed

We can also provide a caption for the embedded images.

HTML provides a semantic element `<figure>` along with `<figcaption>` element which help us to provide caption for our images.

For example, consider the following code.

```
1. <figure>
2.   
6.
7.   <figcaption>A colorful tropical sea view</figcaption>
```

```
8. </figure>
```

The above code snippet generates an output as below:




Audio Element:

Embedding audio to a web page can be done with `<audio>...</audio>` tag. The `<audio>` tag has an attribute 'src' which defines the location of the audio file. It also has an attribute 'controls' which specifies whether to display the player controls.

```
1. <audio src="audio.mp3" controls="controls"></audio>
```

Content between `<audio>` and `</audio>` tag will be shown by browsers who do not support audio element.

Example: `<audio src="myAudio.mp3" controls="controls">`
Your browser does not support audio tag
`</audio>`

Output: 

(Appearance of control bar may differ from browser-to-browser)

Some of the attributes which are supported by the audio element are as follows:

| Attribute | Value | Description |
|-----------|--|--|
| loop | Boolean- any value sets it to true | Loops audio indefinitely |
| autoplay | Boolean- any value sets it to true | Plays audio indefinitely |
| preload | none-preloading metadata- audio metadata is downloaded auto- entire audio file is downloaded | Specifies whether audio should be preloaded or not |
| muted | Boolean- any value sets it to true | Mutes audio |

Video Element:


Videos can be embedded into web pages using `<video>...</video>` tag. The `<video>` tag has an attribute 'src' which defines the location of the video file. It also has an attribute 'controls' which specifies whether to display the player controls.

```
1. <video src ="myVideo.mp4" controls="controls"></video>
```

Content between `<video>` and `</video>` tag will be shown by browsers who do not support the video element.

Example: `<video src="myVideo.mp4" controls">`
 Your browser does not support video tag
`</video>`

Output:



(Appearance of control bar may differ from browser-to-browser)

Some of the attributes which are supported by video element are as follows:

| Attribute | Value | Description |
|-----------|--|---|
| loop | Boolean- any value sets it to true | Loops audio indefinitely |
| autoplay | Boolean- any value sets it to true | Plays audio indefinitely |
| preload | none-preloading metadata- video metadata is downloaded auto- entire audio file is downloaded | Specifies whether the video should be preloaded or not |
| height | pixel | Specifies the height of the video player |
| width | pixel | Specifies the width of the video player |
| poster | URL of an image file | Displays image until the first frame of the video is downloaded |
| muted | Boolean- any value sets it to true | Mutes audio |

Source Element:

All browsers do not support all audio/video formats. Therefore, the audio/video element allows you to list multiple sources with the `<source>` element. The `<source>` element has an attribute 'type' which specifies the type of the file.



The browser iterates through all sources one by one until it finds one which it can play. It should be noted that while listing the audio/video formats, it should be in the order from most desirable to least desirable to avoid the number of unnecessary iterations.

Also, it is suggested to use the <source> element within the audio/video element instead of the src attribute.

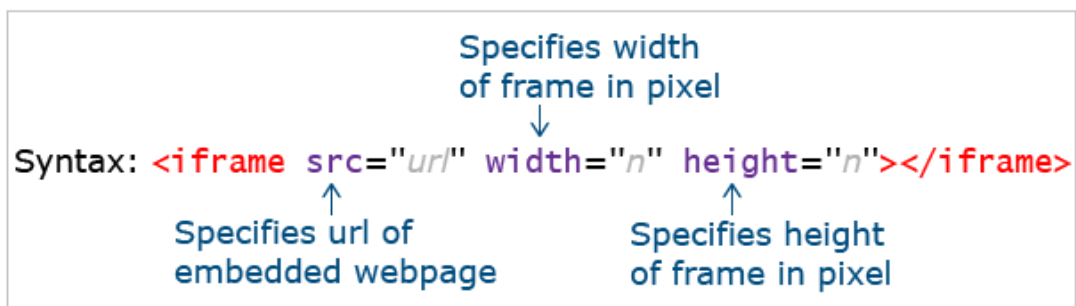
```
1. <audio>
2.   <source src="myaudio.ogg" type="audio/ogg">
3.   <source src="myaudio.mp3" type="audio/mp3">
4. </audio>
```

IFRAME:

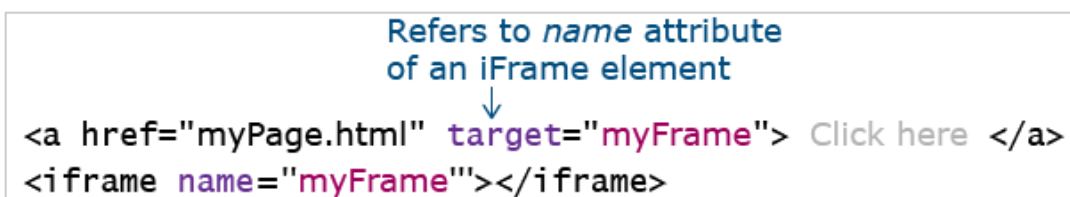
We might have requirements to include documents, videos, interactive videos, or even other web content into a web page directly from external sources.

The <iframe> element is used to meet the above requirement. Using the iframe element, contents from external sources can be integrated anywhere on our web page

It is defined using <iframe>...</iframe> tag.



The below example demonstrates including a web page myPage.html on clicking on the hyperlink.



What is Html Security?

As web developers, we need to take care of writing preventive measures for all possible vulnerabilities in web pages. As we use HTML for designing web pages, we should be aware of all possible vulnerabilities in this language. Let us now see what are the security risks possible in HTML.

HTML Security

As HTML applications are web-based applications, developers should take proper measures to safeguard the stored data and communications. An attacker can inject some malicious code via Bluetooth/wifi/text messages for mobile-based apps or via advertisements/emails for web-based apps. This malicious code can capture sensitive information and can expose it to the attacker or it can run some undesired operations internally like sending false messages or purchasing a product with zero amount etc.



The following is the list of a few vulnerabilities that are possible in HTML:

1. HTML Injection
2. Clickjacking
3. HTML5 attributes and events vulnerabilities
4. Web Storage Vulnerability
5. Reverse Tabnabbing

Let us now understand each vulnerability and its mitigation with examples.

Sometimes users will observe some unexpected data getting rendered on a web page. This might be due to a lack of proper validation for input and output on that page. So, a web developer should always check if input and output are properly validated before getting rendered on a page as this can lead to HTML injection attacks.

Let us now understand what is this attack and its consequences.

What is HTML Injection?

HTML Injection is one of the possible attacks in HTML5 apps. HTML Injection is an attack where an attacker can inject malicious HTML code into a vulnerable web application. An attacker can send malicious code through HTML input fields or website links. Malicious code can be simple HTML tags that display some information on the page or a form replacing the original page etc., This kind of vulnerability happens when user input is not properly sanitized or the output is not properly encoded.

How HTML Injection is Performed?

An attacker will first find the vulnerable parts of a web application by inspecting the source code in the browser. Vulnerable parts may be HTML form fields or website links through which an attacker will inject some malicious HTML code. There are many attributes or methods which can render data on an HTML page. If malicious code is injected using innerHTML property and if the data is not properly sanitized, then it will lead to this attack. document.write() method is another way to inject the malicious code. It is used mostly for form input fields like comments box, registration forms, questionnaire forms, etc., These kinds of elements are most vulnerable to HTML Injection attacks.

The main intention of this injection attack is to either change the website's appearance or to steal the user's identity.

Example:

Following is the malicious HTML code an attacker injects through a website URL:

```
1. http://example.com/home.html?username=<img%20src='image1'%20onerror=alert('error')>
```

Here an attacker is sending an image assigned to the username parameter embedded in a URL.

Following is the vulnerable code which doesn't validate or sanitize the data:

```
1. var position = location.href.indexOf("username=");  
2. var user = location.href.substring(position+5);
```



```
3. document.getElementById("div1").innerHTML = "Hello, " + user;
```

This code is extracting the username position and fetching the value assigned to it. It displays the image injected by an attacker in a div tag and runs some malicious script through it.

Following is an example of using the document.write() method:

```
1. var position = location.href.indexOf("username=");  
2. var user = location.href.substring(position+5);  
  
3. document.write("<h1> Hello, " + user + "</h1>");
```

Let us now understand the different types of HTML Injections possible.

Types of HTML Injection

There are two types of HTML Injections:

1. Stored HTML Injection

In this injection, the malicious code injected by an attacker will get stored in the backend and will get executed whenever a user makes a call to that functionality.

2. Reflected HTML Injection

In this injection, the malicious code will not get code stored in the webserver rather will be executed every time the user responds to the malicious code.

How to prevent HTML injection?

HTML Injection will happen if data is not properly validated. So we need to do proper data validation to prevent this attack. We need to check if data contains any HTML tags and then needs to be removed. This is done differently in different languages or frameworks.

Below are a few mitigation techniques to prevent this attack:

1. Use safe Javascript methods like innerText in place of innerHTML
2. Code Sanitization: Removing illegal characters from input and output refers to HTML code sanitization.
3. Output Encoding: Converting untrusted data into a safe form where data will be rendered to the user instead of getting executed. It converts special characters in input and output to entities form so that they cannot be executed. For example, < will be converted to < etc.,



What is Clickjacking?

It is an attack where an attacker uses low iframes with low opaqueness or transparent layers to trick users into clicking on something somewhat diverse from what they actually see on the page. Thus an attacker is hijacking clicks which will execute some malicious code and hence the name 'Clickjacking'. It is also known as UI redressing or iframe overlay.

For example, on a social networking website, a clickjacking attack leads to an unauthorized user spamming the entire network of your friends by sending some false messages.

How clickjacking is performed?

1. Attackers load a target web page inside a low opacity iframe and on top of that iframe, the attacker's web page will be rendered with a clickable button or link.
2. When a user clicks on that button or link of the rendered web page, it triggers actions on the invisible page. The invisible page might be a malicious web page or a legitimate web page, which the user did not intend to visit.
3. The end user will have no idea that a click has been made on the invisible page and without the user's knowledge actions are performed on the attacker's web page

So, a web developer should also take care of clicks on the web page so that they should not get redirected to some malicious pages. Let us now see how to defend against this attack.

here are two ways to prevent clickjacking:

1. Client-side methods: The most common method is to prevent the webpages from being displayed within a frame which is known as frame-buster or frame-killer. Though this method is effective in a few cases it is not considered a best practice as it can be easily bypassed.

Below is the implementation of frame-buster:

```
1. <script>
2.     // frame busting
3.     if (self == top) {
4.         document.documentElement.style.visibility = 'visible';
5.     } else {
6.         top.location = self.location
7.     }
8.     function register() {
9.         alert('Registered')
10.    }
11. </script>
```

This code ensures that the current frame is the top-level window.

HTML5 Attributes & Events Vulnerabilities

HTML5 has few tags, attributes, and events that are prone to different attacks as they can execute Javascript code. These will be vulnerable to XSS and CSRF attacks.

Below are a few examples of such attacks:



1. Malicious script injection via formaction attribute

```
1. <form id="form1" />

2. <button form="form1" formaction="javascript:alert(1)">Submit</button>
```

In the above code snippet, the malicious script can be injected in formaction attribute. To prevent this, users should not be allowed to submit forms with form and formaction attributes or transform them into non-working attributes.

2. Malicious script injection via an onfocus event

```
1. <input type="text" autofocus onfocus="alert('hacked')"/>
```

This will automatically get focus and then executes the script injected. To prevent this, markup elements should not contain autofocus attributes.

3. Malicious script injection via an onerror event in the video-tag

```
1. <video src="/apis/authContent/content-
store/Infosys/Infosys_Ltd/Public/lex_auth_012782317766025216289/web-
hosted/assets/temp.mp3" onerror="alert('hacked')"></video>
```

This code will run the script injected if the given source file is not available. So, we should not use event handlers in audio and video tags as these are prone to attacks.

Mitigation

To defend against these attacks, we should use HTML Sanitization as a mitigation technique. Now let us understand what is HTML Sanitization.

HTML Sanitization provides protection from a few vulnerabilities like XSS(Cross-site scripting) by replacing HTML tags with safe tags or HTML entities.

The tags such as , <i>, <u>, , and , which are used for changing fonts are often allowed. The sanitization process removes advanced tags like <script> <embed>, <object> and <link>. This process also removes potentially dangerous attributes like 'onclick' attribute in order to prevent malicious code injection into the application.

Let us understand how to sanitize HTML data. The below table lists the entity names for some of the HTML characters.

| Entity names for some HTML characters | |
|---------------------------------------|---------------|
| Input Character | Encoded value |
| < | < or < |



| | |
|---|-----------------|
| > | > or > |
| " | " or " |
| ' | ' or ' |
| & | & or & |

When a web browser finds these entities, they will not be executed. But instead, they will be converted back to HTML tags and printed.

Consider the scenario that an attacker injects the below HTML code into a web page.

1. `Avengers`

On using HTML sanitization, the response will be as below.

1. ` Avengers `

This code will not be executed instead of stored as plain text in the response.

There are many sanitizer libraries available to do this job. Some of the commonly used libraries are DOMPurify, XSS, and XSS-filters.

HTML5 - Cross-browser support

Ideally, all browsers are supposed to implement the W3C specification of HTML5 as it is.

However, in reality, all browser vendors more or less customize the specification.

This leads to different outputs of the same HTML code when viewed on different browsers.

Activity: Copy below-given into your Visual Studio Code IDE workspace and save the file as cross-browser.html.

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <title>Cross Browser</title>
5.     </head>
6.     <body>
7.         <form>
8.             <label for="">Username</label>
9.             <input type="text" name="username"> <br/>
10.            <label for="">Password</label>
11.            <input type="password" name="password"><br/>
12.            <input type="submit">
13.        </form>
14.    </body>
15. </html>
```



ADITYA ENGINEERING COLLEGE(A)

Right-click on the file, copy the path, and paste it into different browsers such as Mozilla Firefox and Google Chrome and observe the below output.

| Google Chrome v.85 | Mozilla Firefox v.81 | Internet Explorer v.11 |
|---------------------------------------|---|---|
| Username <input type="text"/> | Username <input type="text"/> | Username <input type="text"/> |
| Password <input type="password"/> | Password <input type="password"/> | Password <input type="password"/> |
| <input type="submit" value="Submit"/> | <input type="submit" value="Submit Query"/> | <input type="submit" value="Submit Query"/> |

Also, all browsers may not support all features of HTML5.

Hence, while developing your website, you need to consider the capabilities of your user's browser - implement HTML5 features according to the capabilities of your user's browser.

Activity: Copy below-given into your Visual Studio Code IDE workspace and save the file as cross-browser-features.html.

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title>Cross Browser</title>
5.   </head>
6.   <body>
7.     <form oninput="x.value= parseInt(a.value)+parseInt(b.value) ">
8.       0 <input type="range" id="a" value="10" min="1"
max="200">200 +
9.       <input type="number" id="b" value="10">=
10.      <output name="x" for="a b"></output>
11.    </form>
12.  </body>
13. </html>
```

Right-click on the file, copy the path, and paste it into different browsers such as Mozilla Firefox and Google Chrome and observe the below output.

