UNIT-1

HTML 5: Introduction to Web, Overview of Web Technologies, HTML - Introduction, HTML - Need, Case-insensitivity, Platform-independency, DOCTYPE Declaration, Types of Elements, HTML Elements - Attributes, Metadata Element, Sectioning Elements, Paragraph Element, Division and Span Elements, List Element, Link Element, Character Entities, HTML5 Global Attributes, Creating Table Elements, Table Elements : Colspan/ Rowspan Attributes, border, cellspacing and cellpadding attributes, Creating Form Elements, Input Elements - Attributes, Color and Date Pickers, Select and Datalist Elements, Editing Elements, Media, Iframe, Why HTML Security, HTML Injection, Clickjacking, HTML5 Attributes & Events Vulnerabilities, Local Storage Vulnerabilities, HTML5 - Cross-browser support, Best Practices For HTML Web Pages.

HTML – Introduction

HTML stands for Hypertext Markup Language. It is the most basic language, and simple to learn and modify. It is a combination of both hypertext and markup language. It contains the elements that can change/develop a web page's look and the displayed contents. Or we can say that HTML creates or defines the structure of web pages. We can create websites using HTML which can be viewed on internet-connected devices like laptops, android mobile phones, etc. It was created by Tim Berners-Lee in 1991. The first version of HTML is HTML 2.0 which was published in 1999, and the latest version is HTML 5. We can save HTML files with an extension .html.

What is Hypertext?

Text that is not restricted to a sequential format and that includes links to other text is called Hypertext. The links can connect online pages inside a single or different website.

What is Markup Language?

Markup Language is a language that is interpreted by the browser and it defines the elements within a document using "tags". It is human-readable, which means that markup files use common words rather than the complicated syntax of programming languages.

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

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

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

HTML Page Structure

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:

A Simple HTML Document

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

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

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Output:

My First Heading

My first paragraph.

Example Explained

- The `<!DOCTYPE html>` declaration defines that this document is an HTML5 document
- The `<html>` element is the root element of an HTML page
- The `<head>` element contains meta information about the HTML page
- The `<title>` element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The `<body>` element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph

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
<title>	Defines the title that should be displayed on the browser tab
<meta>	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.

<style>	Defines style information for the web page
<link>	Defines a link to other documents like CSS
<script>	Defines script like JavaScript

HTML Need:

- HTML is needed to fill online forms in web applications
- Also, with the help of HTML, we can watch online videos on Youtube.
- While browsing the Web, when you click on a hyperlink, you navigate to another web page. This is possible with the help of HTML.
- web pages with different content types such as text, hyperlinks, forms, images, and videos are created in web applications using HTML.
- As web pages are read by computer programs like search engines and assistive tools, a web developer needs to ensure machine readability.
- HTML allows us to provide correct semantics (meaning) to web page content which in-turn enhances machine readability of web pages.

HTML- Case-insensitivity

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

Consider the code snippets below and observe the output

Code 1:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Homepage </title>
  </head>
  <body>
    Hello World!
  </body>
</html>
```

Code 2:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Homepage </title>
```

```
  </head>
  <body>
    Hello World!
  </boDy>
</html>
```

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

Best Practice: It is recommended to use lowercase for HTML tags as a best practice.

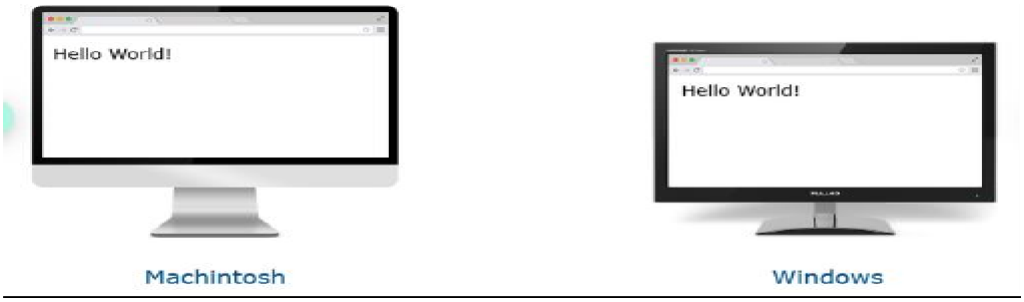
platform-independent

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

Sample.html

```
<!DOCTYPE html>

<html>

<head>

    <title>sample page</title>

</head>

<body>

   <p>Hello World!</p>

</body>

</html>
```

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



Cross-platform support

DOCTYPE Declaration

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

Sample.html:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Sample page </title>
  </head>
  <body>
     Hello world!
  </body>
</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.

Types of Elements

HTML elements can be further categorized into two as below:

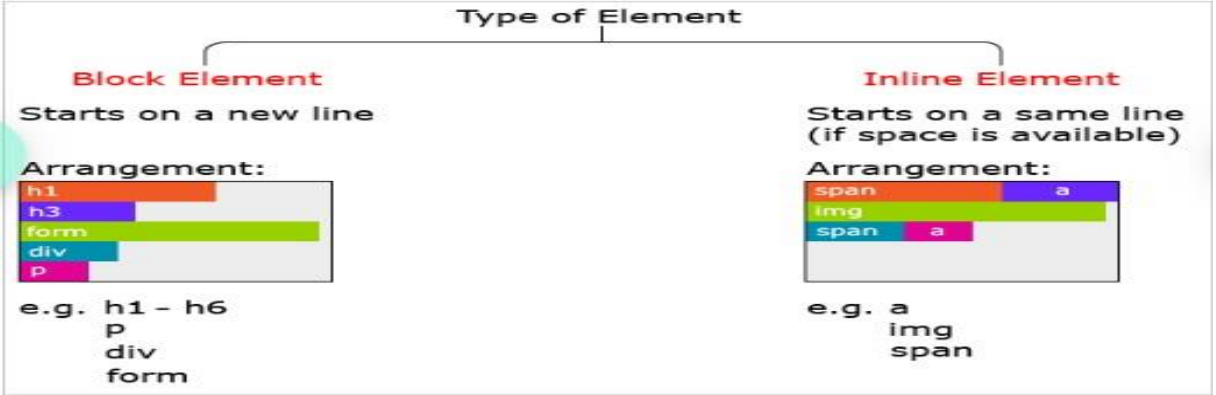
Block Element:

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 – Attributes

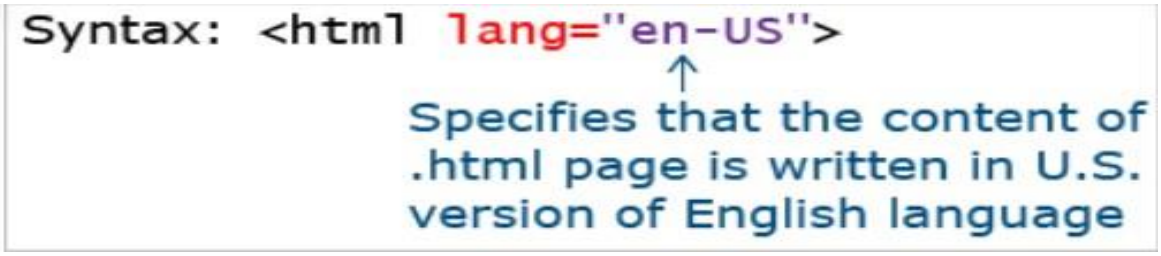
HTML elements can contain attributes that can be considered as an additional feature to set various properties and they are optional.

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.



Best Practices:

- Always quote attribute value since it will not perform as expected if the attribute value contains any special characters.
- It is recommended to use double-quotes for the values of the attributes.
- Use lowercase letters for attribute names for consistency.
- There should not be duplication for the same attribute in an element.

comments

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.

Comments are ignored by the browser.

Best Practices: Use proper comment lines for documentation on the HTML page when the application is in progress and delete them from the final code to reduce the page size and increase the page readability.

MetaData Element

Suppose you want to search tutorials for HTML5 on the web.In this scenario when you type the search string as "MDN" in any search engine and hit the Enter key, then the search engine generates a list of web pages as a result as seen below:

Note that there is a description associated with each website which helps the user to understand the summary of the web page even without opening it.

Let us see how this is possible.

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:

Let us learn the attributes of the metadata element.

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,

```
<!DOCTYPE html>
<html>
    <head>
        <title>Test page</title>
        <meta name="description" content="This is a test web page"/>
        <meta charset="utf-8">
    </head>
    <body>
        <p>Kindly check head section for meta tags</p>
    </body>
</html>
```

Best Practices:

- The title element can also be considered as part of the metadata.
- Any web page should have only one title tag per page.
- The title should be unique for each web page in the application.
- Start the title tag with the main keyword.
- Specify the character encoding of the document.
- Use UTF-8 encoding while designing the web page. It is not recommended to use ASCII incompatible encodings to avoid security risks because browsers that do not support them may interpret insecure content.
- Avoid duplicate descriptions inside metadata and try to include the targeted keywords in the description, as search engines index your page based on the description.
- Use the below http-equiv value to set the HTTP header with content-security-policy by specifying its values with relevant details. This is to update the browser to load the page required resources such as images, scripts to be loaded from the trusted origin only. Therefore helps in preventing security attacks such as cross-site scripting.

Example:

Below line of code in the web app update browser to load page required resources from 2 origins by specifying values to default-src as:

- <meta http-equiv="content-security-policy" content="default-src 'self' http://xyz.com">

  self indicates from the current domain to which page belongs to.

Explicitly mentioning the http://xyz.com value in the above example indicates that this domain is the trusted origin to load the resources.

- Use the below meta code line to specify the page content display to be adjusted to the device width being used to view the content with the initial zoom level as 100%. This is useful to display content on different kinds of devices such as desktop, laptop, and mobile devices.

1. <meta name="viewport" content="width=device-width, initial-scale=1.0">

You can change the zoom level depending on your requirement by specifying values to the initial-scale attribute as any positive value from 0.0 to 10.0. For example you can consider 0.1 = 10% zoom and 1.0 = 100% respectively.

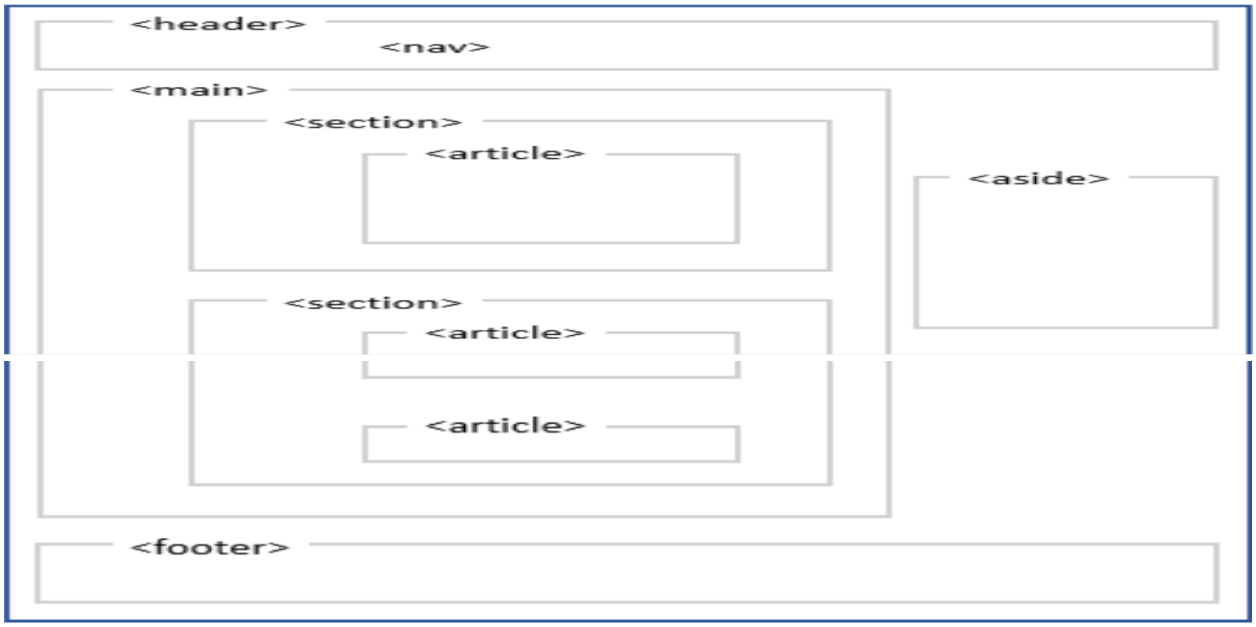
Sectioning Elements

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:

- 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.
- 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.
- 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.



Header and Footer Elements

Below are some of the semantic tags in HTML:

1) <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.

```
<header>
  <h3>About Us</h3>
</header>
```

2) <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.

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

3) <main>

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

```
<main>
  <section>
```

```
      ..
   </section>
   <section>
      <article>
         ..
      </article>
      <article>
         ..
      </article>
   </section>
</main>
```

4) <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>.

```
<nav>
   <a href="Home.html">Home</a>
   <a href="Login.html">Login</a>
</nav>
```

5) <section>

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

```
<main>
   <section>
      <p>Section 1</p>
   </section>
   <section>
      <p>Section2</p>
   </section>
</main>
```

6) <article>:

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

```
<article>
   <h1>MEAN stack</h1>
   <p>MEAN stack training includes discussion on MongoDB, Node,
 Express and Angular with the corresponding certifications</p>
</article>
```

7) <aside>:

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

```
<article>
   <h1>MEAN stack</h1>
   <p>MEAN stack training includes discussion on MongoDB, Node, Express and Angular with the corresponding certifications</p>
   <aside>
      <p>Visit our official website to attempt our certifications</p>
   </aside>
</article>
```

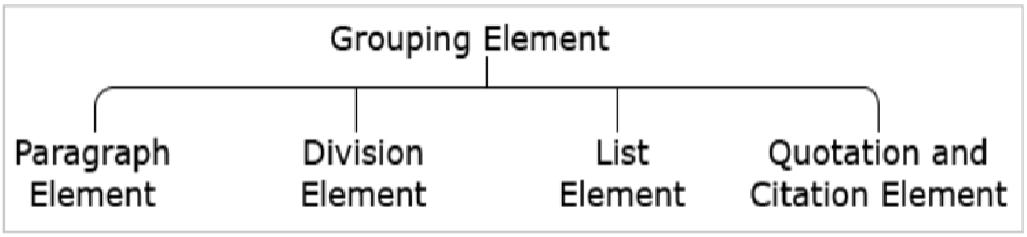
8) <address>:

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

```
<address>
   John
   #231A
   Palace Lane
   Bangalore
</address>
```

Grouping Elements in HTML

Grouping elements in HTML5 can be categorized majorly as below:



Let us see each of them in detail.

Paragraph Element

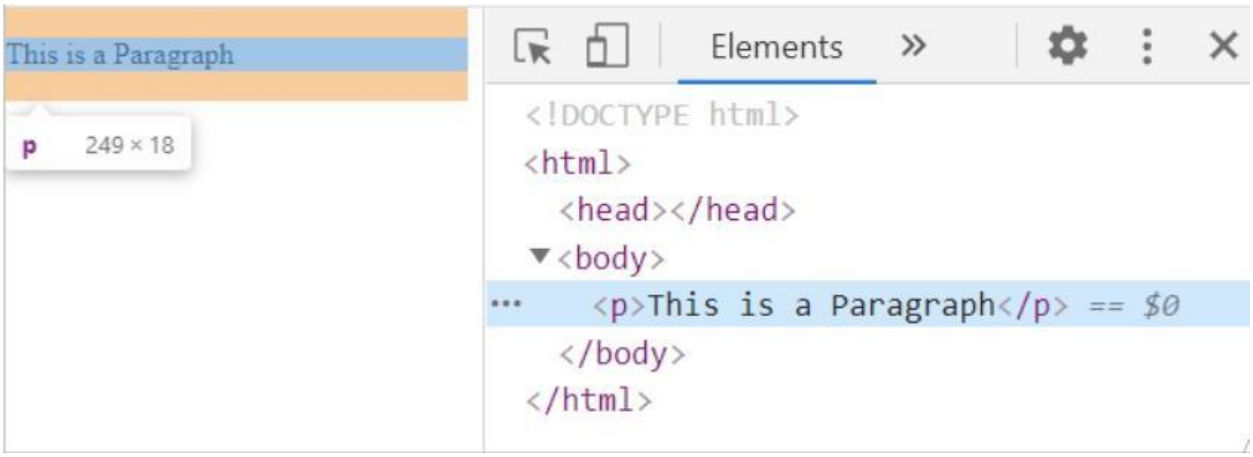
The paragraph element is generally used for denoting a paragraph. Any textual content can be mentioned inside this element.
It is defined using … tag.
Let us understand this with an example.
Copy the below code into your Visual Studio Code workspace.
demo.html

```
<!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 Element:

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 … tag.

Let us understand this with an example

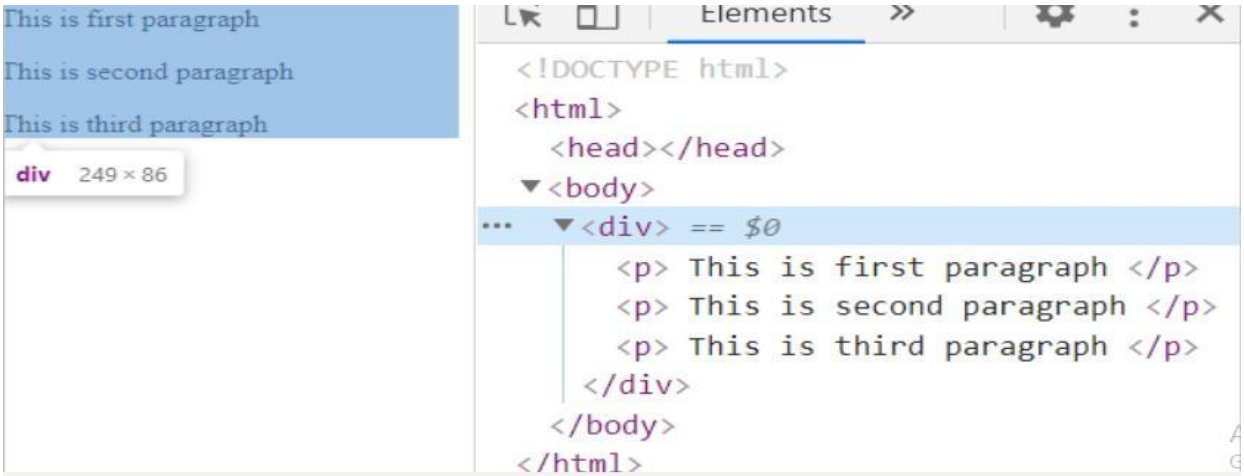
demo.html

```
<!DOCTYPE html>

<html>

  <body>

    <div>

      <p> This is first paragraph </p>

      <p> This is second paragraph </p>

      <p> This is third paragraph </p>

    </div>

  </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:



Span Element

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.

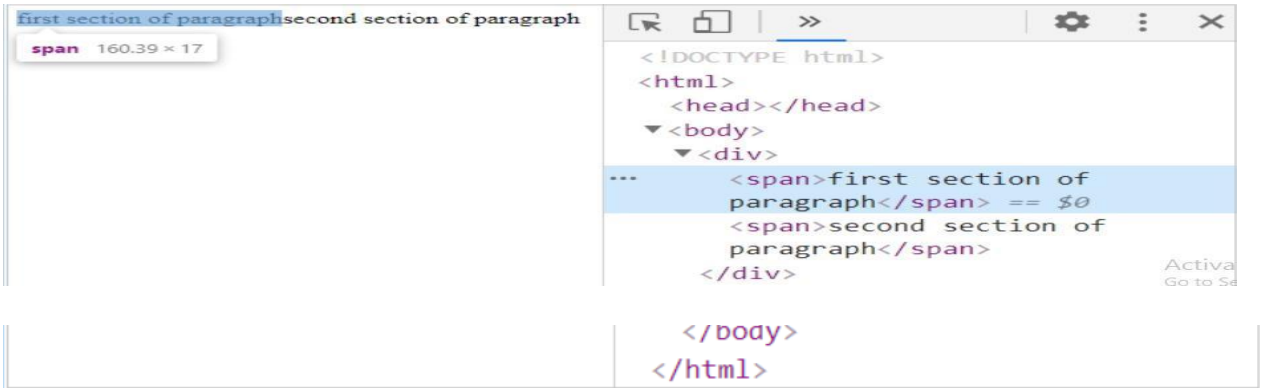
demo.html

```
<!DOCTYPE html>

<html>

   <body>

      <div>

         <span>first section of paragraph</span>

         <span>second section of paragraph</span>

      </div>

   </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:



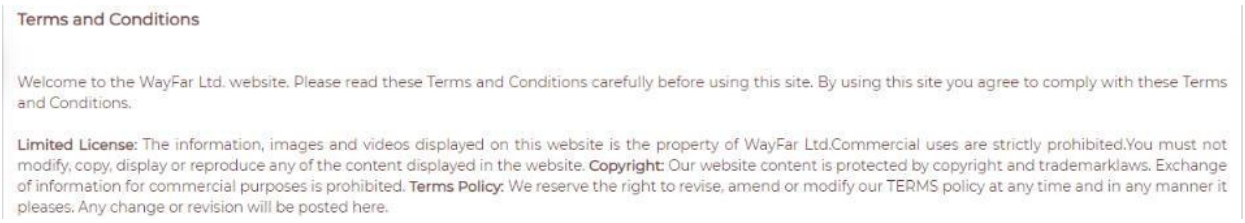
We can observe that no newline is added after the span ends since the span element is an inline-level element by default.

List – Need

We commonly come across requirements in our web applications where we need to group and display related items in an easily readable manner to the end-user.

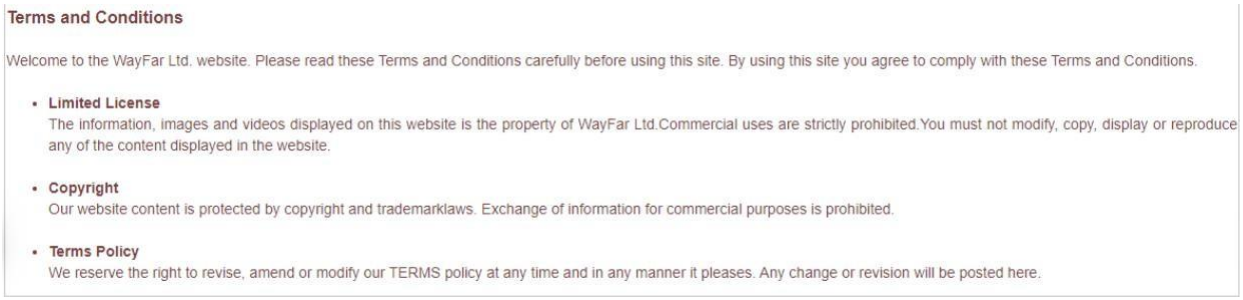
Consider, for example, while developing a tourism website. Terms and Conditions would be definitely one of the web pages which we might need to design for this website.

The terms could possibly be displayed as shown:

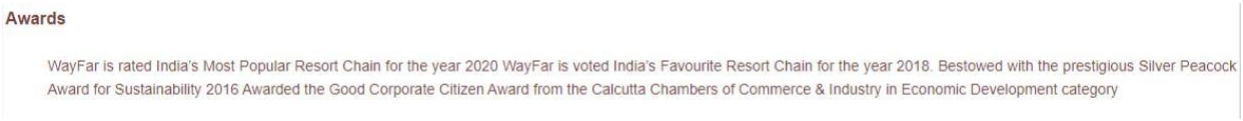


Here, the points are mentioned as continuous sentences in a paragraph. This way of presenting the terms and conditions would lead to difficulty for the reader to make out each point in a uniquely distinguishable and identifiable manner.

Let us learn how to redesign this page to improve readability, make the page as presentable to end-user as shown below:



Consider another scenario where we need to display different awards which have been won by the tourism company as shown below.



Here also, we surely can get to know the awards received. However, it is difficult to comprehend how many awards were won. Displaying the same data in a numbered list as shown below can help make the same content readable, and easy to comprehend details.

This is why we need lists on a web page.

With HTML Lists, we can group related items and display the items one after the other. This makes it easy to locate a particular item on the web page. In short, HTML Lists help in adding structure and order to related items on a web page, thereby ensuring better readability of related content.

Let us see different types of lists

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

Let us see each of them in detail.

HTML lists come in 3 basic flavors:

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

Each one has a specific purpose and meaning.

1) unordered list.

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 <ul> tag.
- Each item within the list technically referred to as 'list-item' enclosed within the <li> tag.

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

The following snippet can be used.

```
<h1>Courses offered:</h1>
    <ul>
        <li>HTML5</li>
        <li>CSS</li>
        <li>JS</li>
        <li>Bootstrap</li>


    </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:

The corresponding code to achieve this requirement is:

```
<h1>Courses offered:</h1>
   <ul style="list-style-type: square;">
       <li>HTML5</li>
       <li>CSS</li>
       <li>JS</li>
       <li>Bootstrap</li>

   </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.

Unordered List – Nesting

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:

Corresponding HTML code to achieve this requirement is:

```
<h1>Courses Offered:</h1>
        <ul>
            <li>Markup
                <ul>
                    <li> Basics of HTML
                        <ul>
                        <li> First level course on HTML </li>8.
                        </ul>
                    </li>
                    <li> Adaptive HTML </li>
                </ul>
            </li>
            <li>Styling
            <ul>
                    <li> CSS3
                        <ul>
                            <li> Latest version of CSS </li>
                        </ul>
                    </li>
                </ul>
            </li>


         </ul>
```

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

Ordered List – Nesting

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

Ordered lists can be nested with ordered lists or unordered lists.

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

The corresponding code to achieve this requirement is:

```
1.  <h1>Courses Offered:</h1>
2.          <ol>
3.              <li>Markup
4.                    <ol>
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.                   </ol>
12.             </li>
13.             <li>Styling
14.             <ol>
15.                       <li> CSS3
16.                           <ul>
17.                               <li> Latest version of CSS </li>
18.                           </ul>
19.                       </li>
20.                   </ol>
21.             </li>


22.           </ol>
```

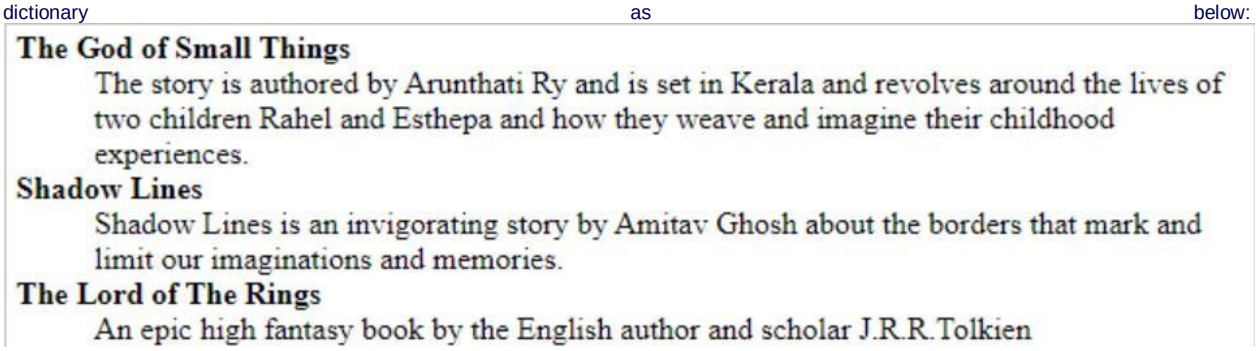
If you observe the above-given code snippet, based on the enclosure of the inner <ol> ... and <ul>... elements on various lines, the customized list has been populated on to the web page.

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:



- 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:

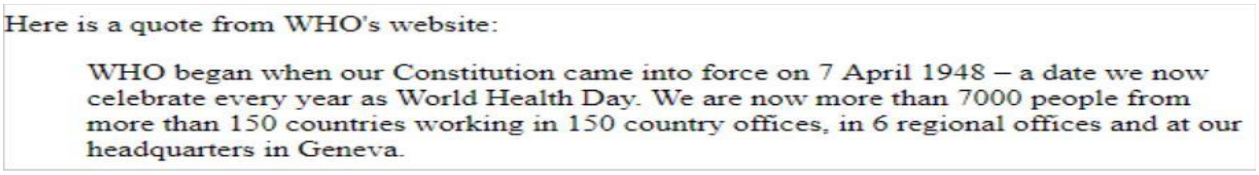
```
<dl>
<dt> Description Term 1 </dt>
<dd> Description Definiton 1 </dd>
<dt> Description Term 2 </dt>
<dd> Description Definition 2 </dd>


</dl>
```

Best Practice:

It is recommended not to use the <dl> tag while designing any dialogues concepts.

Quotation Element

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



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.

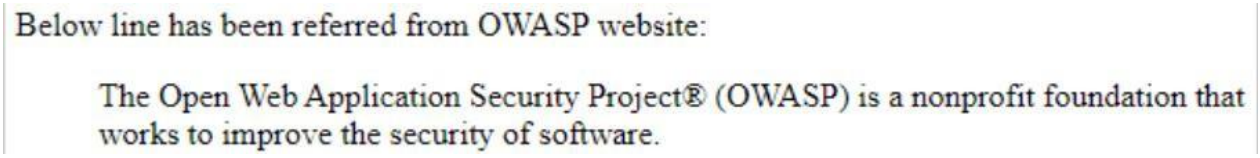
The quotation element also helps to include the citation, i.e., the URL from where the quote has been picked. This helps to retain the reference courtesy to the original site.

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

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

```
<!DOCTYPE html>

<html>

  <body>

    Below line has been referred from OWASP website:

    <blockquote cite="https://owasp.org/">

      The Open Web Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software.

    </blockquote>

  </body>

</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 – Need

A website necessarily is a collection of related web pages, where each web page is typically created for a particular purpose.

When developing any website:

- Each web page is necessarily coded in individual HTML files.
- To see a particular web page, the respective HTML file has to be opened up in a browser.

Below are some concerns pertaining to accessing the website developed:

- For a developer who has access to the source code of the individual HTML files, he or she can open the respective HTML on his browser. But if you are not a developer, you do not have access to the source code. Then how can you view the individual web pages?
- If you are trying to access the website over a network as a regular user, then also you will not be able to view the individual web pages unless you know the exact HTML file name and path.
- Moreover, trying to access individual web pages explicitly does not give a feeling that these web pages necessarily belong to one website or domain.

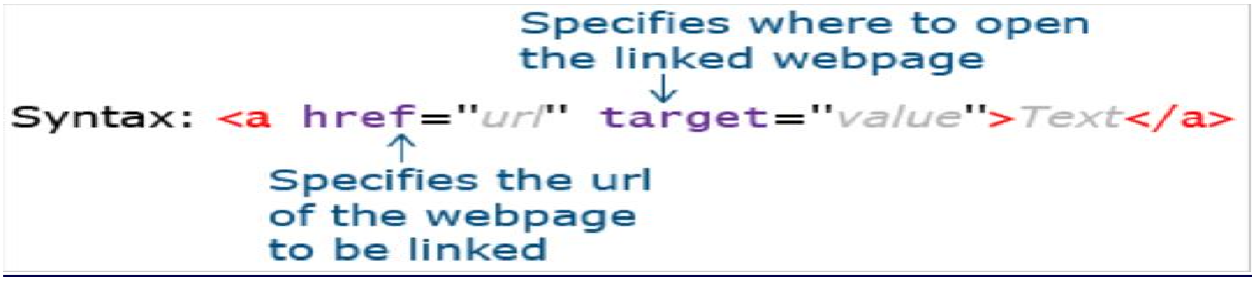
This is why the concept of hyperlinking documents or web pages have been introduced.

Below are the advantages if hyperlinks are used:

- We can create connections or links between HTML documents/web pages and users can navigate from one web page to another by clicking on "hyperlinks".
- We would now feel that we have a website which is a collection of interconnected web pages.

Link Element – Syntax

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".

Link ELement – Types

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.

Creating Link Elements

Text hyperlink:

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

```
<a href="Enquire.html"> Click here to connect to us </a><br/>

<a href="http://www.google.com"> Click here to go to Google website </a>
```

Image hyperlink:

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

```
<a href="http://www.google.com">
    <img src="google.jpg" />

</a>
```

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.

```
<h2 id="top">Topic</h2>
<p>Detail……</p>
<p>Detail……</p>
<p>Detail……</p>

<a href="#top">Go to Top</a>
```

Email hyperlink:

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

```


<a href="mailto:someone@xyz.com?Subject=Hello%20again">Send Mail</a>
```

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:

```
<a href="tel:+9999">Call Us</a>
```

Link Element - Target Attribute

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:

```
<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.

Link Element - Best Practices

Best Practices:

- It is a best practice to use href and rel attributes for links based on the requirement.
- It is recommended to use the values like noopener and noreferrer for the rel attribute in links to avoid security threats such as reverse tabnabbing.

For example:

```
<a href="http://example.com" target="_blank" rel="noopener noreferrer">
Example site

</a>
```

In the above example:

- rel=”noopener” attribute prevents the new page to be accessed by the window.opener property by ensuring it executes in a separate process.
- rel=”noreferrer” attribute is similar to noopener along with it prevents passing on the referrer details to the new page.

Character Entities

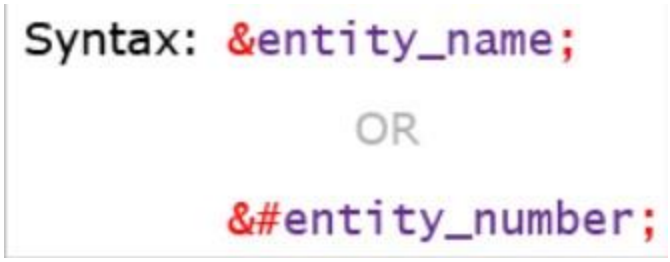
Some characters are reserved in HTML.

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.



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

Character	Description	Entity Name	Entity Number
	Non-breaking space		
<	Less than	&lt;	&#60;
>	Greater than	&gt;	&#62;
&	Ampersand	&amp;	&#38;
©	Copyright	&copy;	&#169;
€	Euro	&euro;	&#8364;
£	Pound	&pound;	&#163;
®	Registered trademark	&reg;	&#174;

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 unoque 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.

Need of HTML Table Elements

Consider the screen below from a web application on tourism, which displays details about the best time to visit different places.

Places, Jan-Mar, Apr-Jun, Jul-Sep, Oct- Dec
Ooty, No, Yes, Yes, No
Kodaikanal, Yes, No, No, Yes
Yercaud, Yes, Yes, No, Yes
Coonoor, Yes, No, No, Yes

Assume that we need to now quickly find out which of the places are best to be visited in the Jul-Sep quarter.

It is quite difficult to comprehend this information from this display. We will need to go line by line and find out yes/no by looking at the order of data displayed in each line. If we miss the correct order of data in each line, there is a possibility of miscalculating the required information. If this data is displayed in a 'tabular format', we can quickly get the required data.

With a tabular format of data display, the information we need can be easily interpreted by making visual associations between row and column headers.

This is why we need HTML Tables. HTML Tables provide a means by which we can create a tabular format of data on the web.

Organizing Data as Tables – Examples

An HTML table is a great way to display data that make more sense in spreadsheet formats as shown below.

Tour package information and comparisons:

Places to Visit	Best time to Visit			
	Jan-Mar	Apr-Jun	Jul-Sep	Oct-Dec
Ooty	No	Yes	Yes	No
Kodaikanal	Yes	No	No	Yes
Yercaud	Yes	Yes	No	Yes
Coonoor	Yes	No	No	Yes

Displaying price comparisons:

COMMODITY PRICES

Commodity	April, 2018	April, 2019	% Variation
English Beans	Rs 45.71	Rs 99.86	118.46
Tomato	Rs 20.79	Rs 39	87.59
Carrot	Rs 34.57	Rs 57.07	65.09
Shallots	Rs 35.71	Rs 46.21	29.40
Garlic	Rs 7.50	Rs 10.21	36.13
Green Peas	Rs 43.17	Rs 66.58	54.23

Nutritional facts:

Nutritional Value Comparison

	Apple (1 cup)	Banana (1 cup)
Calories	74	138
Fat (g)	1	0.8
Carbs (g)	19	36
Protein (g)	0.2	1.6

HTML table arranges content into rows and columns and can be used on websites for the effective display of information in a tabular structure.

We can create a simple table, with or without borders.

Packages	Destinations	Package Details
Family Packages	UAE: Land of Skyscrapers	1N Burj Khalifa, 1N Dubai Mall, 1N Safari Desert ₹ 59640 per head
Affordable Package	India: Incredible Collision of Culture.	2N Manali, 2N Shimla, 1N Agra, 2N Jaipur ₹ 41250 per head
Best Seller Packages	Thailand: A country of contrasts.	1N Phuket, 1N Krabi, 1N Bangkok, 2N Similan Island ₹ 86000 per head

Packages	Destinations	Package Details
Family Packages	UAE: Land of Skyscrapers	1N Burj Khalifa, 1N Dubai Mall, 1N Safari Desert ₹ 59640 per head
Affordable Package	India: Incredible Collision of Culture.	2N Manali, 2N Shimla, 1N Agra, 2N Jaipur ₹ 41250 per head
Best Seller Packages	Thailand: A country of contrasts.	1N Phuket, 1N Krabi, 1N Bangkok, 2N Similan Island ₹ 86000 per head

We can also create a complex tabular structure where contents can span across multiple columns/multiple rows.

Packages	Destinations	Package Details	Best Time to Visit		
			Mar-Jun	Jul-Oct	Nov-Feb
Family Packages	France: Paradise of Love and Life	3N Paris, 2N French Riviera, 1N Biarritz ₹ 108000 per head	✓		✓
Affordable Package	India: Incredible Collision of Culture.	2N Manali, 2N Shimla, 1N Agra, 2N Jaipur ₹ 41250 per head		✓	✓
Best Seller Packages	Thailand: A country of contrasts.	1N Phuket, 1N Krabi, 1N Bangkok, 2N Similan Island ₹ 86000 per head			✓

Additionally, an HTML table can also have a caption.

TOUR PACKAGES

Packages	Destinations	Package Details
Family Packages	UAE: Land of Skyscrapers	1N Burj Khalifa, 1N Dubai Mall, 1N Safari Desert ₹ 59640 per head
Affordable Package	India: Incredible Collision of Culture.	2N Manali, 2N Shimla, 1N Agra, 2N Jaipur ₹ 41250 per head
Best Seller Packages	Thailand: A country of contrasts.	1N Phuket, 1N Krabi, 1N Bangkok, 2N Similan Island ₹ 86000 per head

Let us see how we can design tables in HTML.

Table Element – Syntax

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:

```
<table>
    <caption>Table heading</caption>
    <thead>
        <tr>
            <th>Column 1 heading</th>
            <th>Column 2 Heading</th>
```

```
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Column 1 data</td>
                <td>Column 2 data</td>
            </tr>
        </tbody>

</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.

Table Element - Example 1

Consider the below code snippet:

```
<html>
  <table>
    <thead>
      <th>Back-end JS </th>
      <th>Front-end JS </th>
    </thead>
    <tbody>
      <tr>
        <td>Node</td>
        <td>React</td>
      </tr>
      <tr>
        <td>Express</td>
        <td>Angular</td>
```

```
        </tr>
      </tbody>
   </table>
</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.

Let us have a glance over another table requirement on the next page.

Table Element - Example 2

Suppose the developer has to design an HTML table as shown below:

MERN stack developer program				
Technology	MongoDB	Node	Express	React
Description	The database	Node is the base for server side scripting	JS framework for web server development	JS library to design the front end

Note that, Node and Express are logically related columns, and the same styling should be applied for both.

The <colgroup> element in HTML helps us to group the related columns especially to provide some common CSS property.

The HTML code snippet to achieve the above requirement is:

```
<table>
    <caption>MERN stack developer program</caption>
    <colgroup>
        <col>
        <col style="background-color:lightblue;">
        <col span="2" style="background-color: lightgreen;">
        <col style="background-color: lightpink;">
    </colgroup>
    <tr>
        <th>Technology</th>
        <td >MongoDB</td>
        <td >Node</td>
```

```
13.         <td >Express</td>
14.         <td>React</td>
15.     </tr>
16.     <tr>
17.         <th >Description</th>
18.         <td>The database</td>
19.         <td>Node is the base for server side scripting</td>
20.         <td>JS framework for web server development</td>
21.         <td>JS library to design the front end</td>
22.     </tr>
23. </table>


24.
```

In the above-mentioned code we can observe that:

- The <table> element encloses all the other table elements in it.
- Line numbers 3-7 code which has been enclosed between the <colgroup> element helps us to logically group the related column data and also customize the default look and feel of the respective column through CSS properties.
- The <colgroup> element has an attribute "span" which carries the information regarding how many columns are to be grouped together. The default value of this span attribute is 1.
- The <tr> element helps to create a new table row.
- The <td> element helps to create a new table column.

Let us learn some of the best practices which can be used while designing the HTML table in our application.

Best Practices:

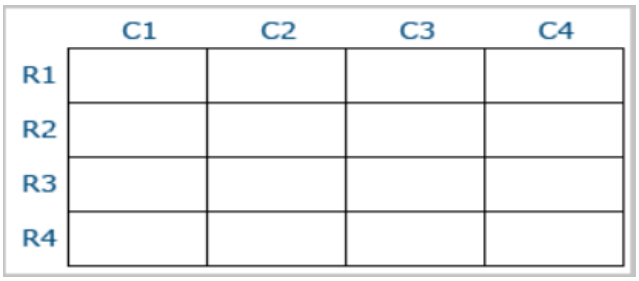
- It is important that the table should be accessible to visually impaired users and hence it is recommended to provide a table description using the <caption> element.
- It is a best practice to associate table headers <th> elements with their <td> cells.
- A table can be styled using appropriate CSS properties like height, and width for customizing the default look and feel of this element.

Table elements-Colspan and Rowspan Attributes

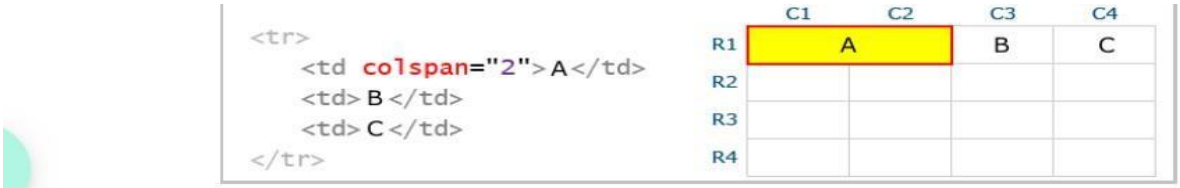
The elements <td > and <th> supports the attributes namely colspan and rowspan which helps to merge the table cells accordingly.

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.



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



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

Need of Form Element

In a web application, providing user's inputs is required in various contexts such as:

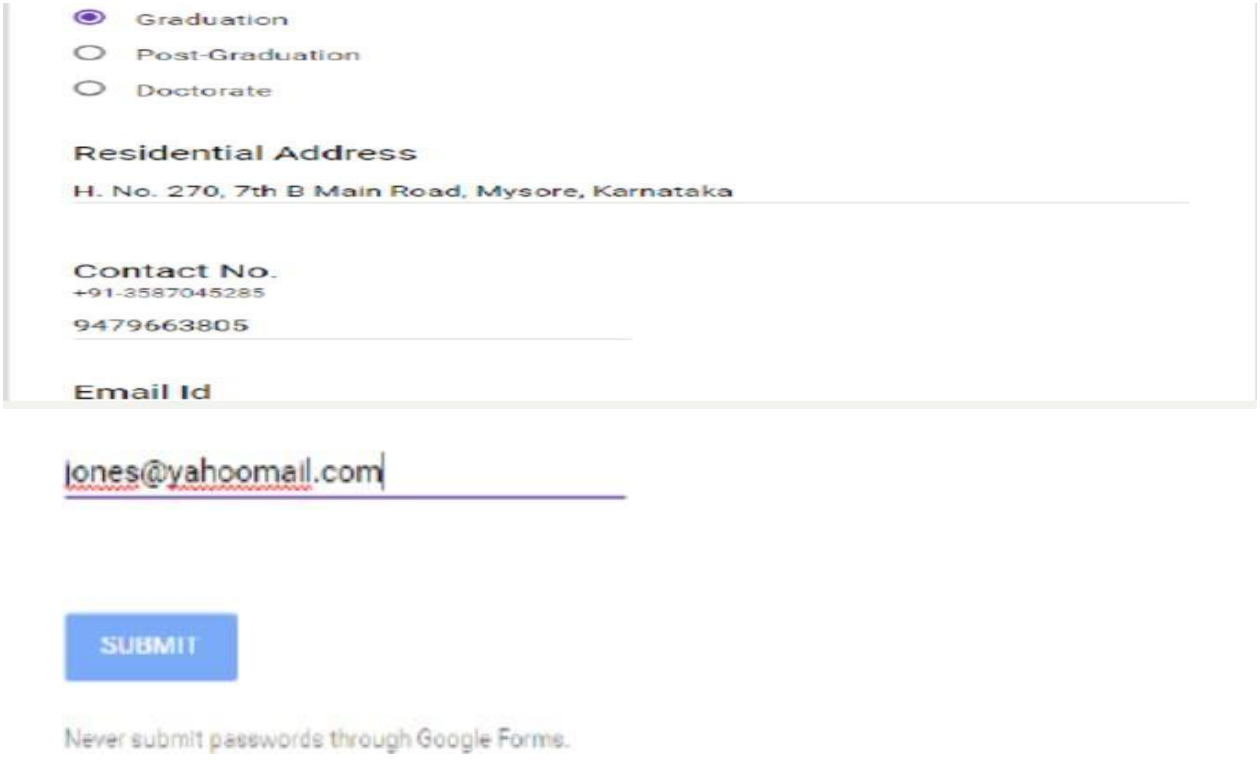
- Provide users' information to create his/her account with a website
- Provide a username and password to login into the account.
- Provide feedback on a tour package the user had used, etc.

There has to be a mechanism through which a website can capture user inputs. This is why HTML forms have been introduced.

HTML Forms, also known as Web Forms, help in capturing information from the user of a web application.

Users can key-in the details such as name, email, phone numbers, comments, dates, and other needed values using the HTML form inputs. Users can also select from a predefined set of values.

A sample form is illustrated below:



Form Elements – Syntax

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

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

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
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 them in detail.

Input Types - Text, Password, Email and Number

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.

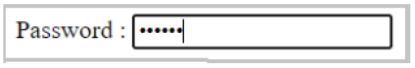
```
1. Name: <input type="text" value="">
```

Name : John

Input type - password:

An input field can be used to enter a password.

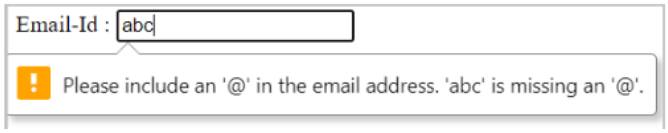
```
1. Password: <input type="password">
```



Input type - email:

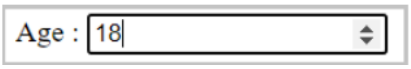
- An input field that accepts email addresses.
- It has in-built validation for an email.

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



Input type - number:

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



Input Types - Checkbox, Radio and File

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:

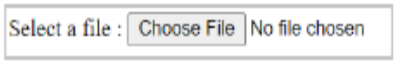
- Defines a radio button.
- The name attribute specifies the associated name of that radio button.
- Radio buttons in a group should have the same name.

```
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.

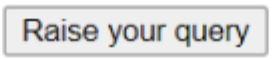
```
1. Select a file: <input type="file">
```



<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)

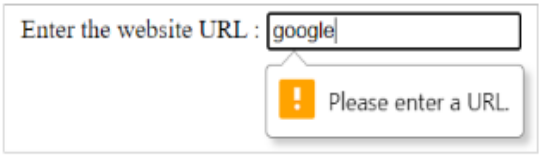
```
1. <button type="submit">Raise your query</button>
```



Input type - URL:

Defines a text input that can capture any input value starting with http:// or https://. If there is a pattern mismatch, it shows an error on form submission.

```
1. Enter the website URL: <input type="url">
```



<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

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:

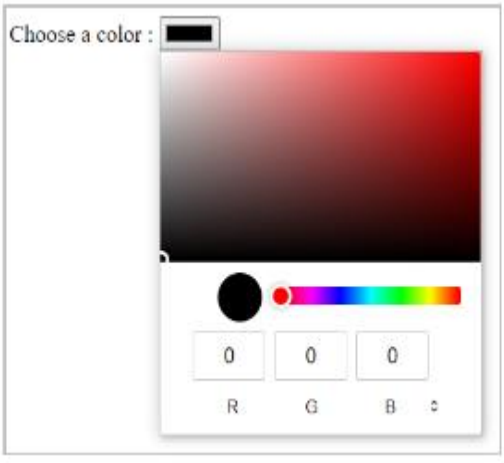
We have various picker-elements in HTML forms such as color-picker and date-picker elements.

Let us see them in detail.

Input type - color:

Defines a color picker.

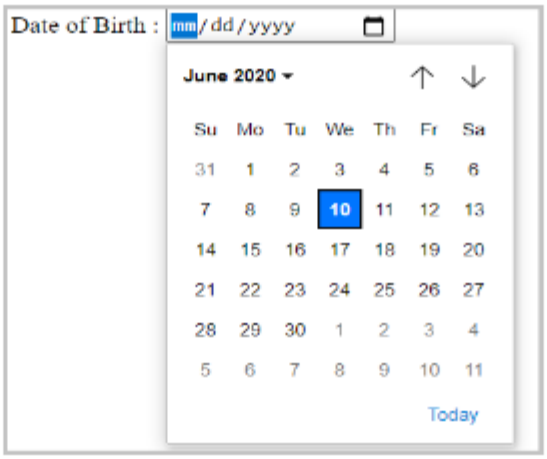
```
1. Choose a color: <input type="color">
```



Input type - date:

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

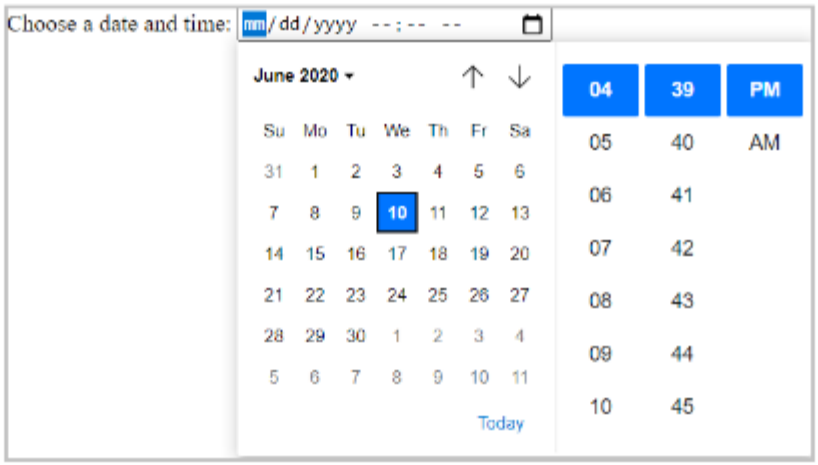
```
1. Date of Birth: <input type="date">
```



Input type – datetime-local:

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

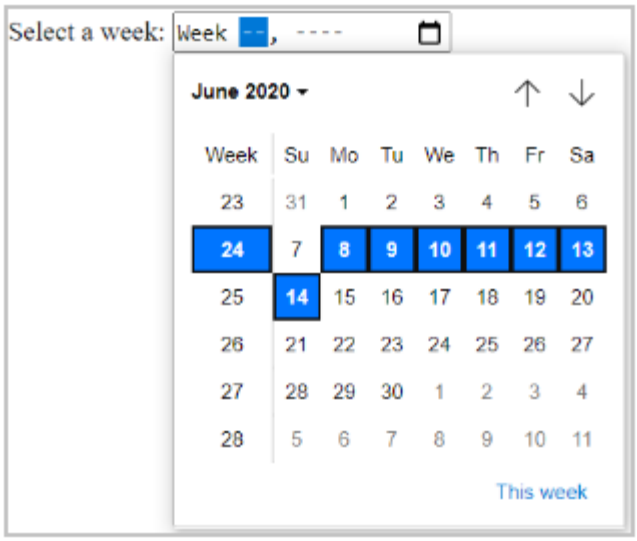
```
1. Choose a date and time: <input type="datetime-local"/>
```



Input type – week:

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

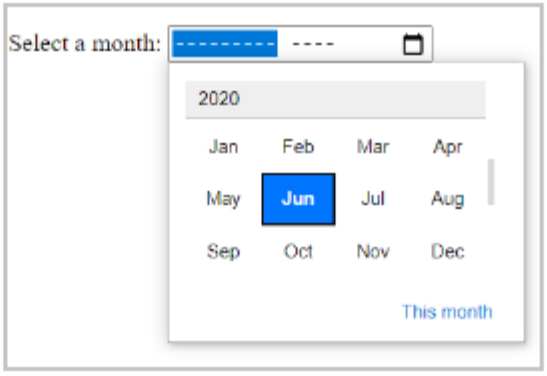
```
1. Select a week: <input type="week"/>
```



Input type – month:

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

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



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

Select and Datalist Elements:

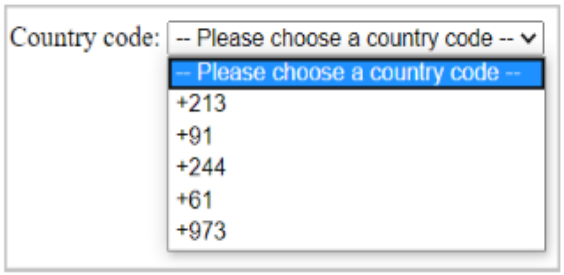
We have <select> and <datalist> elements in HTML which helps to collect input data from user as a drop-downs.

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.

```
Country code: <select >
    <option value="">-- Please choose a country code --</option>
    <option value="+213">+213</option>
    <option value="+91">+91</option>
    <option value="+244">+244</option>
    <option value="+61">+61</option>
    <option value="+973">+973</option>
</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.

```
Country: <input list="countries">
<datalist id="countries">
    <option value="India">
        <option value="France">
            <option value="Singapore">
                <option value="Thailand">
                    <option value="United Arab Emirates">
                        <option value="United States of America">
</datalist>
```

Note: <select> allows the user to select from some pre-defined options.

Whereas, for the <datalist> element, even though it is suggested to select from the given options, the user can actually enter the data to the input field as any other input field.

Also, Output may vary for some input elements according to the browser

Other Form Elements:

Let us see some other input elements in HTML where the data can be collected from the user as a slider.

Input type – range:

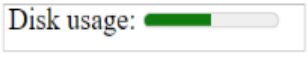
Defines a range slider, where the user can select input.

```
1. Volume: <input type="range"/>
```

Meter:

Can be used to represent a scalar measurement within a known range.

```
1. Disk usage: <meter min="0" max="100" value="50"></meter>
```



Progress:

Can be used to represent the progress of a task.

```
1. Task completed: <progress min="0" max="100" value="50">50 of 100</progress>
```

Note: <progress> can be used to mark up the completion rate/degree of progress of an "in progress" task through a progress bar.
Whereas, <meter> is used to represent a gauge or to represent a measurement in a known scale.

Output:

Displays the output of user input.

For example, consider the below code snippet:

```
1. <form oninput="result.value=parseInt(a.value)+parseInt(b.value)">
2.   <input type="number" id="b" name="b" value="10" /> +
3.   <input type="number" id="a" name="a" value="10" /> =
4.   <output name="result" for="a b">20</output>
5. </form>
```

The 'oninput' attribute carries the logic of generating the output, and the 'for' attribute of <output> tag specifies the control for which output has to be calculated.

The above-given code would populate two input fields on the web page and the sum of the values of these two input fields is generated dynamically based on user activity on this page.

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

Note: A disabled attribute can be used with any input type to restrict user interaction.

For example, a button can be made non-clickable, Checkbox can be made non-selectable.

The output may vary for some input elements according to the browser.

Input Elements - Attributes:

The following are some of the attributes which can be used with HTML input elements.

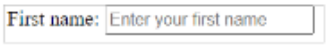
- Placeholder
- Pattern
- Min
- Max
- Step
- Required
- Multiple
- Form-override

Let us see each of them in detail.

Placeholder:

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

```
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.

```
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 elem

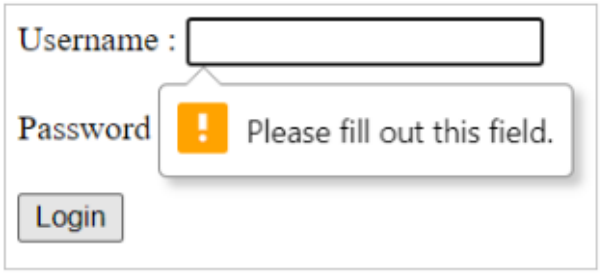
```
1. Job experience: <input type="number" min="2" max="10" step="1"/>
```

Required:

The required attribute specifies that user input is a must.

If the user does not enter any value in the input field which is associated with this attribute, a default error messa appears on the screen.

```
1. Username: <input type="text" required/>
2. Username: <input type="password" required/>
```



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.

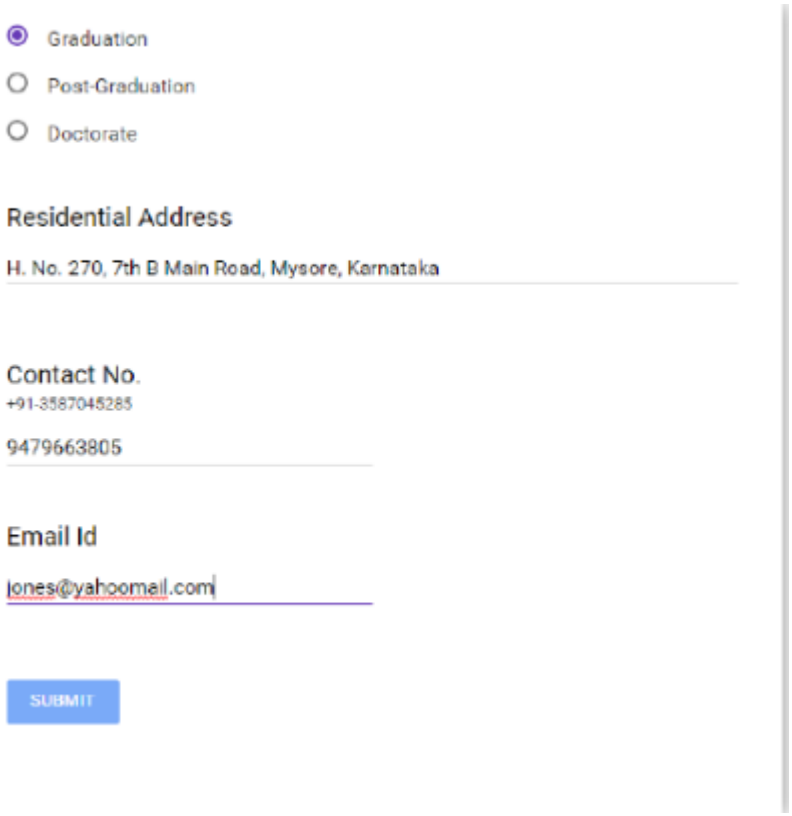
Form-override attribute	Description
formaction	Overrides the form action attribute
formnovalidate	Overrides the form novalidate attribute
formmethod	Overrides the form method attribute
formtarget	Overrides the form target attribute

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>
```

Need of novalidate attribute:

Suppose we are developing a form having multiple input fields as shown below:



To test the form's functionality we may want to temporarily by-pass in-built validations done by form input type elements. This can be done by novalidate attribute.

For example, if we want to bypass an email validation, you can use the below code:

```
<form novalidate action='xyz.html'>
<input type="email"/>
<input type ="submit">
</form>
```

HTML has autocomplete and autofocus attributes for the below-mentioned functionalities.

autocomplete:

It allows the browser to predict user input value.

When the user starts typing, the browser displays possible options, based on earlier typed values.

Possible values of autocomplete are on and off where the default value is "on"

autofocus:

Specifies that an element should automatically get focus when a web page loads.

Example:

```
Email:<input type="email" autocomplete="on"/> <br/>
Contact: <input type="number" autofocus/><br/>
<input type="submit"/>
```

Hidden:

In a web page, when an element is not directly relevant to the page's current state, it's a good choice to hide it.

The hidden attribute is used for this requirement.

Possible values are " " (Empty string) or hidden.

For example, the below-given input field for the language will not be shown to the end-user.

```
<input name="language" value="English" hidden >
```

Editing Elements:

While developing any content-based application, there may be a requirement to get it reviewed.

While reviewing the content of our web page, the reviewer may want to add or delete some content.

For this scenario, editing elements can be used.

Developer

Reviewer

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>
```

HTML is building block of Internet

HTML is building block of ~~Internet~~ WWW

Form Elements - Best Practices:

The following are the best practices to be followed while designing an HTML form.

- Do not nest a form inside another form. If forms are nested this leads to unpredictable behavior of the form.
- Use <label> tag to wrap each form-controls.
- Use appropriate Input types for <input> element.
- Avoid placeholder attribute while designing Label for a web page.
- Include min and max attributes appropriately for <meter> and <progress> elements.
- All the form input values should be sanitized and validated to avoid security threats such as HTML injection.

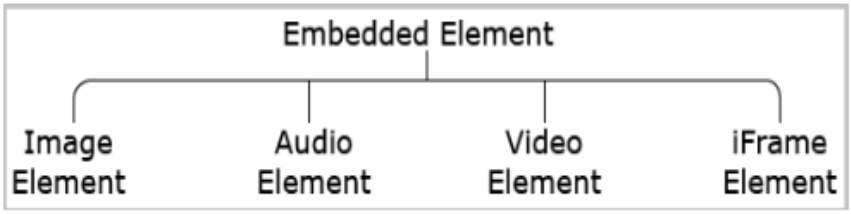
Any website must be able to engage well with its visitors, be entertaining, and be able to quickly deliver information.

Embedding content like audio clips, videos, images, maps, and so on... are a great way of engaging, be entertaining and be able to quickly deliver information to the website users.

Pictures or moving pictures, maps, etc. typically draw user attention and trigger quite a lot of emotions. Humans find it easy to connect to such information, rather than having to go through textual information.

This is why HTML provides tags for embedding media content like audio, video, and images and also for embedding external content like maps.

HTML5 supports the following types of embedded elements:



Let us see each element in detail.

Image Element:

Embedding images to a web page can be done with the <img>...</img> tag. The <img> 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. <img src="Summit.png" alt="image unavailable"/>
```

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.     <img src="tropicalSea.jpg"
3.          alt="An image of tropical sea"
4.          width="400"
5.          height="341">
6.
7.     <figcaption>A colorful tropical sea view</figcaption>
8. </figure>
```

The above code snippet generates an output as below:

A colorful tropical sea view

Note: The <figure> element in HTML is commonly used along with a diagram, illustration, or image to represent them as self-contained content.

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 contol bar may differ from browser-to-browser)

(Appearance of contol 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 <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. <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="controls">
    Your browser does not support video tag
</video>
```

Output:

(Appearance of contol 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.

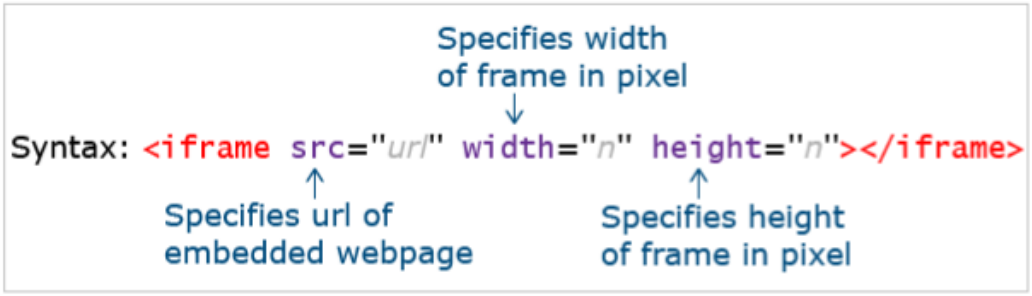
Also, it is suggested to use the <source> element within the audio/video element instead of the src attribute.

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

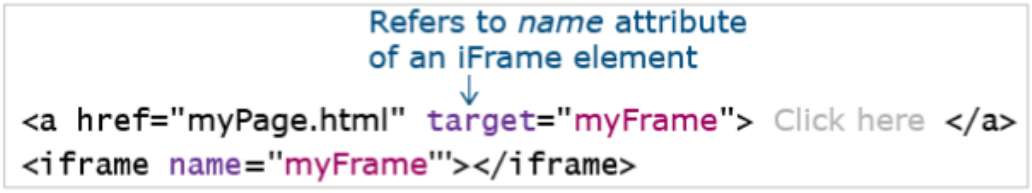
We might have requirements to include documents, videos, interactive videos, or even other web content into a web page directly from external sources.

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.



Embedded Elements - Best Practices:

- Use meaningful descriptive text for the alt attribute.
- The src attribute should not be empty.
- Ensure the usage of sandbox attribute to the iframe to avoid security threats such as clickjacking.

Let us discuss the sandbox attribute in detail.

Sandbox Attribute

This attribute helps the developer to apply extra restrictions for the content which can be embedded into the iframe element.

Some of the scenarios where this attribute can be used are:

- To validate whether the content is populated from a unique origin or not.
- To block the form submission and script execution.
- APIs invocation can be disabled.
- Other browser context linking and usage of various plugins like <applet>, <embed> can be prevented.
- Navigating to a Top-level browsing context can be avoided.
- Auto video/audio pre-load features can be restricted.

Example:

```
1. <iframe sandbox="">
```

The sandbox attribute with an empty value enables all the default restrictions automatically.

Also, the developer has been provided with different attribute values for disabling respective restrictions based on their requirement of the application.

Now let us have a glance over some of the attribute values.

Some possible values of the 'sandbox' attribute are shown below:

Value	Description
(*no value*)	Enables all restrictions
allow-forms	Allows users to submit the form
allow-scripts	Scripts execution would be enabled.
allow-same-origin	iframe content would be considered as accessed from the same origin
allow-popups	Popups can be enabled.
allow-popups-to-escape-sandbox	Popups can be opened in new windows without any sandbox restrictions.
allow-orientation-lock	The orientation of the Screen can be locked.
allow-pointer-lock	Pointer Lock API can be accessed.
allow-presentation	Session Presentation is enabled for the user.
allow-top-navigation	iframe content can access its top-level browsing context.
allow-top-navigation-by-user-activation	iframe content can access its top-level browsing context only through user interaction.

For example, the below-given code demonstrates how to enable Form submission and Script execution for xyz.html page.

```
1. <iframe src="xyz.htm" sandbox="allow-forms allow-scripts "></iframe>
```

Why HTML Security?

Let us now understand why we need to take care of vulnerabilities in HTML5.

We use HTML to create static web pages. HTML5 has introduced some new features which make web pages richer. New features include new semantic elements like 'header', 'footer', etc., new attributes for form elements like date, time, range, etc., new graphic elements like SVG and canvas, and new multimedia elements like audio and video.

Web developers can use the new features in HTML5 for building hybrid applications that can run on the web and mobile devices. Lots of data flow has to be handled in these applications, therefore developers should take care of the attacks that are possible as well. For example, an attacker can steal the data by inserting some wicked code through HTML forms which will be kept in the database. Security flaws are possible if proper security measures are not taken when using HTML5

features like communication APIs, storage APIs, geolocation, sandboxed frames, offline applications, etc.

Therefore there is a need to find the attacks possible in all-new HTML5 features and their preventive measures to defend against those attacks.

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

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:

```
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:

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

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

Clickjacking:

Let us now understand another attack called clickjacking and its consequences.

Hackers will have several ways to trick the user and make them click on something which they are not supposed to. A common form of this attack involves mirroring a login form on a website. Users will think that they are entering values in the actual HTML form but they are actually entering them in the form fields that are overlaid on that web page. The data which the user enters will be sent directly to an attacker's page. In this type of attack, hackers will usually target sensitive data like passwords, bank account information, credit card information, etc.,

Below are some real-time scenarios of clickjacking attacks:

1. There was an attack on one of the popular browser plugins, where an attacker added a modified page in an iframe and can trick a user into changing settings of the plugin to get access to the local system's resources such as a microphone, camera, etc.

2. A popular social networking site was attacked by tricking users to click on a button that will make them retweet the location of the malicious web page which will be propagated hugely.

Clickjacking is frequently used to hijack accounts for spamming purposes or navigate links to malicious web pages on online social network websites.

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.

There 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:

```
<script>
   // frame busting
   if (self == top) {
      document.documentElement.style.visibility = 'visible';
   } else {
      top.location = self.location
   }
   function register() {
       alert('Registered')
   }
</script>
```

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

2. Server-side methods: Security experts recommend server-side methods to be the most effective methods to defend against clickjacking. Below are the two response headers to deal with this.

(i) Using X-Frame-Options response header.

(ii) Using Content Security Policy(CSP) response header.

X-Frame-Options:

It is a response header that is to be set as part of the HTTP response of a web page. It specifies whether a browser should be permitted to show a web page inside a <frame> or <iframe> tag.

The below values can be set for the X-Frame-Options header:

(i) SAMEORIGIN – This choice permits the current page to be displayed in a frame on another page, but only within the current domain.

(ii) DENY – This choice does not permit any domain to render the current page within a frame. This is the most secure option to provide restriction.

(iii) ALLOW-FROM URI – allows the current page to be displayed in a frame, but only in a specific URI – for example, www.example.com/frame-page.

Limitations of X-Frame-Options:

(i) The X-Frame-Options header should be kept as part of every HTTP response for a website on setting the value as SAMEORIGIN.

(ii) X-Frame-Options doesn't work with multi-domain sites

(iii) Any one of the X-Frame-Options header values is allowed to be used on a web page. It is not likely to display a page as a frame both on the current website and on the external website.

(iv) X-Frame-Options is deprecated in most of the current browsers.

So, now the latest defense mechanism for clickjacking is to use CSP.

Content-Security-Policy (CSP):

Content Security Policy is a standard to mitigate and detect numerous content injection attacks. It is an HTTP response header that provides different directives that are used to limit how and from where content can be loaded into a web application. A browser that is compatible with CSP will execute only the scripts mentioned in the allowed list of domains.

To configure Content Security Policy, we need to set the "Content-Security-Policy" response header or in the "<meta>" tag of HTML. Set the values to control the resources that the browser is allowed to load for that particular page.

The policy can be specified as shown below:

```
Content-Security-Policy: policy
```

The policy can be set using different directives that define the policy for a certain resource type. One of the directives "frame-ancestors" can be set to protect against clickjacking which denotes if a browser should be allowed to embed a page in an <frame> or <iframe>.

Examples:

```
Content-Security-Policy: frame-ancestors 'none'
```

This doesn't allow any domain to embed a web page in a frame.

```
Content-Security-Policy: frame-ancestors 'self'
```

This allows only the current page to be embedded inside a frame.

```
Content-Security-Policy: frame-ancestors 'self' '\*.example.com'
'https://mywebsite.com';
```

This allows the current page, any page from the example.com domain, and only the mywebsite.com page to be embedded in a frame.

It is recommended to set the CSP directives in the response header at the server configuration level as it handles the entire application.

Note: If there is a requirement to use an iframe in the application, then use the sandbox attribute to prevent clickjacking.

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

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

```
<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

```
<video src="/apis/authContent/content-
store/Infosys/Infosys_Ltd/Public/lex_auth_01278231776602521628 9/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 <b>, <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 &#62
"	" 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. <a href="#" onmouseover="alert('hacked')">Avengers</a>
```

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

```
1. &lt;a href="#" onmouseover="alert('hacked')"&gt; Avengers &lt;/a&gt;
```

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.

DOMPurify is used for sanitizing HTML code. It is written in JavaScript and works in all modern browsers. This library strip out dangerous HTML and prevents XSS attacks by sanitizing the HTML code. On feeding string full of dirty HTML code to this library returns a string with clean HTML. Many security experts have already reviewed this library and identified that it is the recommended library for HTML sanitization.

To use this library on the website, just include it as shown below.

To use this library on the website, just include it as shown below.

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

Any dirty code can be sanitized using the below method.

```
var clean = DOMPurify.sanitize(dirty);
```

Using document.write() method or using innerHTML property, the resulting HTML code can be written into DOM.

It is very important to identify a suitable sanitizer library for securing your application.

Local Storage Vulnerabilities:

In our web applications, we often store some data in the browser cache. As the data is stored at the client-side, there is a chance of data-stealing by injecting some malicious code, if no proper care is taken. Let us now see how to store the data properly to prevent such attacks.

HTML5 has introduced Web storage or offline storage which deals with storing data in a local cache. Data can be stored using two types of objects in HTML5. Local storage and Session storage. These storages hold data in the form of key-value pairs.

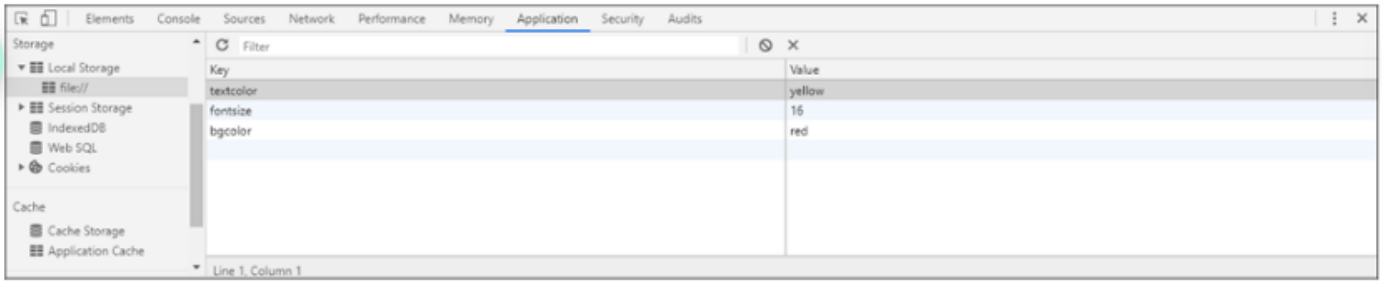
Local storage holds the data in the browser cache until the user deletes it or it expires based on the expiry date given. setItem() method is used to assign data to local storage. The below code creates three items with names bgcolor, textcolor, fontsize and assigns the values to them.

```
localStorage.setItem("bgcolor", document.getElementById("bgcolor").value);
localStorage.setItem("textcolor", document.getElementById("textcolor").value);
localStorage.setItem("fontsize", document.getElementById("fontsize").value);
```

getItem() method is used to fetch values from local storage. Below is the code to retrieve values from localStorage.

```
document.getElementById("page").style.backgroundColor =
localStorage.getItem("bgcolor");
document.getElementById("page").style.color = localStorage.getItem("textcolor");
document.getElementById("page").style.fontSize = localStorage.getItem("fontsize");
```

Users can view the storage data in the browser by pressing F12 as shown below:



Similarly, session storage holds the data until the session ends or the browser/tab is closed.

An attacker can inject some malicious code and can steal the data stored here. So we should always ensure that sensitive information is not stored at the client side.

Let us now understand how to mitigate local storage vulnerability.

Mitigation Techniques:

1. Never store sensitive information on client-side

2. Use cookies with the 'httponly' flag to protect the data stored at the client-side

Cookies

A cookie is a piece of data that is stored in the user's browser. It can store a maximum of 4 KB of data in the browser. Data in cookies are stored as key-value pairs.

Example of a cookie:

```
userName=Smith; expires=Mon, 15 Oct 2018 23:00:00 GMT;
key: userName
value: Smith
expires: Mon, 15 Oct 2018 23:00:00 GMT;
path: \
```

To create a cookie we can use document.cookie property.

```
1. document.cookie = cookieValue;
```

cookieValue is a string in key-value pairs. For example, a cookie to store username and password can be created as below

```
1. document.cookie = "username = Smith;password = secret";
```

In the above code snippet username and password are keys, Smith and secret are their corresponding values.

Below is the syntax of setting a cookie in the HTTP response header:

```
1. Set-Cookie: <name>=<value>[;<Max-Age>=<age>][;expires=<date>][;domain=<domain_name>]
   [;path=<some_path>][;secure][;HttpOnly]
```

Now let us see a demo on how to inject malicious code to steal local storage data and its mitigation.

Reverse Tabnabbing:

Let us now see another type of attack possible in web applications called reverse tabnabbing.

Reverse Tabnabbing is an attack where the linked page can control the parent page and can rewrite it without the notice of the original user. The changed page looks similar to the original one. This may lead to one form of phishing attack where an attacker can steal authentication data or any sensitive information from the user. This attack is mostly possible if a user is in an unsecured network like a public wifi hotspot. It is also possible if the user URL has HTTPS, an attacker has to spoof only the site that is being linked to.

Let us understand this attack by an example:

Consider a message forum or a blog where an attacker can post his own website link. If any user visits that link will be shown some information but in the background that malicious website will redirect the parent login page to a fake page that looks similar to the original login page. When a user comes back to the message forum, they appear to be logged out. Without thinking they will enter their credentials to log in as the page looks similar to the original one. Now the attacker can get hold of that authentication data. Now the user will be redirected to the message forum page automatically so that they won't get a doubt that they have entered credentials in a fake login page.

This attack is possible in two scenarios:

1. If a website is using a hyperlink with a target attribute in which if the user clicks that link the new page will be loaded in a new window/tab keeping the current page opened in the original tab.

```
1. <a href="example.html" target="_blank">Click here to win a lottery</a>
```

2. If a website is using a window.open javascript function

```
1. <button onclick="window.open('example.html')">Click here to win a lottery</button>
```

Following is the malicious script which will run behind the scenes when a user clicks the above hyperlink or button:

```
1. <script>
2.     if (window.opener) {
3.         window.opener.location = "../attack.html";
4.     }
5. </script>
```

As we have understood the reverse tabnabbing attack, let us understand the mitigation technique

1. If a hyperlink is used with a target attribute, use the below attribute

```
1. <a href="example.com" target="_blank" rel="noopener noreferrer">Click here to win a
   lottery</a>
```

noopener option informs the browser not to make the opener object available to the linked website. As this option is not supported by a few older browsers, we should also include the noreferrer option as well. The noreferrer option tells the browser not to make any referrer information available to the linked website.

2. If we don't want to set opener and referrer options for every hyperlink, we can set this as an HTTP header in all responses.

```
1. Referrer-Policy: no-referrer
```

3. If the window.open function is used, use 'noopener, noreferrer' in the third parameter(windowfeatures) of it.

```
function openWindow(url, name, options){
  // Set opener and referrer options
  var newWindow = window.open(null, name, 'noopener,noreferrer,'+options);
  //Reset the opener link
  newWindow.opener = null;
  // Load the correct url
  newWindow.location = url;
}
```

As you observe, we need to set the URL only after clearing the opener to prevent the attack. If we use the URL in the window.open function directly, it will be opened in a new tab/window which will open space for attacks.

HTML Security Checklist:

Every web developer will be using HTML for creating web pages.

Let us understand the security best practices that the developer should follow during HTML coding to create a secure web application.

- **Do not just rely on client-side validation**
  - HTML5 provide features to perform client-side validation. But do not just rely on the client-side validation, ensure that validation is performed on the server-side as well for sensitive actions like authentication.
- **Sanitize user inputs**
  - Any input from the user is to be assumed as untrusted. Proper sanitization of input data provides protection from different kinds of attacks like HTML Injection, etc.
- **Use the attributes "noopener" and "noreferrer" whenever a link is provided to open in a new tab.**
  - Mysite
    - The "noopener" attribute instructs the browser not to grant the browsing context access to the document that opened its new link.
    - The "noreferrer" attribute prevents the browser from sending the page address, or any other value, as the referrer when navigating to another page.
- **Use the sandbox attribute of an iframe**
  - Make sure to add the "sandbox" attribute to the iFrame feature of HTML5 as it provides additional restrictions to the content.
- **Eliminate comments from the final code**
  - Use comments only for documentation purposes. Avoid the usage of comments to store sensitive information in the HTML as it is entirely visible. Any unwanted comment is to be removed before the production build. Modern browsers can inspect the commented code and sensitive information can be retrieved.

- **Refer to the latest security news**
  - Keep checking the latest news in the security area as the internet is growing and new security features and techniques may get implemented every day.
- **Use a Content Security Policy (CSP)**
  - With the aim of protecting the application from potential dangers, use a Content Security Policy to delimit the assets that are permissible to be added to the application
  - To ensure including the media in a secure way,
    - use the CSP directive **img-src** to load images from valid sources
    - use the CSP directive **media-src** to load audio and video from valid sources
- **Use web storage judiciously.**
  - Avoid storing sensitive data in the web storage since it is stored in the browser.
- **Use safe methods.**
  - Know about safe methods in a language/technology and use them for secure coding. For example, using innerText or textContent property is safe instead of innerHTML in JavaScript.
- **Implement CORS Policy**
  - Implement CORS policy to allow access to the application services.

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.

```
<!DOCTYPE html>
<html>
   <head>
      <title>Cross Browser</title>
   </head>
   <body>
      <form>
         <label for="">Username</label>
         <input type="text" name="username"> <br/>
         <label for="">Password</label>
         <input type="password" name="password"><br/>
         <input type="submit">
      </form>
   </body>
</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.

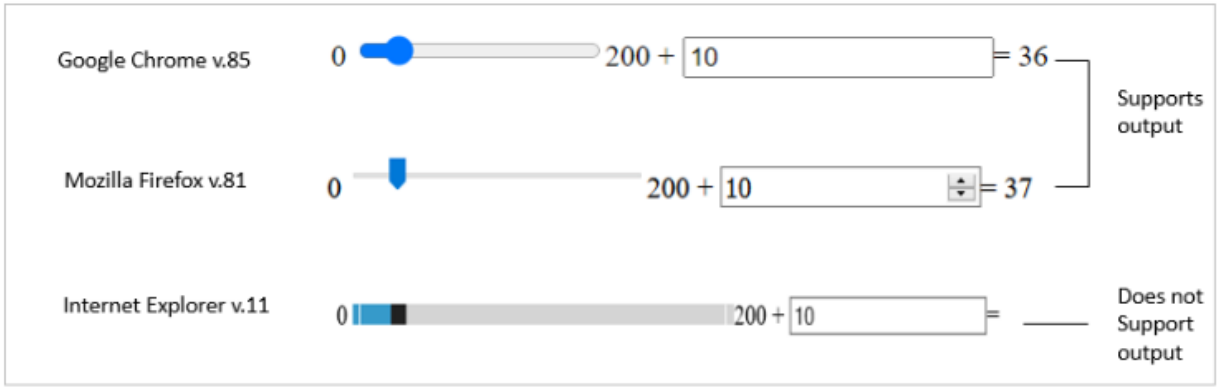
Also, all browsers may not support all features of HTML5.

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.

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Cross Browser</title>
    </head>
    <body>
        <form oninput="x.value= parseInt(a.value)+parseInt(b.value)">
            0 <input type="range" id="a" value="10" min="1" max="200">200  +
            <input type="number" id="b" value="10">=
            <output name="x" for="a b"></output>
        </form>
    </body>
</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.



Browser Compatibleness:

- To identify the capabilities of your browser you can refer www.caniuse.com website.
- To identify whether your browser supports a particular HTML5 element or not, you can refer www.html5test.com website

Best Practices For HTML Web Pages:

The following are some best practices to be followed while developing a web page or an application using HTML5.

1. In the application, ensure that each web page is having proper document structure with divisions such as <head>, <body>, <header>, <footer>.
2. Ensure that correct DOCTYPE is declared, which is required for the browser to understand the standards of the application used to render it properly.
3. Use a meaningful title for the web pages.
4. Always close the tag even if it is not a container tag.
5. Avoid Inline styles and embedded scripts. Try to include script and style with external files keeping your HTML code clean and neat.
6. Include the script as an external file and place this code at the end portion of the <body> tag.
7. Add the lang attribute to the root element, and keep the value as short as possible.
8. Even though HTML is case insensitive, using lowercase for code is the best practice.
9. Use the most appropriate HTML tag for sectioning the web page. Try to avoid the generic tag and use more meaningful tags such as <article>, <section> and appropriate tags in the web pages.
10. Always use <ul> for designing navigation bars.
11. Avoid the use of the 'type' attribute in the HTML list element and use the corresponding CSS property list-style-type property instead.
12. On embedding elements to the web page, use the alt attribute with a proper description text.
13. Indent nested elements properly.
14. Use appropriate type attribute for Input element inside forms, also ensure the form control is wrapped with <label> element.
15. Use a sandbox attribute along with the iframe element whenever it is needed to embed another webpage into the application.
16. Use rel attribute with a value "noopener noreferrer" along with hyperlink, to avoid reverse tabnabbing.
17. Consider all the input data as untrusted information and sanitize those inputs before it is used in some logic in the application.
18. Do proper output encoding to avoid chances of injection of malicious code to the application through input elements.