

UNIT-1

HTML 5

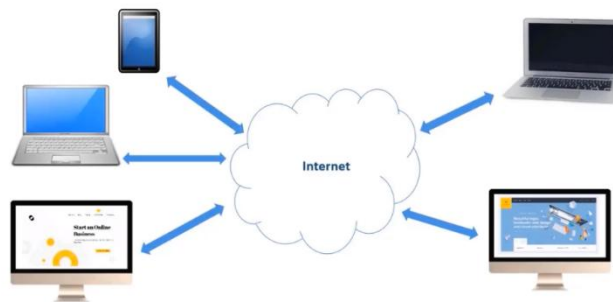
Introduction to Web:

Billions of people use web for everyday activities such as checking out the latest news, ordering food online, online shopping, and conducting meeting etc. We are able to access various applications on web to different devices; it is because of World Wide Web (WWW).

WWW is a collection of web documents and resources, which can be accessible over internet. WWW is not internet, even though we use the terms often interchangeably, they are different.

Then, what is internet?

An internet is a network of devices, connected to each other in order to share information such as email, file transfer and conferencing etc.



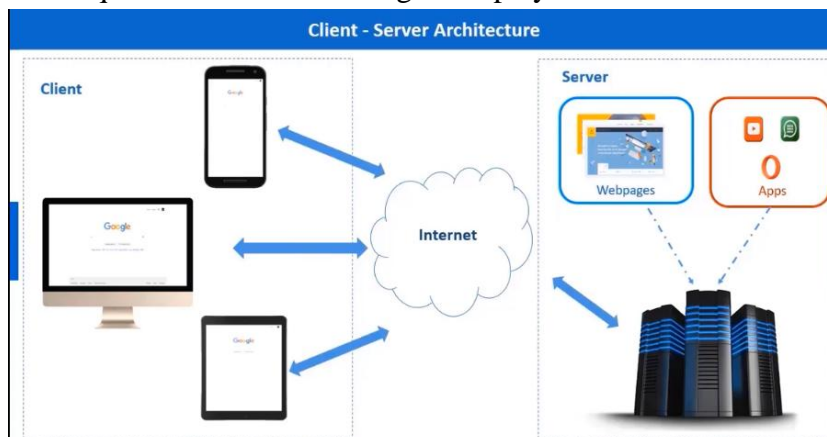
What is Web?

What is Client-Server Architecture?

Client is the network connected devices, it can be a office system, connected to the network through LAN or it can be a home computer connected through the Wi-Fi or it can be a phone connected through the mobile network. **Clients** receive services from other systems over network.

Servers are the systems or devices that stores websites or web applications to provide the services to the requested clients over network.

When a client device wants to access a web page, client sends a request and the respective server responds to the request with required resources which gets displayed on client's web browser.



Browsers are called as the translator for the web. It interprets the needed information in the form of tags on a web page and translates the information to the end users.

Some of the popular browsers are: Google Chrome, Microsoft edge (Internet Explorer), Mozilla Firefox and safari.

How web works?

When we type a web application name in the browser, for example, www.example.com , the client finds the actual address of the server connected over the network and request for the service. The server will respond back with the respective resources, which gets loaded on the client's device.

In case of any issue in the client request reaching the respective server, or server is unable to provide service due to some issue, then an appropriate error message details will be send as a response to the client, which gets displayed on the client device.

“WWW is an information system where billions of web pages and web applications are accessed over the internet from the server by the client.”

Web terminologies that every web developer should know:

1. **URL (Uniform Resource Locator):** uniquely identifies the particular resource in the internet. It has the following components:

Protocol://host:port/path

- a. **Protocol:** used for client-server communication over internet.
- b. **Host and the Port:** (Name of the connected server along with the port number) each device connected over internet will be accessible with unique host name, also called as domain name and different services will be provided through specific port number. If the port number is not explicitly specified in the URL, then the default port will be considered. Standard code for HTTP is 80.
- c. **Path** it specifies the path of the resource on the server. Usually starts with the slash (/), sometimes known as web route.
- d. **Example:** http://www.myapp.com/results.

In this case, the protocol is http. Domain name is myapp with default port number. And /results is the path of resource to be accessed.

2. **HTTP (Hypertext Transfer Protocol):** A set of rules for transferring information such as text, graphics, images, sounds, videos and other multimedia files on the WWW.

The client and server can communicate with each other with an application level protocol called HTTP. The HTTP is a **request/response based protocol**, because the client establishes a connection with the server and sends a request. Over the same connection, the server sends the response. Once the response is delivered, the connection is terminated.

And if the client wants to send the request again, a new connection will be established, the client will send the request and the server sends the response. Hence, HTTP is also called **stateless protocol**. Because each request is executed independently without any knowledge of the previous request.

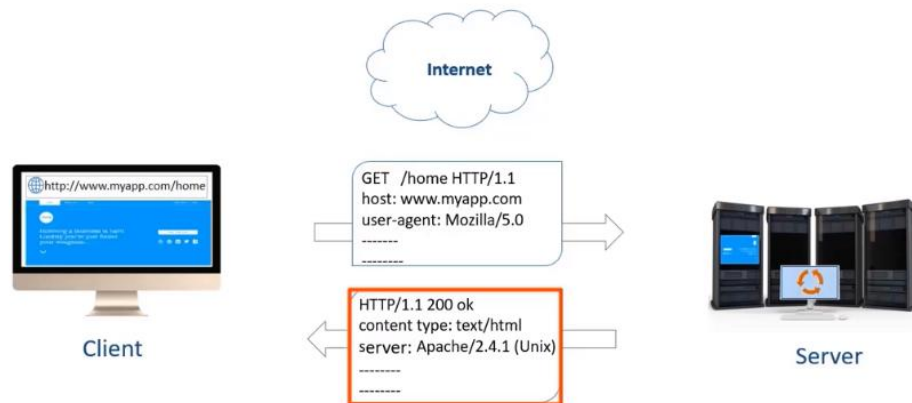


HTTP Request Response Cycle:

Whenever we type URL in the browsers address bar to get a web resource over the internet using the HTTP protocol. For ex, if we type `http://www.myapp.com/home`.

The browser sends a request with some header details such as **method type, resource path details, HTTP version, host and the user details** etc to the server.

The HTTP server interprets the request and returns an appropriate response containing header details such as **HTTP version, status code, content type, server details etc. along with the actual content** in the body which is either the resource, the client has requested or an appropriate error message.



3. **MIME (Multi Purpose Internet Mail Extension):** is a standardized way to indicate the format of the resource content from server to the client. Server sends the content type details in the HTTP response header along with the actual content to the client. So, the client can interpret and display the content appropriately. Some of the content types are:

- i. **Text: (theoretically any human readable text)**
 - a. Text/plain
 - b. Text/html
 - c. Text/css
 - d. Text/javascript
- ii. **Image, represents any animated image**
 - a. Image/png
 - b. Image/jpeg
 - c. Image/gif
- iii. **Video, represents any kind of videos**
 - a. Video/mp4
 - b. Video/x-ms-wmv
 - c. Video/x-flv
 - d. Video/x-msvideo
- iv. **Application, represents different file types**
 - a. Application/json
 - b. Application/sql
 - c. Application/javascript

Overview of Web Technologies:**Overview of core web technologies/ Basic web technologies:**

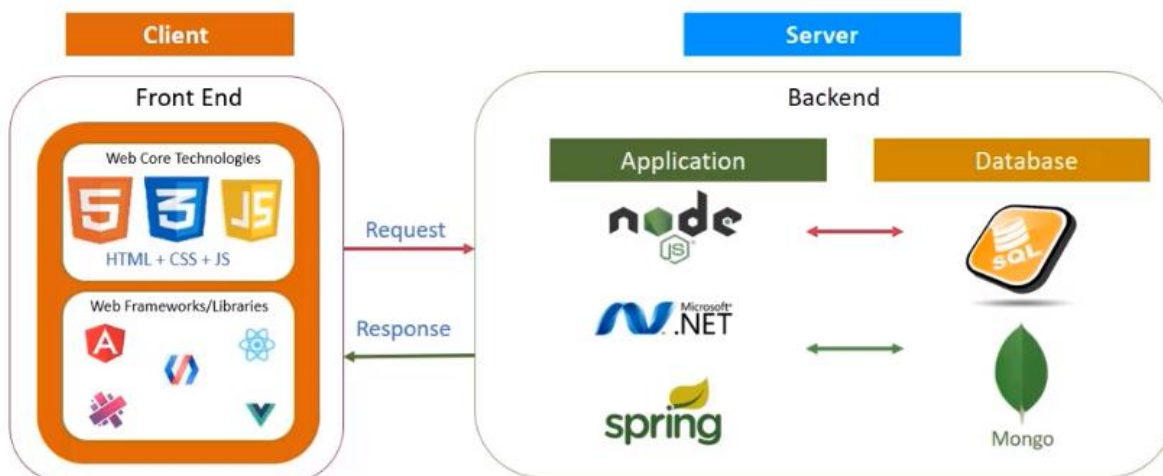
1. **HTML and CSS:** we can relate both the HTML with the invention of the WWW; it is one of the code languages of the web. This is the first technology that every web developer should know the basic structure of the web pages. If we want to make our web pages more impressive and readable to the end users, then we need to use CSS (Cascading Style Sheets), which is used to apply required size and to format the web documents.
2. **Scripting Languages:** these are the languages that are needed to communicate with the browsers and to make the web applications more dynamic and interactive in nature. Some of the popular scripting languages are:



3. **Web frameworks/ Libraries:** are essential to reduce the developers at work. They come with their own methods and packages. Develop web application is much simpler, faster and efficient way. Some of the popular web frameworks/libraries are:

**Full stack web application development – A big picture**

We can develop a full stack application with a front end and backend. This architecture gives a big picture on client and server technology. We can observe the popular front end and back end technologies used to develop a full stack technology.



Knowing the core web technologies, HTML, CSS and JS, you will be able to create our own web applications using the web frameworks/libraries.

We visit many websites every day for business, entertainment, education, etc. A website plays an important role in any company's business expansion and its reachability to customers.

Have you ever been inquisitive about how these websites work, what skills are necessary to build a website?

HTML, CSS, and JavaScript that are used for the creation of web applications. HTML, which is the language used for web page creation.

Below are the common editors/IDE's used to code the web pages:

- Notepad
- Notepad++
- Sublime Text
- Brackets
- Atom
- Eclipse IDE
- Visual Studio Code IDE (VS Code IDE)

Any of the below browsers can be chosen to view the web pages:

- Google Chrome
- Mozilla Firefox
- Internet Explorer

Need for HTML

In the web technology world, below are the fundamental building blocks of any web page.



HTML (Hyper Text Markup Language) is the most fundamental building block of the Web. It defines the meaning and structure of web content.

CSS (Cascading Style Sheet) is used for styling and giving better presentation to the web pages.

JavaScript (JS) is used for implementing behavior required in a web page.

Environmental Setup for HTML5:

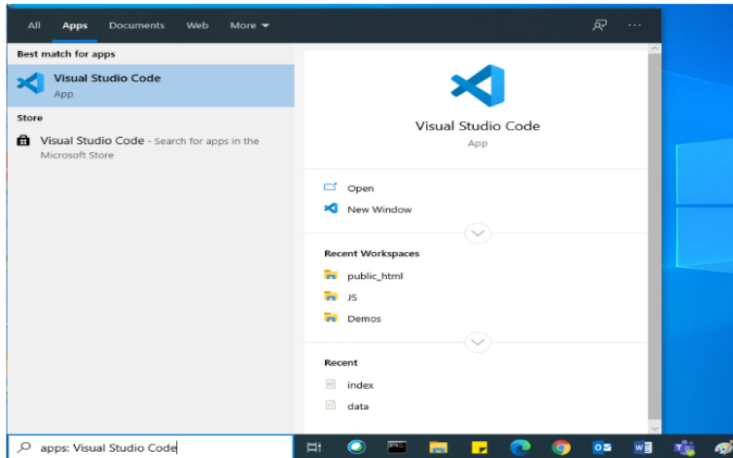
To develop a web application using HTML, you can use simple editors such as Notepad or go for IDEs like Visual Studio Code (recommended), Eclipse etc. which makes coding easier.

And, to execute application, you can use any commonly used browser such as Google Chrome (recommended), Mozilla Firefox etc.

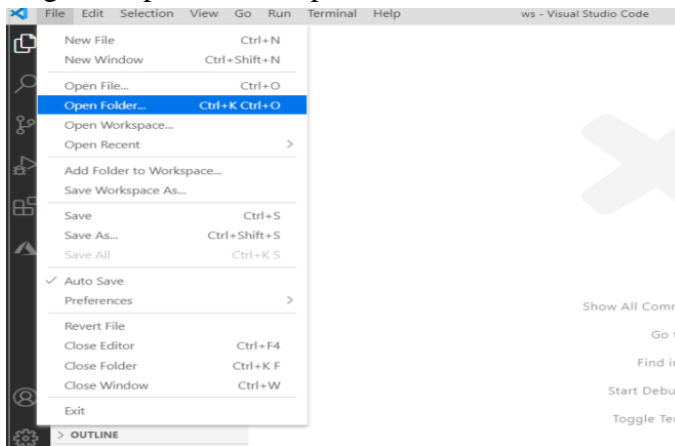
Let us understand how to set up the environment to develop a web application using Visual Studio Code.

Create the below-mentioned HTML page in Visual Studio Code by the following steps:

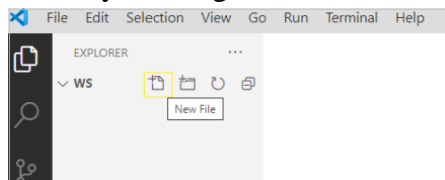
- Open Visual Studio Code from the Start menu.



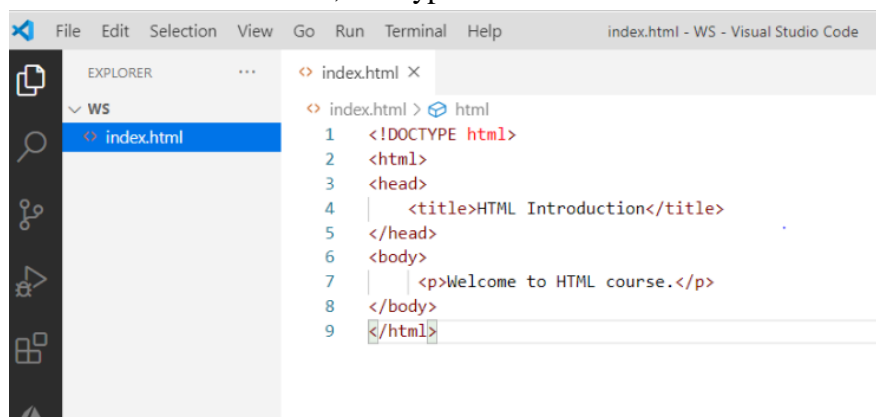
- Once Visual Studio Code is launched, open your local workspace by clicking the File menu and selecting the 'Open Folder' option.



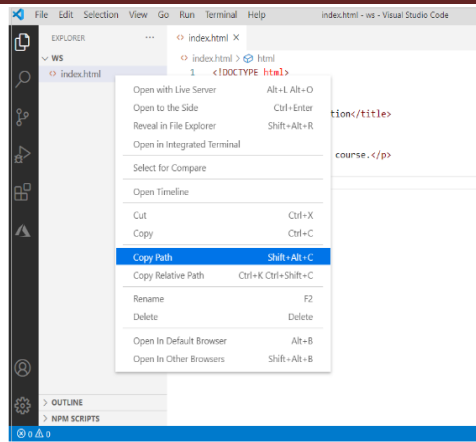
- Create a new file by clicking the new file icon as shown below.



- Name the new file as index.html, and type the below-mentioned code in it.

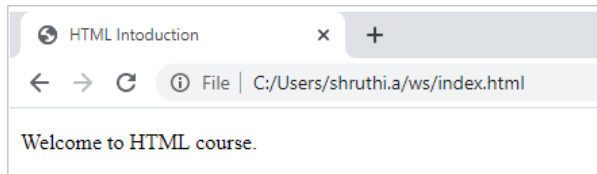


To render this HTML file, right-click on **index.html** as shown below and copy the path of index.html into the browser.



And the output will be displayed as below:

Output:



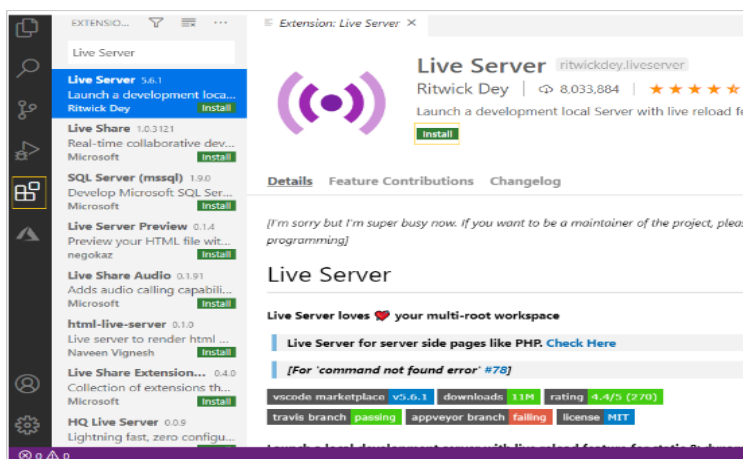
In the previous example, each time whenever we make subtle changes in the code we will be forced to refresh the page for the changes to reflect.

There is a solution to this in the Visual Studio Code.

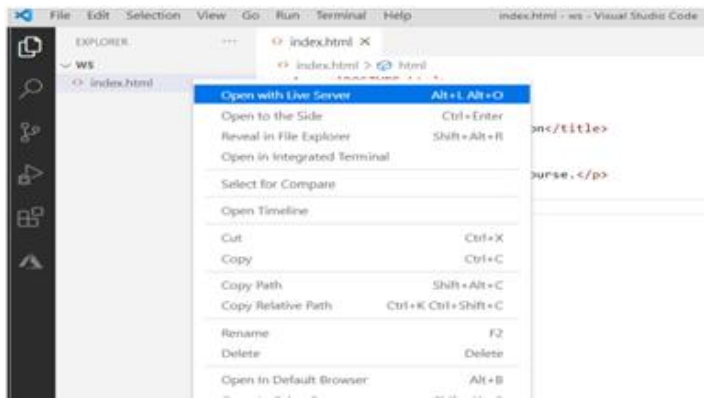
Visual Studio Code IDE provides an extension called **Live Server** using which HTML page can be rendered and any changes that developers make further will be automatically detected and rendered appropriately.

Follow the below steps to add Live Server:

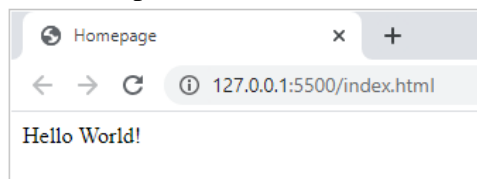
1. Go to the extension tab in Visual Studio Code and search for Live Server and Click on the Install button.



2. Once the Live Server extension is successfully installed, an HTML page can be rendered by right-clicking on the intended HTML page in the Explore tab and select the 'Open with Live Server' option as shown below:



And, you can observe the output as below:

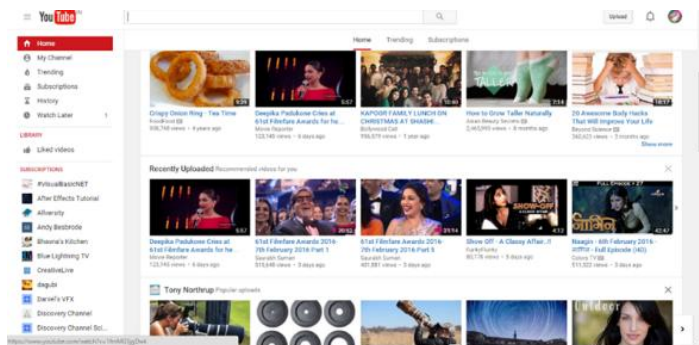


HTML-Need:

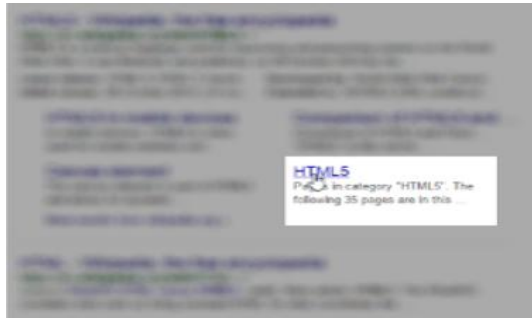
HTML is needed to fill online forms in web applications as below:

A screenshot of a web form titled 'Send us a query' from the website 'WayFar'. The form has a dark red header with navigation links: HOME, POPULAR DESTINATIONS, TOUR PACKAGES, OFFERS, ENQUIRE US, and GALLERY. The form fields include: 'Name' (text input with 'Ex: John Davis'), 'Email Id' (text input with 'Ex: john.davis@example.com'), 'Destination' (text input with 'Ex: India'), 'No. of Travellers' (radio buttons for 'Adult' and 'Children'), 'Phone Number' (text input with 'Ex: 9857979006'), and 'Start & End dates of Trip' (two date pickers). There is also a 'Send notifications' section with checkboxes for 'via Email' and 'via SMS'. A large 'Raise your query' button is at the bottom.

Also, with the help of HTML, we can watch online videos on Youtube.

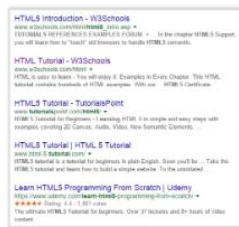


While browsing the Web, when you click on a hyperlink, you navigate to another web page. This is possible with the help of HTML.



Do you know how these web pages with different content types such as text, hyperlinks, forms, images, and videos are created in web applications?

You can achieve all these requirements using HTML.



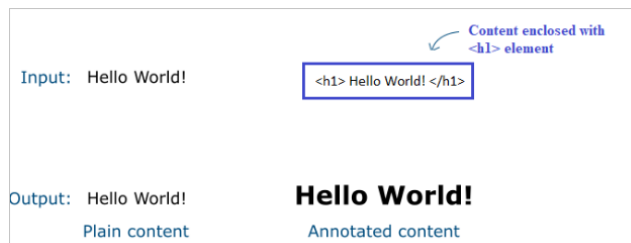
Google reads web pages to generate a result set for the user's search.

People with visual impairments, use screen-readers to access the web application.

What is HTML?

Hyper Text Markup Language (HTML) is a standard markup language to create the structure of a web page. In a web page, all instructions to the browser are given in the form of HTML tags, also known as HTML elements.

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

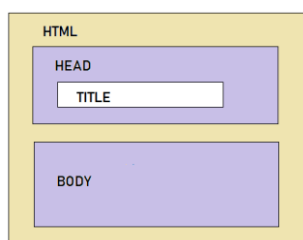


The text content "Hello World!" is annotated to the heading, increasing the font-weight when it is enclosed in the HTML tag `<h1>`.

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

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

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



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

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

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

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

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

The following elements can be provided within the head tag.

Tags	Description
<title>	Defines the title that should be displayed on the browser tab
<meta>	<ul style="list-style-type: none"> • 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.
<style>	Defines style information for the web page
<link>	Defines a link to other documents like CSS
<script>	Defines script like JavaScript

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

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

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

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

Case-Insensitivity:

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

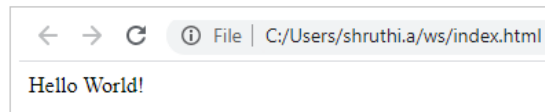
Example 1:

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

Example 2:

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

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

**Best Practice:**

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

Platform Independancy:

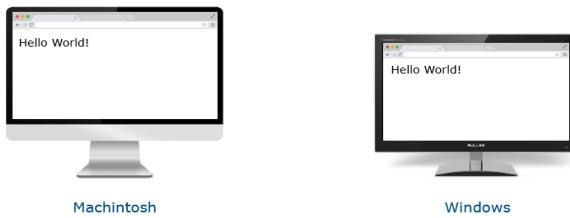
HTML Language is platform-independent. That means the same HTML code can run on different operating systems as shown below.

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:

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

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

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

Syntax: `<!DOCTYPE html>`

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

Sample.html:

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

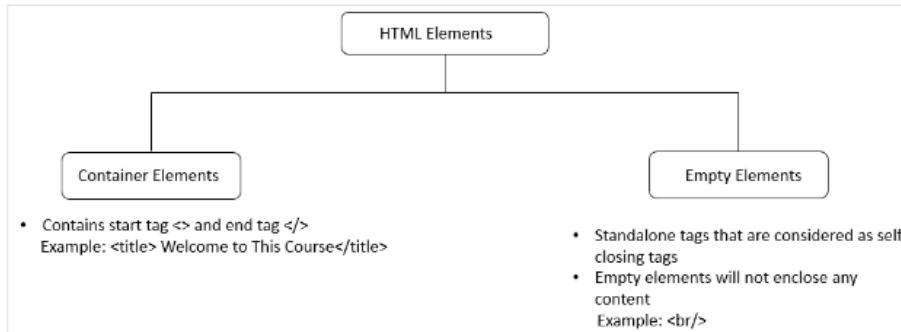
In the above code, `<!DOCTYPE html>` signifies that, the code is written in HTML5.

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:

1. **Container/ Empty Elements:** HTML elements can be broadly categorized into two as below:



Let us see the further classification of HTML elements.

Best Practice:

It is a good practice to use the closing tag for the respective start tag without fail.

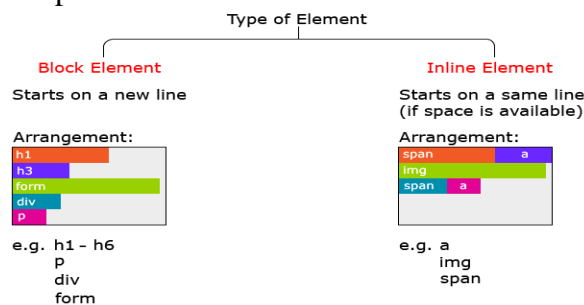
2. Block/ Inline Elements: HTML elements can be further categorized into two as below:

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

All the attributes can contain properties like name and value which can be used by a developer to assign respective details for that HTML elements.

- Attributes are to be set only in the start tag of a container HTML element.
- Attributes are case-insensitive, but it is recommended to use lowercase as a best practice.
- The best practice is always to quote attribute value even though we will not get any execution errors if they are not provided in quotes.

Example:

The lang attribute specifies the language of the content of the HTML page.

Syntax: `<html lang="en-US">`
 ↑
 Specifies that the content of .html page is written in U.S. version of English language

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.

Comment:

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

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.

Meta Data Element:

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.

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: The metadata element is defined within the head element.

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

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

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

For example,

```
<!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.

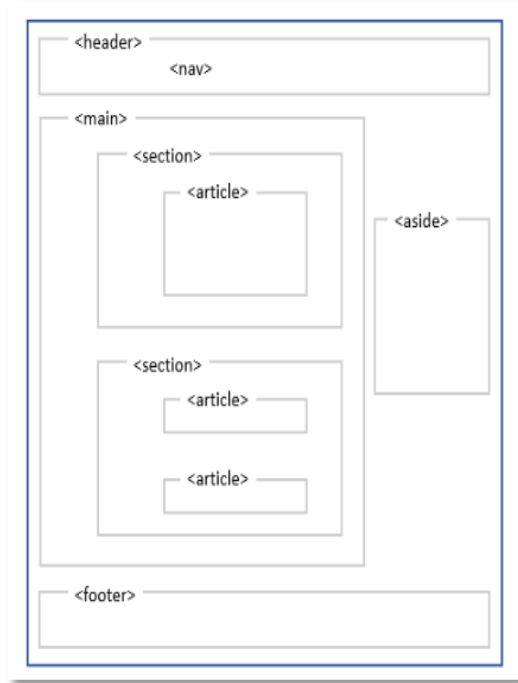
Sectioning Elements:

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:

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

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



Below are some of the semantic tags in HTML:

<header>

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

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

<footer>

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

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

<main>

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

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

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

<nav>

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

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

<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>
```

<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>
```

<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>
```

<address>:

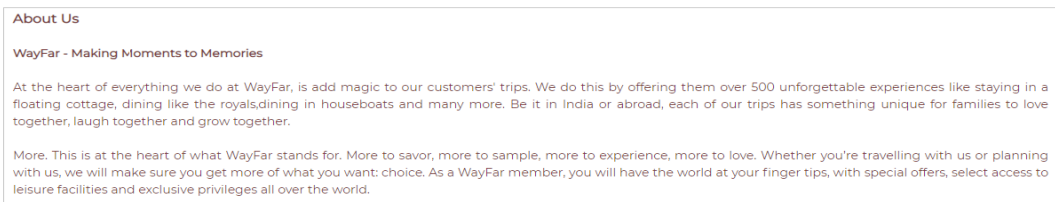
The <address> element helps to semantically organize address details in HTML content.

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

Grouping Elements- Need

While developing any web page there may be needed for us to develop some pages like About Us, Terms and Conditions, News report, etc. which would contain a lot of textual information, in that situation we might commonly come across the below-mentioned scenarios:

1. The need to associate mandatory visual spacing between logical portions of the textual description so as to improve text readability. Have a look at the below About Us page, for example.



The page provides a basic description of what the website essentially does and also details what it especially offers to its members. Logically, these two descriptions are separate and the same need to be conveyed visually to the user as well so that the readability of webpage content is achieved.

2. The need to highlight the topic headings. For example: In the below Terms and Conditions page,



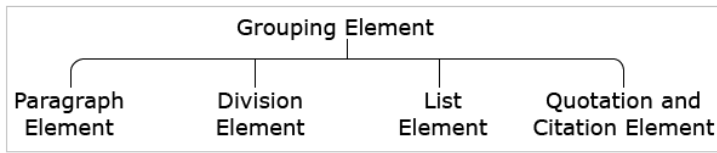
The topic headers like “Limited License”, “Terms Policy”, etc are highlighted so that the user can quickly differentiate the respective topic description on the webpage.

3. The need to group various portions of a web page for various reasons like applying some common styling. For example, in the below screen, descriptions related to Limited License, Copyright, Terms Policy needs to be grouped for applying a common style of black color text.



This is why HTML introduced grouping elements.

Grouping elements in HTML5 can be categorized majorly as below:



Let us see each of them in detail.

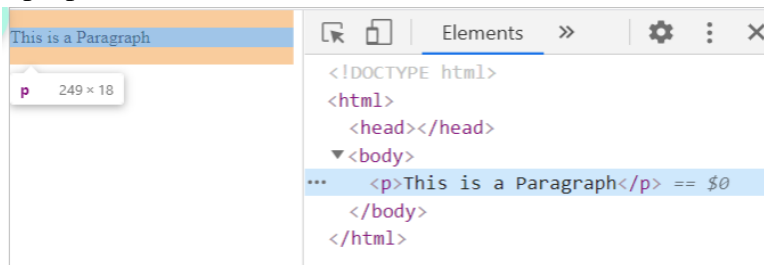
Paragraph Element:

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

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

demo.html

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



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

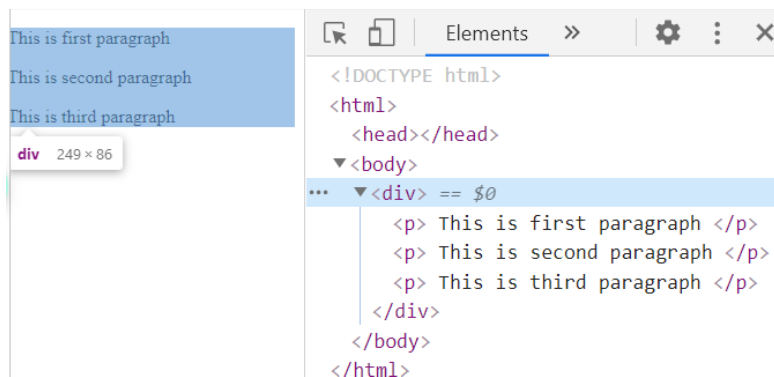
Division and Span Elements:

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

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

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

demo.html



To execute this code:

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

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

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

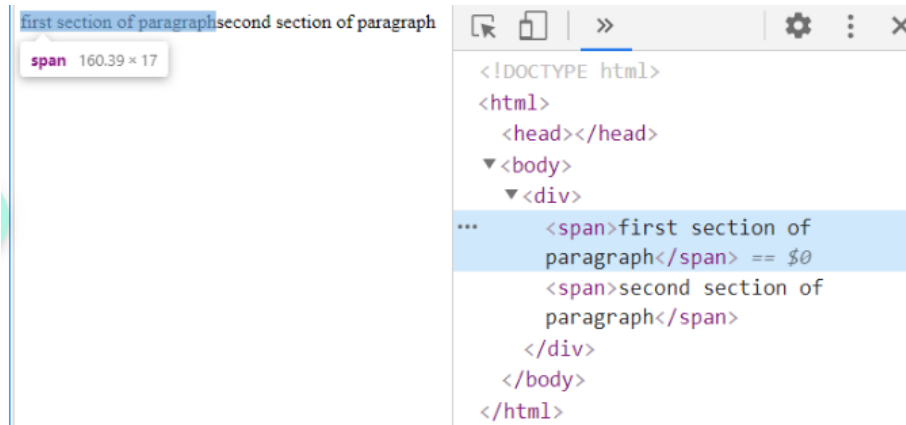
Similar to the division element, the span element is also used to group various other HTML tags to apply some common styles.

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

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

demo.html

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



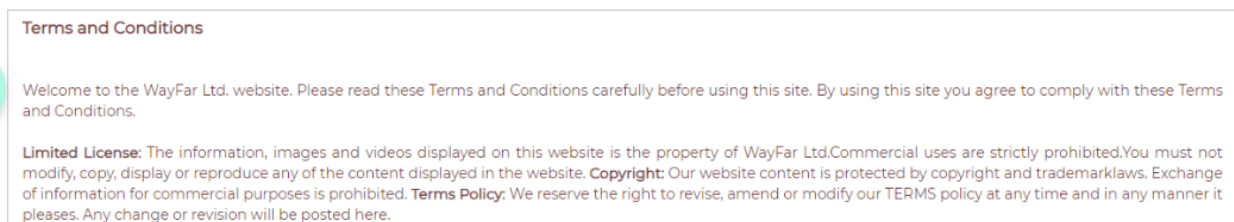
To execute this code:

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

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

List Element:

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.



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:

Terms and Conditions

Welcome to the WayFar Ltd. website. Please read these Terms and Conditions carefully before using this site. By using this site you agree to comply with these Terms and Conditions.

- **Limited License**

The information, images and videos displayed on this website is the property of WayFar Ltd. Commercial uses are strictly prohibited. You must not modify, copy, display or reproduce any of the content displayed in the website.

- **Copyright**

Our website content is protected by copyright and trademark laws. Exchange of information for commercial purposes is prohibited.

- **Terms Policy**

We reserve the right to revise, amend or modify our TERMS policy at any time and in any manner it pleases. Any change or revision will be posted here.

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

Awards

WayFar is rated India's Most Popular Resort Chain for the year 2020 WayFar is voted India's Favourite Resort Chain for the year 2018. Bestowed with the prestigious Silver Peacock Award for Sustainability 2016 Awarded the Good Corporate Citizen Award from the Calcutta Chambers of Commerce & Industry in Economic Development category

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.

Awards

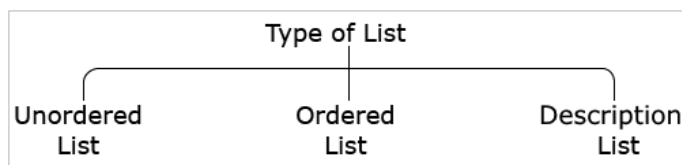
1. WayFar is rated India's Most Popular Resort Chain for the year 2020
2. WayFar is voted India's Favourite Resort Chain for the year 2018.
3. Bestowed with the prestigious Silver Peacock Award for Sustainability 2016
4. Awarded the Good Corporate Citizen Award from the Calcutta Chambers of Commerce & Industry in Economic Development category

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.

Types of lists:

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

**Unordered List:**

HTML lists come in 3 basic flavors:

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

Each one has a specific purpose and meaning.

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

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

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

Courses offered:

- HTML5
- CSS
- JS
- Bootstrap

The following snippet can be used.

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

Unordered Lists- Bullet Types:

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

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

Courses offered:

- HTML5
- CSS
- JS
- Bootstrap

The corresponding code to achieve this requirement is:

```
<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:

Courses Offered:

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

Corresponding HTML code to achieve this requirement is:

```
<h1>Courses Offered:</h1>
<ul>
  <li>Markup
    <ul>
      <li> Basics of HTML
        <ul>
          <li> First level course on HTML </li>
        </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 ... elements on various lines, the default bullet styling of unordered list element has been populated on to the web page.

Ordered List:

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

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

Courses offered:

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

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

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

Let us learn how to customize the ordered list appearance.

Best Practice:

It is a good practice to avoid more than one list item per line.

Ordered List – Bullet Types:

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

Courses offered:

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

The corresponding code to achieve this requirement is:

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

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

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

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

Ordered Lists – Attributes:

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

Name	Description
Start	Specifies the initial value of the list.

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

Consider the following code snippet:

```
<h1>Courses offered:</h1>
<ol type="a" start="d" reversed>
  <li>HTML5</li>
  <li>CSS</li>
  <li>JS</li>
  <li>Bootstrap</li>
</ol>
```

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

Courses offered:

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

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

Best Practice:

It is recommended to use list-style-type CSS property instead of the ... type attribute unless the pattern of numbering ordering of the list is considered to be very important.

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:

Courses Offered:

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

The corresponding code to achieve this requirement is:

```
<h1>Courses Offered:</h1>
<ol>
  <li>Markup
    <ol>
      <li> Basics of HTML
        <ul>
```

```

        <li> First level course on HTML </li>
      </ul>
    </li>
    <li> Adaptive HTML </li>
  </ol>
</li>
<li>Styling
<ol>
  <li> CSS3
  <ul>
    <li> Latest version of CSS </li>
  </ul>
</li>
</ol>
</li>
</ol>

```

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

Description List:

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

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

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

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

The Lord of The Rings
An epic high fantasy book by the English author and scholar J.R.R.Tolkien

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

```

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

```

Link Element:

When developing any website:

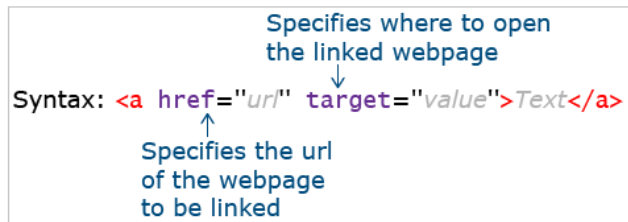
- Each web page is necessarily coded in individual HTML files.
- To see a particular web page, the respective HTML file has to be opened up in a browser.

Below are 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.

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:

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

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">
  
</a>
```

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

Bookmark hyperlink:

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

```
<h2 id="top">Topic</h2>
<p>Detail.....</p>
<p>Detail.....</p>
<p>Detail.....</p>
<a href="#top">Go to Top</a>
```

Email hyperlink:

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

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.

Contact number hyperlink:

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

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:

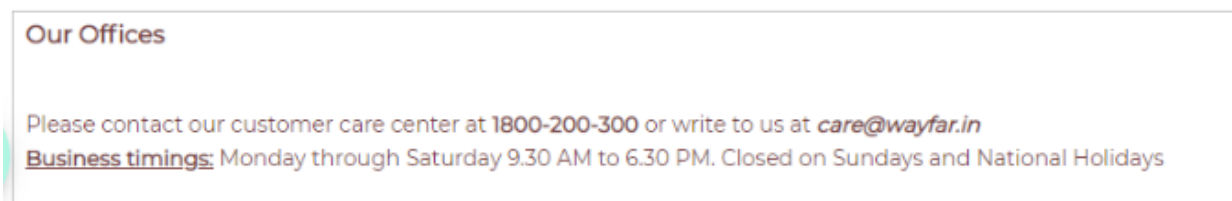
```
<a href="https://owasp.org/" target="_blank">Link to OWASP web site</a>
```

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

Text - level Semantic Elements - Need

When displaying any content, specifically some kind of text/numerals on web pages, we might need to format certain portions of the content.

Consider the below screenshot:



- The phone number is the quickest connection as it invites instant response and hence the number is in bold.

- The email should be clearly readable and distinguishable and hence shown in italics.
- Business Timings details need to be considered as very important and it should be noted by the user to know when the company team would be available to reachable to solve their queries. This field can be underlined.

Adding a bit of formatting to the text can help in adding appropriate emphasis and makes it easier for the user to figure the important information on the screen very quickly.

Widely Used Text-Level Semantic Elements

The table below lists widely used text-level semantic elements supported in HTML5.

Element	Description
Abbr	Defines abbreviation or acronym
Q	Represents text quoted from another source by adding quotation mark (" ")
Small	Displays text in relatively smaller font-size
Mark	Highlights text
Strong	Displays text in bold
Em	Displays text in the italic or emphasized format
Sub	Displays text as subscript
Sup	Displays text as superscript
Span	Provides styling to text
Br	Breaks line of text

Let us see some widely used semantic elements in HTML.

Widely Used Text-Level Semantic Elements

Below are some text semantic tags and their usage:

The <abbr> element: defines an abbreviation or an acronym

```
<h1>The abbr element</h1>
```

```
<p>The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.</p>
```

The abbr element

The WHO was founded in 1948.

The element:

- Make the text appear bolder

```
<strong>Text in bold</strong>
```

Text in bold

The <mark> element:

- Highlighted text like the way we do in a paper using a highlighter

```
HTML <mark>Marked</mark> Formatting
```

HTML **Marked** Formatting

The <sup> element:

- Display text like superscript

Example of ^{superscripted} text.

Example of ^{superscripted} text.

The <sub> element:

- Display text like subscript

Example of _{subscripted} text.

Example of _{subscripted} text.

The element:

- Emphasize text

Text emphasized

Text emphasized

The <small> element:

- Display text in relatively smaller font

<h2>Text <small>Small</small> in HTML</h2>

Text Small in HTML

The element:

- Strikethrough text

This is deleted content.

This is ~~deleted~~ content.

Note: The <sup> and <sub> elements can be replaced by the CSS property "vertical-align" especially while designing a logo of a product or a business that uses a raised baseline.

Character Entities:

Some characters are reserved in HTML.

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

Also, some characters are unavailable on the keyboard.

For example: ©

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

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

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

Character	Description	Entity Name	Entity Number
	Non-breaking space	 	
<	Less than	<	<
>	Greater than	>	>

Character	Description	Entity Name	Entity Number
&	Ampersand	&	&
©	Copyright	©	©
€	Euro	€	€
£	Pound	£	£
®	Registered trademark	®	®

HTML 5 – 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:

```
<div>
  <p contenteditable="true">This is editable</p>
  <p dir="rtl">The direction of the text is from right to left</p>
  <p title="mydemo">Hover your mouse here to see the title</p>
  <p id="id1">ID should be unique for each element</p>
</div>
```

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.

Creating 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.

Organizing data as Tables:

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

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

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.
- 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

Element	Description
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:

Table heading	
Column 1 heading	Column 2 Heading
Column 1 data	Column 2 data

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.

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

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

```

Table Elements: Colspan / Rowspan Attributes:

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

The `colspan` attribute accepts a numeric value and merges specified numeric value of columns together whereas, the `rowspan` attribute accepts a numeric value and merges specified numeric value of rows together. For example, if we provide `colspan` attribute with a value 2, then 2 columns of the table will be merged as shown below:

	C1	C2	C3	C4
R1	A		B	C
R2				
R3				
R4				

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

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

- **Cellpadding:** Cellpadding specifies the space between the border of a table cell and its contents (i.e) it defines the whitespace between the cell edge and the content of the cell.

Syntax: `<table cellpadding="value" >.....</table>`

- **Cellspacing:** Cellspacing specifies the space between cells (i.e) it defines the whitespace between the edges of the adjacent cells.

Syntax: `<table cellspacing="value" >.....</table>`

Example:

```
<html>
  <table border=2 cellpadding=5 cellspacing=4>
    <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>
```

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

Form Elements:

Need:

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

1. Provide users' information to create his/her account with a website
2. Provide a username and password to login into the account.
3. 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:

Creating Form Elements:

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

The `<form>` tag has the below attributes:

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

Used for accessing
form data by the
scripting language

↓

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

↓

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

↑

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

Form Input elements:

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

Specifies type
of element

↓

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

↑

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

The table below lists the various types of input elements.

The possible value of "type"	Description
Text	Creates textbox
Password	Creates textbox that accepts the only password
Checkbox	Creates checkbox
Radio	Creates a radio button
Button	Creates button
Submit	Creates a button that submits values of all form elements to the server
Reset	Creates a button that resets values of all form elements to their default value
Image	Creates a graphical version of a button

The possible value of "type"	Description
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

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.

Name: <input type="text" value="">

Name :

Input type - password:

An input field can be used to enter a password.

Password: <input type="password">

Password :

Input type - email:

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

Email-Id: <input type="email">

Email-Id :


Input type - number:

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

Age: <input type="number">

Age :

Input type - checkbox:

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

Hobbies: ☒ Reading
☒ Singing
☐ Dancing

Hobbies: ☒ Reading ☒ Singing ☐ 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.

Gender: ☒ Male
☐ Female

Gender : ☐ Male ☒ Female

Input type - file:

Creates a control to upload a file to the server.

Select a file:

Select a file : No file chosen

Input Types - Button and URL

<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)

Raise your query

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.

Enter the website URL:

Enter the website URL :


Input Types - Textarea and Hidden

<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.

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.

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

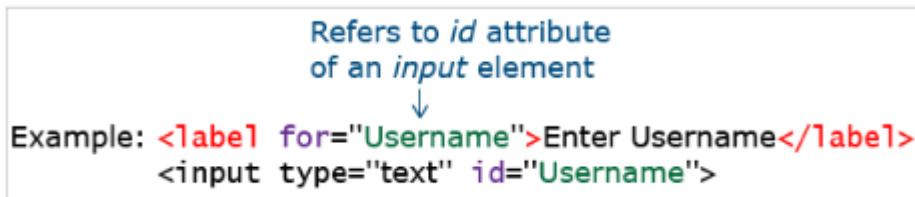
Note: Output may vary for some input fields according to the browser

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.



Color Picker and Date Picker:

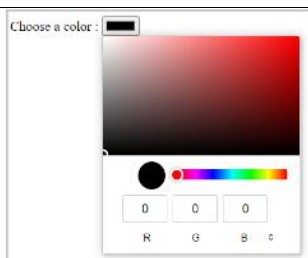
We have various picker-elements in HTML forms such as color-picker and date-picker elements.

Let us see them in detail.

Input type - color:

Defines a color picker.

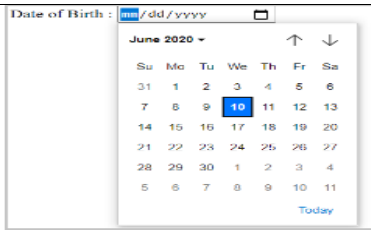
Choose a color: `<input type="color">`



Input type - date:

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

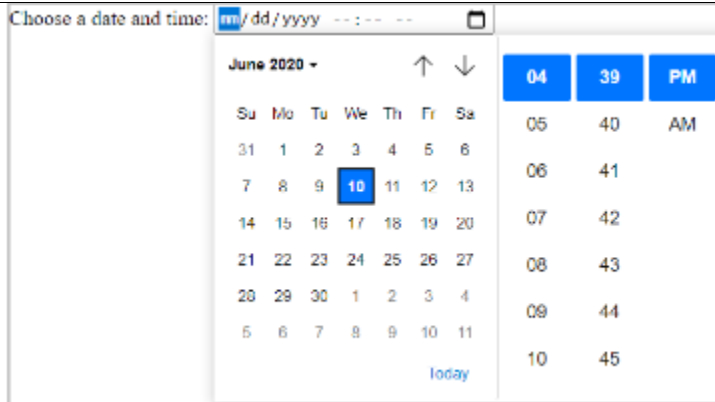
Date of Birth:



Input type – datetime-local:

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

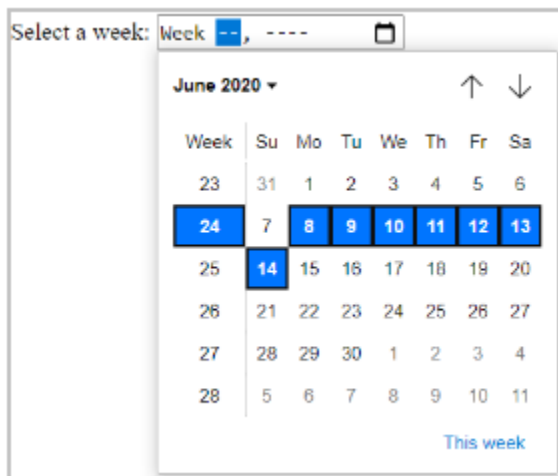
Choose a date and time:



Input type – week:

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

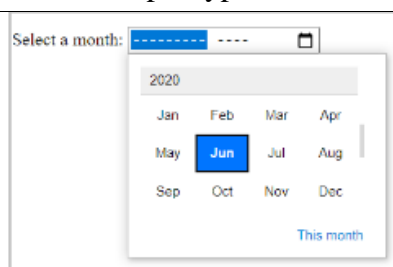
Select a week:



Input type – month:

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

Select a month:



Select and Datalist Elements:

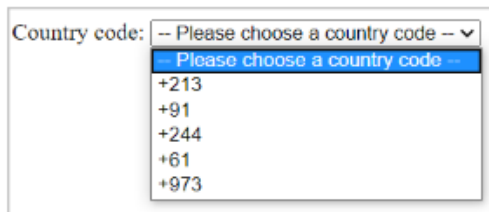
We have `<select>` and `<datalist>` elements in HTML which helps to collect input data from user as a drop-downs.

Let us see them in detail:

`<select>` element :

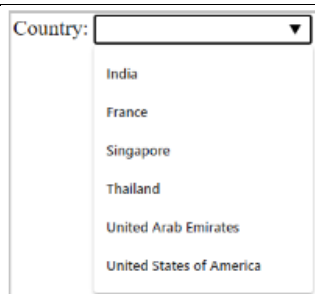
- Defines a drop-down list.
- The "multiple" attribute can be used for having a multi-select dropdown menu.

```
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:

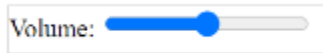
Range, Meter, Progress

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.

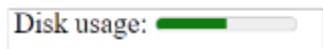
Volume: <input type="range"/>



Meter:

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

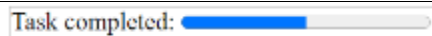
Disk usage: <meter min="0" max="100" value="50"></meter>



Progress:

Can be used to represent the progress of a task.

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 Element:

Displays the output of user input.

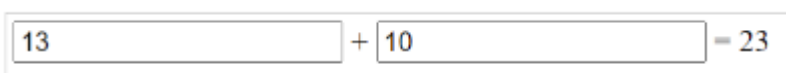
For example, consider the below code snippet:

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

Placeholder, Pattern, Min, Max and Step:**Placeholder:**

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

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

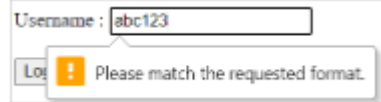
**Pattern:**

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.

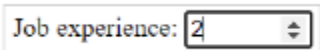
Username: `<input type="text" pattern="[A-Za-z]"/>`

**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 element.

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

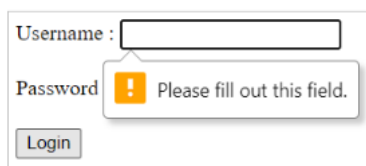
**Required and Multiple:****Required:**

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 message appears on the screen.

Username: `<input type="text" required/>`

Username: `<input type="password" required/>`

**Multiple:**

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

```
<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.

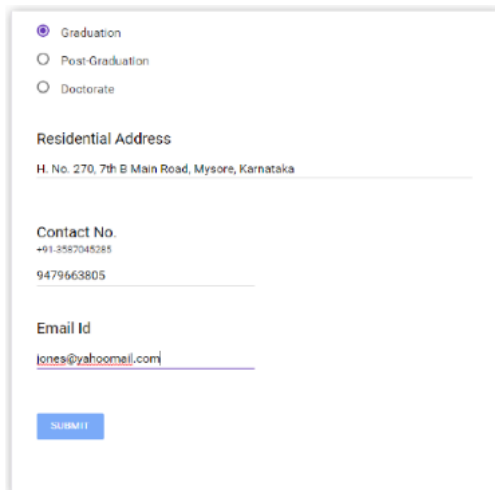
Form-override attribute	Description
Formaction	Overrides the form action attribute
formnovalidate	Overrides the form novalidate attribute
Formmethod	Overrides the form method attribute
Formtarget	Overrides the form target attribute

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.

```
<form method="GET" action="">
  <input type="submit" formmethod="POST">
</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>
```

Editing elements – Need

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.

List of Editing Elements

The following are elements used for editing.

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

Example:

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

HTML is building block of Internet

HTML is building block of Internet WWW

Embedded Elements – Need

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:

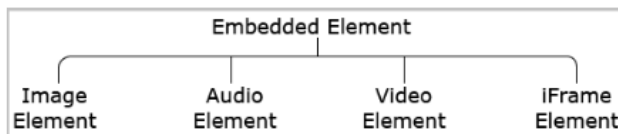


Image Element:

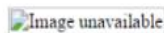
Embedding images to a web page can be done with the `...` tag. The `` tag has attributes 'src' and 'alt' where src defines the location of the image and alt defines the alternative text if it is failed to load the image inside the web page.

Example: ``

Output:



When image is available



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

Image Captions: 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.

```
<figure>
  
  <figcaption>A colorful tropical sea view</figcaption>
</figure>
```

The above code snippet generates an output as below:



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:

HTML5 supports `<audio>` tag which is used to embed sound content in an HTML or XHTML document as follows.

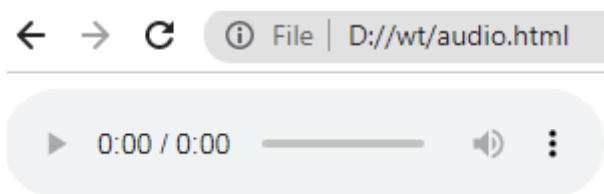
```
<audio src = "foo.wav" controls autoplay>
Your browser does not support the <audio> element.
</audio>
```

The current HTML5 draft specification does not specify which audio formats browsers should support in the audio tag. But most commonly used audio formats are **ogg**, **mp3** and **wav**.

```
<!DOCTYPE HTML>
<html>
  <body>
    <audio controls autoplay>
      <source src = "/html5/audio.ogg" type = "audio/ogg" />
      <source src = "/html5/audio.wav" type = "audio/wav" />
      Your browser does not support the <audio> element.
    </audio>

  </body>
</html>
```

This will produce the following result –



Audio Attribute Specification:

The HTML5 audio tag can have a number of attributes to control the look and feel and various functionalities of the control –

Sr.No.	Attribute & Description
1	Autoplay This Boolean attribute if specified, the audio will automatically begin to play back as soon as it can do so without stopping to finish loading the data.
2	Autobuffer This Boolean attribute if specified, the audio will automatically begin buffering even if it's not set to automatically play.
3	Controls If this attribute is present, it will allow the user to control audio playback, including volume, seeking, and pause/resume playback.
4	Loop This Boolean attribute if specified, will allow audio automatically seek back to the start after reaching at the end.
5	Src The URL of the audio to embed. This is optional; you may instead use the <source> element within the video block to specify the video to embed.

Video Element:

Here is the simplest form of embedding a video file in your webpage –

```
<video src = "foo.mp4" width = "300" height = "200" controls>
```

Your browser does not support the <video> element.

```
</video>
```

The current HTML5 draft specification does not specify which video formats browsers should support in the video tag. But most commonly used video formats are –

- **Ogg** – Ogg files with Theora video codec and Vorbis audio codec.
- **mpeg4** – MPEG4 files with H.264 video codec and AAC audio codec.

You can use <source> tag to specify media along with media type and many other attributes. A video element allows multiple source elements and browser will use the first recognized format – Live Demo

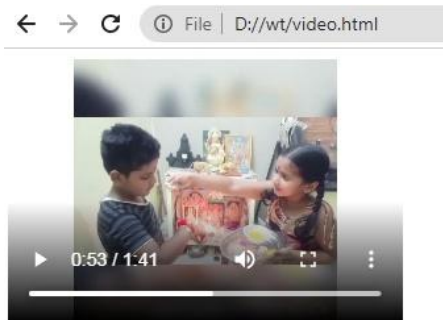
```
<!DOCTYPE HTML>
```

```

<html>
  <body>
    <video width = "300" height = "200" controls autoplay>
      <source src = "/html5/foo.ogg" type = "video/ogg" />
      <source src = "D://wt/video.html" type = "video/mp4" />
      Your browser does not support the <video> element.
    </video>
  </body>
</html>

```

This will produce the following result –



Video Attribute Specification

The HTML5 video tag can have a number of attributes to control the look and feel and various functionalities of the control –

Sr.No.	Attribute & Description
1	Autoplay This Boolean attribute if specified, the video will automatically begin to play back as soon as it can do so without stopping to finish loading the data.
2	Autobuffer This Boolean attribute if specified, the video will automatically begin buffering even if it's not set to automatically play.
3	Controls If this attribute is present, it will allow the user to control video playback, including volume, seeking, and pause/resume playback.
4	Height This attribute specifies the height of the video's display area, in CSS pixels.
5	Loop This Boolean attribute if specified, will allow video automatically seek back to the start after reaching at the end.
6	Preload This attribute specifies that the video will be loaded at page load, and ready to run. Ignored if autoplay is present.

7	Poster This is a URL of an image to show until the user plays or seeks.
8	Src The URL of the video to embed. This is optional; you may instead use the <source> element within the video block to specify the video to embed.
9	Width This attribute specifies the width of the video's display area, in CSS pixels.

iframe element:

The HTML <iframe> tag specifies an inline frame.

An inline frame is used to embed another document within the current HTML document.

Syntax: <iframe src="*url*" title="*description*"></iframe>

Tip: It is a good practice to always include a title attribute for the <iframe>. This is used by screen readers to read out what the content of the iframe is.

Iframe - Set Height and Width

Use the height and width attributes to specify the size of the iframe. The height and width are specified in pixels by default.

Example

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Iframes</h2>
<p>You can use the height and width attributes to specify the size of the iframe:</p>
<iframe src="demo_iframe.htm" height="200" width="300" title="Iframe Example"></iframe>
</body>
</html>
```

HTML Iframes

You can use the height and width attributes to specify the size of the iframe:

**This page is
displayed in an
iframe**

Or you can add the style attribute and use the CSS height and width properties:

Example

```
<iframe src="demo_iframe.htm" style="height:200px;width:300px;" title="Iframe Example"></iframe>
```

Iframe - Remove the Border:

By default, an iframe has a border around it. To remove the border, add the style attribute and use the CSS border property:

Example

```
<iframe src="demo_iframe.htm" style="border:none;" title="Iframe Example"></iframe>
```

Remove the Iframe Border

To remove the default border of the iframe, use CSS:

**This page is
displayed in an
iframe**

With CSS, you can also change the size, style and color of the iframe's border:

Example

```
<iframe src="demo_iframe.htm" style="border:2px solid red;" title="Iframe Example"></iframe>
```

Custom Iframe Border

With CSS, you can also change the size, style and color of the iframe's border:

**This page is
displayed in an
iframe**

Iframe - Target for a Link

An iframe can be used as the target frame for a link. The target attribute of the link must refer to the name attribute of the iframe:

Example

```
<iframe src="demo_iframe.htm" name="iframe_a" title="Iframe Example"></iframe>
```

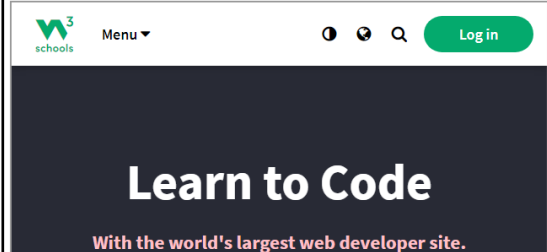
```
<p><a href="https://www.w3schools.com" target="iframe_a">W3Schools.com</a></p>
```

Iframe - Target for a Link

This page is displayed in an iframe

[W3Schools.com](https://www.w3schools.com)

When the target attribute of a link matches the name of an iframe, the link will open in the iframe.

Iframe - Target for a Link

[W3Schools.com](https://www.w3schools.com)

When the target attribute of a link matches the name of an iframe, the link will open in the 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.

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

HTML Injection

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.

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>");
```

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.,

Introduction to Clickjacking

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.

Clickjacking Mitigation Techniques

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.
- 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_012782317766025216289/web-hosted/assets/temp.mp3" onerror="alert('hacked')"></vi
```

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.

HTML Sanitization

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

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

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

Entity names for some HTML characters	
Input Character	Encoded value
<	< or <
>	> or >
"	" or "
'	' or '
&	& or &

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

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

```
<a href="#" onmouseover="alert('hacked')">Avengers</a>
```

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

```
&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.

Let us understand the usage of the DOMPurify library.

DOMPurify library

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.

```
<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

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.

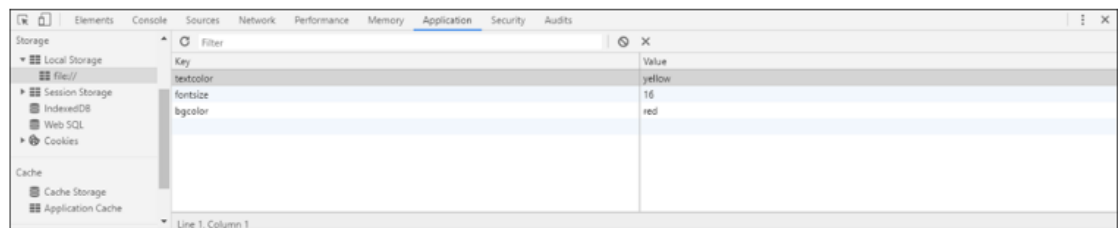
Local storage holds the data in the browser cache until the user deletes it or it expires based on the expiry date given. `setItem()` method is used to assign data to local storage. The below code creates three items with names `bgcolor`, `textcolor`, `fontsize` and assigns the values to them.

```
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.

Local Storage Vulnerabilities-Mitigations

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.

```
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

```
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:

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

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.

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

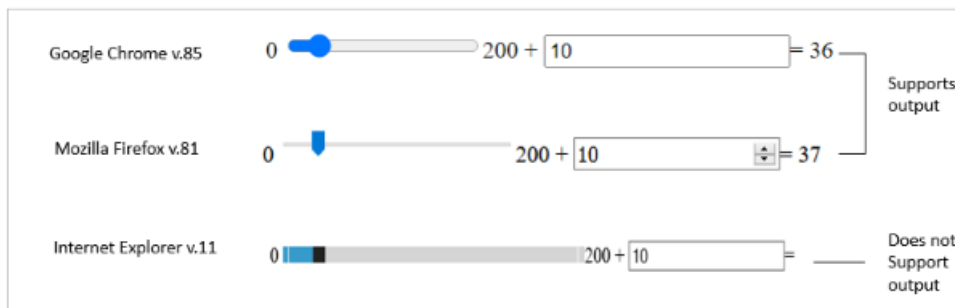
Also, all browsers may not support all features of HTML5.

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

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

```
<!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 Compatiblensess:

- 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 Creating 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 <div> tag and use more meaningful tags such as <article>, <section> and appropriate tags in the web pages.

10. Always use 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.

Cheat Sheet

Head Elements:

Tag	Description	Attributes	Possible Values
<title>	Specifies the title of the web page.	None	
<link>	Specifies the hyperlink to an external source.	href, rel	rel="stylesheet" href="mystyle.css"
<style>	Specifies the CSS styling inside HTML page.	None	
<script>	Specifies the JavaScript Code for the web page.	None	
<meta>	Specifies the additional meta information for the web page.	name,content	name='description', content="details of web page content"

Page Layout and Content Tags:

Tag	Description	Attributes	Possible Values
<header>	Specifies the wrapped text as the header of the web page document.	None	
<footer>	Specifies the wrapped text as the footer of the web page document.	None	
<article>	Specifies independent and self-contained content.	None	
<section>	Specifies a section in a web page document such as chapters.	None	
<aside>	Keeps the content aside from the main content.	None	
<nav>	Specifies the navbar portion of the web page.	None	
<main>	Specifies the main content of the document.	None	

<a>	Specifies hyper-link to either an external web page or some other division of the same web page.	href, rel, target	href="dummy.com" rel="noopener noreferrer" target="_blank, _self, _parent or _top"
<div>	Separate block-level division of the web page document.	None	
	Specifies the Inline division of an HTML element.	None	
	Specifies the unordered list.		
	Specifies the ordered list.	type, start, reversed	1,I,I,a,A
	Specifies the list item element.	None	
<dl>	Specifies the description list.	None	
<dt>	Specifies the Title of the description list element.	None	
<dd>	Specifies the definition/ details of the description list element.	None	

Text Content and Text Formatting Tags:

Tag	Description	Attributes	Possible Values
<h1>,<h2>,<h3>,<h4>,<h5>,<h6>	Heading tags where <h1> is the most important heading and <h6> is the least important heading.	None	
<p>	Specifies paragraph element.	None	
 	Specifies the line break.	None	
<blockquote>	Specifies quotes.	cite	cite="dummy.com"
<sup>	Format the text content as superscript.	None	
<sub>	Format the text content as a subscript.	None	
	Emphasize the text content.	None	
	Highlight the text content as important making it bold.	None	
<mark>	Highlight the text content marking it with a yellow background.	None	
	Specifies deleted text by striking off.	None	None
<ins>	Specifies inserted text by underline.	None	None

Forms Element:

Tag	Description	Attributes	Possible Values
<form>	Specifies the form element.	action, method	action="target.html" method="GET"
<input>	Specifies the input field.	type, placeholder, value, formaction	type="text", placeholder="Enter Username" value="" formmethod="POST"
<button>	Specifies an input button element.	None	None
<label>	Specifies label for an input element.	for	for = "id1"
<select>	Specifies the drop-down list.	value	
<option>	Specifies the options in the <select> drop down.	value, selected	value = "abc" selected = "true"
<datalist>	Specifies the auto-completion list.	None	None
<progress>	Specifies a progress bar	min, max, step, value	min = "0", max = "10", step = "1", value = "5"
<meter>	Specifies a meter gauge	min,max,step,value	min="0", max="10", step="1", value="1"
<output>	Displays output of user input	None	None
<textarea>	Custom sized text box to collect input.	rows, cols	rows = "2" cols = "2"

Table Elements:

Tag	Description	Attributes	Possible Values
<table>	Specifies the table element.	None	None
<thead>	Specifies the header portion of the table element.	None	None
<tbody>	Specifies the content portion of the table	None	None
<caption>	Specifies the heading of the table	None	None
<colspan>	The element that is used to logically group the related columns	None	None
<tr>	Specifies a new table row.	None	None
<td>	Specifies a new table column.	None	None

Embedded Elements:

Tag	Description	Attributes	Possible Values
-----	-------------	------------	-----------------

	The element that is used to embed an image into the web page.	src, alt	src="./demo.img" alt="Alternative text"
<audio>	The element that is used to embed audio into the web page.	src, controls, loop, autoplay, preload, muted	src='./audio.mp3' controls="controls"
<video>	The element that is used to embed video into the web page.	src, controls, loop, autoplay, preload	src="./video.mp4" controls="controls"
<source>	Specifies the different formats for video/ audio elements to be embedded into the web page.	src	src= "abc.mp4"
<figure>	Specifies a self-contained content along with a caption.	None	None
<figcaption>	Specifies a caption for a <figure> element	None	None