AIMERZ.ai
Aim.Act.Achieve

NOTES

Getting Started With HTML

Getting Started With HTML

Introduction to HTML

HTML is the backbone of web development. It stands for **HyperText Markup Language** and it helps set the structure of the content in our web pages.

Lets understand what is the meaning of HTML:

1. **HyperText:** Means the link that we include on our webpage. Word HyperText comes from the hyperlink.
2. **Markup:** To define and present text content.
3. **Language:** Language gives rules and syntax to follow while writing any code.

Without it, browsers wouldn't know the actual structure of your web page or how to display elements like texts, audio or images. The file extension is **.html, ex: index.html.**

.**htm** is also used as an extension for html files.
But we should use .html now for better consistency and compatibility,

Structure Of HTML Document

Let's create our First HTML Doc - “Hello World”

Step 1: Create a new file with the name “index.html”.

Step 2: Paste following code into the file and save.

```javascript
<!DOCTYPE html>
<html lang="en">
  <head>
    <title> Hello, World!</title>
  </head>

  <body>
    <h1>Hello, World!</h1>
    <p>This is my first HTML page.</p>
  </body>
</html>
```

Step 3: Open the file using any Browser and we are done with our first HTML project.

Now let's understand the above code one by one:

<!DOCTYPE html>: We start the file with `<!DOCTYPE html>`. It tells the browser what type of document to expect; in this case it is an HTML document.

- The <DOCTYPE> declaration represents the document type, and helps browsers to display web pages correctly.

- It must appear only once, at the top of the page(before HTML tag).
- The <DOCTYPE> declaration is not case sensitive. We can write any of the following, it does not give any error.

```javascript
<!DOCTYPE html>
<!DocType html>
<!Doctype html>
<!doctype html>
```

html lang="en" : It's the language attribute and it tells the browser the default language. In this case english -en.

Here the <html> tag represents the root of an HTML document or we can say it is the container for all other HTML elements(except <!DOCTYPE> tag)

What is a tag ?

HTML uses a system of "tags" to define the structure and layout of web pages. Tags are the building blocks of HTML documents and consist of angle brackets enclosing specific keywords. These tags provide instructions to web browsers on how to display the content.

An HTML tag is composed of an opening tag and, in some cases, a closing tag. The opening tag starts with the less-than symbol (<), followed by the tag name, and ends

with a greater-than symbol (>). The closing tag is similar, but it also includes a forward slash (/) before the tag name.

For example:
Opening tag: <tagname>
Closing tag: </tagname>

Note: We will learn more **tags** in detail in the upcoming lecture.

<head>: This is the head section of the HTML document, which contains meta-information about the document, such as the title of the web page that appears in the browser's title bar or tab.

meta tag:
The **<meta>** tag in HTML is used to provide metadata about the HTML document. Metadata is data about the data, and in the context of HTML, it includes information such as the character set, page description, keywords, author of the document, and viewport settings.

```javascript
 <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
```

- **<meta charset="UTF-8">:**
  This meta tag specifies the character encoding for the HTML document.

1. **Charset:** This attribute defines the character encoding. Character encoding is a system of converting bytes into characters. UTF-8 (Unicode Transformation Format - 8 bit) is a popular character encoding that can represent every character in the Unicode character set.

2. **UTF-8:** This is a variable-width character encoding used for electronic communication. It can represent any character in the Unicode standard, yet it is backward compatible with ASCII. UTF-8 is the dominant character encoding for the web.

3. **Importance:** Using <meta charset="UTF-8"> ensures that your web page can display any character correctly, including special symbols and characters from different languages. It prevents issues like garbled text when displaying content.

- **<meta name="viewport" content="width=device-width,initial-scale=1.0">:**

  This meta tag controls the layout on mobile browsers.

  1. **Viewport:** The viewport is the user's visible area of a web page. It's different from the screen's actual size, which might be larger or smaller.

  2. **Width:** The width=device-width part sets the width of the viewport to be the same as the width of the device. This means that your page will be as wide as the device screen.

  3. **Initial-scale:** The initial-scale=1.0 part sets the initial zoom level when the page is first loaded. A scale of 1.0 means no zoom; the content is displayed at 100% scale (actual size).

  4. **Importance:** This tag is essential for responsive web design. It helps ensure that your website is displayed correctly on all devices, from desktops to tablets to smartphones. Without this tag, mobile browsers may render pages at a desktop screen width and then scale

them down, which often leads to poor user experience.

<title>: This is the title tag, which sets the title of the web page. In this example, it's set to "Hello, World!".

The **<title>** tag is like a label for a webpage. It should only have words, and it appears at the top of your web browser or on the tab of the page.

We have to use the <title> tag in our webpage!

The words in the title are really important for getting your webpage found in search engines (like Google). Search engines use the title to figure out where to put your page in the search results.

The <title> tag does three things:

- It gives a name for the web page in the web browser.
- It gives a name for the webpage when you save it in your favourites.
- It gives a name for the web page in the search results.

Here are some ideas for making good titles:

- Make the title longer and describe what the page is about. Short titles with only one or two words are not as good.
- Search engines only show about 50-60 letters of the title, so don't make it too long.

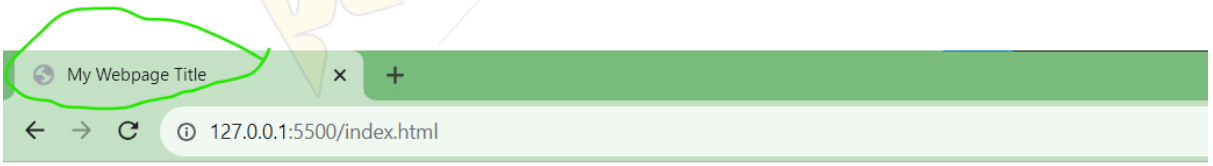
- Don't just list words in the title. If you do, your webpage might not show up as much in search results.
- So, try to make the title accurate and meaningful!

Remember: We can only have one <title> tag in your webpage.

Let's take another example:

```javascript
<!DOCTYPE html>
<html>
<head>
    <title>My Webpage Title</title>
</head>
<body>
    <h1>Welcome to My Webpage</h1>
    <p>This is a sample webpage with a title.</p>
</body>
</html>
```

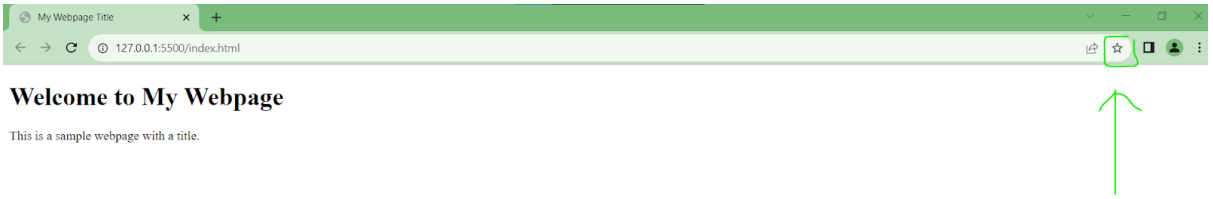
Output:

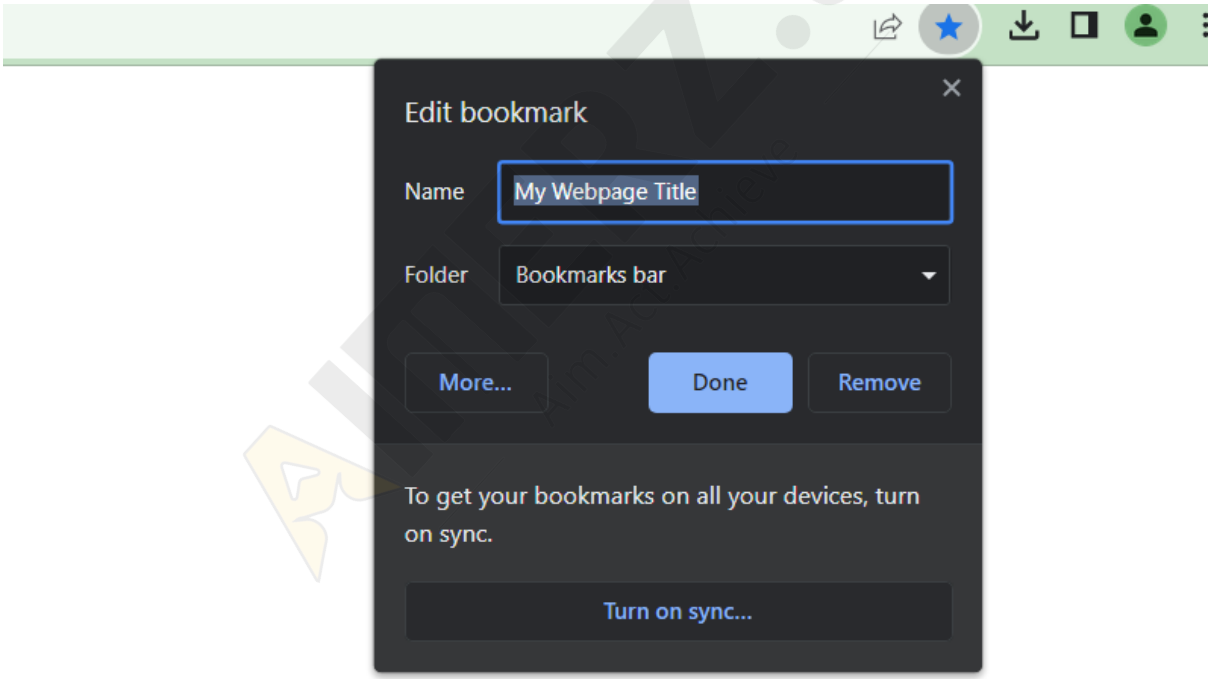


Welcome to My Webpage

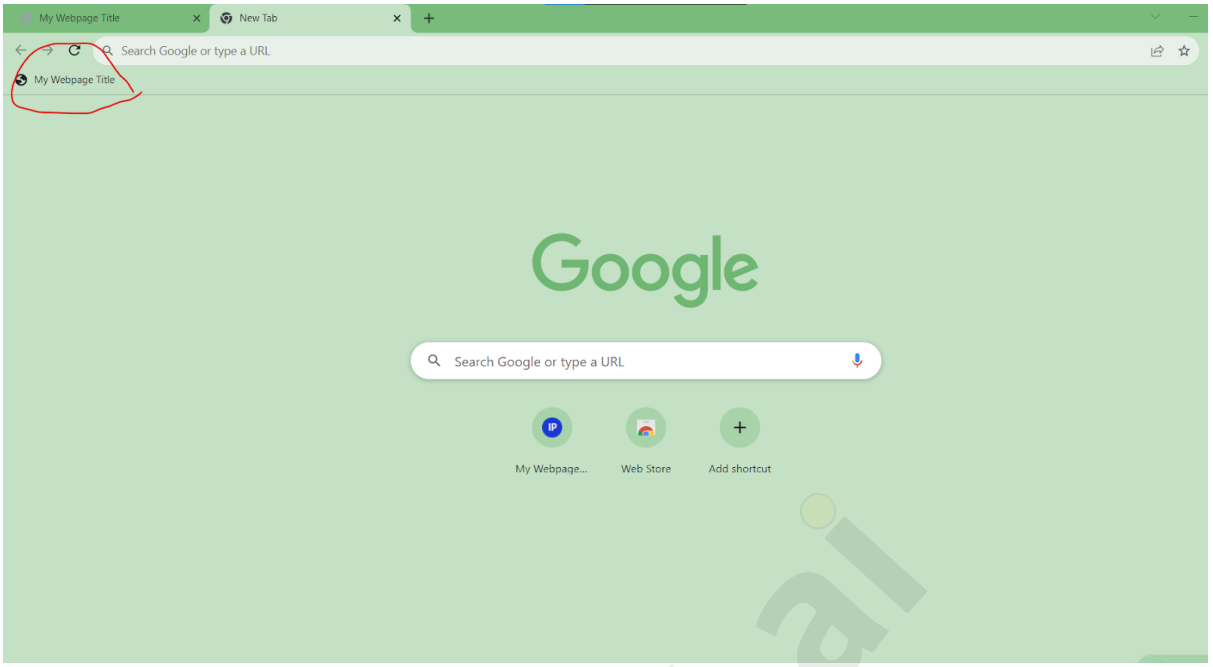
This is a sample webpage with a title.

When we bookmark a webpage, the title we set using the <title> tag appears as the name of the bookmark. For example in the above we set the title tag as **My Webpage Title. When we bookmark it, this shows as below:**

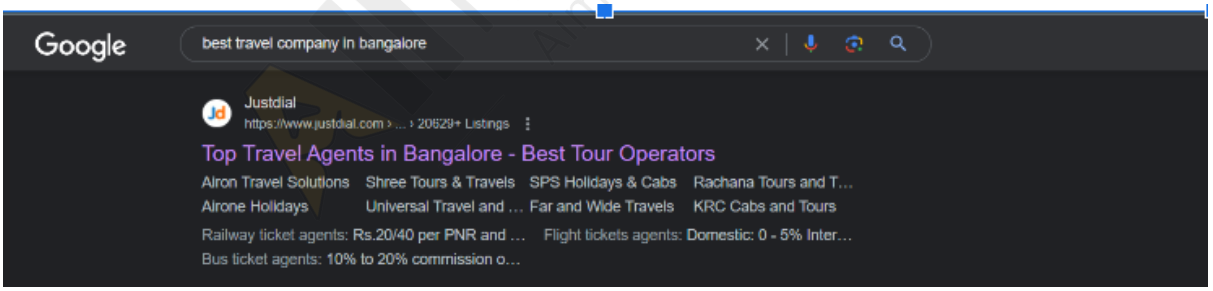




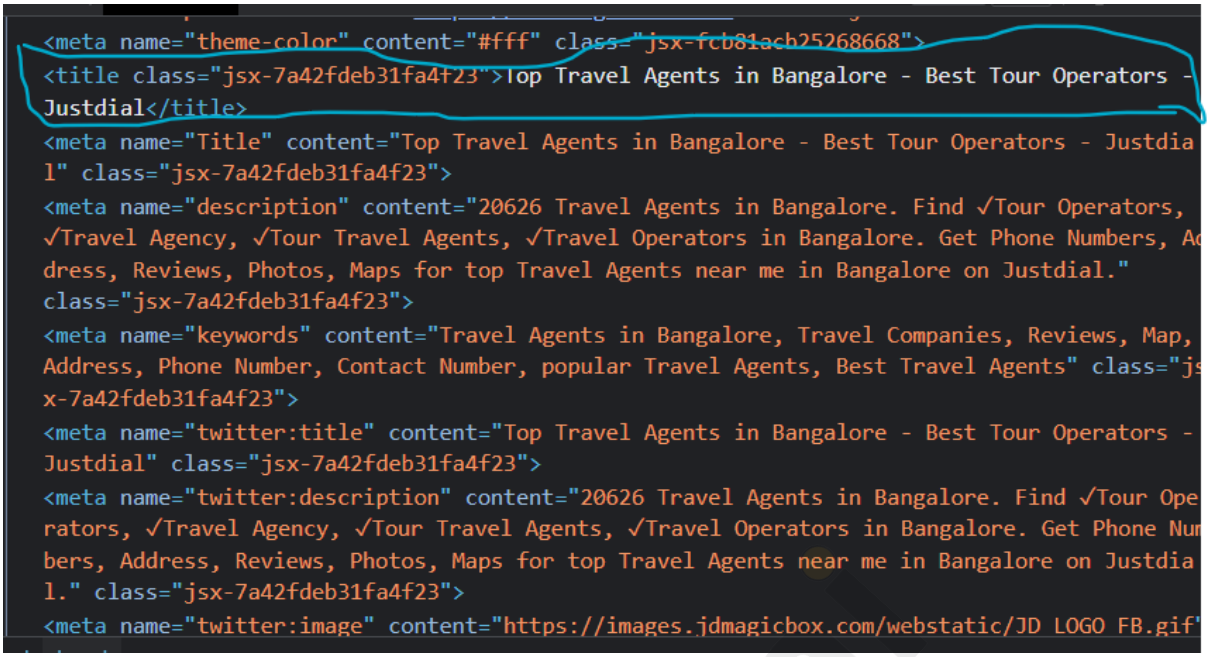
After the bookmark, we can easily see it



When we search for something on a search engine like Google, the titles of the webpages in the search results are the ones defined by the <title> tag. For instance, if you search for **"best travel company in bangalore,"** the titles of the search results will be the titles set using the <title> tag on those webpages.



If we inspect it(right click and tab the inspect button), we can easily see that in the title tag.



Note: We learn more in detail what is the use of inspect and all in the upcoming lectures.

<body>: This is the body section of the HTML document, which contains the visible content of the web page, such as headings, paragraphs, and more.

<h1>: This is a heading tag, used to define a top-level heading. In this example, it displays the text "Hello, World!" as a heading.

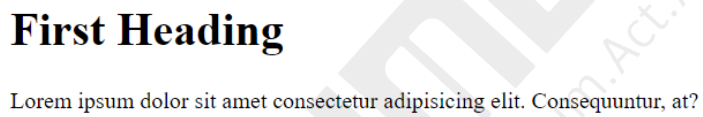
<p>: This is a paragraph tag, used to define a paragraph of text. In this example, it displays the text "This is my first HTML page." as a paragraph.

When you don't close an HTML tag properly, such as an <h1> tag, it leads to what's known as "open" tags. The browser will still try to render the content as best as it can, but it may have unexpected and undesirable consequences on the layout and appearance of the webpage.

Let's take an example where we write the syntax correctly:

```
JavaScript
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <h1> First Heading</h1>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
Consequuntur, at?</p>
</body>
</html>
```

Output:



If we don't close the h1 tag , then the browser behaves differently:

```
JavaScript
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <h1> First Heading
```

```
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
Consequuntur, at?</p>
</body>
</html>
```

Output:

First Heading

Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequuntur, at?

In this example, the <h1> tag is not closed properly with a </h1> tag. When you view this page in a web browser, the rendering will try to interpret the content, but it causes unexpected results as above.

To avoid these issues, it's essential to always close HTML tags properly. In this case, you should close the <h1> tag with </h1> like this

What are Vs Code Extensions ?

Imagine you have a toolbox full of special tools that make building things easier. Vs code extensions are like those tools , but for your coding. They add extra abilities to help you code faster and better.

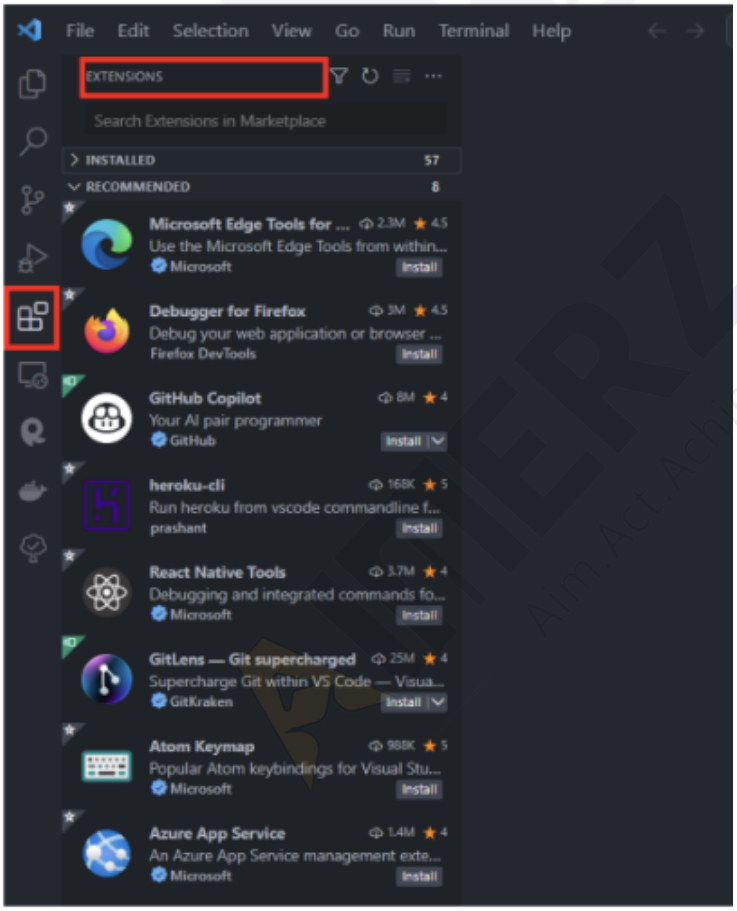
Why Should You Use Extensions?

Using Vs code extensions is like having a secret helper by your side. They speed up your work , fix mistakes, and make coding more fun . Just like adding cool apps to your phone ,extensions make your coding software awesome and tailored to your needs. So let's explore these helpers and make your coding journey amazing!

How to Get and Use VS Code Extensions

Getting VS code extensions is as easy as getting a new app for your phone

1. **Open Vs Code**
2. **Go to extensions:** Look for the extensions icon on the sidebar and click on it.



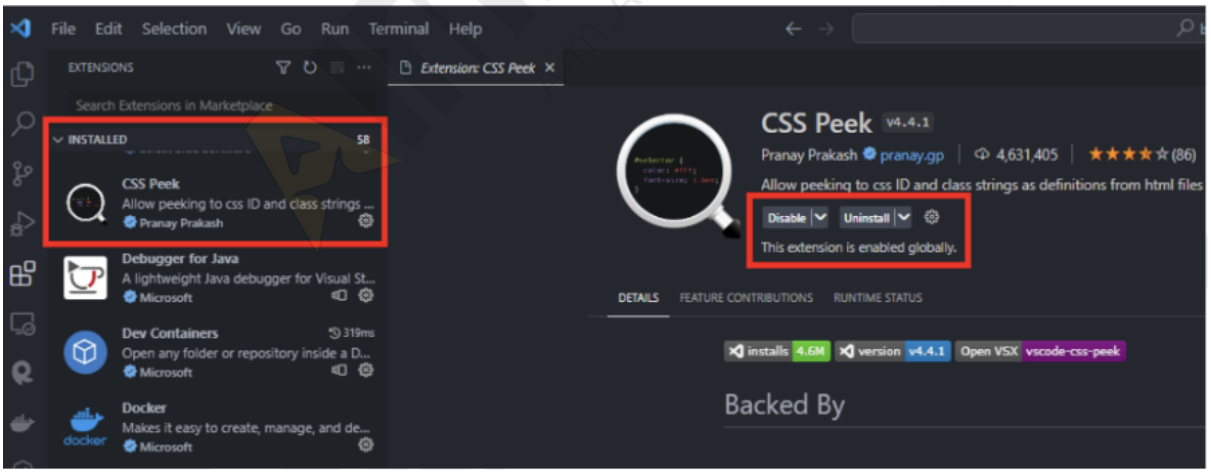
3. Search & install: In the extension marketplace , search for the extensions you want , when you find it , click **install**

Enjoy the magic: once installed , your new extensions are ready to use. You might need to configure it but that's usually easy too.

Enable and Disable Extension (if You Ever Need To)

Extensions are like superhero costumes - you can wear them when you want and take them off when you're done. Here's how you can enable and disable your extensions:

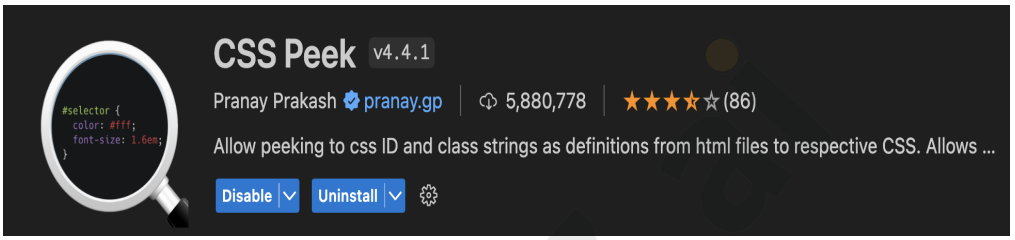
Manage Extensions: Click on the Extensions icon again.
View Installed Extensions: You'll see a list of installed extensions. You can turn them on or off with a switch.



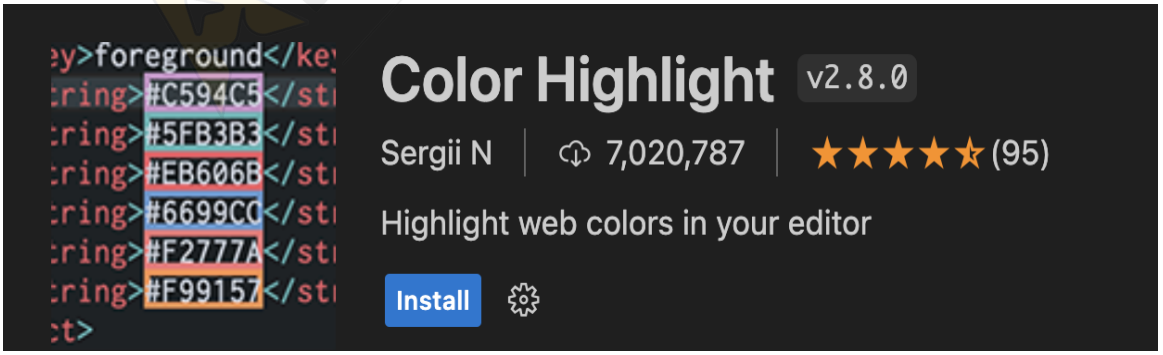
Restart VS Code: Sometimes, you might need to restart VS Code to apply changes after turning an extension on or off.

Must-Have VS Code Extensions for web Development

1. CSS Peek - Link
   Quick Looks: With CSS Peek, you can peek into the styles right from your HTML or JavaScript code.
   Fast Navigation: Imagine having a magic window that shows you the CSS behind an element. That's what
   CSS Peek does - it makes finding styles super fast.
   Understand Styles: When you want to know how something looks so great, CSS Peek lets you see its styles.
   It's like looking behind the scenes of a movie.

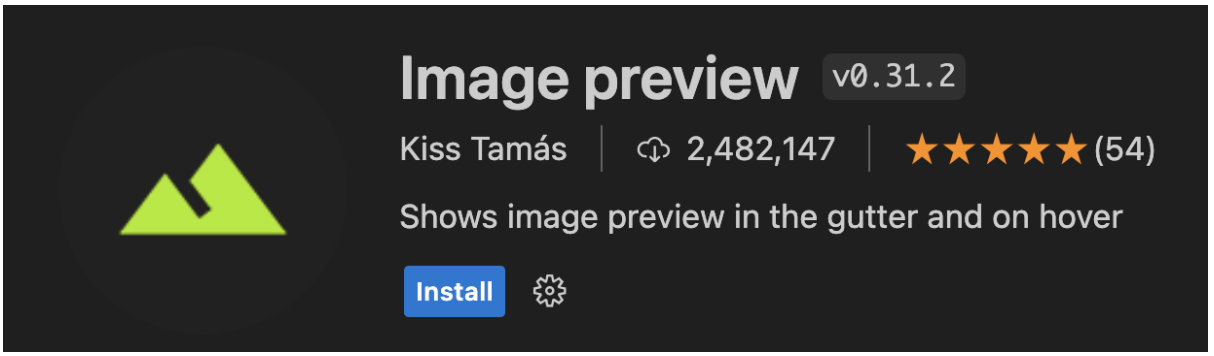


2. Color Highlight - Link
   Color Magic: Color Highlight makes your color codes burst with real colors as you type them.
   Instant Visuals: Imagine typing "blue" and seeing a blue shade. That's what Color Highlight does!
   Design Bliss: When you're coding styles for your websites, colors matter a lot. Color Highlight lets you n visualize them instantly.



3. Image Preview - Link
   Visual Boost: Image Preview works like a magical window that shows images right where you're coding.
   No Extra Steps: Instead of opening images elsewhere, Image Preview brings them to you within your code editor.

Web Design Help: When you're creating websites, images are like puzzle pieces. Image Preview helps you fit them perfectly into your design.



4. HTML CSS Support - Link

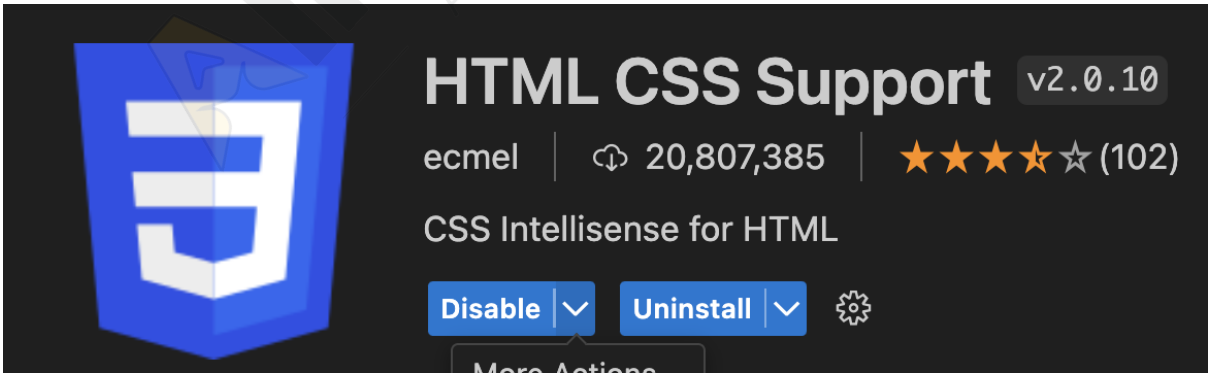
Smart Completion: It suggests and completes HTML id and class names as you type, preventing mistakes.

Stylesheet Help: Whether it's linked or embedded styles, this extension has your back.

Multi-Stylesheet Support: Juggling multiple stylesheets? HTML CSS Support handles it like a pro.

Languages Covered: It's not just for HTML - other HTML-like languages are supported too.

Validation Check: Validate your CSS selectors for accuracy whenever you need.

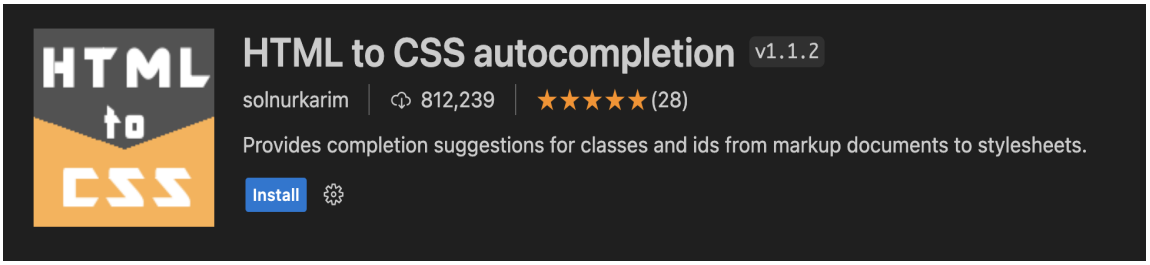


5. HTML to CSS autocompletion - Link

Smart Suggestions: This clever helper, when you're typing HTML class names, suggests CSS class names you've used before.

Error Prevention: By suggesting class names you've already used, it helps you avoid typos and errors.

Consistent Styles: HTML to CSS Auto Completion ensures that your website's styles stay the same across your code.

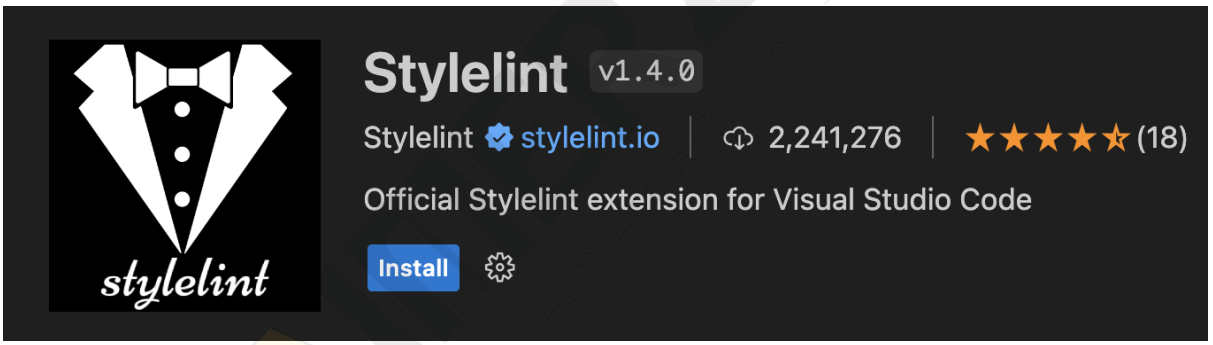


6. Stylelint - Link

   Coding Police: Stylelint checks your code for errors and style issues, making sure your code follows the rules.

   Highlighting Mistakes: It points out mistakes with flashy lights, so you can easily see what needs fixing

   Code Excellence: By enforcing coding standards, Stylelint helps your code look clean, professional, and consistent.



HTML Comments

Comments are useful when you want to write something about the code but you do not want to run that or show in the output. Comments are basically for humans, you write comments for your future self or for other developers who might work on the same codebase.

In HTML we can create a comment as shown below and the browser will not display that on the screen.

To create a comment we write the text between; `<!-- -->`

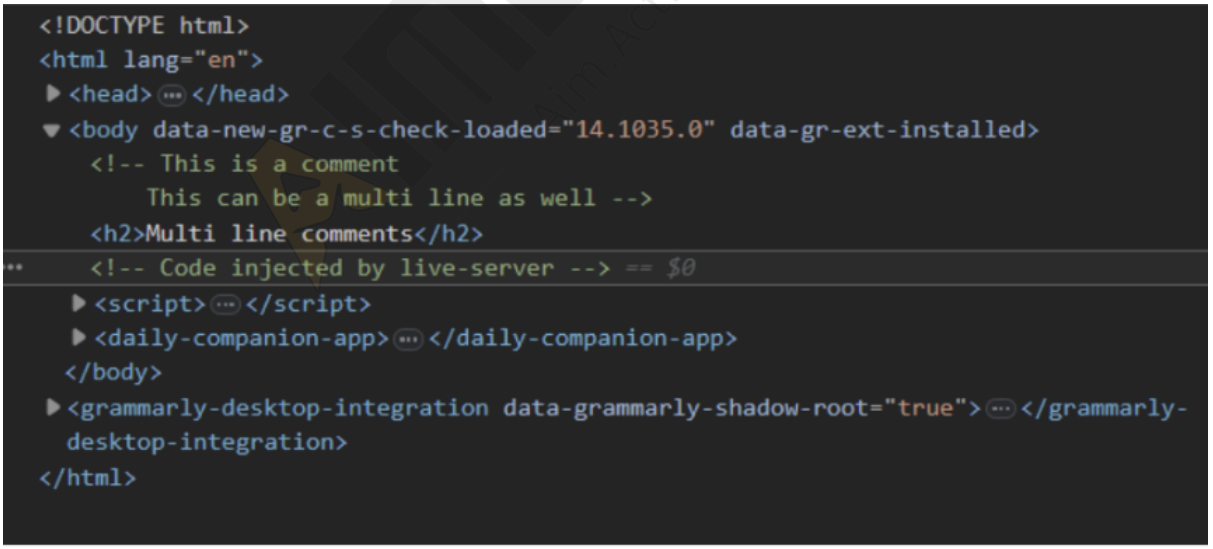
Example:

```
JavaScript
 <!-- This is a comment -->
<h2>Single line comment</h2
```

In the above code we can see a single line comment, and this exactly is the way to create a multi line comment.

```
JavaScript
<!- - This is a comment
This can be a multi line as well -- >
<h2>Multi line comments</h2>
```

In the example above we can see we have a multi line comment and the syntax to create a multi line comment is similar to how we create a single line comment.



HTML comments serve various purposes, including:
Documentation: Developers often use comments to explain the purpose or functionality of certain code blocks, making it easier for themselves or others to understand the codebase.

Debugging: Comments can help in troubleshooting or temporarily disabling sections of code for testing purposes without deleting them.
Reminder: Comments can act as reminders for future modifications or improvements to the code.
Work in progress: Comments can mark unfinished sections or indicate that certain code is under development.

Best Practices

- Use comments sparingly.
- Do not write the obvious in the comments Ex: Do not write, **“this is a div tag,** or **“this is a p tag”**
- Explain why you did what you did.
- If it is an expression then write the output.
- Use comments to document a piece of code, but do not just rely on comments for documentation.

Element & Tag in HTML

What is a tag ?

HTML uses a system of "tags" to define the structure and layout of web pages. Tags are the building blocks of HTML documents and consist of angle brackets enclosing specific keywords. These tags provide instructions to web browsers on how to display the content.

An HTML tag is composed of an opening tag and, in some cases, a closing tag. The opening tag starts with the less-than symbol (<), followed by the tag name, and ends with a greater-than symbol (>). The closing tag is similar, but it also includes a forward slash (/) before the tag name.

For example:

Opening tag: <tagname>
Closing tag: </tagname>

HTML tags can be broadly categorised into two types:

1. **Paired Tags(or Container Tags):**
   Paired tags have both an opening and a closing tag and can contain other HTML elements and text within them. They define a specific section or structure in the web page. Some common paired tags include:
   Some common examples are **<html>, <head>, <p>, etc...**

2. **Singular Tags(or Void Tags)**

In HTML, some are self-closing tags or void tags because they don't require a separate closing tag. These elements are used for elements that don't contain any content or don't have a closing counterpart.

**Example: <img/>, <input/>,
, and <meta/> etc.**

What is an element?

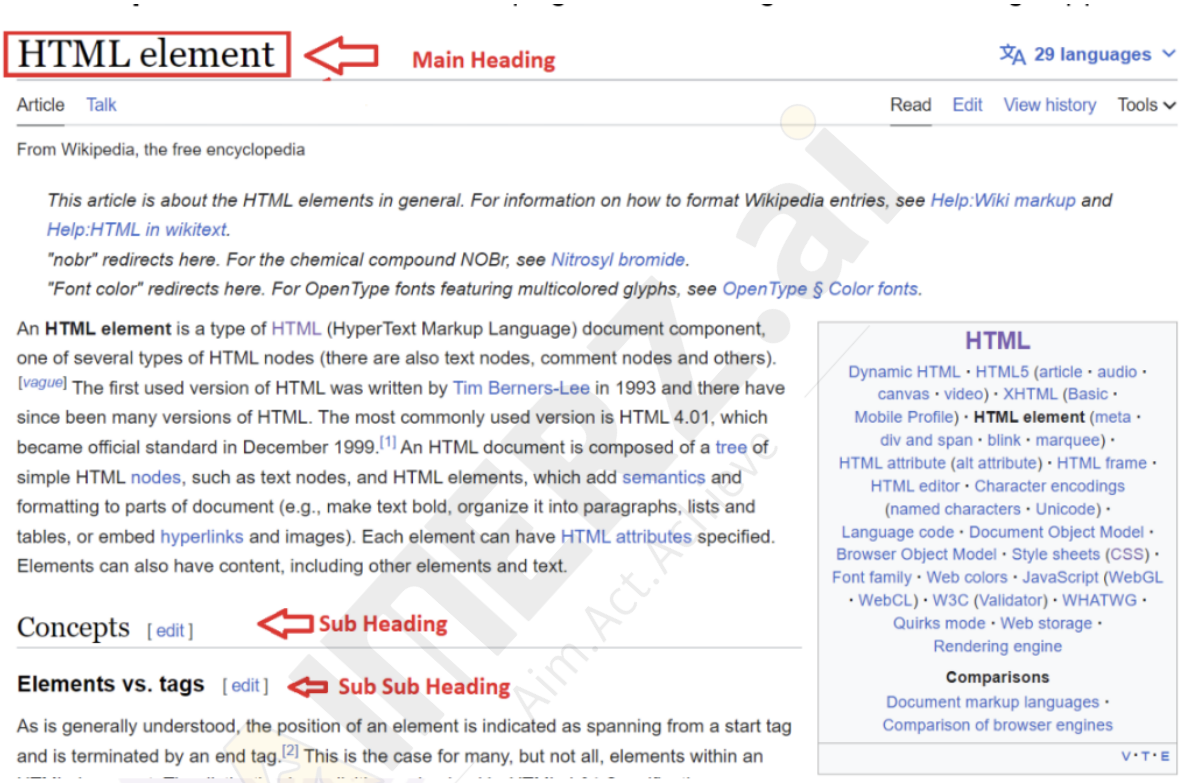
HTML elements are the building blocks of an HTML document. HTML elements consist of a start tag, an end tag (if applicable), and content in between the tags. The start tag defines the beginning of an HTML element, and the end tag defines the end of an HTML element.

Each HTML element has a specific structure that defines its behaviour and appearance on a web page.
Here is a breakdown of the anatomy of an HTML element:

1. **Opening tag** - This is the opening tag that defines the beginning of an HTML element . It is enclosed in angle brackets (< >).
2. **Content** - This is the text or other data that appears between the start and end tags of an HTML element. Some HTML elements do not require content and are self-closing.
3. **Closing tag** - This is the closing tag that defines the end of an HTML element. It is enclosed in angle brackets with a forward slash (/) before the element name.

Some common Tags

1. **Heading(h1, h2,h3,h4,h5,h6):** Headings are used to define the titles or subtitles of a web page or a section of content.
   For Example, look into the below wiki page, how headings and subheadings appear in a web page.

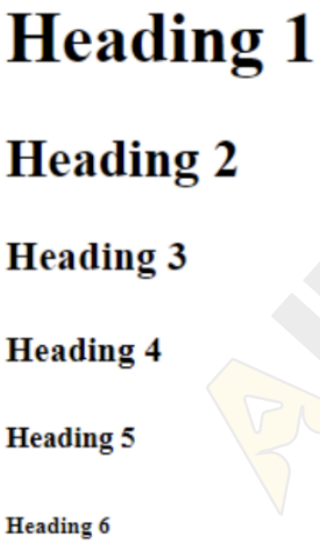


To put Heading in HTML doc, we use <h1>, <h2>, <h3> ...<h6> tags. From h1 to h6, represents different levels of content.

```javascript
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width,
initial-scale=1.0" />
```

```
<title>Heading</title>
</head>
<body>
<h1> Heading 1</h1>
<h2> Heading 2</h2>
<h3> Heading 3</h3>
<h4> Heading 4</h4>
<h5> Heading 5</h5>
<h6> Heading 6</h6>
</body>
</html>
```

Output:



Note: Browsers automatically add some white space(a margin) before and after a heading.

Headings are important:

- Search engines use the headings to index 1he structure and content of your web pages.
- Users often skim a page by its headings. It is important to use headings to show the document structure.
- <h1> headings should be used for main headings, followed by <h2> headings, then the less important <h3> and so on.

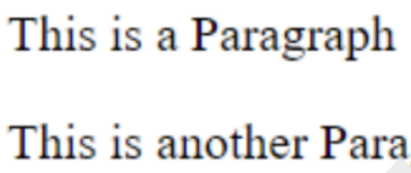
Note: Use HTML headings for headings only. Don't use headings to make the text Big or Bold.

2. **Paragraph:** The HTML <p> element defines a paragraph.
   A Paragraph always starts on a new line, and browsers automatically add some whiłe space(a margin) before and after a paragraph.

   Example:

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

Output:



3. **<hr> tag:** The <hr> tag defines a thematic break in an HTML page, and is most often displayed as a horizontal line.
   The <hr> element is used to separate content(or define a change) in an HTML page.

```
JavaScript
<h1> This is heading 1 </h1>
<p> This is some text </p>
<hr>
<h2> This is heading 2 </h2>
<p> This is some other text </p>
<hr>
```

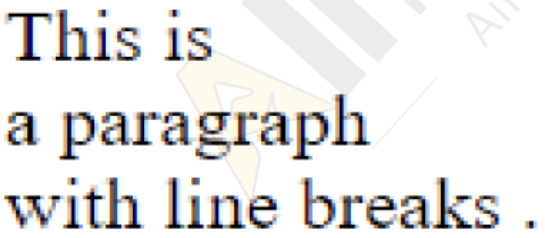
Output:

The <hr> tag is an empty tag, which means that it has no end tag.

4. **HTML Line Breaks:** The HTML
 element defines a line break. Use
 if a line breaks (a new line) without starting a new paragraph.

```javascript
<p> This is <br> a paragraph <br> with line breaks . </p>
```

Output:



The
 tag is an empty tag, which means that it has no end tag.

5. **span tag:** The tag functions as an inline container , allowing us to markup and highlight specific portions of text within a document . It serves to emphasize or apply formatting to those particular sections without affecting the overall layout or structure of the page.

Introduction to Emmet

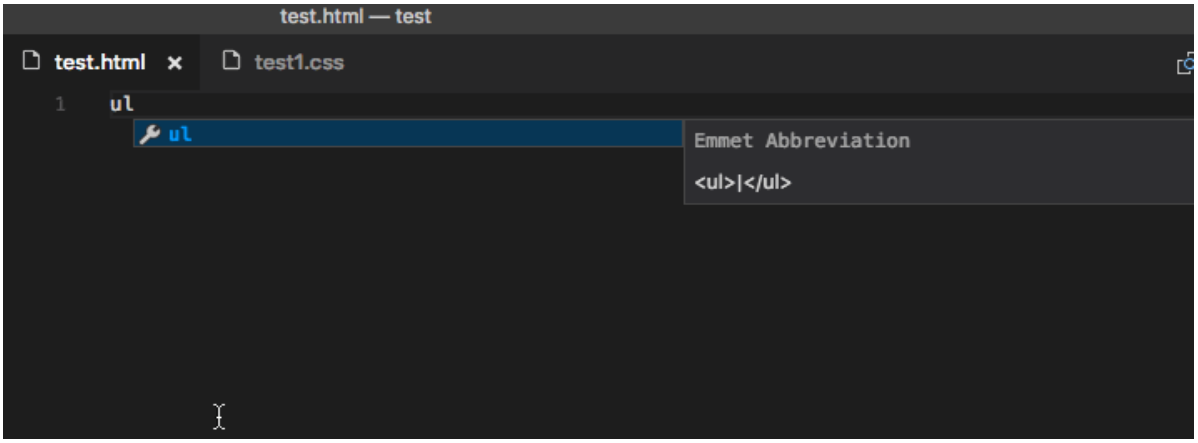
Emmet is a powerful web development tool that enhances productivity by allowing web developers to write HTML and CSS code using abbreviations and shortcuts. It originated as a plugin called Zen Coding, created by Sergey Chikuyonok, and has since evolved into a widely adopted tool by web developers.

What is Emmet?

Emmet is a shorthand syntax that helps you write HTML and CSS code more quickly and efficiently. It uses abbreviations that expand into standard HTML and CSS code, so you can write complex structures with just a few keystrokes. This can save you a lot of time and typing, and it can also help you write more consistent and error-free code.

For example, to create an HTML div element with the class **"my-class",** you would type **div.my-class**. Emmet will then expand this abbreviation into the full HTML code for the div element.

Emmet is a powerful tool that can be used by web developers of all levels of experience. It is a great way to improve your productivity and code quality, and it can also help you learn HTML and CSS faster.



Why use Emmet?

Emmet is a powerful tool that can help web developers write code faster, easier, and more consistently. Here are some of the benefits of using Emmet:

- Increased productivity: Emmet allows you to write code at a faster pace, speeding up your development workflow. With its abbreviation-based approach, you can express complex structures and repetitive patterns using just a few keystrokes.

- Simplified syntax: Emmet introduces a simplified syntax that is easy to learn and use. It leverages the familiarity of CSS selectors and combines it with HTML element names to create powerful and expressive abbreviations.

- Code generation: Emmet can generate repetitive code snippets with ease. **For example,** you can use Emmet to quickly create a list of multiple HTML elements, each with an incremental ID or class, in a single command. This feature saves time and reduces the chances of human error when writing repetitive code.

- Editor and IDE support: Emmet is supported by a wide range of popular text editors and integrated development environments (IDEs), including Visual Studio Code, Sublime Text, Atom, and more. This extensive support ensures that you can integrate Emmet into your preferred coding environment, enhancing your coding experience.

Basic Syntax and Abbreviation.

Emmet follows a specific syntax and uses abbreviations to generate HTML and CSS code. Here's an overview of the basic syntax and abbreviations:

1. **HTML Elements:** To generate HTML elements, you can simply write the name of the element. For example,

div and **p** expands to:

```
JavaScript
<div></div>
<p></p>
```

2. **Nested Elements:**

   You can nest elements inside each other by using the **>** (greater-than) symbol. For example:

div>ul>li expands to:

```
JavaScript

<div>
  <ul>
    <li></li>
  </ul>
</div>
```

3. **Element IDs and Classes:**

To add an ID to an element, use the **#** (hash) symbol, followed by the ID name. To add a class, use the **.** (dot) symbol, followed by the class name. For example:

div#lead and **div.container** expands to

```
JavaScript
<div id="lead"></div>
<div class="container"></div>
```

Note: You can also use **.container** or **#lead** to create div specifically.

4. **Sibling Elements:**
   You can create sibling elements by using the **+** (plus) symbol. For example:

h1+p expands to:

```
JavaScript
<h1></h1>
<p></p>
```

5. **Multiplication:**
   You can use multiplication operators (*****) to generate multiple elements. Specify the number followed by the element abbreviation. For example:

ul>li*3 expands to:

```
JavaScript
<ul>
       <li></li>
       <li></li>
       <li></li>
</ul>
```

6. **Text Contents:**
   To add text content to an element, enclose the desired text within curly braces **{ }**. For example:

h1{Hello} expands to:

```
JavaScript
<h1>Hello</h1>
```

7. **Attributes:**
   You can include attributes in an element by using square brackets **[ ]** and specifying the attribute name and value. For example:

a[href="https://example.com"] expand to:

```
JavaScript
<a href="https://example.com"></a>
```

What are HTML Attributes?

In HTML, attributes provide additional information or modify the behavior of HTML elements. They are used to enhance the functionality and appearance of elements within an HTML document. Attributes are specified within the opening tag of an HTML element and consist of a name-value pair.

Syntax for an HTML attribute :

```
JavaScript
<tagname attribute_name="attribute_value">
```

- **tagname:** Represents the HTML element to which the attribute is applied.
- **attribute_name:** Refers to the specific attribute being used.
- **attribute_value:** Specifies the value associated with the attribute.

Global Attributes:

Global attributes can be used with any HTML element. They provide common functionalities and properties that apply universally across different elements. Some commonly used global attributes include:

- **class:** Specifies one or more CSS classes to be applied to the element.
- **id:** Provides a unique identifier for an element.
- **style:** Defines inline CSS styles to be applied to the element.
- **title:** Specifies additional information about the element (often displayed as a tooltip).
- **data-***: Allows the addition of custom data attributes for JavaScript or CSS usage.

Specific Attributes

They are designed to serve specific purposes and are only applicable to their corresponding elements. Examples of specific attributes include:

- **href (anchor tag <a>):** Defines the URL or destination of a hyperlink.
- **src (image tag <img>)**: Specifies the source URL of an image.
- **value (input tag <input>):** Sets the initial value for an input field.
- **disabled (various tags):** Disables the functionality of an element.

Style Attributes

The style attribute allows you to apply CSS (Cascading Style Sheets) directly to HTML elements. CSS is a language that describes how HTML elements should be displayed on the screen. It provides a wide range of properties and values that you can use to control the appearance of elements.

To use the style attribute, you simply add it to an HTML tag, specifying the desired CSS rules within quotation marks. These rules consist of property-value pairs, where properties define the aspect of an element you want to change, and values determine how that change should be applied.

For example, let's consider a heading **<h1> tag** that you want to style. You can use the style attribute to change its **color, font size, and alignment.**
Here's an example:

```javascript
<h1 style="color: blue; font-size: 24px; text-align: center;">This is a
styled heading</h1>
```

You can customize other elements as well, such as paragraphs, links, images, and more, by applying the style attribute with appropriate CSS rules.

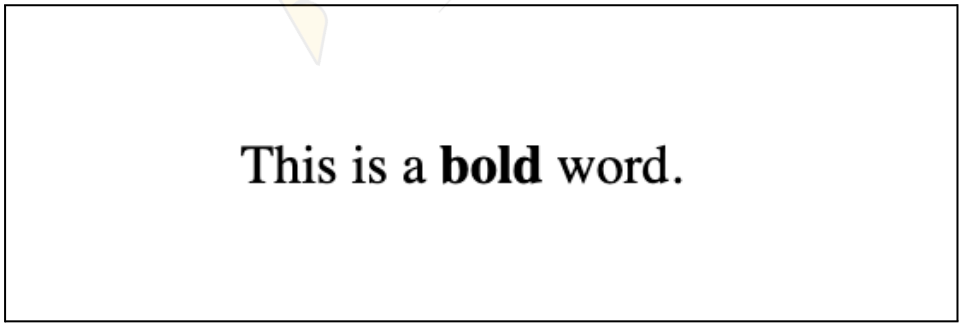
Understanding Formatting Tag

Formatting tags in HTML are used to style text and change its appearance.

By using formatting tags, you can enhance the visual presentation of your content and make it stand out. Here are some commonly used formatting tags in HTML:

1. **Bold(<b>):** The <b> tag is a non-semantic tag used to make the enclosed text appear in a bold font weight.

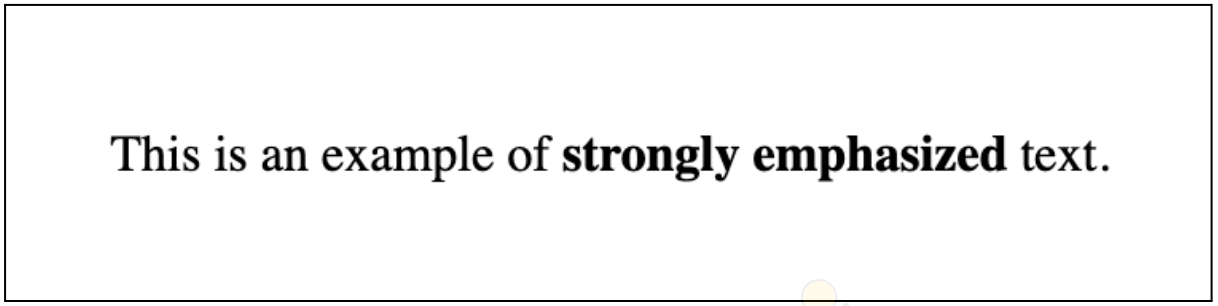
```javascript
<p>This is a <b>bold</b> word.</p>
```



2. **Strong():** The tag is a semantic tag used to indicate strong importance or emphasis.

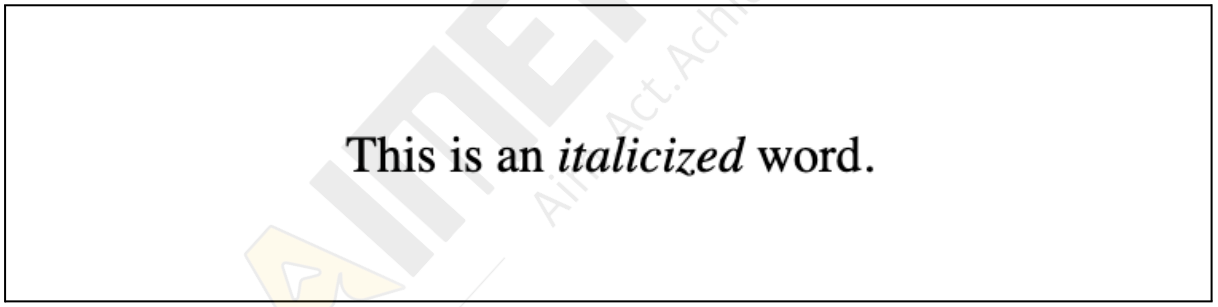
```
JavaScript
<p>This is an example of <strong>strongly emphasized</strong> text.</p>
```

Output:



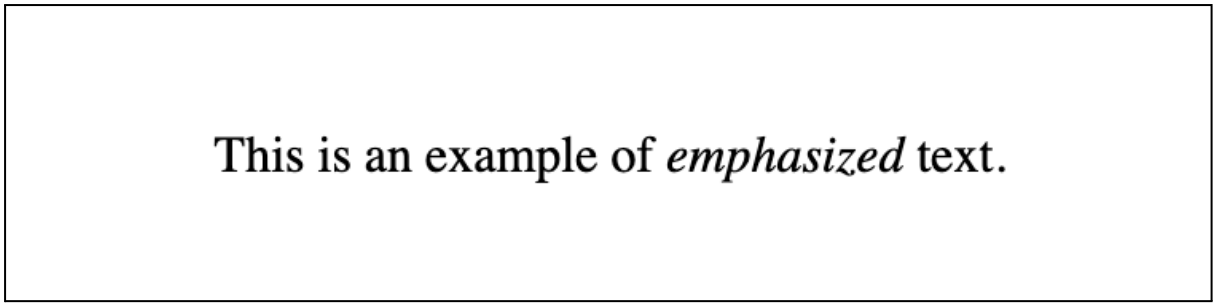
3. **Italic <i> Tag:** The <i> tag makes text italic.

```
JavaScript
<p>This is an <i>italicized</i> word.</p>
```



4. **Emphasis ():** The tag emphasizes text, typically displayed in italic. It has semantic meaning, indicating stress emphasis.

```
JavaScript
<p>This is an example of <em>emphasized</em> text.</p>
```



5. **Mark tag(<mark>):**

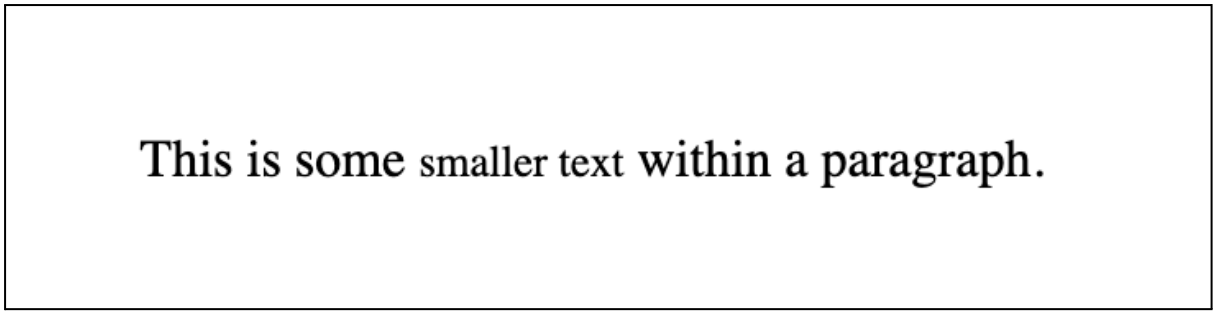
The <mark> tag is a non-semantic tag used to highlight or mark specific sections of text within the content. It is typically rendered with a background color to visually distinguish the marked text. This tag is useful for drawing attention to important or relevant information, such as search terms or key findings within a document.

```
JavaScript
<p>This is <mark>highlighted</mark> text.</p>
```

6. **Small tag(<small>):**

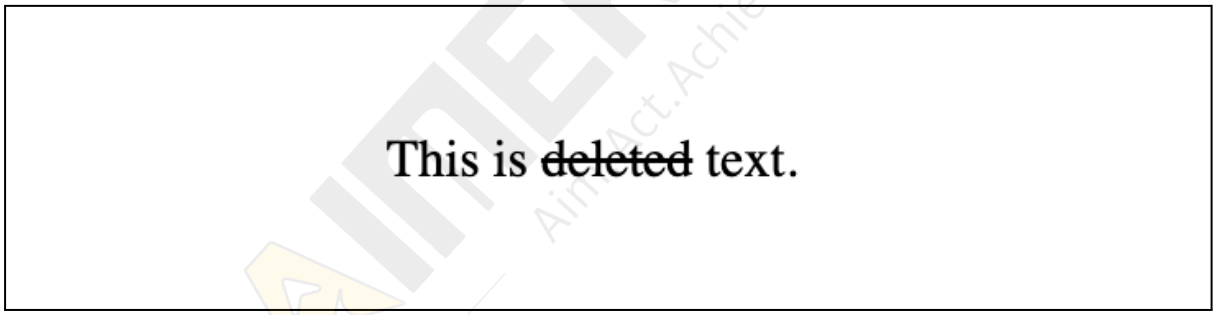
The <small> tag is a non-semantic tag used to decrease the font size of the enclosed text, making it appear smaller than the surrounding text. It is often used to indicate secondary or fine-print information, such as disclaimers, copyright notices, or footnotes.

```
JavaScript
<p>This is some <small>smaller text</small> within a paragraph.</p>
```



7. **<del> tag:**The <del> tag is a non-semantic tag used to indicate deleted or removed text within the content. It typically renders the enclosed text with a strikethrough line. This tag is useful for displaying edited or outdated content, highlighting revisions, or presenting alternative information.

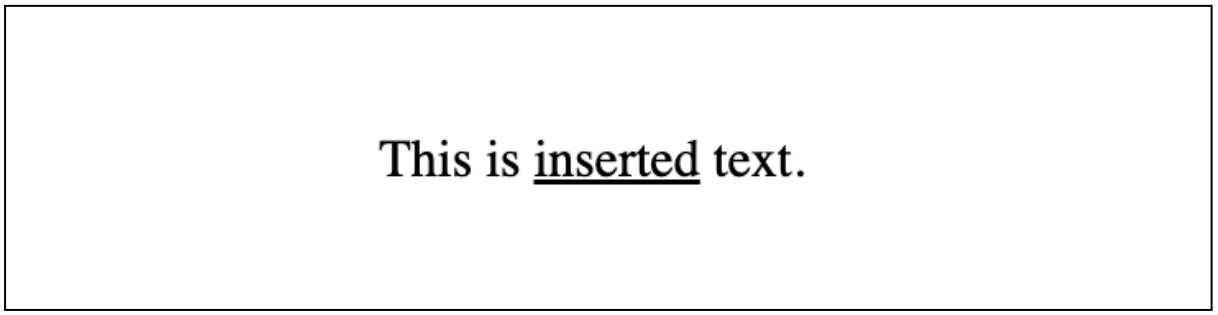
```
JavaScript
<p>This is <del>deleted</del> text.</p>
```



8. **<ins> tag:** The <ins> tag is a non-semantic tag used to indicate inserted or added text within the content. It typically underlines the enclosed text to visually distinguish it as new or recently added. This tag is often used in collaborative editing environments, version control systems, or when presenting changes or updates.

```
JavaScript
<p>This is <ins>inserted</ins> text.</p>
```

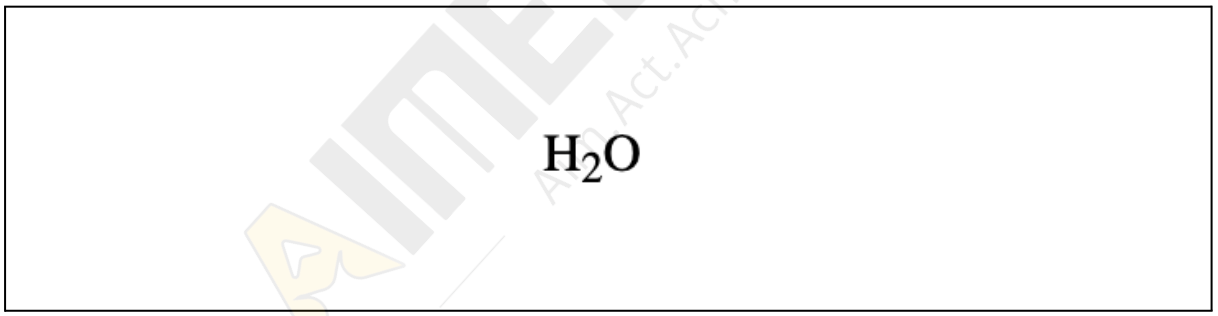
Output:



9. **Subscript(<sub>):** The <sub> tag is a non-semantic tag used to render text as subscript, positioning it slightly below the normal text baseline. It is commonly used for displaying chemical formulas, mathematical equations, or footnotes. It is particularly useful for denoting subscripts or smaller characters in mathematical or scientific notations.

```
JavaScript
<p>H<sub>2</sub>0</p>
```

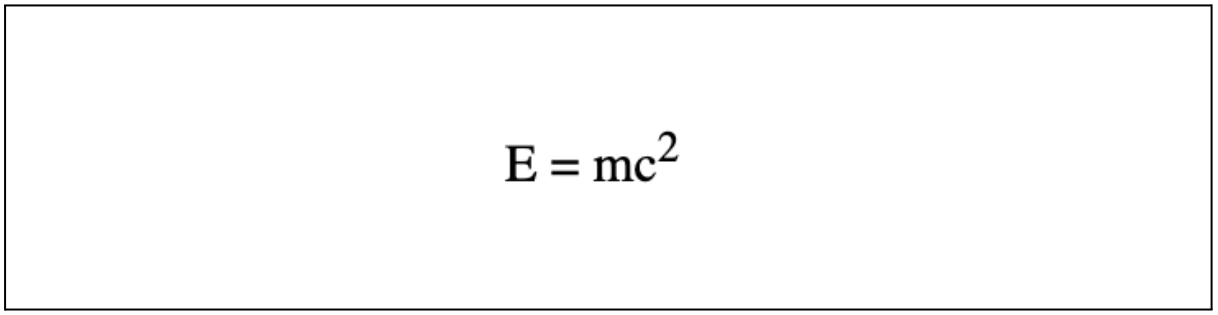
Output:



10. **Superscript(<sup>):** The <sup> tag is a non-semantic tag used to render text as superscript, positioning it slightly above the normal text baseline. It is often used for displaying exponents, mathematical notations, or footnotes. It is particularly useful for denoting superscripts or smaller characters in mathematical or scientific notations.

```
JavaScript
<p>E = mc<sup>2</sup></p>
```

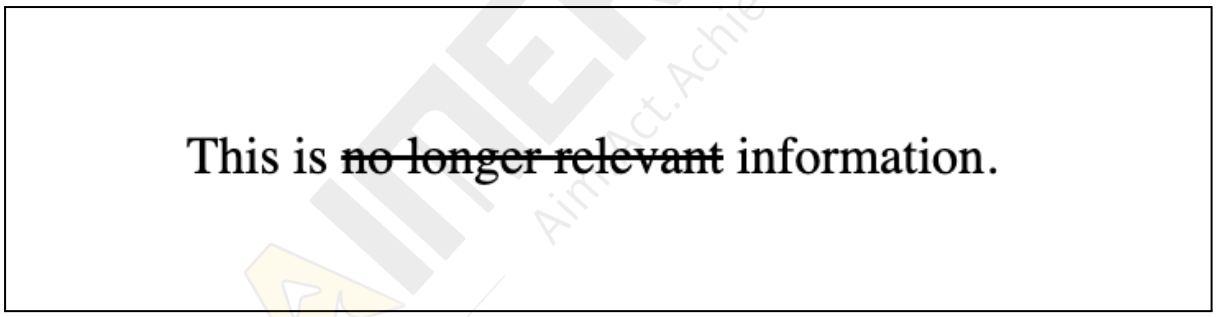
Output:



11. **<s> tag:** The <s> tag is a non-semantic tag used to render text with a strikethrough line, indicating that it is no longer valid, relevant, or accurate. It is typically used for displaying content that has been deprecated, crossed out, or should be ignored.

```
JavaScript
<p>This is <s>no longer relevant</s> information.</p>
```

Output:

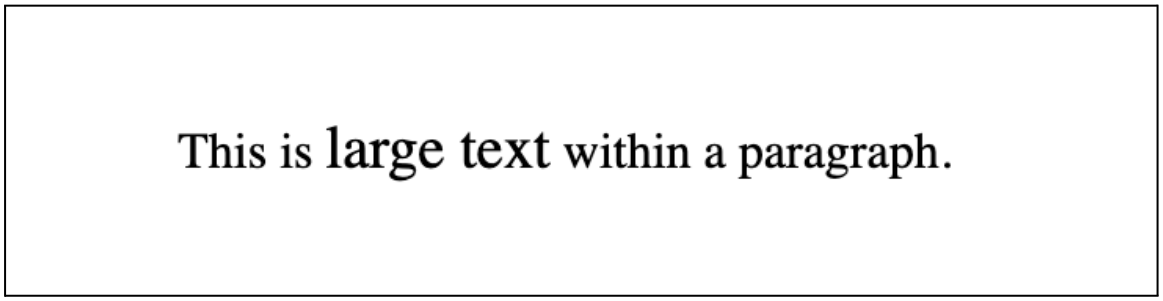


12. **<big> tag:**

The <big> tag, although part of earlier HTML versions, is no longer recommended for use as it has been deprecated. It was used to increase the font size of the enclosed text, making it larger than the surrounding text. Instead, it is recommended to use CSS (Cascading Style Sheets) to control font sizes and achieve the desired visual effects.

```
JavaScript
<p>This is <big>large text</big> within a paragraph.</p>
```

Output:



Inline Element vs Block Level Element

In HTML, elements are categorized as either inline or block elements, which determines how they behave and interact with other elements in terms of layout and formatting. There are also some vertical margins and other CSS differences that we will discuss in detail in the CSS module.

Let us see the most commonly used tags and whether they are inline or block level elements

Inline level element

- span
- a
- strong
- em
- img
- input

Block level element

- div
- main
- section
- article
- p
- All heading tags (h1-h6)
- ul
- li
- table

- form

The above list contains the tags that are most commonly used, there are many more tags apart from the ones mentioned above.

Now, the main question is **what is the difference between an inline or block element?**

Inline Elements: If an element takes up only the required horizontal space and lets another inline element sit next to it then it is an inline element

```javascript
<!DOCTYPE html>
<html lang="en">
<head>
‹meta charset="UTF-8" />
‹meta name="viewport" content="width=device-width,
initial-scale=1.0" />
<title>Document</title>
</head>
<body>
<span>First Span,</span>
<span>Second Span, </span>
<span> Third Span.</span>
</body>
</html>
```

In the example code above we can see we have 3 span elements placed inside the body with some content in them, if we run this in the browser we will see that the 3 of them are sitting next to each other. Which shows that span is indeed an inline element.

Output:

First Span, Second Span, Third Span.

Block Elements: This is the opposite of inline elements. Block-level elements take up the entire horizontal space even if it is not required. Let us see an example of a block element.

```
JavaScript
<! DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width,
initial-scale=1.0" />
<title>Document</title>
</head>
<body>
<div>First div, </div>
<div>Second div, </div>
<div>Third div.</div>
</body>
</html>
```

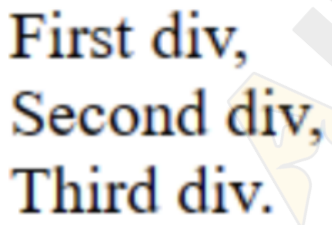
In the example code above we can see we have 3 div elements placed inside the body with some content in them, if we run this in the browser we will see that the 3 of them are displayed on a line each. With this, we can say div is a block-level element.

First div,
Second div,
Third div.

Some of you must be wondering what happens if I have an inline element and a block element next to each other.
Well let us take a look at what happens in that scenario.

```JavaScript
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width,
initial-scale=1.0" />
<title>Document</title>
</head>
<body>
<span>First Span, </span>
<div>Second div, </div>
<div>Third div.</div>
</body>
</html>
```

Output



So they look exactly like the previous example, which shows a block element will always sit in a new line.

Now are there any ways to make them sit inline? In short, yes we can change their behavior using the CSS display property which we will look at later in the course.

TIP: This is also a very important question that gets asked during the interviews

List

In HTML, a list is a structured way to present a collection of related items. Lists are commonly used to organize and display information in a structured and readable format. HTML provides several tags specifically designed for creating different types of lists.

The main types of lists in HTML are :

- ul
- ol
- dl

Let us now discuss each one of them and look at examples:

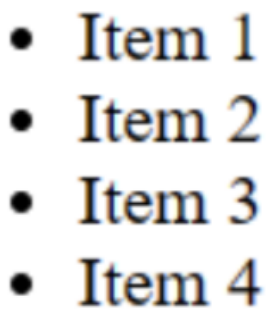
1. **ul:** It stands for an unordered list, which means every list item will have a bullet point in front of it.

JavaScript

```
<!DOCTYPE html>
<html lang="en" >
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta
      name="viewport"
      content="width=device-width,
initial-scale=1.0"
    />
    <title>unordered list</title>
  </head>
  <body>
    <ul>
      <li>Item 1</li>
      <li>Item 2</li>
```

```
      <li>Item 3</li>
      <li>Item 4</li>
    </ul>
  </body>
</html>
```

Output



You see every list item is displayed with a bullet point in front of it. This indicates it is an unordered list.

We can also nest the list as shown below

JavaScript

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <ul>
    <li>item</li>
    <li>item</li>
    <li>item</li>
```

```
    <ul>
      <li>Sub item</li>
      <li>Sub item</li>
      <ul>
        <li>Sub sub item</li>
        <li>Sub sub item</li>
      </ul>
    </ul>
  </ul>
</body>
</html>
```

Output:

- item
- item
- item
    - Sub item
    - Sub item
        - Sub sub item
        - Sub sub item

We have type attribute also available on the ul tag but it is not to be used anymore as it has been deprecated. We will learn how we can achieve this using CSS later in the course. But for reference let us look at them.

- **disc:** This is the default behavior as we have seen in the previous example
- **circle:** This is almost similar to disc but is unfilled

Ex:

JavaScript

```
<ul type="circle">
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
<li>Item 4</li>
```

```
</ul>
```

Output:

- Item 1
- Item 2
- Item 3
- Item 4

Square: This makes the shape of the bullet points square

```
JavaScript
<ul type="square">
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
<li>Item 4</li>
</ul>
```

Output

- Item 1
- Item 2
- Item 3
- Item 4

NOTE: There is also a triangle, but not all the browsers support it.

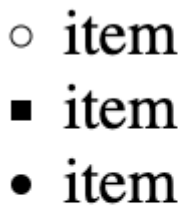
Similar to **ul tag the li tag** also supports type attribute with the following values

type = "circle"
type = "square"
type = "disc"

Example:

```
JavaScript
<ul>
<li type="circle">item‹/li>
<li type="square">item</li>
<li type="disc">item</li>
</ul>
```

Output:

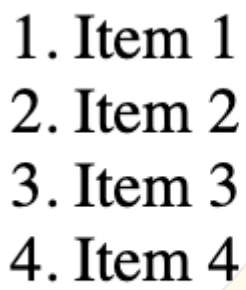


Note:
As these technologies evolved, it was decided that all styling should be handled by CSS, leading to the deprecation of certain attributes in both the li and ul tags. We'll cover how to achieve these effects using CSS later in the course

2. **Ol: It stands for ordered list, which means every list item will have a number in front of it.**

```
JavaScript
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0" />
    <title>Document</title>
  </head>

  <body>
    <ol>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
      <li>Item 4</li>
    </ol>
  </body>
</html>
```

Output:



We can also nest the list as below

```
JavaScript
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0" />
    <title>Document</title>
  </head>
```

```
  <body>
    <ol>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
      <li>Item 4</li>
      <ol>
        <li>Sub Item 1</li>
        <li>Sub Item 2</li>
        <ol>
          <li>Sub sub Item 1</li>
          <li>Sub sub Item 2</li>
        </ol>
      </ol>
    </ol>
  </body>
</html>
```

Output:

1. Item 1
2. Item 2
3. Item 3
4. Item 4
    1. Sub Item 1
    2. Sub Item 2
        1. Sub sub Item 1
        2. Sub sub Item 2

We also have a start attribute which we can use to define the start number

JavaScript

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0" />
    <title>Document</title>
```

```
  </head>
  <body>
    <ol start="3">
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
      <li>Item 4</li>
    </ol>
  </body>
</html>
```

Output:

3. Item 1
4. Item 2
5. Item 3
6. Item 4

There is another handy attribute called type, which lets us change the numbering style of the items list.

JavaScript

```
type = "1"
type = "A"
type = "a"
type = "I"
type = "i"
type = “1” is the default one
```

type = “A” : This will change the numbers to capital A,B,C,D

A. Item 1
B. Item 2
C. Item 3
D. Item 4

type = “a” : This will change the numbers to capital a,b,c,d

type="I": This will change the numbers to capital roman numbers

I. Item 1
II. Item 2
III. Item 3
IV. Item 4

type="i": This will change the numbers to capital roman numbers

i. Item 1
ii. Item 2
iii. Item 3
iv. Item 4

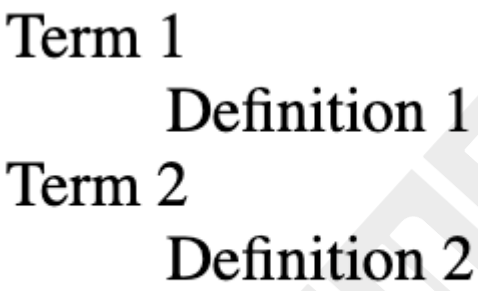
3. **dl: It stands for Definition List. It represents a list of terms and their corresponding definitions.**

   definition list consists of <dt> (Definition Term) and <dd> (Definition Description) pairs.

```
JavaScript
<! DOCTYPE html>
<html lang="en">
  <head>
```

```
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <dl>
      <dt>Term 1</dt>
      <dd>Definition 1</dd>
      <dt>Term 2</dt>
      <dd>Definition 2</dd>
    </dl>
  </body>
</html>
```

Output:



Note : Now, you must be thinking, is there a way I can change the ul bullet list from circles to something else? In short, yes, we can change them to something different with the help of CSS(We will learn this later in the course). There is a way to change them in HTML using type attributes but it is not preferred

Anchor tag

An anchor tag in HTML is represented by the <a> element and is used to create hyperlinks within a web page. It allows you to link to other web pages, files, sections within the same page, email addresses, and more. The anchor tag is one of the fundamental elements for navigation and linking in HTML.

Syntax:

```
JavaScript
<a href= "URL">Link Text</>
```

The most commonly used attributes with the anchor tag are

- href
- target
- rel

1. **href(Hypertext reference):**
   It specifies the destination URL or target of the link

```
JavaScript
<a href="https://aimerz.ai">Visit Aimerz Website</a>
```

With this a link will be created on the text Visit Aimerz website and on clicking on this text the user will be redirected to **aimerz.ai** in the same tab.

2. **target:**
   It specifies where the linked document should be displayed.

- **_self:** This is the default value if we do not specify any target attribute. This means open the link in the same window.

Ex:

```
JavaScript
<a href="https://aimerz.ai" target="_self">Visit Aimerz Website</a>
```

- **_blank:** If we use _blank this means open the link in a new tab.

Ex:

```
JavaScript
<a href="https://aimerz.ai" target="_blank">Visit Aimerz
Website</a>
```

- **_parent:** The parent browsing context of the current one. If no parent behaves as _self.

Ex:

```
JavaScript
<iframe
  width="300"
  height="100"
  style="border: 3px dashed #808cf8; padding: 30px 0 0 30px"
  srcdoc="<a target='_parent' href='https://wikipedia.com'>Wikipedia</a>"
>
</iframe>
```

In the example above, we have an iframe and a target **_parent** attribute, when you click on WikiPedia the link will open outside the frame, if there was no _parent then it would open inside the frame.

- **_top:** the topmost browsing context (the "highest" context that's an ancestor of the current one). If there are no ancestors, it behaves as _self.

```
JavaScript
<iframe
  width="300"
  height="100"
  style="border: 3px dashed #808cf8; padding: 30px 0 0 30px"
  srcdoc="<a target='_top' href='https://wikipedia.com'>Wikipedia</a>"
>
</iframe>
```

This unlike the _parent will move to the outermost section which is the browser and replace the page.

Example, if we are in an online editor where the left hand side has the editor and the right hand side has output preview, using _top will replace the entire page with the destination URL.

This is the biggest difference between **_parent and _top,**
_parent will replace the parent whereas **_top** will replace it from the browser window.

- **target="framename"** attribute value on an <a> tag.
  Clicking the link will open the linked page in the specified <frame>

Ex:

```
JavaScript
<p>
Click to learn more about
<a target="display-frame"
href="https://simple.wikipedia.org/wiki/JavaScript">Javscript Wikipedia
</a>.
< /p>
<iframe name="display-frame"
style="width:100%;height:600px;border:2px solid
#446e5; "></iframe>
```

3. rel

Specifies the relationship between the linked document and the current document. It provides additional information about the link.
Common values include nofollow, noopener, noreferrer, stylesheet, icon, and custom values.

- **rel=”nofollow”:**

Indicates to search engines that they should not follow the link or pass any ranking authority to the linked page.

```
JavaScript
<a href="https://aimerz.ai" rel="nofollow">Visit Aimerz website</a>
```

- **rel="noopener" (or) rel="noreferrer":**

These values are used to enhance security when opening links in a new tab or window, preventing the newly opened page from accessing the window.opener object of the originating page.

When we use target=" _blank" the default behavior for rel is noreferrer

```
JavaScript
<a href="https://aimerz.ai" rel="_blank">Visit Aimerz website</a>
```

- **rel="stylesheet"**

Specifies that the linked document is a style sheet used to define the visual presentation of the current document.
This is one of the most common and important ones out there that you will see yourself using whenever you have an external stylesheet. Mostly you will see this in your head tag.

```
JavaScript
<link href="style.css" rel="stylesheet"
```

- **rel="icon":**

Specifies that the linked document is an icon or image to be used as the website's favicon (a small icon displayed in the browser's tab or bookmark).

```
JavaScript
<link rel="icon" href="favicon.ico" type="image/x-icon" />
```

HTML Classes & ID

HTML classes and IDs are attributes used to identify or group elements within an HTML document uniquely.
They provide a way to target and style specific elements or apply JavaScript functionality. Here's an overview of HTML classes and IDs:

HTML Classes

- The class attribute assigns one or more class names to an HTML element.
- Classes allow multiple elements to share the same styling or behavior.
- Multiple classes can be assigned to an element by separating them with spaces.
- Classes are defined in CSS stylesheets to apply specific styles to the elements with those class names.

Ex:1

```
JavaScript
<div class="container">
<h1 class="heading">Aimerz!</h1>
<p class="content">Pure Hardwork, No Shortcuts! </p>
</ div>
```

Ex-2:

```
JavaScript
<div class="container">
<h1 class="heading">Aimerz</h1>
<span class="content">Pure Hardwork</span>
<span class="content">, No Shortcuts!</span>
</div>
```

The example code above shows how to add class names to the HTML tags.

NOTE: We can write any class name but it is better to write meaningful class names that represent the use case of that class.

Notably, we can give any number of classes to an element and the order of the classes basically does not matter.

HTML ID

- The id attribute is used to give a unique identifier to an HTML element.
- Each ID must be unique within the HTML document.
- IDs are primarily used for targeting specific elements with CSS or JavaScript.
- IDs are often used when you want to apply specific styling or perform targeted actions on a single element.

```
JavaScript
<div id="container">
<h1 id="heading">Aimerz!</h1>
<p id="content">Pure Hardwork, No Shortcuts!</p>
</div>
```

Similar to classes, we can give any number of IDs to an element and the order of the IDs basically does not matter but make sure they are unique.

```javascript
<div id="container">
<h1 id="heading">Aimerz</h1>
<p id="content tagline">Pure Hardwork, No Shortcuts!</p>
</div>
```

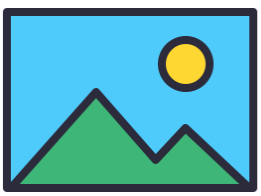
Tips

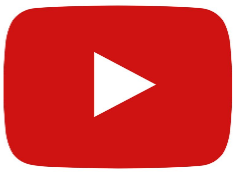
- Use ID for uniquely targeting the element via Javascript, or giving the element some unique style, since ID must not be reused.
- Use classes when you want to give a generic style to elements since classes can be reused.

HTML Media

Introduction to HTML Media

In this topic, we will study how to insert an image, audio, or video in an HTML document. We will also learn how to embed another document (e.g. YouTube video, Google Maps, another website) into our web page. At last, we will explore more media options like Canvas and SVG.





Image

Image tag (<img/>)

If we want to include an image on our web page, we will use the image tag **<img/>.**

Syntax

```
JavaScript
<img src="image.jpg" alt="img tag syntax"/>
```

Here, the **src** attribute can be a path to an image or URL of an image, and the **alt** attribute is used to provide alternative text for an image

Ex:

```
JavaScript
<body>
    <h1>Heading </h1>
    <p>This is a random photo</p>
    <img src="https://picsum.photos/200" alt="image" />
</body>
```

Output:

Suppose, we have given the wrong path to the src attribute, the image will not be displayed we call it a broken image.

Example: Broken image with alt attribute

```
JavaScript
<body>
    <h1>Heading</h1>
    <p>This is a random photo</p>
    <img src="broken-image.jpg" alt="image alt text" />
</body>
```

Output:

Yes!! It will show the **alt** text that we provided in the **<img>** tag. It is important to include **alt** text for images because it provides information to users who may be unable to see the image due to a slow internet connection, or a broken image link.

Image tag attributes

1. **src**: The src attribute is an HTML attribute used to specify the URL of an image file to be displayed on a web page.

2. **alt**: The alt attribute is an HTML attribute used to provide alternative text for an image in case the image cannot be displayed.

3. **width and height:** These attributes specify the width and height of the image in pixels.

```
JavaScript
<img src="image.jpg" width="200" height="150" alt="dog image"/>
```

The above image has 200px width and 150px height.

Note: Initially, attributes like width & height were preferred to use, but later when CSS came, styles were separated from HTML for better code clarity and separation of concerns. We will study more about CSS in later modules.

4. **loading:** The loading attribute is an HTML attribute used to control the loading behavior of an image on a web page. The attribute has two possible values:
    - **lazy** which defers the loading, and loads only when the image is needed.
    - **eager** which loads the image immediately. This is the default behavior.

Supported Image Formats

The **<img> tag** in HTML can display various types of image files. The most commonly supported image formats are:

- .jpeg
- .png
- .svg
- .gif
- .webp

Picture tag <picture>

The <picture> element holds one <img> element and zero or more <source> elements. The browser checks every <source> and selects the most compatible one, if it does not find a suitable match, it will select the URL in the src attribute in <img> tag and display it.

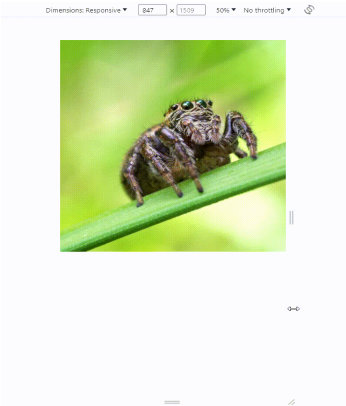
Browse selects one source on the basis of srcset, media, and type attribute of the <source> tags. srcset defines image URL, media defines media query(will study in later modules), and type defines media format(like mp3, mp4, etc.) to understand it better let's see one example,

Example

JavaScript

```
      <source
srcset="https://cdn.pixabay.com/photo/2023/07/30/11/30/spider-8158656_1280.j
pg"
        media="(min-width: 700px)"
      />
      <source
srcset="https://cdn.pixabay.com/photo/2023/07/04/10/30/dragonfly-8105988_128
0.jpg"
        media="(min-width: 450px)"
      />
      <img
src="https://cdn.pixabay.com/photo/2023/08/06/07/19/field-8172323_1280.jpg"
        alt="photos"
      />
</picture>
```

Output:



You can see in the above example, the browser displays different images at different screen sizes, because it selects the most compatible one.

Audio

Audio tag <audio/>

The Audio tag includes audio on our web page. **<audio/>** tag is introduced in HTML5.

Syntax

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

Here, the **src** attribute can be a URL of an audio file, and the **controls** attribute enables play, pause, and volume controls.

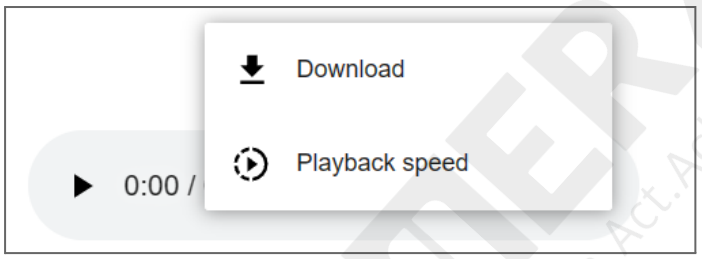
Example: Embed an audio

```
JavaScript
<body>
  <audio
```

```
        controls
src="https://interactive-examples.mdn.mozilla.net/media/c
c0-audio/t-rex-roar.mp3">
  </audio>
</body>
```

Output:

If you click on 3 dots, it will show options to download and set playback speed.



Audio tag Attributes

1. **src:** This attribute specifies the source URL (Uniform Resource Locator) of the audio file that should be played.

2. **controls:** When this attribute is present, it displays the default audio controls, such as play, pause, and volume, allowing users to interact with the audio player.

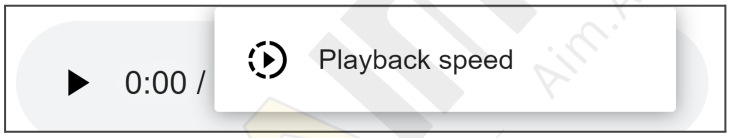
3. **autoplay:** When this attribute is present, the audio will start playing automatically when the page loads.

4. **loop:** The audio will loop and play repeatedly when this attribute is present.

5. **preload:** This attribute specifies how the audio file should be preloaded for a better user experience. It can have values like,
    - **auto** - loads the audio file automatically, even if the user does not require it immediately.
    - **metadata** - loads only the metadata of the audio file (e.g. length of the audio file)
    - **none** - does not preload the audio file

6. **controlslist:** This attribute specifies the controls that should be displayed in the audio player's controls. It can have values like
    - **nodownload -** disable download button
    - **nofullscreen** - disable fullscreen button
    - **nodownload nofullscreen** - disable both download and fullscreen buttons.

```
JavaScript
<audio src="audio.mp3" controls controlsList="nodownload"></audio>
```



7. **muted:** When this attribute is present, the audio will be muted by default.

```
JavaScript
<audio src="audio.mp3" muted></audio>
```

Audio with multiple sources

Not all browsers or devices support the same file format, and using multiple **<source>** tags allows the browser to choose the best-suited file format that it can play.

To specify multiple sources, we use **<source>** tags within **<audio>** tag. Each source tag has a **type** attribute, which identifies the format of the audio file.

Ex:

```javascript
<audio controls>
 <source src="audio.opus" type="audio/ogg; codecs=opus" />
 <source src="audio.ogg" type="audio/ogg; codecs=vorbis" />
 <source src="audio.mp3" type="audio/mpeg" />
</audio>
```

The browser will try to play the first file format specified in the **<source> tag**, and if it cannot play it, it will try the next one until it finds a compatible file format.

Video

Video tag <video/>

The **<video>** Tag embeds a video player into our webpage. **<video>** tag is introduced in HTML5.

Syntax

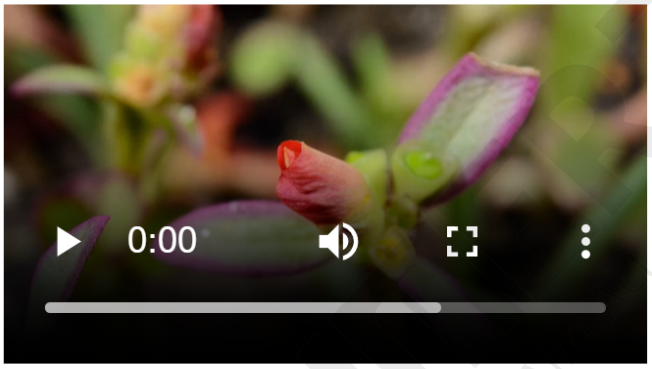
```javascript
<video src="video.mp4" controls"></video>
```

Here, the **src** attribute can be a URL of a video file, and the **controls** attribute enables play, pause, and volume controls.

Example: Embed a video

```
JavaScript
<body>
    <video
src="https://interactive-examples.mdn.mozilla.net/media/cc0-videos/flower.webm" controls width="250">
    </video>
</body>
```

Output:



Video Tag attributes

1. **src:** This attribute specifies the source URL (Uniform Resource Locator) of the video file that should be played.

2. **controls:** When this attribute is present, it displays the default video controls, such as play, pause, and seek, allowing users to interact with the video player.

3. **autoplay:** When this attribute is present, the video will start playing automatically when the page loads.

4. **loop:** The video will loop and play repeatedly when this attribute is present. For example;

5. **preload:** This attribute specifies how the video file should be preloaded. It can have values like,
   - **auto** - loads the video file automatically, even if the user does not require it immediately.
   - **metadata** - loads only the metadata (e.g. length) of the video file
   - **none** - does not preload the video file.

6. **width and height:** These attributes specify the width and height of the video player in pixels.

```
JavaScript
<video src="video.mp4" width="640" height="360"></video>
```

Note: Initially, attributes like width & height were preferred to use, but later when CSS came, styles were separated from HTML for better code clarity and separation of concerns. We will study more about CSS in later modules.

7. poster: This attribute specifies an image URL that should be displayed as a poster frame before the video starts playing.

```
JavaScript
<video src="video.mp4" poster="movie-poster.jpg" width="640" height="360"
controls></video>
```

<video src="video.mp4" poster="movie-poster.jpg" width="640" height="360" controls></video>

8. controlsList: This attribute specifies the controls that should be displayed in the video player's controls. It can have values like,

- **nodownload** - disables download button.
- **nofullscreen** - disables the fullscreen button.
- **nodownload nofullscreen** - disables both download and fullscreen buttons.

9. muted: When this attribute is present, the video will be muted by default.

Video with multiple sources

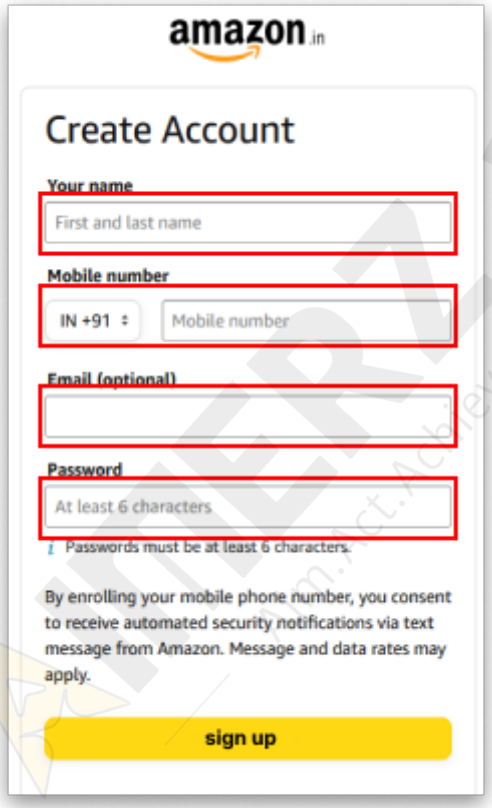
Just like the **<audio> tag**, the **<video> tag** can use the **<source> tag** to provide alternative file formats to ensure that the video can be played on a wide range of devices and web browsers.

Ex:

```
JavaScript
<video width="620" controls>
     <source src="video.ogv" type="video/ogg" />
     <source src="video.avi" type="video/avi" />
     <source src="video.mp4" type="video/mp4" />
     Sorry, your browser doesn't support embedded videos.
</video>
```

Introduction to HTML form

An HTML form is like a virtual questionnaire that you find on websites. It allows you to interact with a website by filling in information and submitting it.



Imagine you're signing up for a new website account. The website asks you for your name, phone number, email, and password. You provide this information by typing it into boxes on the website. These boxes are called "**input fields.**" Once you've filled out everything, you click a button that says **"Sign Up"** or **"Submit."**

When you click that button, the website takes all the information you provided in the input fields and sends it to the website's server, kind of like sending a letter. The server then processes that information and does whatever the website needs it to do, like creating your account with the details you provided.

Basic form code example

To declare a form in HTML, you use the **<form>** element. Inside the **<form>** element, you place various form elements to collect data from users. Here's the basic syntax for declaring a form and placing form elements inside it:

```javascript
<h1>Contact Form</h1>
<form action="/submit_form" method="post">
    <!-- Form elements go here -->
</form>
```

Form attributes

Behind the scenes, HTML forms use attributes to control the behaviour and how the data is processed. Let's understand these attributes using our Amazon sign-up example:

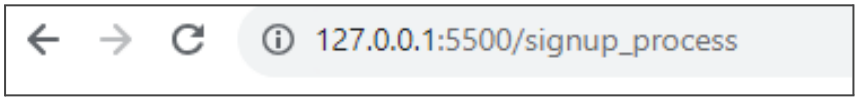
- **action**
  The action attribute is like **specifying the address** where you **want to send your form**. It tells the browser where to submit the form data. The action attribute might look like this: **action="/signup_process"**. This means the form data will be sent to the **"/signup_process"** URL on some website server.

Ex:

```
JavaScript
<form action="/signup_process" method="post">
   <input type="submit" />
</form>
```

Output:

As we click on “submit” button, you will see action url on browser url,



- **method**
  The method attribute determines how the form data is sent to the server.

Syntax:

```
JavaScript
<form method="get"></form>
<form method="post"></form>
```

The most common methods are **"GET"** and **"POST".** In our Amazon sign-up form, you can use the **method="POST"** attribute, which means the data is sent securely and privately.

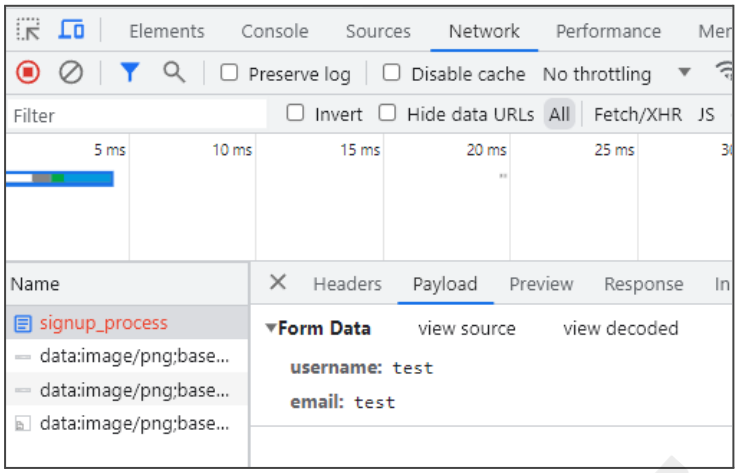
Note: *In the case of the GET method, after submitting the form, form data is visible in the address bar. But the POST method prevents form data from appearing in the address bar after the form has been submitted, via sending data in the request body.*

Example:

```
JavaScript
<form action="/signup_process" method="get">
    <input name="email" />
    <input name="username" />
    <input type="submit" />
</form>
```

As we click on “submit” button, you will see action url on browser url, you will notice browser url changes and all form values are visible in url,

In case of method=”post”, form values will be sent to server as part of body(payload), it can be visulised in network tab of browser dev tool as shown below,



- **target**

  The action attribute accepts different values, each influencing how the form response is handled.

  The target attribute can have one of the following values:

  - **_self:** This is the default value if the “target” attribute is not specified. It means that the form response will be displayed in the same window or tab where the form was submitted.
  - **_blank:** Using **target=”_blank”** opens the form submission response in a new browser window,
  - **_parent**: Using frames or iframes, **target=”_parent”** Opens the form submission response in the parent form.
  - **_top:** If the form is within a nested set of frames or iframes, **target=”_top”** Cause the form response to open in the top-level browsing context, replacing any frames. If the form is not within a frame or iframe, it behaves the same as **_self.**

Syntax

```
JavaScript
<form target="_self"></form>
<form target="_blank"></form>
<form target="_parent"></form>
<form target="_top"></form>
```

Example: In the previous example, added target="_blank"

```
JavaScript
<form action="/signup_process" method="post" target="_blank">
    <input name="username" />
    <input name="email" />
    <input type="submit" />
</form>
```

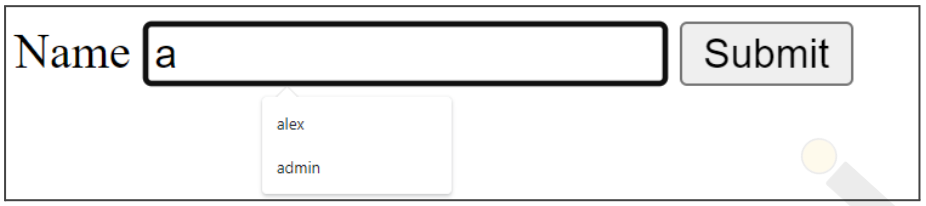
- **autocomplete**
  Controls whether the browser should remember and autofill form input values. Values can be **"on"** (default) or **"off"**

Ex:

```
JavaScript
<form action="/signup_process" method="post" >
```

```
    <label>Name</label>
    <input name="username" autocomplete="on" />
    <input type="submit" />
</form>
```

Output:



As you see above, browser is suggesting text based on input history.

- **enctype**
  specifies how the form data should be encoded before sending it to the server. There are three common enctype values:
  - **application/x-www-form-urlencoded** (Default): It encodes form data in a URL-like format, suitable for regular form submissions with text input fields.
  - **multipart/form-data**: Used for forms with file uploads. It properly handles binary data, like images or documents.
  - **text/plain**: Rarely used, sends form data as plain text without special formatting.

Syntax:

JavaScript

```
<form enctype="application/x-www-form-urlencoded"></form>
<form enctype="multipart/form-data"></form>
<form enctype="text/plain"></form>
```

Choose the appropriate enctype based on your form content. For regular forms, you can stick to the default. For file uploads, use **multipart/form-data**.

Example:

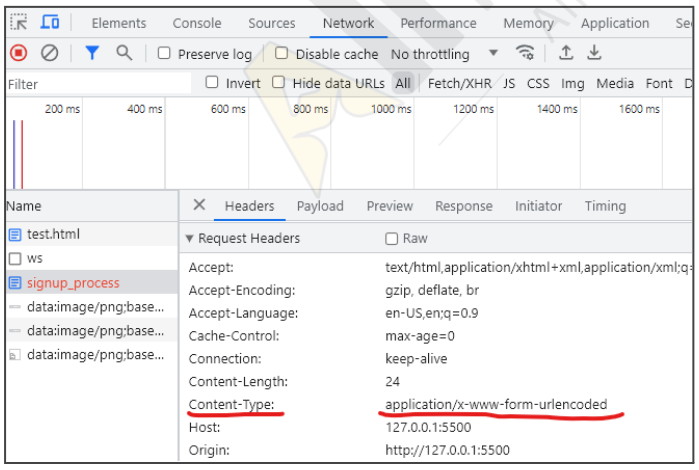
```
JavaScript
<form action="/signup_process" method="post" enctype="multipart/form-data">
      <input name="filename" />
      <input name="file" />
      <input type="submit" />
</form>
```

File Name

Choose File No file chosen

Submit

Upon Submitting the form, you can see enctype as Content-Type in Reaquest Headers in Netowork tab of browser dev tool.



- **name**

  Name of the form, it must a unique non-emply string.

Syntax

```
JavaScript
<form name="signup"></form>
```

- **novalidate**

  A boolean attribute, which tells the browser to not validate field values. We will study about validation in next lesson.

Syntax

```
JavaScript
<form novalidate></form>
```

- **rel**

  It defines the relationship between current document and the linked document. Its values could be,

  - **external:** Indicates that the referenced document exists outside the scope of the current website.
  - **help:** Provides a link to a document designed to offer assistance or guidance.
  - **license:** Directs to copyright-related information associated with the document.
  - **next:** Refers to the subsequent document within a series or selection.
  - **nofollow:** Links to a document without an endorsement, often used for paid links.
  - **noreferrer:** Instructs the browser to not send an HTTP referrer header.

- **nopener:** Advises the browser to open the link in a way that prevents the new page from controlling the originating page.
- **prev:** Points to the preceding document in a series or selection.
- **search:** Connects to a search tool designed for exploring the document.

```JavaScript
<form rel="noreferrer nofollow" action="">
   Username <input type="text" /><br /><br />
   Password<input type="password" /><br />
   <input type="submit" value="submit" />
</form>
```

In above example, rel=”noreferrer nofollow nopener”, protects your site users from potentially malicious external links.

Form elements

Form elements in HTML are used to create interactive sections where users can input data or make selections. Forms are a fundamental part of web development, allowing users to submit information to a server for processing. Here are some elements that form contains:

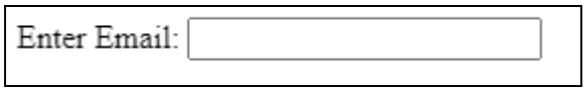
1. Input and label Tag

The **<input>** HTML element is one of the most important form elements which is used to accept data/input from the user.

A wide variety of types of input data and control widgets are available.

A simple example,

```javascript
<label for="email">Enter Email: </label>
<input type="email" name="email" id="email" required>
```



Above code defines an **email input** for entering an email address. It also checks for a valid email address.

Here <label> tags help users to know the purpose of the form control element (input), and for attribute specifies the target control element, which allows users to click on the label to focus on the associated form control.

2. Fieldset and Legend Tags

The **<fieldset>** tag is used to group related form elements together. It will give a border to the grouped elements.

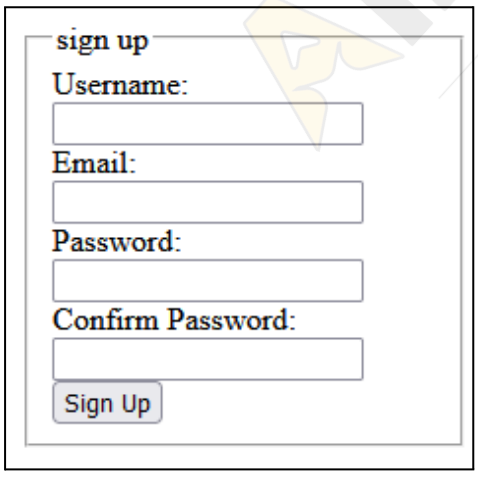
The **<legend> tag** is **used within a <fieldset> to provide** a **caption or title** for the fieldset. It is often used to group related form controls together and provide context or instructions for the user. The text within the **<legend> tag** is typically styled to stand out from the rest of the form controls, making it easier for the user to identify the purpose of the group of controls.

Let's take the **form submission** example, instead of <h1> add fieldset and legend.

index.html

```
JavaScript
<body>
    <fieldset>
        <legend>sign up</legend>
        <form action="/signup_process" method="post" name="sign-up>
        <label for=" username">Username:</label>
            <input type="text" id="username" name="username" required><br>

            <label for="email">Email:</label>
            <input type="email" id="email" name="email" required><br>

            <label for="password">Password:</label>
            <input type="password" id="password" name="password"
required><br>

            <label for="confirm_password">Confirm Password:</label>
            <input type="password" id="confirm_password"
name="confirm_password" required><br>

            <input type="submit" value="Sign Up">
    </fieldset>
    </form>
</body>
```

Output:



3. Button Tag

It offers submitting a form or triggering specific actions, like resetting the form fields.

Syntax

```
JavaScript
<button type="">Button Text<button/>
```

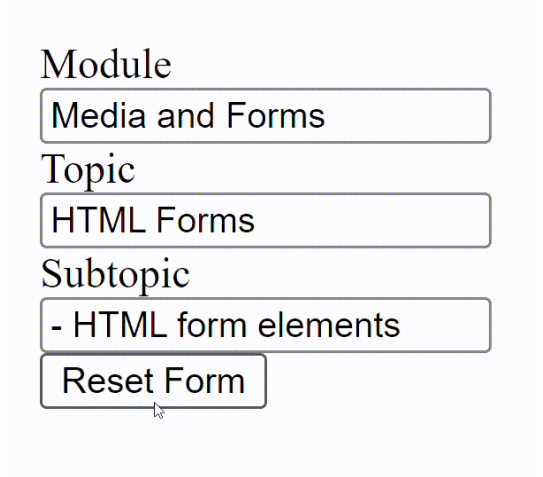
Here, type attribute can be either submit, reset, or button, depending on use cases

- **submit:** When this value is used, the button initiates the submission of form data to the server.
- **reset:** Selecting this value causes the button to reset all associated input fields to their original values.
- **button:** Opting for this value means, button will not have predefined behavious, like reset or submit.

Example: Reset form,

```
JavaScript

<form>
    <label for="module">Module</label><br />
    <input type="text" id="module" name="module" required /><br />

    <label for="topic">Topic</label><br />
   <input type="text" id="topic" name="topic" required /><br />

   <label for="subtopic">Subtopic</label><br />
   <input type="text" id="subtopic" name="subtopic" required /><br />

   <!-- Reset Button -->
   <button type="reset">Reset Form</button>
</form>
```

Output:



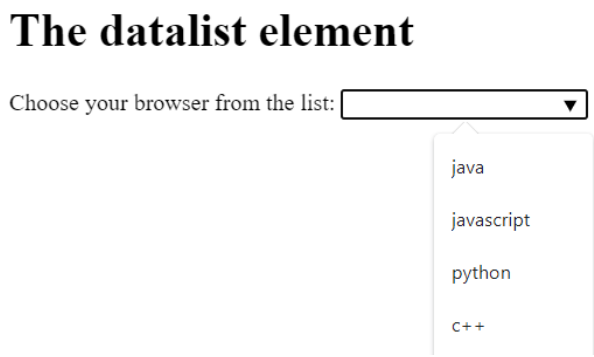
As you see above, clicking a button with type “reset”, sets all fields to its original value.

4. Datalist Tag

The **<datalist>** tag is used to provide an **autocomplete** feature for **<input>** elements (We will study about input tags in detail in next lesson). Users will see a drop-down list of predefined options as they input data.

The <datalist> element's **id** attribute must be equal to the **<input>** element's list attribute (this binds them together).

```JavaScript
<h1>The datalist element</h1>
<label for="language">Choose your browser from the list:</label>
<input list="languages" name="language" id="language" />
    <datalist id="languages">
      <option value="java"> </option>
      <option value="javascript"> </option>
      <option value="python"> </option>
      <option value="c++"> </option>
</datalist>
```



5. Textarea Tag

The <textarea> element is used to create a multi-line text input field that allows users to enter and edit text. The <textarea>

element can be useful when you want users to provide longer comments, descriptions, or other types of textual input.

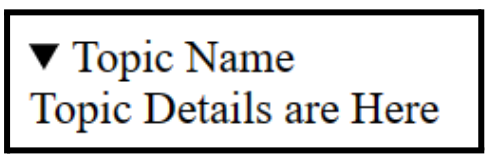
6. Details Tag

The **<details>** HTML element generates a disclosure widget that displays information only when the widget is switched to the **open** state.

```javascript
<details>
    <summary>Topic Name</summary>
    Topic Details Here
</details>
```

Close state

Open State

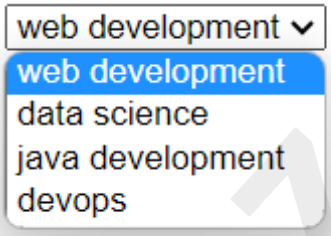


NOTE: All the input elements types with attributes will be discussed in the next section in detail.

7. Select and Option

In HTML, the <select> element is used to create a dropdown list, and the <option> element is used to define the individual options within that dropdown list. This Allows users to select one option from a dropdown of multiple options.

Let's understand with an example of Dropdown



In the above example, it allows users to select one option from the dropdown options.

Input Tag

The <input> HTML element is one of the most important form elements which is used to accept data/input from the user.

A wide variety of types of input data and control widgets are available

```JavaScript
<label for="email">Enter Email: </label>
<input type="email" name="email" id="email"
required>
```

Output:

Above code defines an email input for entering an email address. It also checks for a valid email address.

Here < label > tags help users to know the purpose of the form control element (input), and for attribute specifies the target control element, which allows users to click on the label to focus on the associated form control.

Input Attributes

- **type:** This attribute specifies the type of input control to be created. It is a required attribute and can take various values such as **text, password,email,number,checkbox, radio, submit** and many more.
- **name:** It's used to give a name to the information that a user enters into that field. This name is important because it helps developers and servers understand what the user has typed in. When you submit the form, The value of this attribute is sent to the server. The server uses these names to figure out which data corresponds to which input field. It's like labelling different jars in your kitchen so you can easily find what you need.

Let's look at an example to understand the significance of name attribute:

```
JavaScript
<h2>Contact Information</h2>
<form>
<label for="name" >Name: </label>
<input type="text" id="name" name="name" /><br /><br />
<label for="email" >Email:</label>
```

```
<input type="email" id="email" name="email" /><br /><br />
<label for="phone">Phone: </label>
<input type="tel" id="phone" name="phone" /><br /><br />
<input type="submit" value="Submit" />
</form»
```

Above example shows a form which takes name, email and phone number as input.

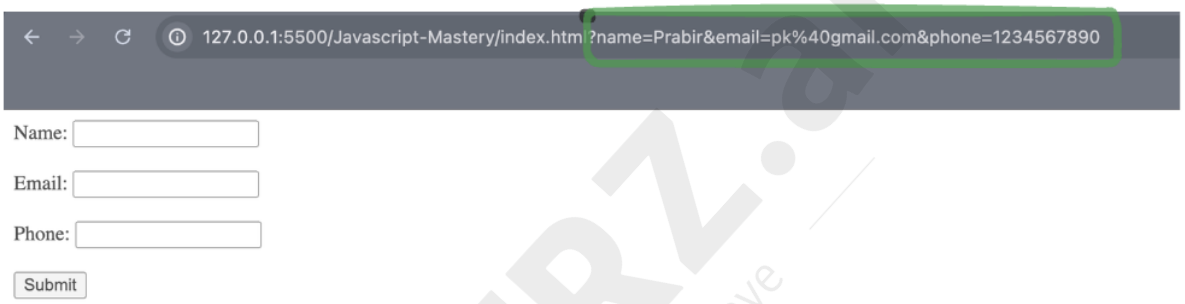
Let's understand the importance of the "name" attribute here.
Below image shows when i filled the form and submitted.

Name: Prabir

Email: pk@gmail.com

Phone: 1234567890

Submit



- **value:**
  This attribute specifies the default value of the input control. It is used to pre-fill the input field with a value that the user can modify.

- **placeholder:**
  This attribute specifies a short hint or example text that is displayed in the input field to provide guidance to the user on what should be entered

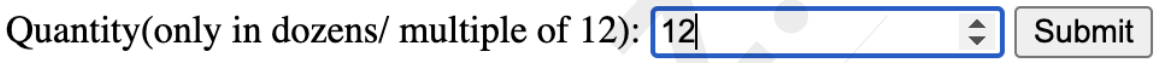
- step:
  The step attribute is used with the < input> element of type number or datetime-local to specify the increment value for the input field. It defines how much the value of the input should increase or decrease.
  when the user interacts with the associated input control, such as clicking the up/down arrows or using keyboard input. Step

attribute works with number, range, date, datetime-local, month, time and week
<input> types

Example:-

```
JavaScript
<form>
  <label for="quantity">Quantity(only in dozens/ multiple of 12):</label>
  <input type="number" id="quantity" name="quantity" step="12" />
  <input type="submit" />
</form>
```

Output:



The above example shows that, suppose you are creating an online-store selling some items online and it comes in a pack of 12, so then you can use this step attribute and add the step value as 12 so that the user can enter only multiple of 12 as the input.

- autocomplete:
  The autocomplete attribute is used in HTML forms to control whether a browser should automatically complete input fields. The browser should display options to fill in the field, based on earlier typed values. This
  attribute can be applied to <input> elements of type text or number, <textarea> elements, <select> elements,
  and <form> elements. By default the value is on. We can turn the auto-complete suggestions on a form by keeping the value as off (autocomplete="off")

Ex:

```JavaScript
<form autocomplete="on">
  <label for="name">Name: </label>
  <input type="text" id="name" name="name" /><br /><br />
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" autocomplete="off" /><br /><br
/>
  <input type="submit" />
</form>
```

First fill the form and submit the form.
Now let's try to fill the form again

Now you can see that it's giving suggestions in the name field but since we added autocomplete="off" in the
email input, it won't give any suggestions there.

Boolean attributes

A boolean attribute means that the presence of that attribute on an element represents the "true" value, and the absence of the attribute represents the "false" value.

These boolean attributes are an essential part of HTML and are used to control the behavior and properties of various elements without requiring additional attribute values. Their presence or absence determines whether a particular behavior or state is enabled or disabled, providing flexibility and simplicity in HTMLmarkup. Kindly observe in attributes below how they do not need any value.

This design principle of boolean attributes serves two primary purposes

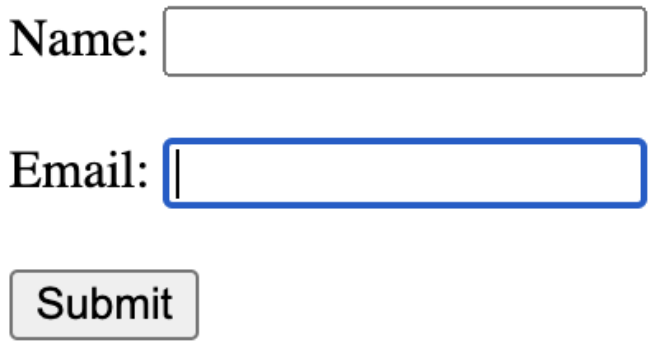
Simplicity: Boolean attributes make the HTML markup more concise and readable. They eliminate the need for explicit values, reducing clutter in the code and making it easier for developers to understand and maintain.

Clarity: Boolean attributes enhance the clarity of HTML documents by clearly indicating the state of an element or its associated behavior. Their presence provides a straightforward visual cue that a particular feature is active, while their absence communicates the opposite.

- autofocus:
  The autofocus attribute is an HTML attribute that can be applied to specify that the input field should receive focus automatically when the web page loads. This means that as soon as the page is loaded, the cursor will be placed inside the input field, allowing the user to start typing or interacting with it without needing to click on it first.

Ex:

```javascript
<form>
  <label for="username">Username: </label>
  <input type="text" id="username" name="username" autofocus />
</form>
```



As you can see in the above example, the input element getting focused once the page is loaded and we can see that the cursor is blinking which indicates that we can start typing

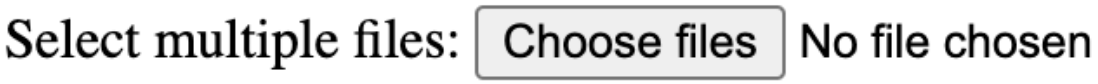
- multiple:

The multiple attribute is used with the ‹ input › element when creating file upload fields "‹ input type="file"›" to allow users to select multiple files for upload in a single input field. This attribute is particularly useful when you want to enable users to upload multiple files simultaneously.

Multiple also works with ‹ input › elements of type email to allow users to enter multiple email addresses in a
single input field. When the multiple attribute is applied to an email input field, it transforms the field into a
"multiple email" input, enabling users to enter a list of email addresses separated by commas or semicolons.

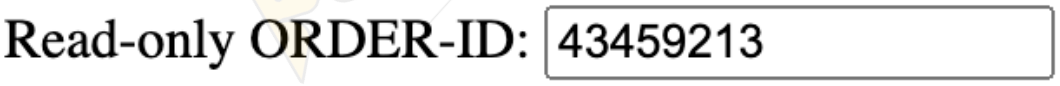
Example:-

```
JavaScript
<label for="fileInput">Select multiple files:</label>
<input type="file" id="fileInput" name="fileInput" multiple />
```



- **readonly:** The readonly attribute is used with form elements in HTML to make an input field or textarea non-editable by the user. When you apply the readonly attribute to an input element or textarea, it prevents the user from modifying the content of that field. However, the user can still view the content, highlight it, and copy the text from it and interact with other elements on the page.

```javascript
<label for="readonlyInput">Read-only ORDER-ID:</label>
<input
  type="text"
  id="readonlyInput"
  name=" readonlyInput"
  value="43459213"
  readonly
/>
```



In the above example, the value in the input field cannot be edited by the user.

- disabled:

The disabled attribute in HTML used to disable user interaction with an HTML element, such as an <input>
field, a <button>, or a <select> element, etc.... When the disabled attribute is applied to an element, it prevents the user from interacting with that element. If you add the disabled attribute to an < input > field within an HTML form, the field will not be included in the form submission. In other words, the data entered into a disabled input field will not be sent to the backend/server when the form is submitted.

Example:-

```
JavaScript
<form>
  <label for="name">Name:</label>
  <input type="text" id="name" name="user_name" value="Prabir" disabled />
  <label for="age">Age:</label>
  <input type="number" id="age" name="age" />
  <input type="submit" value="Submit" />
</form>
```

- **novalidate:**

The novalidate attribute is used in HTML to disable the built- in form validation provided by web browsers.
When this attribute is added to a <form> element, it instructs the browser not to perform its default client-side validation checks when the form is submitted.

For Example, if you have an input field for email and you have added the type email. If you type some random text on that input field and

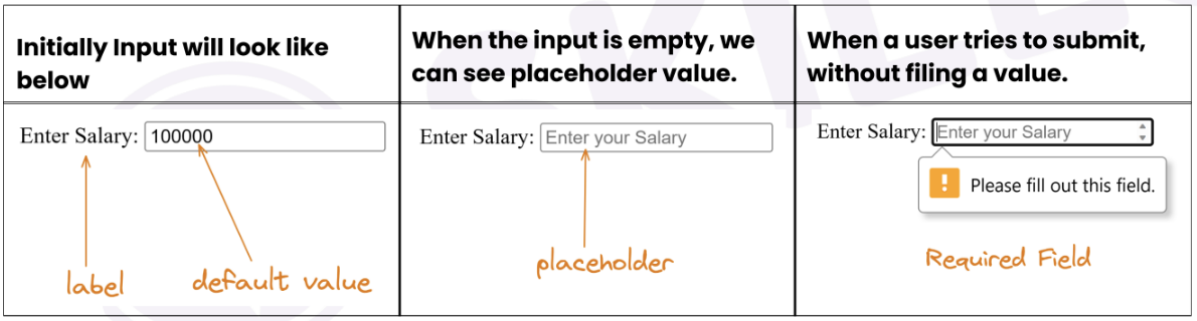
try to submit, generally what happens is that it will throw us a warning
and won't submit that form to the backend/server. But when you add the novalidate attribute on to the form, it won't validate anything, rather it just sends submit the data to the backend.

```JavaScript
<form method="post" novalidate>
  <label for="user_email">Email:</label>
  <input type="email" id="user_email" name="user_email" />
  <input type="submit" value="Submit" />
</form>
```

In the above example, the validation we kept by adding the type email will be ignored when we submit the form because we have added novalidate attribute on the form. We will be understanding different types of attributes and some more validations on the form.

Example of an input attribute

```JavaScript
<form>
  <label for="email">Enter Email: </label>
  <input
    type="number"
    name="salary"
    placeholder="Enter your Salary"
    value="100000"
    required
  />
</form>
```



Common Input Types

1. text: It accepts a single line string .

Example:-

```
JavaScript
<label for="name">Name</label>
<input
  type="text"
  placeholder="write your name"
  id="name"
  name="name"
  required
/>
```

Name write your name

2. **Email:** It is used to create an email input in HTML

Example:-

```
JavaScript
<label for="mailid">Mail Id</label>
<input type="email" id="mailid" name="mailid" />
```

3. **Password:** it allows user to enter passwords secretly

Example:-

```
JavaScript
<label for="password">Password</label>
<input type="password" id="pass" name="pass" />
```

4. Number: only numbers allowed.

```
JavaScript
<label for="salary">Salary</label>
<input type="number" id="salary" name="salary" />
```

5. Checkbox:

- Checkboxes are a type of input field that allows the user to select one or more options from a list of predefined options.

The type attribute of the <input > tag should be set to "checkbox" to create a checkbox.

- The name attribute is used to group related checkboxes together, and the value attribute specifies the value of the checkbox when it is selected. When the user selects a checkbox, the value associated with that checkbox is submitted with the form data.
- If you want to pre-select a checkbox by default, you can add the checked attribute to the < input > tag.

Example:-

```
JavaScript
<label for="male">Male</label>
<input type="checkbox" id="male" name="male" />
<label for="female">Female</label>
<input type="checkbox" id="female" name="female" />
<label for="other">other</label>
<input type="checkbox" id="other" name="other" />
```

Output:

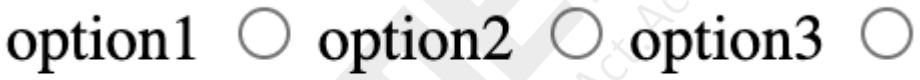
6. Radio:

- Radio buttons are a type of input field that allows the user to select one option from a list of predefined options.
- The **"name"** attribute in the radio button **(<input type="radio">)** tag is used to group related radio buttons together so that only one radio button can be selected at a time within a given group.

- By giving all the radio buttons in a group the same name attribute, you ensure that only one value is submitted for that group of radio buttons.
- This is important when you have a list of options where only one option can be selected, such as in a multiple-choice question or a preference selection.

Example:-

```javascript
<label for="option1">option1</label>
<input type="radio" id="option1" name="selected" value="option1" />
<label for="option2">option2</label>
<input type="radio" id="option2" name="selected" value="option2" />
<label for="option3">option3</label>
<input type="radio" id="option3" name="selected" value="option3" />
```



7. Button:

The "button" type is used to create clickable buttons within a form or on a web page. This works similarly to the <Button> element.

Ex:

```javascript
<input type="button" value="Click me">
```

8. Color:

The "color" type is used to create an input field that allows users to select a color. When this input field is rendered in a web browser, it typically displays a color picker dialog that lets users choose a color visually.

Example:-

```JavaScript
<label for="colorPicker">Choose a color:</label>
<input type="color" id="colorPicker" name="color" />
```

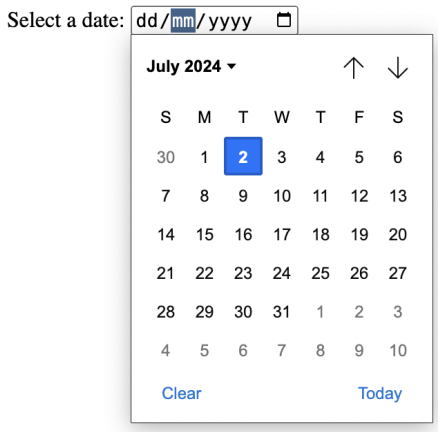
9. Date:

The "date" input type is used to create an input field that allows users to select a date. When you use this input type, it typically displays a date picker or calendar control. This allows users to easily choose a date
without having to manually enter it.

Ex:

```
JavaScript
<label for="datePicker">Select a date:</label>
<input type="date" id="datePicker" name="selectedDate">
```

Output:



Additionally, you can use attributes like min and max to specify a range of acceptable dates and the value attribute to set an initial date value.

Example:-

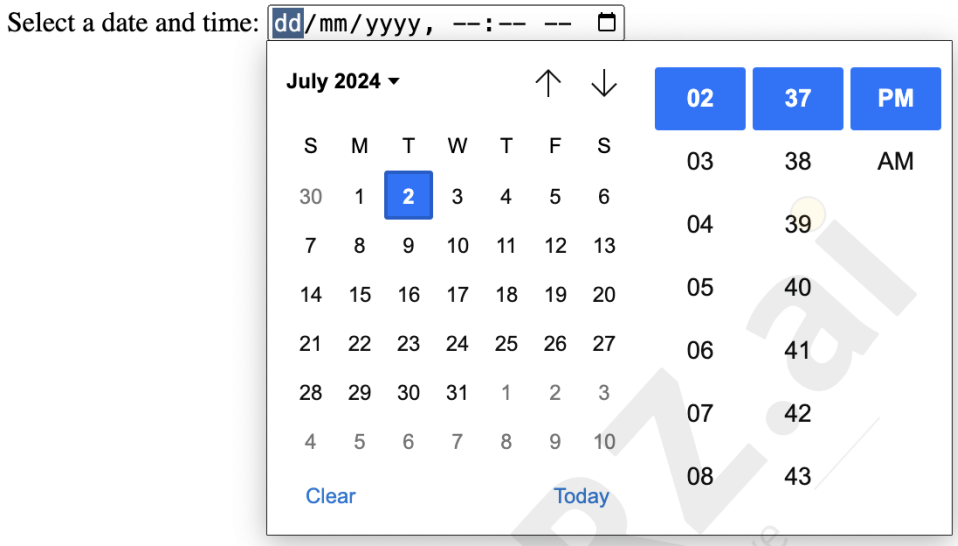
```
JavaScript
<input type="date" id="datePicker" name="selectedDate"
min="2023-01-01" max="2024-06-31" value="2023-08-24">
```

10. Datetime-local:

The "datetime-local" input type in HTML is used to create an input field that allows users to select both a date and a time, including the year, month, day, hour, and minute. It's particularly useful when you need to collect date and time information together, such as scheduling events or appointments.

```
JavaScript
<label for="dateTimePicker">Select a date and time: </label>
<input type="datetime-local" id="dateTimePicker"
name="selectedDateTime">
```

Output:

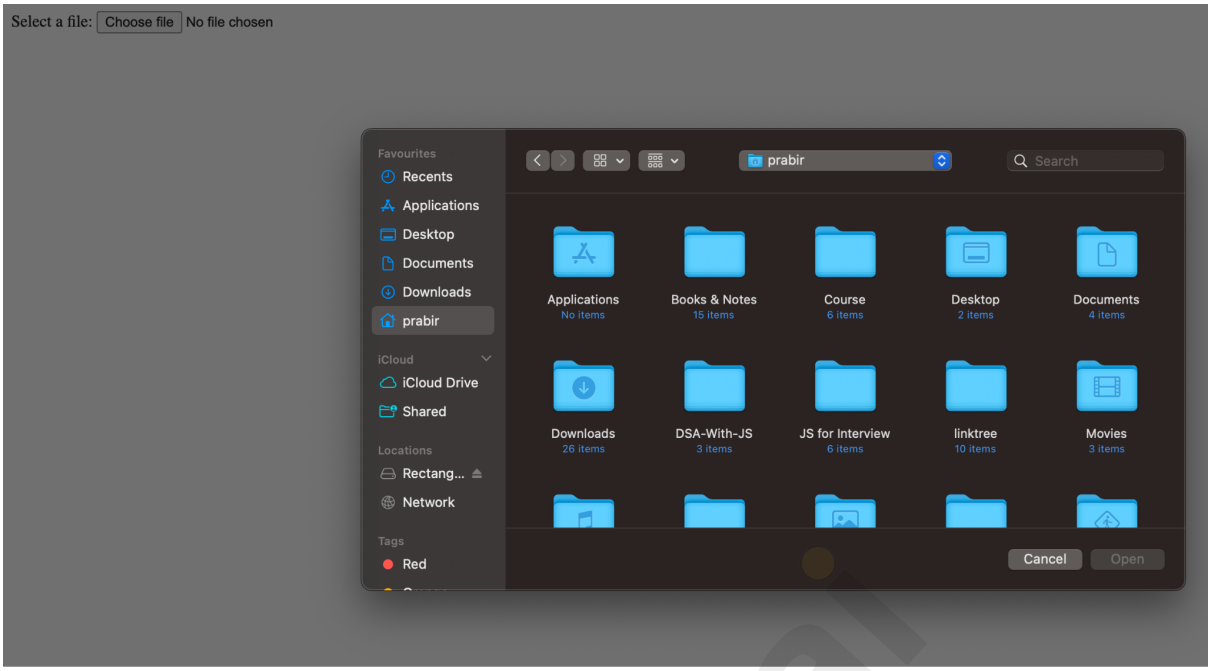


When users interact with the datetime-local input, they can click on it to open a calendar control for selecting the date and a time picker for choosing the time.

11. file:

The "file" type is used to create a file input field in HTML forms. This type of input allows users to select and upload files from their local devices to a web server. It is commonly used for tasks such as uploading images, documents, or other files to a website

```
JavaScript
<label for="fileUpload">Select a file:</label>
<input type="file" id="fileUpload" name="uploadedFile" />
```



When users interact with the file input, they can click on it to open a file selection dialog provided by their operating system. They can then browse their local files and select the files) they want to upload.

12.Hidden

The "hidden" type is used to create a hidden input field in an HTML form. Unlike visible input fields like text boxes or checkboxes, hidden input fields are not visible to users when they view the form on a web page.

Instead, they are used to store data like, user ID, Session Id or many other details which the user may not be aware of or you don't want a user to manually type it but this data you still want to send to the backend/server when the form is submitted.

JavaScript

```
<input type="hidden" name="hiddenField" value="This is a hidden
value" />
```

Hidden input fields are a useful tool for working with forms and managing data on the server-side without requiring user interaction or displaying the data to users

13.image:

The "image" type is used to create an image-based submit button in HTML forms. It allows you to use an image as the button instead of traditional text or a styled button. When the user clicks on the image, it functions as a form submit button and sends the form data to the server. The src attribute is used to set the submit button image and alt attribute will act as the label if it's unable to load that image

```
JavaScript
<input
  type="image"
  src="https://t4.ftcdn.net/jpg/00/28/27/95/360_F
_28279558_SqNXoZWsQLfWYVxyKe9hVzZ49dJtKLsc.jpg"
  width="100"
  height=" 70"
  alt="Submit"
/>
```

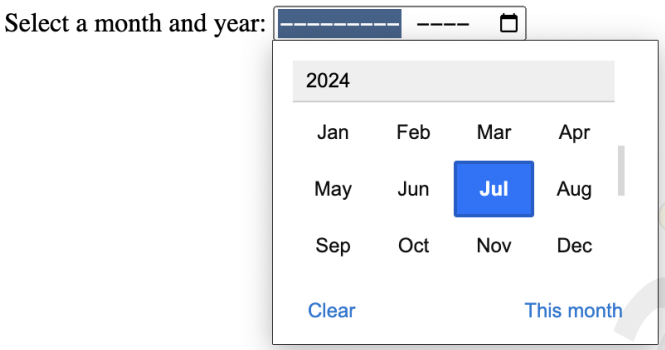
Output:

The image-based submit button is a visually appealing way to create custom submit buttons in your forms, especially when you want to use images or icons for better user experience.

14. month:

The "month" type is used to create an input field that allows users to select a specific month and year. It provides a dropdown or spinner interface that lets users choose a month and year without having to manually enter the information.

```
JavaScript
<label for="monthPicker">Select a month and year:</label>
<input type="month" id="monthPicker" name="selectedMonthYear" />
```



When users interact with the month input, they can click on it to open a dropdown control that allows them to select a month and year. You can also set the min and max attributes & values similar to that of we did in the date type

15.Range:

- The "range" type is used to create an input field that allows users to select a value from a specified range,typically represented as a slider control. It's often used in scenarios where you want users to choose a value within a predefined numeric range, such as setting a volume level, selecting a price range, or adjusting a numerical setting.
- We can define the minimum and maximum and default initial values of the range input element. There is also a step attribute which specifies the increment by which the value can change.

```
JavaScript
<label for="volumeControl">Volume Control:</label>
<input type="range" id="volumeControl" name="volume" min="®" max="100"
step="1"
```

Volume Control: 

Note:Also there are so many properties like reset, search, submit,tel,time,url,week etc.

THANK YOU

AIMERZ.ai
Aim.Act.Achieve