



# WEB DESIGNING

USHNA  
SADAQAT

# INTRODUCTION TO HTML

## WHAT IS HTML?

HTML stands for Hyper Text Markup Language. HTML is the language that helps to create websites. It tells browsers how to show things like headings, paragraphs, and links. It's like a map that guides the layout of web pages so you can easily find what you're looking for.

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

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

-INPUT

## My First Heading

My first paragraph.

-OUTPUT

The `<!DOCTYPE html>` tells browsers this is an HTML5 document. `<html>` is the root element. `<head>` holds info about the page. `<title>` sets the page title seen in the browser tab. `<body>` is where all content, like headings (`<h1>`), paragraphs (`<p>`), and more, goes.

## HTML



## HTML+CSS



## HTML vs CSS

### HTML

1. HTML is a markup language
2. HTML is used to define the structure and content of web pages
3. Uses tags and attributes
4. Separates content and structure
5. Has a hierarchical structure

### CSS

1. CSS is a style sheet language.
2. CSS is used to define the presentation of web pages.
3. Uses selectors, properties, and values
4. Separates presentation from content and structure
5. Has a flat structure



# WHAT IS AN ELEMENT?

An HTML element consists of a start tag, content, and an end tag. It's everything from the start tag ("`<tagname>`") to the end tag ("`</tagname>`"). For example, `<h1>My First Heading</h1>` defines a heading, and `<p>My first paragraph.</p>` defines a paragraph.

Start tag	Element content	End tag
<code>&lt;h1&gt;</code>	My First Heading	<code>&lt;/h1&gt;</code>
<code>&lt;p&gt;</code>	My first paragraph.	<code>&lt;/p&gt;</code>
<code>&lt;br&gt;</code>	<i>none</i>	<i>none</i>

# WHAT IS AN ATTRIBUTE?

HTML elements can have attributes, which offer extra information about them. Attributes are listed in the start tag and often follow a name/value format like: `name="value"`.

```
<!DOCTYPE html>
<html>
<body>
```

-INPUT

```
<h2>The href Attribute</h2>
```

```
<p>HTML links are defined with the a tag. The link address is specified
in the href attribute:</p>
```

```
<a href="https://www.w3schools.com">Visit W3Schools</a>
<!-- ANCHOR TAG -->
```

```
</body>
</html>
```

### The href Attribute

HTML links are defined with the a tag. The link address is specified in the href attribute:

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

-OUTPUT

## HEADINGS

**Heading 1**

**Heading 2**

**Heading 3**

**Heading 4**

**Heading 5**

**Heading 6**

-OUTPUT

```
<!DOCTYPE html>
<html>
<body>
```

-INPUT

```
<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>
```

# TAG REFERENCE:

The tag reference provides extra information about HTML tags and their attributes.

Tag	Description
<u>&lt;html&gt;</u>	Defines the root of an HTML document
<u>&lt;body&gt;</u>	Defines the document's body
<u>&lt;h1&gt; to &lt;h6&gt;</u>	Defines HTML headings

# PARAGRAPH:

```
<!DOCTYPE html>
<html>
<body>

<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>

</body>
</html>
```

This is a paragraph.

This is a paragraph.

This is a paragraph.

# STYLES:

The HTML style attribute lets you apply various styles, like color, font, and size, directly to an element.

```
<!DOCTYPE html>
<html>
<body>

<p>I am normal</p>
<p style="color:red;">I am red</p>
<p style="color:blue;">I am blue</p>
<p style="font-size:50px;">I am big</p>

</body>
</html>
```

I am normal

I am red

I am blue

I am big



# HTML FORMATTING:

HTML formatting elements provide special text styles:

- `<b>`: Bold text
- `<strong>`: Important text
- `<i>`: Italic text
- `<em>`: Emphasized text
- `<mark>`: Marked text
- `<small>`: Smaller text
- `<del>`: Deleted text
- `<ins>`: Inserted text
- `<sub>`: Subscript text
- `<sup>`: Superscript text



```
<!DOCTYPE html>
<html>
<body>
```

```
<p><b>This text is bold</b></p>
<p><i>This text is italic</i></p>
<p>This is<sub> subscript</sub> and
<sup>superscript</sup></p>
<p>My favorite color is <del>blue</del> red.</p>
<p>My favorite color is <del>blue</del>
<ins>red</ins>.</p>
<p>Do not forget to buy <mark>milk</mark> today.</p>

</body>
</html>
```

**BOLD, ITALIC, SUBSCRIPT &  
SUPERSCRIPT, DEL, INS &  
MARKED TEXT.**

**This text is bold**

*This text is italic*

This is <sub>subscript</sub> and <sup>superscript</sup>

My favorite color is ~~blue~~ red.

My favorite color is ~~blue~~ red.

Do not forget to buy **mark** today.

# ORDER LIST:

```
<!DOCTYPE html>
<html>
<body>

<h2>Ordered List with Numbers</h2>

<ol type="1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>

<h2>Ordered List with Letters</h2>

<ol type="A">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

-INPUT

## Ordered List with Numbers

1. Coffee
2. Tea
3. Milk

## Ordered List with Letters

- A. Coffee
- B. Tea
- C. Milk

-OUTPUT



```
<h2>Ordered List with Lowercase Letters</h2>
```

```
<ol type="a">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

```
<h2>Ordered List with Start</h2>
```

```
<ol type="I">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

```
</body>  
</html>
```

-INPUT


## Ordered List with Lowercase Letters

- a. Coffee
- b. Tea
- c. Milk

## Ordered List with Start Attribute

- I. Coffee
- II. Tea
- III. Milk

-OUTPUT



Type	Description
<code>type="1"</code>	The list items will be numbered with numbers (default)
<code>type="A"</code>	The list items will be numbered with uppercase letters
<code>type="a"</code>	The list items will be numbered with lowercase letters
<code>type="I"</code>	The list items will be numbered with uppercase roman numbers
<code>type="i"</code>	The list items will be numbered with lowercase roman numbers

Use the HTML `<ol>` element to define an ordered list

Use the HTML `type` attribute to define the numbering type

Use the HTML `<li>` element to define a list item




## BLOCK ELEMENTS:

- Block-level elements always start on a new line and browsers add margin space before and after them.
- Block-level elements occupy the full width available on a webpage.
- Common block-level elements include `<p>` (paragraph) and `<div>` (division or section).

## INLINE ELEMENTS:

- Inline elements do not start on a new line.
- They only occupy as much width as necessary for their content.
- Example: `<span>` is an inline element typically used within paragraphs (`<p>`).

- `<div>` is a block-level element commonly used as a container for grouping other HTML elements.
- `<span>` is an inline element used to apply styling or mark up a specific part of text or document content.



```
<p style="border: 1px solid black">Hello World</p>
<div style="border: 1px solid black">Hello World</div>
```

<p>The P and the DIV elements are both block elements, and they will always start on a new line and take up the full width available (stretches out to the left and right as far as it can).</p>

<p>This is an inline span <span style="border: 1px solid black">Hello World</span> element inside a paragraph.</p>

```
<div style="background-color:black;color:white;padding:20px;">
```

```
<h2>London</h2>
```

```
<p>London is the capital city of England. It is the most populous city in the United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
```

```
<p>Standing on the River Thames, London has been a major settlement for two millennia, its
history going back to its founding by the Romans, who named it Londinium.</p>
```

```
</div>
```

-INPUT

## -OUTPUT

Hello World

Hello World

The P and the DIV elements are both block elements, and they will always start on a new line and take up the full width available (stretches out to the left and right as far as it can).

This is an inline span Hello World element inside a paragraph.

### **London**

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.



# HTML DIV:

The ``<div>`` element in HTML acts like a container to group and organize content on a webpage.

`<div>` as a container:

```
<div>
  <h1>ISLAMABAD</h1>
  <p>ISLAMABAD IS THE CAPITAL OF PAKISTAN.</p>
  <p>ISLAMABAD IS BEAUTIFUL.</p>
</div>
```





## Multiple <div> :

```
<div>
  <h1>ISLAMABAD</h1>
  <p>ISLAMABAD IS THE CAPITAL OF PAKISTAN.</p>
  <p>ISLAMABAD IS BEAUTIFUL.</p>
</div>
```

```
<div>
  <h1>LONDON</h1>
  <p>LONDON IS THE CAPITAL OF ENGLAND.</p>
  <p>ENGLAND IS BEAUTIFUL.</p>
</div>
```

```
<div>
  <h1>Rome</h1>
  <p>LONDON IS THE CAPITAL OF ITALY.</p>
  <p>ITALY IS BEAUTIFUL.</p>
</div>
```



# Section:

## WHAT IS SECTION?

Certainly! In simpler terms, a "section" in HTML is like a labeled container that holds related content on a webpage. It's used to organize and structure the different parts of a page so that it's easier for browsers, search engines, and screen readers to understand.

Imagine you're writing a story online. You might have sections like "Introduction," "Main Story," and "Conclusion." Each of these would be marked with `<section>` in your HTML code. This helps to clearly separate and identify each part of your story.

So, `<section>` in HTML is just a way to neatly group and organize content, making your webpage easier to read and navigate for everyone.

# NAVBAR:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

-INPUT

```
<ul>
```

```
<li><a href="#">Home</a></li>
```

```
<li><a href="#">News</a></li>
```

```
<li><a href="#">Contact</a></li>
```

```
<li><a href="#">About</a></li>
```

```
</ul>
```

```
<p>Note: We use href="#" for test links. In a real  
website this would be URLs.</p>
```

```
</body>
```

```
</html>
```

-OUTPUT

- [Home](#)
- [News](#)
- [Contact](#)
- [About](#)

Note: We use href="#" for test links. In a real web site this would be URLs.



## WHY WE ADD # IN NAVBAR?

When you see `<a href="#">Home</a>` in HTML, the `#` symbol means that clicking "Home" won't take you anywhere. It's often used when creating navigation menus or buttons that don't need to link to a new page right away. Instead, it's waiting for something else to happen, like JavaScript functionality or further development.

In Simple: `href="#"` is a placeholder to test links used in web development until real URLs are added to website.



## SIMPLE BUTTON:

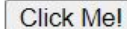
```
<!DOCTYPE html>
<html>
<body>

<h1>The button Element</h1>

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

</body>
</html>
```

### The button Element



Click Me!



# BUTTON WITH LINK:

```
<!DOCTYPE html>
<html>
<body>

<h1>The button Element</h1>

<button type="button" onclick="alert('Hello world!')">
  <a href="https://www.google.com/" style="color: grey;">google</a>
</button>

</body>
</html>
```

—GENERATE YOUR OUTPUT—



# WHAT'S ONCLICK BUTTON & ALERT?

The `<button>` and its attributes like `onclick` are part of HTML—they define how a button looks and behaves on a webpage.

- **HTML part:** `<button>` creates the button and `onclick` tells the button what to do when clicked.
- **JavaScript part:** `onclick="alert('DONE!')"` is like a mini-program written in JavaScript. It triggers an alert message ("DONE!") when someone clicks the button.

So, even though it's mostly HTML, the `onclick` part adds interactive features (like showing alerts) without needing a separate `<script>` tag for JavaScript code. It's a quick way to make things happen when users interact with your webpage



# ADD BACKGROUND COLORS & FONT SIZE:

```
<!DOCTYPE html>
<html>
<body style="background-color:powderblue;">

<h1 style="background-color:skyblue; font-size:50px;">This is a
heading</h1>
<p style="background-color:tomato;">This is a paragraph.</p>


<h1 style="background-color:skyblue; font-size:50px;">This is a
heading</h1>
<p style="background-color:tomato;">This is a paragraph.</p>


<button type="button" onclick="alert('Hello world!')">
  <a href="https://www.google.com/" style="color: grey;">google</a>
</button>


</body>
</html>
```

—GENERATE YOUR OUTPUT—





# INTRODUCTION TO CSS

## WHAT IS CSS?

CSS stands for Cascading Style Sheets. CSS styles HTML documents, specifying how elements appear. .

### Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

### EXTERNAL CSS:

An external style sheet allows changing the entire website's appearance by modifying just one file. Each HTML page must include a link to this external style sheet within the ``<head>`` section.

### INTERNAL CSS:

An internal style sheet is used for applying unique styles to a single HTML page. It is defined within the ``<style>`` element located in the head section of the HTML document.

# EXTERNAL CSS:

```
index.html<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>External CSS Example</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Welcome to My Website</h1>
    <p>This is an example of using external CSS.</p>
  </div>
</body>
</html>
```

## Style.css

```
body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
}
h1 {
  color: blue;
}
.container {
  width: 80%;
  margin: 0 auto;
  padding: 20px;
  background-color: white;
}
```



# INTERNAL CSS:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Internal CSS Example</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f0f0;
    }
    h1 {
      color: blue;
    }
    .container {
      width: 80%;
      margin: 0 auto;
      padding: 20px;
      background-color: white;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Welcome to My Website</h1>
    <p>This is an example of using internal CSS.</p>
  </div>
</body>
</html>
```



## INLINE CSS:

Inline styles apply unique styles to individual elements by using the style attribute, which can contain various CSS properties.

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

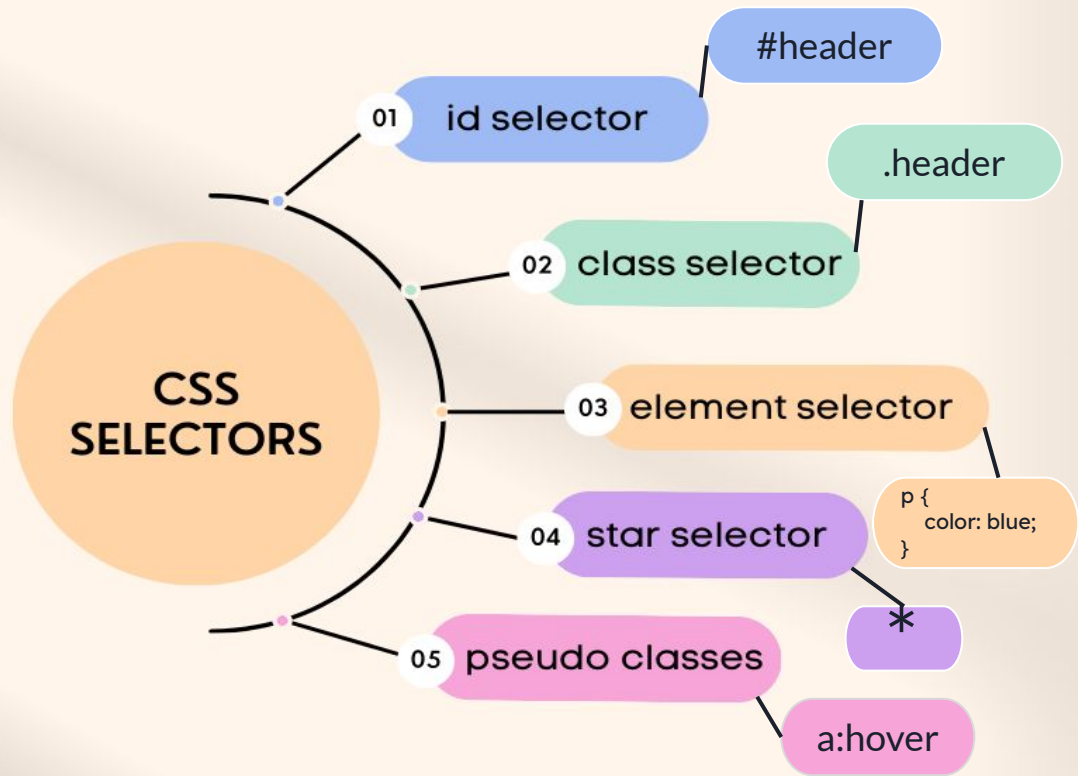
```
<h1 style="color:blue;text-align:center;">This is a heading</h1>
```

```
<p style="color:red;">This is a paragraph.</p>
```

```
</body>
```

```
</html>
```

# CSS SELECTORS:





## Selectors in CSS

### Element Selector

```
h2 {  
  color: #c70039 ;  
}
```

### Universal Selector

```
* {  
  color: #c70039 ;  
}
```

### ID Selector

```
#content {  
  color: #6E4253;  
  font-size: 15px;  
}
```

### Class Selector

```
.main {  
  margin-top: 10px  
  margin-bottom: 10px  
}
```

# INTRODUCTION TO JAVAScript

## **<script> Tag:**

- This tag is where you write JavaScript code that will make your webpage interactive.

## **function myFunction() { ... }:**

- function myFunction() defines a function in JavaScript. Think of a function as a set of instructions that does something specific when you tell it to.

## **document.getElementById("demo"):**

- document.getElementById("demo") is how JavaScript finds a specific part of your webpage. It looks for an element with id="demo".

## **.innerHTML = "JScript Here!";**

- .innerHTML is a way to change what's written inside an HTML element.
- Here, it changes the content inside the element with id="demo" to say "JScript Here!".

## **CODE**

```
<script>
function myFunction() {

    document.getElementById("d
emo").innerHTML = "JScript
Here!";
}
</script>
```



Imagine you have a paragraph (`<p>`) element in your HTML like this.

When you click a button on your webpage that calls `myFunction` (like `<button onclick="myFunction()">Click me</button>`), it runs the JavaScript code inside `myFunction`.

After clicking the button, the text inside the paragraph (`<p>` element) with `id="demo"` will update to say "JScript Here!".

### Summary:

- **Purpose:** This JavaScript function (`myFunction`) updates the text inside an HTML element (`<p>` with `id="demo"`) when called.
- **Functionality:** It shows how JavaScript can find elements on your webpage and change what they say, making your web page more interactive and responsive to user actions.

JavaScript, used with HTML and CSS, allows you to create dynamic and engaging web pages by manipulating elements and responding to user interactions.

### CODE

```
<p id="demo">Demonstration.</p>
  <button type="submit"
onclick="myFunction()" ">GET
STARTED</button>

  <script>
    function
myFunction() {

document.getElementById("demo") .
innerHTML = "JScript Here!";

}
</script>
```





# WHAT IS JAVASCRIPT?

## HTML (Content):

- **What it does:** Defines the structure and content of your web page using tags.
- **Example:** `<h1>Welcome to My Website</h1>`, `<p>This is a paragraph.</p>`

## CSS (Style):

- **What it does:** Styles the HTML content to make it look nice, like choosing colors and fonts.
- **Example:** `h1 { color: blue; }, p { font-size: 16px; }`

## JavaScript (Behavior):

- **What it does:** Makes your web page interactive and dynamic, like responding to clicks or updating content.

## SUMMARY:

- **Use `id`** when you need to uniquely identify an element for JavaScript manipulation or CSS styling that is specific to that element.
- **Use `class`** when you want to apply styles or behavior to multiple elements that share a common characteristic.



# ID AND CLASS DIFFERENCE:

## ID ATTRIBUTE:

**Purpose:** Identifies a unique element on the page.

**Usage:** Used for specific styling or JavaScript operations that target one particular element.

**Example:** `<div id="header">...</div>`

## CLASS ATTRIBUTE:

**Purpose:** Groups elements that share the same styles or behavior.

**Usage:** Used for styling multiple elements similarly or targeting them with JavaScript.

**Example:** `<div class="box red">...</div>`



## KEY DIFFERENCE:

**Uniqueness:** `id` must be unique per page; `class` can be used on multiple elements.

- `id`: Use sparingly for unique elements.
- `class`: Use for styling and grouping multiple elements.

In essence, `id` is for unique identification, while `class` is for grouping and styling.

Both are crucial for creating well-structured and styled web pages.



## **ID:**

- Used to uniquely identify one specific element on a webpage.
- Each id must be unique within the HTML document.
- Useful when you want to target and style or manipulate a particular element with CSS or JavaScript.

## **CLASS:**

- Used to group and style multiple elements that share a common characteristic.
- Many elements can have the same class.
- Great for applying consistent styling or behavior to groups of elements.



## Simple Explanation:

- **ID:** Think of it as a unique name for one special thing on your webpage, used mainly for styling or making it do something special with JavaScript.
- **Class:** Use it to give a bunch of things a common label so you can style them all the same way, or make them all do something similar with JavaScript.

Both id and class attributes help you keep your webpage organized and looking just right!



## -CONCLUSION-

-JavaScript complements HTML and CSS by adding functionality and interactivity to web pages. Learning JavaScript allows developers to create modern, responsive, and user-friendly web applications that meet today's expectations for dynamic content and engaging user experiences.



# WHAT'S MEDIA QUERY?

## Why We Add Media Queries?

1. **Responsive Design:** Media queries help make a website look good on different screen sizes (like phones, tablets, and desktops).
2. **Adapt Layouts:** They allow you to change styles based on the device's width, height, or other features, ensuring a better user experience.
3. **Control Visibility:** You can show or hide elements (like menus) depending on the screen size.
4. **Improve Usability:** Making adjustments for smaller screens (like stacking items or larger buttons) makes it easier for users to interact with the site.

## In Simple Terms:

Media queries help your website look good and work well on all types of devices.



## Why We Add This Line?

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

### Purpose:

1. **Responsive Design:** This meta tag helps the browser understand how to scale the webpage on different devices, making it easier to implement responsive design.
2. **Width Control:** Setting width=device-width ensures the page uses the full width of the device screen, allowing media queries to work effectively.
3. **Initial Scale:** initial-scale=1.0 sets the initial zoom level when the page is first loaded, ensuring users see the content at the intended size.
- 4.

### In Simple Terms:

This line makes sure your website looks good and fits properly on all devices, which helps your media queries work as expected.





# STYLE INSIDE MEDIA QUERY:

```
.menu-toggle:  
.menu-toggle {
```

```
    display: block; /* Show toggle button */
```

```
}
```

**Purpose:** This makes the menu toggle button visible. It's important for smaller screens where the full navigation might be hidden.

```
.nav:  
.nav {
```

```
    display: none; /* Initially hide the navigation */
```

```
    flex-direction: column; /* Stack items vertically */
```

```
    width: 100%; /* Full width */
```

```
}
```

**Purpose:** The navigation menu is hidden by default to save space. It will stack items vertically to fit better on small screens and takes up the full width.



```
.nav.active:  
.nav.active {
```

```
    display: flex; /* Show navigation when active */
```

```
}
```

**Purpose:** When the navigation has the active class (usually added by JavaScript when the toggle button is clicked), it will be displayed. This allows users to see the menu when they want to access it.

```
nav ul:  
nav ul {
```

```
    flex-direction: column; /* Stack nav items */
```

```
    align-items: flex-start; /* Align to start */
```

```
}
```

**Purpose:** This ensures that the navigation items inside the unordered list (ul) are stacked vertically and aligned to the left. This layout works better on smaller screens.

```
nav ul li:  
nav ul li {
```

```
    margin: 5px 0; /* Add vertical spacing */
```

```
}
```

**Purpose:** This adds some space between the navigation items, making them easier to tap on touch screens and improving overall readability.



## Summary of Why We Use These Styles:

- **Visibility and Usability:** The `.menu-toggle` button is shown so users can open the hidden menu, which is crucial for navigation on small screens.
- **Responsive Layout:** The navigation is set to hide by default and only shows when active, ensuring that the interface is clean and user-friendly.
- **Stacking:** Vertical stacking of items and proper spacing enhances usability on mobile devices, making it easier for users to interact with the menu.

## In Simple Terms:

This code helps the navigation work well on small screens by hiding it at first, showing it when needed, and stacking items for easy use.

# APPLYING JAVASCRIPT FOR RESPONSIVE NAVBAR

## Selecting Elements:

- `const menuToggle = document.querySelector('.menu-toggle');`
  - This line finds the button (or element) with the class `menu-toggle` and saves it in the variable `menuToggle`.
- `const nav = document.querySelector('.nav');`
  - This line finds the navigation menu (or element) with the class `nav` and saves it in the variable `nav`.

## Adding a Click Event:

- `menuToggle.addEventListener('click', () => { ... });`
  - This sets up an action that happens when the `menuToggle` button is clicked.

## Toggling the Menu:

- `nav.classList.toggle('active');`
  - When the button is clicked, this line adds or removes the class `active` from the `nav` element. If `nav` has the `active` class, it will be removed; if it doesn't, the class will be added. This is often used to show or hide the menu.

```
<script>
  const menuToggle =
document.querySelector('.menu-t
oggle');
  const nav =
document.querySelector('.nav');

menuToggle.addEventListener('cl
ick', () => {

nav.classList.toggle('active');

});
</script>
```



## IN SIMPLE

1. **\*\*Find the Button and Menu\*\***:
  - It looks for a button called ``menu-toggle`` and a menu called ``nav``.
2. **\*\*Click Action\*\***:
  - When you click the button, it will do something.
3. **\*\*Show or Hide the Menu\*\***:
  - If the menu is open, it will close it. If it's closed, it will open it. This is done by adding or removing a special label called ``active``.

### In Very Simple Terms:

Clicking the button makes the menu open or close.



## —!TASK ALERT!—

- Create Headings. ✓
- Create ANCHOR TAG. ✓
- Create Order List. ✓
- Do Styling and adjust Font Size. ✓
- Create Button with color Edits. ✓
- Add Background Color in Inline CSS. ✓
- Create JAVASCRIPT Buttons By Adding Colors. (NEW!)
- Create NAVBAR. (NEW!)
- Create Header & Footer. (NEW!)
- Add Logo Or Logo Image. (NEW!)
- Add Background Image. (NEW!)
- Do CSS. (NEW!)