# Table of Content:

# HTML

## What is HTML

HTML, or HyperText Markup Language, is the code used to create and structure content on the web. It uses tags to define elements such as headings, paragraphs, and etc. Browsers then interpret HTML to display the web page as intended.

```
<html>
<head>
   <title>Hello World Page</title>
</head>
<body>
   <h1>Hello, World!</h1>
</body>
</html>
```

## How to Run

- Open any text editor on your computer, such as Notepad, TextEdit, or Visual Studio Code.
- Copy and paste the provided HTML code into the text editor.
- Save the file with a .html extension, for example, "hello.html".
- Locate the saved file on your computer and double-click it.
- Your default web browser should open, displaying the "Hello, World!" message.

## list of tags in HTML

```
<!DOCTYPE html>: Document type declaration.
<html>: Root element of an HTML document.
<head>: Contains metadata about the document.
<title>: Sets the title of the HTML document.
<meta>: Provides metadata about the document.
<body>: Contains the content of the HTML document.
<h1> to <h6>: Heading tags.
<p>: Paragraph.
<a>: Hyperlink.
<img>: Image.
<ul>: Unordered list.
<li>: List item in an unordered list.
<ol>: Ordered list.
<li>: List item in an ordered list.
<br>: Line break.
<hr>: Horizontal line.
<b>: Represents strong importance, typically bold.
<i>: Represents emphasized text, typically italicized.
<span>: Inline container for stylistic purposes.
<div>: Block-level container for grouping elements.
<input>: Input field.
<form>: Represents an HTML form.
<button>: Represents a clickable button.
<label>: Describes the purpose of an input element.
<select>: Dropdown list.
<option>: Represents an option in a <select> element.
```

## Examples

1. <!DOCTYPE html>

```
<!DOCTYPE html>
<html>
<head>
   <title>DOCTYPE Example</title>
</head>
<body>
   <!-- Content goes here -->
```

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

2. <html>

```
<!DOCTYPE html>
<html>
<head>
   <title>HTML Example</title>
</head>
<body>
   <!-- Content goes here -->
</body>
</html>
```

3. <head>

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <meta name="description" content="HTML Head Example">
   <meta name="keywords" content="HTML, Head, Example">
   <meta name="author" content="Your Name">
</head>
<body>
   <!-- Content goes here -->
</body>
</html>
```

4. <title>

```
<!DOCTYPE html>
<html>
<head>
   <title>Hello World Title</title>
</head>
<body>
   <!-- Content goes here -->
</body>
</html>
```

5. <body>

```
<!DOCTYPE html>
<html>
<head>
   <title>Body Example</title>
</head>
```

```
<body>
   <h1>Heading 1</h1>
</body>
</html>
```

6. <h1> to <h6>

```
<!DOCTYPE html>
<html>
<head>
   <title>Heading Example</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>
```

7. <p>

```
<!DOCTYPE html>
<html>
<head>
   <title>Paragraph Example</title>
</head>
<body>
   <p>This is a sample paragraph.</p>
</body>
</html>
```

8. <a>

```
<!DOCTYPE html>
<html>
<head>
   <title>Anchor Example</title>
</head>
<body>
   <a href="https://www.example.com">Visit Example Website</a>
</body>
</html>
```

9. <img>

```
<!DOCTYPE html>
<html>
<head>
   <title>Image Example</title>
</head>
<body>
   <img src="example.jpg" alt="Example Image">
</body>
</html>
```

10. <ul> and <li>

```
<!DOCTYPE html>
<html>
<head>
   <title>Unordered List Example</title>
</head>
<body>
   <ul>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
   </ul>
</body>
</html>
```

11. <ol> and <li>

```
<!DOCTYPE html>
<html>
<head>
   <title>Ordered List Example</title>
</head>
<body>
   <ol>
      <li>First Item</li>
      <li>Second Item</li>
      <li>Third Item</li>
   </ol>
</body>
</html>
```

12. <br>

```
<!DOCTYPE html>
<html>
<head>
   <title>Line Break Example</title>
```

```
</head>
<body>
   <p>This is a line of text.<br>Here is a new line.</p>
</body>
</html>
```

13. <b>

```
<!DOCTYPE html>
<html>
<head>
   <title>Bold Example</title>
</head>
<body>
   <p>This is <b>strong</b> text </p>
</body>
</html>
```

14. <i>

```
<!DOCTYPE html>
<html>
<head>
   <title>Emphasis Example</title>
</head>
<body>
   <p>This is <i>emphasized</i> text</p>
</body>
</html>
```

15. <span>

```
<!DOCTYPE html>
<html>
<head>
   <title>Span Example</title>
</head>
<body>
   <p>This is <span style="color: blue;">blue</span> text.</p>
</body>
</html>
```

16. <div>

```
<!DOCTYPE html>
<html>
<head>
   <title>Div Example</title>
```

```
</head>
<body>
   <div>
      <p>This is content inside a div.</p>
      <p>More content in the same div.</p>
   </div>
</body>
</html>
```

17. <input>

```
<!DOCTYPE html>
<html>
<head>
   <title>Input Example</title>
</head>
<body>
   <form>
      <label for="username">Username:</label>
      <input type="text" id="username" name="username">
   </form>
</body>
</html>
```

18. <form> and <button>

```
<!DOCTYPE html>
<html>
<head>
   <title>Form Example</title>
</head>
<body>
   <form>
      <label for="fname">First Name:</label>
      <input type="text" id="fname" name="fname"><br>
      <label for="lname">Last Name:</label>
      <input type="text" id="lname" name="lname"><br>
      <button type="submit">Submit</button>
   </form>
</body>
</html>
```

19. <label> and <select>

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

```
    <form>
      <label for="cars">Choose a car:</label>
      <select id="cars" name="cars">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
        <option value="mercedes">Mercedes</option>
        <option value="audi">Audi</option>
      </select>
    </form>
</body>
</html>
```

## Exercise 1: Build a Personal Website

Create a personal website that includes the following elements:

- A title for the page using <title>.
- Headings (<h1>, <h2>, <h3>) to organise different sections (e.g., About Me, Projects, Contact).
- Paragraphs (<p>) to describe yourself, your interests, and your projects.
- Links (<a>) to navigate to different sections within the page.
- Images (<img>) to showcase your projects or include a profile picture.
- Lists (<ul> and <li>) to highlight your skills or achievements.
- Styling with inline styles (<style>) or an external stylesheet (<link>).

## Exercise 2: Create a Simple Form

Design a simple HTML form that includes:

- A title using <title> for the form.
- <form> tag to wrap the entire form.
- Labels (<label>) for each input field to provide information about the expected input.
- Input fields (<input>) for various data types (text, email, password).
- A textarea (<textarea>) for a longer message or comment.
- Buttons (<button>) for submitting the form or resetting the input fields.
- Styling with inline styles or an external stylesheet.

# CSS

CSS, or Cascading Style Sheets, is a styling language used in web development to control the visual presentation of HTML and XML documents. It allows designers to define the layout, colours, fonts, and other stylistic aspects of a web page, ensuring a consistent and attractive appearance across different devices and screen sizes. CSS works in conjunction with HTML and JavaScript to create a well-designed and responsive user interface for websites.

## Types of CSS:

### Inline:

Inline styles are applied directly to individual HTML elements using the style attribute

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

   <h1 style="color: blue;">Inline CSS Example</h1>
   <p style="font-size: 16px; text-align: center;">This paragraph has inline styles.</p>

</body>
</html>
```

### Internal

Internal styles are defined within the <style> tag in the head section of the HTML document.

```
<!DOCTYPE html>
<html>
<head>
   <style>
      h1 {
         color: green;
      }

      p {
         font-size: 18px;
         text-align: justify;
      }
   </style>
</head>
<body>

<h1>Internal CSS Example</h1>
<p>This paragraph uses internal styles.</p>

</body>
</html>
```

### External

External styles are stored in a separate CSS file and linked to the HTML document using the <link> tag.

Create a new file and store it as a style.css

```
<!-- styles.css -->
/* styles.css */

h1 {
   color: red;
}

p {
   font-size: 20px;
   text-align: center;
}
```

```
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
   <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>

<h1>External CSS Example</h1>
<p>This paragraph uses styles from an external CSS file.</p>

</body>
</html>
```

## List of tags:

Reference: https://www.w3schools.com/cssref/index.php

Few example properties listed below

### Layout:

- display
- position
- top, right, bottom, left
- float
- clear
- z-index

### Box Model:

- width, height
- margin, padding
- border, border-radius
- box-sizing

### Typography:

- font-family, font-size, font-weight
- line-height
- text-align, text-decoration, text-transform
- color

### Background and Borders:

- background-color, background-image
- border-color, border-width, border-style
- box-shadow

### Flexbox:

- display: flex
- flex-direction, flex-wrap
- justify-content, align-items, align-self
- order

### Grid:

- display: grid
- grid-template-rows, grid-template-columns
- grid-gap
- grid-row, grid-column

### Transformations:

- transform
- rotate, scale, translate
- transform-origin

### Transition and Animation:

- transition
- animation, keyframes

### Responsive Design:

- @media

## Ways to Use CSS Properties

### Universal Selector (*):

Definition: Selects all elements on the page.

```
<!DOCTYPE html>
```

```
<html>
<head>
   <style>
      * {
         color: blue;
      }
   </style>
</head>
<body>
   <p>This is a paragraph with blue text.</p>
   <div>This is a div with blue text.</div>
   <!-- All elements will have blue text -->
</body>
</html>
```

## Type Selector (element):

Definition: Selects all instances of a specified HTML element.

```
<!DOCTYPE html>
<html>
<head>
   <style>
      p {
         font-style: italic;
      }
   </style>
</head>
<body>
   <p>This paragraph is italicized.</p>
   <p>So is this one.</p>
   <!-- Only <p> elements will be italicized -->
</body>
</html>
```

## Class Selector (.class):

Definition: Selects all elements with a specific class attribute.

```
<!DOCTYPE html>
<html>
<head>
   <style>
      .highlight {
         background-color: yellow;
      }
   </style>
</head>
<body>
   <p class="highlight">This paragraph is highlighted.</p>
   <div class="highlight">So is this div.</div>
   <!-- All elements with class="highlight" will have a yellow background -->
```

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

## ID Selector (#id):

Definition: Selects a single element with a specific ID attribute.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        #main-title {
            font-size: 24px;
        }
    </style>
</head>
<body>
    <h1 id="main-title">This is the main title</h1>
    <!-- Only the element with id="main-title" will have a font size of 24px -->
</body>
</html>
```

## Descendant Selector (ancestor descendant):

Definition: Selects all descendants of a specified ancestor.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        article p {
            font-weight: bold;
        }
    </style>
</head>
<body>
    <article>
        <p>This paragraph is bold.</p>
    </article>
    <!-- Only <p> elements inside <article> will have bold font weight -->
</body>
</html>
```

## Child Selector (parent > child):

Selects all direct children of a specified parent.

```
<!DOCTYPE html>
<html>
<head>
```

```
    <style>
        .parent > p {
            color: green;
        }
    </style>
</head>
<body>
    <div class="parent">
        <p>This paragraph has green text.</p>
        <span>This span is not affected.</span>
    </div>
    <!-- Only <p> elements that are direct children of an element with class="parent" will
have green text. -->
</body>
</html>
```

## Adjacent Sibling Selector (prev + next):

Selects an element that is immediately preceded by a specified sibling.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        h2 + p {
            font-style: italic;
        }
    </style>
</head>
<body>
    <h2>Heading 2</h2>
    <p>This paragraph is italicized.</p>
    <span>This span is not affected.</span>
    <!-- Only <p> elements immediately preceded by an <h2> will be italicized. -->
</body>
</html>
```

## General Sibling Selector (prev ~ siblings):

Selects all siblings that share the same parent.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        h2 ~ p {
            color: purple;
        }
    </style>
</head>
<body>
```

```
    <h2>Heading 2</h2>
    <p>This paragraph and its siblings have purple text color.</p>
    <p>Another paragraph with the same style.</p>
    <!-- All <p> elements that are siblings of an <h2> will have purple text color. -->
</body>
</html>
```

## Exercise 1: Style a Personal Portfolio

Create a simple HTML page that serves as a personal portfolio. Include sections for your introduction, skills, projects, and contact information. Use CSS to style the layout, fonts, colours, and add hover effects for interactive elements. Practise responsive design to ensure your portfolio looks good on various devices.

Define your portfolio's colour scheme and typography, experiment with different layouts, and make use of flexbox or grid for positioning elements. Add a subtle transition effect to enhance the user experience.

## Exercise 2: Build a Pricing Table

Design a pricing table using HTML and CSS for a fictional product or service. Create columns for different pricing tiers with features listed beneath each tier. Apply CSS styles to make the pricing table visually appealing, with distinct colour highlights for the recommended or most popular plan. Utilise borders, shadows, or background colours to create separation between elements.

Practice responsiveness by adjusting the layout for smaller screens. Experiment with different ways to emphasize certain features or plans, such as using icons, varying font weights, or adding subtle animations for a polished look.

# Java script:

JavaScript is a programming language that makes websites interactive. It allows developers to create dynamic and responsive elements on web pages, making them more engaging for users. With JavaScript, you can control and modify website content, respond to user actions, and enhance the overall browsing experience.

## Concepts in JS

- Variables: Containers for storing data values.
- Data Types: Types of values like strings, numbers, booleans, and objects.
- Operators: Symbols used to perform operations on variables and values.
- Conditional Statements: Code structures like if statements for decision-making.
- Loops: Repeatedly execute a block of code, such as for and while loops.
- Functions: Blocks of reusable code that can be called with a name.

- Arrays: Ordered lists of values, accessible by index.
- Objects: Collections of key-value pairs, often used for complex data structures.
- Event Handling: Responding to user actions like clicks or key presses.
- DOM Manipulation: Changing the structure or style of HTML elements on a webpage.
- Asynchronous JavaScript: Handling tasks that don't need to be completed immediately, like fetching data from a server.
- Callbacks and Promises: Dealing with asynchronous code and managing the order of execution.
- Closures: Functions that remember the environment in which they were created.
- Scope: The context in which variables are declared and can be accessed.
- Hoisting: The behavior where variable and function declarations are moved to the top of their containing scope during compilation.

## Variables:

Definition: Containers for storing data values.

```
<!DOCTYPE html>
<html>
<head>
  <script>
    var greeting = "Hello, ";
    var name = "John";
    var message = greeting + name;
    alert(message);
  </script>
</head>
<body>
  <!-- No visible output, but an alert will display "Hello, John" -->
</body>
</html>
```

## Data Types:

Explore different data types such as strings, numbers, and booleans.

```
<!DOCTYPE html>
<html>
<head>
  <script>
    var message = "Hello"; // String
    var number = 42; // Number
    var isTrue = true; // Boolean
  </script>
</head>
<body>
```

```
    <!-- No visible output, but variables are assigned different data types -->
</body>
</html>
```

## Operators:

Use operators to perform operations on variables and values.

```
<!DOCTYPE html>
<html>
<head>
    <script>
        var num1 = 10;
        var num2 = 5;
        var result = num1 + num2;
        alert("Result: " + result);
    </script>
</head>
<body>
    <!-- No visible output, but an alert will display "Result: 15" -->
</body>
</html>
```

## Conditional Statements:

Implement conditional statements for decision-making.

```
<!DOCTYPE html>
<html>
<head>
    <script>
        var age = 18;
        if (age >= 18) {
            alert("You are eligible to vote.");
        } else {
            alert("You are not eligible to vote yet.");
        }
    </script>
</head>
<body>
    <!-- No visible output, but an alert will display the eligibility message -->
</body>
</html>
```

## Loops:

Use loops to repeatedly execute a block of code.

```
<!DOCTYPE html>
<html>
<head>
   <script>
      for (var i = 0; i < 5; i++) {
         console.log("Iteration " + i);
      }
   </script>
</head>
<body>
   <!-- Open the browser console to see the output -->
</body>
</html>
```

## Functions:

Create and use functions for reusable blocks of code.

```
<!DOCTYPE html>
<html>
<head>
   <script>
      function greet(name) {
         alert("Hello, " + name + "!");
      }

      greet("Alice");
      greet("Bob");
   </script>
</head>
<body>
   <!-- No visible output, but two alerts will display personalized greetings -->
</body>
</html>
```

## Arrays:

Define and manipulate arrays to store and access multiple values.

```
<!DOCTYPE html>
<html>
<head>
   <script>
      var colors = ["red", "green", "blue"];
      console.log(colors[0]); // Output: "red"
   </script>
</head>
<body>
   <!-- Open the browser console to see the output -->
</body>
```

```
</html>
```

## Objects:

Create and use objects to store key-value pairs.

```
<!DOCTYPE html>
<html>
<head>
    <script>
        var person = {
            name: "Alice",
            age: 25,
            occupation: "Engineer"
        };

        console.log(person.name); // Output: "Alice"
    </script>
</head>
<body>
    <!-- Open the browser console to see the output -->
</body>
</html>
```

## Event Handling:

Respond to user actions like clicks using event handlers.

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function showMessage() {
            alert("Button clicked!");
        }
    </script>
</head>
<body>
    <button onclick="showMessage()">Click me</button>
</body>
</html>
```

## DOM Manipulation:

Change the structure or style of HTML elements on a webpage.

```
<!DOCTYPE html>
```

```
<html>
<head>
   <script>
      function changeColor() {
         document.getElementById("targetElement").style.color = "purple";
      }
   </script>
</head>
<body>
   <p id="targetElement">This paragraph can change color.</p>
   <button onclick="changeColor()">Change Color</button>
</body>
</html>
```

## Asynchronous JavaScript:

Handle tasks that don't need to be completed immediately, like fetching data from a server.

```
<!DOCTYPE html>
<html>
<head>
   <script>
      setTimeout(function() {
         console.log("Delayed task completed.");
      }, 2000);
   </script>
</head>
<body>
   <!-- Open the browser console to see the output after a delay -->
</body>
</html>
```

## Callbacks and Promises:

Deal with asynchronous code and manage the order of execution.

```
<!DOCTYPE html>
<html>
<head>
   <script>
      function fetchData(callback) {
         setTimeout(function() {
            console.log("Data fetched successfully.");
            callback();
         }, 1500);
      }

      fetchData(function() {
         console.log("Callback executed.");
      });
```

```
    </script>
</head>
<body>
    <!-- Open the browser console to see the output -->
</body>
</html>
```

## Closures:

Functions that remember the environment in which they were created.

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function outerFunction() {
            var outerVariable = "I am from the outer function.";

            function innerFunction() {
                console.log(outerVariable);
            }

            return innerFunction;
        }

        var closureExample = outerFunction();
        closureExample(); // Output: "I am from the outer function."
    </script>
</head>
<body>
    <!-- Open the browser console to see the output -->
</body>
</html>
```

## Scope:

The context in which variables are declared and can be accessed.

```
<!DOCTYPE html>
<html>
<head>
    <script>
        var globalVariable = "I am a global variable.";

        function exampleFunction() {
            var localVariable = "I am a local variable.";
            console.log(globalVariable); // Accessible
        }

        console.log(localVariable); // Not accessible (produces an error)
    </script>
</head>
<body>
    <!-- Open the browser console to see any error messages -->
</body>
</html>
```

## Hoisting:

Variable and function declarations are moved to the top of their containing scope during compilation.

```
<!DOCTYPE html>
<html>
<head>
    <script>
        console.log(hoistedVariable); // Undefined
        var hoistedVariable = "I am hoisted.";

        hoistedFunction(); // "I am a hoisted function."
        function hoistedFunction() {
            console.log("I am a hoisted function.");
        }
    </script>
</head>
<body>
    <!-- Open the browser console to see the output -->
</body>
</html>
```

# Bootstrap 5

Bootstrap 5 is a popular front-end framework for web development, providing a collection of pre-designed components and styles to create responsive and visually appealing websites. It includes a responsive grid system, customizable CSS styles, and JavaScript components, streamlining the design process and ensuring consistency across different devices. With its flexibility and ease of use, Bootstrap 5 empowers developers to build modern and responsive web applications efficiently.

## some common classes:

### Grid System:

- container: Creates a fixed-width container.
- container-fluid: Creates a full-width container.
- row: Defines a row in the grid.

- col-*: Defines column widths based on a 12-column grid.

## Typography:

- display-*: Adjusts font size for display headings.
- lead: Adds extra styling for lead paragraphs.
- blockquote: Styles blockquotes.

## Colours:

- text-*: Sets text color.
- bg-*: Sets background color.

## Spacing:

- m-* and p-*: Sets margin and padding.
- mt-*, mb-*, ml-*, mr-*: Sets specific margins.
- pt-*, pb-*, pl-*, pr-*: Sets specific padding.

## Borders:

- border: Adds a border to an element.
- rounded-*: Rounds corners of an element.
- border-*: Sets border color.

## Flexbox:

- d-flex: Sets display to flex.
- justify-content-*: Aligns items along the main axis.
- align-items-*: Aligns items along the cross axis.

## Buttons:

- btn: Styles a button.
- btn-*: Sets button styles (e.g., btn-primary, btn-outline-secondary).

## Forms:

- form-control: Styles form controls.
- form-check: Styles checkboxes and radio buttons.
- valid-feedback and invalid-feedback: Feedback messages for form validation.

### Navbar:

- navbar: Styles a navigation bar.
- navbar-expand-*: Defines when the navbar should expand.

### Utilities:

- visible-* and invisible: Show/hide elements based on screen size.
- sr-only: Visually hides an element but keeps it accessible.

## Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css">
   <title>Bootstrap Sample</title>
</head>
<body class="bg-light">

   <div class="container mt-5">
      <h1 class="text-center text-primary">Bootstrap Sample</h1>

      <div class="row">
         <div class="col-md-6">
            <p class="lead text-muted">A simple example demonstrating various Bootstrap
classes.</p>
            <blockquote class="blockquote">
               <p class="mb-0">Bootstrap makes web development faster and easier.</p>
               <footer class="blockquote-footer">Anonymous</footer>
            </blockquote>
         </div>

         <div class="col-md-6">
            <img src="https://via.placeholder.com/300" alt="Bootstrap Image"
class="img-fluid rounded">
         </div>
      </div>

      <div class="row mt-4">
         <div class="col-md-4">
            <div class="card">
               <div class="card-body">
                  <h5 class="card-title">Card Title</h5>
                  <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
```

```
                    <a href="#" class="btn btn-primary">Go somewhere</a>
                </div>
            </div>
        </div>

        <div class="col-md-4">
            <ul class="list-group">
                <li class="list-group-item d-flex justify-content-between align-items-center">
                    Item 1
                    <span class="badge bg-secondary">3</span>
                </li>
                <li class="list-group-item d-flex justify-content-between align-items-center">
                    Item 2
                    <span class="badge bg-secondary">8</span>
                </li>
                <li class="list-group-item d-flex justify-content-between align-items-center">
                    Item 3
                    <span class="badge bg-secondary">5</span>
                </li>
            </ul>
        </div>

        <div class="col-md-4">
            <div class="alert alert-success" role="alert">
                This is a success alert with <a href="#" class="alert-link">an example
link</a>.
            </div>
        </div>
    </div>

  </div>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

## Exercise: Create a Responsive Navigation Bar

Using Bootstrap, design a responsive navigation bar for a website. Include a brand/logo, a set of navigation links, and a responsive menu button for smaller screens. Apply Bootstrap's navigation classes to achieve a clean and mobile-friendly design.

Steps:

- Set up an HTML document with the necessary Bootstrap CDN links.
- Create a navigation bar using Bootstrap's navbar class.
- Add a brand/logo to the left side of the navigation bar using Bootstrap's navbar-brand class.

- Include a set of navigation links using Bootstrap's navbar-nav class and nav-item and nav-link classes for each item.
- Implement a responsive menu button for smaller screens using Bootstrap's navbar-toggler class.
- Ensure the navigation bar collapses into a mobile-friendly menu when viewed on smaller screens.