CSF3133: WEB-BASED INTERFACE DESIGN

Lab Modules

Wan Nural Jawahir Hj Wan Yussof, Ily Amalina Ahmad Sabri,
Rozniza Ali, Faizah Aplop, Mohd Arizal Shamsil Mat Rifin
& Wan Fatin Fatihah Yahya

CSF3133: WEB-BASED INTERFACE DESIGN

Lab Modules

Contents

List of Tables

List of Figures

Lab Module 1

INTRODUCTION TO HTML (PART 1)

HTML (HyperText Markup Language) is the standard language used to create web pages. It provides the structure for websites by defining elements such as headings, paragraphs, links, images, and more. Every website you visit on the Internet is built using HTML, which is the foundation of web development.

In this first lab, you will be introduced to the basics of HTML and learn how to create your own simple webpage. You will understand the key components of an HTML document, explore essential tags, and gain hands-on experience by writing and running your first HTML code.

HTML is essential for any web development role, and mastering these fundamentals will help you progress into more advanced topics like CSS (for styling) and JavaScript (for interactivity).

1.1 Objectives

By the end of this lab, you will have a basic understanding of:

1. Understand the structure of an HTML document.
2. Learn to use basic HTML tags for text. formatting, creating lists, adding links, and inserting images.

1.2 Pre-requisite

No prior coding experience is necessary.

1.3 Materials

1. Text editor (e.g., Visual Studio Code, Sublime Text, or Notepad++).
2. Web browser (e.g., Google Chrome, Firefox).

1.4 Structure of an HTML Document

Instruction

1. Create a new file and save it as **index.html**.
2. Add the basic structure of an HTML document.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>My First Web Page</title>
</head>
<body>
    <h1>Welcome to My First Web Page</h1>
</body>
</html>
```

3. Open the file in a web browser to see the result.

1.5 Basic HTML Tags

1.5.1 Headings

1. Add various heading levels to your document.

```
<h2>This is an H2 Heading</h2>
<h3>This is an H3 Heading</h3>
<h4>This is an H4 Heading</h4>
```

1.5.2 Paragraphs and Text Formatting

1. Learn to use paragraphs, bold, italic, and line breaks.

```
<p>This is a paragraph of text.</p>
<p>This text is <strong>bold</strong> and this is <em>italic</em>.
</p>
<p>Here is a line break:<br>This text is on a new line.</p>
```

1.5.3 Lists (Ordered and Unordered)

1. Create a numbered list and a bullet list.

```
<h2>My Favorite Fruits</h2>
<ol>
    <li>Apples</li>
    <li>Bananas</li>
    <li>Cherries</li>
</ol>

<h2>Things to Do</h2>
<ul>
    <li>Study HTML</li>
    <li>Practice coding</li>
    <li>Build a website</li>
</ul>
```

1.6 Links and Images

1.6.1 Add a Hyperlink

1. Create a clickable link to another website.

```
<p>Visit <a href="https://www.example.com" target="_blank">
Example.com</a>for more information.</p>
```

1.6.2 Insert an Image

1. Display an image on your page.

```
<img src="https://via.placeholder.com/150" alt="Placeholder Image">
```

1.6.3 Commenting in HTML

1. Learn to add comments that will not appear on the webpage.

```
<!-- This is a comment and will not be displayed in the browser -->
```

1.7 Lab Exercise 1

In this exercise, you will create a simple personal webpage using basic HTML elements. This task will help you understand the fundamental structure of an HTML document, including headings, paragraphs, lists, images, and hyperlinks.

You will learn to organize content effectively while exploring essential HTML tags. This exercise aims to strengthen your understanding of the structure of a webpage and ensure that you can build a basic yet well-structured personal webpage.

1.7.1 Designing Simple Personal Webpage

Instruction

1. Create a simple webpage about yourself with:
   - A title in the browser tab.
   - A heading introducing your name.
   - A short paragraph about your hobbies with some text in bold and italic.
   - An ordered list of 3-5 favorite movies or books.
   - An unordered list of your top 3 favorite hobbies.
   - A link to a relevant website (for example, a hobby-related page).
   - An image of something related to your hobbies.
2. Use comments to explain each section of your code.

Submission

Save your HTML file as **lab1.html** and submit it through the learning platform.

1.7.2 Rubric

Table 1.1: Rubric for Lab Exercise 1

Criteria	Description	Marks
HTML Structure	Proper use of `<!DOCTYPE>`, `<html>`, `<head>`, `<body>` tags. Includes a title and basic structure.	3
Text Formatting	Correct usage of text formatting tags like `<h1>`, `<p>`, ` `, and `<strong>`.	2
Lists	Correct creation of ordered `<ol>` or unordered `<ul>` lists with appropriate list items `<li>`.	2
Links	Proper creation of hyperlinks using `<a>` tag with functioning href attribute.	2
File and Code Organization	Code is well-organized and indented. File is named and saved correctly (e.g., lab1.html).	1

Lab Module 2

INTRODUCTION TO HTML (PART 2)

In this lab, we will build on the foundational knowledge you acquired in Lab 1 by diving deeper into more advanced HTML features. You will learn how to create tables to display structured data, build forms for user input, and explore semantic HTML tags that give your web pages more meaningful structure.

Tables allow you to organize information in rows and columns, which is essential for presenting data clearly. Forms, on the other hand, are a key component for interactive websites where users can input data like text, selections, or file uploads. Lastly, semantic HTML improves the readability of your code and ensures that the structure of your web page is clear, making it easier for both developers and search engines to understand.

2.1 Objectives

By the end of this lab, you will have a basic understanding of:

1. Creating and formatting tables in HTML.
2. Building interactive forms with various input types.
3. Utilizing semantic HTML tags to improve the structure of your web pages.

2.2 Pre-requisite

Completion of Lab 1 or basic knowledge of HTML structure and tags.

2.3 Working with Tables

1. Basic Table Structure: Create a table with rows and columns.

```
<table>
    <tr>
        <th>Name</th>
        <th>Subject</th>
        <th>Grade</th>
    </tr>
    <tr>
        <td>John Doe</td>
        <td>Mathematics</td>
        <td>A</td>
    </tr>
    <tr>
        <td>Jane Smith</td>
        <td>Science</td>
        <td>B+</td>
    </tr>
</table>
```

2. Attributes in Tables: You can add more features to your table by using attributes:

- `border`: Adds a border to your table.
- `colspan` and `rowspan`: Merge columns or rows.

```
<table border="1">
    <tr>
        <th colspan="2">Student Information</th>
    </tr>
    <tr>
        <td>Name</td>
        <td>John Doe</td>
    </tr>
    <tr>
        <td colspan="2">Mathematics A+</td>
    </tr>
</table>
```

2.4 Creating Forms

Forms are an essential part of any interactive webpage, allowing users to input data.

2.4.1 Basic Form Elements

1. Create a form with text fields, radio buttons, and a submit button.

```
<h2>Registration Form</h2>
<form action="submit_form.php" method="post">
    <label for="fname">First Name:</label><br>
    <input type="text" id="fname" name="firstname"><br><br>

    <label for="lname">Last Name:</label><br>
    <input type="text" id="lname" name="lastname"><br><br>

    <label for="gender">Gender:</label><br>
    <input type="radio" id="male" name="gender" value="male">
    <label for="male">Male</label><br>
    <input type="radio" id="female" name="gender" value="female">
    <label for="female">Female</label><br><br>

    <input type="submit" value="Submit">
</form>
```

2.4.2 Other Form Elements

1. Include other types of inputs like checkboxes and dropdowns.

```
<form action="submit_form.php" method="post">
    <label for="hobby">Select your hobbies:</label><br>
    <input type="checkbox" id="hobby1" name="hobby1"
    value="Reading">
    <label for="hobby1"> Reading</label><br>
    <input type="checkbox" id="hobby2" name="hobby2"
    value="Gaming">
    <label for="hobby2"> Gaming</label><br><br>

    <label for="country">Choose a country:</label>
    <select id="country" name="country">
        <option value="usa">USA</option>
```

```
            <option value="canada">Canada</option>
            <option value="uk">UK</option>
        </select><br><br>

        <input type="submit" value="Submit">
    </form>
```

2.5 Semantic HTML

Semantic HTML provides meaning to the content, making the structure of the webpage clearer.

2.5.1 Common Semantic Tags

- `<header>`: Represents the header section of a webpage.
- `<nav>`: Contains navigation links.
- `<article>`: Denotes a self-contained piece of content.
- `<section>`: Groups content into thematic sections.
- `<footer>`: Represents the footer section of a webpage.

Example of semantic HTML.

```
<header>
    <h1>Welcome to My Web Page</h1>
</header>

<nav>
    <a href="#home">Home</a> | <a href="#about">About</a> |
    <a href="#contact">Contact</a>
</nav>

<section>
    <article>
        <h2>Article Title</h2>
        <p>This is an article about web development.</p>
    </article>
</section>
```

```
<footer>
    <p>Copyright © 2024 My Web Page</p>
</footer>
```

2.6 Lab Exercise 2

In this exercise, you will apply your knowledge of HTML by creating a structured webpage that includes various fundamental elements of web design. The goal is to reinforce your understanding of tables, forms, and semantic HTML, which are essential building blocks for creating accessible, organized, and user-friendly webpages. You will build a personal profile page that displays information using a table, collects user input through a form, and uses semantic tags to create a meaningful structure.

2.6.1 Building a Personal Profile Page

Instruction

In this task, you need to:

1. Create a Table: Build a table with at least three rows and columns to display personal information (e.g., Name, Age, Occupation, or any other attributes of your choice).
2. Design a Simple Form: Create a form that collects basic user information, including name, email, gender, and favorite hobbies. Include text input, radio buttons, checkboxes, and a submit button.
3. Use Semantic HTML : Structure a simple webpage using semantic tags like `<header>`, `<nav>`, `<section>`, `<article>`, and `<footer>`.

Submission

Save your HTML file as **lab2.html** and submit it through the learning platform.

2.6.2 Rubric

Table 2.1: Rubric for Lab Exercise 2

Criteria	Description	Marks
Table Creation	Table includes three rows and columns, with meaningful sample data and headers.	2
Form Structure	Form contains required fields (text inputs, radio buttons, checkboxes, submit button).	3
Semantic HTML	Proper use of `<header>`, `<nav>`, `<section>`, `<article>`, and `<footer>` tags.	3
Code Organization and Readability	Code is well-organized, properly indented, and easy to read.	2

Lab Module 3

HTML5

This lab will introduce you to more advanced concepts in HTML and HTML5. You will learn about the difference between block and inline elements and how to use them to structure content effectively. We will also explore the `<div>` tag to organize and style web pages better. Additionally, you will learn about HTML entities, which allow you to include special characters in your web content. The lab will cover HTML5 form elements that provide enhanced functionality for web forms, including the use of the `<output>` element to display the results of calculations or actions directly on the page.

3.1 Objectives

By the end of this lab, you will have a basic understanding of:

1. Differentiate between block and inline elements.
2. Use `<div>` for better web page organization.
3. Apply HTML entities to display special characters.
4. Implement various HTML5 form elements.
5. Utilize the `<output>` element to display dynamic results.

3.2 Block and Inline Elements

- HTML elements basically consist of block or inline elements. Block elements are elements that are displayed with a new line in the browser. The block elements you have used are like `<h1>` …`<h6>`, `<p>`, `<ul>`, `<ol>` and

<table>. While inline elements are elements that are displayed on the same line in the browser. Examples are <b>, <d>, <td>, <tr> and <img>.

- An element is a block element that can be used to group HTML elements in a group. It will simplify the decorating process and set the style of a website. is widely used in CSS topics that you will learn later. Using the element, each element displayed will have blank rows before and after the display.
- In addition, the element is specially used to collect text elements. Technically, and have no specific purpose except for the purpose of decorating HTML elements.

Instructions

1. Open your text editor and create a new HTML file named **block.html**. Inside <body>, create a section with examples of block elements:

```
<h1>Block Elements</h1>
<div>This is a block element</div>
<p>This is a paragraph (block element)</p>
<ul>
  <li>List item 1</li>
  <li>List item 2</li>
</ul>
```

2. Below the block elements, add a section for inline elements :

```
 <h2>Inline Elements</h2>
<span>This is an inline element</span>
<a href="#">This is a link (inline element)</a>
<strong>This is bold text (inline element)</strong>
```

3. Save **block.html** and open it in your browser to observe the layout differences between block and inline elements.

3.3 Using to Organize Content

- Previously, you used <table> to arrange the form. Apart from <table>, can also be used for the same purpose. All you have to do is set the style for the blocks created. The activity below is to create a website consisting of headers, menus, content and footers.

Instructions

1. Open a new HTML file and name it **div.html**.
2. Inside `<body>`, use `<div>` elements to separate the webpage into sections:

```
<div id="header">Header Section</div>
<div id="nav">Navigation Bar</div>
<div id="content">Main Content Area</div>
<div id="sidebar">Sidebar Content</div>
<div id="footer">Footer Section</div>
```

3. Save **div.html** and open it in a browser to view the structured sections.

- In this activity, there are many attributes for the style you use such as `background-color`, `height`, `width`, `text-align` and `float`. Try modifying the values of these attributes and note the results. Don't worry about the messy output as long as you understand what the attributes used are doing.

3.4 Understanding HTML Entities

- Entities are used to display reserved symbols in HTML. For example, the symbols `<` and `>` are reserved symbols and we cannot write as follows so that the `<` sign is displayed on the browser:

```
<html>
<body>
This is a sign '<'
</body>
</html>
```

- This is because the browser will assume that the `<` symbol is part of the HTML tag. Therefore, entities should be used to display reserve symbols in HTML code.
- Entities can be used according to the name or number specified. For example, the symbol `<` can be represented by `&lt;` or `&#60;`. The following table shows some of the HTML entities that can be used:

Table 3.1: HTML Entities

Symbol	Entities Name	Entities Symbol
£	&pound;	&#163;
€	&euro;	&#8364;
©	&copy;	&#169;
®	&reg;	&#174;
<	&lt;	&#60;
>	&gt;	&#62;
Blank space without a newline		

- Note: Entity names use lowercase letters as a whole and are case-sensitive. Do a search on the internet for a full list of usable entities.

Instructions

1. Create a file name **entity.html**.

2. Type the code below into the newly created file.

```
<html>
<body>
  <h1> Contoh Penggunaan Entiti HTML </h1>
  <p>
  <!-- Entiti menggunakan nama -->
  Tag html terdiri daripada simbol &lt; dan &gt;
  </p>
  <hr>
  <!-- Entiti menggunakan nombor -->
  Website ini adalah hakmilik NAMAANDA &#169;
  </hr>
</body>
</html>
```

3. Save the file and click on the file icon to see the display results.

4. Do a search on the search engines for a list of other entities and experiment with those entities.

3.5 Hypertext Mark Up Language 5 (HTML5)

Before we learn HTML5 in more depth, we should first get to know the existing HTML versions. As we learned in the lecture, HTML has various versions that change from year to year. In the early stages of the emergence of HTML, it was only known as HTML without any version number at the end. However, starting in 1995, HTML was first given a version number with the name HTML2.0, followed by HTML3.2 (1997), HTML4.01 (1999) and most recently, HTML5 (2012).

In addition to HTML, there is another version called XHTML. XHTML is an improved version of HTML4.01 by W3C in 2000. So what is the difference between HTML and XHTML? Simply put, XHTML is a more strict version of HTML where XHTML writing has to comply with certain requirements. Among them:

- Every written Web document must be declared with `<!DOCTYPE>`. Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional
.dtd">
```

- XHTML tag pair block blocks need to be sorted correctly. For example:

```
<b><i> This text is bold and tilted.</i></b>
```

  This is because in the previous HTML, the following order is still being processed correctly by the browser:

```
<b><i> This text is bold and tilted.</b></i>
```

- Every XHTML element needs to be closed including an empty element. For example, `<br>` should be written as `<br/>`.

- XHTML elements need to be written in lowercase. For example:

```
<p></p> instead of <P> </P>
```

- Attribute names also need to be written in lowercase. For example, `<table width ="70%">` instead of `<table WIDTH ="70%">`

- The value for the attribute needs to be written inside the quote. For example, `<table width ="70%">` instead of `<table width = 70%>`

In addition, XHTML was introduced to support XML-based HTML writing where, if HTML requires new tags, then it can be declared in a Document Type Declaration (DTD) file. In XHTML web documents, the DTD file in question can be accessed at http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd. Discussions on XML and DTD will be in the next topic.

So, what is the appropriate version of HTML to learn and use? As of this writing, it is HTML5. HTML5 is the latest version of HTML that to some extent uses the features available in XHTML. Furthermore, it is considered easier than the use of XHTML which is considered too strict for some website developers.

HTML5 was produced as a result of a collaboration between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG) which started in 2006. Among the advantages of HTML5 are:

- Reduce the need for external plugins (such as Flash).
- Improved error handling.
- More tags to replace scripts (JavaScript for example).
- Ability to work on multiple devices. For example, smartphones, tabs and so on.
- A simpler `<!DOCTYPE>` declaration of `<!DOCTYPE html>` than previous HTML or XHTML versions.

The lab activities you have done before are using the original HTML version which is also the basis of the latest HTML versions. Therefore, the next activities are based on the latest version of HTML, HTML5.

Instructions

1. Create a new file named **intro5.html**.
2. Type html code below:

```
<!DOCTYPE>
<html>
    <head>
    <title> Pengenalan Kepada HTML</title>
    </head>
    <body>
    <h1> HTML5 </h1>
    <p> Saya suka HTML5 </p>
    <br>
```

```
        -Nama Anda
        </body>
    </html>
```

3. Save the newly created file and double-click on the file icon to view the output. You will find that the browser gives the correct display on the screen. However, does it comply with the rules for HTML5? To be sure, we can use the free service offered by W3C at http://validator.w3.org/.

4. Click on the **'Validate by File Upload'** tab.

5. Click on the **'Choose File'** button and select the previously created **intro5.html** file. Press the **'Validate'** button. What display did you find? Does your HTML5 file comply with the requirements?If not, what are the mistakes made? Please note in the conclusion section.

6. Fix the previous HTML code with the following code:

```
<!DOCTYPE html>
```

7. Upload the file to http://validator.w3.org/ and make sure your HTML code has passed the validation test. Note: If you see a **'warning'**, just ignore it.

3.6 Form Elements in HTML5

- The HTML elements previously studied can still be used in HTML5. The most important thing to remember is that the use of `<!DOCTYPE html>` is mandatory in every document that uses HTML5. In addition, some of the HTML5 elements that will be used in subsequent activities are not supported by all browsers.

- Among the significant improvements made in HTML5 are related to form. There are three new elements to the form introduced namely `<datalist>`, `<keygen>` and `<output>`.

- The `<datalist>` tag is used to offer autocomplete features to the user when filling out a form. This will save time for filling in the blanks on the form.

- The `<keygen>` tag is used to increase security during the verification process of users who want to log into a web system. When the form is sent to the server, there are two keys that will be generated, namely the public key and the private key. The Public Key will be sent to the server while the private key will be stored locally.

This public key will be used to generate a digital certificate (digital certificate) that will be used in communication between the browser and the server in the future. If you would like more information on how Public Key Infrastructure works, please do a search on the internet.

- Note: Because there are still tags that are not supported by all browsers. Please note the browser type used. If the activity requires Mozilla, use Mozilla, if the activity requires Chrome, please use Chrome. However, it is not a mistake if you want to experiment with whether a tag is supported by the browser or not.

3.6.1 Input Elements in HTML5

- Inside a form, there are various types of input boxes that can be used to get input from users. Among the types of inputs offered in HTML5 are as follows:
  - number
  - range
  - search
  - color
  - date
  - datetime
  - datetime-local
  - email
  - month
  - search
  - tel
  - time
  - url
  - week
- The input classification provided by HTML5 will facilitate the provision of space and validation of inputs on a website. However, not all HTML5 input types are currently supported by web browsers.

Instruction

1. Create new file named **input5.html**.

2. Enter the following code into the newly created file:

```
<!DOCTYPE html>
<html>
<body>
  <form action="">
    Pilih warna kegemaran anda: <input type="color"
    name="warnapilihan">
    <br>
    Carian: <input type="search" name="carian">
    <br>
    Tarikh: <input type="date" name="tarikh">
    <br>
    Masa: <input type="time" name="masa">
    <br>
    Bulan dan Tahun: <input type="month"
    name="bulandantahun">
    <br>
    Laman Web: <input type="url" name="lamanweb">
    <br>
    Minggu: <input type="week" name="minggu">
    <br>
    Email: <input type="email" name="useremail">
    <br>
    <input type="submit" value="Hantar">
  </form>
</body>
</html>
```

3. Save and view the output of files you create using the Chrome browser.

4. Test each of the input fields by filling out the form that has been created. For the Website and Email space, try experimenting with the following data:

   - Website: urlsaya
   - Email: my email

   Notice what happens when you click the **'Submit'** button.

3.6.2 Attributes for Input HTML5

- `placeholder` is a text box that has shadowed text when no value is entered and not focused. The term 'focus' refers to the cursor that flashes in a text box.
- `autofocus` is an attribute used to make a text box focused as soon as the website is finished displaying in the browser.
- `required` is an attribute used to require an input space to be filled with a value before being sent to a browser.

Instruction

1. Create a new file named **inattribute5.html**.
2. Enter the following code into the newly created file:

```
<!DOCTYPE html>
<html>
<body>
  <form action="">
    Nama Pelajar: <input type="text" name="namaPelajar"
    placeholder="Masukkan nama pelajar di sini."
    autofocus autocomplete>
    <br>
    Nama Bapa: <input type="text" name="namaBapa"
    placeholder="Masukkan nama bapa pelajar di sini."
    required>
    <br>
    <input type="submit" value="Simpan">
  </form>
</body>
</html>
```

3. Save and view the output of files you create using Chrome or Mozilla browsers.
4. Note the output displayed and note the position of the cursor. Which text box has a flashing cursor? Fill in the '**Amen**' value in the '**Student Name**' box and leave it blank in the '**Father Name**' column. Then, click on the '**Save**' button. What happened?
5. Refresh your web files and make sure both spaces are empty. Then, enter the letter '**A**' into the '**Student Name**' box. What did you find? Does autocomplete work?

3.6.3 HTML5 <output> element

The <output> tag is used to display the output of a calculation usually performed using a script. This will make it easier for website builders to place computational output anywhere on the website view they build.

Instruction

1. Create a new file named **output5.html**.

2. Type HTML code below:

```
<!DOCTYPE html>
<html>
<body>
  <form oninput="z.value=parseInt(x.value)+
  parseInt(y.value)">
    <input id="x" type="range" min="0" max="10" step="1"
    value="0"> 0 <br> +
    <input id="y" type="number" value="10">
    <br>=
    <output name="z" for="x y"></output>
  </form>
</body>
</html>
```

3. Save the newly created file. Click on the file icon to see the output. View the output of files you created using Mozilla or Chrome browsers. This element is not currently supported by Internet Explorer.

In the <form> element, oninput refers to which event is in the code above. When the user enters the input, then the operation in "`z.value +_`" will be executed. The <input> type range element is the element used to display the slider on HTML5 pages. In the above activity, the meaning of the attributes used are as follows:

- `min` = smallest value on the slider
- `max` = largest value on slider
- `step` = increase value on slider
- `valu e` = default value (default) on the slider.

The second <input> element is a number type. It will display a list of numbers. Each input element has an id as a reference for any upcoming operation.

For the <output> element, the values in the for attribute refer to the id involved with the calculation. For example for = "x y" refers to the id x and y used in the addition operation as in the example above.

3.7 Lab Exercise 3

A web contact form is an essential feature of modern websites, which allows users to submit inquiries, feedback, or requests easily. In this task, we will build a simple contact form using HTML integrating an inline calculator to enhance user interactivity.

This exercise combines the key concepts from Lab 3, including block and inline elements, for organizing content, HTML entities, HTML5 form elements, and the <output> element. This will allow students to reinforce their understanding of each topic in a practical context.

3.7.1 Building an Interactive Contact Form with an Inline Calculator

Instruction

1. **Create the HTML File:**
    - Name the file **lab3.html**.
2. **Define the Structure:**
    - Organize the page using elements:
        - A header section for the website title.
        - A content section containing the contact form and calculator.
        - A footer section with contact information.
3. **Header Section:**
    - Inside a , create a header that displays the title of the webpage using a block element like <h1>.
    - Use inline elements , such as , within the header to add a slogan or subtitle.

4. **Content Section:**

   - Inside a second `<div>`, create a contact form with HTML5 elements:
     - Email input for user contact.
     - Date input for the user's preferred contact date.
     - Range slider for user satisfaction level.
     - Add a submit button.
   - Use a third `<div>` within the content section for an inline calculator:
     - Provide two `<input>` fields where users can enter numbers.
     - Use the `<output>` element to show the sum of the two numbers dynamically.

5. **Footer Section:**

   - Inside a final `<div>`, create a footer section containing contact information.
   - Use an HTML entity, like `&copy;`, to display the copyright symbol.

3.7.2 Rubric

Table 3.2: Rubric for Lab Exercise 3

Criteria	Excellent (5)	Good (4)	Average (2-3)	Needs Improvement (0-1)
Page Structure	Uses `<div>` elements effectively for layout and organizes each section logically.	Uses `<div>` elements effectively but with minor layout issues.	Uses `<div>` elements with some logical structure but lacks refinement.	Poor organization or does not use `<div>` elements appropriately.
Block and Inline Elements	Correctly demonstrates block vs. inline elements in layout and content.	Demonstrates block vs. inline elements with minor mistakes.	Attempts to use block and inline elements , but lacks clarity.	Does not demonstrate understanding of block vs. inline elements .

Criteria	Excellent (5)	Good (4)	Average (2-3)	Needs Improvement (0-1)
HTML Entities	Correctly includes HTML entities in appropriate locations.	Includes HTML entities with minor errors.	Attempts to use HTML entities but with noticeable mistakes.	Fails to include or misuses HTML entities.
HTML5 Form Elements	Includes email, date, and range inputs and organizes form effectively.	Includes email, date, and range inputs but minor layout issues.	Includes some form elements but lacks full variety or layout.	Lacks required form elements or has significant layout issues.
Use of `<output>` Element	Correctly implements the `<output>` element for calculator functionality.	Implements `<output>` element with minor issues.	Attempts `<output>` element but lacks full functionality.	Fails to implement or incorrectly uses <output> element.

Lab Module 4

CASCADING STYLE SHEETS (CSS) - PART 1

In Lab 4, we will explore how HTML5 and CSS work together to create structured, styled, and visually appealing web pages. While the focus of the lab is primarily on CSS, we will also continue building on two critical HTML5 topics to ensure a comprehensive understanding of how these elements integrate into modern web design.

4.1 Objectives

By the end of this lab, students will be able to:

1. Utilize semantic HTML5 elements to create a well-structured and meaningful webpage layout.
2. Embed video content using the `<video>` tag.
3. Demonstrate an understanding of CSS syntax, including the use of selectors, properties, and values to style HTML elements effectively.
4. Differentiate between ID and class selectors, and apply them appropriately to target specific elements for styling.
5. Build a functional and visually appealing navigation bar using CSS for layout and styling.
6. Use CSS properties to enhance text presentation, including font styling, alignment, and decoration.

7. Recognize the different states of links (e.g., hover, visited, active) and style them using CSS to improve user experience.

4.2 Page Structure Elements

The creation of HTML5 is to support the concept of semantic web. In the semantic web, tags not only serve to rank content on the browser alone, but it also serves to describe the type of content itself. For example, the `<article>` tag explains that the content contained between the opener and the closing tag is an article. That way, the browser can find out type of content displayed.

Instruction

1. Create an HTML file called **structure.html**.

2. Type the HTML5 code as below:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Struktur Halaman</title>
</head>
<body>
<header>Ini adalah header. Gambar juga boleh diletakkan di sini.
</header>
<section id="1">
<h2>Ini adalah seksyen 1.</h2>
</section>
<section id="2">
<h2>Ini adalah seksyen 2.</h2>
</section>
<section id="3">
<h2>Ini adalah seksyen 3.</h2>
<p>Berikut adalah contoh penggunaan meter:</p>
<meter min="0" max="100" value="20">100</meter>
</section>
<footer>
&copy; Hakcipta terpelihara.
<address>
Hubungi kami di <a href="mailto:emailanda@umt.edu.my">
```

```
email saya</a>.
</address>
</footer>
</body>
</html>
```

3. Save the **structure.html** file and open the file using the Chrome browser. Note the resulting output.

Note: There are many more page elements you can explore. Refer to your textbook or website and experiment with those elements.

4.3 Insert Video into HTML5 Document

The tag used to insert a video into an HTML5 document is `<video>` .By using this tag as well, errors for browsers that do not support video can be displayed to the user. The best practice of using video elements is by setting the attribute width and height so that the browser can allocate enough space for the video display. To date, there are 3 video formats supported by browsers namely MP4, WebM and Ogg.

Instruction

1. Create an HTML file called **video.html**

2. Type the HTML5 code as below:

```
<!DOCTYPE html>
<html>
<body>
<video width="400" height="300" controls>
  <source src="upinipin.mp4" type="video/mp4">
  Pelayar anda tidak menyokong tag video.
</video>
</body>
</html>
```

3. Download the video **upinipin.mp4** from ePembelajaran. Make sure the video file is in the same location as the **video.html**.

4. Save the file and open the **video.html** file using the Chrome browser.

Note the resulting output. Modify the width and height values to see the space differences provided by the browser. Chrome supports mp4 format videos but not Mozilla. Do an internet search to find video formats supported by browsers.

4.4 Cascading Style Sheets

- CSS is known as Cascading Style Sheets (CSS).
- CSS determines how to display HTML elements better and organized and so on in troubleshooting.
- Using CSS, the look and layout of the website can be changed by just editing one file.
- CSS has two main parts, namely `"selector"` and `"declaration"`.
- `"selector"` is an HTM element to be styled. Each `"declaration"` has `"property"` and `"value"`. `"property"` is an attribute style to be changed and there is a `"value"` for each `"property"`.

Instruction

1. Create an HTML file called **stail.html**.
2. Type the HTML and CSS code as below:

```
<html>
<head>
<style type="text/css">
p {
   color:red;
   text-align:center;
}
</style>
</head>
<body>
<p>Hello UMT!</p>
</body>
</html>
```

3. Save the file and click on the icon to view the output.

Note: The syntax for this CSS Code is: `[p color: red; text-align: center;]` p is "selector", color & text-align is "property", red & center is "value".

4.5 Recognize ID and Selector Class

- Selector ID is used to determine the unique style of each element.
- The selector ID uses HTML ID element attribute and is defined as "#".
- Selector classes are used to determine styles for elements in a group. This allows the style to be set according to specific HTML elements with the same class.
- The Selector class uses HTML class attributes and is defined by "."

Instruction

1. Create an HTML file called **stailID.html**.
2. Type the HTML and CSS code as below:

```
<html>
<head>
<style type="text/css">
#Part1 {
   text-align: center;
   color: red;
}
</style>
</head>
<body>
<p id="Part1">Hello UMT!</p>
</body>
</html>
```

3. Save the file and click on the icon to view the output.
4. Create a new HTML file called **Class.html** style

5. Type the HTML and CSS code as below:

```
<html>
<head>
<style type="text/css">
.center{
  text-align: center;
}
</style>
</head>
<body>
<h1 class="center">Tajuk</h1>
<p class="center">Perenggan</p>
</body>
</html>
```

6. Save the file and click on the icon to view the output.

Note: Based on the CSS code example above, all HTML elements with class = "center" will be positioned in the center of the page.

4.6 Navigation Bar using CSS

Navigation Bars are easy to use and are important in websites. With CSS, the HTML menu can be changed to better navigation.

Instruction

1. Create an HTML file called **navigation.html**.

2. Type the CSS code as below:

```
<html>
<head>
<style type="text/css">
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
a:link, a:visited {
  display: block;
```

```
    font-weight: bold;
    color: #FFFFFF;
    background-color: #98bf21;
    width: 120px;
    text-align: center;
    padding: 4px;
    text-decoration: none;
    text-transform: uppercase;
}
a:hover, a:active {
    background-color: #7A991A;
}
</style>
</head>
<body>
<ul>
    <li><a href="#home">Halaman Utama</a></li>
    <li><a href="#news">Berita Terkini</a></li>
    <li><a href="#contact">Hubungi Kami</a></li>
    <li><a href="#about">Mengenai Kami</a></li>
</ul>
</body>
</html>
```

3. Save the file and click on the icon to view the output.

4.7 Recognizing Text Decoration

Property text-decoration is used to set or remove decorations from text. Text-decoration properties are mostly used to remove lines from links (hyperlinks) for the purpose of improving website design.

Instruction

1. Create an HTML file called **Teks.html**.

2. Type the HTML and CSS code as below:

```
<html>
<head>
<style type="text/css">
h1 {text-decoration: overline;}
h2 {text-decoration: line-through;}
h3 {text-decoration: underline;}
h4 {text-decoration: blink;}
</style>
</head>
<body>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<p><b>Note:</b> The "blink" value is not supported in IE,
Chrome, or Safari.</p>
</body>
</html>
```

3. Save the file and click on the icon to view the output.

4.8 Recognizing Styling Links

Links can be styled with any CSS Property (e.g. color, font-family, background, etc.). The four types of links are as follows:

- a:link - links are normal, and unvisited
- a:visited - the user-visited link has been visited
- a:hover - hover-link when clicked
- a:active - active-link when clicked

Instruction

1. Create a file called **staillLink.html**.
2. Type the HTML and CSS code as below:

```
<html>
<head>
<style type="text/css">
a:link {color:#FF0000;} /* unvisited link */
a:visited {color:#00FF00;} /* visited link */
a:hover {color:#FF00FF;} /* mouse over link */
a:active {color:#0000FF;} /* selected link */
</style>
</head>
<body>
<p><b><a href="default.asp" target="_blank">
Ini adalah pautan</a></b></p>
</body>
</html>
```

3. Save the file and click on the icon to view the output.

4.9 Lab Exercise 4

In this exercise, you will build a functional webpage that integrates various HTML5 and CSS concepts. You will apply semantic HTML5 elements to structure content effectively, incorporate embedded media for enhanced interactivity, and use CSS to create a visually appealing design.

This lab aims to reinforce your understanding of modern web development principles while improving your ability to design professional and user-friendly webpages. By the end of this task, you will have a well-structured and styled webpage that demonstrates your ability to combine HTML5 and CSS effectively.

4.9.1 Building a Responsive Webpage with HTML5 and CSS

Instruction

1. **Create the HTML File:**
   - Name the file **lab4.html**.
2. **Define the Structure:**
   - Use semantic HTML5 elements such as `<header>`, `<nav>`, `<section>`, `<article>`, and `<footer>` to structure your webpage.

- Create a navigation bar using an unordered list styled with CSS. Include links to various sections of the page.

3. **Insert Media**
   - Embed a video in one section of the webpage.
   - Ensure the video is properly captioned and fits the design of your page.
4. **Apply CSS Styling**
   - Use both internal and external CSS to style the page.
   - Apply text decoration to headings and links using CSS properties such as `text-decoration`: underline, overline, line-through, and others.
   - Create a responsive design by styling your links for various states (`:hover`, `:visited`, `:active`).
5. **Interactive Elements**
   - Include a form with the following fields:
     - Name (Text Input)
     - Email (Text Input)
     - Gender (Radio Buttons)
     - Favorite Hobbies (Checkboxes)
     - Submit Button
   - Style the form using CSS.

Submission Requirements

- A single HTML file linking to an external CSS file.
- The webpage must follow good design practices and be fully functional in modern browsers.

4.9.2 Rubric

Table 4.1: Rubric for Lab Exercise 4

Criteria	Excellent (5)	Good (4)	Average (2-3)	Needs Improvement (0-1)
HTML5 Structure	All semantic tags are correctly used. Navigation and structure are clear and logical.	Semantic tags are mostly used correctly. Structure is good.	Some semantic tags used incorrectly or missing.	Minimal or no use of semantic tags.
Navigation Bar	Well-designed and fully functional with clear navigation links.	Functional but design could be improved.	Navigation bar present but not fully functional.	Navigation bar missing or poorly implemented.
Video Embedding	Video is correctly embedded, properly captioned, and responsive.	Video embedded and functional, minor styling issues.	Video embedded but lacks captions or proper styling.	No video or improperly embedded.
Form Design & Function	Form is complete, well-structured, and fully styled.	Form is functional with minor styling issues.	Form is present but missing some elements.	Form is incomplete or missing.
CSS Styling	Effective use of internal and external CSS . Creative and consistent styling applied.	CSS applied but lacks creativity or consistency.	Some CSS applied but minimal effort shown.	Little or no CSS applied.
Text Decorations	Proper and creative use of text-decoration across headings and links.	text-decoration used correctly but without creativity.	Minimal use of text-decoration.	No use of text-decoration.
Interactive Link States	Link states (`:hover` , `:visited` , `:active`) fully functional and styled creatively.	Link states are functional but need more styling.	Link states applied with minimal effort.	No link states applied.

continue...

Criteria	Excellent (5)	Good (4)	Average (2-3)	Needs Improvement (0-1)
Responsiveness & Design	Page is fully responsive and visually appealing.	Page is mostly responsive with minor design issues.	Limited responsiveness, design is basic.	Not responsive, poor design.
CSS Styling	Effective use of internal and external CSS . Creative and consistent styling applied.	CSS applied but lacks creativity or consistency.	Some CSS applied but minimal effort shown.	Little or no CSS applied.
Text Decorations	Proper and creative use of text-decoration across headings and links.	text-decoration used correctly but without creativity.	Minimal use of text-decoration.	No use of text-decoration.
Interactive Link States	Link states (`:hover` , `:visited` , `:active`) fully functional and styled creatively.	Link states are functional but need more styling.	Link states applied with minimal effort.	No link states applied.
Responsiveness & Design	Page is fully responsive and visually appealing.	Page is mostly responsive with minor design issues.	Limited responsiveness, design is basic.	Not responsive, poor design.

Lab Module 5

CASCADING STYLE SHEETS (CSS) - PART 2

5.1 What is CSS?

CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language. The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to different environments. This is referred to as the separation of structure (or: content) from presentation.

Style Sheets can format the HTML document. HTML documents may contain style sheet rules directly in them or they may import style sheets. There are three ways of providing styling information for the Web browsers.

5.2 Using CSS

CSS can be added to HTML documents in 3 ways:

- Inline - by using the style attribute inside HTML elements
- Internal - by using a `<style>` element in the `<head>` section
- External - by using a `<link>` element to link to an external CSS file

The most common way to add CSS, is to keep the styles in external CSS files.

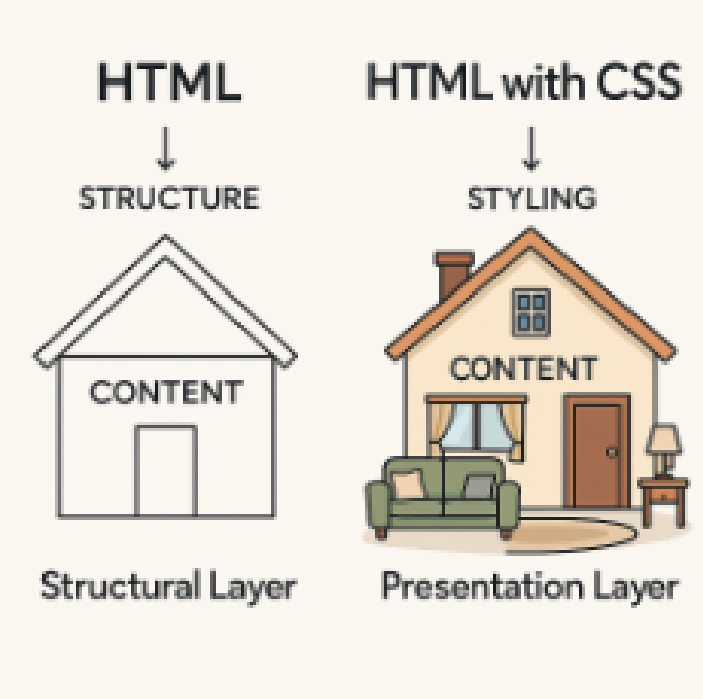

Figure 5.1: HTML vs CSS

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts:

- **Selector** – A selector is an HTML tag at which a style will be applied. This could be any tag like `<h1>` or `<table>` etc.
- **Property** – A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.
- **Value** – Values are assigned to properties. For example, color property can have value either red or `#F1F1F1` etc.

You can put CSS Style Rule Syntax as follows:

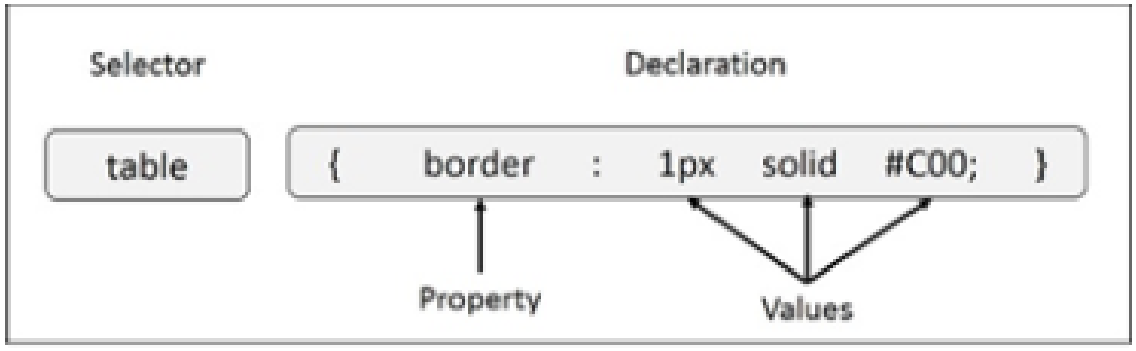

Figure 5.2: CSS Style Rule Syntax

5.3 Inline CSS

Instruction

1. Create an HTML file called **inline.html**

2. Type the HTML5 code as follows:

```
<!DOCTYPE html>
<html>
<head>
    <title>Inline CSS</title>
</head>
<body>
    <h1 style="color: blue; font-size: 24px;">
    Welcome to Inline CSS!</h1>
    <p style="background-color: yellow; padding: 10px;">
    This paragraph is styled using Inline CSS.</p>
</body>
</html>
```

3. Save the **inline.html** file and open the file using the Chrome browser. Note the resulting output.

5.4 Internal CSS

Instruction

1. Create an HTML file called **internal.html.**

2. Type the HTML5 code as below:

```
<!DOCTYPE html>
<html>
<head>
    <title>Internal CSS}</title>
    <style>
        h1 {
            color: green;
            text-align: center;
        }
        p {
```

```
            background-color: #f0f0f0;
            padding: 10px;
        }
    </style>
</head>
<body>
    <h1>Welcome to Internal CSS!</h1>
    <p>This paragraph is styled using Internal CSS.</p>
</body>
</html>
```

3. Save the **internal.html** file and open the file using the Chrome browser. Note the resulting output.

5.5 External CSS

Instruction

1. Create a CSS file called **style.css**.
2. Type the CSS code as below:

```
h1 {
    color: red;
    font-family: Arial, sans-serif;
}
p {
    font-size: 16px;
    line-height: 1.5;
}
```

3. Save the **style.css** file
4. Create an HTML file called **external.html**.
5. Link the HTML with CSS as below:

```
<!DOCTYPE html>
<html>
<head>
    <title>External CSS}</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
```

```
</head>
<body>
    <h1>Welcome to External CSS!</h1>
    <p>This paragraph is styled using External CSS.</p>
</body>
</html>
```

6. Save the **external.html** file and open the file using the Chrome browser. Note the resulting output.

5.6 Link to file CSS (from another website)

To use a CSS file hosted on another website, you can use the `<link>` attribute with the full URL of the CSS file as a reference.

- **Advantages:**
  - Eliminates the need to download the CSS file to your local directory.
- **Disadvantages:**
  - Depends on the availability of the hosting website. If the website is unavailable, the linked styles will not be applied.
  - The hosting website may restrict the use of its CSS files on other domains (due to CORS restrictions).

Instruction

1. Create an HTML file called **link.html**
2. Link the HTML file to an external CSS file hosted online using the following code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Google Fonts Example</title>
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;
    700&display=swap" rel="stylesheet">
    <style>
        body {
            font-family: 'Roboto', sans-serif;
            text-align: center;
```

```
        }
    </style>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>This is styled using Google Fonts.</p>
</body>
</html>
```

3. Save the file as **link.html**.

4. Open the **link.html** file in a Chrome browser and observe the output.

5. Experiment with changing the `font-family` or weights in the link tag (e.g., use a different Google Font) and reload the browser to see the changes.

Google APIs (www.googleapis.com) are application programming interfaces (APIs) developed by Google which allow communication with Google Services and their integration to other services. Examples of these include Search, Gmail, Translate or Google Maps. Third-party apps can use these APIs to take advantage of or extend the functionality of the existing services.

In the css example above, we used one of the font libraries in Googleapis.

5.7 CSS Backgrounds

The CSS background properties are used to add background effects for elements.

The background property is a shorthand property for:

- `background-color`
- `background-image`
- `background-repeat`
- `background-attachment`
- `background-position`
- `background-size`
- `background-origin`
- `background-clip`

CSS Syntax:

```
background: bg-color bg-image position/bg-size bg-repeat bg-origin
bg-clip bg-attachment initial|inherit;
```

Note: If one of the properties in the shorthand declaration is the bg-size property, you must use a / (slash) to separate it from the bg-position property, e.g. `background:url(smiley.gif) 10px 20px/50px 50px;` will result in a background image, positioned 10 pixels from the left, 20 pixels from the top, and the size of the image will be 50 pixels wide and 50 pixels high. If using multiple `background-image` sources but also want a `background-color`, the `background-color` parameter needs to be last in the list.

Instruction

1. Create an HTML file called **bgImage.html**

2. Download the background images : **paper.gif** and **img_flwr.gif** from https://bit.ly/37ltRvs

3. Copy the following code:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("paper.gif");
}
</style>
</head>
<body>

<h1>The background-image Property</h1>

<p>
   This demo shows some different techiques on how to set a
   background image of a HTML element.
</p>
</body>
</html>
```

4. Save the file as **bgImage.html**.

5. Open the **bgImage.html** file in a Chrome browser and observe the output.

6. Now, try with multiple image. Adjust the style using the following code:

```
<style>
body {
  background-image: url("paper.gif"), url("img_flwr.gif");
  bacground-position: right bottom, left top;
  background-repeat: no repeat, repeat;
  padding: 15px;
}
</style>
```

7. Copy this paragraph and insert into your HTML: Terengganu formerly spelled Trengganu or Tringganu, is a sultanate and constitutive state of federal Malaysia. The state is also known by its Arabic honorific, Darul Iman (**"Abode of Fait"**). The coastal city of Kuala Terengganu which stands at the mouth of the broad Terengganu River is both the state and royal capital as well as the largest city in Terengganu. There are many islands located close to the coast of Terengganu state, such as Perhentian Islands and Redang Island.

8. Save and open the **bgImage.html** file in a Chrome browser and observe the output.

Explore the background properties on the W3Schools website. Try out the examples provided and experiment with them in your own HTML and CSS files to see how they affect the background of an element.

5.8 CSS Table

CSS tables allow you to style HTML tables to enhance their appearance and usability. With CSS, you can control table elements such as borders, spacing, padding, background colors, text alignment, and hover effects. These properties make it easy to create visually appealing and responsive tables that improve readability and user experience.

Instruction

1. Create an HTML file called **table.html**.

2. Copy this code:

```
<!DOCTYPE html>
<html>
<head>
<style>
#customers {
  font-family: Arial, Helvetica, sans-serif;
  border-collapse: collapse;
  width: 100%;
}

#customers td, #customers th {
  border: 1px solid #ddd;
  padding: 8px;
}

#customers tr:nth-child(even) {
  background-color: #f2f2f2;
}

#customers tr:hover {
  background-color: #faf79f;
}

#customers th {
  padding-top: 12px;
  text-transform: uppercase;
  padding-bottom: 12px;
  text-align: center;
  background-color: #08d38c;
  color: white;
}
</style>
</head>
<body>
    <table id="customers">
        <tr>
          <th>Company</th>
          <th>Contact</th>
          <th>Country</th>
        </tr>
```

```
            <tr>
              <td>Alfreds Futterkiste</td>
              <td>Maria Anders</td>
              <td>Germany</td>
            </tr>
            <tr>
              <td>Ernst Handel</td>
              <td>Roland Mendel</td>
              <td>Austria</td>
            </tr>
            <tr>
              <td>Island Trading</td>
              <td>Helen Bennett</td>
              <td>UK</td>
            </tr>
            <tr>
              <td>Magazzini Riuniti</td>
              <td>Giovanni Rovelli</td>
              <td>Italy</td>
            </tr>
          </table>
    </body>
    </html>
```

3. Save and open the **table.html** file in a Chrome browser and observe the output.

5.9 CSS Navigation Bar

A CSS navigation bar is a horizontal or vertical menu designed to help users navigate a website. It is styled using CSS to control its layout, colors, fonts, and hover effects. Navigation bars often include links arranged in a list and can be customized to be responsive, interactive, and visually appealing, enhancing the website's usability.

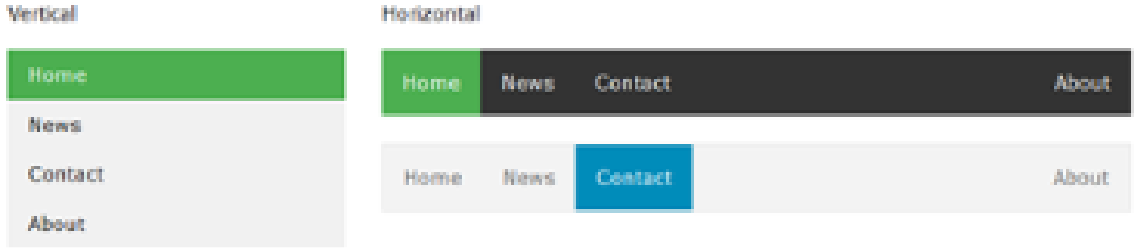

Figure 5.3: Examples of CSS navigation bar

Instruction

1. Create an HTML file called **navigation.html**.

2. Copy this code:

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    <ul>
      <li><a href="#home">Home</a></li>
      <li><a href="#news">News</a></li>
      <li><a href="#contact">Contact</a></li>
      <li><a href="#about">About</a></li>
    </ul>
</body>
</html>
```

3. Save and open the **navigation.html** file in a Chrome browser and observe the output.

4. Now let's remove the bullets and the margins and padding from the list. Add this inside `<head>`:

```
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
</style>
```

5. Save and open the **navigation.html** file in a Chrome browser and observe the output. Example explained:

   - `list-style-type: none;` - Removes the bullets. A navigation bar does not need list markers
   - Set `margin: 0;` and `padding: 0;` to remove browser default settings

6. Change the style to create a basic vertical navigation bar with a gray background color and change the background color of the links when the user moves the mouse over them:

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 200px;
  background-color: #f1f1f1;
}

li a {
  display: block;
  color: #000;
  padding: 8px 16px;
  text-decoration: none;
}

/* Change the link color on hover */
li a:hover {
  background-color: #555;
  color: white;
}
```

7. Save and open the **navigation.html** file in a Chrome browser and observe the output.

8. Now, change the style to create a basic horizontal navigation bar with a dark background color and change the background color of the links when the user moves the mouse over them:

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}

li {
```

```
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

/* Change the link color to #111 (black) on hover */
li a:hover {
  background-color: #111;
}
```

9. Save and open the **navigation.html** file in a Chrome browser and observe the output.

5.10 Lab Exercise 5

In this exercise, you will create a fully functional webpage by applying the concepts and principles of web design learned so far. This includes using semantic HTML for structuring content, linking an external CSS file for styling, and designing a visually appealing and user-friendly layout.

You will explore the importance of structured content and visual hierarchy while incorporating text from the Wikipedia page on **Universiti Malaysia Terengganu (UMT)**. The aim is to practice linking external CSS, implementing background, navigation bars, table and applying web design principles to create a cohesive and professional-looking webpage.

By the end of this exercise, you will have strengthened your skills in combining HTML and CSS, as well as gained experience in designing a webpage with real-world content and aesthetics.

5.10.1 Designing a Webpage with External CSS and Web Design Principles

Instruction

1. **Research and Content Collection**

- Visit the Universiti Malaysia Terengganu Wikipedia page.
- Select and copy two to three paragraphs of text from the page.
- Save the copied content in a plain text editor (e.g., Notepad or TextEdit) for later use.

2. **Create the HTML File**

   - Open your preferred text editor or code editor (e.g., Visual Studio Code, Sublime Text).
   - Create a new file and save it as **umt.html**.

3. **Define the Structure**

   - Use proper HTML5 semantic tags (`<header>` , `<nav>` , `<section>` , `<article>` , `<footer>`) to structure the webpage.
   - Include the following sections:
     - Header: Add a title for the webpage (e.g., Universiti Malaysia Terengganu).
     - Navigation Bar: Use a CSS-styled `<nav>` element to create navigation links (e.g., Home, About, Contact).
     - Inside the `<section>` or `<article>` where you placed the paragraphs, create a table to display some relevant data (e.g., UMT faculties, programs, or key achievements).
     - Main Content: Place the paragraphs you copied earlier inside a `<section>` or `<article>` .
     - Footer: Add basic footer content (e.g., © 2024 Universiti Malaysia Terengganu).

4. **Link an External CSS File**

   - Create a separate CSS file named **umt.css**.
   - Save the file in the same directory as **umt.html**.
   - Link the CSS file in the `<head>` section of your HTML file

5. **Style the Webpage**

   - In **umt.css**, apply the following styles:
     - Background for the body of the webpage.
     - Font styling: Use Google Fonts to enhance the text appearance.

- Navigation Bar: Style it with a distinct color and hover effects for the links.
- Table: Add styles for the table to enhance its appearance.
- Text Alignment: Ensure proper alignment and spacing for the paragraphs.
- Footer Styling: Use a subtle background color and centered text.

6. **Apply Web Design Principles**

- Ensure the webpage is:
  - Readable: Use contrasting colors for text and background.
  - Well-structured: Use spacing and margins to separate sections.

7. **Test Your Webpage**

- Open **umt.html** in a web browser to check the design and functionality.
- Make necessary adjustments to your CSS or HTML code based on the output.

8. **Submit Your Work**

- Save both **umt.html** and **umt.css** files.
- Submit the files via eLearn platform.

5.10.2 Rubric

Table 5.1: Rubric for Lab Exercise 5

Criteria	Excellent (5)	Good (4)	Average (2-3)	Needs Improvement (0-1)
Content Structure	All sections are used correctly and logically organized.	Most sections are used correctly with minor structural issues.	Some sections are missing or improperly used.	Sections are missing or poorly organized.
External CSS Implementation	CSS file is correctly linked and effectively styles the entire webpage.	CSS file is linked with minor styling issues or inconsistencies.	CSS is linked but lacks significant styling or effectiveness.	CSS file is not linked or contains major errors.
Styling and Design	Visually appealing, well-designed, and responsive.	Design is appealing but has minor responsiveness or styling flaws.	Design is basic and lacks proper styling or responsiveness.	Poor or no styling applied; design is unappealing.
Navigation Bar	Navigation bar is functional, styled, and includes hover effects.	Navigation bar is functional but lacks proper styling or hover effects.	Navigation bar is basic and not fully functional.	Navigation bar is missing or non-functional.
Table Inclusion	Table is well-structured, styled with borders, alternating row colors, padding, and hover effects.	Table is styled with borders and row colors but lacks hover effects or minor design flaws.	Basic table styling; limited use of row colors or borders.	Table included but poorly styled or lacks cohesion.
Text Formatting and Readability	Text is formatted consistently with proper fonts, spacing, and alignment.	Text is formatted but has minor alignment or spacing issues.	Text formatting is inconsistent or lacks clarity.	Text formatting is poor, making content hard to read.
Effort and Creativity	Webpage demonstrates originality and thoughtful effort in design.	Webpage shows effort but lacks significant creativity.	Webpage shows minimal effort or originality.	Webpage lacks effort and creativity.

Lab Module 6

JAVASCRIPT

6.1 Introduction to Javascript

JavaScript is a versatile programming language widely used to create dynamic and interactive content on websites. Alongside HTML and CSS, it forms the core technologies of web development, enabling developers to build engaging and responsive user experiences.

Key Features of JavaScript:

1. **Client-Side Execution:** JavaScript runs directly in the user's web browser without the need for server-side interaction, making it ideal for real-time updates and interactive content.

2. **Dynamic and Interactive Content:** JavaScript allows you to update a webpage's content, validate form data, create animations, and more, without requiring a page reload.

3. **Cross-Platform:** JavaScript works across all major browsers (e.g., Chrome, Firefox, Edge), ensuring compatibility for a wide range of users.

4. **Extensibility:** JavaScript can be enhanced with libraries (e.g., jQuery) and frameworks (e.g., React, Angular, Vue.js) to streamline development and introduce advanced functionality.

Basic Uses of JavaScript:

1. **Event Handling:** Responding to user actions such as clicks, mouse movements, or keystrokes.

2. **Manipulating HTML and CSS:** Dynamically changing the structure, content, or style of a webpage.

3. **Form Validation:** Ensuring input accuracy before submission.

4. **Creating Animations:** Enhancing visual appeal with smooth transitions and effects.

In HTML file, JavaScript code is inserted between `<script>` and `</script>` tags. Old JavaScript examples may use a type attribute: `<script type="text/javascript">`. The type attribute is not required. JavaScript is the default scripting language in HTML.

6.2 Objectives

By the end of this lab, students will be able to:

1. Develop familiarity with the JavaScript language.

2. Learn to use best-practice idioms and patterns.

3. Understand concepts commonly used in dynamic language programming.

6.3 Basic Javascript

Instruction

1. Write down this HTML page, and name it as **firstjs.html**:

```
<!DOCTYPE html>
<html>
<head>
    <title>My First JS</title>
</head>
<body>
    <script>
        document.write("My First JS!");
    </script>
</body>
</html>
```

2. Try to call the file **firstjs.html** in your browser.

3. After that, change the statement `document.write("My First JS!")` into: `document.write("<b> My First </b>" + "<i>Java Script</i>");`

4. Observe the result.

6.4 getElementById()

One of many JavaScript HTML methods is `getElementById()`. The example below finds an HTML element (with `id="demo"`), and changes the element content (`innerHTML`) to `"Hello JavaScript"`.

Instruction

1. Edit your **firstjs.html** by replacing with this code:

```
<!DOCTYPE html>
<html>
<body>
<h2>What Can JavaScript Do?</h2>
<p id="demo">JavaScript can change HTML content.</p>
<button type="button" onclick='document.getElementById("demo").
innerHTML = "Hello JavaScript!"'>Click Me!</button>
</body>
</html>
```

2. Observe the result.

Note : JavaScript accepts both double and single quotes.

```
document.getElementById("demo").innerHTML = "Hello JavaScript";
```

```
document.getElementById("demo").innerHTML = 'Hello 'JavaScript;
```

6.5 Date and Time

Instruction

1. Write down this HTML page, and name it as **datetimejs.html**:

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

```
<head>
    <title>Date and Time</title>
</head>
<body>
    <button type="button" onclick= "document.getElementById('demo').
    innerText = Date()";>Click me to display Date and Time</button>
    <p id="demo"></p>
</body>
</html>
```

2. After you try to display date/time and get the result, now add these scripts to change the style of variable (demo):

```
   document.getElementById("demo").style.color = "red";
document.getElementById("demo").style.backgroundColor = "yellow";
```

3. Observe the result.

6.6 Change HTML Attribute Values

In this activity, we will learn how JavaScript changes the value of the src (source) attribute of an `<img>` tag.

Instruction

- Open your code editor and create a new HTML file. Save it as changelight.html.
- Download the images of an ON and OFF light bulb from the following link: https://bit.ly/37ltRvs
- Save these images in the same folder as your HTML file.
  - ON bulb: Name the file **pic_bulbon.gif**.
  - OFF bulb: Name the file **pic_bulboff.gif**.
- Write the following code:

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

<h2>What Can JavaScript \index{JavaScript} Do?</h2>
```

```
<p>JavaScript \index{JavaScript} can change HTML attribute values.
</p>
<p>In this case, JavaScript \index{JavaScript} changes the value of
the `src`(source) attribute of an image.</p>

<!-- Button to turn the light ON -->
<button onclick='document.getElementById("myImage").src=
"pic_bulbon.gif"'>Turn on the light</button>

<!-- Image tag -->
<img id="myImage" src=
"pic_bulboff.gif" style="width:100px">

<!-- Button to turn the light OFF -->
<button onclick='document.getElementById("myImage").src=
"pic_bulboff.gif"'>Turn off the light</button>

</body>
</html>
```

- Save and open the **changelight.html** file in a Chrome browser and observe the output.

- Click the `"Turn on the light"` and `"Turn off the light"` buttons to see how the image changes dynamically.

6.7 Link to file Javascript (*.js)

We can create an external JavaScript file and embed it in many HTML pages. It provides code reusability because a single JavaScript file can be used in several HTML pages.

An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.

6.7.1 JavaScript in `<head>` or `<body>`

You can place any number of scripts in an HTML document. Scripts can be placed in the `<body>`, or in the `<head>` section of an HTML page, or in both. JavaScript in `<head>`.

In this example, a JavaScript function is placed in the `<head>` section of an HTML page.

6.7.2 External JavaScript Advantages

Placing scripts in external files has some advantages:

- It separates HTML and code
- It makes HTML and JavaScript easier to read and maintain
- Cached JavaScript files can speed up page loads

To add several script files to one page, use several script tags:

```
<script src="myscript1.js"></script>
<script src="myscript2.js"></script>
```

6.7.3 External References

External scripts can be referenced with a full URL or with a path relative to the current web page.

This example uses a full URL to link to a script:

```
<script src="https://www.w3schools.com/js/myscript1.js"></script>
```

This example uses a script located in a specified folder on the current website:

```
<script src="/js/myscript1.js"></script>
```

Instruction

- Create a new file and write the following code:

```
function msg() {
    alert("Hello UMT student");
}
```

- Save the file as **message.js**.
- Open the code editor again and create another new file.
- Write the following code:

```
<!DOCTYPE html>
<html>
<head>
    <title>External JavaScript Example</title>
    <!-- Link the external JavaScript file -->
    <script src="message.js"></script>
</head>
<body>
    <h2>Click the Button to See the Message</h2>
    <!-- Button to trigger the JavaScript function -->
    <button onclick="msg()">Click Me</button>
</body>
</html>
```

- Save the file as **extjavascript.html**.
- Ensure both files (**message.js** and **extjavascript.html**) are in the same folder.
- Open the **extjavascript.html** file in a web browser.
- Click the button labeled **"Click Me"**. an alert dialog box should appear with the message: **Hello UMT student.**

6.8 Working with events

Instruction

1. Create an HTML file with the following code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Disable Right-Click</title>
    <script>
        // JavaScript to disable right-click
        document.addEventListener("contextmenu", function(event) {
            alert("Right-click is disabled on this page.");
            event.preventDefault();
        });
    </script>
</head>
```

```
<body>
    <h2>Right-Click Disabled</h2>
    <p>Try right-clicking anywhere on this page.</p>
</body>
</html>
```

2. Save the file as **disable-right-click.html**.

3. Open the file in a web browser.

6.8.1 How It Works

1. `contextmenu` Event Listener: The `contextmenu` event is triggered when a user attempts to open the right-click menu.

2. `preventDefault()` Method: This method prevents the browser's default right-click menu from appearing.

3. Alert Message: The script also displays a message (**"Right-click is disabled on this page."**) to inform the user.

6.9 Multiplication Table

Instruction

1. Open your code editor and create a new HTML file.

2. Add the following code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Multiplication Table</title>
    <!-- Add some styling -->
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        h2 {
            color: #333;
        }
```

```
            table {
                border-collapse: collapse;
                width: 100%;
            }
            th, td {
                border: 1px solid #ddd;
                text-align: center;
                padding: 8px;
            }
            th {
                background-color: #f2f2f2;
            }
            input, button {
                padding: 10px;
                margin: 5px 0;
            }
        </style>
    </head>
    <body>
        <h2>Multiplication Table Generator</h2>
        <label for="rows">Enter number of rows:</label>
        <input type="number" id="rows" min="1" placeholder="Rows">
        <br>
        <label for="columns">Enter number of columns:</label>
        <input type="number" id="columns" min="1" placeholder="Columns">
        <br>
        <button onclick="generateTable()">Generate Table</button>
        <br><br>
        <div id="tableContainer"></div>

        <script src="multiplication-table.js"></script>
    </body>
    </html>
```

3. Save the file as **multiplication-table.html**.

4. Create a new JavaScript file.

5. Add the following code:

```
    function generateTable() {
        // Get the number of rows and columns from user input
```

```
        const rows = parseInt(document.getElementById("rows").value);
        const columns = parseInt(document.getElementById("columns").
        value);

        // Validate the input
        if (isNaN(rows) || rows < 1 || isNaN(columns) || columns < 1) {
            alert("Please enter valid numbers for rows and columns.");
            return;
        }

        // Create the table
        let tableHTML = "<table>";
        for (let i = 1; i <= rows; i++) {
            tableHTML += "<tr>";
            for (let j = 1; j <= columns; j++) {
                tableHTML += `<td>${i * j}</td>`;
            }
            tableHTML += "</tr>";
        }
        tableHTML += "</table>";

        // Display the table
        document.getElementById("tableContainer").innerHTML = tableHTML;
    }
```

6. Save the file as **multiplication-table.js**.

7. Make sure the files **multiplication-table.html** and **multiplication-table.js** are in the same folder.

8. Open **multiplication-table.html** in a web browser.

9. Enter the desired number of rows and columns in the input fields.

10. Click the Generate Table button.

11. The multiplication table will appear below the button.

6.10 Client-side validation

Validation can be defined by many different methods and deployed in many different ways.

- Server-side validation is performed by a web server, after input has been sent to the server.
- Client-side validation is performed by a web-browser, before the input is sent to a server.

6.10.1 Form Validation

Instruction

1. Open a code editor and create a new file.
2. Write the following code to create a basic form:

```
<!DOCTYPE html>
<html>
<head>
    <title>Form Validation</title>
    <!-- Add basic styling -->
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .form-group {
            margin-bottom: 10px;
        }
        label {
            display: block;
            margin-bottom: 5px;
        }
        input {
            padding: 8px;
            width: 300px;
        }
        button {
            padding: 10px 20px;
            background-color: #4CAF50;
            color: white;
            border: none;
            cursor: pointer;
        }
```

```
            button:hover {
                background-color: #45a049;
            }
        </style>
    </head>
    <body>
        <h2>Form Validation Example</h2>
        <form name="myForm" onsubmit="return validateForm()">
            <div class="form-group">
                <label for="fname">First Name:</label>
                <input type="text" id="fname" name="fname">
            </div>
            <button type="submit">Submit</button>
        </form>

        <!-- Include JavaScript \index{JavaScript} -->
        <script src="form-validation.js"></script>
    </body>
    </html>
```

3. Save the file as **form-validation.html**.

4. Create another new file in the same folder.

5. Write the following JavaScript code to validate the form:

```
    function validateForm() {
        let fname = document.forms["myForm"]["fname"].value;
        if (fname === "") {
            alert("First Name must be filled out.");
            return false;
        }
    }
```

6. Save the file as **form-validation.js**.

7. Ensure both files (**form-validation.html** and **form-validation.js**) are in the same folder.

8. Try submitting the form:

    - **Case 1:** Leave the **"First Name"** field empty and click Submit. An alert should appear saying, **"First Name must be filled out."**.

- **Case 2:** Fill in the "**First Name**" field and click Submit. The form should submit successfully.

9. Observe the result.

6.10.2 Numeric Input Validation

Instruction

1. Open a code editor and create a new file.

2. Write the following code to create a form with a numeric input field:

```
<!DOCTYPE html>
<html>
<head>
    <title>Numeric Input Validation</title>
    <!-- Add basic styling -->
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .form-group {
            margin-bottom: 10px;
        }
        label {
            display: block;
            margin-bottom: 5px;
        }
        input {
            padding: 8px;
            width: 300px;
        }
        button {
            padding: 10px 20px;
            background-color: #007BFF;
            color: white;
            border: none;
            cursor: pointer;
        }
        button:hover {
```

```
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <h2>Numeric Input Validation Example</h2>
    <form name="numericForm" onsubmit="return
    validateNumericInput()">
        <div class="form-group">
            <label for="age">Enter Your Age:</label>
            <input type="text" id="age" name="age" placeholder=
            "Enter a number">
        </div>
        <button type="submit">Submit</button>
    </form>

    <!-- Include JavaScript \index{JavaScript} -->
    <script src="numeric-validation.js"></script>
</body>
</html>
```

3. Save the file as **numeric-validation.html**.

4. Create another new file in the same folder.

5. Write the following JavaScript code to validate numeric input:

```
function validateNumericInput() {
    let age = document.forms["numericForm"]["age"].value;

    // Check if the input is empty
    if (age === "") {
        alert("Age field cannot be empty.");
        return false;
    }

    // Check if the input is a numeric value
    if (isNaN(age)) {
        alert("Please enter a valid numeric value.");
        return false;
    }
```

```
        // Check if the input is within a specific range (optional)
        if (age < 1 || age > 120) {
            alert("Please enter a number between 1 and 120.");
            return false;
        }
        // If all validations pass
        alert("Form submitted successfully!");
        return true;
    }
```

6. Save the file as **numeric-validation.js**.

7. Ensure both files (**numeric-validation.html** and **numeric-validation.js**) are in the same folder.

8. Open the **numeric-validation.html** file in a web browser.

9. Try submitting the form:

   - **Case 1:** Leave the input field empty and click Submit. An alert should appear saying, **"Age field cannot be empty."**.
   - **Case 2:** Enter non-numeric characters and click Submit. An alert should appear saying, **"Please enter a valid numeric value"**.
   - **Case 3:** Enter a number outside the range (e.g., -5 or 200) and click Submit. An alert should appear saying, **"Please enter a number between 1 and 120."**.
   - **Case 4:** Enter a valid number (e.g., 25) and click Submit. A success message will appear.

6.11 Math Function

The `Math.ceil()` function is used to get the smallest integer greater than or equal to a given number.

The `Math.random()` function is used to get a floating-point, pseudo-random number in the range [0, 1) that is, from 0 (inclusive) up to but not including 1 (exclusive), which you can then scale to your desired range.

6.11.1 Guess a Number

Instruction

1. Open a code editor and create a new HTML file.

2. Add the following code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Guess the Number Game</title>
    <!-- Add basic styling -->
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        h2 {
            color: #333;
        }
        button {
            padding: 10px 20px;
            background-color: #007BFF;
            color: white;
            border: none;
            cursor: pointer;
        }
        button:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <h2>Guess the Number Game</h2>
    <p>Try to guess the random number between 1 and 10!</p>
    <button onclick="playGame()">Start Game</button>

    <script src="guess-number.js"></script>
</body>
</html>
```

3. Save this file as **guess-number.html**.

4. Create a new JavaScript file.

5. Add the following code:

```
function playGame() {
    // Generate a random integer between 1 and 10
    const randomNumber = Math.floor(Math.random() * 10) + 1;

    // Prompt the user to guess the number
    const userGuess = parseInt(prompt("Guess a number between 1
    and 10:"), 10);

    // Validate the user's input
    if (isNaN(userGuess) || userGuess < 1 || userGuess > 10) {
        alert("Please enter a valid number between 1 and 10.");
        return;
    }

    // Check if the guess is correct
    if (userGuess === randomNumber) {
        alert("Good Work! You guessed it right.");
    } else {
        alert(`Not matched. The correct number was
        ${randomNumber}.`);
    }
}
```

6. Save this file as **guess-number.js**.

7. Ensure both files (**guess-number.html** and **guess-number.js**) are in the same folder.

8. Open the **guess-number.html** file in a web browser.

9. Click the Start Game button.

10. Enter your guess in the prompt dialog box.

11. The program will display:

    - **"Good Work!"** if your guess matches the random number.
    - **"Not matched."** if your guess is incorrect, along with the correct number.

6.11.2 How it Works

1. The `Math.random()` function generates a random decimal between 0 and 1.
2. `Math.floor()` rounds the result of `Math.random()` * 10 down to the nearest integer to ensure a range from 1 to 10.
3. The `prompt()` function asks the user for their guess, and `parseInt()` converts the input into an integer.
4. A simple if condition checks whether the guess matches the random number.

6.12 Lab Exercise 6

In this exercise, you will create a dynamic quiz application using HTML, CSS, and JavaScript. This task will help you integrate fundamental web development concepts to build an engaging user experience.

You will implement multiple-choice questions, a timer to challenge users, and real-time feedback for correct and incorrect answers. Additionally, questions will be randomized to enhance interactivity. By completing this lab, you will strengthen your skills in structuring content, styling with CSS, and adding interactivity with JavaScript.

6.12.1 Quiz Application

Instruction

1. Create the following files in your project directory:
   - index.html (for the HTML structure)
   - style.css (for the CSS styling)
   - script.js (for the JavaScript functionality)
2. In your **index.html**, structure the page with the following elements:
   - A header that introduces the quiz.
   - A section to display the questions, answer options, and a timer.
   - A section to show feedback (correct/incorrect) after each answer.
   - A button to submit answers and navigate between questions.
   - A footer to display the user's score after finishing the quiz.
3. In your **style.css** file, style the quiz application to make it visually appealing.

4. In your **script.js**, implement the following javascript functionalities:

   - **Questions Array:** Stores the questions, possible options, and the correct answer.
   - **shuffleQuestions():** Randomly shuffles the questions to ensure each quiz is different.
   - **startTimer():** Sets a countdown timer for answering each question.
   - **displayQuestion():** Displays the current question and answer options.
   - **checkAnswer():** Checks whether the selected answer is correct and updates the score.
   - **nextQuestion():** Moves to the next question, resetting the timer and updating the score.
   - **startQuiz():** Starts the quiz by shuffling the questions, displaying the first question, and starting the timer.

5. Save **index.html**, **style.css** and **script.js** and zip the file

6. Submit the files via eLearn platform.

6.12.2 Rubric

Table 6.1: Rubric for Lab Exercise 6

Criteria	Excellent (5)	Good (4)	Average (2-3)	Needs Improvement (0-1)
HTML Structure	The HTML is well-structured, with clear semantic tags (`<header>`, `<footer>`, `<section>`, etc.). All elements are in place, and the layout is clear.	The HTML structure is mostly correct, with appropriate tags. Some minor issues with structure or missing elements.	The HTML structure is basic but functional. Some elements might be missing or incorrectly used.	The HTML structure is unclear, missing key elements, and difficult to navigate.
CSS Styling	Excellent use of CSS, creating a visually appealing layout with proper use of spacing, alignment, and colors.	Good styling with minor inconsistencies in layout, alignment, or colors.	Basic styling with noticeable flaws in alignment, color scheme, or layout.	Minimal or poor styling with a lack of attention to visual presentation.
JavaScript Functionality	The quiz is fully functional with a timer, randomized questions, and dynamic feedback. All buttons work as expected.	Most features work correctly, including a timer and randomized questions, but may have minor bugs.	Some features work, but there are major issues (e.g., no timer, randomization not functioning).	JavaScript functionality is incomplete or not working as expected.

Lab Module 7

HTML, CSS AND JAVASCRIPT

In this lab, we will explore how to enhance the interactivity and functionality of web pages through various dynamic elements. You will learn to create and implement:

- **Transparent Image Text:** Overlay text on images, allowing for better design flexibility and aesthetic appeal. Forms on Images: Add interactive forms directly on images to create engaging user experiences.
- **Side Navigation Buttons:** Design a functional and visually appealing side navigation panel to organize and access different sections of a website.
- **Automatic Slideshow:** Develop a seamless, automatic image slideshow to display content in a visually appealing manner.
- **Profile Card:** Create a stylish profile card layout to showcase user information effectively.
- **Progress Bar:** Integrate a progress bar to track completion status in tasks or processes.
- **Collapsible Elements:** Build collapsible sections to manage content dynamically, enhancing the user interface by keeping the page uncluttered.

By the end of this lab, you will have gained hands-on experience in implementing these essential interactive web components, essential for modern dynamic websites.

7.1 Create a Transparent Image Text

Imagine a landscape image with a semi-transparent overlay in the center, displaying text like "**Explore Nature**". This activity helps reinforce concepts of HTML structure,

positioning with CSS, and styling elements effectively.

Instruction

1. Create an HTML file named **transparent-image-text.html**.

2. Write the following code to include an image and a text overlay:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
    <title>Transparent Image Text</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="image-container">
        <img src="your-image.jpg" alt="Beautiful Scenery">
        <div class="overlay-text">Your Transparent Text</div>
    </div>
</body>
</html>
```

3. Create a CSS file named **styles.css**.

4. Style the image and overlay text to make the text transparent and beautifully integrated with the image.

```
body {
    margin: 0;
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    display: block;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.image-container {
    position: relative;
```

```
        width: 80%;
        max-width: 600px;
    }

    .image-container img {
        width: 100%;
        border-radius: 10px;
    }

    .image-container .overlay-text {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        background-color: rgba(0, 0, 0, 0.5);
        /* Semi-transparent background */
        color: white;
        padding: 10px 20px;
        border-radius: 5px;
        font-size: 24px;
        text-align: center;
    }
```

5. Save the HTML and CSS files in the same folder.

6. Open **transparent-image-text.html** file in your browser and observe the result.

7. **Steps to Customize:**

   - Replace **your-image.jpg** with the actual image file name in the img tag.
   - Update the text **"Your Transparent Text"** in the overlay-text div to your desired text.
   - Adjust the styles in the CSS file to match your desired design (e.g., font size, background color, padding).

7.2 Create a Form on Image

In this activity, you will design a login form overlaying an image. The form will be responsive and positioned over a background image to make it visually appealing.

Instruction

1. Create an HTML file named **form-on-image.html**.

2. Structure HTML to include a full-width background image and an overflow of the login form using the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Form on Image</title>
    <link rel="stylesheet" href="form-styles.css">
</head>
<body>
    <div class="image-container">
        <img src="background-image.jpg" alt="Background Image">
        <div class="form-container">
            <h2>Login</h2>
            <form action="#" method="post">
                <label for="email">Email</label>
                <input type="email" id="email" name="email"
                placeholder="Enter Email" required>
                <label for="password">Password</label>
                <input type="password" id="password" name="password"
                placeholder="Enter Password" required>
                <button type="submit">Login</button>
            </form>
        </div>
    </div>
</body>
</html>
```

3. Create a CSS file named **form-styles.css**.

4. Style the background image, form, and input fields to make them responsive and visually appealing with the following code:

```
body {
    margin: 0;
```

```
    font-family: Arial, sans-serif;
}

.image-container {
    position: relative;
    width: 100%;
    height: 100vh;
    overflow: hidden;
}

.image-container img {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

.form-container {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: rgba(255, 255, 255, 0.9);
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    width: 300px;
    text-align: center;
}

.form-container h2 {
    margin-bottom: 20px;
}

.form-container label {
    display: block;
    font-weight: bold;
    margin-bottom: 5px;
}

.form-container input {
    width: 100%;
```

```
        padding: 10px;
        margin-bottom: 15px;
        border: 1px solid #ccc;
        border-radius: 5px;
    }

    .form-container button {
        width: 100%;
        padding: 10px;
        background-color: #28a745;
        color: white;
        border: none;
        border-radius: 5px;
        font-size: 16px;
        cursor: pointer;
    }

    .form-container button:hover {
        background-color: #218838;
    }

    @media (max-width: 768px) {
        .form-container {
            width: 90%;
        }
    }
```

5. Save the HTML and CSS files in the same folder.

6. Open **form-on-image.html** file in your browser and observe the result.

7. **Steps to Customize:**

   - Replace **background-image.jpg** in the img tag with the name of your image file.
   - Modify the labels, placeholder text, or button label to suit your form's purpose.
   - Customize colors, fonts, and styles in the CSS file to better align with your desired design.

7.3 Create a Side Navigation Buttons

In this activity, you will design a webpage with side navigation buttons that allow users to navigate to different sections of the page. The navigation buttons will be positioned on the left side of the page and styled to enhance user experience.

Instruction

1. Create an HTML file named **side-navigation.html**.
2. Structure the HTML to include a set of side navigation buttons and multiple sections on the page using the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Side Navigation</title>
    <link rel="stylesheet" href="side-navigation.css">
</head>
<body>
    <div class="side-nav">
        <a href="#section1">Section 1</a>
        <a href="#section2">Section 2</a>
        <a href="#section3">Section 3</a>
        <a href="#section4">Section 4</a>
    </div>

    <div class="content">
        <section id="section1">
            <h2>Section 1</h2>
            <p>This is the content for Section 1.</p>
        </section>
        <section id="section2">
            <h2>Section 2</h2>
            <p>This is the content for Section 2.</p>
        </section>
        <section id="section3">
            <h2>Section 3</h2>
            <p>This is the content for Section 3.</p>
```

```
        </section>
        <section id="section4">
            <h2>Section 4</h2>
            <p>This is the content for Section 4.</p>
        </section>
    </div>
</body>
</html>
```

3. Create a CSS file named **side-navigation.css**.

4. Style the side navigation buttons and page sections to make the navigation visually appealing and functional using the following code:

```
body {
    margin: 0;
    font-family: Arial, sans-serif;
}

.side-nav {
    position: fixed;
    top: 50%;
    left: 0;
    transform: translateY(-50%);
    width: 200px;
    background-color: #333;
    border-right: 2px solid #ddd;
    padding: 10px;
    box-shadow: 2px 0 5px rgba(0, 0, 0, 0.3);
}

.side-nav a {
    display: block;
    color: white;
    text-decoration: none;
    padding: 10px;
    margin-bottom: 10px;
    background-color: #444;
    text-align: center;
    border-radius: 5px;
    transition: background-color 0.3s ease;
```

```
}

.side-nav a:hover {
    background-color: #555;
}

.content {
    margin-left: 220px;
    padding: 20px;
}

section {
    padding: 50px;
    margin: 20px 0;
    background-color: #f4f4f4;
    border-radius: 10px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
}

section h2 {
    margin-top: 0;
}

/* Smooth scrolling for navigation links */
html {
    scroll-behavior: smooth;
}
```

5. Save the HTML and CSS files in the same folder.

6. Open **side-navigation.html** file in your browser and observe the result.

7. Click on the side navigation buttons to see smooth scrolling to the corresponding sections.

8. **Steps to Customize:**

    - Add or remove section elements in the HTML file as needed.
    - Update the href attributes in the `<a>` tags to match the IDs of your sections.
    - Adjust the colors, font sizes, and styles in the CSS file to fit your design preferences.

7.4 Create Automatic Slideshow

In this activity, you will create a simple and attractive automatic slideshow that displays images, transitioning between them every few seconds.

Instruction

1. Create an HTML file named **slideshow.html**.
2. Structure the HTML to include a container for the slideshow and images using the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Automatic Slideshow</title>
    <link rel="stylesheet" href="slideshow.css">
</head>
<body>
    <div class="slideshow-container">
        <div class="slide fade">
            <img src="image1.jpg" alt="Image 1">
        </div>
        <div class="slide fade">
            <img src="image2.jpg" alt="Image 2">
        </div>
        <div class="slide fade">
            <img src="image3.jpg" alt="Image 3">
        </div>
    </div>

    <script src="slideshow.js"></script>
</body>
</html>
```

3. Create a CSS file named **slideshow.css**.
4. Style the slideshow container and images for smooth transitions using the following code:

```
body {
    margin: 0;
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    display: block;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.slideshow-container {
    position: relative;
    max-width: 80%;
    height: 400px;
    overflow: hidden;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);
}

.slide {
    display: none;
    position: absolute;
    width: 100%;
    height: 100%;
}

.slide img {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

.fade {
    animation: fadeEffect 1s ease-in-out;
}

@keyframes fadeEffect {
    from {
        opacity: 0.4;
    }
```

```
    to {
        opacity: 1;
    }
}
```

5. Create a JavaScript file named **slideshow.js**.

6. Write the following script to automatically cycle through the images:

```
let slideIndex = 0;

function showSlides() {
    let slides = document.getElementsByClassName("slide");

    // Hide all slides
    for (let i = 0; i < slides.length; i++) {
        slides[i].style.display = "none";
    }

    // Increment the slide index
    slideIndex++;

    // Reset to the first slide if the index exceeds the number
    of slides
    if (slideIndex > slides.length) {
        slideIndex = 1;
    }

    // Display the current slide
    slides[slideIndex - 1].style.display = "block";

    // Call this function again after 3 seconds
    setTimeout(showSlides, 3000);
}

// Initialize the slideshow
showSlides();
```

7. Save the HTML, CSS, and JavaScript files in the same folder.

8. Add your own images to the folder.

9. Open the **slideshow.html** file in your browser.

10. Observe the automatic transitions between images.

11. **Steps to Customize:**

    - Replace **image1.jpg**, **image2.jpg**, and **image3.jpg** with your own image file paths in the HTML file.
    - Modify the height and width properties in the CSS to adjust the slideshow dimensions.
    - Change the setTimeout duration in the JavaScript file to adjust the time interval between slides.

7.5 Create Profile Card

In this activity, you will design a user profile card similar to the image provided. The card will include a user image, name, designation, educational background, social media links, and a "**Contact**" button.

Instruction

1. Create an HTML file named **profile-card.html**.

2. Structure the card with appropriate elements for the user image, name, designation, social links, and button using the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>User Profile Card</title>
    <link rel="stylesheet" href="profile-card.css">
</head>
<body>
    <div class="profile-card">
        <div class="card-header">
            <h2>User Profile Card</h2>
        </div>
        <img src="rowan-atkinson.jpg" alt="Rowan Atkinson"
```

```
        class="profile-image">
        <h3 class="profile-name">Rowan Atkinson</h3>
        <p class="profile-title">CEO & British Sitcom Creator</p>
        <p class="profile-education">Oxford University</p>
        <div class="social-links">
            <a href="#" class="social-icon"><i class="fa fa-globe">
            </i></a>
            <a href="#" class="social-icon"><i class="fa fa-twitter">
            </i></a>
            <a href="#" class="social-icon"><i class="fa
            fa-linkedin"></i></a>
            <a href="#" class="social-icon"><i class="fa
            fa-facebook">                </i></a>
        </div>
        <button class="contact-btn">Contact</button>
    </div>
</body>
</html>
```

3. Create a CSS file named **profile-card.css**.

4. Style the card for an attractive and professional look using the following code:

```
body {
    margin: 0;
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    display: block;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.profile-card {
    width: 300px;
    background: #ffffff;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    text-align: center;
    padding: 20px;
    overflow: hidden;
```

```
}

.card-header h2 {
    font-size: 18px;
    color: #555;
    margin: 0 0 20px;
}

.profile-image {
    width: 120px;
    height: 120px;
    border-radius: 50%;
    margin: 0 auto;
    object-fit: cover;
    border: 3px solid #ddd;
}

.profile-name {
    font-size: 20px;
    font-weight: bold;
    color: #333;
    margin: 15px 0 5px;
}

.profile-title {
    font-size: 14px;
    color: #777;
    margin-bottom: 5px;
}

.profile-education {
    font-size: 14px;
    color: #999;
    margin-bottom: 15px;
}

.social-links {
    margin: 15px 0;
}

.social-links a {
```

```
        text-decoration: none;
        font-size: 20px;
        color: #555;
        margin: 0 10px;
        transition: color 0.3s ease;
    }

    .social-links a:hover {
        color: #007BFF;
    }

    .contact-btn {
        background: #000;
        color: #fff;
        border: none;
        padding: 10px 20px;
        border-radius: 5px;
        font-size: 14px;
        cursor: pointer;
        transition: background 0.3s ease;
    }

    .contact-btn:hover {
        background: #333;
    }
```

5. Save the HTML and CSS files in the same folder.

6. Include the image file in the same folder as **rowan-atkinson.jpg** (or modify the path in the img tag).

7. Open the **profile-card.html** file in your browser.

8. **Steps to Customize:**

   - Replace **rowan-atkinson.jpg** with the desired user image file in the img tag.
   - Modify text such as Rowan Atkinson, CEO & British Sitcom Creator, and Oxford University in the HTML file to suit your content.
   - Update the social media links in the <a> tags with the respective URLs.

7.6 Create Progress Bar

In this activity, you will create a progress bar that visually shows the progress of a task, such as uploading files or completion of quizzes. The progress bar will update dynamically as a button is clicked.

Instruction

1. Create an HTML file named **progress-bar.html**.
2. Add elements for the progress bar container, progress indicator, and a button to simulate progress using the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Progress Bar</title>
    <link rel="stylesheet" href="progress-bar.css">
</head>
<body>
    <div class="container">
        <h2>Progress Bar Example</h2>
        <div class="progress-container">
            <div class="progress-bar" id="progress-bar"></div>
        </div>
        <button id="increase-progress">Increase Progress</button>
    </div>

    <script src="progress-bar.js"></script>
</body>
</html>
```

3. Create a CSS file named **progress-bar.css**.
4. Style the progress bar and ensure it looks appealing using the following code:

```
body {
    font-family: Arial, sans-serif;
    background-color: #f9f9f9;
```

```
    display: block;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

.container {
    text-align: center;
    width: 50%;
    background: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

h2 {
    margin-bottom: 20px;
    color: #333;
}

.progress-container {
    width: 100%;
    background: #e0e0e0;
    border-radius: 20px;
    overflow: hidden;
    margin-bottom: 20px;
    height: 30px;
}

.progress-bar {
    height: 100%;
    width: 0%;
    background: #4caf50;
    border-radius: 20px;
    transition: width 0.4s ease-in-out;
}

button {
    padding: 10px 20px;
    font-size: 16px;
```

```
        color: #fff;
        background: #4caf50;
        border: none;
        border-radius: 5px;
        cursor: pointer;
        transition: background 0.3s ease;
    }

    button:hover {
        background: #45a049;
    }
```

5. Create a JavaScript file named **progress-bar.js**.

6. Write logic to increase the progress as the button is clicked using the following code:

```
    document.addEventListener("DOMContentLoaded", function () {
        const progressBar = document.getElementById("progress-bar");
        const increaseButton = document.getElementById
        ("increase-progress");

        let progress = 0;

        increaseButton.addEventListener("click", function () {
            if (progress < 100) {
                progress += 10; // Increase progress by 10%
                progressBar.style.width = progress + "%";
            } else {
                alert("Progress is complete!");
            }
        });
    });
```

7. Save the HTML, CSS, and JavaScript files in the same folder.

8. Open the **progress-bar.html** file in your browser.

9. Click the **"Increase Progress"** button to see the progress bar update dynamically.

0. **Further Enhancements:**

   - Add a **"Reset"** button to reset the progress back to 0%.

- Use additional animations or styles to make the progress bar more interactive.
- Simulate real-world progress, such as file uploads or downloading tasks.

7.7 Create A Collapsible

In this activity, you will create a collapsible section that can be toggled between showing and hiding content when a button is clicked. This feature is commonly used in FAQs or sections with expandable content.

Instruction

1. Create an HTML file named **collapsible.html**.
2. Add a button and a section of content that will collapse or expand when the button is clicked using the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Collapsible Section</title>
    <link rel="stylesheet" href="collapsible.css">
</head>
<body>
    <div class="container">
        <h2>Collapsible Example</h2>
        <button class="collapsible">Click to Expand/Collapse</button>
        <div class="content">
            <p>This is the collapsible content. Click the button
            above to toggle between showing and hiding this section.
            You can add more content here.</p>
        </div>
    </div>

    <script src="collapsible.js"></script>
</body>
</html>
```

3. Create a CSS file named **collapsible.css**.

4. Style the button and collapsible content for an appealing design using the following code:

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    display: block;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

.container {
    width: 50%;
    text-align: center;
    background: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

h2 {
    margin-bottom: 20px;
}

.collapsible {
    background-color: #007bff;
    color: white;
    cursor: pointer;
    padding: 10px 20px;
    border: none;
    text-align: center;
    font-size: 16px;
    border-radius: 5px;
    transition: background-color 0.3s ease;
}

.collapsible:hover {
    background-color: #0056b3;
}
```

```
.content {
    margin-top: 20px;
    padding: 15px;
    background: #f1f1f1;
    border-radius: 5px;
    display: none;
    overflow: hidden;
}
```

5. Create a JavaScript file named **collapsible.js**.

6. Write the logic to toggle the visibility of the collapsible content using the following code:

```
document.addEventListener("DOMContentLoaded", function () {
    const collapsibleButton = document.querySelector
    (".collapsible");
    const content = document.querySelector(".content");

    collapsibleButton.addEventListener("click", function () {
        if (content.style.display === "block") {
            content.style.display = "none";
        } else {
            content.style.display = "block";
        }
    });
});
```

7. Save the HTML, CSS, and Javascript files in the same folder.

8. Open the **collapsible.html** file in your browser.

9. Click the **"Click to Expand/Collapse"** button to toggle the visibility of the collapsible content.

10. **Further Enhancements:**

    - Add smooth animations for expanding and collapsing the content using CSS transitions.
    - Allow multiple collapsible sections with independent toggling.

- Style the collapsible content with additional details, like icons or background images.

7.8 Lab Exercise 7

The objective of this lab is to integrate multiple concepts of HTML, CSS, and JavaScript to create visually appealing and interactive web elements. Students will work on building a homepage that includes all components of Lab 7 to enhance their understanding of front-end development. Use this opportunity to explore your creativity in layout, colors, fonts, and interactivity while following these minimum requirements.

7.8.1 Advanced Web Design and Interaction

Instruction

1. **Transparent Image Text**
   - Use a responsive background image and overlay it with clear and readable text using CSS.
   - The text should be descriptive and act as the main title or tagline of the homepage.
2. **Form on Image**
   - Include a login or sign-up form over a background image.
   - Ensure the form is responsive and visually appealing.
3. **Side Navigation**
   - Add a vertical navigation bar linking to different sections of your homepage.
   - Ensure smooth scrolling or visibility when a section is clicked.
4. **Automatic Slideshow**
   - Include an image carousel or slideshow showcasing relevant images for your homepage theme.
   - Use JavaScript to automate the slideshow transitions.
5. **Profile Card**
   - Create a profile card for yourself or a fictional character.
   - Include an image, title, description, and social media icons.

6. **Progress Bar**
    - Add a progress bar to show loading or progress in an area of your homepage (e.g., page load, skill levels, or project completion).
7. **Collapsible Section**
    - Include a collapsible section that hides and shows content dynamically (e.g., FAQs, additional information, or testimonials).
    - Use JavaScript for interactivity.

Instruction

1. While you must include all the components mentioned above, you are free to design the homepage in your unique style.
2. Choose a theme (e.g., personal portfolio, travel blog, company homepage, or online store).
3. Use external CSS to style your webpage. Experiment with colors, fonts, margins, and animations to make your page visually appealing.
4. Use JavaScript to make the page interactive (e.g., for the collapsible section, slideshow, and progress bar).
5. Zip your **Lab7_Project** folder and name it **Lab7_Project.zip**.
6. Submit your project through the provided submission portal by the due date.

7.8.2 Rubric

Table 7.1: Rubric for Lab Exercise 7

Criteria	Excellent (5)	Good (4)	Average (2-3)	Needs Improvement (0-1)
Design and Layout	Aesthetic and highly responsive designs with creativity.	Clean design with minor responsiveness issues.	Basic design with significant improvement needed.	Poor or incomplete design.
Functionality	All features implemented correctly without errors.	Most features implemented with minor issues.	Features implemented with noticeable errors.	Features missing or not functional.
Code Structure	Well-structured and commented code.	Organized code with minimal comments.	Messy code with some structure.	Poorly structured code, lacks comments.
Interactivity	Excellent use of JavaScript for dynamic functionality.	Good interactivity with minor issues.	Limited interactivity.	Little to no interactivity.

Lab Module 8

BOOTSTRAP

Bootstrap is a popular open-source framework for building responsive and mobile-first websites. It was originally developed by Twitter and is now maintained by the Bootstrap development community. Bootstrap simplifies web development by providing a collection of pre-designed HTML, CSS, and JavaScript components.

Responsive Web Design (RWD) is an approach to web design that ensures a website's layout, content, and functionality adjust seamlessly to fit various screen sizes and device types, providing an optimal user experience. It enables websites to look and perform well on desktops, tablets, smartphones, and other devices without requiring separate designs for each platform.

8.1 Key Features of Bootstrap

- Bootstrap uses a 12-column grid system that allows developers to create layouts that adapt to different screen sizes.
- Includes ready-to-use UI components like buttons, forms, modals, navigation bars, carousels, alerts, and more.
- Developers can customize Bootstrap's appearance using its built-in Sass variables or by overriding its default CSS.
- Ensures that websites look consistent across modern browsers like Chrome, Firefox, Safari, and Edge.
- Provides utility classes for spacing, alignment, typography, and more, making it easy to style elements without writing additional CSS.

- Includes interactive components like dropdowns, modals, tooltips, and carousels, powered by JavaScript and jQuery.
- Designed to prioritize mobile devices by default, ensuring websites are optimized for smaller screens.

8.2 Getting Started with Bootstrap

There are two ways to start using Bootstrap on your own website. You can either:

- Download Bootstrap from getbootstrap.com, or
- Include Bootstrap from a CDN

8.2.1 Downloading Bootstrap

Downloading and using Bootstrap is straightforward. Here's a step-by-step guide:

Direct Download

1. Go to the official Bootstrap website: https://getbootstrap.com/.
2. Click on the Download button to download the latest version of Bootstrap.
3. You can download:
   - Compiled CSS and JS: Includes precompiled CSS and JavaScript files.
   - Source Files: Includes the CSS source files for customization.

Use a Content Delivery Network (CDN)

This is the easiest and fastest way to use Bootstrap without downloading files. Just add the following lines to your HTML file:

1. Add Bootstrap CSS:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
bootstrap/4.5.2/css/bootstrap.min.css">
```

2. Add Bootstrap JavaScript :

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/
bootstrap.min.js"></script>
```

8.3 Create First Web Page with Bootstrap

1. **Add the HTML5 doctype:** Bootstrap uses HTML elements and CSS properties that require the HTML5 doctype. Always include the HTML5 doctype at the beginning of the page, along with the lang attribute and the correct character set:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
    </head>
</html>
```

2. **Bootstrap 4 is mobile-first:** Bootstrap 4 is designed to be responsive to mobile devices. Mobile-first styles are part of the core framework. To ensure proper rendering and touch zooming, add the following `<meta>` tag inside the <head> element:

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

3. **Add link to CDN files:** If you do not want to download and host Bootstrap 4 yourself, you can include it from a CDN (Content Delivery Network). MaxCDN provides CDN support for Bootstrap's CSS and JavaScript. You must also include jQuery: (put them inside tag `<head>`

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
bootstrap/4.5.2/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/
jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/
1.16.0/umd/popper.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/
bootstrap.min.js"></script>
```

4. **Example of your First bootstrap page:** Below is an example of a basic Bootstrap 4 webpage that incorporates all the components mentioned in 1, 2, and 3 above (HTML structure, viewport meta tag, and Bootstrap links):

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

```
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Basic Bootstrap 4 Page</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
    bootstrap/4.5.2/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/
    3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/
    1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/
    js/bootstrap.min.js"></script>
</head>
<body>
    <!-- Navbar -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">My Website</a>
        <button class="navbar-toggler" type="button"
        data-toggle="collapse" data-target="#navbarNav">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav">
                <li class="nav-item active"><a class="nav-link"
                href="#">Home</a></li>
                <li class="nav-item"><a class="nav-link"
                href="#about">About</a></li>
                <li class="nav-item"><a class="nav-link"
                href="#services">Services</a></li>
                <li class="nav-item"><a class="nav-link"
                href="#contact">Contact</a></li>
            </ul>
        </div>
    </nav>

    <!-- Hero Section -->
    <div class="jumbotron text-center">
        <h1>Welcome to My Website</h1>
        <p>Professional IT consulting services</p>
    </div>
```

```
<!-- Responsive Container -->
<div class="container">
    <!-- About Section -->
    <section id="about">
        <h2>About Us</h2>
        <p>We are a startup dedicated to providing innovative IT
        consulting solutions to businesses worldwide.</p>
    </section>

    <!-- Services Section -->
    <section id="services">
        <h2>Our Services</h2>
        <ul>
            <li>IT Strategy and Planning</li>
            <li>Cloud Computing</li>
            <li>Cybersecurity Consulting</li>
            <li>Software Development</li>
        </ul>
    </section>

    <!-- Contact Section -->
    <section id="contact">
        <h2>Contact Us</h2>
        <form>
            <div class="form-group">
                <label for="name">Name:</label>
                <input type="text" class="form-control" id="name"
                placeholder="Enter your name">
            </div>
            <div class="form-group">
                <label for="email">Email:</label>
                <input type="email" class="form-control"
                id="email" placeholder="Enter your email">
            </div>
            <div class="form-group">
                <label for="message">Message:</label>
                <textarea class="form-control" id="message"
                rows="3" placeholder="Your message"></textarea>
            </div>
            <button type="submit" class="btn btn-primary">
```

```
                Submit</button>
            </form>
        </section>
    </div>

    <!-- Footer -->
    <footer class="text-center mt-5 bg-dark text-white py-3">
        <p>&copy; 2024 FutureTech Solutions. All rights reserved.
        </p>
    </footer>
</body>
</html>
```

8.4 Responsive Image

Images come in all sizes. So do screens. Responsive images automatically adjust to fit the size of the screen.

Create responsive images by adding an `.img-fluid` class to the `<img>` tag. The image will then scale nicely with the parent element.

The `.img-fluid` class applies `max-width: 100%;` and `height: auto;` to the image.

8.5 Bootstrap Text/Typography

Bootstrap provides a robust set of utilities and classes for styling text and typography, making it easy to create visually appealing and consistent text across your website. Here's an overview of the typography features in Bootstrap:

8.5.1 Headings

Bootstrap offers predefined heading classes (.h1, .h2, etc.) to style headings.

```
<h1 class="h1">Heading 1</h1>
<h2 class="h2">Heading 2</h2>
<h3 class="h3">Heading 3</h3>
<h4 class="h4">Heading 4</h4>
<h5 class="h5">Heading 5</h5>
<h6 class="h6">Heading 6</h6>
```

8.5.2 Display Headings

For larger and more prominent headings, use display classes (`.display-1` to `.display-6`).

```
<h1 class="display-1">Display 1</h1>
<h2 class="display-2">Display 2</h2>
<h3 class="display-3">Display 3</h3>
<h4 class="display-4">Display 4</h4>
<h5 class="display-5">Display 5</h5>
<h6 class="display-6">Display 6</h6>
```

8.5.3 Inline Text Elements

- **Bold text:** Use `<strong>` or `.fw-bold` .
- **Italic text:** Use `<em>` or `.fst-italic` .
- **Small text:** Use `<small>` or `.fs-6` .
- **Underline:** Use `<u>` for underlined text.
- **Strikethrough:** Use `<s>` for strikethrough text.
- **Monospace:** Use `<code>` or `.font-monospace` .

```
<p class="text-start">Left-aligned text.</p>
<p class="text-center">Center-aligned text.</p>
<p class="text-end">Right-aligned text.</p>
```

8.5.4 Text Color and Background

- **Text color:** Use classes like `.text-primary` , `.text-success` , `.text-danger` , `.text-warning` , `.text-info` , `.text-dark` , `.text-body` , `.text-muted` , `.text-white` , and `.text-black-50` .
- **Background color:** Use `.bg-primary` , `.bg-success` , etc.

```
<p class="text-primary">Primary colored text.</p>
<p class="bg-warning text-dark">Text with background color.</p>
```

8.5.5 Text Wrapping and Overflow

- **No wrapping:** Use `.text-nowrap` to prevent text wrapping.
- **Truncation:** Use `.text-truncate` to truncate long text with an ellipsis.
- **Breakable text:** Use `.text-break` for breaking long words.

```
<div class="text-nowrap">This text will not wrap.</div>
<div class="text-truncate" style="width: 200px;">This text will truncate
if it is too long to fit in the available width.</div>
<div class="text-break">ThisIsAReallyLongWordThatWillBreakOntoANewLine.
</div>
```

8.5.6 Font Size

Bootstrap provides `.fs-*` classes to adjust font sizes: `.fs-1` (largest) to `.fs-6` (smallest).

```
<p class="fs-1">Large text</p>
<p class="fs-3">Medium text</p>
<p class="fs-6">Small text</p>
```

8.5.7 Font Weight and Style

- **Font weight:** `.fw-bold`, `.fw-normal`, `.fw-light`.
- **Font style:** `.fst-italic`, `.fst-normal`.

```
<p class="fw-bold">Bold text</p>
<p class="fw-light fst-italic">Light and italic text</p>
```

8.5.8 Line Height

Use `.lh-*` classes to adjust line height: `.lh-1`, `.lh-sm`, `.lh-base`, `.lh-lg`.

```
<p class="lh-lg">This text has a larger line height.</p>
```

8.5.9 Lists

Bootstrap styles lists with utilities for inline and unstyled lists.

Unstyled List:

```
<ul class="list-unstyled">
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```

Inline List:

```
<ul class="list-inline">
  <li class="list-inline-item">Item 1</li>
  <li class="list-inline-item">Item 2</li>
</ul>
```

8.6 Bootstrap CSS Forms

Bootstrap provides a comprehensive set of classes and components for creating stylish and responsive forms. Here's an overview of the features and utilities Bootstrap offers for form design:

8.6.1 Basic Form Structure

Bootstrap uses the `<form>` element with `.form-control` and `.form-label` classes for input elements and labels.

```
<form>
  <div class="mb-3">
    <label for="exampleInputEmail" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail"
    placeholder="name@example.com">
  </div>
  <div class="mb-3">
    <label for="exampleInputPassword" class="form-label">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword">
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

8.6.2 Form Layouts

Vertical Forms (Default)

Fields are stacked vertically by default. Use `.mb-3` for spacing between form groups.

Horizontal Forms

Add `.row` to align labels and inputs horizontally and use `.col-*` classes for layout.

```
<form>
  <div class="row mb-3">
    <label for="inputEmail" class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="inputEmail">
    </div>
  </div>
  <div class="row mb-3">
    <label for="inputPassword" class="col-sm-2 col-form-label">Password
    </label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword">
    </div>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

8.6.3 Inline Forms

```
<form class="d-flex">
  <input class="form-control me-2" type="search" placeholder="Search">
  <button class="btn btn-outline-success" type="submit">Search</button>
</form>
```

8.6.4 Form Controls

Bootstrap provides styles for common form controls:

- **Input Types:** text, email, password, file, number, date, etc.
- **Textarea:** Use `<textarea>` with `.form-control`.

```
<div class="mb-3">
  <label for="exampleTextarea" class="form-label">Example Textarea</label>
  <textarea class="form-control" id="exampleTextarea" rows="3"></textarea>
</div>
```

8.6.5 Select Menus

Use `<select>` with `.form-select`.

```
<div class="mb-3">
  <label for="exampleSelect" class="form-label">Select Example</label>
  <select class="form-select" id="exampleSelect">
    <option>Option 1</option>
    <option>Option 2</option>
    <option>Option 3</option>
  </select>
</div>
```

8.6.6 Form Validation

Bootstrap includes validation classes to style forms based on their state.

```
<form>
  <div class="mb-3">
    <label for="validationExample" class="form-label">Username</label>
    <input type="text" class="form-control is-valid" id="validationExample"
    value="Correct Input">
    <div class="valid-feedback">Looks good!</div>
  </div>
  <div class="mb-3">
    <label for="validationExampleInvalid" class="form-label">Password</label>
    <input type="password" class="form-control is-invalid"
    id="validationExampleInvalid">
    <div class="invalid-feedback">Please choose a stronger password.</div>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

8.6.7 Checkboxes and Radios

Use `.form-check` for styling checkboxes and radio buttons.

Checkboxes:

```
<div class="form-check">
  <input class="form-check-input" type="checkbox" id="exampleCheck">
  <label class="form-check-label" for="exampleCheck">Check me out</label>
</div>
```

Radio Buttons:

```
<div class="form-check">
  <input class="form-check-input" type="radio" name="exampleRadios"
  id="radio1" value="option1" checked>
  <label class="form-check-label" for="radio1">Option 1</label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="exampleRadios"
  id="radio2" value="option2">
  <label class="form-check-label" for="radio2">Option 2</label>
</div>
```

8.6.8 Input Groups

Combine inputs and add-ons (like buttons or text) with `.input-group`.

```
<div class="input-group mb-3">
  <span class="input-group-text" id="basic-addon1">@</span>
  <input type="text" class="form-control" placeholder="Username"
  aria-label="Username" aria-describedby="basic-addon1">
</div>
```

8.6.9 Floating Labels

Use `.form-floating` for labels inside inputs.

```
<div class="form-floating mb-3">
  <input type="email" class="form-control" id="floatingInput"
  placeholder="name@example.com">
  <label for="floatingInput">Email address</label></div>
```

8.6.10 Disabled and Readonly Inputs

Disabled:

```
<input type="text" class="form-control" placeholder="Disabled input"
disabled>
```

Readonly:

```
<input type="text" class="form-control" placeholder="Readonly input"
readonly>
```

8.6.11 Example Forms

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Bootstrap \index{Bootstrap} Forms \index{Forms (HTML)}</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/
  bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <div class="container mt-5">
    <form>
      <div class="mb-3">
        <label for="inputEmail" class="form-label">Email address</label>
        <input type="email" class="form-control" id="inputEmail"
        placeholder="name@example.com">
      </div>
      <div class="mb-3">
        <label for="inputPassword" class="form-label">Password</label>
        <input type="password" class="form-control" id="inputPassword">
      </div>
      <div class="form-check">
        <input type="checkbox" class="form-check-input" id="exampleCheck">
        <label class="form-check-label" for="exampleCheck">Remember me
        </label>
      </div>
      <button type="submit" class="btn btn-primary mt-3">Submit</button>
```

```
    </form>
  </div>
</body>
</html>
```

8.7 Using Bootstrap Theme

Using a Bootstrap theme allows you to quickly style your web application with pre-designed components and layouts that go beyond the default Bootstrap framework. Here's a guide on how to use a Bootstrap theme:

1. Find a Bootstrap Theme: You can download free or premium Bootstrap themes from popular platforms.
   - Official Bootstrap Themes: https://themes.getbootstrap.com/
   - Third-Party Websites:
     - Start Bootstrap
     - ThemeForest
     - Bootswatch
2. Once you select a theme:
   (a) Download the theme as a ZIP file.
   (b) Extract the ZIP file to a folder on your computer. The folder usually includes:
   - **HTML files:** Example pages and templates.
   - **CSS files:** Pre-styled Bootstrap components.
   - **JS files:** Optional JavaScript for additional functionality.
   - **Assets:** Fonts, images, and icons.
3. Replace or include the theme's CSS and JS files in your project.

```
project/
├── css/
│   ├── theme.css
│   ├── bootstrap.min.css
├── js/
│   ├── bootstrap.min.js
│   ├── theme.js
├── index.html
├── assets/
│   ├── fonts/
│   ├── images/
```

Figure 8.1: Basic File Structure

4. Replace the default Bootstrap CDN links with the theme's files.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
  scale=1.0">
  <title>Bootstrap Theme Example</title>
  <!-- Theme CSS -->
  <link rel="stylesheet" href="css/theme.css">
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container">
      <a class="navbar-brand" href="#">My Theme</a>
    </div>
  </nav>
  <div class="container mt-5">
    <h1>Welcome to My Themed Website</h1>
    <p>This is an example of a page styled with a Bootstrap theme.
    </p>
  </div>
  <!-- Theme JavaScript -->
  <script src="js/theme.js"></script>
```

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

5. Many themes allow further customization. Open the theme's CSS file and adjust properties such as colors, fonts, or spacing. Example, modify the theme's primary color in theme.css:

```
:root {
  --bs-primary: #ff5722; /* Change primary color to orange */
}
```

6. Most themes include prebuilt templates for pages like:

   - Home
   - About
   - Contact
   - Portfolio

   You can copy and edit these templates to suit your needs.

7. Example: Themed Page

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
  initial-scale=1.0">
  <title>Themed Bootstrap Page</title>
  <link href="css/theme.css" rel="stylesheet">
</head>
<body>
  <header class="bg-primary text-white text-center py-5">
    <h1>Welcome to My Themed Page</h1>
    <p>Using a Bootstrap theme</p>
  </header>
  <div class="container mt-5">
    <div class="row">
      <div class="col-md-6">
        <h2>About</h2>
        <p>This theme gives a fresh and modern look to your website.</p>
```

```
        </div>
        <div class="col-md-6">
          <img src="assets/images/example.jpg" class="img-fluid rounded"
          alt="Example">
        </div>
      </div>
    </div>
  </body>
  </html>
```

8. Open your **index.html** in a web browser. Verify that the styles and components match the downloaded theme.

8.8 Lab Exercise 8

Students will use a Bootstrap theme to create a responsive website for a fictional business. They will customize the theme to meet the branding and functionality requirements of the business.

8.8.1 Building a Themed Website for a Fictional Business

Instruction

You have been hired by "**FutureTech Solutions**", a startup that offers IT consulting services. The company needs a simple but professional website with the following features:

- **Home Page:** Brief introduction and company highlights.
- **About Page:** Information about the company and its mission.
- **Services Page:** A list of services offered.
- **Contact Page:** A form to capture customer inquiries.

Download a free Bootstrap theme from a platform like Start Bootstrap or Bootswatch. Customize the theme to match FutureTech's branding by changing colors, adding a logo, and updating the text content. Ensure all navigation links work, the site is fully responsive, and test it on various devices. For a polished touch, consider hosting the site on GitHub Pages for a live demo.

8.8.2 Rubric

Table 8.1: Rubric for Lab Exercise 8

Criteria	Excellent (5)	Good (4)	Average (2-3)	Needs Improvement (0-1)
Design and Layout	Aesthetic and highly responsive design with creative customization.	Clean design with minor responsiveness or aesthetic issues.	Basic design; lacks appeal and needs significant improvement.	Poor or incomplete design with major flaws.
Functionality	All features fully implemented and functional without errors.	Most features implemented; minor issues present.	Some features implemented with noticeable errors or gaps.	Features are missing, incomplete, or not functional.
Code Structure	Clean, well-organized, and thoroughly commented code.	Structured and organized code with some comments.	Partially structured code; lacks consistency and sufficient comments.	Disorganized or poorly written code with no comments.
Interactivity	Excellent use of Bootstrap and JavaScript to enhance user interactivity.	Good interactivity with minor improvements needed.	Limited interactivity; minimal dynamic features.	No or very minimal interactivity implemented.
Customization	Exceptional branding and theme customization, fully aligned with requirements.	Good customization; meets most branding requirements.	Minimal customization; partially aligned with requirements.	No significant customization; default theme used with minimal changes.

References

Websites and Official Documentations

- Can I Use?. (2025). *Supports for HTML5, CSS3, etc.* Retrieved from https://caniuse.com/.
- CSS Tricks. (2025). *A Website About Making Websites.* Retrieved from https://css-tricks.com/.
- Mozilla Developer Network (MDN). (2025). *MDN Web Docs - HTML, CSS, JavaScript.* Retrieved from https://developer.mozilla.org/
- W3Schools. (2025). *HTML, CSS, and JavaScript Tutorials.* Retrieved from https://www.w3schools.com/.
- WHATWG. (2025). *Web Hypertext Application Technology Working Group.* Retrieved from https://whatwg.org/

Standard and Guidelines

- Web Content Accessibility Guidelines (WCAG). (2025). *Web Accessibility Initiative.* Retrieved from https://www.w3.org/WAI/standards-guidelines/wcag/
- World Wide Web Consortium (W3C). (2025). *Web standards and guidelines.* Retrieved from https://www.w3.org/

Index