WEB DEVELOPMENT LESSON
BY :

FLINIX TECH
PIONEERING TOMOROW'S TECHNOLOGY TODAY

AND

FOR BEGINNERS

PREPARED BY:
MUHAMMAD HUSSAIN GHALI

Table of Contents

Introduction to Web Development

What is Web Development?

Web Development is the process of creating websites or web applications that can be viewed using web browsers like Google Chrome, Firefox, Safari, and Edge. It involves building the structure, design, functionality, and interactive elements of a website.

At its core, web development focuses on:

- **Creating Content**: Using HTML (HyperText Markup Language) to add text, images, videos, and links to a website.
- **Styling Content**: Using CSS (Cascading Style Sheets) to make the website visually appealing.
- **Adding Interactivity**: Using JavaScript to make websites dynamic (e.g., buttons that react when clicked).

Web development allows businesses, individuals, and organizations to have an online presence where they can:

1. Share information (e.g., blogs, news websites).
2. Provide services or sell products (e.g., e-commerce stores like Amazon).
3. Offer platforms for communication (e.g., social media like Facebook and Twitter).

Web development can range from simple static websites (e.g., a single page of text and images) to complex web applications (e.g., Facebook, YouTube).

Frontend vs Backend

Web development is divided into **Frontend** and **Backend**.

Frontend Development (Client-Side)

- **What is it?**
  Frontend development focuses on everything that the user **sees and interacts with** on a website. It is the visual and interactive part of a website.
- **Technologies Used**:

    - **HTML** (HyperText Markup Language): Used to structure content (e.g., headings, paragraphs, images).
    - **CSS** (Cascading Style Sheets): Used to style and design the content (e.g., colors, fonts, layout).
    - **JavaScript**: Used to add interactivity (e.g., sliders, animations, form validations).
- **Examples of Frontend Work**:
    - Adding buttons, forms, images, and videos to a website.
    - Designing a navigation menu and layout.
    - Making a button change color when hovered over.

Backend Development (Server-Side)

- **What is it?**
  Backend development focuses on the "behind the scenes" operations of a website. It deals with databases, servers, and logic that ensure the website works as intended.
- **Technologies Used**:
    - Programming languages: **PHP**, **Python**, **Java**, **Node.js**.
    - Databases: **MySQL**, **MongoDB**, **PostgreSQL**.
    - Server tools: **Apache**, **Nginx**, and cloud services like **AWS**.
- **Examples of Backend Work**:
    - Saving user data when they submit a form (e.g., registration form).
    - Managing login systems where users enter their usernames and passwords.
    - Handling product purchases in an online store.

Full Stack Development

A **Full Stack Developer** is someone who works on both the **frontend** and **backend** of a website. They can create a complete, functional website from scratch.

Understanding Websites and Browsers

What is a Website?

A **website** is a collection of web pages that are accessible via the internet using a browser.

- **Web Page**: A single document on the internet (e.g., a single page of text, images, and links).
- **Website**: A group of related web pages that are connected by links.

Websites can be:

- **Static**: Content does not change and remains the same for all users.
- **Dynamic**: Content changes based on user input or database updates (e.g., Facebook or YouTube).

How Do Websites Work?

1. **User Request**: A user types a website address (URL) into their browser, like `www.example.com`.
2. **Browser Contacting the Server**:
    - The browser sends a request to the website's **server**.
    - A server is a computer that stores the website's files.
3. **Server Response**:
    - The server sends the required web page files (HTML, CSS, images, etc.) back to the browser.
4. **Displaying the Website**:
    - The browser reads the files and displays the website to the user.

What is a Browser?

A **web browser** is software that allows users to access websites. Examples of browsers include:

- **Google Chrome**
- **Mozilla Firefox**
- **Safari**
- **Microsoft Edge**

How Browsers Work:

1. The browser sends a request to the web server for a webpage.
2. The server sends back the necessary code (HTML, CSS, and JavaScript).
3. The browser **reads and processes** the code, displaying the final website.

Website Components

Websites are made up of the following core components:

1. **HTML**: Adds structure and content (headings, images, text, links).
2. **CSS**: Adds styling and design (colors, fonts, layouts).
3. **JavaScript**: Adds interactivity (animations, form validation).

When these components are combined, they allow developers to build fully functional and visually appealing websites.

Summary

- **Web Development** is about building websites that can be viewed using browsers.
- It is divided into **Frontend** (visible part) and **Backend** (hidden logic and data).
- Websites work by combining HTML, CSS, and JavaScript to create functional and interactive pages.
- Browsers act as a "bridge" between the user and the website, displaying content sent by the server.

Tools and Setup

Code Editor (Visual Studio Code)

What is a Code Editor?

A **code editor** is a software tool that allows developers to write and edit code efficiently. It highlights syntax, provides auto-completion, and helps debug code, making coding much easier.

One of the most popular code editors for web development is **Visual Studio Code (VS Code)**.

Why Use Visual Studio Code?

- **Free**: It's free to download and use.
- **Lightweight**: It runs fast and doesn't consume a lot of system resources.
- **Customizable**: It allows extensions, themes, and plugins to improve productivity.
- **Multi-Language Support**: It supports multiple programming languages, including **HTML**, **CSS**, and **JavaScript**.
- **Live Preview**: Extensions allow you to see the changes you make to your code in real-time.

Steps to Install Visual Studio Code

1. Go to the official **Visual Studio Code**
2. Click the **Download** button for your operating system (Windows, macOS, or Linux).
3. Run the downloaded file and follow the installation instructions.
4. After installation, open **VS Code**.

Creating and Running a Web Page

Once you have your `index.html` file ready, you can run and view it in your browser. Follow these steps:

1. **Save the File**:
   Press `Ctrl + S` (Windows) or `Cmd + S` (Mac) to save your HTML file.
2. **Open in Browser**:

- Go to the folder where your project is saved.
- Double-click on the `index.html` file.
- It will automatically open in your default browser (e.g., Chrome, Firefox, or Safari).

3. **View the Output**:

   You should see the following output in your browser:

Welcome to My First Web Page!

This is a simple HTML file.

Using Live Server in VS Code (Optional)

To speed up development and see real-time updates, you can use the **Live Server extension** in VS Code.

Steps to Install Live Server:

1. Go to the **Extensions** section in VS Code (left sidebar, or press `Ctrl + Shift + X`).
2. Search for “Live Server” and install it.
3. After installation, right-click on your `index.html` file and select **Open with Live Server**.
4. Your website will open in your browser, and any changes you make will be reflected immediately after saving the file.

Lesson 1: HTML Basics

What is HTML?

HTML stands for **HyperText Markup Language**. It is the standard language used to create and structure web pages. HTML provides the **skeleton** or **framework** of a website by using a system of **tags**.

- **HyperText**: Refers to text that contains links to other documents or pages.
- **Markup Language**: Uses tags to describe the structure and content of a web page.

HTML files are simple text documents saved with the **.html** extension. Browsers like Chrome, Firefox, and Safari read these files and display them as web pages.

Basic HTML Structure

Every HTML document follows a basic structure that includes essential tags. Below is an example of the basic structure of an HTML file:

Code Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My First Web Page</title>
  </head>
  <body>
    <h1>Welcome to My Website</h1>
    <p>This is my first HTML page!</p>
  </body>
</html>
```

Explanation:

1. `<!DOCTYPE html>`: Declares that the document is an HTML5 document.
2. `<html>`: The root element; all content goes inside this tag.
3. `<head>`: Contains meta-information about the document, like the title.

- `<title>`: Sets the title of the web page (shown in the browser tab).

4. `<body>`: Contains all the visible content of the web page.
    - `<h1>`: A heading element.
    - `<p>`: A paragraph of text.!

Text Elements (Headings, Paragraphs)

Headings

Headings are used to display titles or subtitles. HTML has six levels of headings, from `<h1>` (largest) to `<h6>` (smallest).

Code Example:

```
<h1>This is Heading 1</h1>
<h2>This is Heading 2</h2>
<h3>This is Heading 3</h3>
<h4>This is Heading 4</h4>
<h5>This is Heading 5</h5>
<h6>This is Heading 6</h6>
```

Paragraphs

The `<p>` tag is used to write paragraphs of text.

Code Example:

```
<p>This is the first paragraph of my web page.</p>
<p>This is another paragraph.</p>
```

Links and Images

Links

Links allow users to navigate to other web pages or external websites. The `<a>` tag is used for links, and the `href` attribute specifies the URL.

Code Example:

```
<a href="https://www.google.com">Visit Google</a>
```

Code Example:

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

Unordered List

Unordered lists display items with bullet points. Use the `<ul>` tag for unordered lists and `<li>` for list items.

Code Example:

```
<h3>Fruits</h3>
<ul>
  <li>Apple</li>
  <li>Banana</li>
  <li>Cherry</li>
</ul>
```

Code Example:

```
<h3>Steps to Make Tea</h3>
<ol>
  <li>Boil water</li>
  <li>Add tea leaves</li>
  <li>Pour into a cup</li>
</ol>
```

Lesson 2: Introduction to CSS

What is CSS?

CSS stands for **Cascading Style Sheets**. It is a language used to style and design the appearance of web pages. While HTML provides the structure of a webpage, CSS adds visual style, including colors, fonts, layouts, and spacing.

Why Use CSS?

1. **Separation of Content and Style**: CSS separates the visual design from the HTML structure, making it easier to manage.
2. **Consistency**: You can apply consistent styles across multiple pages.
3. **Customization**: CSS allows full control over the layout and appearance of a website.

What is "Cascading" in CSS?

The term "Cascading" means that when multiple rules apply to the same element, the rule with the highest **priority** is used. CSS follows a specific order:

1. **Inline styles** (highest priority).
2. **Internal styles**.
3. **External styles**.

How to Add CSS

There are three main ways to add CSS to an HTML document: **Inline**, **Internal**, and **External**.

1. Inline CSS

Inline CSS applies styles directly to individual HTML elements using the `style` attribute.

Syntax:

```
<p style="color: red; font-size: 20px;">This is an inline styled paragraph.</p>
```

When to Use:

- For quick changes or testing styles.
- Not recommended for large-scale projects because it makes the code messy.

2. Internal CSS

Internal CSS is written within the `<style>` tag inside the `<head>` section of an HTML document.

Syntax:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p {
        color: blue;
        font-size: 18px;
      }
    </style>
  </head>
  <body>
    <p>This paragraph is styled using internal CSS.</p>
  </body>
</html>
```

When to Use:

- When styling a single HTML document.
- Good for small projects or when external CSS is unnecessary.

3. External CSS

External CSS uses a separate file with the `.css` extension, which is linked to the HTML document using the `<link>` tag.

Steps:

1. Create a CSS file (e.g., `style.css`).
2. Write your styles in the CSS file.
3. Link the CSS file to the HTML document using the `<link>` tag in the `<head>` section.

HTML Example:

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <p>This paragraph is styled using external CSS.</p>
  </body>
</html>
```

CSS File (style.css):

```
p {
  color: green;
  font-size: 20px;
}
```

When to Use:

- For larger projects.
- When styles need to be applied to multiple pages for consistency.

Basic Styling: Colors, Backgrounds, and Fonts

1. Colors

CSS allows you to set text color and background color using:

- **Named Colors**: Example: `red`, `blue`, `green`.
- **Hexadecimal Values**: Example: `#ff0000` (red).
- **RGB Values**: Example: `rgb(255, 0, 0)` (red).

Syntax:

```
p {
  color: red; /* Text color */
  background-color: yellow; /* Background color */
}
```

2. Backgrounds

CSS allows you to set images or solid colors as backgrounds.

Example:

```
body {
  background-color: lightgray;
```

```
  background-image: url('background.jpg');
  background-size: cover; /* Adjust the image to cover the entire page */
}
```

3. Fonts

You can change the font family, size, and style of text.

Example:

```
p {
  font-family: Arial, sans-serif; /* Font type */
  font-size: 20px; /* Font size */
  font-weight: bold; /* Font weight: normal, bold, etc. */
  font-style: italic; /* Font style: normal, italic */
}
```

Margins, Padding, and Borders

CSS provides properties to control spacing and borders around elements.

1. Margins

The **margin** property controls the space **outside** an element.

Syntax:

```
div {
  margin: 20px; /* Adds 20px space outside the element */
}
```

- `margin: 10px;` → Adds the same margin on all four sides.
- `margin: 10px 20px;` → Adds 10px top/bottom and 20px left/right.

2. Padding

The **padding** property controls the space **inside** an element, between the content and the border.

Syntax:

```
div {
  padding: 15px; /* Adds space inside the element */
}
```

3. Borders

The **border** property adds a border around an element. You can specify the **style**, **width**, and **color**.

Syntax:

```
div {
  border: 2px solid black; /* 2px wide, solid black border */
}
```

Combined Example:

```
div {
  margin: 20px;
  padding: 15px;
  border: 2px solid blue;
  background-color: lightyellow;
}
```

Lesson 3: Building a Simple Web Page

Step-by-Step Guide to Build a Website

In this lesson, we will combine what we have learned about HTML and CSS to build a simple web page step by step. By the end of this lesson, you will be able to:

1. Create an **HTML home page**.
2. Add content such as headings, paragraphs, links, and images.
3. Apply **CSS styling** to make the page visually appealing.

1. Creating a Home Page (HTML)

Follow these steps to create a simple HTML home page:

Step 1: Create Your Project Folder

1. Create a new folder on your computer and name it `my-website`.
2. Inside the folder, create a file named `index.html`. This will be your home page.
3. Also, create a folder called `css` where you will store your CSS file.

Step 2: Write the HTML Structure

Here is the basic HTML code for your home page:

HTML Code (index.html):

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My First Website</title>
    <link rel="stylesheet" href="css/style.css">
```

```
  </head>
  <body>
    <!-- Header Section -->
    <header>
      <h1>Welcome to My First Website</h1>
      <p>This is a simple website built with HTML and CSS.</p>
    </header>

    <!-- Navigation Section -->
    <nav>
      <a href="#">Home</a> |
      <a href="#">About</a> |
      <a href="#">Contact</a>
    </nav>

    <!-- Main Content Section -->
    <main>
      <h2>About Me</h2>
      <p>Hello! My name is John, and this is my first web page. I’m learning web
development step by step.</p>
      <h3>My Favorite Things:</h3>
      <ul>
        <li>Reading Books</li>
        <li>Coding</li>
        <li>Watching Movies</li>
      </ul>

      <!-- Adding an Image -->
      <img src="https://via.placeholder.com/300" alt="Sample Image">
    </main>

    <!-- Footer Section -->
    <footer>
      <p>&copy; 2024 My First Website. All rights reserved.</p>
    </footer>
  </body>
</html>
```

Explanation:

1. The `<header>` section contains the main heading of the website.
2. The `<nav>` section contains navigation links (these do not lead anywhere yet).
3. The `<main>` section contains the main content, such as headings, paragraphs, a list, and an image.
4. The `<footer>` section contains copyright information.

2. Styling the Home Page (CSS)

Now that we have written the HTML, let’s style it using CSS.

1. Inside the `my-website` folder, go to the `css` folder.
2. Create a new file named `style.css`.

Write the CSS Code

Here is the CSS to style the home page:

CSS Code (css/style.css):

```
/* General Styles */
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f8f9fa;
  color: #333;
}

header {
  background-color: #007bff;
  color: white;
  text-align: center;
  padding: 20px;
}

nav {
  background-color: #0056b3;
  text-align: center;
  padding: 10px;
}

nav a {
  color: white;
  text-decoration: none;
  margin: 0 10px;
  font-weight: bold;
}

main {
  padding: 20px;
}

h1, h2, h3 {
  color: #007bff;
}

ul {
  list-style-type: square;
}

img {
  display: block;
  margin: 20px auto;
  border: 2px solid #ccc;
```

```
}

footer {
  background-color: #333;
  color: white;
  text-align: center;
  padding: 10px;
  position: absolute;
  bottom: 0;
  width: 100%;
}
```

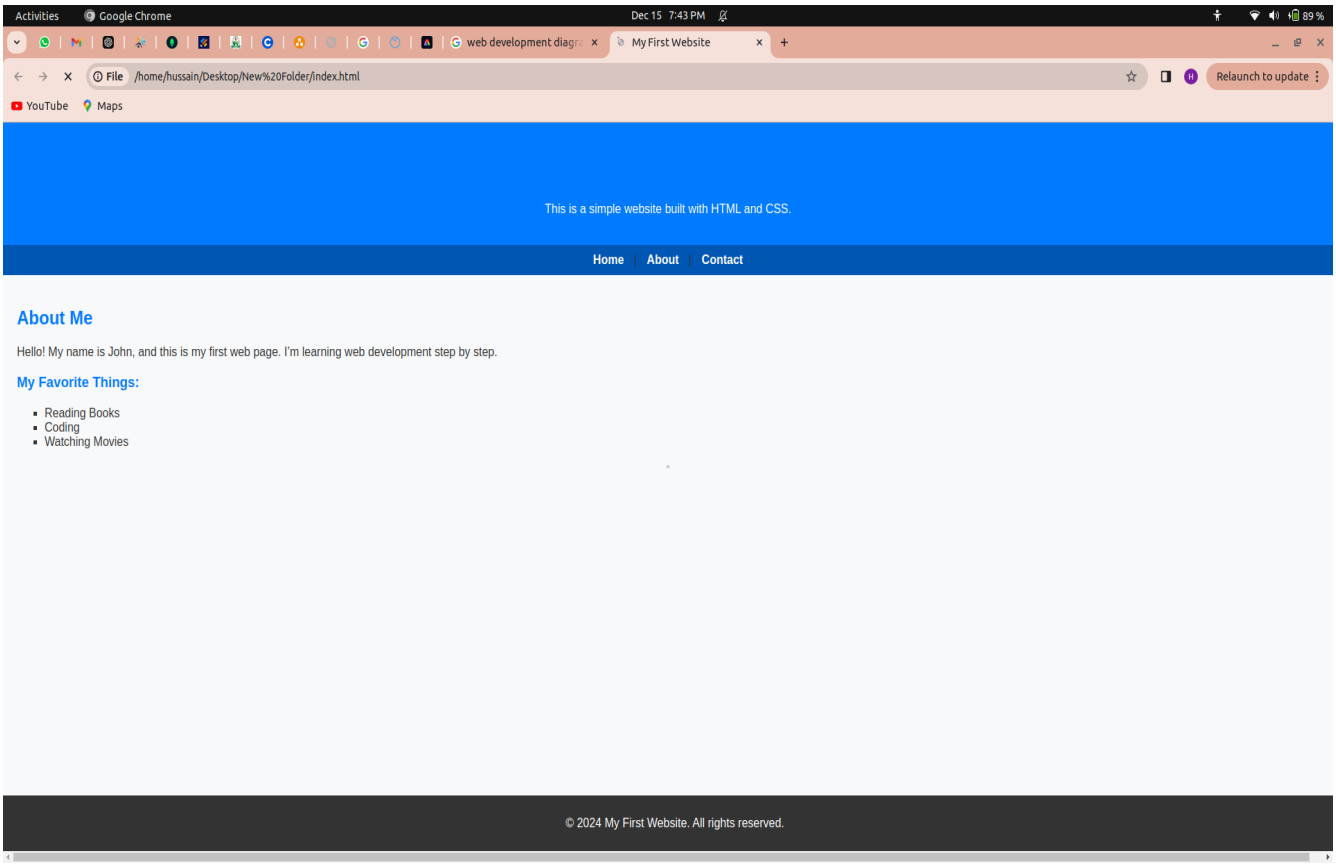
Explanation of the CSS:

1. **General Styles**:
    - `font-family: Arial, sans-serif`: Sets a clean, readable font.
    - `margin: 0; padding: 0`: Removes default spacing.
    - `background-color` and `color`: Sets the background and text colors.
2. **Header Section**:
    - Adds a blue background and white text.
    - Aligns the heading to the center.
3. **Navigation Section**:
    - Adds a darker blue background.
    - Styles links to have no underlines and adds spacing.
4. **Main Section**:
    - Adds padding for spacing.
    - Changes heading colors to blue.
5. **Image**:
    - Centers the image and adds a border.
6. **Footer Section**:
    - Adds a dark background and white text.

How to Run the Website

1. Save the `index.html` and `style.css` files.

2. Open the `index.html` file in a browser (e.g., Chrome, Firefox).
3. You will see your styled home page.



`Diagram of the Designed website`

6. Lesson 4: Multi-Page Website

Introduction

In the previous lessons, you created a single-page website using **HTML** and **CSS**. However, most websites consist of **multiple pages** linked together. In this lesson, you will learn how to create multiple web pages and connect them using **navigation links**.

1. Linking Multiple Pages Together

Web pages are connected using **hyperlinks**, which are created using the `<a>` tag in HTML. The `href` attribute specifies the file or page to link to.

Steps to Create Multiple Pages

1. **Create Your Project Folder**:
    - Start with a folder for your project (e.g., `multi-page-website`).
2. **Create the HTML Files**:
    - Create a main page (`index.html`), an about page (`about.html`), and a contact page (`contact.html`).

Creating the Pages

Here's the structure of your project folder:

```
multi-page-website/
│
├── index.html        (Home Page)
├── about.html        (About Page)
├── contact.html      (Contact Page)
└── css/
    └── style.css     (CSS File)
```

Main Page (index.html)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home Page</title>
    <link rel="stylesheet" href="css/style.css">
  </head>
  <body>
    <!-- Header Section -->
    <header>
      <h1>Welcome to My Multi-Page Website</h1>
      <p>This is the home page.</p>
    </header>

    <!-- Navigation Section -->
    <nav>
      <a href="index.html">Home</a> |
      <a href="about.html">About</a> |
```

```
      <a href="contact.html">Contact</a>
    </nav>

    <!-- Content Section -->
    <main>
      <p>Welcome to the home page of my multi-page website. Use the navigation
links to explore the other pages.</p>
    </main>

    <!-- Footer Section -->
    <footer>
      <p>&copy; 2024 My Multi-Page Website</p>
    </footer>
  </body>
</html>
```

About Page (about.html)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About Page</title>
    <link rel="stylesheet" href="css/style.css">
  </head>
  <body>
    <!-- Header Section -->
    <header>
      <h1>About Us</h1>
      <p>Learn more about us on this page.</p>
    </header>

    <!-- Navigation Section -->
    <nav>
      <a href="index.html">Home</a> |
      <a href="about.html">About</a> |
      <a href="contact.html">Contact</a>
    </nav>

    <!-- Content Section -->
    <main>
      <p>This is the about page. Here, you can write information about yourself or
your website.</p>
    </main>

    <!-- Footer Section -->
    <footer>
      <p>&copy; 2024 My Multi-Page Website</p>
    </footer>
  </body>
</html>
```

Contact Page (contact.html)

```
<!DOCTYPE html>
```

```
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact Page</title>
    <link rel="stylesheet" href="css/style.css">
  </head>
  <body>
    <!-- Header Section -->
    <header>
      <h1>Contact Us</h1>
      <p>Get in touch with us!</p>
    </header>

    <!-- Navigation Section -->
    <nav>
      <a href="index.html">Home</a> |
      <a href="about.html">About</a> |
      <a href="contact.html">Contact</a>
    </nav>

    <!-- Content Section -->
    <main>
      <p>This is the contact page. You can provide contact details here.</p>
    </main>

    <!-- Footer Section -->
    <footer>
      <p>&copy; 2024 My Multi-Page Website</p>
    </footer>
  </body>
</html>
```

2. Adding Navigation Links

In the above examples, the navigation bar was created using the `<nav>` element and the `<a>` (anchor) tags.

Understanding the `<a>` Tag:

- `href` specifies the page to link to.
- The text between the opening and closing `<a>` tags is the clickable text.

Example:

```
<nav>
  <a href="index.html">Home</a>
  <a href="about.html">About</a>
  <a href="contact.html">Contact</a>
</nav>
```

- `index.html`, `about.html`, and `contact.html` are the file names of the pages.

Styling the Navigation Links

To make the navigation links look better, you can add styles using **CSS**.

CSS Code (style.css):

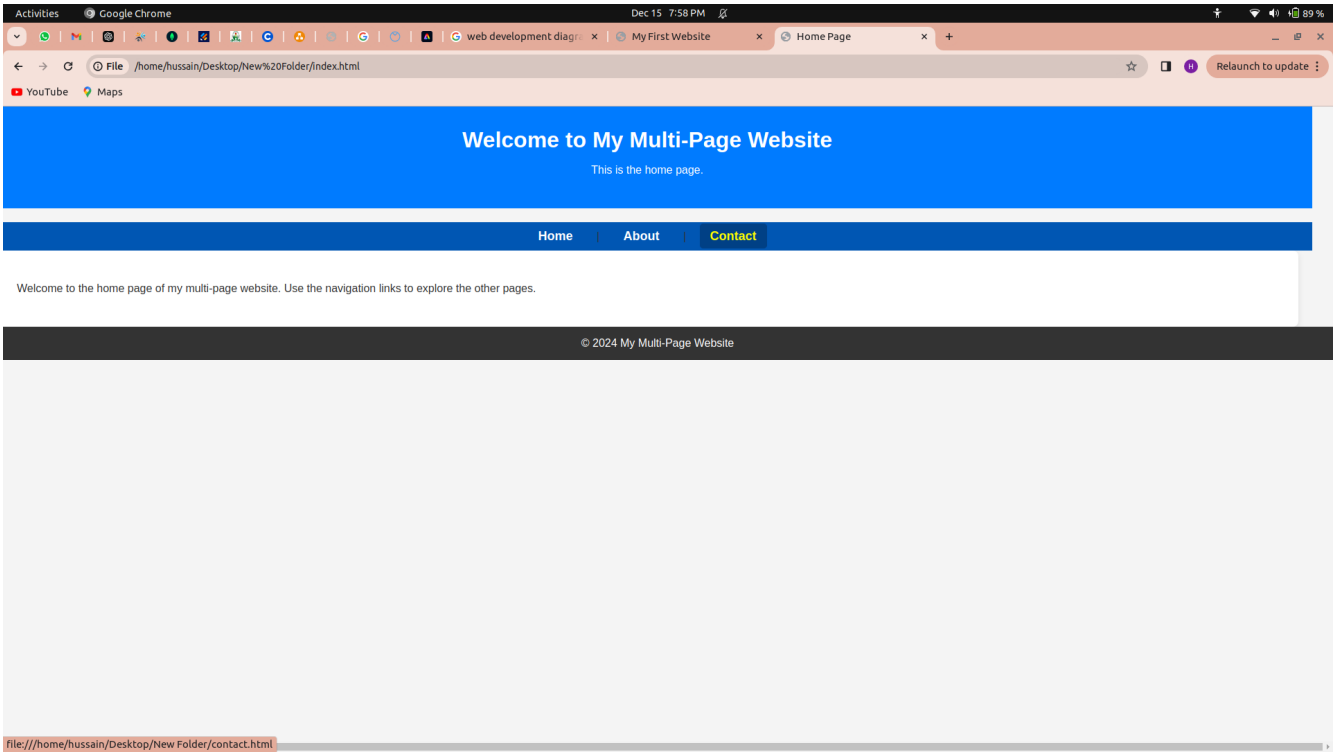
```
/* Navigation Styles */
nav {
  background-color: #333;
  padding: 10px;
  text-align: center;
}

nav a {
  color: white;
  text-decoration: none;
  margin: 0 10px;
  font-size: 18px;
  font-weight: bold;
}

nav a:hover {
  color: yellow;
}
```

How It Works

1. Create multiple HTML files (e.g., `index.html`, `about.html`, `contact.html`).
2. Use `<a>` tags with `href` attributes to link these pages together.
3. Add a navigation bar at the top of each page for easy navigation.
4. Apply CSS styles to make the navigation bar look clean and consistent.



Sample of the contact page