ISDM (INDEPENDENT SKILL DEVELOPMENT MISSION

☑ INTRODUCTION TO WEB DEVELOPMENT – HOW WEBSITES WORK

CHAPTER 1: WHAT IS WEB DEVELOPMENT?

1.1 Understanding Web Development

Web development refers to the process of **creating, designing, and maintaining websites**. It involves:

✓ Writing code to build web pages.

✓ Making websites interactive and user-friendly.

✓ Storing and managing data on servers.

Web development is divided into three main areas:

- **Front-End Development** – The part of the website users see and interact with.
- **Back-End Development** – The part that runs behind the scenes, handling logic and databases.
- **Full-Stack Development** – A combination of both front-end and back-end development.

📌 **Example:** A website like **Amazon** has:

✓ A **front-end** (buttons, images, product listings).

✓ A **back-end** (manages inventory, processes payments).
✓ A **database** (stores user details and purchase history).

CHAPTER 2: HOW WEBSITES WORK

2.1 The Basic Components of a Website

A website consists of several components:

1. **Client (Browser)** – The application users use to access a website (e.g., Chrome, Firefox).
2. **Server** – The computer that stores website files and processes requests.
3. **Database** – Stores website content, user data, and other information.

When you visit a website, this is what happens:

- You type a **URL** (e.g., www.google.com) in your browser.
- The browser sends a **request** to the website's **server**.
- The server **processes the request** and fetches the required data.
- The website content is **displayed on your browser**.

📌 **Example:** When you search for a product on **Amazon**, the website fetches **data from the database** and shows the results on the page.

2.2 What is a Domain Name and Hosting?

✓ **Domain Name:** The address of a website (e.g., www.example.com).
✓ **Hosting:** The service that stores and serves website files to users.

📌 **Example:**

- **Google.com** is a domain name.
- Google's data is hosted on multiple servers worldwide.

To make a website live, you need:

1. **A Domain Name** (like yourwebsite.com).
2. **Web Hosting** (like Bluehost, AWS, or GoDaddy).

CHAPTER 3: FRONT-END VS. BACK-END DEVELOPMENT

3.1 Front-End Development

The **front-end** is the part of the website users **see and interact with**.

🔹 **Technologies Used:**
✓ **HTML (HyperText Markup Language)** – Defines the structure of web pages.
✓ **CSS (Cascading Style Sheets)** – Styles and formats web pages.
✓ **JavaScript** – Adds interactivity and dynamic features.

📌 **Example:** A **login form** that allows users to enter their email and password is built using **HTML, CSS, and JavaScript**.

3.2 Back-End Development

The **back-end** is the part of a website that users **do not see**. It processes requests, manages data, and sends responses to the front-end.

🔹 **Technologies Used:**

✓ **Programming Languages:** Python, JavaScript (Node.js), PHP, Java, C#.

✓ **Databases:** MySQL, MongoDB, PostgreSQL.

✓ **Servers:** Apache, Nginx.

📌 **Example:** When a user submits a **login form**, the back-end **checks the database** for matching credentials and responds accordingly.

3.3 Full-Stack Development

A **Full-Stack Developer** works on both **front-end** and **back-end** development.

📌 **Example:** A full-stack developer can build a **complete e-commerce website**, handling:

✓ **Front-end** (Product listings, shopping cart).

✓ **Back-end** (Order processing, payment integration).

✓ **Database** (Customer and order details).

CHAPTER 4: HOW WEBPAGES ARE DISPLAYED IN BROWSERS

When you open a website, your browser processes different files:

1. **HTML** – Defines headings, text, and buttons.
2. **CSS** – Styles the webpage (colors, fonts, layout).

3. **JavaScript** – Adds interactivity (animations, pop-ups).

📌 **Example:**

A webpage that displays "Hello, World!" in red text using HTML and CSS.

HTML File (index.html)

```
<!DOCTYPE html>
<html>
<head>
  <title>My First Webpage</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <h1>Hello, World!</h1>
</body>
</html>
```

CSS File (style.css)

```
h1 {
  color: red;
  text-align: center;
}
```

✓ When you open this file in a browser, you'll see "Hello, World!" in red text.

CHAPTER 5: POPULAR WEB DEVELOPMENT TOOLS

Web developers use different tools to build, test, and optimize websites.

✓ **Text Editors & IDEs:** VS Code, Sublime Text, Atom.
✓ **Frameworks:** React.js, Angular, Django, Flask.
✓ **Version Control:** Git, GitHub.
✓ **Web Browsers:** Chrome, Firefox (for testing).

📌 **Example: GitHub** is used for **collaborating** and storing web development projects.

CHAPTER 6: EXERCISE

6.1 Multiple Choice Questions

1. What does HTML stand for?
    - (a) Hyper Text Markup Language
    - (b) High Tech Modern Language
    - (c) Hyper Transfer Management Language
    - (d) High-Level Text Management
2. Which language is used to style a webpage?
    - (a) JavaScript
    - (b) HTML
    - (c) CSS

- (d) Python

3. What is the role of a web server?
   - (a) Stores web pages and serves them to users
   - (b) Writes HTML code
   - (c) Designs graphics for websites
   - (d) Runs JavaScript in the browser

6.2 Practical Tasks

📌 **Task 1: Create a Simple Webpage Using HTML**

Write a **basic webpage** that displays a **welcome message**.

```
<!DOCTYPE html>
<html>
<head>
  <title>Welcome Page</title>
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This is my first webpage.</p>
</body>
</html>
```

📌 Task 2: Modify the Webpage with CSS

- Change the **background color** of the webpage.
- Make the text **center-aligned**.

CHAPTER 7: SUMMARY

✅ **Web development** is the process of building and maintaining websites.
✅ **Front-end development** handles the **visual part** of the website using **HTML, CSS, JavaScript**.
✅ **Back-end development** handles the **logic and database** using **Python, PHP, Node.js, and SQL**.
✅ **Web servers** store and serve web pages, while **domain names** provide an address for users to access websites.
✅ **Browsers interpret HTML, CSS, and JavaScript** to display websites correctly.

☑ BASICS OF HTML – CREATING WEB PAGES

CHAPTER 1: INTRODUCTION TO HTML

1.1 What is HTML?

HTML (**HyperText Markup Language**) is the standard language used to create **web pages**. It defines the **structure** of a webpage using **elements and tags**.

- HTML is **not a programming language**; it is a **markup language** used to format content.
- Web browsers interpret **HTML code** to display websites.
- HTML works alongside **CSS (for styling) and JavaScript (for interactivity).**

📌 **Example of a Basic HTML Page:**

```
<!DOCTYPE html>
<html>
<head>
  <title>My First Webpage</title>
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This is a simple webpage created using HTML.</p>
```

```
</body>

</html>
```

✓ **Output in Browser:**
Welcome to My Website
This is a simple webpage created using HTML.

CHAPTER 2: BASIC STRUCTURE OF AN HTML PAGE

Every HTML document has a standard structure:

Tag	Description
<!DOCTYPE html>	Declares the document as an HTML5 page.
<html>	The root tag that contains all HTML content.
<head>	Contains meta-information, title, and links to styles/scripts.
<title>	Defines the title of the webpage (shown in the browser tab).
<body>	Contains all the visible content of the webpage.

📌 **Example of an HTML Page Structure:**

```
<!DOCTYPE html>

<html>

<head>

  <title>My Web Page</title>
```

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

CHAPTER 3: COMMON HTML TAGS

3.1 Headings (<h1> to <h6>)

HTML provides **six levels of headings**, where <h1> is the largest and <h6> is the smallest.

📌 **Example:**

```
<h1>Main Heading</h1>
<h2>Subheading</h2>
<h3>Section Title</h3>
<h4>Small Heading</h4>
<h5>Smaller Heading</h5>
<h6>Smallest Heading</h6>
```

3.2 Paragraphs and Line Breaks (<p>,
)

- The <p> tag defines a **paragraph** of text.

- The
 tag is used for **line breaks** (without starting a new paragraph).

📌 **Example:**

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

<p>This is another paragraph with a line break.<br>See how this appears.</p>
```

3.3 Formatting Text (<b>, <i>, <u>, ,)

- <b> – Makes text **bold**.
- <i> – Makes text *italic*.
- <u> – Underlines text.
- – Indicates **important** text (bold).
- – Indicates *emphasized* text (italic).

📌 **Example:**

```
<p>This is <b>bold</b> text.</p>

<p>This is <i>italic</i> text.</p>

<p>This is <u>underlined</u> text.</p>

<p>This is <strong>important</strong> text.</p>

<p>This is <em>emphasized</em> text.</p>
```

CHAPTER 4: LISTS IN HTML

4.1 Ordered List (<ol> – Numbered List)

📌 **Example:**

```
<ol>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ol>
```

✓ **Output:**

1. HTML
2. CSS
3. JavaScript

4.2 Unordered List (<ul> – Bulleted List)

📌 **Example:**

```
<ul>
  <li>Apple</li>
  <li>Banana</li>
  <li>Orange</li>
</ul>
```

✓ **Output:**

- Apple
- Banana

- Orange

CHAPTER 5: LINKS AND IMAGES

5.1 Creating Hyperlinks (<a> Tag)

The <a> tag is used to create **clickable links**.

📌 **Example:**

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

✓ **Output:**
Visit Google (Opens in a new tab)

5.2 Adding Images (<img> Tag)

The <img> tag is used to **display images** on a webpage.

📌 **Example:**

```
<img src="image.jpg" alt="A beautiful scenery" width="300">
```

✓ **Output:** Displays an image named "image.jpg" with width **300px**.

Attribute	Description
src	Specifies the image URL or file path.
alt	Alternative text (for accessibility and SEO).
width & height	Sets the image size.

CHAPTER 6: CREATING FORMS IN HTML

Forms allow users to **enter data** (e.g., login, search, contact forms).

📌 **Example: A Simple Contact Form**

```
<form>
  <label for="name">Name:</label>
  <input type="text" id="name" name="name"><br><br>

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

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

✓ **Output:** A simple form with name, email, and submit button.

CHAPTER 7: CREATING A BASIC WEB PAGE PROJECT

Now, let's create a **complete webpage** using what we have learned.

📌 **Full Code:**

```
<!DOCTYPE html>
<html>
<head>
  <title>My First Web Page</title>
```

```
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This is a simple webpage created using HTML.</p>

  <h2>About Me</h2>
  <p>My name is John, and I love coding!</p>

  <h2>My Favorite Websites</h2>
  <ul>
    <li><a href="https://www.google.com"
target="_blank">Google</a></li>
    <li><a href="https://www.wikipedia.org"
target="_blank">Wikipedia</a></li>
  </ul>

  <h2>My Picture</h2>
  <img src="myphoto.jpg" alt="John's Photo" width="200">

  <h2>Contact Me</h2>
  <form>
    <label for="name">Name:</label>
```

```
    <input type="text" id="name" name="name"><br><br>


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


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

CHAPTER 8: EXERCISE

8.1 Multiple Choice Questions

1. What does HTML stand for?
    - (a) Hyper Transfer Markup Language
    - (b) HyperText Markup Language
    - (c) High Tech Marking Language
    - (d) Hyper Tool Machine Learning
2. Which tag is used for inserting images?
    - (a) <image>
    - (b) <pic>
    - (c) <img>

- (d) <photo>

3. What is the correct way to create a link in HTML?
   - (a) <link>www.example.com</link>
   - (b) Click Here
   - (c) <hyper>Click Here</hyper>
   - (d) <url>Click Here</url>

📌 SUMMARY

✅ **HTML** is used to **structure webpages**.
✅ It includes **headings, paragraphs, links, images, lists, and forms**.
✅ Websites are built using **HTML, CSS, and JavaScript**.
✅ You can now create **your first simple webpage**! 🚀

☑ Adding Headings, Paragraphs, Links, and Images in HTML

Chapter 1: Introduction to HTML Elements

HTML (**HyperText Markup Language**) is the standard language used to create webpages. It consists of **elements** that structure content, such as:

✓ **Headings** for titles.

✓ **Paragraphs** for text.

✓ **Links** for navigation.

✓ **Images** to display pictures.

These elements help organize content and improve webpage readability.

Chapter 2: Adding Headings in HTML

2.1 What are Headings?

Headings in HTML define the **title or subtitles** of a webpage.

- There are **six levels of headings** (<h1> to <h6>).
- <h1> is the **largest** heading, and <h6> is the **smallest**.

📌 **Example: HTML Headings**

```
<h1>Main Heading</h1>

<h2>Subheading</h2>

<h3>Smaller Subheading</h3>

<h4>Smaller Text</h4>
```

```
<h5>Even Smaller</h5>

<h6>Smallest Heading</h6>
```

✓ **Output:**
Main Heading (H1)
Subheading (H2)
Smaller Subheading (H3)

CHAPTER 3: ADDING PARAGRAPHS IN HTML

3.1 What is a Paragraph Tag?

- The <p> tag is used to **define paragraphs**.
- It automatically adds **spacing** between paragraphs.

📌 **Example: Adding Paragraphs**

```
<p>This is my first paragraph in HTML.</p>

<p>HTML makes structuring web content easy!</p>
```

✓ **Output:**
This is my first paragraph in HTML.
HTML makes structuring web content easy!

CHAPTER 4: ADDING LINKS IN HTML

4.1 What are Links?

Links (**Hyperlinks**) allow users to **navigate between webpages**.

- Links are created using the <a> tag (**anchor tag**).
- The href attribute defines the **URL of the webpage**.

📌 **Example: Creating a Link**

Visit Google

✓ **Output:**

Visit Google

4.2 Opening Links in a New Tab

- The target="_blank" attribute makes the link open in a **new tab**.

📌 **Example:**

Open YouTube in a new tab

✓ **Effect:** Clicking the link opens YouTube in **a new tab**.

4.3 Linking to Another Page on the Same Website

If linking to another page **within the same website,** use a **relative path**.

📌 **Example:**

Go to About Page

✓ This assumes **"about.html"** is in the same folder as the current page.

CHAPTER 5: ADDING IMAGES IN HTML

5.1 What is the <img> Tag?

The <img> tag is used to **display images** on a webpage.

- The src attribute specifies the **image source (URL or file location)**.
- The alt attribute provides **alternative text** for accessibility.

📌 **Example: Adding an Image**

<img src="image.jpg" alt="A beautiful scenery" width="500" height="300">

✓ **Effect:** Displays an image of **500px width** and **300px height**.

5.2 Adding an Image from an Online Source

Use the **direct URL** of an image from the internet.

📌 **Example:**

<img src="https://example.com/picture.jpg" alt="Online Image">

5.3 Using an Image as a Link

Wrap the <img> tag inside an <a> tag to make it clickable.

📌 **Example:**

  <img src="logo.png" alt="Wikipedia Logo" width="200">

✓ **Effect:** Clicking the image takes the user to **Wikipedia**.

CHAPTER 6: EXERCISE

6.1 Multiple Choice Questions

1. Which HTML tag is used for the largest heading?
    - (a) <h6>
    - (b) <h1>
    - (c) <p>
    - (d) <head>
2. What does the href attribute in <a> tag define?
    - (a) The link color
    - (b) The destination URL
    - (c) The image size
    - (d) The paragraph style
3. Which attribute is required for the <img> tag?
    - (a) alt
    - (b) href
    - (c) src
    - (d) title

6.2 Practical Tasks

📌 **Task 1: Create an HTML page with headings and paragraphs**

```
<!DOCTYPE html>
<html>
<head>
  <title>My First Webpage</title>
```

```
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This is my first paragraph in HTML.</p>
  <h2>About Me</h2>
  <p>I am learning web development.</p>
</body>
</html>
```

📌 **Task 2: Create a webpage with a link and an image**

```
<!DOCTYPE html>
<html>
<head>
  <title>Links and Images</title>
</head>
<body>
  <h1>Visit Google</h1>
  <a href="https://www.google.com" target="_blank">Click here to go to Google</a>

  <h2>Sample Image</h2>
  <img src="image.jpg" alt="Beautiful Scenery" width="400">
```

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

CHAPTER 7: SUMMARY

✅ **Headings (<h1> to <h6>)** define different levels of text importance.

✅ **Paragraphs (<p>)** organize content into readable sections.

✅ **Links (<a>)** connect webpages using URLs.

✅ **Images (<img>)** display pictures on webpages.

☑ Introduction to CSS – Styling Web Pages

Chapter 1: What is CSS?

1.1 Understanding CSS

CSS (**Cascading Style Sheets**) is a language used to **style HTML elements** and make web pages visually appealing.

- HTML provides the **structure** of a webpage, while **CSS controls its design**.
- CSS is used to modify **colors, fonts, layout, spacing, animations, and more**.
- It allows developers to **create responsive and dynamic designs**.

📌 **Example: Styling an HTML Element with CSS**

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p {
      color: blue;
      font-size: 20px;
    }
  </style>
```

```
</head>
<body>
  <p>This is a styled paragraph.</p>
</body>
</html>
```

✓ **Output:** The paragraph text appears in **blue** with a **font size of 20px**.

CHAPTER 2: WAYS TO APPLY CSS IN HTML

CSS can be added to an HTML page in **three ways**:

Method	Description
Inline CSS	Applied directly inside an HTML tag using the style attribute.
Internal CSS	Placed inside a <style> tag within the <head> section.
External CSS	Written in a separate .css file and linked to the HTML file.

2.1 Inline CSS

Inline CSS is applied directly within an HTML tag using the style attribute.

📌 **Example:**

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

✓ **Output:** The paragraph text appears in **red and bold**.

2.2 Internal CSS

Internal CSS is written inside the <style> tag in the <head> section.

📌 **Example:**

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 {
      color: green;
      text-align: center;
    }
  </style>
</head>
<body>
  <h1>This heading is styled using Internal CSS</h1>
</body>
</html>
```

✓ **Output:** The heading is **green** and **center-aligned**.

2.3 External CSS

External CSS is written in a separate file (e.g., styles.css) and linked to the HTML file using <link>.

📌 **Example:**

styles.css (External CSS file)

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

h1 {
  color: navy;
  font-size: 30px;
}
```

index.html (HTML file linking the CSS)

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <h1>This heading is styled using External CSS</h1>
</body>
</html>
```

✓ **Output:** The heading is **navy blue** and **30px in size**, with a **light gray background**.

CHAPTER 3: CSS SELECTORS

CSS **selectors** are used to select HTML elements and apply styles to them.

3.1 Common CSS Selectors

Selector	Description	Example
*	Universal selector (applies to all elements)	* { color: red; }
element	Targets all elements of a specific type	p { color: blue; }
#id	Targets a specific element by ID	#header { text-align: center; }
.class	Targets elements of a specific class	.button { background: orange; }

📌 **Example:** Styling an element using an **ID and class**.

```
<style>
  #main-title {
     color: purple;
     text-align: center;
  }

  .highlight {
```

```
    background-color: yellow;
  }
</style>

<h1 id="main-title">Welcome to My Website</h1>
<p class="highlight">This paragraph has a highlighted background.</p>
```

✓ **Output:**

- The heading **"Welcome to My Website"** is **purple** and **center-aligned**.
- The paragraph has a **yellow background**.

CHAPTER 4: CSS PROPERTIES FOR STYLING

CSS allows customization of **text, background, spacing, and layout** using various properties.

4.1 Text Styling

Property	Description	Example
color	Changes text color	color: blue;
font-size	Sets text size	font-size: 20px;
font-family	Changes text font	font-family: Arial, sans-serif;
text-align	Aligns text	text-align: center;

text-decoration	Adds underline, overline, etc.	text-decoration: underline;

📌 **Example:**

```
p {
  color: darkblue;
  font-size: 18px;
  font-family: "Arial", sans-serif;
  text-align: justify;
}
```

4.2 Background Styling

Property	Description	Example
background-color	Sets background color	background-color: yellow;
background-image	Sets a background image	background-image: url('image.jpg');
background-size	Defines background size	background-size: cover;

📌 **Example:**

```
body {
  background-color: lightblue;
  background-image: url("background.jpg");
  background-size: cover;
```

}

4.3 Box Model (Padding, Margin, Border)

CSS **Box Model** consists of:

✓ **Margin** – Space outside the element.

✓ **Border** – A boundary around the element.

✓ **Padding** – Space inside the element, around the content.

✓ **Width & Height** – Controls element size.

📌 **Example:**

```
div {
  width: 300px;
  height: 200px;
  margin: 20px;
  padding: 10px;
  border: 2px solid black;
}
```

✓ **Effect:** Creates a **300x200px box with spacing and a border**.

CHAPTER 5: CREATING A BASIC WEB PAGE WITH CSS

Now, let's apply what we have learned to **design a simple webpage** using CSS.

📌 **styles.css (External CSS File)**

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

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

h1 {
    color: navy;
    text-align: center;
}

p {
    color: darkslategray;
    font-size: 18px;
    text-align: justify;
}

button {
    background-color: orange;
    color: white;
    padding: 10px 20px;
    border: none;
    cursor: pointer;
}
```

```
button:hover {
  background-color: darkorange;
}
```

📌 index.html (HTML File)

```
<!DOCTYPE html>
<html>
<head>
  <title>Styled Web Page</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <h1>Welcome to My Styled Web Page</h1>
  <p>This is an example of a webpage designed with CSS.</p>
  <button>Click Me</button>
</body>
</html>
```


✓ Final Output:

A beautifully styled webpage with:

- **A navy heading**
- **Dark gray paragraph text**
- **A stylish orange button**

CHAPTER 6: EXERCISE

6.1 Multiple Choice Questions

1. What does CSS stand for?
    - (a) Computer Style Sheets
    - (b) Cascading Style Sheets
    - (c) Creative Styling System
2. Which selector targets a class?
    - (a) #classname
    - (b) .classname
    - (c) classname
3. What property changes the background color?
    - (a) color
    - (b) bg-color
    - (c) background-color

📌 SUMMARY

✅ **CSS styles HTML elements** using **colors, fonts, backgrounds, and layouts**.
✅ **Three types of CSS**: Inline, Internal, and External.
✅ **Selectors** allow targeting elements by **ID, class, or type**.
✅ **The Box Model** helps control spacing, borders, and padding.

☑ Creating a Personal Portfolio Website

Chapter 1: What is a Personal Portfolio Website?

A **Personal Portfolio Website** is a **digital resume** that showcases your **skills, projects, experience, and achievements**. It helps you:

✔ **Showcase your work** to potential employers or clients.
✔ **Demonstrate your skills** in web development, design, or any other field.
✔ **Create an online presence** that enhances your professional brand.

📌 **Example:** A web developer's portfolio may include **projects, certifications, contact information, and a blog**.

Chapter 2: Planning Your Portfolio Website

2.1 Define Your Objectives

Before building your portfolio, decide:

- **Who is your target audience?** (Employers, clients, recruiters)
- **What content should you include?** (Projects, skills, experience)
- **What is your goal?** (Get a job, freelance work, build a brand)

2.2 Essential Sections of a Portfolio Website

A good portfolio website includes:

1. **Home (Introduction)** – A short summary about you.
2. **About Me** – Your skills, experience, and background.
3. **Projects** – A showcase of your best work.
4. **Skills** – List of technologies and tools you know.
5. **Contact** – Ways to reach you (email, LinkedIn, GitHub, etc.).

📌 **Example Structure:**

- Home
- About Me
- Projects
- Skills
- Contact

CHAPTER 3: SETTING UP YOUR PORTFOLIO WEBSITE

3.1 Creating the Basic HTML Structure

Every website needs an **HTML file**.

📌 **Basic HTML File (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 Portfolio</title>
    <link rel="stylesheet" href="style.css">  <!-- Linking CSS file -->
</head>
<body>
    <header>
        <h1>Welcome to My Portfolio</h1>
        <nav>
            <ul>
                <li><a href="#about">About Me</a></li>
                <li><a href="#projects">Projects</a></li>
                <li><a href="#skills">Skills</a></li>
                <li><a href="#contact">Contact</a></li>
            </ul>
        </nav>
    </header>
</body>
</html>
```


✔ **Explanation:**

- <header> contains the **website title and navigation menu**.

- <nav> helps users navigate different sections of the site.
- The <link> tag **connects the CSS file** for styling.

CHAPTER 4: ADDING CONTENT TO YOUR PORTFOLIO WEBSITE

4.1 About Me Section

This section describes who you are, what you do, and your experience.

📌 **Example About Me Section (HTML)**

```
<section id="about">
  <h2>About Me</h2>
  <p>Hello! I'm a web developer passionate about creating beautiful and functional websites.</p>
</section>
```

4.2 Projects Section

Showcase your best work with links to GitHub or live projects.

📌 **Example Projects Section (HTML)**

```
<section id="projects">
  <h2>My Projects</h2>
  <ul>
    <li><a href="https://github.com/myproject" target="_blank">Project 1</a></li>
```

```
    <li><a href="https://mywebsite.com/project2"
target="_blank">Project 2</a></li>
  </ul>
</section>
```

✔ **Each project should have:**

- **Project name**
- **Link to GitHub or live demo**
- **Brief description**

4.3 Skills Section

List your technical and soft skills.

📌 **Example Skills Section (HTML)**

```
<section id="skills">
  <h2>Skills</h2>
  <ul>
    <li>HTML, CSS, JavaScript</li>
    <li>Python, Django</li>
    <li>React, Node.js</li>
    <li>Database Management (SQL, MongoDB)</li>
  </ul>
</section>
```

4.4 Contact Section

Allow visitors to contact you via email, LinkedIn, or GitHub.

📌 **Example Contact Section (HTML)**

```
<section id="contact">
    <h2>Contact Me</h2>
    <p>Email: <a href="mailto:youremail@example.com">youremail@example.com</a></p>
    <p>GitHub: <a href="https://github.com/yourprofile" target="_blank">github.com/yourprofile</a></p>
    <p>LinkedIn: <a href="https://linkedin.com/in/yourprofile" target="_blank">linkedin.com/in/yourprofile</a></p>
</section>
```

✔ **Pro Tip:** Add a **contact form** for visitors to send messages.

CHAPTER 5: STYLING YOUR PORTFOLIO WITH CSS

5.1 Creating the CSS File

Use CSS to **style the webpage and make it visually appealing**.

📌 **Example CSS File (style.css)**

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

```
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

header {
    background-color: #333;
    color: white;
    padding: 15px;
    text-align: center;
}

h2 {
    color: #444;
}

ul {
    list-style-type: none;
}

a {
```

```
    text-decoration: none;
    color: blue;
}
```

✔ **Explanation:**

- background-color: #f4f4f4; – Sets a **light gray background**.
- header { background-color: #333; } – Makes the header **dark with white text**.
- a { text-decoration: none; color: blue; } – Styles **links** to be blue.

CHAPTER 6: MAKING YOUR PORTFOLIO RESPONSIVE

To make the website **mobile-friendly**, use **CSS media queries**.

📌 **Example Responsive CSS:**

```
@media screen and (max-width: 600px) {
    body {
        font-size: 14px;
    }
    header {
        padding: 10px;
    }
}
```

✔ This ensures the website **looks good on mobile devices**.

CHAPTER 7: HOSTING YOUR PORTFOLIO WEBSITE

After building your website, you need to **host it online**.

7.1 Free Hosting Options

- ✔ **GitHub Pages** – Best for developers.
- ✔ **Netlify** – Easy to deploy.
- ✔ **Vercel** – Great for Next.js projects.

CHAPTER 8: EXERCISE

8.1 Multiple Choice Questions

1. What tag is used to create a navigation menu?
    - (a) <nav>
    - (b) <menu>
    - (c) <header>
    - (d) <footer>
2. Which CSS property is used to change background color?
    - (a) color
    - (b) background-color
    - (c) font-color
    - (d) bgcolor
3. What attribute makes a link open in a new tab?
    - (a) target="_blank"

- (b) open="new"
- (c) window="new"
- (d) tab="new"

8.2 Practical Tasks

📌 **Task 1: Create an HTML portfolio with an About Me and Projects section.**

📌 **Task 2: Add CSS to style your portfolio with a dark theme.**

📌 **Task 3: Deploy your portfolio using GitHub Pages or Netlify.**

CHAPTER 9: SUMMARY

✅ A **portfolio website** is a personal webpage showcasing skills and projects.
✅ HTML is used to create the **structure**, CSS for **styling**, and JavaScript for **interactivity**.
✅ A **navigation bar** helps users move between sections.
✅ Hosting the portfolio on **GitHub Pages** or **Netlify** makes it accessible online.

ASSIGNMENT 1:

DESIGN A BASIC PERSONAL PORTFOLIO WEBPAGE WITH HTML & CSS.

💡 Assignment Solution 1: Design a Basic Personal Portfolio Webpage with HTML & CSS

🎯 Objective:

✓ Learn how to create a **simple and clean** portfolio webpage.
✓ Use **HTML** to structure the webpage.
✓ Use **CSS** to style and enhance the design.
✓ Make the webpage **responsive and visually appealing**.

🛠 Step-by-Step Guide

Step 1: Set Up Your Project Folder

1. Create a new folder on your computer, e.g., **"Portfolio"**.
2. Inside the folder, create two files:
    - **index.html** → The main webpage file.
    - **style.css** → The stylesheet for styling.

Step 2: Create the Basic HTML Structure

The **index.html** file will contain the structure of the webpage.

📌 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 Portfolio</title>
  <link rel="stylesheet" href="style.css">  <!-- Linking CSS file -->
</head>
<body>

  <!-- Header Section -->
  <header>
    <h1>My Portfolio</h1>
    <nav>
      <ul>
        <li><a href="#about">About</a></li>
        <li><a href="#projects">Projects</a></li>
        <li><a href="#contact">Contact</a></li>
      </ul>
    </nav>
  </header>

  <!-- About Section -->
  <section id="about">
```

```
    <h2>About Me</h2>
    <p>Hello! I'm a web developer passionate about creating beautiful and functional websites.</p>
  </section>

  <!-- Projects Section -->
  <section id="projects">
    <h2>My Projects</h2>
    <ul>
      <li><a href="#">Project 1</a></li>
      <li><a href="#">Project 2</a></li>
    </ul>
  </section>

  <!-- Contact Section -->
  <section id="contact">
    <h2>Contact Me</h2>
    <p>Email: <a href="mailto:youremail@example.com">youremail@example.com</a></p>
  </section>

</body>
```

```
</html>
```

✓ **Explanation:**

- <header> contains the **portfolio title** and **navigation menu**.
- <section> is used to create different parts of the portfolio: **About, Projects, and Contact**.
- <nav> contains **navigation links** to different sections.

Step 3: Style the Portfolio with CSS

Now, let's create the **style.css** file to add styling.

📌 **Code (style.css):**

```
/* General Styling */
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f4f4f4;
}

/* Header Section */
header {
  background-color: #333;
  color: white;
```

```
    text-align: center;
    padding: 20px;
}

nav ul {
    list-style: none;
    padding: 0;
}

nav ul li {
    display: inline;
    margin: 0 15px;
}

nav ul li a {
    color: white;
    text-decoration: none;
}

/* Sections */
section {
    padding: 40px;
```

```
  text-align: center;
}

h2 {
  color: #444;
}

/* Links */
a {
  color: blue;
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}
```

✓ **Explanation:**

- The body style sets the **background color and font style**.
- The header section has a **dark background** and white text.
- The nav styles create a **horizontal navigation menu**.
- Each section is **center-aligned** with proper spacing.

Step 4: Add a Profile Picture (Optional)

You can add an image to the **About Me** section.

📌 **Modify index.html (inside About Section):**

```
<section id="about">
  <h2>About Me</h2>
  <img src="profile.jpg" alt="My Profile Picture" width="150">
  <p>Hello! I'm a web developer passionate about creating beautiful and functional websites.</p>
</section>
```

✓ Place an image named **profile.jpg** in the **Portfolio** folder.

Step 5: Making the Portfolio Responsive

To make the website **mobile-friendly**, use **CSS media queries**.

📌 **Update style.css:**

```
@media screen and (max-width: 600px) {
  body {
    font-size: 14px;
  }

  nav ul li {
    display: block;
    margin: 10px 0;
```

```
  }
}
```

✓ **Effect:** The menu becomes **stacked** on smaller screens.

Step 6: Test Your Portfolio Website

1. **Save all files** in the Portfolio folder.
2. Open **index.html** in a web browser.
3. Click the **navigation links** to test the layout.

Step 7: Host Your Portfolio Online

To make your portfolio accessible online, you can **host it for free** using **GitHub Pages** or **Netlify**.

Option 1: Deploy Using GitHub Pages

1. **Create a GitHub repository** for your portfolio.
2. Upload your **HTML & CSS files**.
3. Go to **Settings** → **Pages** and enable **GitHub Pages**.
4. Your portfolio will be live at https://yourusername.github.io/portfolio/.

Option 2: Deploy Using Netlify

1. Go to Netlify and log in.
2. Click **New Site from Git** and select your GitHub repository.
3. Deploy your site and get a **free custom URL**.

Step 8: Enhancing Your Portfolio

To make your portfolio more interactive, try:

✅ **Adding a "Download Resume" button.**

✅ **Embedding a contact form.**

✅ **Adding project descriptions with images.**

✅ **Using animations with CSS.**

📌 **Example Resume Download Button:**

```
<a href="resume.pdf" download>Download My Resume</a>
```

📌 **Example Contact Form:**

```
<form>
  <input type="text" placeholder="Your Name" required><br><br>
  <input type="email" placeholder="Your Email" required><br><br>
  <textarea placeholder="Your Message" required></textarea><br><br>
  <button type="submit">Send Message</button>
</form>
```

🎯 Summary

✅ **Step 1:** Set up project folder with index.html and style.css.

✅ **Step 2:** Structure the webpage with **HTML**.

✅ **Step 3:** Style the website with **CSS**.

✅ **Step 4:** Add **profile pictures and images**.

✅ **Step 5:** Make the website **responsive**.

✅ **Step 6:** Test the website in a browser.

✅ **Step 7: Deploy** it online using **GitHub Pages** or **Netlify**.

ASSIGNMENT 2:

CREATE A WEBPAGE WITH AN INTERACTIVE FORM.

💡 Assignment Solution 2: Create a Webpage with an Interactive Form

---

🎯 Objective:

✓ Create a **simple webpage** with a form using **HTML**.
✓ Style the form using **CSS**.
✓ Use **JavaScript** to validate user input and provide **real-time feedback**.

🛠 Step-by-Step Guide

Step 1: Set Up Your Project Folder

1. Create a new folder, e.g., **"InteractiveForm"**.
2. Inside the folder, create three files:
    - **index.html** → The main webpage file.
    - **style.css** → The stylesheet for styling.
    - **script.js** → JavaScript for form validation.

Step 2: Create the Basic HTML Structure

📌 **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>Interactive Form</title>
  <link rel="stylesheet" href="style.css">  <!-- Linking CSS file -->
</head>
<body>

  <h2>Contact Us</h2>
  <form id="contactForm">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required>
    <small id="nameError"></small>

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

    <label for="message">Message:</label>
    <textarea id="message" name="message" required></textarea>
    <small id="messageError"></small>

    <button type="submit">Submit</button>
```

```
  </form>


  <script src="script.js"></script>  <!-- Linking JavaScript file -->
</body>
</html>
```

✓ **Explanation:**

- <form> creates a **contact form**.
- <input> fields collect **name, email, and message**.
- <small> elements will **display validation errors** if the user enters incorrect data.
- The form is connected to a **CSS file (style.css)** and a **JavaScript file (script.js)**.

Step 3: Style the Form with CSS

📌 **Code (style.css):**

```
/* General Styling */
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background-color: #f4f4f4;
  margin: 0;
  padding: 20px;
}
```

```
/* Form Styling */
form {
  background: white;
  max-width: 400px;
  margin: 20px auto;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
}

input, textarea {
  width: 100%;
  padding: 8px;
  margin: 8px 0;
  border: 1px solid #ccc;
  border-radius: 5px;
}

/* Error Messages */
small {
  color: red;
```

```
    display: block;
}

/* Submit Button */
button {
    background: blue;
    color: white;
    border: none;
    padding: 10px;
    cursor: pointer;
    width: 100%;
}

button:hover {
    background: darkblue;
}
```

✓ **Explanation:**

- The form is styled with a **white background, box shadow, and centered layout**.
- The input and textarea fields have a **border and padding** for a clean look.
- The **submit button** changes color on hover.
- **Error messages** are styled in red.

Step 4: Add JavaScript for Form Validation

📌 **Code (script.js):**

```
document.getElementById("contactForm").addEventListener("submit", function(event) {
  event.preventDefault(); // Prevents form from submitting

  let isValid = true;

  // Name Validation
  let name = document.getElementById("name").value;
  let nameError = document.getElementById("nameError");
  if (name.length < 3) {
    nameError.textContent = "Name must be at least 3 characters.";
    isValid = false;
  } else {
    nameError.textContent = "";
  }

  // Email Validation
  let email = document.getElementById("email").value;
  let emailError = document.getElementById("emailError");
```

```
  let emailPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;  // Basic email format check
  if (!email.match(emailPattern)) {
    emailError.textContent = "Enter a valid email.";
    isValid = false;
  } else {
    emailError.textContent = "";
  }

  // Message Validation
  let message = document.getElementById("message").value;
  let messageError = document.getElementById("messageError");
  if (message.length < 10) {
    messageError.textContent = "Message must be at least 10 characters.";
    isValid = false;
  } else {
    messageError.textContent = "";
  }

  // If all fields are valid, show success alert
  if (isValid) {
    alert("Form submitted successfully!");
```

```
    document.getElementById("contactForm").reset(); // Clears the form
  }
});
```

✓ **Explanation:**

- Prevents form submission with event.preventDefault().
- Checks if the **name is at least 3 characters long**.
- Validates if the **email format is correct**.
- Ensures the **message has at least 10 characters**.
- Shows an **alert** and **resets the form** if all fields are valid.

Step 5: Test Your Webpage

1. **Save all files** (index.html, style.css, and script.js) in the **InteractiveForm** folder.
2. Open **index.html** in a browser.
3. Try entering:
    - A **short name (less than 3 characters)** → Error appears.
    - An **invalid email (without @ or .com)** → Error appears.
    - A **short message** → Error appears.
    - A **valid form submission** → Success alert appears.

Step 6: Make Your Form Responsive

To make the form mobile-friendly, modify **style.css**:

📌 **Add Responsive CSS:**

```
@media screen and (max-width: 500px) {
  form {
    width: 90%;
  }
}
```

✓ **Effect:** The form adapts to smaller screens.

Step 7: Deploy Your Webpage Online

To make your form accessible online, you can host it for **free** using:

Option 1: GitHub Pages

1. Create a GitHub repository.
2. Upload your files.
3. Go to **Settings** → **Pages** and enable GitHub Pages.
4. Your form will be live at https://yourusername.github.io/InteractiveForm/.

Option 2: Netlify

1. Go to Netlify and log in.
2. Click **New Site from Git** and select your GitHub repository.
3. Deploy your site and get a **free custom URL**.

🎯 Summary

✅ **Step 1:** Set up project folder with **HTML, CSS, and JavaScript** files.

✅ **Step 2:** Create the **form structure** in **HTML**.

✅ **Step 3:** Style the form using **CSS**.

✅ **Step 4:** Add **JavaScript validation** for user input.

✅ **Step 5:** Test the form and handle errors.

✅ **Step 6:** Make the form **responsive**.

✅ **Step 7: Deploy** it online using **GitHub Pages** or **Netlify**.