Introduction to Web Technologies

(IFT 221)

By

Odeh Christopher
Azaka Maduabuchuku
Adeoye Abosede

Faculty of Computing
Department of Computer Science
Dennis Osadebay Universuty, (DOU)
Awai, Asaba, Delta State

Contents

Using Editors for Web Design on Different Devices (Android, iPhone, and PC)
Tools: Termux, Acode, VS Code

This appendix explains how to:

- Open and use your chosen editor
- Write HTML/CSS code
- Save your file
- View it in a browser
  All steps are numbered, based on the device you're using.

1. Using Android

A. Using Acode (Recommended for Beginners)

Step-by-Step to Open and Design in Acode:

1. **Install Acode** from the Play Store (by Foxdebug).
2. Open **Acode** app.
3. Tap the **three-line menu** (top-left) → tap **"Open Folder"**.
4. Navigate to this folder:
   /storage/emulated/0/ (usually your internal storage)
5. Tap the **"+" button** to create a new file.
6. Name the file: index.html
7. Type your HTML code. Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Web Page</title>
</head>
<body>
  <h1>Hello, Acode!</h1>
</body>
</html>
```

8. (i)Tap the **Save icon** (💾) or tap the three dot at top right inside Acode environment (ii) Tap save-as from drop down menu (iii) select folder e.g. document or create your folder (iv) Enter the file name using .html extension e.g. pract.html)
9. Tap the **icon (>) close to three dot at top right** to view in browser.

B. Using Termux (Advanced Users)

Step-by-Step:

1. Install **Termux** from F-Droid (not Play Store).
2. Open Termux and install nano editor:
   pkg update
   pkg install nano
3. Create a file:
   nano index.html
4. Type HTML code using your keyboard.
5. Press CTRL + X, then Y, then Enter to save.
6. To view it:
   - Move file to shared storage:
     termux-setup-storage
     mv index.html /sdcard/
   - Open with mobile browser using file path:
     file:///sdcard/index.html

2. Using iPhone
Using Kodex or Textastic Apps:

Steps for Kodex:

1. Download **Kodex** or **Textastic** from App Store.
2. Open the app → tap **New File**.
3. Type HTML code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello iPhone</title>
</head>
<body>
  <h1>This is Web Design on iPhone</h1>
</body>
</html>
```

4. Save the file as index.html.
5. Open **Files App**, find the file, and tap **Preview in Browser** or **Share → Open in Safari**.

3. Using Laptop/Desktop
Using VS Code (Visual Studio Code)
Steps:

1. Download and install **VS Code** from: https://code.visualstudio.com
2. Open VS Code.
3. Click **File** → **New File**.
4. Type your HTML code.
5. Click **File** → **Save As** → name it index.html.
6. Right-click the file and select **"Open with Browser"**.
    - OR install the VS Code extension: **Live Server**
        1. Go to Extensions (🔍 icon)
        2. Search "Live Server" → Install
        3. Right-click index.html → click **Open with Live Server**

4. How to Save and View in Browser (All Devices)

Device	Save File As	How to View
Android (Acode)	index.html	Tap Eye Icon (Preview), or open via file browser
Android (Termux)	index.html → Move to /sdcard	Open file:///sdcard/index.html in browser
iPhone	index.html in Kodex/Textastic	Preview in app or open via Safari
Laptop	index.html in VS Code	Use "Live Server" or open in any browser

MODULE 1: Introduction to the Internet, World Wide Web (WWW), and Web Development

1.0 Definition of Key Concepts
1.1 What is the Internet?

- The **Internet** is a global network of connected computers and devices that share data using common communication protocols.

- **Function**: It enables communication, file sharing, website access, cloud services, and online collaboration.
- **When/How to Use**: The Internet is used anytime you want to browse, stream, email, or connect with people worldwide. You connect using mobile data, Wi-Fi, or LAN.

1.2 What is the World Wide Web (WWW)

- The **WWW** is a system of interlinked hypertext documents and resources accessible through the Internet using a **web browser**.
- **Function**: To allow access to documents, media, apps, and services over the Internet.
- **When/How to Use**: Type a web address in a browser to view and interact with websites, videos, and online content.

1.3 What is Web Development

- Web Development is the **process of creating websites or web applications**.
- It involves coding using **HTML, CSS, JavaScript**, and optionally server-side languages like Python or PHP.
- **Function**: It powers everything you see and use on the web—from blogs and news sites to e-commerce platforms and online learning portals.
- **When/How to Use**: When you want to build a web page, blog, portfolio, or full web application using tools and programming languages.

2.0 Functions of Internet, WWW, and Web Development

Component	Main Function
Internet	Allows devices to communicate and access services globally
WWW	Delivers formatted documents (web pages) via browsers
Web Development	Creates structured, interactive, and functional websites and web applications

3.0 The WWW as a Platform for:

3.1 Interactive Applications

- Examples: Gmail, Facebook, Zoom, online forms.
- Use **JavaScript** and frameworks to provide real-time interaction.
- Students, businesses, and governments use these apps to automate tasks and improve engagement.

3.2 Content Publishing

- Allows anyone to share content using blogs, YouTube, online journals, and digital books.
- Tools like **WordPress**, **Blogger**, and **Wix** make this easy even without coding.

3.3 Social Services

- Examples: e-Government platforms, e-Health systems, online job portals.
- The web connects citizens to services and governments to citizens, improving access and efficiency.

4.0 Simple Web Page Example

HTML Code Example (with comments)

```
<!DOCTYPE html> <!-- Declares the document is HTML5 -->
<html> <!-- Root element -->
<head> <!-- Contains metadata -->
  <title>My First Web Page</title> <!-- Sets browser tab title -->
</head>
<body> <!-- Visible content of the page -->
  <h1>Welcome to My Website</h1> <!-- Main heading -->
  <p>This is an introductory web page built using HTML.</p> <!-- Paragraph of text -->
</body>
</html> <!-- End of HTML document -->
```

5.0 Numbered Steps to View Your First Web Page

On Android (Using Acode)

1. Open **Acode** app.
2. Tap **Menu (☰)** → Open Folder → Navigate to internal storage.
3. Tap **"+"** to create a file and name it index.html.
4. Type your HTML code.
5. Tap 🖫 **Save**.
6. Tap 👁 **Preview** to view your page in browser.

On Android (Using Termux)

1. Open **Termux** → run: pkg update && pkg install nano
2. Run: nano index.html
3. Type your HTML code and save (Ctrl+X → Y → Enter).
4. Move file: mv index.html /sdcard/

5. Open your browser → Type: file:///sdcard/index.html

On iPhone (Using Kodex or Textastic)

1. Install **Kodex** or **Textastic**.
2. Open the app → New File → Type HTML code.
3. Save as index.html.
4. Open **Files app**, locate file, tap **Preview** or open in **Safari**.

On Laptop/Desktop (Using VS Code)

1. Install **VS Code** from https://code.visualstudio.com.
2. Click **File** → **New File** → Type your HTML code.
3. Save as index.html.
4. Right-click file → Choose **"Open with Browser"** or use **Live Server extension**.

6.0 Homework/Practice Exercises

Exercise 1

Q: Define Internet, WWW, and Web Development.

A:

- **Internet**: A global system of connected devices for sharing info.
- **WWW**: A service on the internet that links and displays content.
- **Web Development**: The process of building websites and apps.

Exercise 2

Q: List 2 examples each of interactive apps, content publishing tools, and social web services.

A:

- **Interactive Apps**: Google Docs, Facebook
- **Publishing Tools**: YouTube, WordPress
- **Social Services**: e-Government, online hospitals

Exercise 3

Q: Write HTML code that displays a heading “My Website” and a paragraph “Learning web is fun!”.

A:

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

```
  <title>My Website</title>
</head>
<body>
  <h1>My Website</h1>
  <p>Learning web is fun!</p>
</body>
</html>
```

Exercise 4

Q: Differentiate between the Internet and the World Wide Web.

A:

- Internet: The physical and technical infrastructure (networks, cables, satellites).
- WWW: A service that runs on top of the Internet using browsers and hyperlinks.

Exercise 5

Q: Name 2 tools used for web development and what they are used for.

A:

- Acode: Mobile editor for writing HTML/CSS
- VS Code: Desktop editor for writing and testing code

Exercise 6

Q: How can WWW be used to improve governance and education?

A:

- Governance: e-Voting, online tax filing, public info portals
- Education: Online learning platforms, university portals, e-libraries

End of Module 1

MODULE 2: HTTP and HTTPS in Web Applications

1.0 Definition of Key Concepts

1.1 What is HTTP?

- **HTTP** stands for **HyperText Transfer Protocol**.
- It is a **communication protocol** used between **web browsers** (clients) and **web servers**.
- It follows a **request-response model**, meaning your browser sends a request, and the server responds.

1.2 What is HTTPS?

- **HTTPS** stands for **HTTP Secure**.

- It is the **secure version of HTTP**, where all data exchanged is **encrypted** using **SSL/TLS** (Secure Sockets Layer / Transport Layer Security).
- Websites using HTTPS show a **lock 🔒 icon** in the browser's address bar.

2.0 Function of HTTP and HTTPS

Protocol	Full Meaning	Function	Use Case
HTTP	HyperText Transfer Protocol	Transfers web content (HTML, images, etc.)	For simple pages (non-sensitive)
HTTPS	HTTP Secure	Same as HTTP but adds encryption for privacy	For secure pages (login, payment, forms)

3.0 When and How to Use

Use HTTP:

- When building or testing simple static pages on your local computer.
- Not recommended for live production websites.

Use HTTPS:

- Always use HTTPS for:
  - Login forms
  - Payment pages
  - Contact forms
  - Any website collecting user information

Browsers today often **block HTTP-only pages** or mark them as "Not Secure."

4.0 How HTTP/HTTPS Work: Step-by-Step

Request-Response Model:

1. **User enters** a web address (URL) in the browser: https://example.com
2. The browser sends an **HTTP or HTTPS request** to the web server.
3. The web server **processes** the request.
4. The server sends back a **response** (usually an HTML page).
5. The browser displays the page to the user.

If the site uses **HTTPS**, the data is encrypted to prevent hackers from seeing it.

5.0 Basic Example: Using a Form with HTTPS

<!DOCTYPE html> <!-- Tells browser it's HTML5 -->

```
<html>
<head>
  <title>Contact Form</title>
</head>
<body>
  <h2>Contact Us</h2>
  <!-- This form uses HTTPS to submit data securely -->
  <form action="https://example.com/submit" method="post">
    <label for="name">Name:</label><br>
    <input type="text" id="name" name="name"><br><br>
    <label for="email">Email:</label><br>
    <input type="email" id="email" name="email"><br><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

Comments on Code:

- action="https://example.com/submit": sends form data to the server securely using HTTPS.
- method="post": indicates that data will not appear in the URL (more secure).

6.0 Practice Exercises

Exercise 1

Q: What is the difference between HTTP and HTTPS? Why is HTTPS important?
A:

- HTTP sends data in plain text.
- HTTPS encrypts data, making it secure.
- HTTPS protects user information during login, form submissions, or payments.

Exercise 2

Q: Identify three types of websites that must use HTTPS.
A:

- Online banking platforms
- E-commerce websites
- School or government login portals

Exercise 3

Q: In the HTML form below, identify where HTTPS is used and explain its purpose.

```
<form action="https://myschool.edu/register" method="post">
  <input type="text" name="student_name">
  <input type="submit" value="Register">
</form>
```

A:

- action="https://myschool.edu/register" uses HTTPS to send student data securely.
- It ensures that hackers cannot read the data in transit.

Exercise 4

Q: Fill in the blanks:

1. HTTP stands for _____
2. HTTPS includes extra _____ for security.
3. A browser shows a _____ symbol when HTTPS is active.

A:

1. HyperText Transfer Protocol
2. Encryption
3. Lock

Exercise 5

Q: What happens if a website collects passwords but uses HTTP instead of HTTPS?

A:

- Hackers or attackers can intercept the password.
- The browser may show a “Not Secure” warning to users.

7.0 Summary

Term	Purpose
HTTP	Sends data between browser and server in plain text
HTTPS	Sends encrypted data; protects privacy and security
Use HTTPS	When collecting sensitive data (login, forms, payments)

End of Module 2

MODULE 3: Web Browsers and Web Servers

1.0 Definitions

1.1 What is a Web Browser

- A **web browser** is a software application used to **access and display websites** on the World Wide Web (WWW).
- Examples: **Google Chrome**, **Mozilla Firefox**, **Safari**, **Microsoft Edge**, **Opera**.
- **Function**: It interprets HTML, CSS, JavaScript, and other web code to display content like text, images, and videos.

1.2 What is a Web Server?

- A **web server** is a **computer program or device** that **stores web pages and files**, and delivers them to users (clients) when requested.
- Examples: **Apache**, **Nginx**, **Microsoft IIS**.
- **Function**: Responds to browser requests using protocols like **HTTP/HTTPS** and returns the correct website content.

2.0 Functions of Web Browsers and Web Servers

Tool	Function
Web Browser	Sends requests, receives and displays websites
Web Server	Stores files and serves them when browsers request them

3.0 When and How to Use Them

Tool	When to Use	How to Use
Web Browser	When you want to view a website	Type a URL and press Enter
Web Server	When hosting a website	Upload files and configure server software (e.g., Apache or Nginx)

4.0 How They Work Together: Step-by-Step Process

Client-Server Communication Process:

1. **User opens browser** and enters a URL (e.g., https://example.com)
2. The browser sends an **HTTP or HTTPS request** to the **web server**
3. Server checks the URL and **responds** with the requested file (e.g., HTML page)
4. The browser receives the response and **displays the web page**

Diagram Representation (Text Format)

[User] → [Browser] → [Internet] → [Web Server]

↓

[Sends Back Web Page]

↓

[User] ← [Browser] ← [Internet] ← [Web Server]

5.0 Hands-On Example: Using a Browser and Local Server

HTML File to be Hosted:

```
<!DOCTYPE html> <!-- Declares HTML5 -->
<html>
<head>
  <title>Test Page</title>
</head>
<body>
  <h1>This page is served by a web server</h1>
  <p>Hello from a simple web server setup!</p>
</body>
</html>
```

Steps to Use a Local Web Server (Desktop/Laptop with VS Code)

PART A: Local Web Server on PC Using VS Code + Live Server

Step 1: Install VS Code

1. Open your browser and go to: https://code.visualstudio.com
2. Click the **Download for Windows** (or Mac/Linux depending on your OS).
3. After downloading, **run the installer** and follow the prompts.
4. After installation, **launch VS Code**.

Step 2: Install Live Server Extension

1. In VS Code, look at the **left sidebar** and click the **Extensions icon** ( ) or press Ctrl + Shift + X.
2. In the **search bar** at the top, type:
   Live Server

3. Look for the result by **Ritwick Dey**.
4. Click the **Install** button.
5. Wait until it shows ✓ **Installed**.

Step 3: Create a New Project Folder and HTML File

1. On your desktop (or any location), **right-click** → New > Folder.
2. Name it: MyWebsite
3. Go back to VS Code.
4. Click: File > Open Folder… and choose the MyWebsite folder → Click **"Select Folder"**.
5. In the left Explorer panel, click the **"New File"** icon.
6. Name it: index.html
7. Paste this HTML code:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Web Page</title>
</head>
<body>
  <h1>Hello, world!</h1>
  <p>Welcome to my web page.</p>
</body>
</html>
```

8. Press Ctrl + S or go to File > Save.

Step 4: Open with Live Server

1. In VS Code's Explorer panel, **right-click** on index.html.
2. Click: 🌐 **"Open with Live Server"**
3. Your default browser will open automatically at:
   http://127.0.0.1:5500/index.html

Step 5: View and Edit the Page

1. Your page will show:
   Hello, world!
   Welcome to my web page.
2. Make any change in the HTML and press **Save** — the browser updates **automatically**.

Example 1: Show a List of Items

Paste this into your index.html:

```
<h2>Shopping List</h2>
<ul>
  <li>Apples</li>
  <li>Bananas</li>
  <li>Milk</li>
</ul>
```

Example 2: Add an Image

Ensure the image (e.g., dog.jpg) is in the same folder. Then add:

```
<h2>My Dog</h2>
<img src="dog.jpg" alt="A cute dog" width="300">
```

PART B: Local Web Server on Android Using Acode + Termux

Step 1: Install the Apps

1.1. Install Acode (Code Editor for Android)

- Go to **Play Store**
- Search: Acode
- Install the app with the icon </>

1.2. Install Termux

- Download Termux via F-Droid — don't use Play Store.
- Visit: https://f-droid.org
- Search Termux, download, and install it.

Step 2: Set Up the HTML File in Acode

1. Open **Acode**.
2. Tap the **folder icon** (🗀) → Tap **"Add Folder"**
3. Create or open a folder like mysite.
4. Tap the **+** button → Choose **New File**
5. Name the file index.html
6. Paste this code:

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

```
</head>
<body>
  <h1>Hello, world!</h1>
  <p>Welcome to my mobile web page.</p>
</body>
</html>
```

7. Tap the **save icon 🖫**.

Step 3: Create a Folder and Move HTML (Optional)
If you created the HTML file in Acode but want to serve it via Termux:

1. Note down where Acode saved your file (likely in /storage/emulated/0/ or Download/acode/).
2. You can copy/move this file into Termux's folder later (see below).

Step 4: Setup and Start Termux HTTP Server
4.1. Open Termux
4.2. Type:

```
mkdir mysite
cd mysite
```

4.3. Create the HTML file in Nano editor:

```
nano index.html
```

4.4. Paste the HTML code:
(Use long press to paste or type manually.)
4.5. Save the file:

- Volume Down + X → Press Y → Press Enter

Step 5: Start Python HTTP Server
{ You will **install Python inside the Termux app** on your **Android phone**.
Where and How to Install Python:
In Termux (Android terminal):

1. **Open the Termux app** on your phone.
2. At the command prompt, type:

```
pkg install python
```

3. Press **Enter**.
4. Wait a few moments — it will download and install Python.
5. When it finishes, you can check it's installed by typing:

```
python --version
```

You should see something like:

```
Python 3.x.x
```

Why Install Python in Termux?

Because we use Python's built-in **HTTP server** to serve your HTML files locally on Android:

python -m http.server 8080

This command **runs the local web server**, making your page accessible at:

http://localhost:8080/index.html

}

5.1. First install Python (if not already):

pkg install python

5.2. Start the server:

python -m http.server 8080

You will see:

Serving HTTP on 0.0.0.0 port 8080 ...

Step 6: View Page in Browser

1. Minimize Termux (don't close it).
2. Open **Chrome or Firefox** on your phone.
3. Go to:

http://localhost:8080/index.html

Your page should appear!

BONUS: Connect Acode + Termux (Serve Acode Files via Termux)

Method 1: Access Acode Files in Termux via Shared Storage

1. In Termux, type this:

termux-setup-storage

2. Give permission when prompted.
3. You can now access files like:

/storage/emulated/0/Download/acode/index.html

4. To serve it:

cd /storage/emulated/0/Download/acode

python -m http.server 8080

5. Now open in browser:

http://localhost:8080/index.html

6.0 Practice Exercises

Exercise 1

Q: Define a web browser and name three examples.
A:

- A **web browser** is a program used to view websites.
- Examples: Google Chrome, Mozilla Firefox, Opera.

Exercise 2
Q: What is the function of a web server?
A:
A web server **stores and delivers website files** when requested by a browser.

Exercise 3
Q: Explain how a browser and server communicate when you visit https://myschool.edu.
A:

1. Browser sends an HTTPS request to myschool.edu
2. Server responds with the appropriate page (e.g., index.html)
3. Browser displays the content to the user.

Exercise 4
Q: Fill in the blanks:

1. Web browsers use _____ or _____ to talk to servers.
2. The server sends back a _____.
3. One common web server software is _____.

A:

1. HTTP, HTTPS
2. Response (e.g., HTML file)
3. Apache

Exercise 5
Q: Write a short HTML page to be served by a server, and explain how to run it using "Live Server" in VS Code.
A:

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello Server</title>
</head>
```

```
<body>
  <h1>Served by Live Server</h1>
</body>
</html>
```

How to Run:

1. Save as index.html
2. Right-click the file in VS Code
3. Choose **"Open with Live Server"**

7.0 Summary

Component	Role
Web Browser	Requests and displays web pages
Web Server	Responds to browser requests by sending back files
Live Server Tool	Helps run a local server for testing HTML/CSS/JS

End of Module 3

How to Add CSS, Images & Link Multiple Pages

We'll cover:

1. How to create folders
2. How to add a CSS file
3. How to add an image
4. How to create a second HTML page and link between pages

We'll do it **step-by-step for both**:

- **PC (VS Code + Live Server)**
- **Android (Termux + Acode)**

1. How to **Place an Image in an HTML Web Page**

Steps:

1. **Choose Your Image**
    - Make sure it is in a web-friendly format like .jpg, .png, or .gif.
2. **Rename the Image File** (optional but helpful)
    - For example: photo.jpg
3. **Place the Image in the Same Folder as Your HTML File**
    - Example structure:

      my-website/

      ├── index.html

└── photo.jpg

4. **Use the <img> Tag in HTML**
    - Example:
      <img src="photo.jpg" alt="My Photo" width="300">
5. **Explanation:**
    - src – path to the image file
    - alt – text shown if image doesn't load
    - width – optional, to resize the image

2. Link Two or Multiple HTML Pages

Steps:

1. **Create Multiple HTML Files**
    - Example: index.html, about.html, contact.html
2. **Use the <a> Tag to Link Pages**
   Example (inside index.html):
   About Us
   Contact
3. **Make Sure All HTML Files Are in the Same Folder**
    - Example structure:
      my-website/
      ├── index.html
      ├── about.html
      └── contact.html
4. **Clicking the Link Will Take You to the Other Page**
    - When the user clicks **"About Us"**, it opens about.html.

A. Displaying an Image

HTML Example:

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

  <h2>This is an Image</h2>
```

```
  <!-- ✅ Display an image -->
  <img src="photo.jpg" alt="A beautiful photo" width="300">

</body>
</html>
```

What to do:

- Save this as index.html
- Put photo.jpg in the same folder.
- Open index.html in your browser to see the image.

B. Linking Multiple Pages with Text and Buttons

1. index.html:

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

  <h1>Welcome to My Website</h1>

  <!-- ✅ Link using text -->
  <a href="about.html">Go to About Page</a><br>

  <!-- ✅ Link using button -->
  <button onclick="window.location.href='contact.html'">Contact Us</button>

  <!-- ✅ Link using image -->
  <br><br>
  <a href="about.html">
    <img src="photo.jpg" alt="Click to go to About" width="200">
  </a>

</body>
</html>
```

2. about.html:

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

```
<head>
  <title>About Page</title>
</head>
<body>

  <h1>About Us</h1>
  <p>This is the About page.</p>
  <a href="index.html">Back to Home</a>

</body>
</html>
```

3. contact.html:

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

  <h1>Contact Information</h1>
  <p>Email: example@example.com</p>
  <a href="index.html">Back to Home</a>

</body>
</html>
```

Folder Structure:

```
my-website/
├── index.html
├── about.html
├── contact.html
└── photo.jpg
```

Let's walk through **step-by-step how to create and test this mini-website** on both **phone** and **PC**. Just follow whichever device you're using.

IF YOU'RE USING A PHONE (Android)

A. Install an HTML Editor App

Choose one of these free apps:

1. **Acode** (Recommended – Android only)

2. **Dcoder**
3. **Spck Editor**
4. **JStudio**

Download from Google Play Store.

B. Create Your Files

Step 1: Open the editor and create a new folder called my-website.

Step 2: Inside that folder, create 3 new files:

- index.html
- about.html
- contact.html

Step 3: Copy and paste the code I gave you earlier into each file:

- Paste the *index.html* code into the index.html file
- Paste the *about.html* code into the about.html file
- Paste the *contact.html* code into the contact.html file

Step 4: Add an image to the folder:

- Download or rename a .jpg image as photo.jpg
- Place it in the **same folder** as the HTML files

Step 5: Open the index.html file in the built-in preview browser.

IF YOU'RE USING A PC (Windows or Mac)

A. Create Your Website Folder

Create a folder on your Desktop called my-website

Open Notepad or any text editor (VS Code, Notepad++, Sublime)

B. Create and Save HTML Files

Create and save these three files:

- index.html
- about.html
- contact.html

Use **Save As... > File type: All Files** and add .html extension

C. Add the Image

4. Place a .jpg image (e.g., photo.jpg) in the **same folder**

D. View Your Site

Double-click index.html

- It will open in your browser.
- You can test the links and image.

PART A: PC (Windows) Using VS Code + Live Server

1. Create Project Folders (Using File Explorer)

1.1. Right-click on your Desktop → Click: New → Folder
1.2. Name it: MyWebsite
1.3. Open the MyWebsite folder
1.4. Inside it, create these subfolders:

- Right-click inside → New Folder → Name it: css
- Right-click again → New Folder → Name it: images

You now have:

```
MyWebsite/
├── index.html
├── about.html
├── css/
│   └── style.css
├── images/
│   └── myimage.jpg
```

2. Open in VS Code

2.1. Launch **VS Code**
2.2. Click File → Open Folder
2.3. Navigate to and select the MyWebsite folder
2.4. Click "Select Folder"

✅ 3. Create and Edit index.html

3.1. In VS Code, click **New File** → Type: index.html → Press Enter
3.2. Paste this HTML code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Home Page</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <h1>Welcome to My Site</h1>
  <p>This is the home page.</p>
  <img src="images/myimage.jpg" alt="My Image" width="300">
  <p><a href="about.html">Go to About Page</a></p>
```

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

3.3. Save the file (Ctrl + S)

4. Create and Edit style.css

4.1. Right-click css folder → Click “New File”

4.2. Name it: style.css

4.3. Paste this CSS:

```
body {
  background-color: #f0f8ff;
  font-family: Arial, sans-serif;
  color: #333;
  text-align: center;
}

h1 {
  color: darkblue;
}
```

4.4. Save the file

5. Add Image

5.1. Copy an image from your computer

5.2. Paste it into the images folder

5.3. Rename it to: myimage.jpg (if not already)

***{* Steps for Adding Images in HTML**

1. Prepare your image

- Make sure your image is in **JPG, PNG, or GIF** format.
  (JPG is common, but HTML supports PNG, GIF, SVG, etc.)
- Rename the file to something simple, like image1.jpg.
  Avoid spaces or special characters in the filename.

2. Place the image in your project folder

- Copy or move the image into the **same folder** where your HTML file (e.g., index.html) is located.
  This helps the browser find it easily.

3. Add the HTML code to display the image

- Use the <img> tag with the src attribute like this:

<img src="image1.jpg" alt="Description of image">
The alt text is important for accessibility and when the image fails to load.

4. Save and run your HTML code

- Save your HTML file.
- Open the file in your browser or run it using **Live Server** or **Python HTTP server**.

Example
If your folder contains:
index.html
image1.jpg
Then in index.html, use:
<img src="image1.jpg" alt="My Image" width="300">
This will display the image at 300px wide.

6. Create and Edit about.html

6.1. Right-click in the root folder → “New File” → Name: about.html
6.2. Paste this HTML:

```
<!DOCTYPE html>
<html>
<head>
  <title>About Page</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <h1>About This Website</h1>
  <p>This is a second page.</p>
  <p><a href="index.html">Back to Home</a></p>
</body>
</html>
```

6.3. Save the file

7. Run with Live Server

7.1. Right-click index.html → Click “Open with Live Server”

7.2. Browser opens at:

http://127.0.0.1:5500/index.html

7.3. You can:

- Click to go to About page
- See image and styles
- Navigate back

PART B: Android Using Termux + Acode

✅ 1. Create Project Folders in Termux

1.1. Open Termux

1.2. Type:

```
cd ~                                # Go to home directory
mkdir -p MyWebsite/css images       # Create folders (css inside MyWebsite)
cd MyWebsite                        # Enter project folder
touch index.html about.html         # Create HTML files
touch css/style.css                 # Then create CSS file inside css/
```

Or put all together:

```
cd ~
mkdir -p MyWebsite/css MyWebsite/images
cd MyWebsite
touch index.html about.html
touch css/style.css
```

This will result in:

```
~/MyWebsite/
├── index.html
├── about.html
├── css/
│   └── style.css
```

├── images/

2. Open and Edit Files in Acode

2.1. Open Acode

2.2. Tap the folder icon → Navigate to:

/data/data/com.termux/files/home/MyWebsite/

2.3. Tap each file to edit:

- index.html → Paste the home page HTML code
- about.html → Paste the about page HTML code
- css/style.css → Paste the CSS code

2.4. Save each file (use top-right icon)

3. Add Image File

3.1. Download an image using your browser

3.2. Open your **File Manager app**

3.3. Move the image to:

/data/data/com.termux/files/home/MyWebsite/images/

3.4. Rename the image to myimage.jpg

4. Start Web Server in Termux

4.1. Return to Termux

4.2. Start the server:

```
cd ~/MyWebsite
python -m http.server 8080
```

4.3. You'll see:

```
Serving HTTP on 0.0.0.0 port 8080 ...
```

5. Open Browser to View

5.1. Open Chrome or Firefox on your phone

5.2. In the address bar, type:

http://localhost:8080/index.html

You should now see your website with:

- Styled content
- Image
- Link to About page

Done!

Now you know how to:

- Structure your site folders
- Add a stylesheet
- Display an image
- Link between pages
- Serve it on both PC and Android

Add JavaScript, Create a Contact Form, and Make It Mobile Responsive

You'll learn how to:

1. Add JavaScript (e.g., button that shows alert)
2. Create a simple **Contact Form** (name, email, message)
3. Make the page look good on mobile (using **media queries**)

We'll cover steps **for both PC and Android**, including:

- How to create files
- What to paste
- How to test it

PART A: FOR ANDROID USERS

(**Tools used: Acode + Termux (optional)**)

STEP 1: Open Acode

1. Tap the **Acode app** to open it.
2. On the top bar, tap the **folder icon** (if visible).
   If you can't find the folder, tap **three dots (⋮)** → "Open folder".

STEP 2: Create a New HTML File

1. Tap the **➕ Plus icon** at the top.
2. Select **"New file"**
3. Name the file:
   index.html

4. Tap **OK** or Save.

STEP 3: Paste the Complete Code Below
Copy all this and paste it directly into index.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Contact Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
    }

    form {
      max-width: 400px;
      margin: auto;
      background: #f9f9f9;
      padding: 15px;
      border-radius: 10px;
      box-shadow: 0 0 10px #ccc;
    }

    label {
      display: block;
      margin-top: 10px;
    }

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

```
    button {
      margin-top: 10px;
      padding: 10px 15px;
      background-color: blue;
      color: white;
      border: none;
      border-radius: 5px;
    }

    /* Mobile responsiveness */
    @media (max-width: 600px) {
      body {
        background-color: #f0f0f0;
      }

      form {
        padding: 10px;
      }

      input, textarea {
        font-size: 16px;
      }
    }
  </style>
</head>
<body>

  <h2>Contact Us</h2>

  <!-- Contact Form -->
  <form>
    <label>Name:</label>
    <input type="text" name="name" required>

    <label>Email:</label>
    <input type="email" name="email" required>
```

```
    <label>Message:</label>
    <textarea name="message" required></textarea>

    <button type="submit">Send</button>
  </form>

  <br><br>

  <!--  JavaScript Button -->
  <button onclick="showMessage()">Click Me</button>

  <script>
    function showMessage() {
      alert("Hello! This is a JavaScript alert.");
    }
  </script>

</body>
</html>
```

STEP 4: Preview the File

1. Tap the ▶ **Play icon** (Preview) in Acode.
2. It will open the page in your browser.
3. Test the:
    - **Contact form fields**
    - **Button** to see if alert works
    - **Responsive layout** by shrinking the browser width

PART B: FOR PC USERS

(**Tools used: Visual Studio Code + Browser**)

 STEP 1: Open Visual Studio Code

1. Launch **VS Code**.
2. Click **File > Open Folder**
3. Choose or create a folder like:
   C:\Users\YourName\Documents\MyWebProject

STEP 2: Create HTML File

1. Inside VS Code, click **File > New File**
2. Name it index.html
3. Save it inside the project folder

STEP 3: Paste the Same HTML Code
(Paste the exact same HTML content above into index.html)

STEP 4: Open the File in the Browser
Two options:

Option A: Live Server

1. Install the **Live Server extension** (from Extensions tab)
2. Right-click index.html → **Open with Live Server**

Option B: Manual Method

1. Go to the project folder on your PC
2. Double-click the index.html file to open it in any browser

FINAL THINGS TO TEST

Feature	What to Check
JavaScript Button	Click it → Alert should pop up
Form	Can you type and submit?
Responsiveness	Shrink screen or rotate phone/tablet

OBJECTIVE:

Create 3 HTML pages and link them using a top menu:

- index.html (Home Page)
- about.html (About Page)
- contact.html (Contact Page)

SECTION A: FOR ANDROID USERS (USING Acode)

Tools You Must Have Installed:

1. **Acode** – HTML editor (from F-Droid or Play Store)
2. (Optional) **Any browser** (Chrome, Firefox, etc.)

STEP 1: Open Acode

1. Tap the **Acode app** to launch it.
2. Wait for the app to fully open.

STEP 2: Create/Open a Folder for Your Website

1. Tap the **three-dot menu (⋮)** at the top-right.
2. Select **“Open Folder”**.
3. Tap the **➕ (plus icon)** to create a new folder.
4. Name the folder:

   ☞ MyWebsite

   Then tap **OK** or **CREATE**.
5. Acode now opens inside this folder.

STEP 3: Create the First Page (index.html)

1. Tap the **➕ (plus icon)** again.
2. Select **“New File”**
3. Type the filename as:

   ☞ index.html

   Then tap **OK** or **SAVE**.
4. You will now see a blank screen with an empty file open.

STEP 4: Paste the Following Code into index.html

Copy everything exactly. This creates the **home page** with a menu.

```
<!DOCTYPE html>
<html>
<head>
  <title>Home Page</title>
  <style>
    nav {
      background-color: #333;
      padding: 10px;
    }
    nav a {
      color: white;
      text-decoration: none;
      margin-right: 20px;
    }
    nav a:hover {
      text-decoration: underline;
```

```
  }
 </style>
</head>
<body>

 <!-- Top Menu -->
 <nav>
  <a href="index.html">Home</a>
  <a href="about.html">About</a>
  <a href="contact.html">Contact</a>
 </nav>

 <h1>Welcome to My Website</h1>
 <p>This is the home page.</p>

</body>
</html>
```

STEP 5: Save the File

1. Tap the **disk/save icon** 🖫 at the top-right corner.

STEP 6: Preview the Page

1. Tap the ▶ **Play icon** at the top.
2. It will open the index.html page in your browser.
3. You will see:
   A navigation menu
   A heading "Welcome to My Website"
   A paragraph

STEP 7: Create about.html

1. Tap the ✚ > “New File”
2. Name it: about.html
3. Tap OK
4. Paste the following code:

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

```
<head>
  <title>About Page</title>
</head>
<body>

  <!-- Top Menu -->
  <nav>
    <a href="index.html">Home</a>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </nav>

  <h1>About Us</h1>
  <p>This is the about page.</p>

</body>
</html>
```

5. Tap the **save icon** 🖫.

STEP 8: Create contact.html

1. Tap ➕ > New File
2. Name it: contact.html
3. Tap OK
4. Paste the following code:

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

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

```
  </nav>

  <h1>Contact Us</h1>
  <p>This is the contact page.</p>

</body>
</html>
```

5. Tap the **save icon** 🖫.

STEP 9: Test Navigation Between Pages

1. In Acode, open index.html.
2. Tap ▶ to preview in browser.
3. Tap “About” → Should open about.html
4. Tap “Contact” → Should open contact.html
5. All pages have the same top menu!

SECTION B: FOR PC USERS (USING VS CODE)

What You Need Installed:

- **Visual Studio Code (VS Code)**
- A web browser (Chrome, Edge, Firefox, etc.)

STEP 1: Create a Folder for Your Website

1. Open **File Explorer** on your PC.
2. Go to any location (like Desktop or Documents).
3. Create a new folder and name it:
   MyWebsite

STEP 2: Open VS Code

1. Open the **Visual Studio Code** app.
2. Click **File > Open Folder**
3. Select the folder you just created (MyWebsite)
4. Click **"Open"**

STEP 3: Create index.html

1. Click **File > New File**
2. Name the file index.html

3. Copy and paste the same code from Android index.html above
4. Save it (File > Save or Ctrl + S)

STEP 4: Create about.html and contact.html

- Repeat the same process:
    - File > New File > Name it about.html, paste its code.
    - File > New File > Name it contact.html, paste its code.
    - Save all files.

STEP 5: Open in Browser

1. Go to the folder MyWebsite on your computer.
2. Double-click index.html
3. It will open in your browser.
4. Click the menu links to navigate between pages.

You now have a 3-page website fully linked together.

MODULE 4: Web Forms and User Interaction

1.0 What Are Web Forms?

Definition:

A **web form** is a webpage section containing **input fields** where users can **enter and submit data** to the website or server.

Function:

- Web forms allow **interaction** between the user and the website.
- Examples: Login pages, contact forms, search boxes, registration forms.

When to Use:

- When collecting user data like names, emails, passwords, messages, etc.
- For actions such as search, login, subscribe, feedback, upload, etc.

2.0 Structure of a Basic HTML Form

Example Code with Comments

```
<!DOCTYPE html> <!-- HTML5 declaration -->
<html>
<head>
  <title>Contact Form</title> <!-- Page title -->
```

```
</head>
<body>
  <h2>Contact Us</h2>

  <!-- Start of form -->
  <form action="https://example.com/contact" method="post">
    <!-- Label and input field for name -->
    <label for="name">Name:</label><br>
    <input type="text" id="name" name="user_name"><br><br>

    <!-- Label and input field for email -->
    <label for="email">Email:</label><br>
    <input type="email" id="email" name="user_email"><br><br>

    <!-- Submit button -->
    <input type="submit" value="Send Message">
  </form>
</body>
</html>
```

Code Breakdown:

Tag	Purpose
<form>	Starts the form. action defines where data goes; method defines how.
<label>	Describes each input field.
<input type="text">	Field for text (e.g., name).
<input type="email">	Field for email with built-in validation.
<input type="submit">	Sends form data to the server.

3.0 Key HTML Form Elements

Tag	Use	Example
<input type="text">	Single-line text	Name, username
<input type="password">	Password fields	Login forms

Tag	Use	Example
<input type="email">	Validates email format	Registration
<textarea>	Multi-line text	Comments, messages
<select>	Drop-down menu	Country, gender
<input type="checkbox">	Multiple options	Hobbies
<input type="radio">	Single option from a group	Gender
<input type="submit">	Submit button	Save form

4.0 Validating Form Input (Introductory Level)

Using HTML5 Validation

<input type="text" name="name" required> <!-- Must be filled -->

<input type="email" name="email" required> <!-- Must be a valid email -->

- required: Makes a field mandatory
- type="email": Validates format automatically

Here is the **full HTML code** demonstrating **HTML5 validation** using required and type="email":

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML5 Form Validation</title>
</head>
<body>

  <h2>✍ Using HTML5 Validation</h2>

  <form action="#" method="post">
    <!-- Name input: required field -->
    <label for="name">Full Name:</label><br>
    <input type="text" id="name" name="name" required>
    <br><br>

    <!-- Email input: must be valid email format and required -->
    <label for="email">Email Address:</label><br>
    <input type="email" id="email" name="email" required>
    <br><br>

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

```
  </form>

  <p>
    • <code>required</code>: Makes a field mandatory<br>
    • <code>type="email"</code>: Validates format automatically
  </p>

</body>
</html>
```

What it does:

- required: Prevents form submission if the field is empty.
- type="email": Ensures the entered value follows the correct email format (like user@example.com).

.0 Hands-on Practical: Feedback Form

Code Example with Comments

```
<!DOCTYPE html>
<html>
<head>
  <title>Feedback Form</title>
</head>
<body>
  <h2>Student Feedback</h2>

  <!-- Feedback form -->
  <form action="/submit-feedback" method="post">
    <label for="fullname">Full Name:</label><br>
    <input type="text" id="fullname" name="fullname" required><br><br>

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

    <label for="feedback">Your Feedback:</label><br>
    <textarea id="feedback" name="feedback" rows="4" cols="40"
required></textarea><br><br>

    <input type="submit" value="Submit Feedback">
```

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

6.0 How to Run This Form Locally

On Laptop (VS Code + Live Server)

1. Save the code as form.html
2. Right-click → "Open with Live Server"
3. The form appears in the browser
4. Since there's no server backend, submission will not store data (for now)

On Android (Acode)

1. Create a new file form.html
2. Paste the code
3. Save and preview using 👁 icon

HTML5 Multimedia and Forms continue

Overview

In this module, you will learn:

1. How to use **audio and video** in a webpage
2. How to handle **non-supporting browsers**
3. How to create **interactive forms**
4. New HTML5 input types (email, tel, range, date, etc.)

Everything will be explained in simple terms with:

- Definitions
- When and how to use
- Hands-on examples
- 3 practice exercises + solutions

1. HTML5 AUDIO & VIDEO

What is Multimedia in HTML5?

Multimedia includes content like **audio and video** — music, podcasts, tutorials, movies that you can embed directly into a web page.

Before HTML5, you needed Flash. Now you can use:

- <audio> for sound/music
- <video> for movies/lectures

Supported Media Types

Media Tag	Supported Formats
<audio>	MP3, OGG, WAV
<video>	MP4, WebM, OGG

Example 1: HTML5 Audio

Here's a **clean, mobile-friendly HTML5 audio player** that works perfectly on **Android phones, tablets, and most mobile browsers** (like Chrome, Firefox, Opera, Edge, Brave, etc.).

HTML5 Audio Code for Android/Mobile Support

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Audio Player</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
      background-color: #f0f8ff;
      text-align: center;
    }
    audio {
      width: 90%;
      max-width: 400px;
      margin-top: 20px;
    }
  </style>
</head>
<body>

  <h2> Play Audio on Mobile</h2>

  <audio controls>
    <source src="song.mp3" type="audio/mpeg">
    Your browser does not support the audio element.
  </audio>
```

Make sure the file song.mp3 is saved in the same folder as this HTML file.

</body>
</html>

How to Use on Android

1. **Open a code editor** like:
    - Acode (recommended)
    - AWD
    - HTML Editor
    - TrebEdit
2. Create a new file: audio.html
3. Save your MP3 file as: song.mp3
4. Place both audio.html and song.mp3 in the **same folder**
5. Open audio.html in a mobile browser (e.g., Chrome)
6. Tap ▶ to play — works offline too!

Notes:

- Works without internet (as long as the .mp3 file is stored locally)
- If you rename the file (e.g., music.mp3), make sure to update: <source src="music.mp3" type="audio/mpeg">

FEATURES:

1. **Multiple Songs**
2. **Playlist-Style Navigation**
3. **Download Buttons for Each Song**
4. **Works Perfectly on Android / Mobile Browsers**

BEFORE YOU BEGIN:

Ensure you have your audio files in the same folder as your HTML file. For this example, we'll use:

- song1.mp3
- song2.mp3
- song3.mp3

Full Code: playlist.html

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

```
<head>
  <meta charset="UTF-8">
  <title>My Music Playlist</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
      background-color: #f8f9fa;
      color: #333;
    }

    h2 {
      text-align: center;
      margin-bottom: 30px;
    }

    .song {
      margin-bottom: 25px;
      padding: 15px;
      border: 1px solid #ccc;
      border-radius: 10px;
      background-color: #fff;
    }

    .song h3 {
      margin: 0 0 10px;
    }

    audio {
      width: 100%;
    }

    .download {
      display: inline-block;
      margin-top: 10px;
      padding: 8px 12px;
      background-color: #007bff;
      color: white;
```

```
    text-decoration: none;
    border-radius: 5px;
   }

   .download:hover {
    background-color: #0056b3;
   }
  </style>
</head>
<body>

  <h2>♪ My Music Playlist</h2>

  <!-- ✅ Song 1 -->
  <div class="song">
   <h3>Track 1: My First Song</h3>
   <audio controls>
    <source src="song1.mp3" type="audio/mpeg">
    Your browser does not support the audio element.
   </audio>
   <br>
   <a href="song1.mp3" download class="download">⬇ Download</a>
  </div>

  <!-- ✅ Song 2 -->
  <div class="song">
   <h3>Track 2: Chill Vibes</h3>
   <audio controls>
    <source src="song2.mp3" type="audio/mpeg">
    Your browser does not support the audio element.
   </audio>
   <br>
   <a href="song2.mp3" download class="download">⬇ Download</a>
  </div>

  <!--  Song 3 -->
  <div class="song">
   <h3>Track 3: Motivation Beats</h3>
```

```
    <audio controls>
      <source src="song3.mp3" type="audio/mpeg">
      Your browser does not support the audio element.
    </audio>
    <br>
    <a href="song3.mp3" download class="download">⬇ Download</a>
  </div>

</body>
</html>
```

How to Use This on Android:

1. **Create this file** as playlist.html
2. Save it in the same folder with:
    - song1.mp3
    - song2.mp3
    - song3.mp3
3. Open playlist.html in:
    - **Acode**, **TrebEdit**, or any HTML editor
    - Or open directly in **Chrome/Firefox browser**
4. Tap ▶ to play
5. Tap ⬇ **Download** to save the song to your phone's music/download folder

Tips:

- You can rename tracks and audio files — just update the src and title.
- Add more tracks by repeating the same ... block.
- Files must be .mp3 for best browser support.

Example 2: HTML5 Video

The following code **can work in Acode on Android** — but only **if the video file (movie.mp4) is available in the correct location**.

CODE:

```
<!DOCTYPE html>
<html>
<head><title>Video Test</title></head>
<body>
  <h2>Watch Video</h2>
  <video width="320" height="240" controls>
```

```
    <source src="movie.mp4" type="video/mp4">
    Your browser does not support video.
  </video>
</body>
</html>
```

TO MAKE IT WORK IN ACODE:

STEP 1: Prepare the video

- Place your video file named movie.mp4 **in the same folder** as the HTML file.

STEP 2: Open Acode

- Launch **Acode** app.

STEP 3: Create a new file

- Tap ➕ → **New File**
- Save it as: index.html

STEP 4: Paste the full code above

STEP 5: Save the file

- Tap the 💾 icon.

STEP 6: Preview the HTML

- Tap ▶ or 👁 (eye icon) to preview in your browser (like Chrome).

STEP 7: Watch the video

- If the movie.mp4 is in the right folder, the video player will appear and play the video.

IMPORTANT:

File Location Matters!

Both the index.html and movie.mp4 **must be in the same folder** inside Acode.

- If you see **"Your browser does not support video"**, then:
    - The video file wasn't found
    - Or the file name is wrong (e.g., Movie.mp4 ≠ , movie.mp4)
    - Or the format is not supported

What it does:

- Plays a video directly in the browser.

2. Dealing with Non-Supporting Browsers

If a browser doesn't support a format, use a **fallback message**:

```
<video controls>
  <source src="video.mp4" type="video/mp4">
  <source src="video.ogg" type="video/ogg">
  Your browser does not support this video.
</video>
```

The browser picks the first one it can play.

3. HTML5 FORMS

What is a Form?

Definition:

An **HTML form** is a section of a web page that contains **input elements** (like text boxes, buttons, checkboxes, etc.) and allows users to **submit data** to a server or handle it directly on the page.

It's like a **digital data collection tool** that helps you gather:

- Contact information
- Survey responses
- Login credentials
- File uploads
- And more!

Example of a Simple Form:

```
<form action="submit.php" method="POST">
  <label for="name">Your Name:</label>
  <input type="text" id="name" name="name" required>

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

  <button type="submit">Send</button>
</form>
```

Common HTML Form Elements:

Element	Description
<form>	The container that holds all form inputs
<input>	Text fields, emails, passwords, checkboxes, etc.
<textarea>	Multiline input for messages/comments
<label>	Describes each input field
<select>	Drop-down list
<button>	Submit or reset the form
<fieldset>	Groups related inputs

Key Attributes:

Attribute	Meaning
action	The URL or page where form data is sent
method	HTTP method to use: GET or POST
name	Unique name used to identify each input
required	Makes the field mandatory

Types of <input>:

```
<input type="text">       <!-- Single-line text -->
<input type="email">      <!-- Email address -->
<input type="password">   <!-- Hidden characters -->
<input type="file">       <!-- Upload a file -->
<input type="radio">      <!-- Single-choice option -->
<input type="checkbox">   <!-- Multiple-choice option -->
<input type="submit">     <!-- Submit button -->
```

What Happens When a Form is Submitted?

1. The form collects the values from all inputs.
2. Sends the data to the location in the action (e.g., a PHP page, a script, or an email).
3. The server or tool processes the data (e.g., saves, sends email, authenticates login).

Where Are Forms Used?

- Contact forms
- User registration/login
- Feedback/survey forms
- Shopping carts
- Booking/reservation systems

HTML contact form that lets users send a message — and it **works on mobile or desktop**. You'll be able to receive the message **by email**, using FormSubmit.co (no coding or server needed).

Simple Contact Form That Sends Message via Email

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

```
<head>
  <meta charset="UTF-8">
  <title>Simple Contact Form</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    body {
      font-family: Arial, sans-serif;
      background: #f2f2f2;
      padding: 20px;
    }
    h2 {
      text-align: center;
    }
    form {
      background: white;
      padding: 20px;
      max-width: 500px;
      margin: auto;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
    }
    input, textarea {
      width: 100%;
      padding: 10px;
      margin-top: 10px;
      border-radius: 5px;
      border: 1px solid #ccc;
    }
    button {
      margin-top: 15px;
      padding: 10px;
      width: 100%;
      background-color: #007bff;
      color: white;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }
    button:hover {
```

```
    background-color: #0056b3;
   }
  </style>
</head>
<body>

  <h2>Contact Me</h2>
  <form action="https://formsubmit.co/YOUR_EMAIL@example.com" method="POST">
   <!-- Replace YOUR_EMAIL@example.com with your real email -->
   <input type="text" name="name" placeholder="Your Name" required>
   <input type="email" name="email" placeholder="Your Email" required>
   <textarea name="message" rows="5" placeholder="Your Message"
required></textarea>
   <!-- Optional: Disable CAPTCHA -->
   <input type="hidden" name="_captcha" value="false">

   <!-- Optional: Redirect after submit -->
   <input type="hidden" name="_next" value="thank-you.html">

   <button type="submit">Send Message</button>
  </form>

</body>
</html>
```

How to Make This Work

Step-by-Step:

1. Replace YOUR_EMAIL@example.com with **your real email**.
   Example:
   action="https://formsubmit.co/myname@gmail.com"
2. Open the file in a browser or mobile app (like Acode or TrebEdit)
3. When a user fills and submits the form:
   - You'll get an email with their message!
   - Optionally, they'll be redirected to thank-you.html (you can create this page too).
4. **You'll get a confirmation email from FormSubmit** the first time. Click it once — and then you're all set!

Optional: Simple thank-you.html

Create this as thank-you.html in the same folder:

```
<!DOCTYPE html>
<html>
<head>
  <title>Thank You</title>
</head>
<body>
  <h2>✅ Message Sent!</h2>
  <p>Thanks for reaching out. I'll get back to you soon.</p>
  <a href="index.html">Back to Home</a>
</body>
</html>
```

Adding features to HTML forms:

1. FILE UPLOAD FIELD

This allows users to upload files like PDFs, images, or documents.

2. LOGIN-STYLE FORM

A simple form that collects username and password.

PART 1: Contact Form with File Upload (Email-enabled)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Contact Form with File Upload</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    body {
      font-family: sans-serif;
      background: #eef;
      padding: 20px;
    }
    form {
      background: white;
      padding: 20px;
      max-width: 500px;
      margin: auto;
```

```
    border-radius: 10px;
    box-shadow: 0 0 8px rgba(0,0,0,0.1);
   }
   input, textarea {
    width: 100%;
    padding: 10px;
    margin: 8px 0;
    border-radius: 5px;
    border: 1px solid #ccc;
   }
   button {
    padding: 10px;
    background: #007bff;
    color: white;
    border: none;
    width: 100%;
    border-radius: 5px;
   }
   button:hover {
    background: #0056b3;
   }
  </style>
</head>
<body>

  <h2> Contact + File Upload</h2>

  <form action="https://formsubmit.co/YOUR_EMAIL@example.com" method="POST"
enctype="multipart/form-data">
   <input type="text" name="name" placeholder="Your Name" required>

   <input type="email" name="email" placeholder="Your Email" required>

   <textarea name="message" rows="5" placeholder="Your Message" required></textarea>

   <!-- ✅ File upload field -->
   <input type="file" name="attachment">

   <!-- Optional hidden fields -->
```

```
    <input type="hidden" name="_captcha" value="false">
    <input type="hidden" name="_next" value="thank-you.html">

    <button type="submit">Send Message + File</button>
  </form>

</body>
</html>
```

Just replace YOUR_EMAIL@example.com with your real email. FormSubmit will send the uploaded file to your inbox.

PART 2: Login-Style Form (No backend – only visual)

This is a frontend-only form, great for mockups, demos, or pairing with JavaScript/PHP later.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Login Form</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    body {
      background: #f0f0f0;
      font-family: sans-serif;
      padding: 40px;
    }
    .login-form {
      background: white;
      padding: 30px;
      max-width: 400px;
      margin: auto;
      border-radius: 10px;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
    }
    input {
      width: 100%;
      padding: 10px;
      margin: 10px 0;
```

```
    border-radius: 5px;
    border: 1px solid #ccc;
  }
  button {
    width: 100%;
    padding: 10px;
    background: #28a745;
    color: white;
    border: none;
    border-radius: 5px;
  }
  button:hover {
    background: #218838;
  }
 </style>
</head>
<body>
 <h2 style="text-align:center;"> Login Form</h2>
 <form class="login-form" action="#" method="POST">
  <input type="text" name="username" placeholder="Username" required>
  <input type="password" name="password" placeholder="Password" required>
  <button type="submit">Log In</button>
 </form>
</body>
</html>
```

Note: This login form won't actually log anyone in unless you connect it to a backend (like PHP, Django, Node.js). It’s perfect for learning and UI testing.

ENHANCING BOTH FORMS STEP BY STEP WITH:

A. File Upload Form with Preview

Users can upload a file **and see its name/preview before submitting**.

B. Login Form with JavaScript Validation

We'll check if the **username and password are filled** before submission.

A. File Upload Form with Live File Name Display

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>File Upload with Preview</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    body {
      background: #f9f9f9;
      font-family: sans-serif;
      padding: 20px;
    }
    form {
      background: white;
      padding: 20px;
      max-width: 500px;
      margin: auto;
      border-radius: 10px;
      box-shadow: 0 0 8px rgba(0,0,0,0.1);
    }
    input, textarea {
      width: 100%;
      padding: 10px;
      margin: 10px 0;
      border-radius: 5px;
      border: 1px solid #ccc;
    }
    button {
      background-color: #007bff;
      color: white;
      border: none;
      padding: 10px;
      width: 100%;
      border-radius: 5px;
    }
    button:hover {
      background-color: #0056b3;
    }
    #fileLabel {
      font-size: 14px;
```

```
    margin-top: 5px;
    color: #333;
  }
 </style>
</head>
<body>

 <h2> File Upload with Preview</h2>

 <form action="https://formsubmit.co/YOUR_EMAIL@example.com" method="POST"
enctype="multipart/form-data">
  <input type="text" name="name" placeholder="Your Name" required>

  <input type="email" name="email" placeholder="Your Email" required>

  <textarea name="message" rows="5" placeholder="Your Message" required></textarea>

  <!-- ✅ File upload -->
  <input type="file" id="fileInput" name="attachment" required>
  <p id="fileLabel">No file chosen</p>

  <!-- Hidden fields -->
  <input type="hidden" name="_captcha" value="false">
  <input type="hidden" name="_next" value="thank-you.html">

  <button type="submit">Send Message + File</button>
 </form>

 <script>
  const fileInput = document.getElementById("fileInput");
  const fileLabel = document.getElementById("fileLabel");

  fileInput.addEventListener("change", function () {
   fileLabel.textContent = fileInput.files.length > 0
    ? "Selected file: " + fileInput.files[0].name
    : "No file chosen";
  });
 </script>
```

```
</body>
</html>
```
Replace YOUR_EMAIL@example.com with your email to receive submissions.

B. Login Form with JavaScript Validation

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Login Validation</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    body {
      font-family: sans-serif;
      background: #f0f0f0;
      padding: 20px;
    }
    .login-form {
      background: white;
      padding: 25px;
      max-width: 400px;
      margin: auto;
      border-radius: 10px;
      box-shadow: 0 0 8px rgba(0,0,0,0.1);
    }
    input {
      width: 100%;
      padding: 10px;
      margin: 10px 0;
      border-radius: 5px;
      border: 1px solid #ccc;
    }
    button {
      width: 100%;
      padding: 10px;
      background: #28a745;
      color: white;
      border: none;
```

```
    border-radius: 5px;
   }
   button:hover {
    background: #218838;
   }
   .error {
    color: red;
    font-size: 14px;
    margin-top: -5px;
   }
  </style>
</head>
<body>

  <h2 style="text-align:center;"> Login</h2>

  <form class="login-form" onsubmit="return validateLogin()">
   <input type="text" id="username" placeholder="Username">
   <div id="userError" class="error"></div>

   <input type="password" id="password" placeholder="Password">
   <div id="passError" class="error"></div>

   <button type="submit">Log In</button>
  </form>

  <script>
   function validateLogin() {
    let valid = true;
    const user = document.getElementById("username");
    const pass = document.getElementById("password");
    const userError = document.getElementById("userError");
    const passError = document.getElementById("passError");

    userError.textContent = "";
    passError.textContent = "";

    if (user.value.trim() === "") {
     userError.textContent = "Username is required";
```

```
    valid = false;
  }
  if (pass.value.trim() === "") {
    passError.textContent = "Password is required";
    valid = false;
  }

  return valid;
 }
</script>

</body>
</html>
```

This form checks that **both fields are filled** before allowing submission.

How to Use on Mobile (Android):

1. Use a mobile HTML editor (like **Acode**, **TrebEdit**, or **Spck**).
2. Create two files:
    - upload.html for the file upload form
    - login.html for the login form
3. Replace email in the form action (YOUR_EMAIL@example.com)
4. Open the files in browser and test.

7.0 Practice Exercises

Exercise 1

Q: Create a form that collects:

- Name
- Gender (drop-down)
- Favorite Subject (checkboxes for HTML, CSS, JS)

Solution:

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple Form</title>
</head>
<body>
```

```
  <form action="#" method="post">
    <label>Name:</label>
    <input type="text" name="name"><br><br>

    <label>Gender:</label>
    <select name="gender">
      <option>Male</option>
      <option>Female</option>
      <option>Other</option>
    </select><br><br>

    <label>Favorite Subject:</label><br>
    <input type="checkbox" name="subject" value="HTML"> HTML<br>
    <input type="checkbox" name="subject" value="CSS"> CSS<br>
    <input type="checkbox" name="subject" value="JS"> JavaScript<br><br>

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

</body>
</html>
```

Exercise 2

Q: What is the difference between input type="text" and textarea?

A:

- input type="text": Single-line input (e.g., name)
- textarea: Multi-line input (e.g., comments, feedback)

Exercise 3

Q: What is the function of the action and method attributes in a form?

A:

- action: The URL or script where form data is sent
- method: How data is sent — post (secure, hidden in URL), or get (visible in URL)

Exercise 4

Q: Why is required important in form inputs?

A:

It ensures users **cannot submit** the form until that field is filled, improving **data quality**.

8.0 Summary

Concept	Purpose
Web Form	Collect user data on a web page
<form>	Container for input fields
Input types	Used to collect specific kinds of data (text, email, password, etc.)
HTML Validation	Ensures users enter correct and complete data

End of Module 4

MODULE 5: Introduction to Style Sheets (CSS)

Overview

In this module, you'll learn:

1. What CSS is and why it's important
2. The different types of CSS (inline, internal, external)
3. How to add CSS to your HTML page
4. CSS3 basics and key terms
5. Hands-on examples and 3 practical exercises with solutions

1. What is CSS?

Definition

CSS stands for **Cascading Style Sheets**.

It is used to **style** and **beautify** web pages.

With CSS, you can:

- Change colors, fonts, margins, and spacing
- Add borders, background images
- Create responsive layouts

Why is CSS Important?

- Keeps HTML clean (structure only)
- Separates **content** from **design**
- Makes it easier to update the look of a website

2. Types of CSS

Type	Where It's Used	Example
Inline	Inside an HTML tag	Hello

Type	Where It's Used	Example
Internal	Inside <style> tag in HTML <head>	<style> p {color:blue;} </style>
External	In a separate .css file	styles.css linked using <link> in HTML

---

3. Adding CSS to Your HTML

Example 1: Inline CSS

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

Use only for quick or one-time styling.

Example 2: Internal CSS (inside <head>)

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 {
      color: green;
      text-align: center;
    }
  </style>
</head>
<body>
  <h1>Welcome to My Site</h1>
</body>
</html>
```

Example 3: External CSS (recommended)

styles.css (Create this file):

```
body {
  background-color: #f0f0f0;
}

h1 {
  color: navy;
}
```

index.html:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Styled by External CSS</h1>
</body>
</html>
```

External CSS is best for big projects and clean design.

4. Understanding CSS3 & Terminology

CSS Rule Structure

```
selector {
  property: value;
}
```

🔍 Example:

```
p {
  color: blue;
  font-size: 18px;
}
```

Term	**Meaning**
Selector	Targets the HTML element (e.g. p)
Property	What you want to change (e.g. color)
Value	New value for that property (e.g. blue)

HANDS-ON EXAMPLES

Example 4: Add Background and Font

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
```

```
    background-color: lightyellow;
    font-family: Arial;
  }

  h2 {
    color: purple;
  }
 </style>
</head>
<body>
 <h2>Styled Header</h2>
 <p>This is a styled paragraph.</p>
</body>
</html>
```

Example 5: Create a Styled Box

```
<!DOCTYPE html>
<html>
<head>
 <style>
  .box {
    width: 200px;
    height: 100px;
    background-color: lightblue;
    border: 2px solid blue;
    text-align: center;
    line-height: 100px;
  }
 </style>
</head>
<body>
 <div class="box">This is a box</div>
</body>
</html>
```

PRACTICAL EXERCISES

Exercise 1: Change Text Color Using Inline CSS

Solution:

```
<p style="color:green;">This is green text.</p>
```

Exercise 2: Style a Page with Internal CSS

Solution:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      background-color: beige;
    }
    h1 {
      color: red;
      font-family: 'Verdana';
    }
  </style>
</head>
<body>
  <h1>My Styled Page</h1>
</body>
</html>
```

Exercise 3: Create a CSS File and Link It

styles.css:

```
p {
  font-size: 18px;
  color: navy;
}
```

index.html:

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

```
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <p>This paragraph is styled externally.</p>
</body>
</html>
```

7.0 Hands-on Practice: Simple Styled Page

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      background-color: #f0f0f0;
    }

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

    p {
      font-size: 18px;
      padding: 10px;
    }
  </style>
</head>
<body>
  <h1>CSS Demo Page</h1>
  <p>This web page is styled using internal CSS.</p>
</body>
</html>
```

8.0 How to Use on Devices

Android (Acode or Termux)

1. Create two files: index.html and style.css

2. In index.html, link the CSS: <link rel="stylesheet" href="style.css">
3. Save both files
4. Preview in browser

Laptop (VS Code)

1. Create both files in a folder
2. Use **Live Server** to preview changes in real time

9.0 Practice Exercises

Exercise 1
: Create a paragraph styled to:

- Font size: 20px
- Color: blue
- Background color: yellow

Solution:

```
<p style="font-size: 20px; color: blue; background-color: yellow;">
  This is a styled paragraph.
</p>
```

Here's the **full HTML code** that includes your styled paragraph inside a complete HTML page:

```
<!DOCTYPE html>
<html>
<head>
  <title>Styled Paragraph Example</title>
</head>
<body>

  <p style="font-size: 20px; color: blue; background-color: yellow;">
    This is a styled paragraph.
  </p>

</body>
</html>
```

Exercise 2
Q: Write a CSS rule to make all <h2> headings green and center-aligned.

Solution:

```
h2 {
```

```
  color: green;
  text-align: center;
}
```

Here is the **full HTML + CSS code** that uses your rule to make all <h2> headings green and center-aligned:

```
<!DOCTYPE html>
<html>
<head>
  <title>Styled h2 Example</title>
  <style>
    h2 {
      color: green;
      text-align: center;
    }
  </style>
</head>
<body>

  <h2>This is a Green and Centered Heading</h2>
  <h2>Another Styled h2 Heading</h2>

</body>
</html>
```

Exercise 3

Q: Differentiate between inline, internal, and external CSS.

Answer:

Type	Where	Advantage
Inline	Inside HTML tag	Quick but not reusable
Internal	Inside <style>	Good for one-page sites
External	In .css file	Best for large, multi-page sites

Exercise 4

Q: Fill in the blanks:

1. CSS stands for _____
2. The best way to reuse CSS across many pages is _____
3. CSS uses _____ and _____ pairs to style HTML.

Answers:

1. Cascading Style Sheets
2. External CSS
3. Property and value

10.0 Summary

Key Point	Description
CSS	Adds style and layout to HTML pages
Inline CSS	Inside element tag
Internal CSS	Inside <style> tag in <head>
External CSS	Linked via .css file for scalability
Best Practice	Use external CSS for clean, maintainable code

End of Module 5

MODULE 6: Web Development Tools and Frameworks

1.0 What Are Web Development Tools?

Definition:

Web development tools (also called dev tools or editors) are **software applications** that help developers **write, edit, test, and manage web code** like HTML, CSS, JavaScript, and server-side languages.

2.0 What Are Web Frameworks?

Definition:

A **web framework** is a **pre-built collection of tools, templates, and libraries** that makes building websites or web apps faster and easier.

3.0 Functions and Benefits

Tool/Framework	Function	Benefit
Code Editor	Write/edit code	Syntax highlighting, auto-completion
Browser Dev Tools	Inspect/debug websites	View styles, console errors
Frameworks (e.g. Bootstrap)	Pre-designed UI components	Speeds up development
Local Servers (Live Server, XAMPP)	Run and test pages locally	View real-time changes

Tool/Framework	Function	Benefit
Version Control (e.g., Git)	Track changes	Collaborate and back up code

4.0 Popular Web Development Tools

Platform	Tool	Purpose
Mobile (Android)	Acode, Termux	Mobile code editing and server setup
Mobile (iPhone)	Textastic, Kodex	Code writing and file preview
Desktop	VS Code, Sublime Text	Full-featured code editing
Browser	DevTools (Chrome/Firefox)	Inspect HTML/CSS/JS in real time

5.0 Practical Steps: Getting Started with Tools

Using VS Code on Laptop/Desktop

1. **Download VS Code** from https://code.visualstudio.com
2. Install **Live Server Extension**
    - Click Extensions icon → Search for **Live Server** → Click Install
3. Create a folder for your project
4. Inside it, create two files: index.html and style.css
5. Right-click index.html → Click **"Open with Live Server"**
6. Your site opens in the browser automatically

Using Acode on Android

1. Open the **Acode** app
2. Tap menu → Open Folder → Navigate to desired location
3. Tap **+** to create index.html
4. Write HTML/CSS code
5. Tap **Preview** to open in browser
6. For CSS, create a style.css file and link it in HTML

Using Termux on Android

1. Open **Termux**
2. Run:
   pkg install nano python
   mkdir mysite
   cd mysite

nano index.html

3. Paste your HTML code
4. Save with Ctrl + X → Y → Enter
5. Start a local server:
   python -m http.server 8080
6. In browser, visit:
   http://localhost:8080/index.html

Using Kodex or Textastic on iPhone

1. Open Kodex or Textastic
2. Create a file → name it index.html
3. Write your HTML code
4. Use built-in preview or open via Files app

6.0 Popular Frontend Frameworks

Framework	Function	Why Use It
Bootstrap	Layout and UI components	Mobile-ready, fast design
Tailwind CSS	Utility-first CSS framework	Highly customizable
Foundation	Responsive grid & components	Alternative to Bootstrap

7.0 Web Development Workflow Using Tools

1. **Plan** your site layout
2. **Write HTML** in your editor
3. **Style with CSS** (external file preferred)
4. **Test locally** using Acode preview or Live Server
5. **Debug** using DevTools
6. **Publish** your files using GitHub Pages or a hosting service

8.0 Sample Project Using VS Code

Files:

- index.html
- style.css

index.html

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

```
<head>
  <title>My Project</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This is styled using external CSS.</p>
</body>
</html>
```

style.css

```
body {
  background-color: #f5f5f5;
  font-family: Arial, sans-serif;
}

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

9.0 Practice Exercises

Exercise 1

What is the difference between a web development tool and a framework?

Answer:

- A **tool** helps you write or test code (e.g., VS Code, Acode).
- A **framework** provides ready-made components to speed up development (e.g., Bootstrap).

Exercise 2

Q: Create a web page using VS Code with:

- An external CSS file
- A heading "Web Tools"
- A paragraph saying "VS Code and Live Server are amazing!"
- Color: green
- Font size: 20px

Solution:

index.html

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
  <title>Web Tools</title>
</head>
<body>
  <h1>Web Tools</h1>
  <p>VS Code and Live Server are amazing!</p>
</body>
</html>
```

style.css

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

Exercise 3

Q: List three web development tools for Android, iPhone, and Desktop.

Answer:

Platform	Tool
Android	Acode, Termux
iPhone	Textastic, Kodex
Desktop	VS Code, Sublime Text

Exercise 4

Q: What does the Live Server extension do in VS Code?

Answer:

It creates a local web server and reloads your browser instantly when you make changes to your HTML/CSS/JS files.

10.0 Summary

Tool Type	Examples	Purpose
Code Editors	VS Code, Acode, Kodex	Write and edit code
Preview Tools	Live Server, Acode Preview	Test locally
Frameworks	Bootstrap, Tailwind	Speed up and style website design

End of Module 6

MODULE 7: Building a Simple Website (Using HTML5 and CSS)

1.0 Goal of This Module

In this module, you will learn to:

- Create a complete **multi-section webpage** using valid **HTML5** and **CSS**
- Organize information clearly (using semantic tags)
- Apply proper **web standards** (W3C-compliant)

2.0 What You Will Build

A Simple Personal Website with:

- A homepage
- About section
- Services section
- Contact form

  Styled with **CSS**, and mobile-friendly using **basic responsive rules**.

3.0 Website Folder Structure

Create a folder named mywebsite with the following files:

```
mywebsite/
│
├── index.html
├── style.css
```

4.0 Step-by-Step Website Code

index.html

```
<!DOCTYPE html> <!-- Declares HTML5 document -->
```

```
<html lang="en">
<head>
  <meta charset="UTF-8"> <!-- Character encoding -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!--
Responsive setup -->
  <title>My Personal Website</title>
  <link rel="stylesheet" href="style.css"> <!-- Link external CSS -->
</head>
<body>

  <!-- Header Section -->
  <header>
    <h1>Welcome to My Website</h1>
    <p>Your one-stop intro to my web journey</p>
  </header>

  <!-- About Section -->
  <section id="about">
    <h2>About Me</h2>
    <p>I am a beginner in web design, building my first site with HTML and CSS.</p>
  </section>

  <!-- Services Section -->
  <section id="services">
    <h2>What I Can Do</h2>
    <ul>
      <li>Create basic HTML pages</li>
      <li>Style websites with CSS</li>
      <li>Organize website content</li>
    </ul>
  </section>

  <!-- Contact Form -->
  <section id="contact">
    <h2>Contact Me</h2>
    <form action="#" method="post">
      <label for="name">Name:</label><br>
```

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

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

      <label for="message">Message:</label><br>
      <textarea id="message" name="message" rows="4" required></textarea><br><br>

      <input type="submit" value="Send">
    </form>
  </section>

  <!-- Footer -->
  <footer>
    <p>© 2025 My Personal Site. All rights reserved.</p>
  </footer>

</body>
</html>
```

style.css

```
/* Set default styles for the body */
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  line-height: 1.6;
  background-color: #f9f9f9;
}

/* Style the header */
header {
  background-color: #333;
  color: white;
  padding: 20px;
  text-align: center;
```

```
}

/* Style sections */
section {
  padding: 20px;
  margin: 10px;
}

/* Style the contact form inputs */
input[type="text"],
input[type="email"],
textarea {
  width: 100%;
  padding: 10px;
  margin-top: 5px;
  border: 1px solid #ccc;
}

/* Submit button style */
input[type="submit"] {
  background-color: #007BFF;
  color: white;
  border: none;
  padding: 10px 20px;
  margin-top: 10px;
  cursor: pointer;
}

/* Footer style */
footer {
  background-color: #222;
  color: #eee;
  text-align: center;
  padding: 10px;
}

/* Make site responsive */
```

```
@media (max-width: 600px) {
  body {
    font-size: 16px;
  }
}
```

5.0 How to View It

On Laptop (Using VS Code + Live Server)

1. Open VS Code
2. Open the mywebsite folder
3. Right-click index.html → **Open with Live Server**

On Android (Using Acode)

1. Open Acode
2. Create files index.html and style.css
3. Tap preview 👁 to open it in a browser

6.0 Practice Exercises

Exercise 1

Q: Add a new section to your website titled **"My Goals"** with a list of three goals.

Solution:

Here is the **full HTML code** that includes the "My Goals" section in a complete web page:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Personal Website</title>
</head>
<body>

  <header>
    <h1>Welcome to My Website</h1>
  </header>

  <section id="goals">
    <h2>My Goals</h2>
    <ul>
      <li>Learn HTML</li>
      <li>Master CSS</li>
```

```
      <li>Build interactive websites</li>
    </ul>
  </section>

</body>
</html>
```

Exercise 2

Q: Change the background color of the header to dark blue (#001f3f).

Solution:

Here's the **complete HTML and CSS code** that applies your solution — a dark blue background for the <header>:

```
<!DOCTYPE html>
<html>
<head>
  <title>Styled Header Example</title>
  <style>
    header {
      background-color: #001f3f;
      color: white;
      padding: 20px;
      text-align: center;
    }
  </style>
</head>
<body>

  <header>
    <h1>Welcome to My Website</h1>
    <p>Your journey to web development starts here.</p>
  </header>

</body>
</html>
```

This styles the header with:

- Dark blue background (#001f3f)
- White text
- Padding for spacing
- Center-aligned content

Exercise 3

Q: Make all paragraph text (<p>) appear in dark green color.

✅ **Solution**:

```
p {
  color: darkgreen;
}
```

Here is the **full HTML + CSS code** to make **all paragraph text (<p>) appear in dark green**:

```
<!DOCTYPE html>
<html>
<head>
  <title>Dark Green Paragraph</title>
  <style>
    /* ✅ Make all paragraph text dark green */
    p {
      color: darkgreen;
    }
  </style>
</head>
<body>

  <h1>Example Page</h1>

  <p>This is the first paragraph. It should be dark green.</p>
  <p>This is the second paragraph, also in dark green color.</p>
  <p>Another example of a paragraph with dark green text.</p>

</body>
</html>
```

What this does:

- The <style> block inside the <head> applies a CSS rule that changes the text color of all <p> elements to darkgreen.

7.0 Summary

Component	Role
HTML5	Structures the content using valid tags
CSS	Styles the page layout and elements
<header>, <section>, <footer>	Semantic tags for better readability and SEO
style.css	External file for styling, reusable and clean

End of Module 7

MODULE 8: HTML vs XHTML vs XML

1.0 Definitions

HTML (HyperText Markup Language)

- The **standard markup language** used to **create and structure content** on the web.
- Current version: **HTML5**.
- Not case-sensitive (e.g., <p> is same as <P>).
- Flexible in syntax; browsers auto-correct small errors.

XHTML (Extensible HyperText Markup Language)

- A **stricter, XML-based** version of HTML.
- Combines **HTML's structure** with **XML's strict rules**.
- Designed for **cleaner, well-formed code**.

XML (Extensible Markup Language)

- A **general-purpose markup language** for **storing and transporting data**.
- Does **not define how data is displayed**, only how it is structured.
- Used in **data exchange** (APIs, databases, etc.).

2.0 Functions and Usage

Language	Function	Common Use
HTML	Display web content	Web pages
XHTML	Display with stricter syntax	Clean web documents
XML	Store & transport data	APIs, configuration files

3.0 Syntax Differences

Feature	HTML	XHTML	XML
Tag Case	Not case-sensitive	Lowercase only	Lowercase only
Closing Tags	Optional in some tags	Mandatory	Mandatory
Attribute Quotes	Optional	Required	Required
Self-Closing Tags	 or 	 only	<tag /> only
Root Element	Not required	Required	Required
Error Tolerance	High	Low	Very Low

4.0 Code Examples

HTML Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Page</title>
</head>
<body>
  <h1>Welcome</h1>
  <p>This is a simple HTML page.</p>
</body>
</html>
```

- Browser still works if a tag is not closed properly.
- Tags like
 don’t need /.

 is an HTML tag that stands for **line break**. It is used to move the text to a new line **without starting a new paragraph**.

Example:

```
<p>Hello<br>World!</p>
```

Output:

Hello
World!

Notes:

- It is an **empty tag** (no closing tag required).
- Use
 when you want text to appear on the next line but still stay within the same block of content.

XHTML Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>XHTML Page</title>
</head>
<body>
  <h1>Welcome</h1>
  <p>This is an XHTML-compliant page.</p>
  <br />
</body>
</html>
```

- All tags are in lowercase.
- All tags are closed properly.
- Quotes are required.

XML Example

Here is the **full XML code** with proper formatting and comments:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Root element -->
<student>

  <!-- Child elements -->
  <name>John Doe</name>
  <email>john@example.com</email>
  <course>IFT 221</course>

</student>
```

Explanation:

- <?xml version="1.0" encoding="UTF-8"?>: XML declaration; always placed at the top.
- <student>: Root tag that wraps all other data.
- <name>, <email>, <course>: Child tags containing individual pieces of data.

- Cannot be viewed as a web page directly.
- Used for **data representation**.
- Strict closing and nesting rules.

5.0 When to Use Each

Language	Use When...
HTML	Creating regular web pages
XHTML	You want well-formed, strict HTML with XML tools
XML	Storing, transferring, or exchanging structured data (e.g., API responses)

6.0 Advantages and Disadvantages

Language	Advantages	Disadvantages
HTML	Flexible, widely supported	Can allow bad coding habits
XHTML	Clean, XML-compatible	Too strict; fails if syntax is wrong
XML	Great for data exchange	Not for direct viewing; verbose

7.0 Practice Exercises

Exercise 1
: Create a valid XHTML snippet that shows your name and favorite course.
Solution:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Student Info</title>
  </head>
  <body>
    <p>Name: John Doe</p>
    <p>Course: IFT 221</p>
```

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

Exercise 2

Q: Fill in the blanks

1. HTML is used to _____ web content.
2. XML is used to _____ data.
3. XHTML requires _____ tag names and _____ tag closing.

Answers:

1. Display
2. Store or transport
3. Lowercase, strict

Exercise 3

Q: Write an XML document that stores information about a book (title, author, year).

Solution:

Here is the **full XML code** with clean formatting and optional comments for clarity:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Root element representing a book -->
<book>

  <!-- Title of the book -->
  <title>Web Technologies</title>

  <!-- Author of the book -->
  <author>Jane Smith</author>

  <!-- Year of publication -->
  <year>2025</year>

</book>
```

Notes:

- This XML structure describes a single book with its title, author, and year.
- It can be extended easily to include more information like publisher, ISBN, or nested elements.

8.0 Summary

Language	Use	Characteristics
HTML	Web page design	Flexible, forgiving
XHTML	Strict HTML	Well-formed, XML-based
XML	Data transport	Not for display, very strict

End of Module 8

MODULE 9: CSS and XSLT for Formatting and Transforming Web Content

1.0 Introduction

What is CSS?

We already introduced **Cascading Style Sheets (CSS)** in Module 5 — it styles HTML content by applying layout, colors, fonts, and spacing.

What is XSLT?

XSLT (Extensible Stylesheet Language Transformations) is a **language for transforming XML documents** into other formats like:

- HTML
- Plain text
- Another XML format

2.0 Purpose and Use Cases

Technology	Purpose	Common Use
CSS	Format and style HTML elements	Website design
XSLT	Convert and format XML data into viewable formats	Data display, reports

3.0 Differences Between CSS and XSLT

Feature	CSS	XSLT
Works With	HTML	XML
Function	Styles and formats content	Transforms content
Output	Styled HTML page	New document (HTML, XML, or text)
Language Type	Declarative	XML-based scripting
Use Case	Web page design	Data-driven content transformation

4.0 Example: Using CSS with HTML

Code Example (Inline CSS)

Here is the **full HTML code** using **inline CSS** as shown in your example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Inline CSS Example</title>
</head>
<body>

  <!-- Paragraph with inline CSS styling -->
  <p style="color: blue; font-size: 18px;">
    Hello, I am learning CSS.
  </p>

</body>
</html>
```

Explanation:

- style="color: blue; font-size: 18px;": This is **inline CSS**, applied directly inside the HTML tag.
- You can change the values (e.g., color, font-size) to style the text differently.

Effect: Blue text, 18px in size.

Code Example (External CSS)

style.css

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

Here's the **full code example using External CSS**:

index.html

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

```
  <h1>Welcome to My Website</h1>
  <p>This heading is styled using external CSS.</p>

</body>
</html>
```

6.0 Tools to View XSLT Output

Platform	Tool
Laptop	Any browser (Chrome, Firefox)
Android	HTML viewer app or Acode preview
iPhone	Textastic or XML viewer app

7.0 When to Use CSS vs XSLT

Situation	Use CSS	Use XSLT
Styling HTML pages	✅ Yes	❌ No
Transforming XML into HTML	❌ No	✅ Yes
Displaying raw XML in user-friendly way	❌ No	✅ Yes
Designing user interface	✅ Yes	❌ No

8.0 Practice Exercises

Exercise 1

Q: Create a paragraph styled with CSS that:

- Is red
- Font size 20px
- Aligned center

Solution:

```
<p style="color: red; font-size: 20px; text-align: center;">
  This is a centered, red paragraph.
</p>
```

Exercise 2

Q: What does XSLT do that CSS cannot?

Answer:

- XSLT can **transform XML** into HTML or other formats.
- CSS only **styles existing HTML**, not transform content.

Exercise 3

Q: Fill in the blanks:

1. CSS is used to _____ HTML pages.
2. XSLT is used to _____ XML documents.
3. CSS cannot be used on _____ files.

Answers:

1. Style
2. Transform
3. XML

9.0 Summary

Concept	Description
CSS	Styles HTML content (colors, layout, font)
XSLT	Converts XML into another display format like HTML
Use Together?	No — CSS is for HTML, XSLT is for XML

End of Module 9

MODULE 10: Interactive Graphics and Multimedia Content on the Web

1.0 Introduction

Modern websites are no longer static. To improve **user experience**, developers use **graphics**, **audio**, **video**, and **interactive features**. These multimedia elements can be embedded directly in HTML5 and enhanced with CSS and JavaScript.

2.0 Definitions

Term	Definition
Graphics	Visual images like logos, icons, and illustrations
Multimedia	Content that uses a combination of text, audio, images, animation, or video
Interactive Content	Content that users can engage with, like slideshows, videos, or maps

3.0 Types of Multimedia on the Web

Type	Description
Images	Static visual content (JPG, PNG, SVG)
Audio	Sound or music (MP3, WAV)
Video	Moving visuals (MP4, WebM)
SVG/Canvas	Vector-based interactive graphics
Animations	Transitions, hover effects, or animated illustrations

4.0 Embedding Images

```
<!-- Basic image tag -->
<img src="logo.png" alt="My Logo" width="200" height="150">
```

Explanation:

- src: File path to the image
- alt: Alternate text if image fails to load (for accessibility)
- width and height: Optional for controlling size

5.0 Embedding Audio

```
<audio controls>
  <source src="audio.mp3" type="audio/mpeg">
  Your browser does not support audio.
</audio>
```

Explanation:

- controls: Adds play/pause buttons
- <source>: Path to the audio file

6.0 Embedding Video

```
<video width="320" height="240" controls>
  <source src="video.mp4" type="video/mp4">
  Your browser does not support video.
</video>
```

Explanation:

- video: HTML5 tag for embedding video
- controls: Provides playback controls
- width/height: Set video size

7.0 Interactive Graphics with SVG

Explanation:

- SVG allows you to draw shapes using code.
- Can be styled with CSS and manipulated with JavaScript.

8.0 Using Canvas for Dynamic Graphics

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
  Your browser does not support the HTML canvas tag.
</canvas>

<script>
  const c = document.getElementById("myCanvas");
  const ctx = c.getContext("2d");
  ctx.fillStyle = "#FF0000";
  ctx.fillRect(20, 20, 150, 50);
</script>
```

Explanation:

- canvas: A blank drawing space
- getContext("2d"): Used to draw shapes, text, or animations
- fillRect(): Draws a red rectangle

9.0 Best Practices

Practice	Why Important
Use optimized file sizes	Improves website load time
Add alt attributes to images	For accessibility
Use modern formats (WebP, MP4)	Better compression and performance
Provide fallback text/media	In case media doesn’t load

10.0 Practice Exercises

Exercise 1

Q: Embed an image school.jpg with width 300px and alternate text "School Building".

Solution:

Here's the **full HTML code** that includes an embedded image with the specified properties:

index.html

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

  <h2>Our School</h2>
  <img src="school.jpg" alt="School Building" width="300">

</body>
</html>
```

How to use:

1. Save the code as index.html.
2. Make sure the image file named school.jpg is in the same folder as your HTML file.
3. Open index.html in your browser to see the image displayed.

Exercise 2

Q: Create a simple audio player that plays lecture.mp3.

Solution:

Here's the **full HTML code** to create a simple audio player for lecture.mp3:

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Audio Player Example</title>
</head>
<body>

  <h2>Lecture Audio</h2>

  <audio controls>
    <source src="lecture.mp3" type="audio/mpeg">
    Your browser does not support the audio tag.
```

```
 </audio>

</body>
</html>
```

How to use:

1. Save the file as index.html.
2. Place lecture.mp3 in the same folder as the HTML file.
3. Open index.html in a web browser to play the audio.

11.0 Summary

Element	Tag	Use
Image	<img>	Embeds pictures
Audio	<audio>	Plays sound
Video	<video>	Plays video clips
SVG	<svg>	Draws scalable shapes
Canvas	<canvas>	Creates dynamic graphics

End of Module 10

MODULE 11: Client-Side Programming Using JavaScript

Overview

In this module, you will learn:

1. The difference between **static** and **dynamic** web content
2. What **client-side** and **server-side** scripting mean
3. What **JavaScript** is and how to use it
4. How multiple technologies (HTML + CSS + JS) combine
5. Introduction to the **World Wide Web Consortium (W3C)**

As always: simplified explanations, examples, and exercises.

1. Static vs Dynamic Web Content

Static Content

- Content that does **not change** unless a human edits the code
- Built with **HTML + CSS only**
- Example: About Us page

Dynamic Content

- **Changes automatically** based on user input or data

- Uses **JavaScript**, or **server-side languages** (like PHP, Python, Django)
- Examples: Login pages, calculators, search results, comment boxes

2. Client-Side vs Server-Side Scripting

Scripting Type	Runs Where?	Languages Used	Examples
Client-Side	In the browser	JavaScript, HTML, CSS	Alerts, form validation, menus
Server-Side	On a web server	PHP, Python (Django)	Login systems, database work

Client = fast response
Server = secure data processing

3. Introduction to JavaScript

What is JavaScript?

JavaScript is a **programming language** that runs inside the web browser.
It makes websites **interactive and dynamic**.

What Can It Do?

- Show pop-up messages (alerts)
- Validate user forms
- Change HTML content dynamically
- Animate content

Example 1: Show an Alert

```
<!DOCTYPE html>
<html>
<head><title>Alert</title></head>
<body>
  <script>
    alert("Welcome to my website!");
  </script>
</body>
</html>
```

What it does: Shows a welcome message as a pop-up

Example 2: Change Content with a Button

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

```
<head><title>JS Button</title></head>
<body>
  <p id="demo">Click the button</p>
  <button onclick="document.getElementById('demo').innerHTML = 'Hello, Chris Odeh!'">Click Me</button>
</body>
</html>
```

What it does: Updates the text when the button is clicked

4. Combination Technologies

HTML + CSS + JavaScript = Powerful Websites

Technology	What It Does
HTML	Structure and content (the skeleton)
CSS	Appearance and design (the clothes)
JS	Logic and interactivity (the brain)

When combined:

- You build professional, responsive, interactive web applications

5. World Wide Web Consortium (W3C)

What is W3C?

The **W3C** is the **World Wide Web Consortium**.

It is the official organization that creates and maintains **web standards** for:

- HTML
- CSS
- JavaScript
- Accessibility (for all users)

Why is W3C Important?

- Ensures your website works across **all devices and browsers**
- Promotes best practices for **safe, fast, and clean** web development

PRACTICAL EXERCISES

Exercise 1: Write an Alert Box

Solution:

```
<script>
```

```
  alert("You are amazing!");
</script>
```

Exercise 2: Use Button to Change Page Content

Solution:

```
<p id="text">Original Text</p>
<button onclick="document.getElementById('text').innerHTML = 'Updated!';">Update Text</button>
```

Exercise 3: Combine HTML + CSS + JS

Solution:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    #btn {
      background-color: green;
      color: white;
      padding: 10px;
    }
  </style>
</head>
<body>

<p id="demo">Click the button</p>
<button id="btn" onclick="document.getElementById('demo').innerHTML='Text changed by JavaScript!'">Click</button>

</body>
</html>
```

END OF MODULE 6

Now you know:

- How dynamic websites work
- The role of JavaScript in web design
- How client/server scripts affect interactivity

- W3C and why standards matter

Final Steps

HTML + CSS + JavaScript: The Trinity of Web Design)

1. Overview: Who Does What?

Technology	Role	Example Use
HTML	Structure – what appears on page	Headings, paragraphs, buttons
CSS	Style – how it looks	Colors, fonts, layout
JS	Behavior – how it responds	Click actions, validations

2. Building a Real Example: Interactive Web Card

Let's build a webpage with:

- **HTML**: A button and a text box
- **CSS**: Styling the button and text box
- **JS**: Button changes the text when clicked

✍ Example: All 3 Combined (with full comments)

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML + CSS + JS Demo</title>

  <!--  CSS: Style the page -->
  <style>
    body {
      font-family: Arial;
      background-color: #f0f8ff;
      text-align: center;
      padding: 50px;
    }

    #box {
      padding: 20px;
      border: 2px solid navy;
      width: 300px;
      margin: auto;
      background-color: white;
```

```
    border-radius: 8px;
    box-shadow: 2px 2px 10px #888;
   }

   button {
    background-color: navy;
    color: white;
    padding: 10px 15px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
   }

   button:hover {
    background-color: darkred;
   }
  </style>
</head>

<body>

  <!--  HTML: Structure -->
  <div id="box">
   <h2>Hello, Visitor!</h2>
   <p id="message">Click the button to personalize this message.</p>
   <button onclick="updateMessage()">Click Me</button>
  </div>

  <!-- ⚙ JS: Behavior -->
  <script>
   function updateMessage() {
    // This function runs when the button is clicked
    document.getElementById("message").innerHTML = "Welcome, Chris Odeh! 🎉";
   }
  </script>
```

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

What's Happening?

Part	Action
HTML	Sets up the text and button
CSS	Styles everything (font, colors, hover effect, shadow)
JavaScript	Adds interactivity: when you click the button, it changes the text

3. Tips for Combining HTML + CSS + JS

A. Always link external CSS and JS files in <head> or at the end of <body>:

```
<link rel="stylesheet" href="styles.css">     <!-- For CSS -->
<script src="script.js"></script>             <!-- For JS -->
```

B. Use id and class in HTML to target elements from CSS and JS:

```
<p id="greeting">Hello!</p>
#greeting { color: blue; }
document.getElementById("greeting").innerHTML = "Hi there!";
```

C. Use JS to:

- Show/hide elements
- Change colors or styles dynamically
- Validate form inputs
- Create carousels, tabs, popups, calculators, games

PART 1: FULL WORKING SITE WITH EMAIL SUBMISSION + THANK-YOU PAGE

FILE 1: index.html (Main Website + Contact Form)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Test</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <!-- Font Awesome Icons -->
```

```
  <script src="https://kit.fontawesome.com/a076d05399.js"
crossorigin="anonymous"></script>

  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 15px;
      background-color: #f9f9f9;
      color: #111;
      transition: all 0.3s ease;
    }

    body.dark-mode {
      background-color: #121212;
      color: #eee;
    }

    nav {
      background-color: #001f3f;
      padding: 12px 0;
      text-align: center;
    }

    nav a {
      color: white;
      margin: 0 20px;
      text-decoration: none;
      font-weight: bold;
      font-size: 16px;
    }

    nav a i {
      margin-right: 8px;
    }

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

```
img {
  max-width: 100%;
  height: auto;
  margin: 15px 0;
  border-radius: 4px;
}

.highlight-section {
  background-color: #e0f7fa;
  padding: 15px;
  border-radius: 8px;
  margin-top: 25px;
}

body.dark-mode .highlight-section {
  background-color: #1a1a1a;
}

ul {
  padding-left: 25px;
}

.contact-form {
  margin-top: 30px;
}

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

.contact-form input,
.contact-form textarea {
  width: 100%;
  padding: 10px;
  margin-bottom: 15px;
  border-radius: 5px;
```

```
  border: 1px solid #ccc;
}

body.dark-mode .contact-form input,
body.dark-mode .contact-form textarea {
  background-color: #333;
  color: #eee;
  border: 1px solid #666;
}

.contact-form button {
  background-color: #001f3f;
  color: white;
  padding: 10px 15px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

.contact-form button:hover {
  background-color: #003366;
}

footer {
  margin-top: 40px;
  background-color: #333;
  color: white;
  text-align: center;
  padding: 12px 0;
  font-size: 14px;
}

.dark-mode-toggle {
  position: fixed;
  top: 10px;
  right: 15px;
  padding: 10px 15px;
  background-color: #444;
  color: white;
```

```
      border: none;
      border-radius: 5px;
      cursor: pointer;
      z-index: 1000;
    }

    @media (min-width: 600px) {
      body {
        max-width: 800px;
        margin: auto;
      }
    }
  </style>
</head>
<body>

  <button class="dark-mode-toggle" onclick="toggleDarkMode()"> Dark Mode</button>

  <nav>
    <a href="#"><i class="fas fa-home"></i>Home</a>
    <a href="#"><i class="fas fa-info-circle"></i>About</a>
    <a href="#"><i class="fas fa-envelope"></i>Contact</a>
  </nav>

  <h1>Welcome to My Test Page</h1>
  <p>This is a simple paragraph to demonstrate basic HTML elements like headings, links, lists, and images.</p>

  <p>Visit this website: <a href="https://www.example.com" target="_blank">Example Site</a></p>

  <img src="image.jpg" alt="Sample Image" width="100">

  <h2>My Favorite Fruits</h2>
  <ul>
    <li>Banana</li>
    <li>Strawberry</li>
    <li>Mango</li>
  </ul>
```

```
  <div class="highlight-section">
    <h2>About This Section</h2>
    <p>This section is grouped using a &lt;div&gt; and styled with a background color to make it stand out.</p>
  </div>

  <!--  Contact Form with Email Submission -->
  <div class="contact-form">
    <h2>Contact Me</h2>
    <form action="https://formsubmit.co/YOUR_EMAIL@example.com" method="POST">
      <!--  Replace with your actual email -->
      <label for="name">Name:</label>
      <input type="text" id="name" name="name" placeholder="Your Name" required>

      <label for="email">Email:</label>
      <input type="email" id="email" name="email" placeholder="Your Email" required>

      <label for="message">Message:</label>
      <textarea id="message" name="message" rows="4" placeholder="Your Message" required></textarea>

      <!--  Disable captcha -->
      <input type="hidden" name="_captcha" value="false">

      <!--  Redirect to thank-you.html -->
      <input type="hidden" name="_next" value="thank-you.html">

      <button type="submit">Send Message</button>
    </form>
  </div>

  <footer>
    &copy; 2025 My Test Page. All rights reserved.
  </footer>

  <script>
    function toggleDarkMode() {
```

```
    document.body.classList.toggle("dark-mode");
  }
 </script>

</body>
</html>
```

FILE 2: thank-you.html (Simple Thank-You Page)

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Thank You</title>
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <style>
  body {
   font-family: Arial, sans-serif;
   background-color: #f0f8ff;
   text-align: center;
   padding: 50px;
  }

  h1 {
   color: green;
  }

  p {
   font-size: 18px;
  }

  a {
   display: inline-block;
   margin-top: 20px;
   background-color: #001f3f;
   color: white;
   padding: 10px 20px;
   border-radius: 5px;
   text-decoration: none;
  }
```

```
    a:hover {
      background-color: #003366;
    }
  </style>
</head>
<body>

  <h1>🎉 Thank You!</h1>
  <p>Your message has been sent successfully. I will get back to you shortly.</p>

  <a href="index.html">⬅ Back to Home</a>

</body>
</html>
```

PART 2: HOW TO MAKE IT WORK — FULL EXPLANATION

1. Email Form Setup with FormSubmit

- Go to https://formsubmit.co
- Enter your email (e.g., **myemail@gmail.com**)
- You'll receive a confirmation email → confirm it once
- Now, use this link in your form:

```
<form action="https://formsubmit.co/myemail@gmail.com" method="POST">
```

2. How to Use the Thank-You Page

- Add this line to the form to redirect after submit:

```
<input type="hidden" name="_next" value="thank-you.html">
```

- This tells FormSubmit to take users to thank-you.html after the message is sent.

3. File Structure

Make sure all files are in the same folder:

```
/my-website-folder/
│
├── index.html
├── thank-you.html
├── image.jpg (optional)
```

Then just open index.html in your browser.

4. Test the Setup

- Open index.html in your browser
- Fill the form and click "Send Message"
- You'll be redirected to thank-you.html
- Check your email inbox — you'll see the message 🎉

MODULE 12: Impact of the World Wide Web on People's Lives Over Time

1.0 Introduction

The **World Wide Web (WWW)** has evolved from a simple document-sharing system into a dynamic, global platform that influences nearly every aspect of modern life — education, business, governance, healthcare, communication, and entertainment.

2.0 Evolution of the Web (Brief Timeline)

Era	Description
1990s – Web 1.0	Static web pages, read-only content, little interaction
2000s – Web 2.0	Interactive applications, user-generated content, social media
2010s – Web 3.0	Intelligent apps, personalization, semantic search
2020s+ – Web 4.0	AI-powered services, voice interaction, IoT integration

3.0 Major Impacts of the Web

3.1 Communication

- **Then**: Letters, fax, or landlines
- **Now**: Instant messaging, email, video calls
- **Example**: WhatsApp, Zoom, Gmail
- **Effect**: Global communication is now **instant, cheap, and multimedia-rich**

3.2 Education

- **Then**: Traditional classroom learning
- **Now**: E-learning platforms (MOOCs, LMS), online degrees
- **Example**: Coursera, Khan Academy, Zoom for virtual classes
- **Effect**: Education is **accessible to all**, even in remote areas

3.3 Business and Commerce

- **Then**: Physical shops, cash payments

- **Now**: E-commerce, online banking, digital payments
- **Example**: Jumia, Amazon, Flutterwave, Paystack
- **Effect**: Businesses can now **sell globally** and customers can **shop 24/7**

3.4 Government and E-Governance

- **Now**: Online tax filing, license renewal, identity verification
- **Example**: National ID portal, INEC voter registration
- **Effect**: Improved **transparency, speed, and convenience**

3.5 Healthcare

- **Now**: Online appointments, telemedicine, digital patient records
- **Example**: Telehealth platforms, online pharmacy services
- **Effect**: **Quicker access** to medical help and better **health monitoring**

3.6 Entertainment and Social Life

- **Then**: TV and radio only
- **Now**: YouTube, Netflix, TikTok, Spotify, Instagram
- **Effect**: The web has **transformed entertainment** into a **personalized, on-demand experience**

3.7 Employment and Remote Work

- **Now**: Freelancing, online job portals, remote work tools
- **Example**: Upwork, Fiverr, LinkedIn, Microsoft Teams
- **Effect**: Workers can now **earn globally without relocating**

4.0 Digital Divide

Despite all these benefits, the web has also:

- **Widened the gap** between connected and unconnected populations
- Made **data privacy and misinformation** major challenges

5.0 Future Trends

Trend	Impact
AI and Chatbots	Smarter web interaction
Voice Search	Easier accessibility
5G and IoT	Faster, connected homes and cities
Blockchain/Web3	Decentralized data and identity management

6.0 Practice Exercises

Exercise 1

Q: List 3 ways the web has changed how we do business.

Solution:

1. Online shopping and payments
2. 24/7 customer support using chatbots
3. Global advertising through web marketing

Exercise 2

Q: Mention 3 web tools used in education and describe their use.

Solution:

1. **Zoom** – Virtual classes
2. **Google Classroom** – Assignments and communication
3. **Khan Academy** – Video lessons and quizzes

Exercise 3

Q: Identify one negative effect of the World Wide Web and explain.

Solution:

- **Misinformation**: People can spread false information quickly, leading to panic or wrong decisions (e.g., fake health tips).

7.0 Summary

Area	Impact
Communication	Instant messaging and video conferencing
Education	Online learning and remote teaching
Commerce	Digital payments and e-commerce
Healthcare	Online consultations and record-keeping
Work	Remote jobs and freelancing
Challenges	Misinformation, privacy, digital divide

✓ **End of Module 12**

Here are **3 highly recommended books** for learning **Web Design using HTML, CSS, and JavaScript**, ideal for beginners to intermediate learners:

1. HTML and CSS: Design and Build Websites
Author: *Jon Duckett*
Why it's great:

- Beginner-friendly with **beautiful visuals** and color-coded syntax.
- Explains HTML and CSS in an **easy-to-understand language**.
- Covers layout, forms, multimedia, and basic web structure.

Best for: Beginners who want a **visual, clear introduction** to HTML and CSS.

2. JavaScript and JQuery: Interactive Front-End Web Development
Author: *Jon Duckett*
Why it's great:

- Companion book to the one above, focusing on **JavaScript and jQuery**.
- Uses the same **visual and accessible style**.
- Great for learning how to **add interactivity** (like animations, dynamic content).

Best for: Beginners to intermediate users who want to **move from static to interactive sites**.

3. Eloquent JavaScript (3rd Edition)
Author: *Marijn Haverbeke*
Why it's great:

- A **deep dive into JavaScript**, starting from fundamentals.
- Includes hands-on exercises and small projects.
- Covers **modern JavaScript (ES6+)** and how to think like a developer.

Best for: Those who want to **master JavaScript more seriously** after basics.

Bonus Tip:
If you're looking for **free online alternatives**, check:

- Mozilla Developer Network (MDN)
- freeCodeCamp