Task 01: Portfolio Website

Learn by Building

Tech Stack Fundamentals Session - CSE 24 Batch

19th December 2025

Abstract

Important: This is a **teaching project**. You'll learn **WHAT to do and WHY**, not just copy code. We intentionally don't give you complete solutions—you learn more by building it yourself!

Contents

Project Philosophy

Copy-pasting code teaches NOTHING. Building from understanding teaches EVERYTHING.

The Copy-Paste Trap

If you just copy-paste the code from this guide:

- You won't understand what you built
- You can't modify it when it breaks

Instead:

- Read the explanation
- Understand the concept
- Type the code yourself (not copy!)
- Make it work (debugging teaches!)
- Modify and experiment

Each step follows this pattern:

1. **What you're learning** — The concept
2. **How it works** — Why this matters
3. **Small code examples** — Shows individual pieces
4. **Your task** — What YOU need to build (not complete code!)
5. **Test it** — How to verify it works

1 Step 0: Setup

1.1 0.1 Create Folder Structure

You need these files:

```
portfolio/
            index.html          # Your HTML (you'll create)
            style.css           # Your CSS (you'll create)
            images/
                profile.jpg     # Your photo (you'll add)
```

Your task:

1. Create a folder named `portfolio`
2. Inside, create empty files: `index.html`, `style.css`
3. Create subfolder: `images`
4. Put your profile photo in `images/profile.jpg`

Setup Complete When

Folder `portfolio/` exists

Files `index.html` and `style.css` exist (empty is fine)

Subfolder `images/` exists

Profile photo saved as `images/profile.jpg`

2 Step 1: Understanding HTML Structure

2.1 1.1 What You're Learning

Every HTML page is like a house:

- **Foundation** = `<!DOCTYPE html>` (must be first)
- **Head** = `<head>` (metadata, not visible)
- **Body** = `<body>` (visible content)

2.2 1.2 The Minimum HTML Page

This is the MINIMUM every page needs:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale
   =1.0">
    <title>Page Title</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Content goes here -->
</body>
</html>
```

2.3 1.3 What Each Line Does

```
<!DOCTYPE html>
        Means: "This is HTML5 format"

<html lang="en">
        Means: "Root element, language is English"

<meta charset="UTF-8">
        Means: "Support all characters (emojis, Arabic, etc.)"

<meta name="viewport" content="width=device-width, initial-scale=1.0
   ">
        Means: "Make mobile-responsive"

<title>My Portfolio</title>
        Means: "This text appears in browser tab"
```

```
15
16 <link rel="stylesheet" href="style.css">
17         Means: "Load styles from style.css file"
```

2.4 1.4 Semantic Tags (Structure Your Page)

Inside <body>, use semantic tags to organize content:

```
1 <nav>...</nav>            <!-- Navigation links -->
2 <header>...</header>      <!-- Top section (profile, title) -->
3 <main>...</main>          <!-- Main content -->
4 <footer>...</footer>      <!-- Bottom section (copyright) -->
```

Why semantic?

- Search engines understand structure
- Screen readers help disabled users
- Cleaner, more readable code
- Standard practice in industry

2.5 1.5 Your Task

What to Build

Create `index.html` with the basic structure:

1. **DO NOT COPY!** Type it yourself to understand
2. Start with `<!DOCTYPE html>` (first line)
3. Add `<html lang="en">` tags
4. Inside `<head>`:
   - `<meta charset="UTF-8">`
   - `<meta name="viewport" ...>`
   - `<title>My Portfolio</title>`
   - `<link rel="stylesheet" href="style.css">`
5. Inside `<body>`:
   - `<nav></nav>` (empty for now)
   - `<header></header>` (empty for now)
   - `<main></main>` (empty for now)
   - `<footer>` with your name
6. Save and open in browser — should show blank page with title in tab

Success Criteria

File opens in browser

Title visible in browser tab

No red errors (F12 console)

3 Step 2: Navigation Menu

3.1 2.1 What You're Learning

Anchor tags (<a>) create links. They navigate to different parts of page using **ID**s.

3.2 2.2 How Links Work

```
<!-- Link -->
<a href="#section-name">Click Me</a>

<!-- Target (must have matching ID) -->
<section id="section-name">
    This content appears when you click link above
</section>
```

The # means: "Find an ID on THIS page"

3.3 2.3 Your Task

What to Build

Inside <nav>, add 5 anchor tag links:

- Link 1: `href="#home"` with text "Home"
- Link 2: `href="#about"` with text "About"
- Link 3: `href="#skills"` with text "Skills"
- Link 4: `href="#projects"` with text "Projects"
- Link 5: `href="#contact"` with text "Contact"

Hint: Each <a> tag is on its own line in <nav>
Example (first link):

```
<a href="#home">Home</a>
```

Your page should show:

Home | About | Skills | Projects | Contact

Test It

Navigation appears at top

All 5 links visible

Links are clickable

4 Step 3: Understanding Lists

4.1 3.1 What You're Learning

Lists organize items. There are two types:

```
<!-- Unordered list (bullets) -->
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>

<!-- Ordered list (numbers) -->
<ol>
    <li>First</li>
    <li>Second</li>
    <li>Third</li>
</ol>
```

4.2 3.2 Why Use Lists?

- Semantically correct (not just paragraphs)
- Search engines understand structure
- CSS can style bullets/numbers
- Screen readers work better

WRONG way:

```
<p>HTML, CSS, JavaScript, React</p>
```

RIGHT way:

```
<ul>
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript</li>
    <li>React</li>
</ul>
```

4.3 3.3 Your Task

What to Build

Create `<main>` with 4 sections:

1. Create `<section id="about">` with:
   - `<h2>About Me</h2>`
   - One paragraph about yourself
2. Create `<section id="skills">` with:
   - `<h2>Skills</h2>`
   - 3 `<ul>` lists (Frontend, Backend, Tools)
   - Each list has 3-4 `<li>` items with YOUR skills
3. Create `<section id="projects">` with:
   - `<h2>Projects</h2>`
   - Just placeholder text for now
4. Create `<section id="contact">` with:
   - `<h2>Contact</h2>`
   - Just placeholder text for now

Example Skills Section:

```
<section id="skills">
    <h2>Skills</h2>

    <h3>Frontend</h3>
    <ul>
        <li>HTML5</li>
        <li>CSS3</li>
        <li>JavaScript</li>
    </ul>

    <h3>Backend</h3>
    <ul>
        <li>Python</li>
        <li>Node.js</li>
    </ul>
</section>
```

Test It

Click navigation links

Page scrolls to sections

All 4 sections appear

5 Step 4: Understanding HTML Forms

5.1 4.1 What You're Learning

Forms let users enter data. Key parts:

```
<form>
    <label for="field-id">Label Text</label>
    <input type="text" id="field-id" name="field-name">
</form>
```

5.2 4.2 Understanding Each Part

```
<form>
        Container for all form elements

<label for="email">
        Describes what input is for
        "for=" connects to input's id

<input type="email" id="email" name="user-email">
        Input field
        type="email" validates email format
        id="email" must match label's for=
        name="user-email" identifies field when submitted
```

5.3 4.3 Form Field Types

```
<input type="text">          <!-- Single line text -->
<input type="email">         <!-- Email with validation -->
<input type="number">        <!-- Numbers only -->
<textarea></textarea>        <!-- Multi-line text -->
<button type="submit">       <!-- Submit button -->
<input type="checkbox">      <!-- Checkbox -->
<input type="radio">         <!-- Radio button -->
```

5.4 4.4 Your Task

What to Build

Add contact form to `<section id="contact">`:
Create a `<form>` with 4 inputs:

- **Name field:**
  - `<label for="name">Your Name:</label>`
  - `<input type="text" id="name" name="name">`
- **Email field:**
  - `<label for="email">Your Email:</label>`
  - `<input type="email" id="email" name="email">`
- **Message field:**
  - `<label for="message">Message:</label>`
  - `<textarea id="message" name="message"></textarea>`
- **Submit button:**
  - `<button type="submit">Send</button>`

CRITICAL: Each input's `id` MUST match label's `for=`

Example:

```
<label for="name">Your Name:</label>
<input type="text" id="name" name="name" placeholder="John Doe">
```

Test It

- Form appears on page
- Can click input fields
- Placeholder text visible
- Button clickable

6 Step 5: Understanding <details> and <summary>

6.1 5.1 What You're Learning

Collapsible sections without JavaScript! Native HTML5 feature.

6.2 5.2 How It Works

```
<details>
    <!-- When collapsed, only shows: -->
    <summary>Click to expand</summary>

    <!-- Hidden content appears when clicked: -->
    <p>This content is hidden by default</p>
    <p>Click summary to show/hide</p>
</details>
```

6.3 5.3 Real Example

```
<details>
    <summary>Education</summary>
    <div>
        <p><strong>University:</strong> University of Moratuwa</p>
        <p><strong>Degree:</strong> B.Sc. Computer Science</p>
        <p><strong>Semester:</strong> 4th</p>
    </div>
</details>

<!-- Multiple details can be open at same time -->
<details>
    <summary>Achievements</summary>
    <ul>
        <li>Winner - Enigma 2025</li>
        <li>Team Lead - SMILE LABS</li>
    </ul>
</details>
```

6.4 5.4 Your Task

What to Build

Update `<section id="about">` with collapsible sections:

1. Keep the opening paragraph
2. Add 3 `<details>` sections:
   - "About Me" — About yourself
   - "Education" — Your education info
   - "Achievements" — Your achievements as `<ul>`
3. Each `<details>` has:
   - `<summary>` with clickable text
   - Content inside `<details>` but outside `<summary>`

Key: Content MUST be inside `<details>` but OUTSIDE `<summary>`

Test It

- [ ] Sections collapse when clicked
- [ ] Sections expand again when clicked
- [ ] No JavaScript needed
- [ ] Multiple sections can be open

7 Step 6: Images and Attributes

7.1 6.1 What You're Learning

Images require proper attributes for accessibility and functionality.

7.2 6.2 How Images Work

```
<img src="path/to/image.jpg" alt="Description">

    src = location of file      alt = text if image doesn't load
```

7.3 6.3 Why Alt Text Matters

- If image doesn't load, alt text shows
- Screen readers read alt text to blind users
- Search engines index alt text for images
- SEO benefit

BAD:

```
<img src="profile.jpg">
```

GOOD:

```
<img src="images/profile.jpg" alt="Himath Jayasinghe's profile photo
    ">
```

7.4 6.4 Your Task

What to Build

Add profile image to <header> section:

1. Inside <header>, add:
   - <img> tag with:
     - src="images/profile.jpg"
     - alt="Your Name's profile photo"
     - class="profile-image" (for CSS later)
   - <h1> with your name
   - <p> with your title (e.g., "Web Developer")

Example:

```
<header>
    <img src="images/profile.jpg" alt="Himath Jayasinghe's profile
   photo" class="profile-image">
    <h1>Himath Jayasinghe</h1>
    <p>Web Developer | Computer Science Student</p>
</header>
```

Test It

Image appears in header

No broken image icon

Name and title visible below image

8 Step 7: Project Cards with `<article>` Tag

8.1 7.1 What You're Learning

`<article>` is semantic tag for self-contained content (like blog post or product card).

8.2 7.2 Structure of Project Card

```
<article class="project-card">
    <h3>Project Title</h3>
    <p>Short description</p>

    <p><strong>Technologies:</strong> HTML, CSS, JavaScript</p>

    <p><strong>Features:</strong></p>
    <ul>
        <li>Feature 1</li>
        <li>Feature 2</li>
    </ul>

    <a href="https://github.com/yourname/project">View on GitHub</a>
</article>
```

8.3 7.3 Why Structure Matters

Organization makes it:

- Easy to style with CSS (apply class to all project-card)
- Easy to duplicate (copy one card, change content)
- Easy to debug (consistent structure)
- Semantically correct (`<article>` for independent content)

8.4 7.4 Your Task

What to Build

Add 3+ projects to `<section id="projects">`:
For each project create:

1. `<article class="project-card">`
2. Inside each article:
   - `<h3>` with project name
   - `<p>` with description
   - `<p>` with technologies used
   - `<ul>` with 3-4 key features
   - `<a>` link to your GitHub (use `target="_blank"`)
3. Add at least 3 different projects

Link Format:

```
<a href="https://github.com/yourname/project-name" target="_blank">
    View Project
</a>
```

Test It

At least 3 projects visible

GitHub links work

Links open in new tab (target="_blank")

9 Step 8: CSS Grid for Layout

9.1 8.1 What You're Learning

CSS Grid arranges items in rows and columns automatically.

9.2 8.2 Grid Basics

```
.container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 20px;
}
```

What each line means:

- `display: grid;` — Enable grid layout
- `grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));`
  - `repeat(auto-fit, ...)` — Repeat as many times as fit
  - `minmax(300px, 1fr)` — Each column min 300px, max fills space
  - Result: On mobile (1 column), tablet (2 columns), desktop (3+ columns)
- `gap: 20px;` — Space between items

9.3 8.3 Your Task

What to Build

In `style.css`, create professional styling:

1. Add basic styling:
   - `body` font, colors, background
   - `nav` styling (buttons, hover effects)
   - `header` gradient background
   - `section` white cards with shadow
2. For skills section:
   - Class `.skills-container` with CSS Grid
   - Make 3 columns that stack on mobile
3. For projects section:
   - Class `.projects-grid` with CSS Grid
   - Each card `.project-card` has border, shadow, hover effect
4. For forms:
   - `input` and `textarea` styling
   - `button` styling with hover effect

IMPORTANT: Don't copy entire CSS file. Build it step-by-step!

Start with basics:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', sans
   -serif;
    background-color: #f5f5f5;
    color: #333;
    line-height: 1.6;
}

nav {
    background-color: #4285F4;
```

```
    padding: 15px;
}

nav a {
    color: white;
    text-decoration: none;
    margin: 0 15px;
}
```

Test It

- Design looks professional
- Colors consistent
- Responsive (test on mobile)
- Grid layouts work
- Hover effects visible

10 Step 9: Test and Submit

10.1 9.1 Self-Test Checklist

Before submitting, verify everything works:

HTML Checklist

- [ ] Page loads in browser
- [ ] Navigation links scroll to sections
- [ ] Profile image visible
- [ ] Skills show as lists
- [ ] Projects display with links
- [ ] Contact form has all fields with labels
- [ ] Details/summary sections collapse and expand
- [ ] No errors in console (F12)

CSS Checklist

- [ ] Page looks professional
- [ ] Colors consistent
- [ ] Responsive (test on mobile size)
- [ ] Grid layouts work
- [ ] Hover effects visible
- [ ] No broken images
- [ ] Text is readable

10.2 9.2 Folder Structure for Submission

Create this exact structure:

```
Task 01/submissions/Your_First_Name_Your_Last_Name/
            index.html
            style.css
            images/
                profile.jpg
```

Examples:

```
Himath_Jayasinghe/
Anusha_Perera/
Kamal_Silva/
```

10.3 9.3 Git Submission Instructions

Follow these steps carefully:

```
# Step 1: Clone repository
git clone https://github.com/HimathX/tech-stack-fundamentals-cse24.git
cd tech-stack-fundamentals-cse24

# Step 2: Create your submission folder
mkdir -p "Task 01/submissions/Your_First_Name_Your_Last_Name"
cd "Task 01/submissions/Your_First_Name_Your_Last_Name"

# Step 3: Copy your files here
# (Copy index.html, style.css, and images/ folder)

# Step 4: Go back to root
cd ../../..

# Step 5: Stage all changes
git add .

# Step 6: Commit with descriptive message
git commit -m "Task 1: Portfolio Website - Your Name

- Semantic HTML structure
- Navigation with anchor links
- Profile section with image
- Collapsible about section
- Skills organized as lists
- 3+ project cards with GitHub links
- Contact form with proper labels
- Responsive CSS Grid layout
- Professional styling"

# Step 7: Push to GitHub
git push
```

10.4 9.4 Verify Submission

After pushing, check on GitHub:

1. Go to https://github.com/HimathX/tech-stack-fundamentals-cse24

2. Navigate to: `Task 01/submissions/Your_Name/`
3. Verify:
    - `index.html` is visible
    - `style.css` is visible
    - `images/` folder exists
    - Your commit message appears in history

Submission Complete When

Folder appears on GitHub

All files visible

Commit message shows

No merge conflicts

11 What You've Learned

11.1 Core Concepts

- **Semantic HTML** — Use proper tags (`<header>`, `<nav>`, etc.)
- **HTML Forms** — Input fields with labels and validation
- **Lists** — `<ul>` for unordered, `<ol>` for ordered
- **Anchor Links** — Navigate using `href="#id"`
- **Interactive Elements** — `<details>` for collapsible content
- **CSS Grid** — Responsive layouts without media queries
- **Image Optimization** — Alt text for accessibility
- **Professional Design** — Colors, spacing, typography

11.2 Why This Matters

You've built a **real portfolio** that you can:

- Use for job applications
- Deploy on GitHub Pages
- Add more projects to continuously
- Modify when you learn new skills
- Show interviewers (demonstrates HTML/CSS mastery)

12 Troubleshooting

12.1 Images Not Showing

Common Issues

- Check path: Should be `images/profile.jpg` (relative path)
- Check file name: Case-sensitive on Mac/Linux
- Check format: JPG, PNG, or WebP
- Solution: Use F12 → Network tab to debug

12.2 Navigation Links Don't Work

Debug Steps

- Check `<section id="about">` has matching ID
- Check link is `<a href="#about">`
- Make sure # is included in href
- Test manually: type #about in URL

12.3 Details Don't Expand/Collapse

- Make sure you have `<details>` AND `<summary>`
- Content must be INSIDE `<details>` but OUTSIDE `<summary>`
- No JavaScript needed — it's native HTML5

12.4 Form Doesn't Look Right

- Add CSS to `input`, `textarea`, `button`
- Set width, padding, border, colors
- Add hover effects for better UX
- Use `<label>` to connect text to inputs