Portfolio building guide

- HTML — structure your content
- CSS — style and layout
- JavaScript — interactivity
- Responsive design — make it work on mobile
- Final touches & deployment

Build Your Own Personal Portfolio Website: Step-by-Step Learning Guide

Structure of the Documentation

The workshop is divided into 8 modules:

1. **Setup and Project Structure - Github Setup**
2. **HTML Foundations**
3. **Building the Header Section**
4. **The About Me Section**
5. **Projects, Posts, and Contact Sections**
6. **Styling with CSS**
7. **Adding Interactivity with JavaScript**
8. **Finishing Touches and Deployment**

Module 1 — Setup and Project Structure + GitHub Setup

Concepts:

- How a website is structured.
- What are HTML, CSS, and JavaScript files.
- Folder organization.

- Introduction to Git and GitHub for version control.

Tasks:

Step 1 — Create Your Local Project Structure

1. Create a new folder for your project.

   Example: `my_portfolio`

2. Inside it, create the following structure:

```
my_portfolio/
│
├── index.html
├── Styles/
│   ├── general.css
│   ├── about.css
│   ├── projects.css
│   ├── contact.css
│   ├── phonescreencss.css
│
├── Images/
│   ├── user.png
│   ├── work1.png
│   ├── background.png
│
├── Files/
│   ├── My CV - en.pdf
│   ├── My CV - fr.pdf
│
└── Scripts/
    ├── main.js
```

Hints:

- Each CSS file will handle a different part of your website.
- The `index.html` file is your main entry point — where your webpage starts.

- The `Scripts/` folder is where you'll write your JavaScript later.

Challenge:

Find out what the `<link>` and `<script>` tags do in HTML. You'll need them later to connect your CSS and JS files.

GitHub Setup — Version Control & Hosting Preparation

Now that you've created your project locally, let's set it up on GitHub to save your progress and prepare for future deployment.

Concepts:

- What Git and GitHub are.
- How to initialize a Git repository.
- How to publish a website on GitHub Pages.

Tasks:

Step 2 — Install Git on Your Computer

If you don't have Git installed:

- Download Git here
- During installation, keep default settings.

You can check if it's installed by opening a terminal (or VS Code terminal) and typing:

```
git --version
```

Step 3 — Initialize Git in Your Project

1. Open your project folder in VS Code.
2. Open the terminal (`Ctrl + ~`) and type:

```
git init
git add .
```

```
git commit -m "Initial commit"
```

Explanation:

- `git init` creates a local Git repository.
- `git add .` adds all files to be tracked.
- `git commit` saves a snapshot of your work.

Step 4 — Create a New Repository on GitHub

1. Go to your GitHub profile.
2. Click **New Repository**.
3. Name it for example: `my_portfolio`.
4. Don't add a README for now (you'll push yours later).
5. Copy the repository URL (it ends with `.git`).

Step 5 — Link Your Local Project to GitHub

In your terminal (inside the project folder), type:

```
git remote add origin https://github.com/yourusername/my_portfolio.git
git branch -M main
git push -u origin main
```

This connects your local folder to the GitHub repo and uploads your files online.

Step 6 — Verify Everything

1. Go to your GitHub profile → `Repositories`.
2. Click your `my_portfolio` repository.
3. You should see your project files online.

Challenge:

- Try editing your `index.html`, committing, and pushing again to GitHub.

```
git add .
git commit -m "Added index.html structure"
git push
```

- Watch how GitHub keeps track of every version — this is how professionals work collaboratively.

Module 2 — HTML Foundations

🧠 Concepts:

- What HTML is.
- The purpose of tags like `<div>`, `<h1>`, `<p>`, `<a>`, `<img>`.
- The `<head>` and `<body>` structure.

Tasks:

1. Open `index.html` and write the basic HTML structure:
   - `<!DOCTYPE html>` or type `!` in vscode
   - `<html>`, `<head>`, `<body>`
2. Inside the `<head>`, add:
   - A `<title>` tag for your website.
   - Links to your CSS files using `<link rel="stylesheet">`.
   - A favicon image.
3. Inside the `<body>`, create a main container `<div id="header">` that will hold your navigation and introduction text.

Hints:

- The `<head>` contains metadata (not visible content).
- The `<body>` contains everything visible on the page.

Module 3 — Building the Header Section

Concepts:

- Navigation bars (`<nav>` , `<ul>` , `<li>` , `<a>`).
- Structuring text with `<h1>` , `<p>` .
- Linking to internal sections using anchors (`href="#about"`).

Tasks:

1. Inside `#header` , create:
    - A `<nav>` containing:
        - A logo or title (`<h1>`).
        - A list of navigation links (`<ul><li><a href="#..."></a></li></ul>`).
    - A container `<div>` for your intro text (`Hi, I'm [Your Name]`).
2. Add social icons (GitHub, LinkedIn, etc.) using `<a>` tags.
3. Create buttons that allow users to download your CVs.

Hints:

- To add icons, use **Font Awesome** (include their `<script>` in the `<head>`).
- Use `<a href="Files/My CV - en.pdf" download>` to make a download link.

Module 4 — The About Me Section

Concepts:

- Layout design using `<div>` and CSS grid/flexbox.
- Using images (`<img>` tag).
- Creating tab-like behavior with JS (Skills / Experience / Education).

Tasks:

1. Add a new section `<div id="about">` .
2. Inside it, create two columns:
    - One column with your image (`user.png`).
    - One column with a paragraph describing yourself.

3. Add a small menu for "Skills", "Experience", and "Education".
4. Write short sample texts inside each.

Hints:

- Use `<div class="tablinks">` for the clickable titles.
- You'll later use JS to switch between the tabs.

Try to write your own short bio

Module 5 — Projects, Posts, and Contact Sections

Projects

Create a `<div id="projects">` section:

- Add a few project "cards", each with:
  - An image.
  - A short description.
  - A link to the GitHub repo.

Hint: Use a `<div class="worklist">` to hold your projects.

Posts

Add a `<div id="posts">` section:

- Each post will have a title, description, and link.
- Use icons for consistency.

Contact

Create `<div id="contact">`:

- Add your email with a mailto link.
- Add a contact form with:
  - Name
  - Email
  - Message

- Submit button

Hint: You'll use Google Sheets or a script later to collect form submissions.

Add a footer (`<div class="copyright">`) with your name and copyright notice.

Module 6 — Styling with CSS

Concepts:

- CSS selectors and syntax.
- Classes vs IDs.
- Box model (margin, padding, border).
- Colors, fonts, layout design.

Tasks:

1. Open `general.css` .
   - Define your main colors and font family.
   - Style the header and navigation.
2. Create consistent spacing using margins and paddings.
3. Style buttons (`.btn` , `.btn2`).
4. Add hover effects for interactivity.

Hints:

- You can use Google Fonts (for example: `Poppins` or `Montserrat`).
- Keep a color palette with 2–3 main colors for simplicity.

Challenge:

Add a background image to your header and apply an opacity + blur effect using CSS:

```
#header {
  background: url('../Images/background.png') center/cover no-repeat;
  filter: blur(1px);
  opacity: 0.9;
```

```
}
```

Module 7 — Adding Interactivity with JavaScript

Concepts:

- DOM manipulation.
- Functions and event listeners.
- Smooth scrolling, tabs, and menus.

Tasks:

1. Open `Scripts/main.js` (or inline script in HTML).
2. Add code to:
    - Open and close the mobile side menu.
    - Switch tabs in the “About” section.
    - Handle form submissions using `fetch()`.
    - Show a “scroll to top” button when the user scrolls down.

Hints:

- Use `document.getElementById()` and `document.getElementsByClassName()` to access HTML elements.
- Add event listeners for clicks and scrolls.
- Use `window.scrollTo({ top: 0, behavior: 'smooth' })` for smooth scrolling.

Module 8 — Finishing Touches and Deployment

Concepts:

- Responsiveness.
- Hosting & deployment.

Tasks:

1. Add responsive CSS in `phonescreencss.css` using media queries:

```
@media (max-width: 768px) {
  nav ul { display: none; }
  .fa-bars { display: block; }
}
```

2. Test your website on both desktop and mobile from the inspect window.
3. commit and push your files.
4. Enable **GitHub Pages** to publish your portfolio. you can also use N**etlify** or V**ercel.**

Next Steps

Now that you've built your first portfolio:

- Keep improving it, add animations, **blog posts**, or new projects.
- Learn frameworks like **React** or **Tailwind CSS** later.
- Explore hosting options like **Vercel**, **Netlify**, or **Firebase**.