Lab Assignment 1: Building an Interactive Profile Card

TECNOLOGIA SETUBAL
SETÚBAL POLYTECHNIC UNIVERSITY

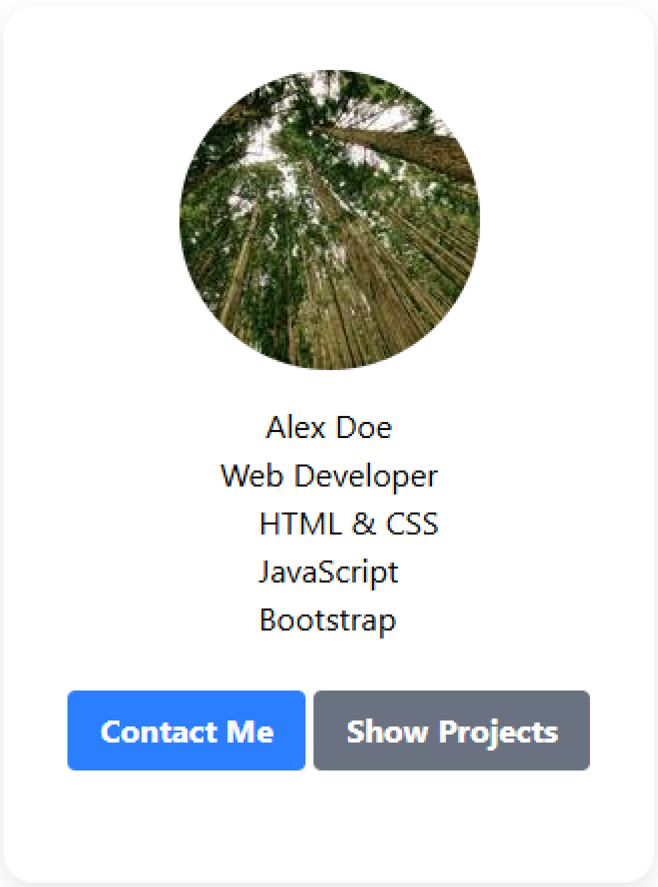
Course: Programação Avançada para a Internet (Mestrado em Engenharia de Software) **Objective:** To apply foundational knowledge of HTML, CSS, and JavaScript by building a dynamic, single-page component.

1. Introduction

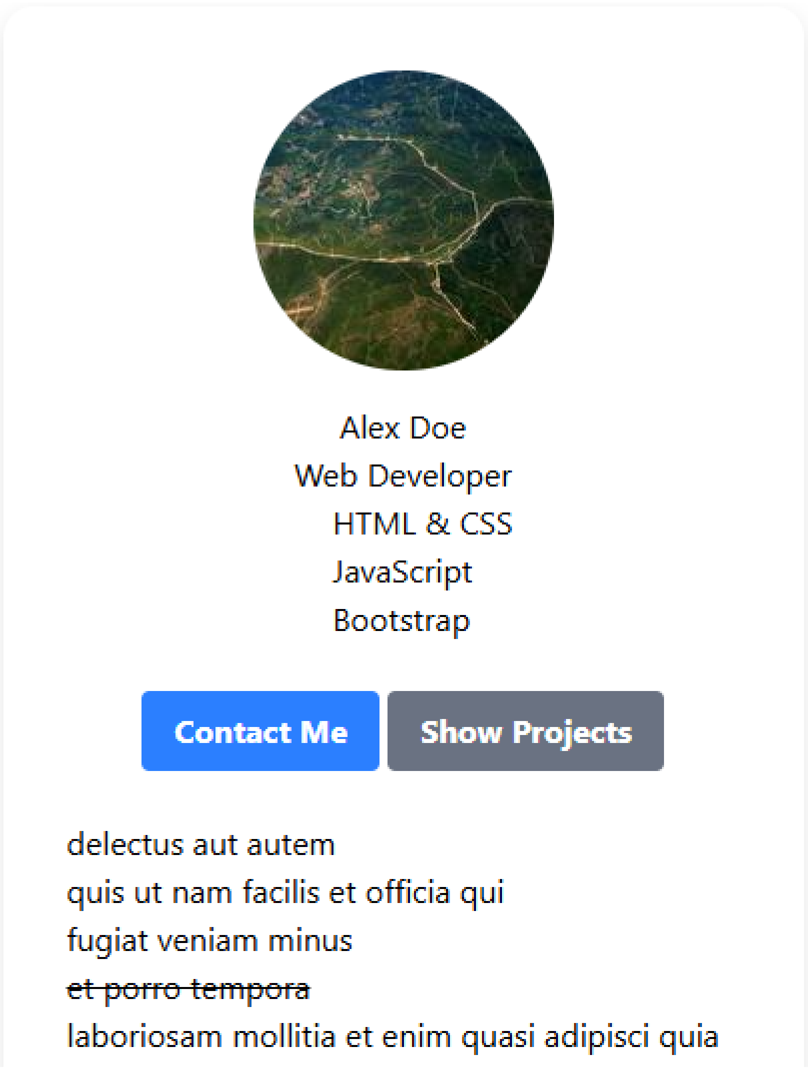
The goal of this exercise is to put theory into practice by building an "Interactive Profile Card." This project requires you to structure content with HTML, style it with CSS, and add interactivity with JavaScript.

You will create a **single standalone `index.html` file** that contains all your HTML, CSS (in a `<style>` tag), and JavaScript (in a `<script>` tag).

The resulting "Interactive Profile Card" should look similar to the figures below and should be placed in the center of the page:

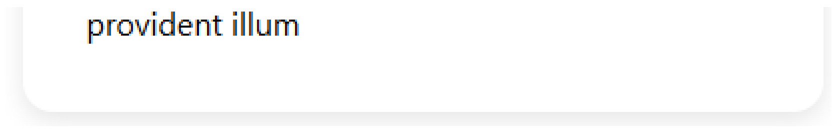


Alex Doe
Web Developer
HTML & CSS
JavaScript
Bootstrap



Alex Doe
Web Developer
HTML & CSS
JavaScript
Bootstrap

delectus aut autem
quis ut nam facilis et officia qui
fugiat veniam minus
~~et porro tempora~~
laboriosam mollitia et enim quasi adipisci quia


provident illum

2. Getting Started: Setting up a CSS Framework (Choose One)

Before you start writing code, you need to include a CSS framework. This allows you to use pre-made styles, a common practice in modern web development.

Choose **one** of the two options below and add the corresponding line of code to the `<head>` section of your `index.html` file.

Option A: Bootstrap

Bootstrap provides pre-styled components like buttons and cards. To use it, add this `<link>` tag inside your `<head>`:

```
<!-- Add this line to your <head> to include Bootstrap CSS -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.8/dist/css/bootstrap.min.css"
rel="stylesheet">
```

(Note: Bootstrap also has a JavaScript file for interactive components like dropdowns, but it is not needed for this lab.)

You can check the Bootstrap Documentation here.

Option B: Tailwind CSS

Tailwind provides utility classes to build custom designs directly in your HTML. To use it for this lab, add this `<script>` tag inside your `<head>`:

```
<!-- Add this line to your <head> to include Tailwind CSS -->
<script src="https://cdn.jsdelivr.net/npm/@tailwindcss/browser@4"></script>
```

You can check the Tailwind CSS Documentation here.

---

3. Project Brief & Tasks

Follow the steps below to create your interactive profile card.

Part 1: HTML Structure

Create the skeleton of your profile card. The `<body>` should contain a `<main>` element that holds the card, and the card itself should be an `<article>` tag.

Required Elements inside the `<article>`:

1. An `<img>` for a profile picture (e.g., `<img src="https://picsum.photos/150" alt="Profile Picture">`).
2. An `<h2>` for a name (e.g., "Alex Doe").
3. A `<p>` for a job title (e.g., "Web Developer").
4. A `<section>` for skills, containing a `<ul>` with at least three `<li>` skill items.
5. A `<footer>` containing two buttons:
   - A `<button>` with the ID `contactBtn` and text "Contact Me".
   - A `<button>` with the ID `projectsBtn` and text "Show Projects".
6. An empty `<div>` with the ID `projectsContainer`. JavaScript will add content here later.

Part 2: CSS Styling

Bring your card to life. Add all your styles within a `<style>` tag in the `<head>` of your HTML document.

Styling Requirements:

1. **Card Layout:**
   - Give the main card (`<article>`) a `max-width` so it doesn't stretch too wide on large screens.
   - Center the card horizontally and vertically on the page.
   - Add a `box-shadow` and `border-radius` to make it look like a modern UI card.
   - Use `padding` inside the card to give the content some space.
2. **Flexbox:**
   - Use `display: flex;` and related properties to arrange the content inside the card (e.g., to center the image and text).
3. **Framework Integration:**
   - Find the "Contact Me" button you created in the HTML.
   - If you chose **Bootstrap**, add the classes `btn btn-primary` to it.
   - If you chose **Tailwind CSS**, add classes like `bg-blue-500 text-white font-bold py-2 px-4 rounded` to it.

Part 3: JavaScript Interactivity

Add the dynamic functionality. Place all your JavaScript code inside a `<script>` tag just before the closing `</body>` tag.

Functionality Requirements:

1. **Contact Button:**
   - Add a `click` event listener to the "Contact Me" button (`#contactBtn`).
   - When clicked, it should show an `alert()` with a simple message like "Contact details coming soon!".
2. **Show Projects Button:**
   - Add a `click` event listener to the "Show Projects" button (`#projectsBtn`).
   - When clicked, it should fetch data from this URL: `https://jsonplaceholder.typicode.com/users/1/todos`.
   - After the data arrives, clear any old content from the `#projectsContainer`.
   - Loop through the **first 5 items** of the fetched data. For each item:
     1. Create a new `<p>` element.
     2. Set its text content to the item's `title`.

3. If the item's `completed` property is `true`, add the class name `completed` to the `<p>` element (e.g., `newParagraph.classList.add('completed')`).
4. Append the new `<p>` element to the `#projectsContainer`.

- Finally, define the `.completed` class in your `<style>` tag to give it a distinct look (e.g., `text-decoration: line-through;`).