WDD 231

Individual Website Project Description

Overview

This NEW individual website project is a **comprehensive assessment** of each student's proficiency in achieving the course learning outcomes. Students are tasked with creating a three page website using a contemporary approach with HTML, CSS, and JavaScript. The goal is to develop a dynamic and responsive website that integrates data sources, while adhering to accessibility, usability, and development standards.

> "The desire to create is one of the deepest yearnings of the human soul. No matter our talents, education, or backgrounds, you each have an inherent wish to create something that did not exist before. Everyone can create. You don't need money, position, or influence in order to create something of substance or beauty. Creation brings satisfaction and fulfillment. We develop ourselves and others when you take unorganized matter into our hands and mold it into something of beauty."
>
> [Happiness, Your Heritage](#) – Dieter F. Uchtdorf

Project Requirements

This project will be of your own subject, design, and development. You may not expand upon nor submit the chamber of commerce, learning project. You may not consult nor work with any other individual. The content and subject is driven from your own proposed topic and the site should be complete without any placeholders.

> This is a web course intended to help you learn how to construct dynamic and responsive websites using the core web technologies of contemporary HTML, CSS, and JavaScript. Third party templates, frameworks, and libraries are **NOT** allowed, like TailwindCSS, Bootstrap, Foundation, etc. Pages built from site builder software or drag-and-drop tools or that are based on existing sites are not allowed and will lead to no credit on the term project.

Location and Hosting

Your project must be stored in its own subfolder within your GitHub **wdd231** repository. Something like "final" or "finalproject"

Update the "Final" link in the nav bar on your week 1 page to point to your week 6 final project.

File and Folder Naming

All files and folders adhere to the course [naming conventions](#).
— lowercase, no spaces, standard, meaningful (semantic), etc.

HTML Standards

Structure the pages with valid, semantic HTML markup. This includes the proper use of header, nav, main, and footer elements.

Each page should meet the baseline [development standards](#).

CSS Standards

Use valid CSS that does not contain unused and unnecessarily duplicated declarations and rules.

Design Principles and Layout

Design Principles: The design, in all views, must demonstrate a consistent look and feel and adhere to web design principles of proximity, alignment, repetition, contrast, and appropriate white-space.

Responsive Navigation: Small screen links expand when a hamburger icon is clicked and larger screens display links horizontally.

Wayfinding: Use wayfinding with the site's main navigation links.

Responsive Layout: The layout of each page must be responsive to mobile (320px) (portrait and landscape) and larger screen views with no horizontal scrolling.

Page Weight: Make sure each page size is below the 500kB of total data transfer from an empty cache.

Accessibility: The layout and design must support accessibility.

Usability: The site must be usable and supports a positive user experience.

Content

Page Requirement: Three (3) total pages are required. This includes the landing (home) page named "index.html".

Purpose: The content must be cohesive and must be relevant to the purpose of the site as outlined in your website plan.

Branding with Favicon: Integrate a favicon on each page that is consistent with the site's branding, logo, or overall design.

Semantic HTML Titles: Use distinct and descriptive `title` tags to accurately reflect each page's content.

SEO-Friendly Descriptions: Implement unique `meta name="description"` tags that are concise and relevant to each page, optimizing them for search engines.

Author Attribution: Include a `meta name="author"` tag on each page.

Social Sharing Optimization: Implement necessary social media metadata, such as Open Graph tags, to control how content is displayed when shared on social platforms.

Images

All images must be **optimized** for the web and use intrinsic aspect ratios.

Use a **lazy loading** technique to support progressive design and to increase page performance.

HTML Form

Use an HTML **form** that meets the standards presented in the course.

Display the form data on a form action page.

(This form action page does **not** count toward your site 3 page requirement.)

JavaScript Functionality and Components

Your website must incorporate dynamic features and content using JavaScript.

The following functionality and components are required:

Data Fetching: Retrieve data from an external source or a local JSON file.

- Use the Fetch API to make asynchronous requests *(demonstrated in the video)*.
- Incorporate `try...catch` blocks for robust error handling of asynchronous operations *(demonstrated in the video)*.
- Handle the response appropriately (e.g., parsing JSON).

Dynamic Content Generation:

- Dynamically generate and display at least fifteen (15) items from your data source.
- For each item, display at least four (4) distinct data properties/values.

Local Storage: Implement local storage to persist data client-side (e.g., user preferences, application state).

Modal Dialogs:

- Implement at least one modal dialog structure for user interaction (e.g., displaying detailed information, confirming actions).

- Ensure the modal is accessible and follows best practices for user experience.

DOM Manipulation and Event Handling: Implement JavaScript to interact with the Document Object Model (DOM). This should include:

- Selecting elements using appropriate methods (e.g., `querySelector`, `querySelectorAll`).

- Modifying element properties, style, and/or content.

- Attaching event listeners to elements and responding appropriately to events (e.g., `click`, `submit`, `change`).

Array Methods: Utilize at least one appropriate array method (e.g., `map`, `filter`, `reduce`, `forEach`) to process data efficiently.

Template Literals: Employ template literals for string construction, especially when dealing with dynamic content or multi-line strings.

ES Modules: Structure your JavaScript code using ES Modules to demonstrate proper code organization and modularity.

There are many public and free APIs that you can use for your project. Check the [Web Frontend Development Resources](#) page for some examples.

Professionalism

Proofread Carefully: Fix all spelling and grammar mistakes.

Acknowledge Sources: If you used external content, add an "Attributions" link to your footer.

> The images and verbiage may be referenced from other sources which will need to be cited in a resource attributions page, which must be linked from the footer of the landing (home) page. This attributions page does not need to be styled.

Video Demonstration and Reflection

Create a focused, brief video that captures your screen as you demonstrate how you met certain JavaScript requirements. You do not need to talk about anything else except to demonstrate where and how you met the following requirements.

Video Specifications Requirements

What to Capture: Use your camera showing your **face** as you record your **screen**.

Length: Record a **3-5 minute** video. Focus on the required demonstrations and nothing else.

Location: Upload your completed video to Youtube, Loom, or an equivalent service. Make sure the video is public so that the graders can view it.

Recording Tools: Use the video capture and editing tool of your choice.

> Some video recording tools that include free options are:
> - ClipChamp
> - Screencastify
> - Loom
> - OBS Studio
> - QuickTime Player (macOS)

Link your video in the **footer of each page** of your site so it can be easily found by the grading team.

Video Content Requirements

How you used your **API/Data** integration and demonstrate the output.

How you used an **asynchronous functionality** with a try block.

Testing

Use this page audit tool. The audit will be used by the graders in your assessment.

Test your site in multiple browsers and devices.

Use the browser's DevTools to check for JavaScript runtime errors.

Use DevTools CSS Overview to check your color contrast.

Generate the DevTools Lighthouse report and run diagnostics for accessibility, best practices, and SEO (Search Engine Optimization) in both the mobile and desktop views.

Be sure that that your site can be opened by the graders.

Make sure your video is available to the graders.

Submission

Return to **Canvas** and submit:

1. Your project's GitHub Pages (github.io) enabled URL
2. Insure that there is a link to the video in the site footer!