TASK

Capstone Project - Build Your Developer Portfolio

Visit our website

Introduction

WELCOME TO THE BUILD A COMPLETE WEBSITE CAPSTONE PROJECT!

In this task, you will be consolidating the knowledge you have learnt and applying it to a real-world application. You'll be tasked with a set of criteria to meet, and the rest is up to you! Remember, it is worth putting some extra time and effort into this project as it can become part of your developer portfolio.

DEVELOPER PORTFOLIO

Developers who have extraordinary skills are those who find ways to apply their newfound skills from the get-go. A **developer portfolio** is a collection of online creations you have made which allow you to demonstrate your skills rather than just telling people about them. It is a way of bringing your CV to life and introducing yourself to the world. As you learn more skills and put these into practice, each project that you complete will become more efficient and eye-catching.

This Capstone Project offers opportunities to create projects for your developer portfolio, allowing you to walk away from this course with a certificate and, more importantly, with a headstart into your career!

THE TASK AT HAND

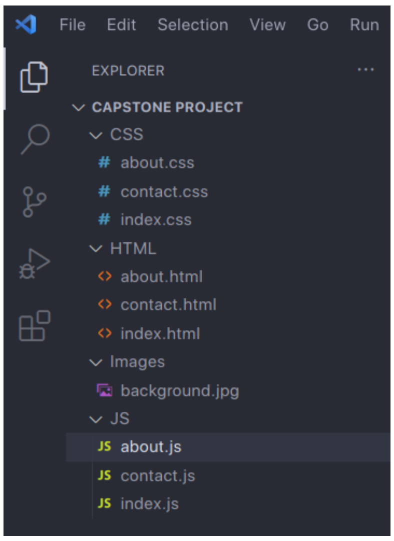
In this project, you will focus on JavaScript on top of HTML and CSS. JavaScript is a programming language that adds background functionality and animations on all the beautiful websites you see. You will be tasked with JavaScript requirements to enhance a basic site with certain functionalities. Feel free to add any functionality to your website during this process. Remember to use this project to showcase the skills you have acquired so far!

BEFORE YOU BEGIN

A key focus of this project will be ensuring that your JavaScript, CSS, and HTML are correct, well-formatted, and readable. In this regard, make sure that you do the following before submitting your work:

1. Ensure to use a good directory structure to organise your web application. Keep your entire project in a directory with a meaningful name. Within this directory, make sure you have separate directories for any images, CSS, and

JavaScript you use. Below is an example of what an organised folder structure would look like:



```
CAPSTONE PROJECT
  CSS
    about.css
    contact.css
    index.css
  HTML
    about.html
    contact.html
    index.html
  Images
    background.jpg
  JS
    about.js
    contact.js
    index.js
```

2. You can also use tools like **Chrome's Developer Console** to troubleshoot your code.

3. Make sure to refactor your code and include comments throughout your JavaScript, HTML and CSS code. During the process, ask yourself the following questions:
   - Are you using functions to improve the modularity and readability of your code?
   - Have you given your variables and functions meaningful names that adhere to the **Google Style Guide** naming conventions?

Capstone Project

Imagine you are a freelance web developer and would like to create a developer portfolio page to showcase your skills to potential clients.

First, you will build a simple webpage with information about yourself, testimonials, services you provide, and your portfolio. Then, you will create a restaurant ordering system to showcase your JavaScript skills. Finally, you will link this project in the portfolio section of your developer portfolio page. Let's get building!

Step 1: Setting up your HTML structure

1. Create an HTML file ('**index.html**') for your developer portfolio page. It should include the following sections:
    - **Menu:** Include navigational links to direct users to different sections within your developer portfolio.
    - **About:** Display your name, profile image and why a client should choose you.
    - **Services:** Create a list of the services you provide.
    - **Testimonials:** Two testimonials related to the services you listed.
    - **Portfolio:** Construct a place to link to your projects.

   The information in these sections can be fictitious except for the developer portfolio section that you will populate with the JavaScript project you create in Step 3. You may explore some **freelancer portfolios** on the web to get inspiration for your webpage.

2. Use semantic HTML elements such as `<header>`, `<section>`, `<main>`, etc. to structure your webpage better.

Step 2: Styling with CSS

1. Create a separate CSS file ('**style.css**') to style your index.html.
2. Style the elements in your HTML file, with a colour and font of your choice. Ensure your profile image is an appropriate size, and the layout is responsive to different screen sizes.
3. **Optional:** Experiment with **Bootstrap** to style your web page instead of CSS.

Step 3: JavaScript project

Build a JavaScript application that interacts with **The Meal DB** API to take and complete orders of the 'chef's favourite meals'. The application will utilise the `prompt()` and `alert()` functions for user interactions, the fetch API for consuming the Meal DB API, and `sessionStorage` to store order details.

You will be utilising the API method to '**filter by main ingredient**'. Explore the list of **ingredients** available, and observe that the '`strIngredient`' encompasses items such as Pork, Lemon, Mint, and more. In the '**filter by main ingredient**' URL, experiment by substituting "chicken_breast" with "beef" and then with "Fanta", for instance, to observe the URL modification and variations in returned values. This exercise demonstrates that the API can yield null values.

1. Taking Orders:
   - 1.1. Use the `prompt()` function to take the users' order by asking for their *main ingredient*.
   - 1.2. Call the API to retrieve a list of the chef's favourite meals based on the main ingredient provided.
   - 1.3. Make use of the `random()` function to randomly select a chef's favourite meal and set it as the order.
   - 1.4. The order will be defined by its description, order number and the completion status, which can either be completed or incomplete.
   - 1.5. The main ingredient is only used to retrieve the description, i.e. the `strMeal` field, from the API and needn't be stored as part of the order details.
   - 1.6. It's important to highlight that when the user enters the main ingredient, you have to convert the characters to lowercase and replace each space with an underscore when calling the Meal DB API. For example, 'Garlic Powder' should be 'garlic_power'. The resulting URL for the call with be: `https://www.themealdb.com/api/json/v1/1/filter.php?i=garlic_powder`.

1.7. If the API returns null, provide a suitable response, and prompt the user for another ingredient. A URL example that would return a `{"meals":null}` value is:
`https://www.themealdb.com/api/json/v1/1/filter.php?i=fanta`.
Tip: you may find it useful to **make use of recursion** in this case.

2. Storing Orders:
   - 2.1. Store the order details in `sessionStorage`.
   - 2.2. Ensure that each new order has a unique order number.
   - 2.3. Make sure that your session storage contains the last generated order number (which is the number of items in storage session `+ 1`), to avoid having to loop over all the orders when storing them.
3. Displaying and Completing Orders:
   - 3.1. Use the `prompt()` function to display all the incomplete orders stored in session storage. Present the order number and description only.
   - 3.2. Prompt the user to enter the order number to mark as complete or, alternatively, enter zero for not completing an order.
   - 3.3. Once an order's completion status is updated, the status should be committed to the session storage.
   - 3.4. Display an appropriate response message if the order number doesn't exist.
   - 3.5. **Hint**: You can use the `filter()` function to select only incomplete orders.
4. Storage Format:
   - 4.1. Store the collection of orders as a single value in a JSON array.
   - 4.2. The last order number should be stored as a separate value.
   - 4.3. **Hint**: You can retrieve and return values from sessionStorage using the `getItem()` method.
5. Portfolio Integration:
   - 5.1. After completing your JavaScript project, add a link for potential clients to download this project in the portfolio section of your web page.

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we’ve done a good job?

Click here to share your thoughts anonymously.