Learn To Code

Capstone 2: Enjoy the Outdoors

JavaScript Fundamentals

Description

In this project, you will build a website that helps a user find things to do to enjoy the great outdoors. The site specializes in finding national parks to enjoy and mountains to climb. You will use what you know about HTML, CSS, and JavaScript to complete this project. You will also need to flex your research skills to solve any blockers you run into.

Basic Requirements

The website should include the following pages in order to be considered complete:

- **A Homepage** - This page should highlight our simple organization and contain links to two pages: a National Parks search page and a Mountains information page!
- **A National Parks Search Page** - The National Parks search page provides a user interface that allows the user to search for the park that is just right for them. Data comes from a file named nationalParks.js provided below.
- **Mountains Information Page** - The mountains information page provides a user interface that allows the user to explore the details of 48 different mountains. Data comes from a file named mountains.js provided below.

NOTE There are ample opportunities in this project to keep you busy and stretch your skills. Just make sure the basic requirements above are met before you decide to tackle any optional features!!

Implementation Details

Below you will find the requirements for implementation and details about the **National Parks search page**, **Mountains Information page**, **stretch goals**, and general hints.

National Parks Search Page

This page will allow the user to search for national parks that they might be interested in. The parks are provided to you in an array located inside `nationalParkData.js` provided below. Spend some time examining this file.

Ultimately, the user would like to have two ways to search for a national park:

- By location
- By park type

Search by Location - This option allows users to select the state/territory from a dropdown. Values for the state/territory dropdown will be provided in an array defined in the file named `locationData.js`. You will know a park matches the location by comparing it to the park's "State" property**.**

NOTE: Search by Location is the most important search option and should be the highest priority to complete.

Search by Park Type - This option allows users to select a park type from a dropdown. Values for the park type dropdown will be provided in an array defined in the file named `parkTypeData.js`. You will know a park matches the description by checking to see if the park's "LocationName" property *contains* the description**.**

One of the challenges will be how you decide to present the user with two search options populated with the appropriate values. Do you use radio buttons to select the search type? Do you use a dropdown with the search types as options?

National Parks Search Page - Stretch Goals

Challenge yourself with some of the stretch goals below. These stretch goals should be treated as the lowest priority tasks

- Provide a **View All National Parks** option.
- Some, but not all National Parks, contain a **"Visit"** property that contains a URL to a page about the park. Display the URL in a hyperlink along with park details so the user can click on it and visit the park's page from your list. Make sure to open the visited page in a different tab/window!

Implementation Hints

- You will need to include the three scripts (`locationData.js`, `parkTypeData.js`, and `nationalParkData.js`) into your mountain search page in addition to any custom script(s) you write. Take a few minutes to examine each file.
- **Suggestion:** Get the "Search by Location" feature working first.
- When you are working on the "Search by Park Type", you need to make sure the two strings have the same casing when you search. The easiest way to do this is to

use the String objects `.toLowerCase()` or `.toUpperCase()` to make the strings the same case.

Mountains Information Page

This page will provide a dropdown list of the 48 mountains defined in an array in the `mountainData.js` file. Make sure to spend some time examining this file. When the user selects a mountain, your page will display information about that mountain, including:

- Mountain Name
- Description
- Elevation
- Any other information you find interesting about the mountain

NOTE: *This page will NOT allow the user to search mountains using any type of filter.*

Mountain Information Page Stretch Goals

Challenge yourself with some of the stretch goals below. These stretch goals should be treated as the lowest priority tasks

- Display the image of the mountain along with the mountain information. The mountain object contains the name of an image file and the images are included in the files distributed to you.

- Impress the user by displaying the sunrise and sunset time "today" for any mountain along with the other mountain data.

You can "fetch" the sunrise/sunset times using the JavaScript Fetch API. You will learn more about it in the coming weeks, but in the meantime, you can simply the following function:

```
// function that can "fetch" the sunrise/sunset times
async function getSunsetForMountain(lat, lng){
   let response = await fetch(
     `https://api.sunrise-sunset.org/json?lat=${lat}&lng=${lng}&date=today`);
   let data = await response.json();
   return data;
}
```

And you would call the function using code like this (where 44.320686 is a longitude and -71.291742 is a latitude):

```
// Fetch the sunset/sunrise times for a specific mountain
getSunsetForMountain(44.320686, -71.291742).then(data => {
   console.log(data.results)
});
```

The value returned in the `data` variable is the object returned from the API call. For example, it might contain:

```
{
  "results": {"sunrise":"5:18:00 AM",
              "sunset":"6:43:52 PM",
              "solar_noon":"12:00:56PM",
              "day_length":"13:25:52",
              "civil_twilight_begin":"4:49:23 AM",
              "civil_twilight_end":"7:12:29 PM",
              "nautical_twilight_begin":"4:13:02 AM",
              "nautical_twilight_end":"7:48:50 PM",
              "astronomical_twilight_begin":"3:34:23 AM",
              "astronomical_twilight_end":"8:27:29 PM"},
   "status":"OK"
}
```

NOTE(S): the lat and lng values are included in the mountain data. The times returned are in UTC and are not adjusted for local summer variations. Label the output as UTC time when you display them.

What Makes A Good Capstone Project?

You Should:

- Build a consistent look-and-feel throughout the site with a working navigation
    - This can be accomplished by your theming choices and by use of a consistent header, a navbar, and (if desired) a footer across your pages
- Implement at least the required pages
- Have a responsive user interface
- Build a public repo for your code, use an appropriate branching strategy, and have a commit history with meaningful commit messages
- It must contain an informative README that:
    - describes your project
    - includes images of ***each*** of the site's pages
    - describes/shows one interesting piece of JavaScript that you wrote

Implement best practices:

- Have a good directory structure (ex: `css`, `images` and `scripts` folders)
- Use good file naming conventions (ex: lowercase file names with no spaces)
- Have well-formatted HTML, CSS and JavaScript (indentions, blank lines, etc)
- Use good names for your HTML elements and JavaScript variables/functions
- Use HTML, CSS and JavaScript comments effectively
- Make sure there are no JavaScript errors at run time (check the Console tab in the browser frequently)

Class Demonstrations

Each student will be given 10 minutes to demonstrate their site to the class on "project demonstration day". During this time, you will:

- Show off your website and the pages within it
- Describe/show one interesting piece of HTML/CSS/Bootstrap you wrote
- Answer questions from the audience if time permits