Project Summary

This project aims to give you the opportunity to put all of the skills you have learned into one project to build your own custom travel app. Due to the nature of this course, it is very JavaScript heavy, but it is still expected you create clean and appealing HTML/CSS. You will also be targeting the DOM, working with objects, and retrieving data from 3 APIs in which one of those is reliant on another to work. Finally, this is all going to be done in a Webpack environment, using an express server, and wrapped up with service workers.

For this project, refactor and test as much as possible while you are building. You should figure for every piece of functionality you add, you will likely spend just as much time testing and refactoring your code. If it takes you 5 hours to figure out the logic, it should likely take you another 5 hours determining that you wrote the best code possible. As your skills improve, this process will feel more natural. Make sure to remove any debugging code from your final submission.

The minimum requirements ask a fair amount from you, but the final app is quite simple. A roadmap to expand on the application and make it uniquely your own is provided.

What You Will Build:

You will be building a travel application. It's common to pull basic data from an API, but many applications don't just pull the weather, they pull in multiple types of data, from different sources and occasionally one API will be required to get data from another API.

The project will include a simple form where you enter the location you are traveling to and the date you are leaving. If the trip is within a week, you will get the current weather forecast. If the trip is in the future, you will get a predicted forecast. The OpenWeather API is fantastic but it doesn't let you get future data for free and it's not that flexible with what information you enter; we are going to use the Weatherbit API for you to see how another API accomplishes the same goals. Weatherbit API has one problem, it only takes in coordinates for weather data -- it's that specific. So, we'll need to get those coordinates from the Geonames API. Once we have all of this data, we'll want to display an image of the location entered; for this, we will be using the Pixabay API.

This may not sound like a lot, but there is a fair amount of moving pieces that rely on each other to work. You'll need to plan out the logic of what you are trying to accomplish before you begin developing. There are a lot of paths you can take, and what you choose to display and how you display it is somewhat flexible. It is highly recommended that after you meet the minimum requirements in the rubric, you continue debugging the UX and improve the project.

What will I learn?

At this point, you have learned all of the technical skills necessary to complete this project. You will likely stumble along the way and find that there are some pieces you didn’t encounter in the past projects. Remember, if you get stuck, you should always look things up. Sometimes just articulating the problem renders a solution.

The following are just some of the questions that you’ll experience along the way:

- What’s the ideal workflow?
- How many files do I need?
- How do I convert one project into another?
- Should I redo the HTML/CSS first or go straight to the javascript?
- How many JavaScript functions do I need?
- Should my function be this many lines of code?
- Is readability or performance more important?

There’s no single correct answer to each question. While building this project, working with peers, and getting feedback from the project reviewer -- you will naturally develop your own answers to these questions!

NEXT