I originally wrote my solution in vanilla JavaScript with HTML and CSS as this is the environment I am most comfortable in, and the initial solution took me a couple of hours to complete. I used MDN and W3schools as references for my JavaScript solution to make it a front-end only application, as my initial idea was to use a back end to access the API using get/post requests and the express and ejs library.

I also decided to stretch myself further and teach myself the fundamentals of a React framework, which took me about a half a day to integrate my original code into it. I thought it would make for a good learning experience and gave me a better understanding of the React framework and the use of JSX! I used the create-react-app setup provided by React to initialise my solution, and maintained that structure. While I think I could have turned the widget generation into a separate component, I found this option the simplest as it resembled a standard JavaScript file more closely.

I found that my react solution created cleaner code, because I was unable to simply manipulate the DOM like I did in my JavaScript solution, instead having to change the state and render the result.

I chose to import the Bootstrap library to create a streamlined look to the widget creator, as I wanted to follow the example image as close as possible. I also built on my own custom CSS on top of it. I did not choose to include any other libraries to keep the code as vanilla as possible.

I chose to use minimal error handling, as the app is pretty small and I felt the only major error that could be thrown would be when handling and accessing data from OpenWeatherMap’s API. The catch clause in the Fetch method was the only error handling I opted to use.

I felt pretty proud of myself for stretching my skills to learn enough React to get a fully functional app. I think the test over all was pretty fair, and didn’t present me with any real challenges when using vanilla Javascript – the challenge for me was to teach myself React and create an implementation.

Assumptions I made for the code:

1. The User must press a button to create the widget. I decided against having the program listen in on each action and checking conditions each time before creating the widget.
2. I decided not to preselect any radios, which does leave the user with the option to choose none. It doesn’t result in an error, but it does bring up a widget that asks the user to please select a temperature measurement and wind option.
3. For the sake of conciseness, I decided to only include four potential weather images. The API documentation showed that there were multiple different conditions with sub-conditions that could be returned, so I decided to stick with images for Sun, Rain, Clouds and a default Overcast for all other weather.
4. I also chose not to change the image based on the current time.
5. Since an API key was required, I used to .env file to hide my API key, because it’s best practice not to include keys in a public repository. Because of this, you’ll need to put in your own API key for the service.
6. All the code is in the App.js file, as I chose not to divide it up into components.
7. I left the app in a development state, and decided not the compile it for production.
8. Although the requirements asked that the widget editor must be a ‘form’, I assumed it was a form to be filled out and not a HTML form element. Form elements make get/post requests to the backend, and the task asked for a front-end only solution.