UDACITY

Return to Classroom

Weather Journal App

REVIEW

CODE REVIEW 8

HISTORY

Meets Specifications

Congratulations, Kudos for such brilliant work.

👏 Your project meets all the rubrics.

Below are some appreciation comments for you:

- I am really impressed with the hard work and efforts which you devoted for this project.
- The app looks great!!
- Keep up the awesome work and problem solving approach which you followed.
- Learning is a continuous and comprehensive process, keep it going.
- Implementation done for promise chaining and response mapping is awesome.

Refer asynchronous functions and promise chaining for more knowledge and future learnings.

There are lot more challenges to encounter. Good luck with your upcoming project and keep rocking.

⭐ Special appreciation for such great implementations and hard work. To me it

seems to be the work of the great JAVASCRIPT Developer.

HAVE A HAPPY LEARNING THANK YOU!

So this was my feedback and no one is perfect, I would really appreciate you giving your feedback on my review 😄

Good luck! Happy learning👍 Stay

Stay safe and healthy. Together we will defeat corona.

Project Environment Setup

Node and Express should be installed on the local machine. The project file `server.js` should require `express()`, and should create an instance of their app using express.

The Express app instance should be pointed to the project folder with .html, .css, and .js files.

Hey, this rubric requires installation of `Node` and `Express` dependencies through command line. We have to create instance of our application up and running by making use of `express()` calls. Express is also needed to load the static content of website. All this is perfectly achieved in current implementation. Great work!

Suggestions

Here we installed dependencies by mentioning each of them in `npm install` command. It seems easy since there are only three dependencies needed. If we move towards large number of dependencies, this becomes a hectic task.

We make use of file `package.json` in such cases. All the dependencies are mentioned there and now by only using command npm install, everything works smoothly. Refer below process to create this file:

- `npm init`
- `npm install --save cors express body-parser`

Please refer package.json for more details.

File is now created. Please create such file in upcoming projects and future implementation. Thanks.

The 'cors' package should be installed in the project from the command line, required in the project file `server.js`, and the instance of the app should be setup to use `cors()`.

The `body-parser` package should be installed and included in the project.

Hey, rubric here requires couple of specifications that need to be fulfilled. We need to install `cors` and `body-parser` dependencies and instantiate for the same in file `server.js`. We are really impressed that you achieved this. We have marked one suggestion as well below, please refer that. Thanks

Meets Expectations

- We need to install `cors` dependency through command line. Also we need to instantiate it in file `server.js`. It is needed for making cross-origin request. Great work pal!

- We need to install `body-parser` dependency through command line. Also we need to instantiate it in the file `server.js`. It is needed for parsing request to endpoints in form acceptable. This is also achieved as per the requirements.

Suggestions

As marked in the above rubric that we could have also added these dependencies in file `package.json` as well. This reduces the overhead of installing each one of them separately. Just one command and all will work as per the requirements.

Command to add dependency in file `package.json` is:

1. npm install --save express
2. npm install --save cors
3. npm install --save body-parser

Next time only command needed will be `npm install`. Thanks.

Local server should be running and producing feedback to the Command Line through a working callback function.

Hey, rubric here requires couple of specification that need to be fulfilled. We have to setup server and log the feedback on command line. We are really impressed that you achieved it in this submission. We have explained the rest of details below:

Meets Expectations

- We need to setup server using express dependency and using method `app.listen`. We need to specify some port and a callback function which will produce feedback on command line. This implementation is done as per the specifications. Great work pal!

Overall kudos for the brilliant work done.

Create API credentials on OpenWeatherMap.com

Hey, requirement of rubric here was to create credentials on `OpenWeatherMap.com`. These credentials will serve as API_KEY. We are really impressed that you achieved all of it in one go. Please refer below for more details:

Meets Expectations

- We need to create credentials on `OpenWeatherMap.com`. These credentials will help for successful API calls and fetch the response.
- We need to place these credentials at the top of file `app.js`. This is required as per the standards all the constants should be placed on the top of file. This helps in easy maintenance.

- We need to append directive `&units=imperial` or `&units=metric` for fahrenheit or celsius respectively. This reduces the boilerplate code and with the use of 2 3 words we can achieve temperature conversion in desired units. This is achieved perfectly in the implementation. Please refer imperial units for more details.

Overall kudos for such brilliant work here!

APIs and Routes

There should be a JavaScript Object named `projectData` initiated in the file `server.js` to act as the app API endpoint.

Hey, rubric here requires couple of specifications that need to be fulfilled. We need to initialise a empty `projectData` object and map the necessary data received from POST calls. Our task here basically was to initialise the empty `projectData` object. We are really impressed that you achieved it. Great work.

Meets Expectations

- Initialisation of empty `projectData` object is as per the standards and specifications. JSON format is used in place and proper care is taken to in the implementation of this part. Great work pal!

Overall kudos for the brilliant work done. Keep it up pal!

The personal API Key for OpenWeatherMap API is saved in a named `const` variable.

The API Key variable is passed as a parameter to `fetch()` .

Data is successfully returned from the external API.

Hey, rubric here requires couple of specifications that need to be fulfilled. We need to create a full fledged URL and pass it in inside fetch method and retrieve successful response. We are really impressed that you made successful API calls and use of const keyword. Please refer below for more details:

Meets Expectations

- We need to form a proper request URL, which contains URL+ZIPCODE+API_KEY. This needs to be passed to fetch method to retrieve API response.
- This response is then filtered and relevant data is passed to POST endpoint where we map our response

to empty `projectData` object.

- We need to make use of const keyword for our API Key variables. This makes the value immutable in the course of program flow. Using for credentials related data is awesome. Please refer about const for more details.

Overall kudos for such brilliant work. Thanks.

There should be a GET route setup on the server side with the first argument as a string naming the route, and the second argument a callback function to return the JS object created at the top of server code.

Hey, rubric here requires a single specification that need to be met. We need to setup a GET endpoint which is called at the time of rendering response on UI. We are really impressed with the way you achieved this in first attempt. I have explained it below:

Meets Expectations

- We need to add a GET endpoint in the code. This need to be achieved using callback functions. Inside these functions we need to implement the logic which return `projectData` as the response which contains mapped response.

Overall kudos for brilliant work. Thanks.

There should be an asynchronous function to fetch the data from the app endpoint

Hey, rubric requires here one of the specification that need to be met. We need to setup a asynchronous method to successfully call GET endpoint to fetch `projectData` object which contains API response mapped in it. We are really impressed that you achieved this in first attempt. I have explained it below:

Meets Expectations

- We need to make use of async keyword and syntax to setup asynchronous function calls. We need to call fetch method here with GET endpoint defined on server side. This will fetch response from mapped in `projectData` object and render it on UI.

Overall kudos for such brilliant work pal!

You should be able to add an entry to the project endpoint using a POST route setup on the server side and executed on the client side as an asynchronous function.

The client side function should take two arguments, the URL to make a POST to, and an object holding the data to POST.

The server side function should create a new entry in the apps endpoint (the named JS object) consisting of

The server side function should create a new entry in the apps endpoint (the named js object) consisting of the data received from the client side POST.

Hey, rubric requires couple of specification that need to be fulfilled. We need to setup a POST endpoint using callback functions. Inside this functions we need to implement our mapping logic i.e. map values in

`projectData` object. Once this is done, we will use this `projectData` object for rendering response on UI. This is working as per the expectations.

Meets Expectations

- We need to implement asynchronous calling by making use of promise chaining where we will pass the API response to POST endpoint on server side. This is where we pass our data from client to server.
- We need to specify the first parameter as String for server endpoint and callback function where we map the response in `projectData` object. Mapping of response in `projectData` object is needed so it can be returned with values present in it at the time of GET calls.

Overall great work on all the specifications. Keep it up!

Dynamic UI

The `input` element with the `placeholder` property set to "enter zip code here" should have an `id` of `zip`.

The `textarea` included in project HTML should have an `id` of `feelings`.

The button included in project HTML should have an `id` of `generate`.

This rubric is considered with declaring ids and placeholders on UI for input fields. These input fields contains data which we need for API calls.

Meets Expectations

- Implementation contains all the valid ids and placeholder in place. Feelings, ZipValues and Generate button have there id defined in the UI code. Good work!

The div with the `id`, `entryHolder` should have three child divs with the ids:

- `date`
- `temp`
- `content`

Hey, rubric here requires couple of specifications that need to be met. We need to have three divs setup for response. These response parameter will contains date, temperature and feeling we enter. We are really

response. These response parameter will contains date, temperature and feeling we enter. We are really impressed that you have achieved all of it in one attempt. We have explained below:

Meets Expectations

- We need to create div with id date for rendering current date values.
- We need to create div with id temp for rendering temperature values received from API endpoint.
- We need to create div with id content for rendering feelings which we enter on UI.

Overall kudos for brilliant work done here!

Adds an event listener to an existing HTML button from DOM using Vanilla JS.

In the file `app.js`, the element with the `id` of `generate` should have an `addEventListener()` method called on it, with `click` as the first parameter, and a named callback function as the second parameter.

Hey, rubric here requires couple of specifications that need to be met. We need to setup an event listener functionality that will trigger our promise chaining logic for API calls, POST and GET endpoint. We are really impressed that you achieved this in first attempt. We have explained all of them below:

Meets Expectations

- We need to attach event listeners to submit button so that on click callback functions are invoked. These callbacks functions invokes the promise chaining logic for API calls. All is implemented as per the requirements.

Overall kudos for the brilliant work done!

Sets the properties of existing HTML elements from the DOM using Vanilla JavaScript.

Included in the async function to retrieve that app's data on the client side, existing DOM elements should have their `innerHTML` properties dynamically set according to data returned by the app route.

Hey, rubric here requires couple of specification that need to be met. We need to implement a method named updateUI, where GET endpoint are consumed and response is rendered on UI. We are really impressed that you achieved all in one attempt. We have explained all of them below:

Meets Expectations

- We have a method setup for response rendering where GET endpoint is consumed. This GET endpoint returns the `projectData` which contains response mapped in it.
- We need to make use of `innerHTML` tags for mapping our response on UI elements.
- JSON format of `projectData` is used for rendering response.

Overall kudos for the brilliant work done here. Thanks.

DOWNLOAD PROJECT

8 CODE REVIEW COMMENTS ›

RETURN TO PATH