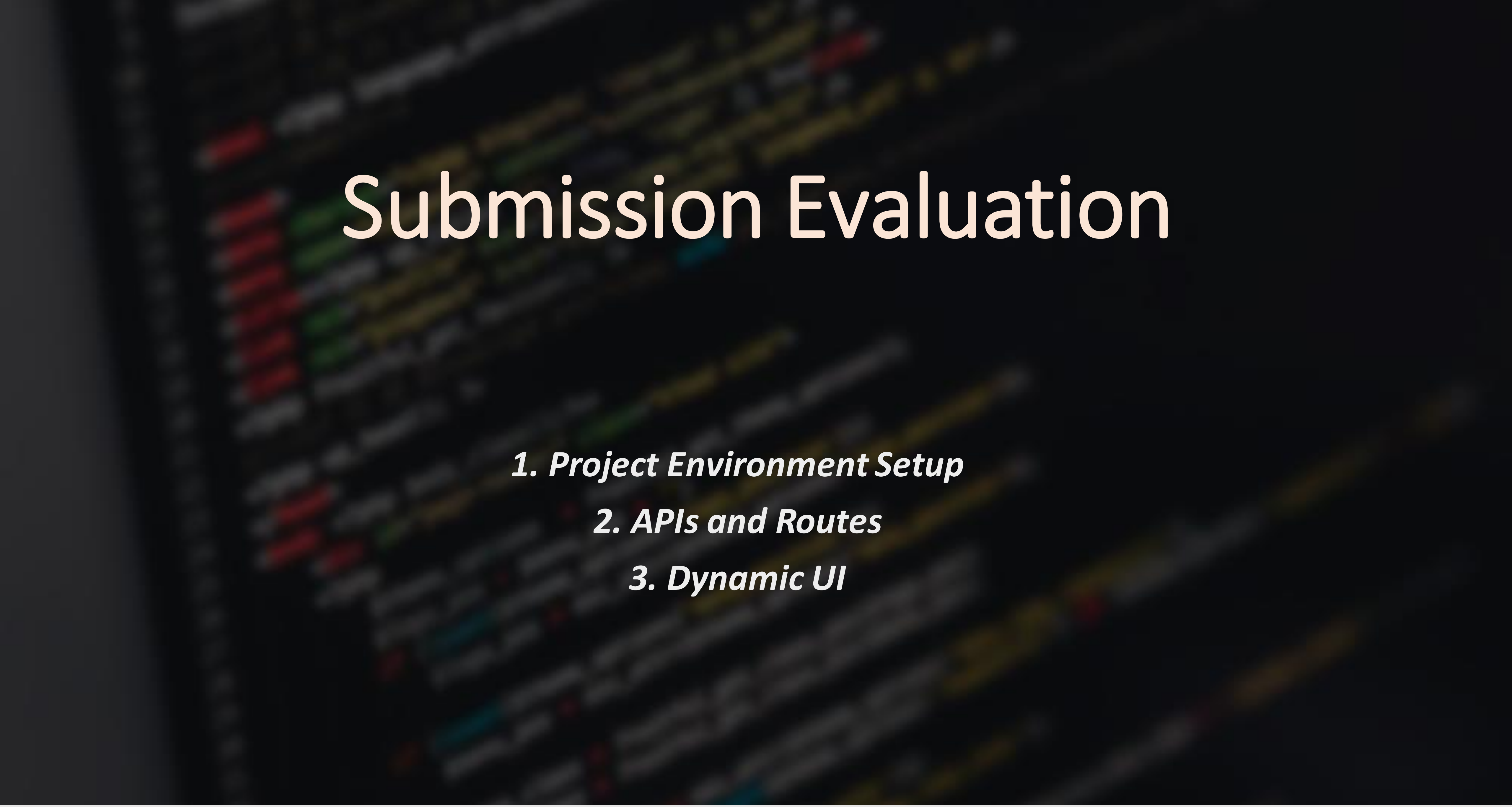
Udacity Front End Nanodegree

Third project: Weather Journal App

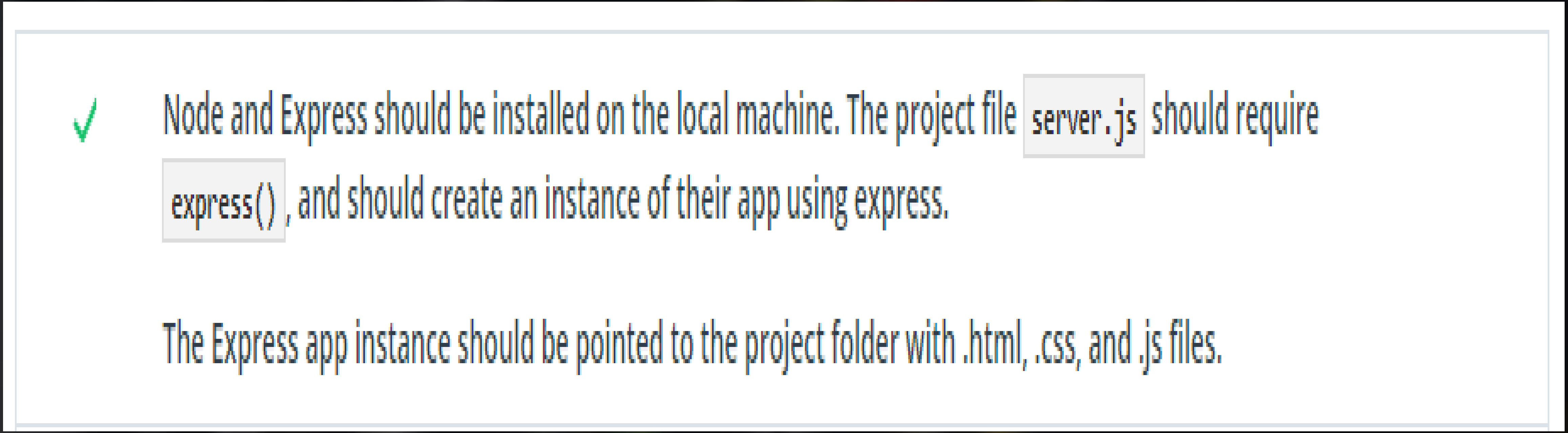
https://www.youtube.com/channel/UC51OIiCEJ1hAvJlpky5cuSA



Submission Evaluation

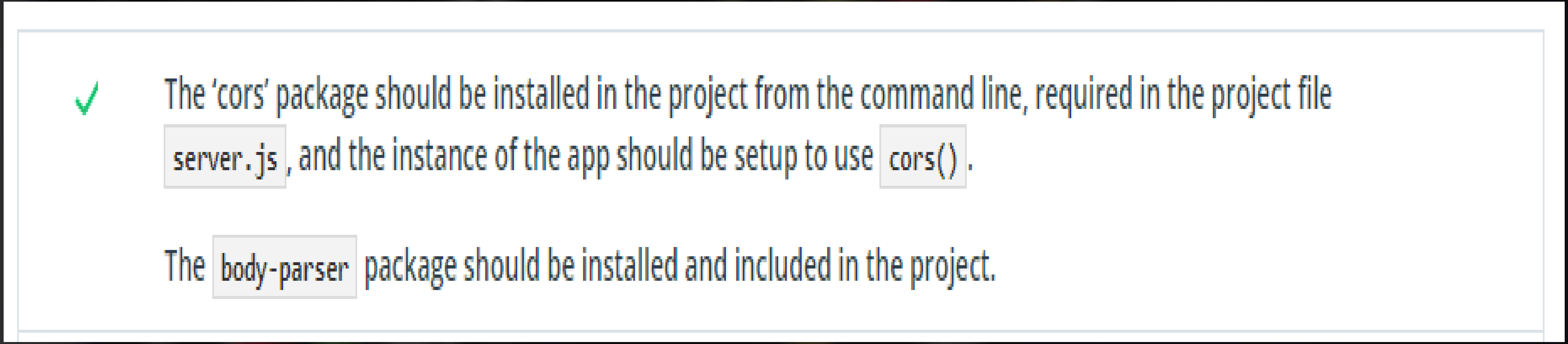
1. ***Project Environment Setup***
2. ***APIs and Routes***
3. ***Dynamic UI***

1. 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.

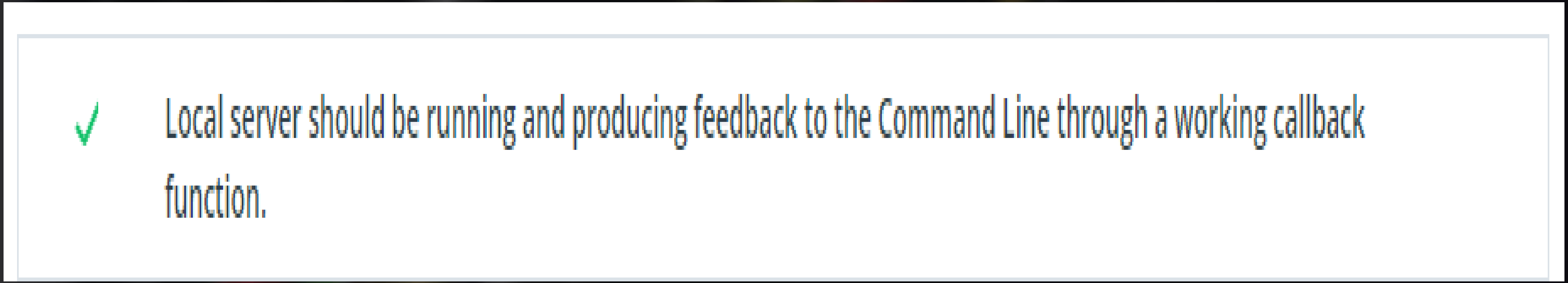
- **This is the first specification in the project environment setup section. In this specification you should install the node and express into your project. Specifically in the server.js file then you need to create an instance of the express and point it to the project folder.**



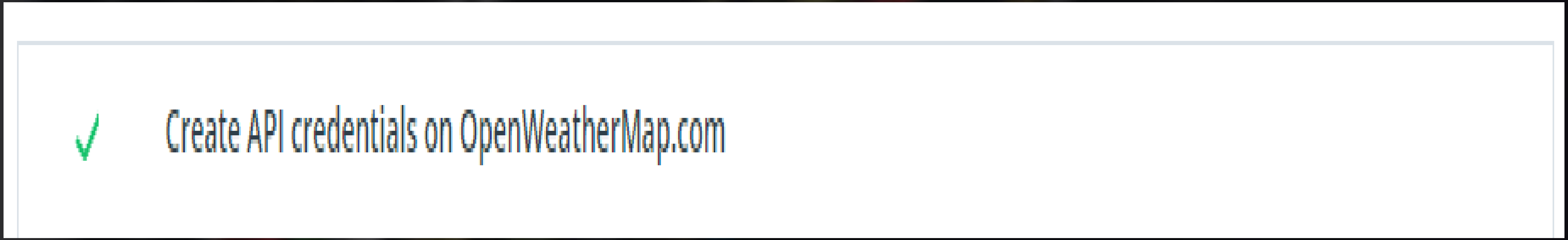
✓ The 'cors' package should be installed in the project from the command line, required in the project file `server.js`, and the instance of the app should be setup to use `cors()`.

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

- **This is the second specification in the project environment setup section. In this specification you should install the cors and the body-parser packages into your project. And the instance we created from the express should use the cors package as well.**


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

- This is the third specification in the project environment setup section. In this specification all you need to is to do is to create your local server and add some sort of indication that your server is up and running. This is usually done by using console.log() and writing some sort of message telling your audience that the project is running on this specific port.

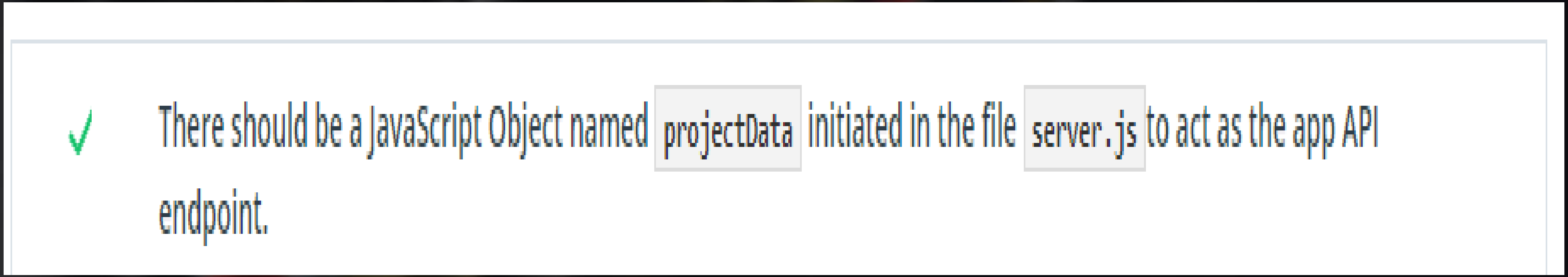


✓ Create API credentials on OpenWeatherMap.com

- This is the forth and final specification in the project environment setup section. In this specification all you need to do is adding the OpenWeatherMap.com credentials to the app.js file.

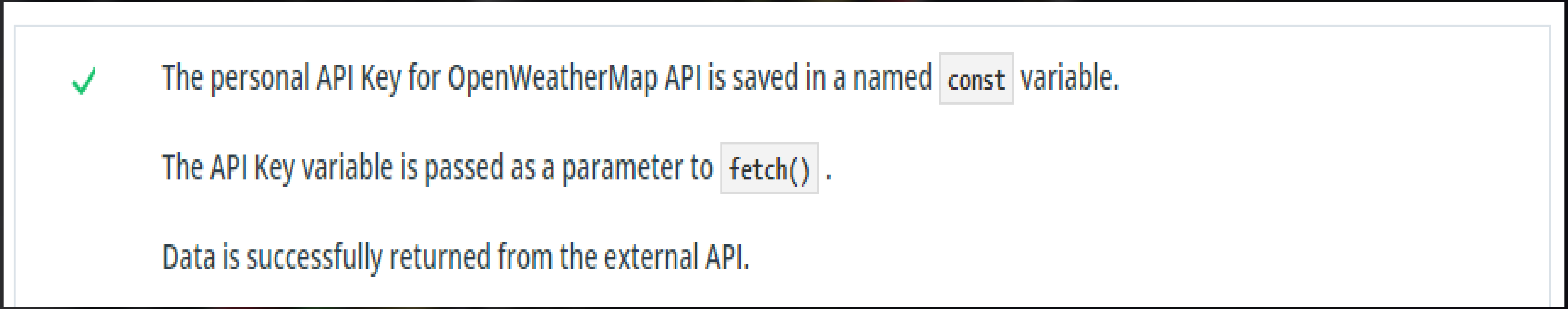
2. APIs and Routes

https://www.youtube.com/channel/UC51OIiCEJ1hAvJlpky5cuSA



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

- **This is the first specification in the *APIs and Routes* section. In this specification you need to make sure that you have a javascript object named projectData that will act as your routes endpoint.**

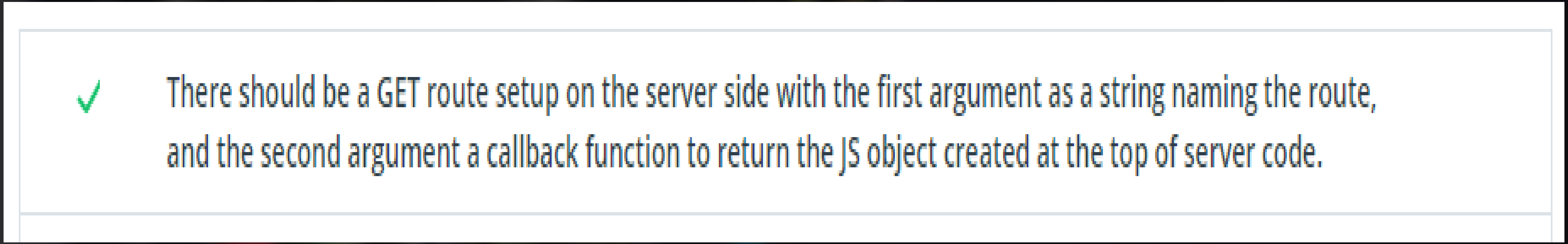


✓ The personal API Key for OpenWeatherMap API is saved in a named `const` variable.

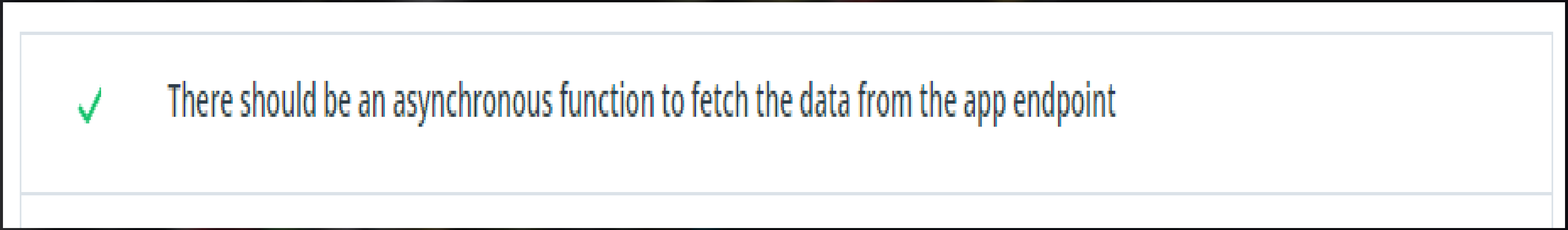
The API Key variable is passed as a parameter to `fetch()` .

Data is successfully returned from the external API.

- **This is the second specification in the *APIs and Routes* section. In this specification you need to have your credentials stored in a const variables, the variable that's storing your API key passed to the fetch() and finally that your data is successfully returning from the API.**

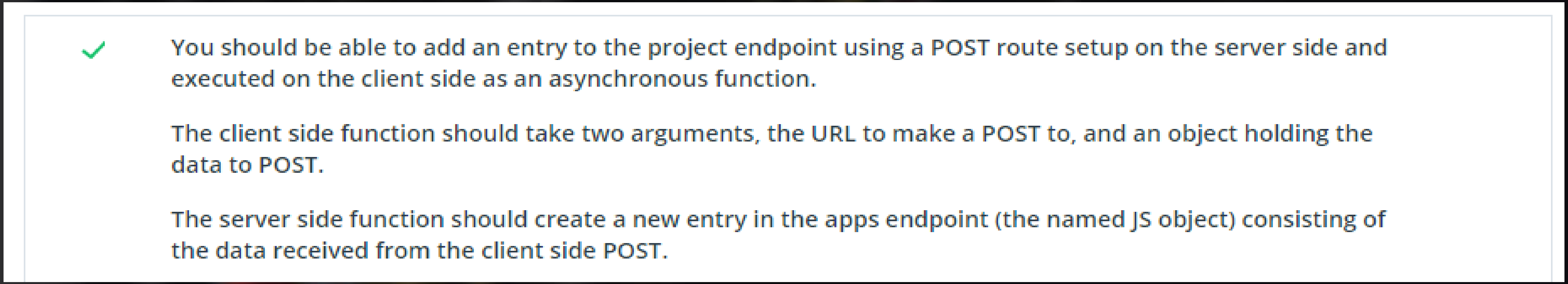

✓ There should be a GET route setup on the server side with the first argument as a string naming the route, and the second argument a callback function to return the JS object created at the top of server code.

- **This is the third specification in the *APIs and Routes* section. In this specification you need to add a get function in your server side and give it the name of the route and the call back function and return the projectData object we have created that resembles the endpoint.**



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

- This is the forth specification in the *APIs and Routes* section. In this specification you need to have an async fetch function in the client side (app.js) file that will get the data from the endpoint that is in the server side (server.js).

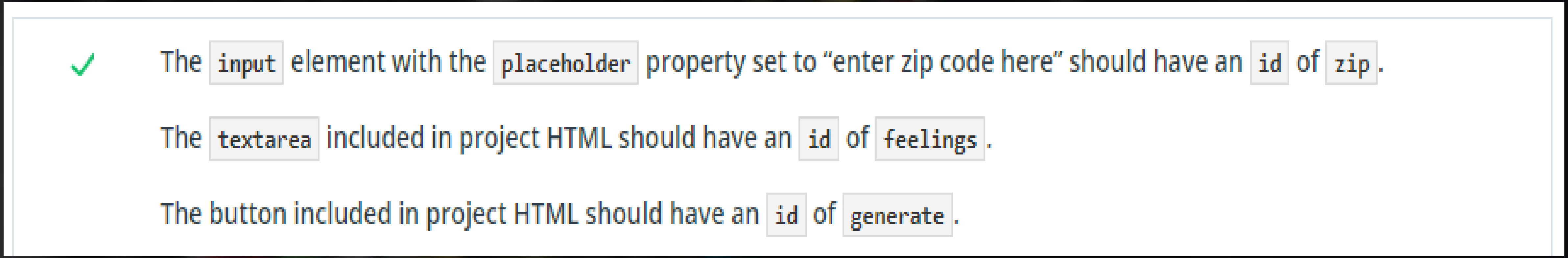

✓ You should be able to add an entry to the project endpoint using a POST route setup on the server side and executed on the client side as an asynchronous function.

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 data received from the client side POST.

- This is the fifth and last specification in the *APIs and Routes* section. In this specification you need to make sure that in your server.js file you need to be able to add an entry in the post function. the client side there should be a function that will take the URL and the object that’s having the data to post. And finally the server side should have its own function that will get the data from the client side and add it as an entry.

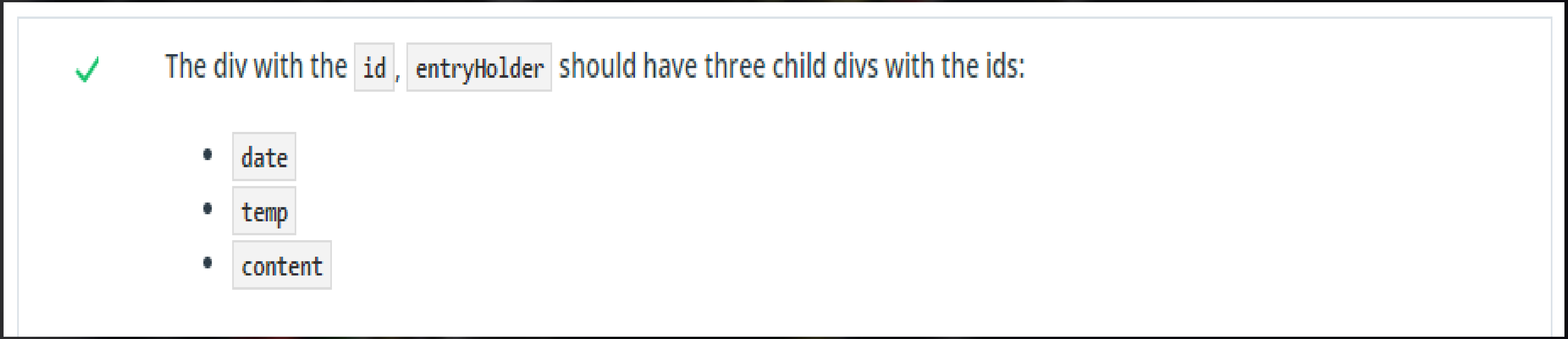
3. Dynamic UI


✓ 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`.

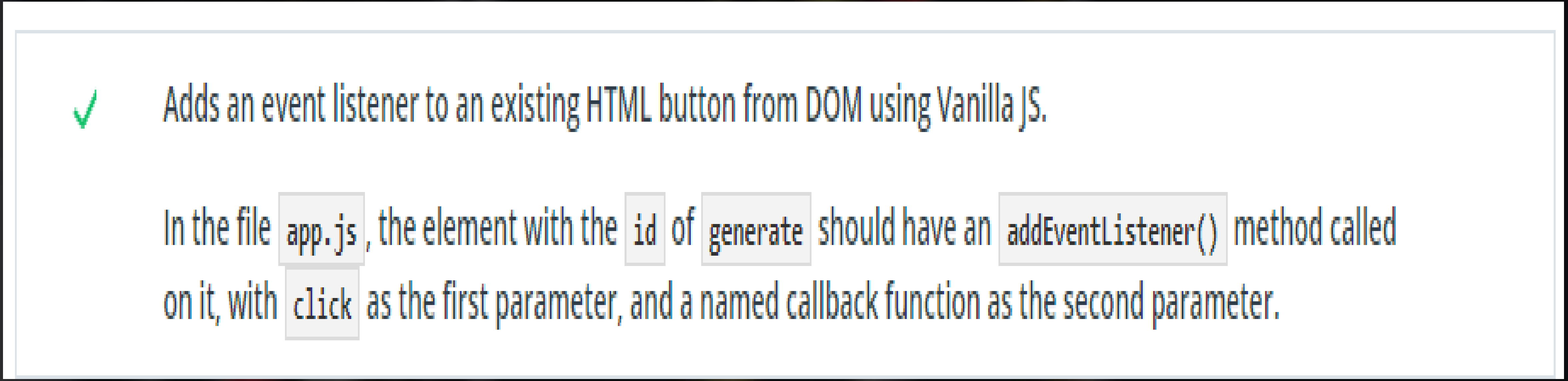
- **This is the first specification in the *Dynamic UI* section. In this specification you need to make sure that you have an input element with the placeholder of "enter zip code here" and an id of "zip". A textarea with the id of "feelings" and finally a button with the id of "generate".**



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

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

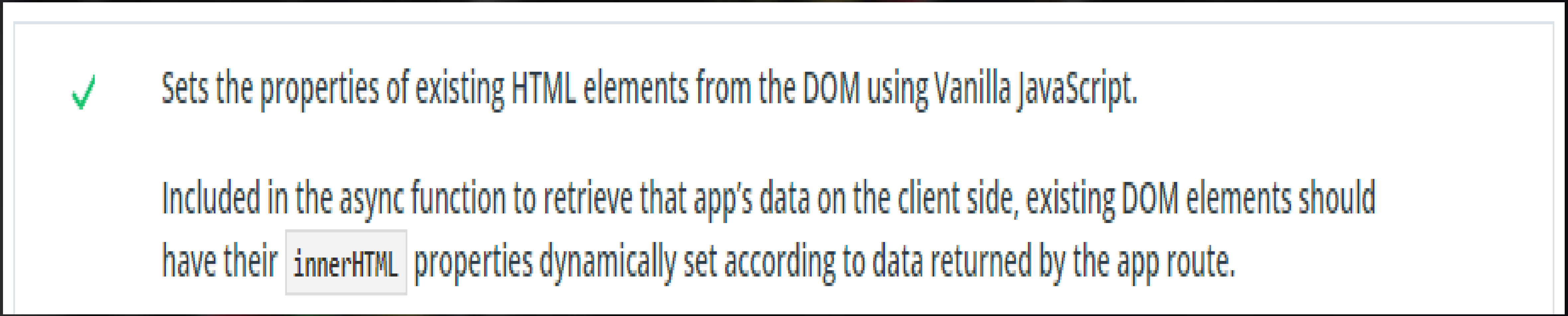
- **This is the second specification in the *Dynamic UI* section. In this specification you need to have a div with the id of “entryHolder” with the three divs that have the following ids: date, temp and content.**



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

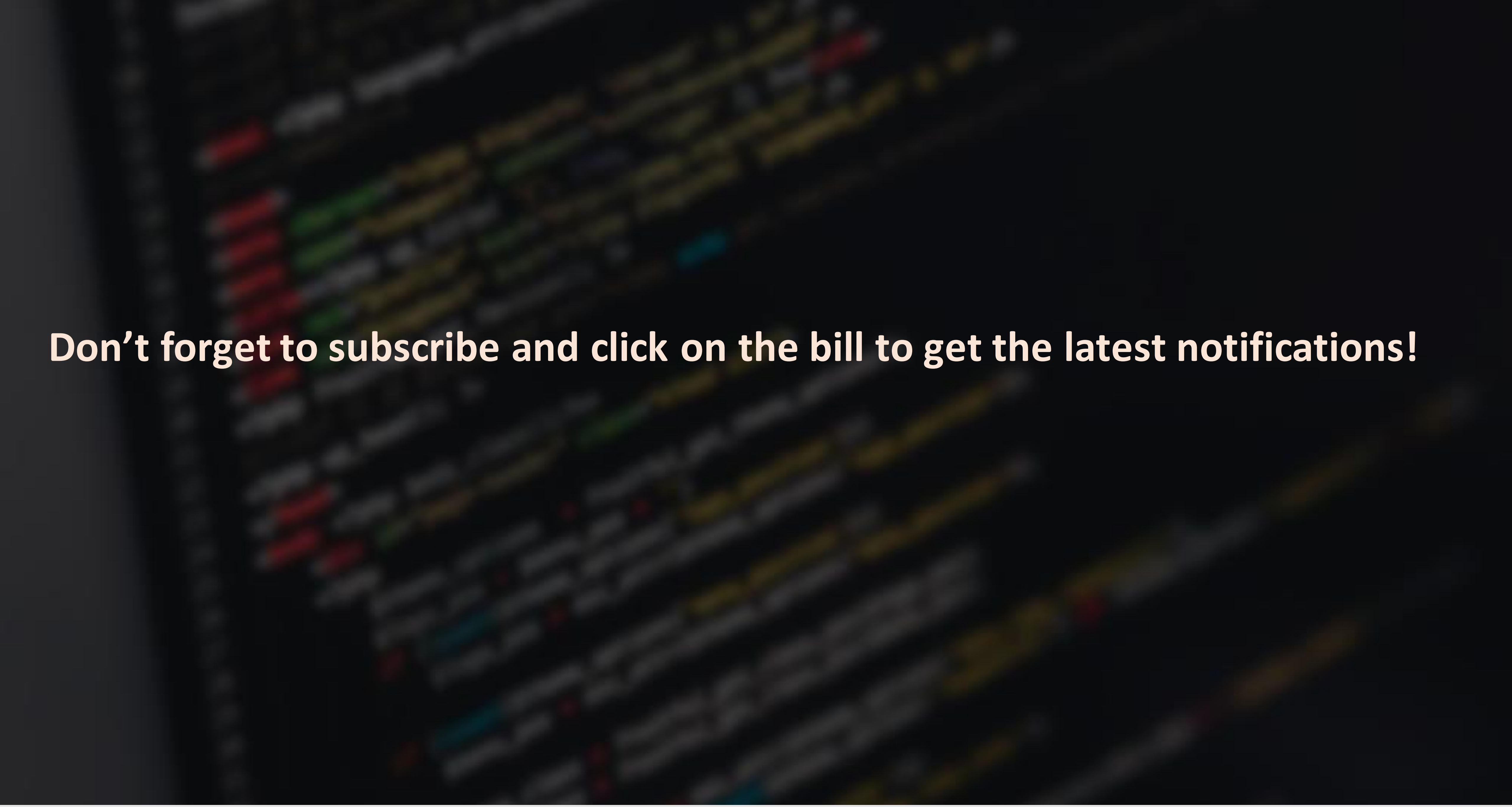
In the file `app.js`, the element with the `id` of `generate` should have an `addEventListener()` method called on it, with `click` as the first parameter, and a named callback function as the second parameter.

- **This is the third specification in the *Dynamic UI* section. In this specification you need to have an event listener to the button in the html, and another event listener with the id of generate and have the click action and uses a callback function.**


✓ 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.

- **This is the forth and last specification in the *Dynamic UI* section. In this specification you need to update the UI with the new data that you are importing from the API. You need to add the async function in the app.js file and use the innerHTML property.**



Don’t forget to subscribe and click on the bill to get the latest notifications!

https://www.youtube.com/channel/UC51OIiCEJ1hAvJlpky5cuSA