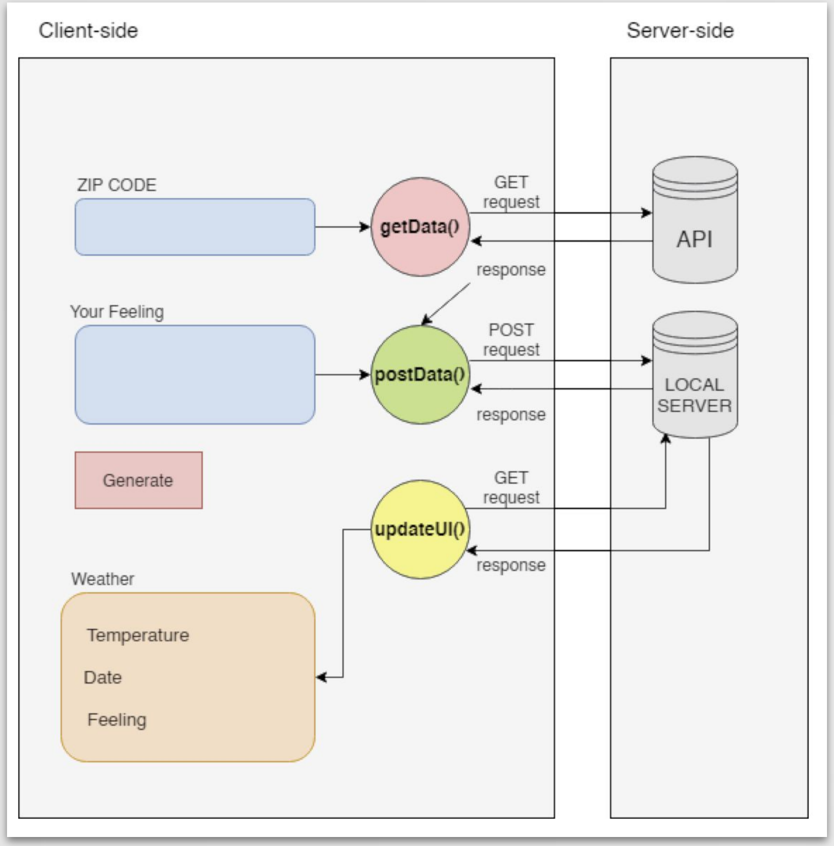
Weather App

Get Temperature Data From Api Using Zip Code

Weather APP


Client-side
Server-side
ZIP CODE
getData()
GET request
response
API
Your Feeling
postData()
POST request
response
LOCAL SERVER
Generate
updateUI()
GET request
response
Weather
Temperature
Date
Feeling

Tip 1

👉 Subscribe at OpenWeatherMap API. After subscription you will get API key. Store it in a constant in app.js file *const apiKey*

Get the 👉 URL you will use to fetch data from API with zip code. It should be something like that:

api.openweathermap.org/data/2.5/weather?zip={zip code},{country code}&appid={API key}

Delete the ,{country code}part from the URL. If no country code in URL, the API works for the USA by default, and that's fine.👍

You will replace the **{zip code} and {API key} dynamically in your code after that.**

Tip 2

In Terminal (Be sure to be at the folder where server.js file is located)

Install dependencies (in one line 😉)

npm install express body-parser cors

Tip 3

In server.js file

Require dependencies const express = require(‘express’); const bodyParser = require(‘body-parser’); const cors = require(‘cors’);

Start an instance of express const app = express();

Configure express to use cors app.use(cors());

Declare a port and give any 4 digits number (not used by another application at the same time). const port = 8888;

Setup the server and make it listen to the port app.listen(port, () => {console.log(“server is running and listening to port ${port} ”);});

Others configurations are ready-made for you in the downloaded file.

Tip 4

In app.js file

Use click event listener on the Generate Button
btn.addEventListener('click', handleGenerateBtnClick);

Inside *handleGenerateBtnClick* function create an if condition to check if user enters a value in the zip code text input or not *if (!zipCode.value) {} else {}*

If no value, alert the user to enter a zip code *alert('Please, enter a zip code');*

If there is a value, you should start calling our 3 functions and chaining them using *.then()* method. Complete the tips to know how.

GetData function (In app.js file)

Declare getData async function

const getData = async () => {}

- Inside this function use *await fetch(url)* to get temperature data from OpenWeatherMap API using the url you get in TIP #1. Replace {zip code} and {API key} with *zipCode.value* and apiKey variables respectively. Store the returned value in a constant. *const request = await fetch(....*
- Then start a try {} catch {} blocks. Inside *try {}* convert the data stored in request constant from json format to javascript using *await request.json();* and return the value.
- Inside *catch (error) {}* handle the error by *console.log(error);*

Tip 5

Tip 6

postData function (In app.js file)

- Declare postData async function with a default parameters *url = “”*, and *data = {}*
  const postData = async (url = “”, data = {}) {}
- Inside this function use
  await fetch(url, {
  “method”: “POST”,
  “credentials”: “same-origin”,
  headers: {“Content-Type”: “application/json”},
  body: JSON.stringify(data)
  }) ;
- After that start a try {} catch {} blocks. Inside try {} just return;
  Inside *catch (error) {}* handle the error by *console.log(error);*

Tip 7

In server.js file

- Use express post method to post data to the server endpoint object *projectData*.

```
app.post('/addData', (req, res) => {

projectData.temp = req.body.temp;

projectData.date = req.body.date;

projectData.userResponse = req.body.feelings;

}
```

- Be sure that the url used in post method is the same like the one used when calling postData function mentioned in TIP #6

Tip 8

updateUI function (in app.js file)

- Declare updateUI async function
  const updateUI = async () => {}
- Inside this function use *await fetch('/all')*; to get the data from the server.
- Then start a try {} catch {} blocks. Inside *try {}* convert the data stored in request constant from json format to javascript using *await request.json();*
- Then use the returned data to update the UI HTML elements via *innerHTML* property.

Tip 9

In server.js file

- Use express get method to send data from server endPoint to browser.

```
app.get('/all', (req, res) => {

res.send(projectData);

}
```

- Be sure that the url used in get method is the same like the one used in updateUI function fetch method mentioned in TIP #8

Tip 10

In app.js file

- Now return back to TIP #4 and in the else block of the if condition start calling our 3 functions chained together via .then() method
- Take care that .then() method takes a callback function s an argument and inside this function you can call one of our functions

```
 getData.then(data => postData("/", {

 temp: data.main.temp,

 date: newDate,

 feelings: feelings.value,

 })

). Then (() => updateUI() )
```