

# Hands-on Lab: Call API to Fetch Weather Data Using fetch()



Estimated time needed: 30 minutes

## What you will learn

In this lab, you will integrate an external API **OpenWeatherMap** into a web application using JavaScript. You will learn how to fetch data asynchronously, parse JSON responses, and dynamically update the webpage based on the received information. You will understand the core concepts of API integration, asynchronous operations, DOM manipulation, and user interaction, providing a foundational understanding of web development practices for utilizing external data sources in a simple interface.

## Learning objective

After completing this lab, you will be able to:

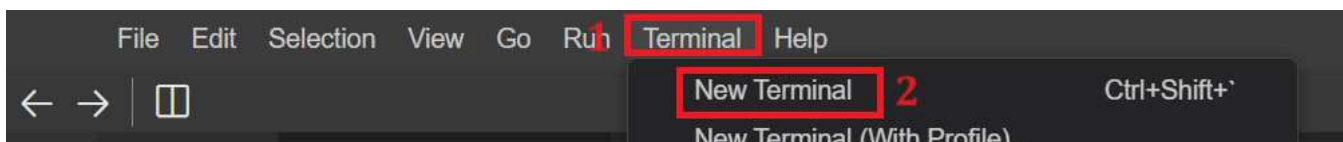
- **User-friendly weather retrieval:** Enable user input for city names, facilitating the retrieval of real-time weather information via an intuitive web interface.
- **API integration for weather data:** Utilize the OpenWeatherMap API to fetch precise weather data based on user-entered cities, dynamically displaying temperature and weather descriptions on the webpage.
- **HTML form submission handling and JS event implementation:** Manage form submissions within HTML and implement event listeners in JavaScript, ensuring smooth user interactions and data retrieval processes.
- **Demonstration of asynchronous requests and dynamic DOM updates:** Showcase the practical application of asynchronous requests using fetch(), parsing JSON responses, and dynamically updating the DOM to display fetched weather details seamlessly, eliminating the need for page refreshes.

## Prerequisites

- Basic Knowledge of HTML.
- Web browser with a console (Chrome DevTools, Firefox Console, and so on).

## Step 1: Setting up the environment

1. Firstly, you need to clone your main repository in the **Skills Network Environment** which you have created in the first lab and where you have pushed all of your previous labs files and folders. Follow given steps:
  - For this click on terminal on the top-right window pane and then select **New Terminal**.

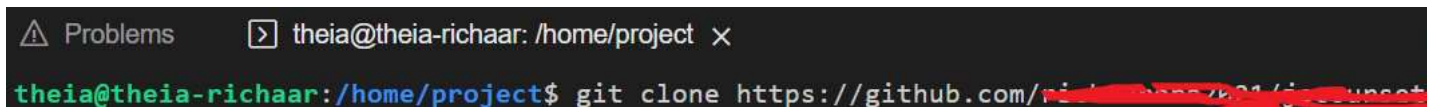


- Perform `git clone` command by writing given command in the terminal.

```
1. 1  
1. git clone <github-repository-url>
```

Copied!

**Note:** Put your own GitHub repository link instead of `<github-repository-url>`.



- Above step will clone folder for your GitHub repository under project folder in explorer. You will also see multiple folders inside cloned folder.
- Now you need to navigate inside the cloned folder. For this write given command in the terminal:

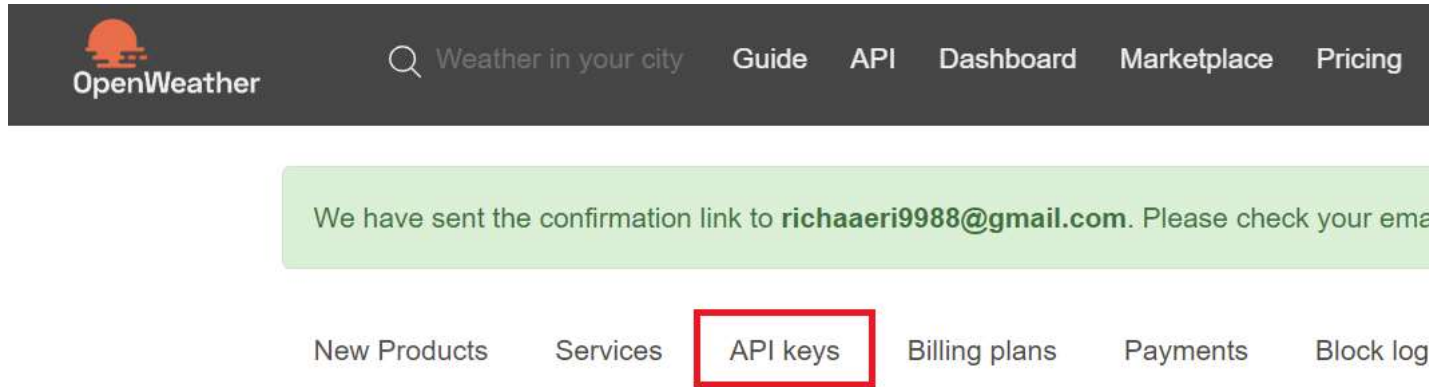
```
1. 1  
1. cd <repository-folder-name>
```

Copied!

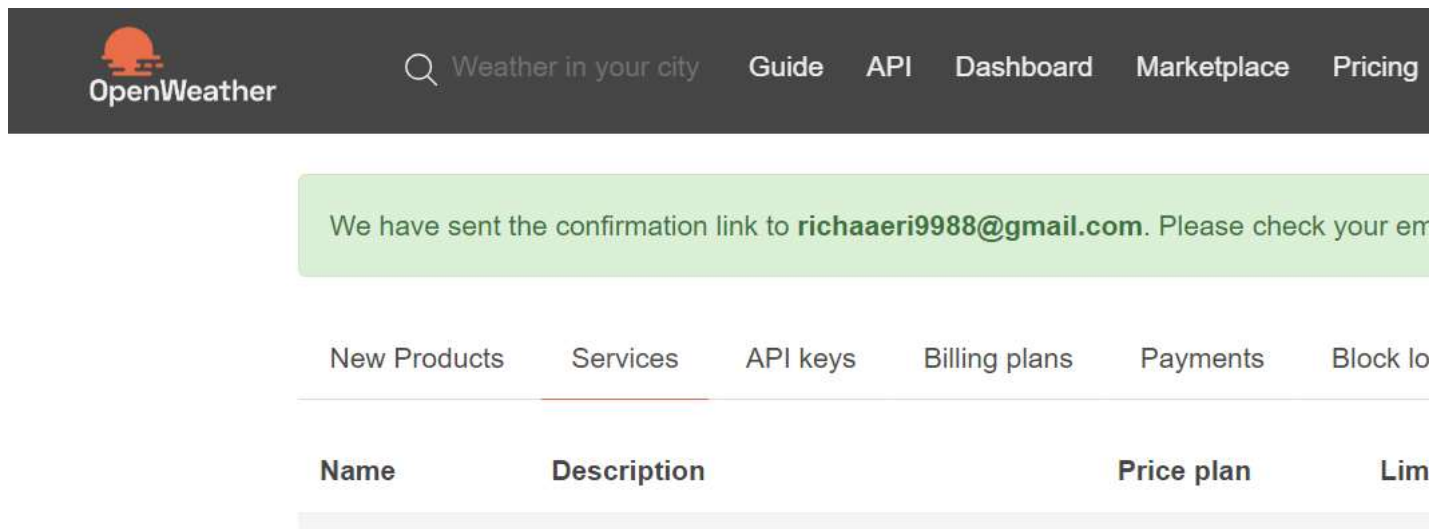


**Note:** Write your cloned folder name instead of <repository-folder-name>. Perform `git clone` if you have logged out of **Skills Network Environment** and you cannot see any files or folder after you logged in.



- Now select **cloned Folder Name** folder, right-click on it and click on **New Folder**. Enter folder name as **weatherReport**. It will create the folder for you. Then select **weatherReport** folder, right-click and select **New File**. Enter file named as **weather\_report.html** and click OK. It will create your HTML file.
- Now select **weatherReport** folder again, right click, and select **New File**. Enter file named as **weather\_report.js** and click OK. It will create your javascript file.
- To fetch data from API you need to have your personal **API Key** of that particular website from where you will generate request to fetch data.
- For this you need to open given website “<https://openweathermap.org/>” and Signup for this website.
- After signing up it will redirect you to the dashboard page in which you need to click on **API Keys** menu item.



- Note: If it will not redirect you to above said page, then you can also access it by clicking on your profile manu item in navigation bar and then select **My API Keys** as shown in given screenshot.



- After accessing page for API keys, you will navigate to page where the website has already generated a personal key for you. This key will play a vital role while fetching data from specific URL.

Key	Name	Status	Actions
d80f280c9e256dcf041daa8984d9714a	Default	Active	 

Note: Persoanl key act as password for your URL from where you will get the desired data related to weather.

8. Now you need the URL as well with the help of which data will be fetched for particular city. For this you need to click on **Services** as shown in given screenshot.

New Products	<b>Services</b>	API keys	Billing plans	Payments	Block logs	My orders
--------------	-----------------	----------	---------------	----------	------------	-----------

Name	Description	Price plan	Limits
Weather	Current weather and forecast	Free plan	Hourly forecast: unavailable Daily forecast: unavailable Calls per minute: 60

9. Then you need to click on **view**.

New Products	<b>Services</b>	API keys	Billing plans	Payments	Block logs	My orders
--------------	-----------------	----------	---------------	----------	------------	-----------

Name	Description	Price plan	Limits
Weather	Current weather and forecast	Free plan	Hourly forecast: unavailable Daily forecast: unavailable Calls per minute: 60

10. As soon as you will click on view, it will open new window and navigate to new page. You need to scroll down little in the current page where you will see option to use API for free. From there you need to click on **Current Weather**.

<b>Free</b>	<b>Startup</b> 30 GBP / month	<b>Developer</b> 140 GBP / month	<b>Professio</b> 370 GBP / month
Get API key	Subscribe	Subscribe	Subscribe
60 calls/minute 1,000,000 calls/month	600 calls/minute 10,000,000 calls/month	3,000 calls/minute 100,000,000 calls/month	30,000 calls/minute 1,000,000,000 calls/month
<b>Current Weather</b>	Current Weather	Current Weather	Current Weather
3-hour Forecast 5 days	3-hour Forecast 5 days	3-hour Forecast 5 days	3-hour Forecast 5 days
Hourly Forecast 4 days	Hourly Forecast 4 days	Hourly Forecast 4 days	Hourly Forecast 4 days
Daily Forecast 16 days	Daily Forecast 16 days	Daily Forecast 16 days	Daily Forecast 16 days
Climatic Forecast 30 days	Climatic Forecast 30 days	Climatic Forecast 30 days	Climatic Forecast 30 days

11. Again it will open new window and navigate you to the new page. Then you need to scroll down again, where you will multiple URL links provided to you which you can use to fetch data from.

12. From the current page you need to select URL as highlighted in red color. You will also use this URI in coming instructions to fetch weather related data.

## Built-in API request by city name

You can call by city name or city name, state code and country code. Please note that searching by states available only for the USA locations.

### API call

```
https://api.openweathermap.org/data/2.5/weather?q={city
name}&appid={API key}
```

```
https://api.openweathermap.org/data/2.5/weather?q={city
name},{country code}&appid={API key}
```

```
https://api.openweathermap.org/data/2.5/weather?q={city
name},{state code},{country code}&appid={API key}
```

13. An API key is like a secret passcode, and a URL is the web address where we get the information.

## Create HTML Structure

1. Create the basic template structure of HTML file by adding content the provided content.

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18

1. <!DOCTYPE html>
2. <html>
3. <head>
4.   <title>Weather Report</title>
5. </head>
6. <body>
7.   <h1>Weather Report</h1>
8.
9.   <form id="weatherForm">
10.    <label for="city">Enter City:</label>
11.    <input type="text" id="city" name="city">
12.    <button type="submit">Get Weather</button>
13.  </form>
14.
15.  <div id="weatherInfo"></div>
16. <script src="./weather_report.js"></script>
17. </body>
18. </html>
```

Copied!

**Note:** When you have pasted the code, save your file.

2. Above code has given content and elements:

- `<form>` Tag: The `<form>` tag creates a section that contains input elements and a submit button. It has an id attribute set to “weatherForm”.
- `<label>` Tag: This tag creates a label for the input field with text “Enter City:” to be displayed by the label.
- `<input>` Tag: It is used to take the entered city name by user.
- `<button>` Tag: This button indicates that this button submits the form when clicked.
- `<div>` Tag: id=“weatherInfo”: An empty `<div>` element with an identifier. This is where the weather information will be displayed.
- `<script>` Tag: This references an external JavaScript file named “weather\_report.js” to be included in current “weather\_report.html” file

## Step 2: Defining variables and functions

1. Create one function named as **showweatherDetails** and include it in **weather\_report.js** file.

```
1. 1
2. 2
3. 3
1. function showweatherDetails(event) {
2.     event.preventDefault();
3. }
```

Copied!

- Above code prevents the default behavior of an event, such as form submission, within a function named **showweatherDetails**.

2. Within **showweatherDetails** function, initialize three variables for city, apiKey and apiUrl as follows:

```
1. 1
2. 2
3. 3
1. const city = document.getElementById('city').value;
2.     const apiKey = 'c4f86ece00bc8aa272652ac9065af12d'; // Replace 'YOUR_API_KEY' with your actual API key
3.     const apiUrl = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;
```

Copied!

- Above code obtains the value entered by the user in the input field labeled 'city'.
- A URL for the OpenWeatherMap API has been constructed by combining the user-entered city name with a personal API key, essential for accessing weather data. Key has already been created for you. You can directly use this key.
  - Note: You will use your own API key which has been generated by you in the first step of **Setting up the environment**.
- This code has variable name **apiUrl** which will contain the same URL which you will instructed to save for further use in the first step of **Setting up the environment**.

3. Use fetch api method to fetch details related to city which user will enter in the input box provided in the HTML file. Include below code inside the **showweatherDetails** function below variables initialization.

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
1. fetch(apiUrl)
2.     .then(response => response.json())
3.     .then(data => {
4.         const weatherInfo = document.getElementById('weatherInfo');
5.         weatherInfo.innerHTML = `<h2>Weather in ${data.name}</h2>
6.                                 <p>Temperature: ${data.main.temp} &#8451;</p>
7.                                 <p>Weather: ${data.weather[0].description}</p>`;
8.     })
```

Copied!

4. Above function fetch initiates an asynchronous HTTP request to the specified apiUrl (OpenWeatherMap API) to retrieve weather data.
5. Response handling is being utilized by promise with **.then()** to process the response by first converting it to JSON format **response.json()**, and then accessing the resulting data.
6. HTML file is also being Updated the HTML content dynamically by selecting the 'weatherInfo' element, populating it with structured weather information such as city name **data.name**, temperature **data.main.temp**, and weather description **data.weather[0].description**, ensuring a user-readable display of fetched weather details on the webpage.
7. Then include below give code outside the function. It attaches an event listener to the 'weatherForm' element, listening for a 'submit' event and triggering the showweatherDetails function upon form submission, enabling customized handling or manipulation of the form's behavior.

```
1. 1
1. document.getElementById('weatherForm').addEventListener('submit', showweatherDetails );
```


Copied!

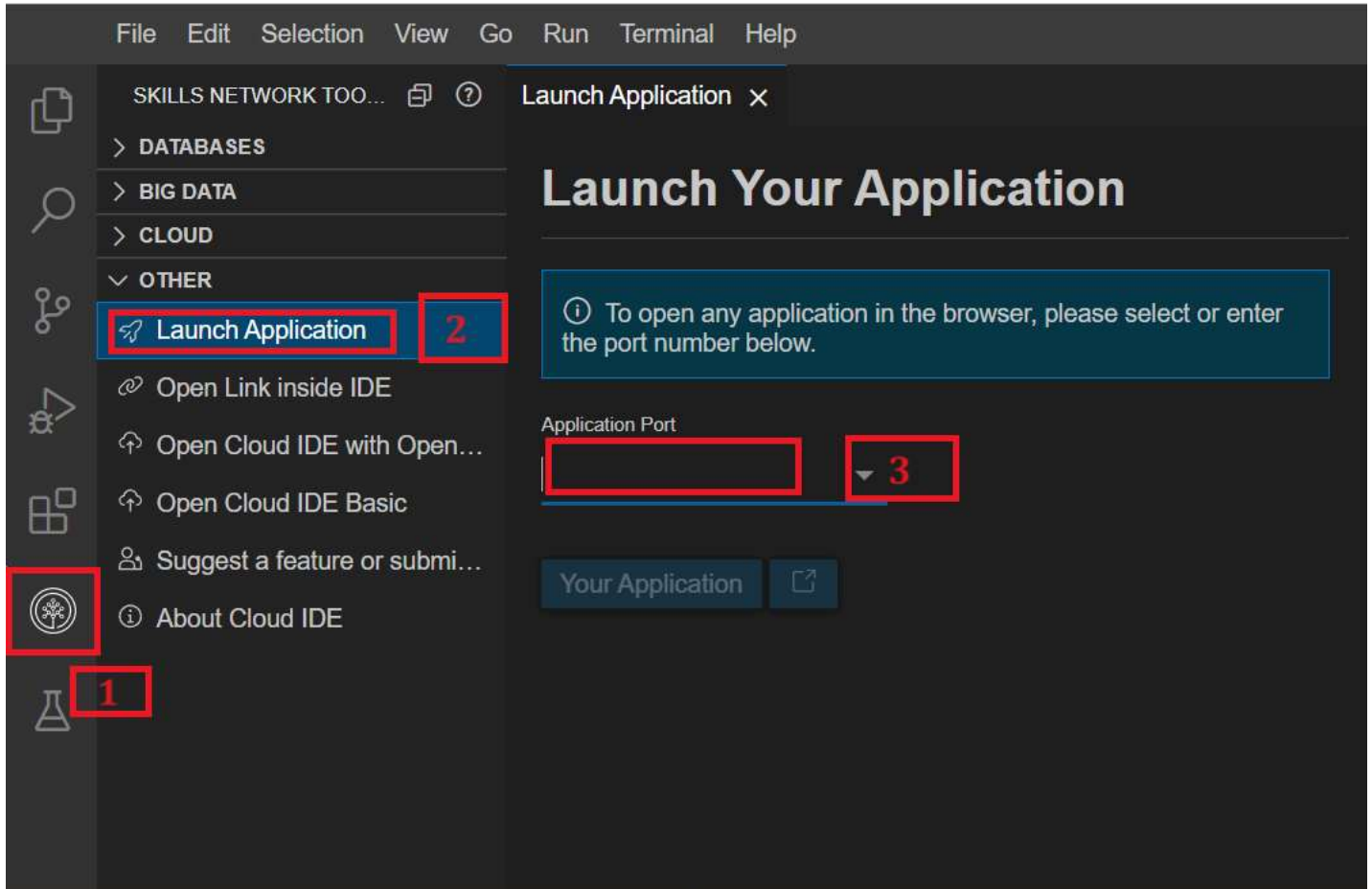
## Step 3: Check the output

1. To view how your HTML page, right-click the **weather\_report.html** file after selecting this file, then select “Open with Live Server”.

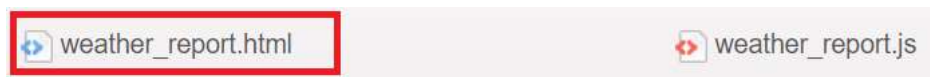
2. The server should start on port 5500, indicated by a notification on the bottom right side.



3. Click the Skills Network button on the left at number 1. It will open the "Skills Network Toolbox". Next, click on "Other" and then select "Launch Application" at number 2. From there, you enter port number 5500 at number 3 and click this button .



4. It will open your default browser where you will see **cloned-folder-name** folder name. Click on that **cloned-folder-name** folder name and then click on **weatherReport** folder name. You will see files related to this folder where you will click again on **weather\_report.html** file as shown below.



5. It will open the front page and you will see the output as shown below.

# Weather Report

Enter City:

6. Enter any city name in input box and then click on **Get Weather** button and then you will see the result.



# Weather Report

Enter City:

## Weather in Chicago

Temperature: 1.26 °C

Weather: broken clouds

7. You can also include catch method to catch error if user enters wrong city name. Include below code before end of function.

```
1. 1
2. 2
3. 3
4. 4
5. 5
1. .catch(error => {
2.     console.error('Error fetching weather:', error);
3.     const weatherInfo = document.getElementById('weatherInfo');
4.     weatherInfo.innerHTML = `<p>Failed to fetch weather. Please try again.</p>`;
5. });
```

Copied!

**Note:** After pasting the code, save your file. You can use any output method for saving. If you make further edits to your code, simply refresh your browser running on port number 5500. This eliminates the need to relaunch the application repeatedly.

## Step 4: Perform Git commands

1. Perform `git add` to add latest files and folder by writing given command in terminal in git environment.

```
1. 1
1. git add --a
```

Copied!

Make sure terminal should have path as follows:



2. Then perform `git commit` in the terminal. While performing `git commit`, terminal can show message to set up your `git config --global` for user name and user email. If yes, then you need to perform `git config` command as well for user name and user email as given.

```
1. 1
1. git config --global user.email "you@example.com"
```

Copied!

```
1. 1
1. git config --global user.name "Your Name"
```

Copied!

**Note:** Replace data within quotes with your own details.

Then perform commit command as given:

```
1. 1
1. git commit -m "message"
```

Copied!

3. Then perform `git push` just by writing given command in terminal.

```
1. 1
1. git push origin
```

Copied!

- After the push command, the system will prompt you to enter your username and password. Enter the username for your GitHub account and the password that you created in the first lab. After entering the credentials, all of your latest folders and files will be pushed to your GitHub repository.

## Practice Task

1. You will use another URL to access details weather reports. For example you need to use URL as shown in given screenshot by accessing the same page from where you use URL to be used in the lab instructions.

## Call current weather data

### How to make an API call

#### API call

```
https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}
```

Note: Instead of city name, now you will gather details for latitude and longitude from user.

2. For this you need to take two `<input>` fields, one to access latitude and to get longitude value from user.

## Call current weather data

latitude value

### How to make an API call

longitude value

api key generated by you

#### API call

```
https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}
```

3. You need to create one submit button after clicking on which it will display the results.
4. Then apply fetch api method to get data related to weather for the user input and create javascript code for the same.

## Summary

1. **Weather retrieval form:** Constructed an HTML page with a form enabling users to input a city name and fetch weather data from OpenWeatherMap upon submission.
2. **JavaScript functionality:** Defined a JavaScript function `showweatherDetails` to handle form submission, preventing default behavior, extracting the entered city, and generating a URL to fetch weather details using the OpenWeatherMap API.
3. **Dynamic weather display:** Utilized JavaScript's `fetch()` to retrieve weather data, updating the webpage dynamically with the fetched weather information, including city name, temperature in Celsius, and weather description, ensuring a seamless display of weather details without page reload.