WEATHEROASIS - API BASED WEATHER APPLICATION

A PROJECT REPORT ON OPEN SOURCE WEATHER API BASED APPLICATION

Created & Submitted via

ANURAG GUPTA, 19317

RATNESH TRIPATHI, 19348

Under Guidance of

Er. Praveen Mishra

Assistant Professor

Department of IT

Bachelor of Technology

Department of Information Technology

Institute of Engineering and Technology, Ayodhya

2021

Dr. Rammanohar Lohia Avadh University

Ayodhya, Uttar Pradesh - 224001

CERTIFICATE

This is to certify that the Mini Project Report entitled **"WEATHEROASIS - API BASED WEATHER APPLICATION"** is being submitted by **ANURAG GUPTA (19317)** in partial fulfillment of the requirements for the award of the Degree of **Bachelor of Technology** in **Information Technology**, to the **Institute of Engineering & Technology, Dr. RMLAU, Ayodhya,** during the academic year 2020-2021, is a bonafide work carried out by him under my guidance and supervision.

The results presented in this Project Work have been verified and are found to be satisfactory. The results embodied in this Project Work have not been submitted to any other University for the award of any other degree or diploma.

Project Guide

Er. Praveen Mishra
Assistant Professor
Department of IT

Head of Department

Prof. Dr. Mohit Singh Gangwar
Professor
Department of IT

DECLARATION

We hereby declare that the work which is being presented in this dissertation entitled, **“WEATHEROASIS - API BASED WEATHER APPLICATION”,** submitted towards the partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Information Technology**, **Institute of Engineering & Technology, Dr. RMLAU, Ayodhya** is an authentic record of his own work carried out under the supervision of **Er. Praveen Mishra, Assistant Professor, Department of IT,** Institute of Engineering & Technology, Dr. RMLAU, Ayodhya.

To the best of our knowledge and belief, this project bears no resemblance with any report submitted to IET, Dr. RLAU or any other University for the award of any degree or diploma.

ANURAG GUPTA (19317)

TABLE OF CONTENT

S.No.	*Topic*	*Page No.*
1.	To use an API	5
2.	What is API	5
3.	Getting Started with our Web App	5-6
4.	Setting up the Basic Structure of our page	6-11
5.	Making our Request to the OpenWeatherMap API	11-13
6.	Register for an API key on OpenWeatherMap	13
7.	Making Our Final Request	13-16
8.	Project Result Output Overview	16
9.	Live Web App Link	16
10	Bibliography	16

To use an API.

Using an API allows us to make any website or web application more exciting and dynamic than what we can accomplish with HTML, CSS and JavaScript by themselves.
In this project we will be making use of the Weather API from OpenWeatherMap to create a Weather App Application to check live weather reports.

What is an API?

API is an acronym that stands for Application Programming Interface and according to wikipedia, an application programming interface (API) is a computing interface which defines interactions between multiple software intermediaries.

What that definition basically means is that APIs allow our web application to communicate with other web applications. This “communication” can be best explained with an example specific to the project we are going to build.

For this project, we will be using the Weather API from OpenWeatherMap to create a Weather App. We are using the API to communicate with OpenWeatherMap (openweathermap.org). In this communication, we are making a “request” to OpenWeatherMap’s server to get access to some of their weather data, so we can use them for our Web Application.

The most important part to take note of here is that we are using the API to get access to OpenWeatherMap’s weather data for us to use in this project; what this means is that we can use an API to get data from an outside source which is very cool and very useful.

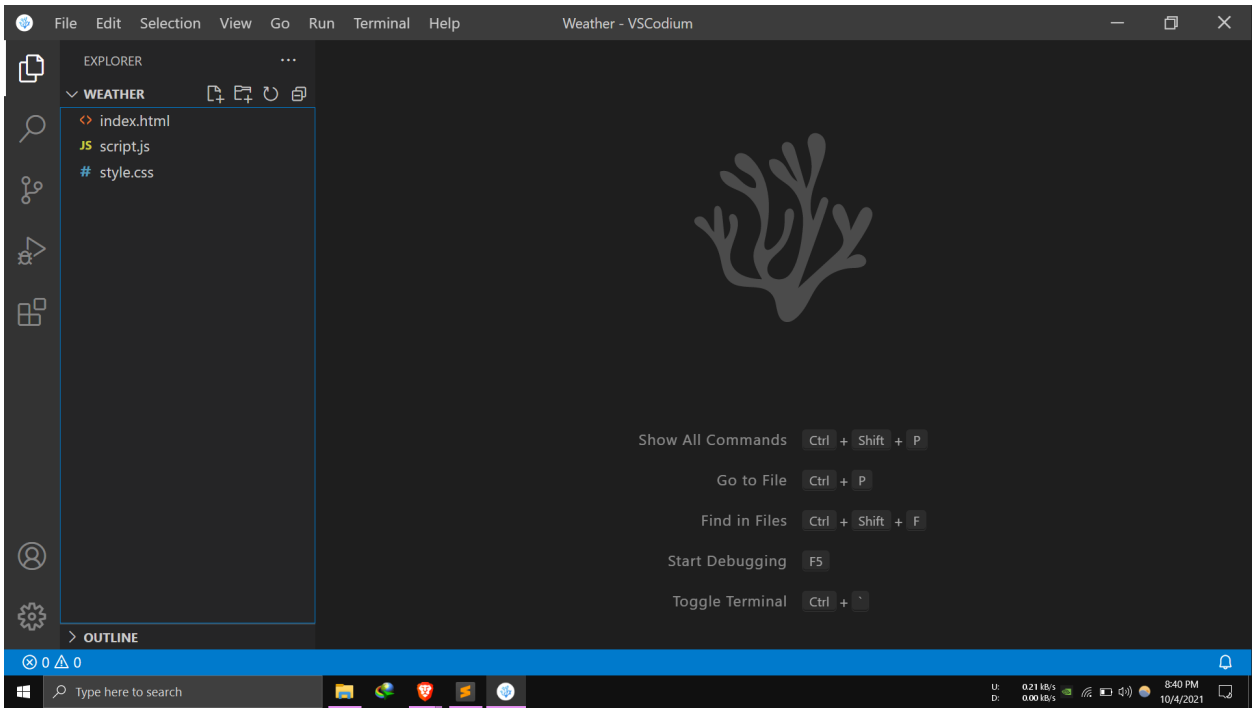
Getting Started with our Web App

Before we can get started with using the API, we have to set up a basic structure for our project. For this project we will need a code editor to write our code in. I personally use the VSCodium and Sublime code editor but any code editor will work.

Open up the code editor and create a directory (folder) with the title “WeatherApp”. Inside the directory we need mainly the following three files to get started:

- index.html
- style.css
- script.js

Look to the picture below for a VSCodium on what it should look like:



We need these 3 files in our project directory to get us started

Setting up the Basic Structure of our page

The content of our web page is going to be made up mostly of cards generated in our JavaScript file but we do need to set up a basic structure of our web page using some HTML and CSS.

Below is code snippet of the HTML code used for this project:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>WeatherOasis</title>
    <link rel="shortcut icon" href="icon.png" width="10px">
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
 <body>
```

```
  <section class="top-banner">
    <div class="container">
      <h1 class="heading">Anurag's Weather App</h1>
      <form>
        <input type="text" id="anuragoasis" placeholder="Search for a City" autofocus
autocomplete="off">
        <button type="search">SEARCH</button>
        <span class="msg"></span>
      </form>
    </div>
  </section>
  <section class="ajax-section">
    <div class="container">
      <div class="card">
          <div class="content cities">

          </div>
      </div>
    </div>
  </section>
  <script type="text/javascript" src="vanilla-tilt.js"></script>
  <script src="index.js"></script>
<script type="text/javascript" src="script.js"></script>
</body>
</html>
```

In the html code above, notice that there is a section of it that says <section class="ajax-section">. This part is important because this is the html tag that will contain all of the city data cards that we will get from our call to the OpenWeatherApp API. The *style.css* and the *vanilla-tilt.js* are used in our html for styling purposes. Lastly all the code inside the <head></head> will be used to make sure our html document meets with the web's best practices and to allow our css file and javascript file to be loaded.

Now in the style.css file the following is the code:

```
:root {  --bg_main: #0a1f44;
 --text_light: #fff;
 --text_med: #53627c;
 --text_dark: #1e2432;
 --red: #ff1e42;
 --darkred: #c3112d;
 --orange: #ff8c00;}

* {  margin: 0;
```

```
  padding: 0;
  box-sizing: border-box;
  font-weight: normal;}

a {  color: inherit;
  text-decoration: none;}

button {  cursor: pointer;}

input {  -webkit-appearance: none;}

button, input {  border: none;
  background: none;
  outline: none;
  color: inherit;}

img {  display: block;
  max-width: 100%;
  height: auto;}

ul {  list-style: none;}

body {  font: 1rem/1.3 "Roboto", sans-serif;
  background-image: url("GTA-ViceCity.jpg");
  background-position: center;
  background-attachment: fixed;
  background-size: cover;
  background-repeat: no-repeat;
  color: var(--text_dark);
  padding: 70px;}

.container {  width: 100%;
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 15px;}

.cont1 {  color:#cccccc;
  text-align: right;}

.top-banner {  color: var(--text_light);}

.heading {   color:#191c21;
  font-weight: bold;
  font-size: 4rem;
```

```
  letter-spacing: 0.02em;
  padding: 0 0 30px 0;}

.top-banner form {  position: relative;
  display: flex;
  align-items: center;}

.top-banner form input {  font-size: 2rem;
  height: 40px;
  padding: 5px 5px 10px;
  border-bottom: 1px solid;}

.top-banner form input::placeholder {  color: currentColor; }

.top-banner form button {  font-size: 1rem;
  font-weight: bold;
  letter-spacing: 0.1em;
  padding: 15px 20px;
  margin-left: 15px;
  border-radius: 5px;
  background: var(--red);
  transition: background 0.3s ease-in-out;}

.top-banner form button:hover {  background: var(--darkred);}

.top-banner form .msg {  position: absolute;
  bottom: -40px;
  left: 0;
  max-width: 450px;
  min-height: 40px;}

.cont1 {  text-align: right;}

.ajax-section {  margin: 70px 0 20px;}

.ajax-section .cities {  display: grid;
  grid-gap: 32px 20px;
  grid-template-columns: repeat(4, 1fr);}

.ajax-section .city {  position: relative;
  padding: 40px 10%;
  border-radius: 20px;
  background: var(--text_light);
  color: var(--text_med);}
```

```
.ajax-section .city::after {  content: '';
  width: 90%;
  height: 50px;
  position: absolute;
  bottom: -12px;
  left: 5%;
  z-index: -1;
  opacity: 0.3;
  border-radius: 20px;
  background: var(--text_light);}

.ajax-section figcaption {  margin-top: 10px;
  text-transform: uppercase;
  letter-spacing: 0.05em;}

.ajax-section .city-temp {  font-size: 5rem;
  font-weight: bold;
  margin-top: 10px;
  color: var(--text_dark);}

.ajax-section .city sup {  font-size: 0.5em;}

.ajax-section .city-name sup {  padding: 0.2em 0.6em;
  border-radius: 30px;
  color: var(--text_light);
  background: var(--orange);}

.ajax-section .city-icon {  margin-top: 10px;
  width: 100px;
  height: 100px;}

@media screen and (max-width: 1000px) {
body {   padding: 30px;  }
    .ajax-section .cities {    grid-template-columns: repeat(3, 1fr);  }
}

@media screen and (max-width: 700px) {
  .heading,
  .ajax-section .city-temp {    font-size: 3rem;  }

  .ajax-section {    margin-top: 20px;  }

  .top-banner form {    flex-direction: column;
```

```
   align-items: flex-start;  }

  .top-banner form input,
  .top-banner form button {    width: 100%;  }

  .top-banner form button {    margin: 20px 0 0 0;  }

  .top-banner form .msg {    position: static;
    max-width: none;
    min-height: 0;
    margin-top: 10px;  }

  .ajax-section .cities {    grid-template-columns: repeat(2, 1fr);  }
}

@media screen and (max-width: 500px) {
  body {    padding: 15px;  }

  .ajax-section .cities {    grid-template-columns: repeat(1, 1fr);  }
}

/* API  */

.api {  background: #053275;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  padding: 10px;}

.api a {  text-decoration: underline;}

.api a:hover {  text-decoration: none;}
```

Making our Request to the OpenWeatherMap API

So we're finally ready to start working with APIs in our JavaScript file…

In order to use the OpenWeatherMap API in JavaScript, we need to learn how to use the Fetch method and we need to register for an API key.

What is the Fetch Method?

The Fetch method is a method provided by the Fetch API that is built into the browser. It is used to get access to data through the use of the HTTP protocol,which the system used to transfer files throughout the internet.

Below is a snippet of JavaScript code that will show the syntax of how to make a request using the Fetch method.

```
const requestUrl = "url of whatever we want to make a request to";
fetch(requestUrl)
.then(response => response.json())
.then(data => {
  // do something with the data the API has returned
})
```

Explanation of what is going on in the code above.

In the first line of code we have defined a variable that contains the url of the API we would like to make a request to.
In the second line, we started the fetch request with the fetch method.
It is important to note that the fetch method takes 2 arguments (the 2nd argument is optional). If we only use one argument the request will default to a GET request; if we use the 2nd argument, we have access to all the other requests types.

So let's just say we decided to log the result of fetch(requestUrl) with a valid url as an argument to the browser console using console.log(). When logged to the console we will get back a promise object. A promise is a special JavaScript Object that will allow us to receive our data from the fetch request if it is successful or execute some snippet of code if the requested data fails to get back to us.

Since our fetch returns a promise object, we have access to the following three methods:

- .then(callback) which is is used when the promise is fulfilled (success)
- .catch(callback) which is used when the promise is rejected (fail)
- .finally(callback) happens on both a fulfilled and rejected promise

In the above code snippet we use .then() to allow our code to do something with a successful promise. Our .then() method, and all other methods of promise objects, takes a callback function as an argument. It is considered best practice to use an ES6 arrow function as a callback function so we use it in both of our .then() methods.

Here, we have chained two .then() methods onto our fetch request; the reason why we did this is because the result of calling our first .then() on the promise returned from fetch() is actually another promise object!

Because our .then() returns another promise object, we have to use another .then() on it to get access to our data from our api request.

Lastly at the callback functions inside each of the .then() method invocations . Inside the first .then() we put:

```
response => response.json()
```

What this does is allow us to get access to our data from the second promise in the form of a JavaScript object.

We need to include this part in our fetch request in order to get back the data we want as a result for us to use in the callback of our second .then()

Register for an API key on OpenWeatherMap

Now that we have finished covering how to use the fetch method, the last thing we need is to register for an API key on OpenWeatherMap's website. We need an API key in order to make fetch requests to OpenWeatherMap's API.

The API key will provide us with authorization to use OpenWeatherMap's data. In order to get our API key go to https://openweathermap.org/api, and Create a New Account. After registration, we should be given access to our unique API key.

API keys are as important as the password to our phone or bank account so make sure that we do not make our API keys publically available or share them with anyone. What this means is that if we ever want to host this project or to use an API in a future website or application, we will need to look into how to hide our API keys properly.

Let's Finally Make Our Request

Below is code snippet of the JavaScript code used for this project

```
const form = document.querySelector(".top-banner form");
const input = document.querySelector(".top-banner input");
const msg = document.querySelector(".top-banner .msg");
const list = document.querySelector(".ajax-section .cities");
const apiKey = "4d8fb5b93d4af21d66a2948710284366";

form.addEventListener("submit", e => {
  e.preventDefault();
  let inputVal = input.value;
```

```
 const listItems = list.querySelectorAll(".ajax-section .city");
 const listItemsArray = Array.from(listItems);

 if (listItemsArray.length > 0) {
   const filteredArray = listItemsArray.filter(el => {
     let content = "";

     if (inputVal.includes(","))
     {
       if (inputVal.split(",")[1].length > 2) {
         inputVal = inputVal.split(",")[0];
         content = el
           .querySelector(".city-name span")
           .textContent.toLowerCase();
       } else {
         content = el.querySelector(".city-name").dataset.name.toLowerCase();
       }
     } else {

       content = el.querySelector(".city-name span").textContent.toLowerCase();
     }
     return content == inputVal.toLowerCase();
   });

   if (filteredArray.length > 0) {
     msg.textContent = `You already know the weather for ${
       filteredArray[0].querySelector(".city-name span").textContent
     } ...otherwise be more specific by providing the country code as well 😉`;
     form.reset();
     input.focus();
     return;
   }
 }

 //ajax
 const url =
`https://api.openweathermap.org/data/2.5/weather?q=${inputVal}&appid=${apiKey}&units=metric`;

 fetch(url)
   .then(response => response.json())
   .then(data => {
     const { main, name, sys, weather } = data;
```

```
    const icon = `https://s3-us-west-2.amazonaws.com/s.cdpn.io/162656/${
      weather[0]["icon"]
    }.svg`;

    const li = document.createElement("li");
    li.classList.add("city");
    const markup = `
      <h2 class="city-name" data-name="${name},${sys.country}">
        <span>${name}</span>
        <sup>${sys.country}</sup>
      </h2>
      <div class="city-temp">${Math.round(main.temp)}<sup>°C</sup></div>
      <figure>
        <img class="city-icon" src="${icon}" alt="${
      weather[0]["description"]
    }">
        <figcaption>${weather[0]["description"]}</figcaption>
      </figure>
    `;
    li.innerHTML = markup;
    list.appendChild(li);
  })
  .catch(() => {
    msg.textContent = "Please search for a valid city 😩";
  });

 msg.textContent = "";
 form.reset();
 input.focus();
});
```

Explanation of what the code in this snippet is doing.

In the fourth line we got access to the following tag in our html code: <section class="ajax-section">. We need this because this is where our cities will be added in the page. In the const url line we created a variable to store api call to the OpenWeather API.

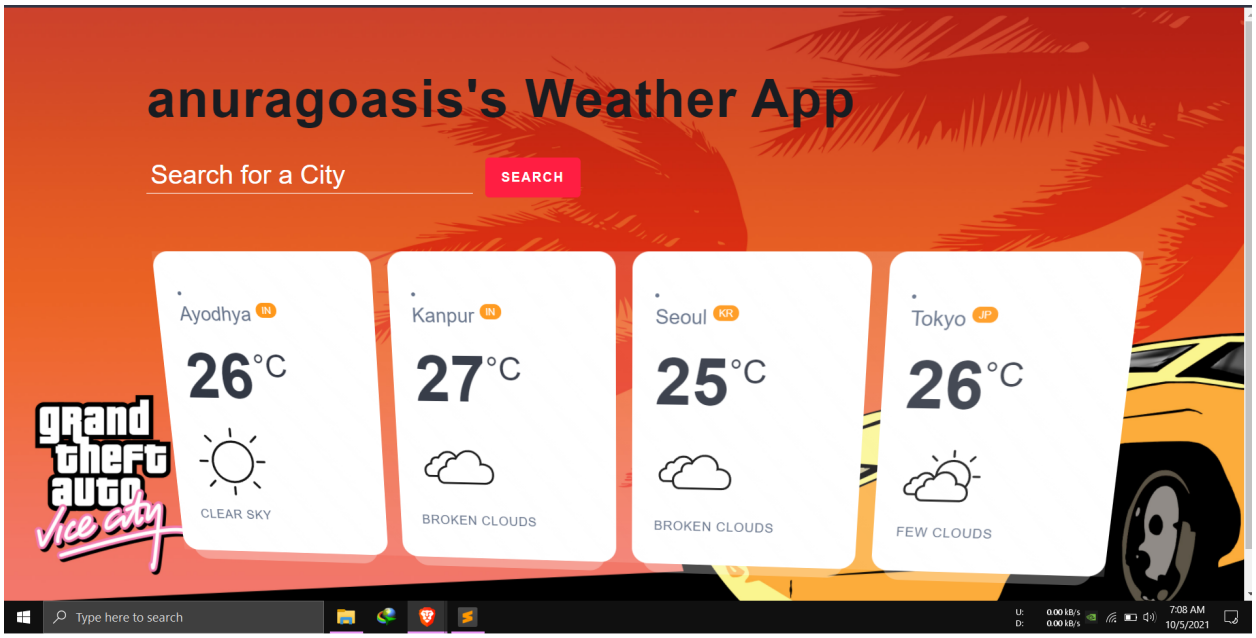
At the string stored inside our url variable. Inside it, we have a string of the url of a request to be used inside the fetch method. The main url used in all requests to the OpenWeather API is https://api.openweathermap.org and it is used in every API call that we make to OpenWeatherMap. Everything we put after the main url is called an API endpoint, and each endpoint will allow us to access a different piece of OpenWeather's database.

The endpoint that we are using in the url variable will allow us to search OpenWeather's database for a specific keyword which will be the city name and return data associated with that city. The keyword that we are using in our search is the name or country code of any valid city.

It is important to note here that when we have access to our data from the second .then() , we should make sure to log the result of that data to the console to see what kind of data structure we are working with. This is a great practice to have as we should always be checking to see what form our data comes in.

Inside our second .then() method, our code takes the data from openweather and uses it to create a city card with name, temperature, weather icon, weather description and card styling using open source vanilla-tilt js code. This is the only point in which we can use the data that we requested.

Adding this code snippet will make our page look like this:



Use Live on: https://anuragoasis.github.io/WeatherOasis/

BIBLIOGRAPHY

Weather API - OpenWeatherMap: https://openweathermap.org/api
VSCodium - Open Source Binaries of VSCode: https://vscodium.com/
Sublime Text - Text Editing, Done Right: https://www.sublimetext.com/
GitHub - Where the world builds software: https://github.com/
Brave Browser - Secure, Fast & Private Web Browser with Adblocker: https://brave.com/