**Making Weather App**

A screenshot of a weather app

Description automatically generated with medium confidence**HTML5:**

1. *First we make a container(the red giant square) that will holds all the elements needed for the site.*
2. *Then make a form for the AutoComplete function and the input and submit function*
3. *Adding <ul>[[1]](#endnote-1) tag for the list for the AutoComplete feature*
4. *Adding <div> for the blue square in the picture*

**JavaScript:**

**GetInfo.js:**

*Line 1: adding the apikey for the open weather api*

*Line 3: Getting the container for the blue square*

*Line 5: Getting the container for the form(Input and Submit)*

const apikey = "166cd4304640df33421572aeffaee337";// the api for the open weather site

const weatherData = document.getElementById("weather\_data");// getting the container for weather data

const formEl = document.querySelector("form");[[2]](#endnote-2)

*Line 7 - 12: Making an our own event so the other js files can use it(addEventListener)*

document.addEventListener('myCustomEvent', function() {

// The event has been triggered, perform actions here

const cityInput = document.getElementById("city\_input").value;

getWeatherData(cityInput);

});

*Line 15: Adding event for the submit button when is clicked*

// Add the form submission event listener

formEl.addEventListener("submit", event => {

event.preventDefault();[[3]](#endnote-3)

const cityInput = document.getElementById("city\_input").value;

getWeatherData(cityInput);

});

*Line 23 – 49: Making the function getWeatherData()*

1. making variable(response) that await to fetch the info from the api
2. Check if the response is ok
3. Then making the response into json()

const response = await fetch(`https://api.openweathermap.org/data/2.5/weather?q=${cityInput}&appid=${apikey}&units=metric`); if(!response.ok){

throw new Error("The Network response is an error, please try again later!");

}

const data = await response.json();

1. *Then making array for the <div> in the blue square*

const details = [

`Feels like: ${Math.round(data.main.feels\_like)}°C`,

`Humidity: ${Math.round(data.main.humidity)}%`,

`Wind spee: ${Math.round(data.wind.speed)}m/s`

]

1. *At the end I fills the elements*

weatherData.querySelector(".icon").innerHTML = `<img src="https://openweathermap.org/img/wn/${data.weather[0].icon}.png" alt = "Weather Icon">`;

weatherData.querySelector(".temperature").textContent = `${Math.round(data.main.temp)}°C`;

weatherData.querySelector(".description").textContent = `${data.weather[0].description}`;

weatherData.querySelector(".details").innerHTML = details.map((detail) => `<div>${detail}</div>`).join("");

**AutoComplete.js:**

This js file is only for the <ul> tag for the AutoComplete feature or Suggestions when write a city

*Line 2-3: Making variables for getting the city input and then he <ul> id*

// Get the input element and the suggestion list element

let input = document.getElementById("city\_input");

let suggestionList = document.getElementById("list");

*Line 6-16: Getting Matches*

1. Getting the names of the cities in the world by fetching them from the json file
2. Then makes response variable that turn the file into json
3. Then I made eventListener when a key in press and released
4. Then made a variable for matchigCity with filter to lower the letter in the name of the cities so they can completely match
5. Then sends the matchingCities to the function renderSuggestion(matchingCity)

// Fetch the city data from the provided URL

fetch("https://raw.githubusercontent.com/lutangar/cities.json/master/cities.json")

.then(res => res.json())

.then(data => {

// Add event listener to the input element for keyup eventt

input.addEventListener("keyup", event => {

// Filter the city data based on the input value

let matchingCity = data.filter(city => city.name.toLowerCase().startsWith(input.value.toLowerCase()));

// Call the renderSuggestion function to display the matching cities

renderSuggestion(matchingCity);

});

});

*Line 18-49: renderSuggestion(matchingCity)*

1. First make sure that <ul> is empty
2. Then for each city that matches create element <li>[[4]](#endnote-4)
3. Then make an event when one of the <li> in the <ul> is clicked to fills the name into the input and make the list empty again and trigger our own event which I have made in the GetInfo.js

function renderSuggestion(cities) {

// Clear the existing suggestions

suggestionList.innerHTML = '';

// Iterate over the cities and create suggestion elements

cities.forEach(city => {

const suggestion = document.createElement("li");

suggestion.textContent = city.name;

// Add event listener to each suggestion for click event

suggestion.addEventListener("click", event => {

// Set the input value to the selected city name

input.value = city.name;

// Clear the suggestion list

suggestionList.innerHTML = ``;

//trigger the event of Get Weather

// Define the custom event

const customEvent = new Event('myCustomEvent');

// Dispatch the custom event to trigger it in File B

document.dispatchEvent(customEvent);[[5]](#endnote-5)

});

// Append the suggestion element to the suggestion list

suggestionList.appendChild(suggestion);[[6]](#endnote-6)

});

}

1. **<ul>** in html is used to create unordered list, “UL” stands for “unordered list” [↑](#endnote-ref-1)
2. **QuerySelector()** method is a built-in method that allows you to select elements in the DOM (Document Object Model) using CSS selector syntax. [↑](#endnote-ref-2)
3. **PreventDefualt()** this built-in function prevent the site from refreshing when button is click [↑](#endnote-ref-3)
4. **<li>** in HTML stands for "list item." It is used to define an individual item within an ordered or unordered list. [↑](#endnote-ref-4)
5. **DispatchEvent()** method is used to manually trigger or dispatch an event on a DOM element. It allows you to simulate user interactions or trigger custom events programmatically. [↑](#endnote-ref-5)
6. **AppendChild()** method is used to add a new child element as the last child of a parent element. It is a DOM (Document Object Model) method that allows you to dynamically manipulate the structure of an HTML document. [↑](#endnote-ref-6)