WD.js

```
const inputBox = document.querySelector(".input-box");
const searchBtn = document.getElementById("searchBtn");
const weather_img = document.querySelector(".weather-img");
const date = document.getElementById("date");
let todayDate = new Date();
const temperature = document.querySelector(".temperature");
const description = document.querySelector(".description");
const humidity = document.getElementById("humidity");
const wind_speed = document.getElementById("wind-speed");
const location_not_found = document.querySelector(".location-not-found");
const weather_body = document.querySelector(".weather-body");

const apikey = "0affb46e7c900cceb8e0871dbee5fb16";
const apiUrl = "https://api.openweathermap.org/data/2.5/weather?units=metric";

// Function to get user's current location and fetch weather data
function getCurrentLocationWeather() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(
      async (position) => {
        // Get latitude and longitude
        const latitude = position.coords.latitude;
        const longitude = position.coords.longitude;

        // Fetch weather data based on user's location
        try {
          const response = await fetch(
            `${apiUrl}&lat=${latitude}&lon=${longitude}&appid=${apikey}`
          );
          const weatherData = await response.json();

          // Update weather information
          updateWeatherInfo(weatherData);
        } catch (error) {
          console.error("Error fetching weather data:", error);
          displayLocationNotFound();
        }
      },
      (error) => {
        console.error("Error getting user's location:", error);
        // If geolocation is denied or not available, use Delhi as default location
        getDefaultLocationWeather();
      }
    );
  } else {
    // If geolocation is not supported, use Delhi as default location
    getDefaultLocationWeather();
  }
}

// Function to fetch weather data for Delhi (default location)
async function getDefaultLocationWeather() {
  try {
    const response = await fetch(`${apiUrl}&q=Delhi&appid=${apikey}`);
    const weatherData = await response.json();

```

```
    // Update weather information
    updateWeatherInfo(weatherData);
  } catch (error) {
    console.error("Error fetching weather data:", error);
    displayLocationNotFound();
  }
}

// Function to handle search button click and fetch weather data for the searched location
async function handleSearch() {
  const city = inputBox.value.trim(); // Trim the input to remove leading/trailing spaces
  if (city !== "") {
    checkWeather(city);
  }
}

// Function to get weather data based on the city name
async function checkWeather(city) {
  try {
    const response = await fetch(`${apiUrl}&q=${city}&appid=${apikey}`);
    const weatherData = await response.json();

    // Update weather information
    updateWeatherInfo(weatherData);
  } catch (error) {
    console.error("Error fetching weather data:", error);
    displayLocationNotFound();
  }
}

// Function to update weather information on the page
function updateWeatherInfo(weatherData) {
  // Update weather information on the page as before
  console.log(weatherData);

  name = weatherData.name;
  updateTable(name);

  date.innerText = dateManage(new Date());
  temperature.innerHTML = `${Math.round(weatherData.main.temp)}°C`;
  description.innerHTML = weatherData.weather[0].description;
  humidity.innerHTML = `${weatherData.main.humidity}%`;
  wind_speed.innerHTML = `${weatherData.wind.speed}Km/H`;

  // Update weather image
  switch (weatherData.weather[0].main) {
    case "Clouds":
      weather_img.src =
"https://res.cloudinary.com/dcladcqtf/image/upload/v1691169151/cloudy.jpeg_pjrq5p.jpg";
      break;
    case "Clear":
      weather_img.src =
"https://res.cloudinary.com/dcladcqtf/image/upload/v1691169205/clear-sky.jpeg_v0sk6s.jpg";
      break;
    case "Rain":
      weather_img.src =
"https://res.cloudinary.com/dcladcqtf/image/upload/v1691169619/rain.jpeg_gwhpsm.jpg";
      break;
    case "Mist":
      weather_img.src =
"https://res.cloudinary.com/dcladcqtf/image/upload/v1691077732/wind_qkwsx8.png";
```

```
      break;
    case "Snow":
      weather_img.src =
"https://res.cloudinary.com/dcladcqtf/image/upload/v1691169577/snowy.jpeg_ucir2z.jpg";
      break;
    case "Haze":
      weather_img.src =
"https://res.cloudinary.com/dcladcqtf/image/upload/v1691169406/haze.jpeg_hilu6q.jpg";
      break;
    case "Thunderstorm":
      weather_img.src =
"https://res.cloudinary.com/dcladcqtf/image/upload/v1691169545/thunder.jpeg_z6to7j.jpg";
      break;
    default:
      weather_img.src =
"https://res.cloudinary.com/dcladcqtf/image/upload/v1691169465/default.jpeg_y12eey.jpg";
  }

  // Update city name on the page
  document.querySelector(".city").innerHTML = weatherData.name;

  updateTable(name);

  // Hide the location not found message
  location_not_found.style.display = "none";
  weather_body.style.display = "flex";
}

let apiurl2 = "https://api.openweathermap.org/data/2.5/forecast?units=metric";

// Function to get weather data based on coordinate
async function updateTable(city) {
  try {
    const response = await fetch(`${apiurl2}&q=${city}&appid=${apikey}`);
    const forecastData = await response.json();
    console.log(forecastData);

    // Update weather information
    console.log("updateTable");
    console.log(forecastData);
    updateWeatherTable(forecastData);
  } catch (error) {
    console.error("Error fetching weather data:", error);
    displayLocationNotFound();
  }
}

// updateTable();

function updateWeatherTable(forecast) {
  console.log(" update weatherupdate Table");
  console.log(forecast);
  const tableBody = document.getElementById("forecastTableBody");

  // Clear existing table rows
  tableBody.innerHTML = "";

  // Days of the week
  const daysOfWeek = [
    "Sunday",
    "Monday",
```

```
    "Tuesday",
    "Wednesday",
    "Thursday",
    "Friday",
    "Saturday",
  ];

  // Loop through forecastData for the first 8 days
  for (let i = 4; i < forecast.list.length; ) {
    const data = forecast.list[i];

    const row = document.createElement("tr");

    const dateCell = document.createElement("td");
    // const datePart = data.dt_txt.split(" ")[0]; // Extract the date part
    const timestamp = new Date(data.dt_txt);
    const dayOfWeek = daysOfWeek[timestamp.getDay()]; // Get the day of the week
    dateCell.textContent = dayOfWeek; // Update with the actual property from your data
    row.appendChild(dateCell);

    // Math.round(data.main.temp)
    const tempCell = document.createElement("td");
    tempCell.textContent = Math.round(data.main.temp); // Update with the actual property
from your data
    row.appendChild(tempCell);

    const windCell = document.createElement("td");
    windCell.textContent = data.wind.speed; // Update with the actual property from your
data
    row.appendChild(windCell);

    const humidityCell = document.createElement("td");
    humidityCell.textContent = data.main.humidity; // Update with the actual property from
your data
    row.appendChild(humidityCell);

    const descCell = document.createElement("td");
    descCell.textContent = data.weather[0].description; // Update with the actual property
from your data
    row.appendChild(descCell);

    // Append the row to the table body
    tableBody.appendChild(row);
    i = i + 8;
  }
}

// Function to display location not found message
function displayLocationNotFound() {
  location_not_found.style.display = "flex";
  weather_body.style.display = "none";
}

// Call the function to get user's current location weather on page load
document.addEventListener("DOMContentLoaded", () => {
  getCurrentLocationWeather();
});

// Call the function to handle search button click
searchBtn.addEventListener("click", handleSearch);

```

```
function dateManage(dateArg) {
  let days = [
    "Sunday",
    "Monday",
    "Tuesday",
    "Wednesday",
    "Thursday",
    "Friday",
    "Saturday",
  ];

  let months = [
    "January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December",
  ];

  //   let year = dateArg.getFULLYear();
  let year = dateArg.getFullYear(); // Corrected method name
  let month = months[dateArg.getMonth()];
  let date = dateArg.getDate();
  let day = days[dateArg.getDay()];

  return `${date} ${month} (${day}), ${year}`;
}

```