

## WD.js

```
1  const inputBox = document.querySelector(".input-box");
2  const searchBtn = document.getElementById("searchBtn");
3  const weather_img = document.querySelector(".weather-img");
4  const date = document.getElementById("date");
5  let todayDate = new Date();
6  const temperature = document.querySelector(".temperature");
7  const description = document.querySelector(".description");
8  const humidity = document.getElementById("humidity");
9  const wind_speed = document.getElementById("wind-speed");
10 const location_not_found = document.querySelector(".location-not-found");
11 const weather_body = document.querySelector(".weather-body");
12
13 const apikey = "0affb46e7c900cceb8e0871dbee5fb16";
14 const apiUrl = "https://api.openweathermap.org/data/2.5/weather?units=metric";
15
16 // Function to get user's current location and fetch weather data
17 function getCurrentLocationWeather() {
18   if (navigator.geolocation) {
19     navigator.geolocation.getCurrentPosition(
20       async (position) => {
21         // Get latitude and longitude
22         const latitude = position.coords.latitude;
23         const longitude = position.coords.longitude;
24
25         // Fetch weather data based on user's location
26         try {
27           const response = await fetch(
28             `${apiUrl}&lat=${latitude}&lon=${longitude}&appid=${apikey}`
29           );
30           const weatherData = await response.json();
31
32           // Update weather information
33           updateWeatherInfo(weatherData);
34         } catch (error) {
35           console.error("Error fetching weather data:", error);
36           displayLocationNotFound();
37         }
38       },
39       (error) => {
40         console.error("Error getting user's location:", error);
41         // If geolocation is denied or not available, use Delhi as default location
42         getDefaultLocationWeather();
43       }
44     );
45   } else {
46     // If geolocation is not supported, use Delhi as default location
47     getDefaultLocationWeather();
48   }
49 }
50
51 // Function to fetch weather data for Delhi (default location)
52 async function getDefaultLocationWeather() {
53   try {
54     const response = await fetch(`${apiUrl}&q=Delhi&appid=${apikey}`);
55     const weatherData = await response.json();
56   }
```

```

57     // Update weather information
58     updateWeatherInfo(weatherData);
59 } catch (error) {
60     console.error("Error fetching weather data:", error);
61     displayLocationNotFound();
62 }
63 }
64
65 // Function to handle search button click and fetch weather data for the searched location
66 async function handleSearch() {
67     const city = inputBox.value.trim(); // Trim the input to remove leading/trailing spaces
68     if (city !== "") {
69         checkWeather(city);
70     }
71 }
72
73 // Function to get weather data based on the city name
74 async function checkWeather(city) {
75     try {
76         const response = await fetch(`${apiUrl}&q=${city}&appid=${apiKey}`);
77         const weatherData = await response.json();
78
79         // Update weather information
80         updateWeatherInfo(weatherData);
81     } catch (error) {
82         console.error("Error fetching weather data:", error);
83         displayLocationNotFound();
84     }
85 }
86
87 // Function to update weather information on the page
88 function updateWeatherInfo(weatherData) {
89     // Update weather information on the page as before
90     console.log(weatherData);
91
92     name = weatherData.name;
93     updateTable(name);
94
95     date.innerText = dateManage(new Date());
96     temperature.innerHTML = `${Math.round(weatherData.main.temp)}°C`;
97     description.innerHTML = weatherData.weather[0].description;
98     humidity.innerHTML = `${weatherData.main.humidity}%`;
99     wind_speed.innerHTML = `${weatherData.wind.speed}Km/H`;
100
101     // Update weather image
102     switch (weatherData.weather[0].main) {
103         case "Clouds":
104             weather_img.src =
105 "https://res.cloudinary.com/dcladcqtf/image/upload/v1691169151/cloudy.jpeg_pjrq5p.jpg";
106             break;
107         case "Clear":
108             weather_img.src =
109 "https://res.cloudinary.com/dcladcqtf/image/upload/v1691169205/clear-sky.jpeg_v0sk6s.jpg";
110             break;
111         case "Rain":
112             weather_img.src =
113 "https://res.cloudinary.com/dcladcqtf/image/upload/v1691169619/rain.jpeg_gwhpsm.jpg";
114             break;
115         case "Mist":
116             weather_img.src =
117 "https://res.cloudinary.com/dcladcqtf/image/upload/v1691077732/wind_qkwsx8.png";

```

```

114     break;
115     case "Snow":
116         weather_img.src =
117         "https://res.cloudinary.com/dcladcqtf/image/upload/v1691169577/snowy.jpeg_ucir2z.jpg";
118     break;
119     case "Haze":
120         weather_img.src =
121         "https://res.cloudinary.com/dcladcqtf/image/upload/v1691169406/haze.jpeg_hilu6q.jpg";
122     break;
123     case "Thunderstorm":
124         weather_img.src =
125         "https://res.cloudinary.com/dcladcqtf/image/upload/v1691169545/thunder.jpeg_z6to7j.jpg";
126     break;
127     default:
128         weather_img.src =
129         "https://res.cloudinary.com/dcladcqtf/image/upload/v1691169465/default.jpeg_y12eey.jpg";
130     }
131
132     // Update city name on the page
133     document.querySelector(".city").innerHTML = weatherData.name;
134
135     updateTable(name);
136
137     // Hide the location not found message
138     location_not_found.style.display = "none";
139     weather_body.style.display = "flex";
140 }
141
142 let apiurl2 = "https://api.openweathermap.org/data/2.5/forecast?units=metric";
143
144 // Function to get weather data based on coordinate
145 async function updateTable(city) {
146     try {
147         const response = await fetch(`${apiurl2}&q=${city}&appid=${apikey}`);
148         const forecastData = await response.json();
149         console.log(forecastData);
150
151         // Update weather information
152         console.log("updateTable");
153         console.log(forecastData);
154         updateWeatherTable(forecastData);
155     } catch (error) {
156         console.error("Error fetching weather data:", error);
157         displayLocationNotFound();
158     }
159 }
160
161 // updateTable();
162
163 function updateWeatherTable(forecast) {
164     console.log(" update weatherupdate Table");
165     console.log(forecast);
166     const tableBody = document.getElementById("forecastTableBody");
167
168     // Clear existing table rows
169     tableBody.innerHTML = "";
170
171     // Days of the week
172     const daysOfWeek = [
173         "Sunday",
174         "Monday",

```

```

171     "Tuesday",
172     "Wednesday",
173     "Thursday",
174     "Friday",
175     "Saturday",
176 ];
177
178 // Loop through forecastData for the first 8 days
179 for (let i = 4; i < forecast.list.length; ) {
180     const data = forecast.list[i];
181
182     const row = document.createElement("tr");
183
184     const dateCell = document.createElement("td");
185     // const datePart = data.dt_txt.split(" ")[0]; // Extract the date part
186     const timestamp = new Date(data.dt_txt);
187     const dayOfWeek = daysOfWeek[timestamp.getDay()]; // Get the day of the week
188     dateCell.textContent = dayOfWeek; // Update with the actual property from your data
189     row.appendChild(dateCell);
190
191     // Math.round(data.main.temp)
192     const tempCell = document.createElement("td");
193     tempCell.textContent = Math.round(data.main.temp); // Update with the actual property
from your data
194     row.appendChild(tempCell);
195
196     const windCell = document.createElement("td");
197     windCell.textContent = data.wind.speed; // Update with the actual property from your
data
198     row.appendChild(windCell);
199
200     const humidityCell = document.createElement("td");
201     humidityCell.textContent = data.main.humidity; // Update with the actual property from
your data
202     row.appendChild(humidityCell);
203
204     const descCell = document.createElement("td");
205     descCell.textContent = data.weather[0].description; // Update with the actual property
from your data
206     row.appendChild(descCell);
207
208     // Append the row to the table body
209     tableBody.appendChild(row);
210     i = i + 8;
211 }
212 }
213
214 // Function to display location not found message
215 function displayLocationNotFound() {
216     location_not_found.style.display = "flex";
217     weather_body.style.display = "none";
218 }
219
220 // Call the function to get user's current location weather on page load
221 document.addEventListener("DOMContentLoaded", () => {
222     getCurrentLocationWeather();
223 });
224
225 // Call the function to handle search button click
226 searchBtn.addEventListener("click", handleSearch);
227

```

```
228 function dateManage(dateArg) {
229     let days = [
230         "Sunday",
231         "Monday",
232         "Tuesday",
233         "Wednesday",
234         "Thursday",
235         "Friday",
236         "Saturday",
237     ];
238
239     let months = [
240         "January",
241         "February",
242         "March",
243         "April",
244         "May",
245         "June",
246         "July",
247         "August",
248         "September",
249         "October",
250         "November",
251         "December",
252     ];
253
254     // let year = dateArg.getFULLYear();
255     let year = dateArg.getFullYear(); // Corrected method name
256     let month = months[dateArg.getMonth()];
257     let date = dateArg.getDate();
258     let day = days[dateArg.getDay()];
259
260     return `${date} ${month} (${day}), ${year}`;
261 }
262
```