

WEB-DEV-FINALS

BSCPE-2C

MEMBERS:

JOHN RUSSEL ESPIRITU

JERICO LUMIBAO

KHATE SIBUG

DANIEL E TANGI

MAIN OBJECTIVES:

The main objective of this code is to create a responsive and interactive Weather App that fetches and displays real-time weather information for any city entered by the user. It leverages the OpenWeatherMap API to obtain current weather data, including temperature, weather conditions, humidity, and wind speed. The app updates the UI dynamically, changing the displayed weather information and background image based on the searched city. Additionally, it implements a WebSocket server to facilitate real-time communication between the client and server, enabling the app to broadcast weather updates to all connected clients. This comprehensive setup ensures users receive up-to-date weather information in an engaging and visually appealing manner, with the added capability of real-time data sharing through WebSocket connections.

FEATURES AND FUNCTIONALITIES

This code implements a Weather App with several features and functionalities. The HTML structure provides a responsive design that includes a header, search bar, and weather display card. The CSS styling enhances the visual appeal using background images, gradients, and custom fonts while ensuring content is centered and properly aligned. The JavaScript code integrates with the OpenWeatherMap API to fetch and display real-time weather information, including the city name, temperature, weather icon, description, humidity, and wind speed. The app dynamically updates the weather details and background image based on the user's search input. Additionally, it employs a WebSocket client to send the fetched weather data to a WebSocket server. The server, created using Node.js and the `ws` library, facilitates real-time communication between clients, broadcasting weather updates received from one client to all connected clients. This setup ensures users receive up-to-date weather information with real-time data sharing capabilities, making the app interactive and engaging.

PROGRAM STRUCTURE AND CODE DOCUMENTATION

Program Structures

1. index.html

This HTML structure of the Weather App. It includes:

- **Header:** A header section with the title "Weather App" and an icon.
- **Search Bar:** An input field for users to enter a city name and a button to trigger the search.
- **Weather Display Card:** A card that displays weather information such as the city name, temperature, weather icon, description, humidity, and wind speed.
- **Scripts:** Links to external CSS for styling and JavaScript files for functionality.

2. style.css

This styling for the Weather App. It includes:

- **Global Styles:** General styling for the body, including font settings, background color, and image.
- **Header Styles:** Styles for the header section, including background color and text alignment.
- **Content Styles:** Styles for the content section, centering the weather display card.
- **Card Styles:** Specific styles for the weather display card, including background gradient, padding, border-radius, and margin.
- **Search Bar and Button Styles:** Styles for the search bar and button, including border, padding, background color, and hover effects.
- **Weather Info Styles:** Styles for weather information, including temperature, description, and loading state.

3. script.js

This JavaScript code that handles the app's functionality. It includes:

- **Weather Object:** An object that handles fetching and displaying weather data from the OpenWeatherMap API.
 - **fetchWeather:** A method that fetches weather data based on the city name entered by the user.
 - **displayWeather:** A method that updates the HTML elements with the fetched weather data.
 - **search:** A method that initiates the fetchWeather method when the search button is clicked or the Enter key is pressed.

- **Event Listeners:** Event listeners for the search button and search bar to handle user interactions.
- **WebSocket Client:** Establishes a WebSocket connection to send weather data to a server and handle messages from the server.

4. server.js

This file contains the Node.js code for the WebSocket server. It includes:


- **WebSocket Server Setup:** Creates a WebSocket server on port 8080.
- **Connection Handling:** Logs new client connections and disconnections.
- **Message Handling:** Broadcasts received messages to all connected clients except the sender.
- **Welcome Message:** Sends a welcome message to newly connected clients.
- **Server Running Log:** Logs that the WebSocket server is running.

CODE DOCUMENTATION

Index.html

Style.css

```
1  body {
2    display: flex;
3    flex-direction: column;
4    align-items: center;
5    height: 100vh;
6    margin: 0;
7    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
8    background: #000000;
9    background-image: url('https://source.unsplash.com/1600x900/?landscape');
10   font-size: 120%;
11 }
12
13 .header {
14   width: 100%;
15   padding: 1rem;
16   color: var(--secondary-light);
17   text-align: center;
18   background-color: rgb(76, 164, 231);
19 }
20
21 .page-name i {
22   color: var(--primary-light);
23   margin-right: 1rem;
24 }
25
26 .content {
27   display: flex;
28   justify-content: center;
29   align-items: center;
30   flex-grow: 1;
31   width: 100%;
32 }
33
34 .card {
35   background: linear-gradient(#243ac7, #a59eea);
36   color: white;
37   padding: 2em;
38   border-radius: 30px;
39   width: 100%;
40   max-width: 420px;
41   margin: 1em;
42 }
43
44 .search {
45   display: flex;
46   align-items: center;
47   justify-content: center;
48 }
49
50 button {
51   margin: 0.5em;
52   border-radius: 50%;
53   border: none;
54   height: 44px;
55   width: 44px;
56   outline: none;
57   background: #696597;
58   color: white;
59   cursor: pointer;
60   transition: 0.2s ease-in-out;
61 }
62
63 input.search-bar {
64   border: none;
65   outline: none;
66   padding: 0.4em 1em;
67   border-radius: 24px;
68   background: #010102;
69   color: white;
70   font-family: inherit;
71   font-size: 105%;
72   width: calc(100% - 100px);
73 }
```



```
1  button:hover {
2      background: #7c7c7c6b;
3  }
4
5  h1.temp {
6      margin: 0;
7      margin-bottom: 0.4em;
8  }
9
10 .flex {
11     display: flex;
12     align-items: center;
13 }
14
15 .description {
16     text-transform: capitalize;
17     margin-left: 8px;
18 }
19
20 .weather.loading {
21     visibility: hidden;
22     max-height: 20px;
23     position: relative;
24 }
25
26 .weather.loading:after {
27     visibility: visible;
28     content: "Loading...";
29     color: white;
30     position: absolute;
31     top: 0;
32     left: 20px;
33 }
34
```

script.js

```
1 let weather = {
2   apiKey: "a4e79c6eee21d31c9457c62eb8f2fa41",
3   fetchWeather: function (city) {
4     fetch("https://api.openweathermap.org/data/2.5/weather?q=" + city + "&units=metric&appid=" + this.apiKey)
5       .then((response) => {
6         if (!response.ok) {
7           alert("No weather found.");
8           throw new Error("No weather found.");
9         }
10        return response.json();
11      })
12      .then((data) => this.displayWeather(data));
13  },
14  displayWeather: function (data) {
15    const { name } = data;
16    const { icon, description } = data.weather[0];
17    const { temp, humidity } = data.main;
18    const { speed } = data.wind;
19    document.querySelector(".city").innerText = "Weather in " + name;
20    document.querySelector(".icon").src = "https://openweathermap.org/img/wn/" + icon + ".png";
21    document.querySelector(".description").innerText = description;
22    document.querySelector(".temp").innerText = temp + "°C";
23    document.querySelector(".humidity").innerText = "Humidity: " + humidity + "%";
24    document.querySelector(".wind").innerText = "Wind speed: " + speed + " km/h";
25    document.querySelector(".weather").classList.remove("loading");
26    document.body.style.backgroundColor = url('https://source.unsplash.com/1600x900/?' + name + '');
27
28    socket.send(JSON.stringify({ name, icon, description, temp, humidity, speed }));
29  },
30  search: function () {
31    this.fetchWeather(document.querySelector(".search-bar").value);
32  },
33 };
34
35 document.querySelector(".search button").addEventListener("click", function () {
36   weather.search();
37 });
38
```

```
1 document.querySelector(".search-bar").addEventListener("keyup", function (event) {
2   if (event.key === "Enter") {
3     weather.search();
4   }
5 });
6
7 weather.fetchWeather("Philippines");
8
9
10 const socket = new WebSocket('ws://localhost:8080');
11
12 socket.addEventListener('open', function (event) {
13   console.log('Connected to WebSocket server');
14 });
15
16 socket.addEventListener('message', function (event) {
17   console.log('Message from server ', event.data);
18 });
19
20 socket.addEventListener('close', function (event) {
21   console.log('Disconnected from WebSocket server');
22 });
23
```

Server.js



```
1  const WebSocket = require('ws');
2
3  const server = new WebSocket.Server({ port: 8080 });
4
5  server.on('connection', socket => {
6      console.log('New client connected');
7
8      socket.on('message', message => {
9          console.log(`Received message => ${message}`);
10
11          server.clients.forEach(client => {
12              if (client !== socket && client.readyState === WebSocket.OPEN) {
13                  client.send(message);
14              }
15          });
16      });
17
18      socket.on('close', () => {
19          console.log('Client disconnected');
20      });
21
22      socket.send('Welcome to the WebSocket server!');
23  });
24
25  console.log('WebSocket server is running on ws://localhost:8080');
26
27
```