Vibe Check: A JS Mini Project

Welcome to the Vibe Check! This exercise is designed to help you hone your ES6 JavaScript skills by building a simple, interactive web page. We'll be using modern JavaScript, a third-party API, and Bootstrap to create a visually appealing and dynamic experience. This project is beginner-friendly and broken down into easy-to-follow steps.

Project Overview

We'll be creating a one-page application called "Vibe Check." It will feature a gallery of "vibey" images. When a user clicks on an image, a description of that "vibe" will be displayed. The page will also have a "Weather Vibe" section where users can enter a US zip code to get the current weather, described with a Gen Z twist.

Here's a sneak peek of what we'll be building:

- A responsive image gallery using Bootstrap's grid system.
- Dynamic content that changes based on user interaction.
- A weather feature that fetches data from a third-party API.
- Error handling for a smoother user experience.

Ready to get started? Let's go!

Part 1: Setting Up the HTML Structure

First, we'll create the basic HTML structure for our page. We'll be using Bootstrap to make our page look great with minimal effort. Bootstrap is a popular CSS framework that provides pre-built components and styles.

Step 1: Create the HTML File

Create a new file named `index.html` in your project folder.

Step 2: Add the Basic HTML Boilerplate

Open `index.html` and add the following HTML boilerplate code. This is the standard starting point for any web page.

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Vibe Check</title>
</head>
<body>

</body>
</html>
```

Step 3: Link to Bootstrap CDN

To use Bootstrap, we need to link to its CSS and JavaScript files. We'll use a Content Delivery Network (CDN), which is a fast and easy way to include Bootstrap in our project. Add the following lines inside the `<head>` tag of your `index.html` file.

HTML

```
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css
" rel="stylesheet">
```

And this `<script>` tag just before the closing `</body>` tag:

HTML

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.mi
n.js"></script>
```

For more information on the Bootstrap CDN, you can check out the official Bootstrap documentation.

Step 4: Create the Main Container

Inside the `<body>` tag, let's add a container for our content. This will help us organize our layout.

HTML

```
<div class="container mt-5">
    <h1 class="text-center">Vibe Check</h1>
    <p class="text-center text-muted">What's the vibe today?</p>
</div>
```

Step 5: Add the Image Grid Section

Now, let's create the section for our image grid. We'll use Bootstrap's grid system to create a responsive layout. Add the following code inside the main container, below the header.

HTML

```
<div class="row" id="image-grid">
    <!-- Images will be added here with JavaScript -->
</div>
```

Step 6: Add the Vibe Description Section

Next, we'll add a section to display the vibe description when an image is clicked. Add this code below the image grid.

HTML

```
<div id="vibe-display" class="text-center p-4 my-4 bg-light rounded shadow-sm" style="display: none;">
    <h2 id="vibe-title"></h2>
    <p id="vibe-description"></p>
</div>
```

Step 7: Add the Weather Vibe Section

Finally, let's add the section for our weather feature. This will include an input field for the zip code and a button to fetch the weather. Add this code below the vibe description section.

HTML

```
<div class="row justify-content-center mt-5">
    <div class="col-md-6">
        <h2 class="text-center">Weather Vibe</h2>
        <div class="input-group mb-3">
            <input type="text" id="zip-code-input" class="form-control" placeholder="Enter US Zip Code">
            <button class="btn btn-primary" type="button" id="get-weather-btn">Get Weather</button>
        </div>
        <div id="weather-display" class="text-center p-4 bg-light rounded shadow-sm" style="display: none;">
            <!-- Weather info will be displayed here -->
        </div>
    </div>
</div>
```

Step 8: Create the JavaScript File

Create a new file named `app.js` in the same folder as your `index.html` file. This is where we'll write our JavaScript code.

Step 9: Link the JavaScript File

Now, let's link our `app.js` file to our `index.html` file. Add the following `<script>` tag just before the closing `</body>` tag, *after* the Bootstrap script.

HTML

```
<script src="app.js"></script>
```

Step 10: Cumulative Code

Great job! You've set up the basic HTML structure for our project. If you got lost at any point, here's the complete code for `index.html` so far:

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Vibe Check</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css
" rel="stylesheet">
</head>
<body>
    <div class="container mt-5">
        <h1 class="text-center">Vibe Check</h1>
        <p class="text-center text-muted">What's the vibe today?</p>

        <div class="row" id="image-grid">
            <!-- Images will be added here with JavaScript -->
        </div>

        <div id="vibe-display" class="text-center p-4 my-4 bg-light rounded
shadow-sm" style="display: none;">
            <h2 id="vibe-title"></h2>
            <p id="vibe-description"></p>
        </div>

        <div class="row justify-content-center mt-5">
            <div class="col-md-6">
                <h2 class="text-center">Weather Vibe</h2>
                <div class="input-group mb-3">
                    <input type="text" id="zip-code-input" class="form-
control" placeholder="Enter US Zip Code">
                    <button class="btn btn-primary" type="button" id="get-
```

```
weather-btn">Get Weather</button>
                </div>
                <div id="weather-display" class="text-center p-4 bg-light
rounded shadow-sm" style="display: none;">
                    <!-- Weather info will be displayed here -->
                </div>
            </div>
        </div>
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.mi
n.js"></script>
    <script src="app.js"></script>
</body>
</html>
```

In the next part, we'll start adding some JavaScript to bring our page to life!

Part 2: Bringing the Vibe to Life with JavaScript

Now that our HTML structure is ready, let's dive into JavaScript to make our page interactive. We'll start by defining our "vibe" images and their descriptions, then dynamically add them to the page.

Step 10.5: Prepare Your Images

To use local images, first create a new folder named `images` in your project directory (the same level as `index.html` and `app.js`). Then, download 6 images that represent different "vibes" (e.g., chill, hype, cozy, confused, salty, boujee) and save them into this `images` folder. Name them descriptively, for example: `chill.jpg`, `hype.jpg`, `cozy.jpg`, `confused.jpg`, `salty.jpg`, `boujee.jpg`. You can use any image format (jpg, png) you prefer.

Step 11: Define Vibe Data

Open your `app.js` file. First, let's define an array of objects, where each object represents a "vibe" with an image and a description. Add the following code to `app.js`:

JavaScript

```
const vibes = [
    {
        id: 'chill',
        image: 'images/chill.jpg',
        description: 'Feeling super chill, just vibing with the universe. No
stress, just good energy. Maybe some lo-fi beats in the background.',
        title: 'Chill Vibe'
```

```
    },
    {
        id: 'hype',
        image: 'images/hype.jpg',
        description: 'The energy is high, we are ready to slay! Bring on the
challenges, we are here for it. Main character energy activated.',
        title: 'Hype Vibe'
    },
    {
        id: 'cozy',
        image: 'images/cozy.jpg',
        description: 'Wrapped in a blanket, hot cocoa in hand, watching my
comfort show. Pure cozy vibes, no cap. Self-care is key.',
        title: 'Cozy Vibe'
    },
    {
        id: 'confused',
        image: 'images/confused.jpg',
        description: 'Brain not braining. What is even happening right now?
Send help, or at least a meme. This is giving very confused energy.',
        title: 'Confused Vibe'
    },
    {
        id: 'salty',
        image: 'images/salty.jpg',
        description: 'Someone just tested my patience. I am not okay, I am
salty. The audacity! Periodt.',
        title: 'Salty Vibe'
    },
    {
        id: 'boujee',
        image: 'images/boujee.jpg',
        description: 'Living my best life, feeling expensive and luxurious.
Champagne wishes and caviar dreams. We love to see it.',
        title: 'Boujee Vibe'
    }
];
```

Step 12: Render Images to the DOM

Now, let's write a function to take our `vibes` data and dynamically create HTML elements for each image, adding them to our `image-grid` div. This involves using DOM manipulation methods.

Add the following function to `app.js`:

JavaScript

```
function renderVibes() {
    const imageGrid = document.getElementById('image-grid');
    if (!imageGrid) return; // Safety check

    imageGrid.innerHTML = ''; // Clear existing content

    vibes.forEach(vibe => {
        const colDiv = document.createElement('div');
        colDiv.className = 'col-6 col-md-4 col-lg-2 mb-4'; // Bootstrap
responsive columns

        colDiv.innerHTML = `
            <div class="card h-100 shadow-sm vibe-card" data-vibe-
id="${vibe.id}">
                <img src="${vibe.image}" class="card-img-top"
alt="${vibe.title}">
                <div class="card-body text-center">
                    <h5 class="card-title">${vibe.title}</h5>
                </div>
            </div>
        `;
        imageGrid.appendChild(colDiv);
    });
}
```

- **document.getElementById()** : Used to get a reference to an HTML element by its ID. MDN Link
- **document.createElement()** : Used to create a new HTML element. MDN Link
- **element.className** : Used to set the CSS class of an element. MDN Link
- **element.innerHTML** : Used to set the HTML content inside an element. MDN Link
- **element.appendChild()** : Used to add a new child element to an existing element. MDN Link
- **Array.prototype.forEach()** : Used to iterate over array elements. MDN Link

Step 13: Display Vibe Description on Click

Now, let's make the images clickable. When an image is clicked, we want to display its corresponding description in the `vibe-display` section. We'll use event delegation for this, which is an efficient way to handle events on multiple elements.

Add the following function and event listener to `app.js` :

JavaScript

```
function displayVibe(vibeId) {
    const selectedVibe = vibes.find(vibe => vibe.id === vibeId);
    const vibeDisplay = document.getElementById('vibe-display');
    const vibeTitle = document.getElementById('vibe-title');
    const vibeDescription = document.getElementById('vibe-description');

    if (selectedVibe && vibeDisplay && vibeTitle && vibeDescription) {
        vibeTitle.textContent = selectedVibe.title;
        vibeDescription.textContent = selectedVibe.description;
        vibeDisplay.style.display = 'block'; // Make the div visible
    }
}

// Event listener for the image grid (using event delegation)
document.addEventListener('click', function(event) {
    const vibeCard = event.target.closest('.vibe-card');
    if (vibeCard) {
        const vibeId = vibeCard.dataset.vibeId;
        displayVibe(vibeId);
    }
});
```

- **Event.target.closest()** : Returns the closest ancestor of the current element (or the current element itself) which matches the selectors given in parameters. MDN Link
- **Element.dataset** : Allows access to custom data attributes (e.g., data-vibe-id). MDN Link
- **Array.prototype.find()** : Used to find a specific element in an array. MDN Link
- **Element.style.display** : Used to set the CSS display property of an element. MDN Link

Step 14: Initial Render

Finally, call the renderVibes() function when the page loads to display all the images.

Add this line to the end of your app.js file:

JavaScript

```
renderVibes();
```

Step 15: Cumulative Code

Here's the complete app.js code up to this point:

JavaScript

```
const vibes = [
    {
```

```
        id: 'chill',
        image: 'images/chill.jpg',
        description: 'Feeling super chill, just vibing with the universe. No
stress, just good energy. Maybe some lo-fi beats in the background.',
        title: 'Chill Vibe'
    },
    {
        id: 'hype',
        image: 'images/hype.jpg',
        description: 'The energy is high, we are ready to slay! Bring on the
challenges, we are here for it. Main character energy activated.',
        title: 'Hype Vibe'
    },
    {
        id: 'cozy',
        image: 'images/cozy.jpg',
        description: 'Wrapped in a blanket, hot cocoa in hand, watching my
comfort show. Pure cozy vibes, no cap. Self-care is key.',
        title: 'Cozy Vibe'
    },
    {
        id: 'confused',
        image: 'images/confused.jpg',
        description: 'Brain not braining. What is even happening right now?
Send help, or at least a meme. This is giving very confused energy.',
        title: 'Confused Vibe'
    },
    {
        id: 'salty',
        image: 'images/salty.jpg',
        description: 'Someone just tested my patience. I am not okay, I am
salty. The audacity! Periodt.',
        title: 'Salty Vibe'
    },
    {
        id: 'boujee',
        image: 'images/boujee.jpg',
        description: 'Living my best life, feeling expensive and luxurious.
Champagne wishes and caviar dreams. We love to see it.',
        title: 'Boujee Vibe'
    }
];

function renderVibes() {
    const imageGrid = document.getElementById('image-grid');
    if (!imageGrid) return;

    imageGrid.innerHTML = '';
```

```
    vibes.forEach(vibe => {
        const colDiv = document.createElement('div');
        colDiv.className = 'col-6 col-md-4 col-lg-2 mb-4';

        colDiv.innerHTML = `
            <div class="card h-100 shadow-sm vibe-card" data-vibe-
id="${vibe.id}">
                <img src="${vibe.image}" class="card-img-top"
alt="${vibe.title}">
                <div class="card-body text-center">
                    <h5 class="card-title">${vibe.title}</h5>
                </div>
            </div>
        `;
        imageGrid.appendChild(colDiv);
    });
}

function displayVibe(vibeId) {
    const selectedVibe = vibes.find(vibe => vibe.id === vibeId);
    const vibeDisplay = document.getElementById('vibe-display');
    const vibeTitle = document.getElementById('vibe-title');
    const vibeDescription = document.getElementById('vibe-description');

    if (selectedVibe && vibeDisplay && vibeTitle && vibeDescription) {
        vibeTitle.textContent = selectedVibe.title;
        vibeDescription.textContent = selectedVibe.description;
        vibeDisplay.style.display = 'block';
    }
}

document.addEventListener('click', function(event) {
    const vibeCard = event.target.closest('.vibe-card');
    if (vibeCard) {
        const vibeId = vibeCard.dataset.vibeId;
        displayVibe(vibeId);
    }
});

renderVibes();
```

In the next part, we'll integrate a third-party weather API to fetch and display weather information!

Part 3: Getting the Weather Vibe (API Integration)

Now for the exciting part: integrating a third-party API to fetch real-time weather data! We'll use the OpenWeatherMap API for this. You'll need to sign up for a free API key.

Step 16: Get an API Key

1. Go to OpenWeatherMap.
2. Sign up for a free account.
3. Once logged in, navigate to the "API keys" tab to find your personal API key. Keep this key private!

Step 17: Define API Key and Base URL

In your `app.js` file, add your API key and the base URL for the weather API. Replace `YOUR_API_KEY_HERE` with the actual key you obtained.

JavaScript

```
const OPENWEATHER_API_KEY = 'YOUR_API_KEY_HERE'; // Replace with your actual API key
const OPENWEATHER_BASE_URL = 'https://api.openweathermap.org/data/2.5/weather';
```

Step 18: Implement `fetchWeather` Function

We'll create an asynchronous function `fetchWeather` that takes a zip code, makes a request to the OpenWeatherMap API, and handles the response. This function will use the `fetch` API, which is a modern way to make network requests in JavaScript.

Add the following function to `app.js`:

JavaScript

```
async function fetchWeather(zipCode) {
    const weatherDisplay = document.getElementById('weather-display');
    if (!weatherDisplay) return;

    weatherDisplay.style.display = 'none'; // Hide previous weather info
    weatherDisplay.innerHTML = ''; // Clear previous weather info

    try {
        // Construct the API URL
        const url = `${OPENWEATHER_BASE_URL}?zip=${zipCode},us&appid=${OPENWEATHER_API_KEY}&units=imperial`;

        // Make the API request using fetch
        const response = await fetch(url);

        // Check if the response was successful (status code 200-299)
```

```
        if (!response.ok) {
            // If not successful, throw an error with the status text
            throw new Error(`HTTP error! status: ${response.status}`);
        }

        // Parse the JSON response
        const data = await response.json();

        // Display the weather data
        renderWeather(data);

    } catch (error) {
        // Catch and handle any errors during the fetch operation
        console.error('Fetch error:', error);
        weatherDisplay.style.display = 'block';
        weatherDisplay.innerHTML = `<p class="text-danger">Oops! Couldn't
get the weather vibe. Check your zip code or API key. Error: ${error.message}
</p>`;
    }
}
```

- **async** / **await** : Modern JavaScript syntax for handling asynchronous operations, making them look and behave more like synchronous code. MDN Link
- **fetch()** ** API**: A browser API for making network requests. It returns a Promise that resolves to the `Response` to that request. MDN Link
- **Response.ok** : A boolean indicating if the HTTP response status is in the 200-299 range. MDN Link
- **Response.json()** : Parses the response body as JSON. Returns a Promise that resolves with the JavaScript object. MDN Link
- **try...catch** : A statement that allows you to test a block of code for errors while it is being executed. MDN Link

Step 19: Implement `renderWeather` Function

This function will take the weather data received from the API and display it in our `weather-display` div with a Gen Z twist.

Add the following function to `app.js` :

JavaScript

```
function renderWeather(data) {
    const weatherDisplay = document.getElementById('weather-display');
    if (!weatherDisplay) return;
```

```
    const temp = Math.round(data.main.temp);
    const description = data.weather[0].description;
    const city = data.name;

    let vibeMessage = '';
    if (temp < 32) {
        vibeMessage = `It's giving arctic chill in ${city}. Bundle up,
bestie!`;
    } else if (temp < 50) {
        vibeMessage = `Low-key chilly in ${city}. Grab a hoodie, no cap.`;
    } else if (temp < 70) {
        vibeMessage = `The weather in ${city} is pretty mid. Perfect for a
cozy vibe.`;
    } else if (temp < 85) {
        vibeMessage = `It's a whole vibe in ${city}! Sun's out, good times
ahead.`;
    } else {
        vibeMessage = `Sizzling hot in ${city}! Stay hydrated, fam.`;
    }

    weatherDisplay.innerHTML = `
        <h3>${city}</h3>
        <p class="lead">${temp}°F - ${description}</p>
        <p>${vibeMessage}</p>
    `;
    weatherDisplay.style.display = 'block'; // Make the div visible
}
```

Step 20: Attach Event Listener to Weather Button

Finally, we need to call our `fetchWeather` function when the "Get Weather" button is clicked. We'll also add an event listener for the Enter key on the input field.

Add the following event listeners to `app.js`:

JavaScript

```
const getWeatherBtn = document.getElementById('get-weather-btn');
const zipCodeInput = document.getElementById('zip-code-input');

if (getWeatherBtn && zipCodeInput) {
    getWeatherBtn.addEventListener('click', () => {
        const zipCode = zipCodeInput.value.trim();
        if (zipCode) {
            fetchWeather(zipCode);
        } else {
            alert('Please enter a valid US Zip Code!');
        }
```

```
    });

    zipCodeInput.addEventListener('keypress', (event) => {
        if (event.key === 'Enter') {
            getWeatherBtn.click(); // Simulate a click on the button
        }
    });
}
```

- **EventTarget.addEventListener()** : Attaches an event handler to the specified element. MDN Link
- **Element.value** : Gets or sets the value of an input element. MDN Link
- **String.prototype.trim()** : Removes whitespace from both ends of a string. MDN Link

Step 21: Cumulative Code

Here's the complete `app.js` code up to this point, including the weather API integration:

JavaScript

```
const vibes = [
    {
        id: 'chill',
        image: 'images/chill.jpg',
        description: 'Feeling super chill, just vibing with the universe. No
stress, just good energy. Maybe some lo-fi beats in the background.',
        title: 'Chill Vibe'
    },
    {
        id: 'hype',
        image: 'images/hype.jpg',
        description: 'The energy is high, we are ready to slay! Bring on the
challenges, we are here for it. Main character energy activated.',
        title: 'Hype Vibe'
    },
    {
        id: 'cozy',
        image: 'images/cozy.jpg',
        description: 'Wrapped in a blanket, hot cocoa in hand, watching my
comfort show. Pure cozy vibes, no cap. Self-care is key.',
        title: 'Cozy Vibe'
    },
    {
        id: 'confused',
        image: 'images/confused.jpg',
        description: 'Brain not braining. What is even happening right now?
Send help, or at least a meme. This is giving very confused energy.',
```

```
        title: 'Confused Vibe'
    },
    {
        id: 'salty',
        image: 'images/salty.jpg',
        description: 'Someone just tested my patience. I am not okay, I am
salty. The audacity! Periodt.',
        title: 'Salty Vibe'
    },
    {
        id: 'boujee',
        image: 'images/boujee.jpg',
        description: 'Living my best life, feeling expensive and luxurious.
Champagne wishes and caviar dreams. We love to see it.',
        title: 'Boujee Vibe'
    }
];

const OPENWEATHER_API_KEY = 'YOUR_API_KEY_HERE'; // Replace with your actual
API key
const OPENWEATHER_BASE_URL =
'https://api.openweathermap.org/data/2.5/weather';

function renderVibes() {
    const imageGrid = document.getElementById('image-grid');
    if (!imageGrid) return;

    imageGrid.innerHTML = '';

    vibes.forEach(vibe => {
        const colDiv = document.createElement('div');
        colDiv.className = 'col-6 col-md-4 col-lg-2 mb-4';

        colDiv.innerHTML = `
            <div class="card h-100 shadow-sm vibe-card" data-vibe-
id="${vibe.id}">
                <img src="${vibe.image}" class="card-img-top"
alt="${vibe.title}">
                <div class="card-body text-center">
                    <h5 class="card-title">${vibe.title}</h5>
                </div>
            </div>
        `;
        imageGrid.appendChild(colDiv);
    });
}

function displayVibe(vibeId) {
```

```
    const selectedVibe = vibes.find(vibe => vibe.id === vibeId);
    const vibeDisplay = document.getElementById('vibe-display');
    const vibeTitle = document.getElementById('vibe-title');
    const vibeDescription = document.getElementById('vibe-description');

    if (selectedVibe && vibeDisplay && vibeTitle && vibeDescription) {
        vibeTitle.textContent = selectedVibe.title;
        vibeDescription.textContent = selectedVibe.description;
        vibeDisplay.style.display = 'block';
    }
}

async function fetchWeather(zipCode) {
    const weatherDisplay = document.getElementById('weather-display');
    if (!weatherDisplay) return;

    weatherDisplay.style.display = 'none';
    weatherDisplay.innerHTML = '';

    try {
        const url = `${OPENWEATHER_BASE_URL}?
zip=${zipCode},us&appid=${OPENWEATHER_API_KEY}&units=imperial`;
        const response = await fetch(url);

        if (!response.ok) {
            throw new Error(`HTTP error! status: ${response.status}`);
        }

        const data = await response.json();
        renderWeather(data);

    } catch (error) {
        console.error('Fetch error:', error);
        weatherDisplay.style.display = 'block';
        weatherDisplay.innerHTML = `<p class="text-danger">Oops! Couldn't
get the weather vibe. Check your zip code or API key. Error: ${error.message}
</p>`;
    }
}

function renderWeather(data) {
    const weatherDisplay = document.getElementById('weather-display');
    if (!weatherDisplay) return;

    const temp = Math.round(data.main.temp);
    const description = data.weather[0].description;
    const city = data.name;
```

```
    let vibeMessage = '';
    if (temp < 32) {
        vibeMessage = `It's giving arctic chill in ${city}. Bundle up,
bestie!`;
    } else if (temp < 50) {
        vibeMessage = `Low-key chilly in ${city}. Grab a hoodie, no cap.`;
    } else if (temp < 70) {
        vibeMessage = `The weather in ${city} is pretty mid. Perfect for a
cozy vibe.`;
    } else if (temp < 85) {
        vibeMessage = `It's a whole vibe in ${city}! Sun's out, good times
ahead.`;
    } else {
        vibeMessage = `Sizzling hot in ${city}! Stay hydrated, fam.`;
    }

    weatherDisplay.innerHTML = `
        <h3>${city}</h3>
        <p class="lead">${temp}°F - ${description}</p>
        <p>${vibeMessage}</p>
    `;
    weatherDisplay.style.display = 'block';
}

document.addEventListener('click', function(event) {
    const vibeCard = event.target.closest('.vibe-card');
    if (vibeCard) {
        const vibeId = vibeCard.dataset.vibeId;
        displayVibe(vibeId);
    }
});

const getWeatherBtn = document.getElementById('get-weather-btn');
const zipCodeInput = document.getElementById('zip-code-input');

if (getWeatherBtn && zipCodeInput) {
    getWeatherBtn.addEventListener('click', () => {
        const zipCode = zipCodeInput.value.trim();
        if (zipCode) {
            fetchWeather(zipCode);
        } else {
            alert('Please enter a valid US Zip Code!');
        }
    });

    zipCodeInput.addEventListener('keypress', (event) => {
        if (event.key === 'Enter') {
            getWeatherBtn.click();
```

```
        }
    });
}

renderVibes();
```

In the final part, we'll refine the UI and add some finishing touches to make our Vibe Check truly ✨ aesthetic ✨.

Part 4: Aesthetic Touches and Final Polish

Our Vibe Check app is functional, but let's give it some extra ✨ aesthetic ✨ to truly capture that Gen Z essence. We'll add a small custom CSS file to tweak some Bootstrap styles and ensure our vibe is on point.

Step 22: Create a Custom CSS File

Create a new file named `style.css` in your project folder.

Step 23: Link the Custom CSS File

Open `index.html` and add the following line inside the `<head>` tag, *after* the Bootstrap CSS link. This ensures our custom styles can override Bootstrap's defaults.

HTML

```
<link rel="stylesheet" href="style.css">
```

Step 24: Add Custom Styles

Open `style.css` and add the following CSS rules. These styles will enhance the look of our vibe cards and the weather display.

CSS

```
body {
    background-color: #f8f9fa; /* Light gray background */
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; /* Modern
font */
}

.vibe-card {
    transition: transform 0.2s ease-in-out, box-shadow 0.2s ease-in-out;
    cursor: pointer;
    border-radius: 15px;
    overflow: hidden;
}
```

```css
.vibe-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.15);
}

.vibe-card img {
    height: 150px;
    object-fit: cover;
    border-bottom: 1px solid #eee;
}

#vibe-display, #weather-display {
    border-radius: 15px;
    background-color: #ffffff;
    border: 1px solid #e0e0e0;
}

#vibe-display h2, #weather-display h2, #weather-display h3 {
    color: #6f42c1; /* Bootstrap purple */
    font-weight: 600;
}

.input-group .form-control {
    border-top-left-radius: 15px;
    border-bottom-left-radius: 15px;
}

.input-group .btn {
    border-top-right-radius: 15px;
    border-bottom-right-radius: 15px;
    background-color: #6f42c1;
    border-color: #6f42c1;
}

.input-group .btn:hover {
    background-color: #5a36a0;
    border-color: #5a36a0;
}

.text-center.text-muted {
    font-style: italic;
    color: #6c757d !important;
}
```

Step 25: Cumulative Code

Here's the complete `index.html` file with the custom CSS link:

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Vibe Check</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css
" rel="stylesheet">
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container mt-5">
        <h1 class="text-center">Vibe Check</h1>
        <p class="text-center text-muted">What's the vibe today?</p>

        <div class="row" id="image-grid">
            <!-- Images will be added here with JavaScript -->
        </div>

        <div id="vibe-display" class="text-center p-4 my-4 bg-light rounded
shadow-sm" style="display: none;">
            <h2 id="vibe-title"></h2>
            <p id="vibe-description"></p>
        </div>

        <div class="row justify-content-center mt-5">
            <div class="col-md-6">
                <h2 class="text-center">Weather Vibe</h2>
                <div class="input-group mb-3">
                    <input type="text" id="zip-code-input" class="form-
control" placeholder="Enter US Zip Code">
                    <button class="btn btn-primary" type="button" id="get-
weather-btn">Get Weather</button>
                </div>
                <div id="weather-display" class="text-center p-4 bg-light
rounded shadow-sm" style="display: none;">
                    <!-- Weather info will be displayed here -->
                </div>
            </div>
        </div>
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.mi
n.js"></script>
    <script src="app.js"></script>
```

```
</body>
</html>
```

And here's the complete `style.css` file:

CSS

```css
body {
    background-color: #f8f9fa; /* Light gray background */
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; /* Modern
font */
}

.vibe-card {
    transition: transform 0.2s ease-in-out, box-shadow 0.2s ease-in-out;
    cursor: pointer;
    border-radius: 15px;
    overflow: hidden;
}

.vibe-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.15);
}

.vibe-card img {
    height: 150px;
    object-fit: cover;
    border-bottom: 1px solid #eee;
}

#vibe-display, #weather-display {
    border-radius: 15px;
    background-color: #ffffff;
    border: 1px solid #e0e0e0;
}

#vibe-display h2, #weather-display h2, #weather-display h3 {
    color: #6f42c1; /* Bootstrap purple */
    font-weight: 600;
}

.input-group .form-control {
    border-top-left-radius: 15px;
    border-bottom-left-radius: 15px;
}

.input-group .btn {
```

```
    border-top-right-radius: 15px;
    border-bottom-right-radius: 15px;
    background-color: #6f42c1;
    border-color: #6f42c1;
}

.input-group .btn:hover {
    background-color: #5a36a0;
    border-color: #5a36a0;
}

.text-center.text-muted {
    font-style: italic;
    color: #6c757d !important;
}
```

Conclusion

Congratulations! You've successfully built the "Vibe Check" application. You've learned how to:

- Set up a responsive HTML structure using Bootstrap.
- Dynamically render content to the DOM using JavaScript.
- Handle user interactions with event listeners.
- Integrate a third-party API (`fetch`) to retrieve external data.
- Implement basic error handling for network requests.
- Apply custom CSS to enhance the user interface.

This project covered essential ES6 JavaScript features and modern web development practices. Keep experimenting, keep building, and keep those good vibes flowing! ✨

References

- Bootstrap Documentation
- OpenWeatherMap API
- MDN Web Docs - Document.getElementById()
- MDN Web Docs - Document.createElement()
- MDN Web Docs - Element.className
- MDN Web Docs - Element.innerHTML
- MDN Web Docs - Element.textContent

- MDN Web Docs - Node.appendChild()
- MDN Web Docs - Array.prototype.forEach()
- MDN Web Docs - Event.target.closest()
- MDN Web Docs - HTMLElement.dataset
- MDN Web Docs - Array.prototype.find()
- MDN Web Docs - HTMLElement.style
- MDN Web Docs - async function
- MDN Web Docs - Fetch API
- MDN Web Docs - Response.ok
- MDN Web Docs - Response.json()
- MDN Web Docs - try...catch
- MDN Web Docs - EventTarget.addEventListener()
- MDN Web Docs - HTMLInputElement.value
- MDN Web Docs - String.prototype.trim()