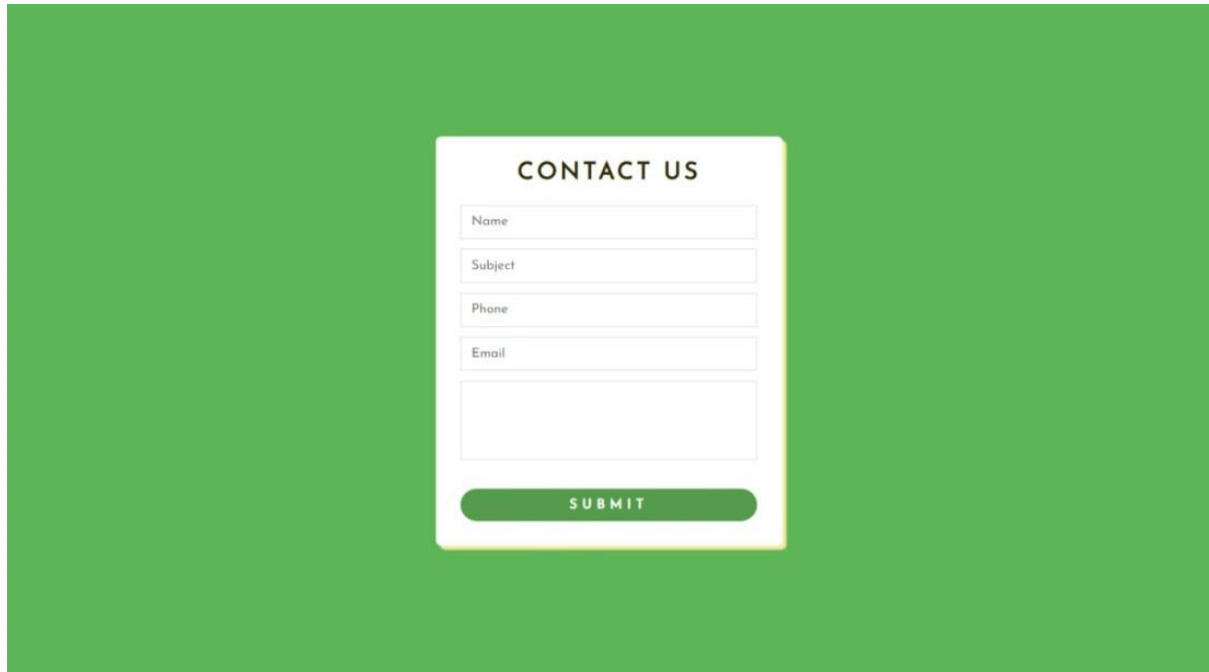


Create a Contact Form using HTML, CSS & JavaScript

A contact form is a web form used to collect user information and messages, facilitating communication between visitors and site owners. It is essential for feedback, inquiries, and support. Create a contact form using HTML for structure, CSS for styling, and JavaScript for validation and submission.

Preview Image:

A preview image of a contact form titled "CONTACT US" centered on a solid green background. The form is a white rectangular box with a thin yellow border. It contains four labeled input fields: "Name", "Subject", "Phone", and "Email", each with a light gray border. Below these is a larger, empty text area. At the bottom of the form is a green, rounded rectangular button with the word "SUBMIT" in white, uppercase letters.

Prerequisites

- [HTML](#)
- [CSS](#)
- [JavaScript](#)

Approach

- Step 1: Create a HTML structure for the contact form component having proper id and classes for the styles.
- Step 2: Add a CSS file which contains all the styles related to the contact form component.
- Step 3: Add a JavaScript file having all the logic for validation and boundary checks.
- Form Validations Conditions

- Length of name should be 5 or more.
- Length of subject should be 10 or more.
- Length of number should be 10.
- Email must include @ and length more than 6.
- Message length must be 40 characters.

Then, link the JavaScript and CSS file to the HTML file.

Example: This example describes the basic implementation of the contact form using **HTML, CSS, and JavaScript**.

Step 1: HTML File :

Creating the HTML for the Contact Form

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta name="viewport"
```

```
    content="width=device-width, initial-scale=1.0">
```

```
  <title>Contact Form</title>
```

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

```
</head>
```

```
<body>
```

```
  <div class="contact-form-container">
```

```
    <h2>Contact Us</h2>
```

```
    <form id="contactForm">
```

```
      <div class="form-group">
```

```
        <label for="name">Name</label>
```

```
        <input type="text" id="name"
```

```
        name="name"

        placeholder="Your Name" required>

        <span class="error-message" id="nameError"></span>

    </div>

    <div class="form-group">

        <label for="email">Email</label>

        <input type="email" id="email"

            name="email"

            placeholder="Your Email" required>

            <span class="error-message" id="emailError"></span>

        </div>

        <div class="form-group">

            <label for="phone">Phone</label>

            <input type="tel" id="phone"

                name="phone"

                placeholder="Your Phone Number" required>

                <span class="error-message" id="phoneError"></span>

            </div>

            <div class="form-group">

                <label for="message">Message</label>

                <textarea id="message"

                    name="message"

                    placeholder="Your Message"

                    rows="5" required></textarea>

                <span class="error-message"

                    id="messageError"></span>

            </div>

            <button type="submit">
```

```

        class="submit-button">

        Send Message

    </button>

</form>

</div>

<script src="scripts.js"></script>

</body>

</html>

```

Step 2: CSS file: Adding CSS for Basic Styling

Create a styles.css file to make the form look nicer:

```

/* styles.css */

body { font-family: Arial, sans-serif; padding: 20px; background-color: #f4f4f4; } form {
background: white; padding: 20px; border-radius: 8px; box-shadow: 0 0 10px
rgba(0,0,0,0.1); } .form-group { margin-bottom: 15px; } label { display: block; margin-
bottom: 5px; } input[type="text"], input[type="email"], textarea { width: 100%; padding:
8px; border: 1px solid #ccc; border-radius: 4px; } button { background-color: #0056b3;
color: white; padding: 10px 15px; border: none; border-radius: 5px; cursor: pointer; }
button:hover { background-color: #004494; }

```

Step 3: Client-Side JavaScript for Basic Validation

```

// scripts.js

document.getElementById('contactForm').addEventListener('submit', function(event) {

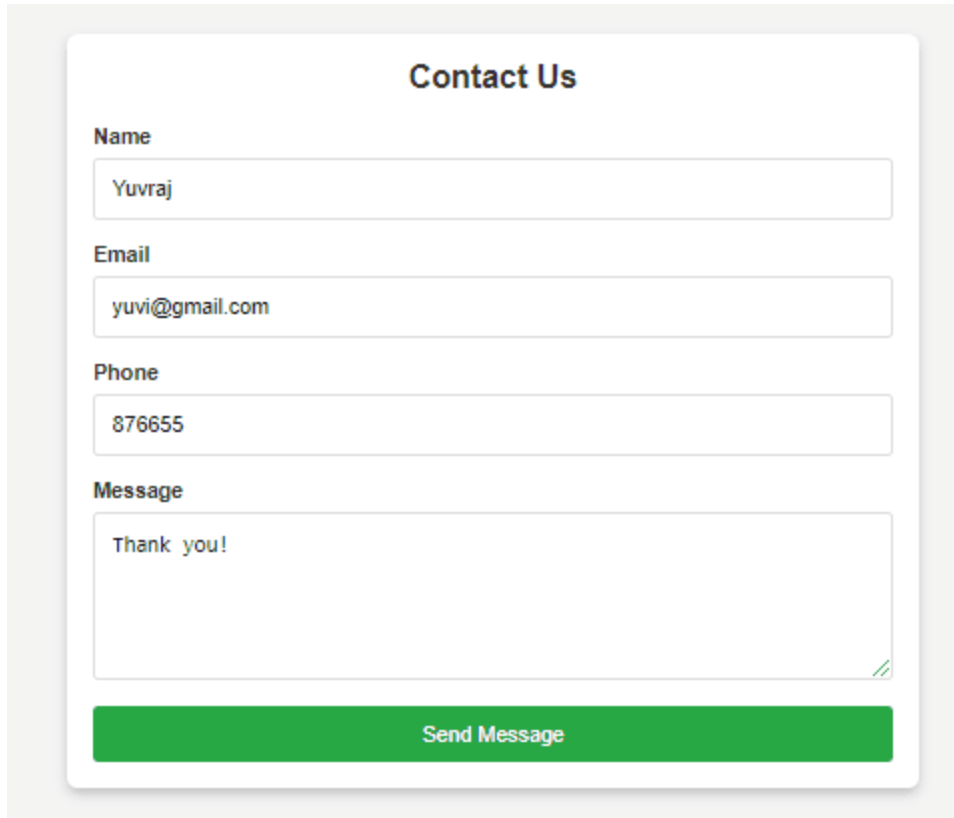
    var name = document.getElementById('name').value; var email =
document.getElementById('email').value; var message =
document.getElementById('message').value; if (!name || !email || !message) {

    alert('Please fill in all fields. '); event.preventDefault(); // Prevent form from submitting }
});

```

Above all are the basic setup for the front-end part of your contact form. Before we move on to setting up the server side with Express and SQLite, do you have any modifications or additions you'd like to make to this part? If everything looks good, we'll proceed with.

Output:



Contact Us

Name
Yuvraj

Email
yuvi@gmail.com

Phone
876655

Message
Thank you!

Send Message

Server Side :

Step 4: Setting Up Your Node.js and Express Server

First, ensure you have Node.js installed. You can download it from nodejs.org if you haven't done so. After installing Node.js, you'll need to set up your project and install the necessary packages.

1.Initialize a new Node.js project:

- Open your terminal or command prompt.
- Navigate to your project directory.

2. Install Express and other required packages:

In the same directory, run the following command to install Express, body-parser (to handle POST data), and sqlite3:

3. Create your server file:

Create a file named **index.js** in your project directory.

Add the following code to set up a basic Express server:

```
const express = require('express'); const bodyParser = require('body-parser'); const
sqlite3 = require('sqlite3').verbose(); const app = express(); const port = 5000; // Port as
specified in your brief // Middlewares app.use(bodyParser.urlencoded({ extended: true
})); app.use(express.static('public')); // To serve static files like CSS and JS // Database
setup const db = new sqlite3.Database('./database.db', sqlite3.OPEN_READWRITE |
sqlite3.OPEN_CREATE, (err) => { if (err) { console.error("Error opening database " +
err.message); } else { db.run('CREATE TABLE IF NOT EXISTS contacts (id INTEGER
PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, email TEXT NOT NULL,
message TEXT NOT NULL)', (err) => { if (err) { console.error("Error creating table " +
err.message); } }); } }); // Routes app.get('/', (req, res) => { res.sendFile(__dirname +
'/index.html'); // Ensure this points to your actual HTML file location });
app.post('/submit-form', (req, res) => { const { name, email, message } = req.body;
db.run('INSERT INTO contacts (name, email, message) VALUES (?, ?, ?)', [name, email,
message], function(err) { if (err) { return console.error(err.message); } console.log(` A row
has been inserted with rowid ${this.lastID}` ); res.send("Thank you for your
message!"); }); }); // Start server app.listen(port, () => { console.log(` Server running on
http://localhost:${port}` ); });
```

Step 5: Update the Form to Submit to the Server

Modify your HTML form in **index.html** to send data to the server:

```
<form id="contactForm" action="/submit-form" method="POST">
```

This change will ensure the form submits data to your Express server's route **/submit-form**.

What's Next

- **Testing the Server:** Make sure your server handles the form submission correctly and stores the data in the SQLite database.

- **Security and Authentication:** After testing, we can discuss adding basic authentication to your application.

Please go ahead and implement the server setup. Let me know if everything works correctly or if you run into any issues!

To properly test your new server setup with the contact form submission, follow these steps to ensure everything works correctly:

1. Run Your Server

- Open your terminal or command prompt.
- Navigate to your project directory where your **index.js** file is located.
- Run the server by typing **node index.js**. You should see a message like "Server running on http://localhost:5000" indicating that the server is active.

2. Access the Form in Your Browser

- Open a web browser and go to **http://localhost:5000**. This should display your contact form.
- Fill out the form and hit submit.

3. Check the Server Response

- After submitting the form, you should receive a response from the server, such as "Thank you for your message!" as indicated in the **index.js** route setup for **/submit-form**.
- Watch the terminal for any console messages or errors. If a row is successfully inserted, you should see a console log stating, "A row has been inserted with rowid X" where X is the row ID.

4. Verify Data Insertion in SQLite Database

To check that data is actually being stored in the SQLite database, you can use a SQLite database viewer or command line tools. Here's how you can check using the SQLite command line:

- Open your terminal or command prompt.
- Connect to your SQLite database by navigating to your project directory and typing:

```
sqlite3 database.db
```

- Once connected, you can run SQL queries. To view all entries in the **contacts** table, type:

```
SELECT * FROM contacts;
```

- This should display all the records you've inserted through the contact form.

Troubleshooting Common Issues

If you encounter issues, here are a few things to check:

- **Server not starting:** Ensure there are no syntax errors in your **index.js**. Also, check that all required packages (**express**, **body-parser**, **sqlite3**) are installed.
- **Form not submitting:** Make sure the form action is set to **/submit-form** and method is **POST**. Also, ensure your server is running while you're testing.
- **Data not saving:** Check for errors in your SQL statements and ensure the database file (**database.db**) has appropriate write permissions.
- **No response or errors in browser:** Check the browser's developer console for any client-side errors and ensure the server logs don't show any issues handling the request.

Once you confirm that everything is working as expected, we can proceed with adding authentication to your server setup. Let me know how the testing goes or if you need help with any specific part!

To help you troubleshoot the error effectively, it would be helpful to know more details about what specific error message you received. Errors can stem from several sources, such as configuration issues, syntax errors, or problems with package installations.

Here are some common areas to check and questions that might help narrow down the issue:

Common Error Sources:

- Server Doesn't Start: Is there an error message in the terminal when you try to start the server with **node index.js**?
- Common issues might include syntax errors in your code or problems with Node.js not being properly installed.
-
- Database Connection Issues: If the error is related to SQLite, what does the error message say?
- Possible issues could be related to the way the database is opened or SQL syntax errors.
-
- Form Submission Errors: Are there any error messages in the browser console when you submit the form?
- Network tab in the browser's developer tools: What status code do you get when you submit the form? Is the server returning an error response?
-
- Package Dependencies: Are all required packages installed? You can check if **express**, **body-parser**, and **sqlite3** are listed in your **package.json** under **dependencies**.
- Run **npm install** again to make sure all dependencies are properly installed.

Error Details:

Could you provide:

- The exact error message you see in your terminal or console.
- The part of the code where the error seems to be occurring, if indicated by the error message.
- Any additional details about what you were trying to do when the error occurred