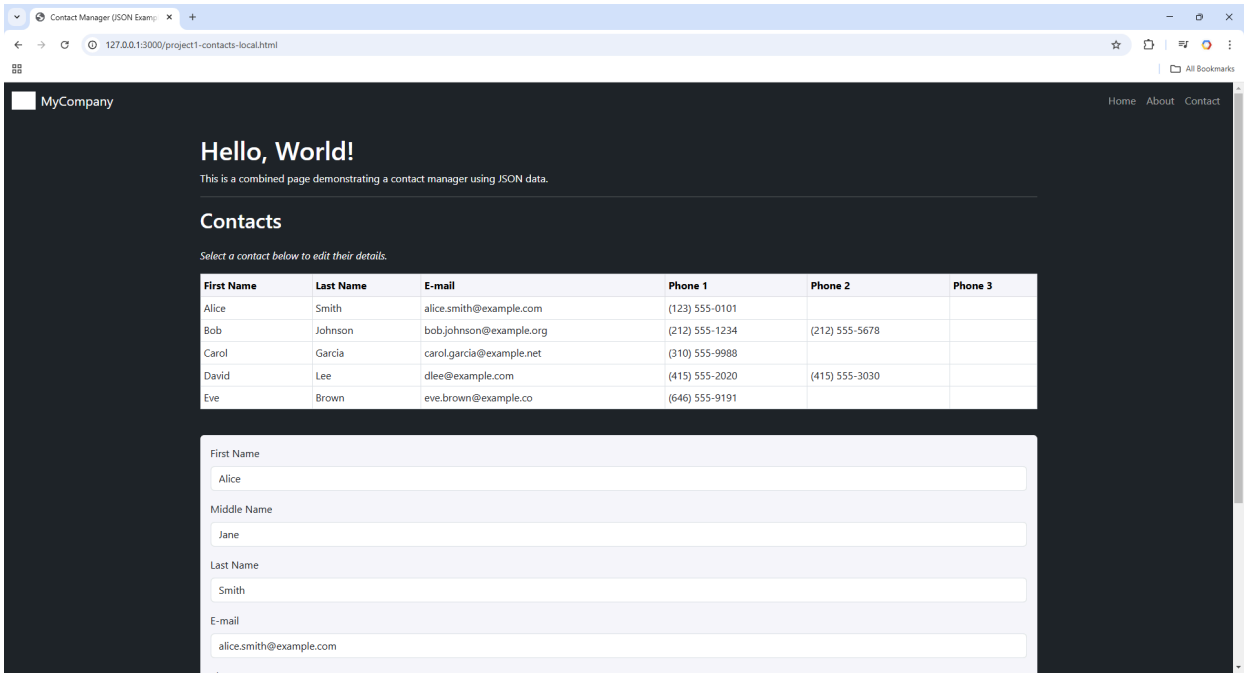
collegejavascript.com

Project1-Contacts
index.html
HTML/CSS/JS Page - index.html
Bootstrap 5.3




```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Contact Manager (JSON Example)</title>

  <!--
    BOOTSTRAP 5.3 CSS
    Using Bootstrap to style the page with a dark theme for the background
  -->
  <link
    rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
  />

  <!--
```

```
    STYLES
    Setting body to dark mode and giving the contacts table a white background
  -->
  <style>
    /* Dark mode background, white text */
    body {
      background-color: #121212;
      color: #ffffff;
    }

    /* Contacts table should have white background, black text */
    .contacts-table {
      background-color: #ffffff;
      color: #000000;
    }

    /* Slightly smaller padding for "table-sm" */
    .table-sm tr th,
    .table-sm tr td {
      padding: 0.3rem;
    }
  </style>

  <!--
    ENVIRONMENT VARIABLES (example)
    You might have configuration data here (e.g., API endpoints).
    In a real app, "ENV" could be used to store environment-specific data.
  -->
  <script>
    var ENV = {
      contactsFile: "contacts.json" // For demonstration; not used in this static example.
    };
  </script>
</head>

<body class="bg-dark text-white">

  <!--
    NAVBAR
    In a real application, you might load this from a separate file
    (e.g., "navbar.html") so any change to the navbar is updated site-wide.
  -->
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container-fluid">
      <!-- Simple Logo or Icon -->
      <svg
        version="1.1"
        id="svg1"
        xmlns="http://www.w3.org/2000/svg"
        x="0px"
        y="0px"
        width="37px"
        height="28px"
        viewBox="252 340 37 28"
        fill="#FFF"
      >
        <!-- Placeholder shape -->
        <rect x="252" y="340" width="37" height="28" />
      </svg>
```

```
    <a class="navbar-brand ms-2" href="#">MyCompany</a>
    <button
      class="navbar-toggler"
      type="button"
      data-bs-toggle="collapse"
      data-bs-target="#navbarSupportedContent"
      aria-controls="navbarSupportedContent"
      aria-expanded="false"
      aria-label="Toggle navigation"
    >
      <span class="navbar-toggler-icon"></span>
    </button>

    <!-- Example Nav Links -->
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link" href="#home">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#about">About</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#contact">Contact</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
<!-- END NAVBAR -->

<!-- MAIN CONTENT CONTAINER -->
<div class="container mt-4">
  <!-- Greeting / Introduction -->
  <h1>Hello, World!</h1>
  <p>This is a combined page demonstrating a contact manager using JSON data.</p>

  <hr class="text-light">

  <!-- SECTION: Contacts Table -->
  <h2 class="mb-4">Contacts</h2>
  <p><i>Select a contact below to edit their details.</i></p>

  <!-- Scrollable table container to show the contact list -->
  <div class="table-responsive" style="max-height: 300px; overflow-y: auto;">
    <table id="contactsTable" class="table table-sm table-bordered table-hover contacts-table">
      <thead class="table-light">
        <tr>
          <th scope="col">First Name</th>
          <th scope="col">Last Name</th>
          <th scope="col">E-mail</th>
          <th scope="col">Phone 1</th>
          <th scope="col">Phone 2</th>
          <th scope="col">Phone 3</th>
        </tr>
      </thead>
      <tbody id="contactsTableBody">
        <!-- Rows will be injected by JavaScript -->
      </tbody>
    </table>
```

```
  </div>
  <!-- END Contacts Table -->

  <!-- SECTION: Contact Detail Form (shown after selecting a contact) -->
  <div id="selectedContact" class="mt-4"></div>
</div>
<!-- END MAIN CONTENT CONTAINER -->

<!--
  BOOTSTRAP 5.3 JS BUNDLE
  Needed for any interactive Bootstrap components (dropdowns, modals, etc.)
-->
<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"
></script>

<!--
  CONTACT MANAGER SCRIPT
  This is where we define our JSON data and the functions to handle it.
  In a real application, you might load JSON from:
    - a REST API using fetch()
    - a local JSON file
    - a separate function call, etc.
-->
<script>
  /*
   * 1) EXAMPLE CONTACTS DATA (JSON format).
   *    - This array is stored in a variable directly in the code.
   *    - In real usage, you might fetch this data from a server or local file.
   */
  var contactsData = [
    {
      "First Name": "Alice",
      "Middle Name": "Jane",
      "Last Name": "Smith",
      "E-mail 1 - Value": "alice.smith@example.com",
      "Phone 1 - Value": "(123) 555-0101",
      "Phone 2 - Value": "",
      "Phone 3 - Value": "",
      "Address 1 - Formatted": "123 Main St, Anytown USA"
    },
    {
      "First Name": "Bob",
      "Middle Name": "",
      "Last Name": "Johnson",
      "E-mail 1 - Value": "bob.johnson@example.org",
      "Phone 1 - Value": "(212) 555-1234",
      "Phone 2 - Value": "(212) 555-5678",
      "Phone 3 - Value": "",
      "Address 1 - Formatted": "456 Oak Avenue, Big City"
    },
    {
      "First Name": "Carol",
      "Middle Name": "Ann",
      "Last Name": "Garcia",
      "E-mail 1 - Value": "carol.garcia@example.net",
      "Phone 1 - Value": "(310) 555-9988",
      "Phone 2 - Value": "",
      "Phone 3 - Value": "",
      "Address 1 - Formatted": "789 Pine Lane, Smallville"
```

```
  },
  {
    "First Name": "David",
    "Middle Name": "",
    "Last Name": "Lee",
    "E-mail 1 - Value": "dlee@example.com",
    "Phone 1 - Value": "(415) 555-2020",
    "Phone 2 - Value": "(415) 555-3030",
    "Phone 3 - Value": "",
    "Address 1 - Formatted": "101 Market St, TechTown"
  },
  {
    "First Name": "Eve",
    "Middle Name": "Marie",
    "Last Name": "Brown",
    "E-mail 1 - Value": "eve.brown@example.co",
    "Phone 1 - Value": "(646) 555-9191",
    "Phone 2 - Value": "",
    "Phone 3 - Value": "",
    "Address 1 - Formatted": "202 Elm Dr, Uptown"
  }
];

/*
 * 2) When the page loads (window.onload), we want to:
 *    - Render the table by injecting rows for each contact in contactsData.
 */
window.onload = function() {
  renderTable(contactsData);
};

/*
 * 3) Renders the table rows with selected columns from our JSON array.
 *    - We only show first name, last name, email, phone1, phone2, phone3
 */
function renderTable(contacts) {
  // Get the table body element
  var tableBody = document.getElementById("contactsTableBody");
  tableBody.innerHTML = ""; // Clear any existing content

  // Loop over each contact
  for (var i = 0; i < contacts.length; i++) {
    var contact = contacts[i];

    // Create a new table row
    var tr = document.createElement("tr");

    /*
     * We access the contact's fields by the keys:
     *   "First Name"
     *   "Last Name"
     *   "E-mail 1 - Value"
     *   "Phone 1 - Value"
     *   "Phone 2 - Value"
     *   "Phone 3 - Value"
     *
     * If any field is missing, use an empty string as fallback.
     */
    var firstName = contact["First Name"] || "";
    var lastName  = contact["Last Name"]  || "";
```

```
      var email     = contact["E-mail 1 - Value"] || "";
      var phone1    = contact["Phone 1 - Value"]   || "";
      var phone2    = contact["Phone 2 - Value"]   || "";
      var phone3    = contact["Phone 3 - Value"]   || "";

      // Create table cells and append them
      addCell(tr, firstName);
      addCell(tr, lastName);
      addCell(tr, email);
      addCell(tr, phone1);
      addCell(tr, phone2);
      addCell(tr, phone3);

      // When a row is clicked, we show that contact's detail form
      tr.onclick = (function(index) {
        return function() {
          showContactForm(contacts[index]);
        };
      })(i);

      // Finally, append the row to the table body
      tableBody.appendChild(tr);
    }
  }

  /*
   * 4) Helper function to create and append a table cell (<td>) with text
   */
  function addCell(tr, text) {
    var td = document.createElement("td");
    td.textContent = text;
    tr.appendChild(td);
  }

  /*
   * 5) Displays the selected contact details in a form below the table.
   *    - For simplicity, we're only showing a few fields.
   *    - This could be extended to show all fields.
   */
  function showContactForm(contact) {
    var container = document.getElementById("selectedContact");
    // Clear anything that was previously displayed
    container.innerHTML = "";

    // Create a simple form element
    var form = document.createElement("form");
    // Make the form's background light, text dark, add some padding
    form.className = "bg-light text-dark p-3 rounded";

    // Append various fields. For demonstration, we show:
    //   First Name, Middle Name, Last Name, E-mail, Phone 1, etc.
    form.appendChild(createFormGroup("First Name",  contact["First Name"]));
    form.appendChild(createFormGroup("Middle Name", contact["Middle Name"]));
    form.appendChild(createFormGroup("Last Name",   contact["Last Name"]));
    form.appendChild(createFormGroup("E-mail",      contact["E-mail 1 - Value"]));
    form.appendChild(createFormGroup("Phone 1",     contact["Phone 1 - Value"]));
    form.appendChild(createFormGroup("Phone 2",     contact["Phone 2 - Value"]));
    form.appendChild(createFormGroup("Phone 3",     contact["Phone 3 - Value"]));
    form.appendChild(createFormGroup("Address",     contact["Address 1 - Formatted"]));
```

```
      // Append the form to the page
      container.appendChild(form);
    }

    /*
     * 6) Utility to create a Bootstrap-style "form group":
     *    - A label
     *    - A text input
     */
    function createFormGroup(labelText, value) {
      var div = document.createElement("div");
      div.className = "mb-3";

      var label = document.createElement("label");
      label.className = "form-label";
      label.textContent = labelText; // e.g., "First Name"

      var input = document.createElement("input");
      input.type = "text";
      input.className = "form-control";
      input.value = value || ""; // default to empty string if no value

      // Put label and input together
      div.appendChild(label);
      div.appendChild(input);

      return div;
    }

    /*
     * 7) ALTERNATE DATA LOADING (Educational placeholder):
     *    - Instead of using a local variable, you could fetch JSON from a server:
     *
     *  fetch('https://api.example.com/contacts')
     *    .then(response => response.json())
     *    .then(data => {
     *      contactsData = data;
     *      renderTable(contactsData);
     *    });
     *
     *  - Or read a local file with an <input type="file"> in the browser,
     *    or from a Node.js script on the server side, etc.
     */
  </script>
</body>
</html>
```