How AJAX Works?

1. A browser makes an **asynchronous request** to the server.
2. The server processes the request and sends a **response** (usually JSON or XML).
3. JavaScript processes the response and updates the webpage **without refreshing**.

Lab Manual: Creating a User Management Portal

Objective:

The objective of this lab is to build a simple User Management Portal. This portal will allow users to:

- View a list of users.
- Add new users.
- Edit existing users.
- Delete users.

Tools Required:

- Text Editor (e.g., VS Code, Sublime Text)
- Web Browser (e.g., Chrome, Firefox)

Step-by-Step Guide:

Step 1: Set up the HTML Structure

Start by creating the basic HTML structure. This will include the `DOCTYPE`, `html`, `head`, and `body` elements.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Users List - User Management Portal</title>
  <style>
    /* Styles go here */
  </style>
```

```
</head>
<body>
  <!-- Content goes here -->
</body>
</html>
```

Step 2: Add Basic Styling

Inside the `<style>` tag, include styles for resetting defaults and creating a clean user interface.

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Roboto', sans-serif;
  background-color: #f0f4f8;
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  color: #333;
}
```

Step 3: Add Form and Table Elements

Inside the `<body>`, create the form and the users table. The form allows adding new users, and the table will display the user list.

```
<div class="container">
  <h1>Users List</h1>

  <div class="form-container">
    <input type="text" id="user-name" placeholder="Name" required />
    <input type="email" id="user-email" placeholder="Email" required
/>
```

```
    <input type="text" id="user-phone" placeholder="Phone" required
/>
    <input type="url" id="user-website" placeholder="Website"
required />
    <button onclick="addOrUpdateUser()">Add/Update User</button>
  </div>

  <table id="users-list" class="users-list">
    <thead>
      <tr>
        <th>Name</th>
        <th>Email</th>
        <th>Phone</th>
        <th>Website</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody></tbody>
  </table>
</div>
```

Step 4: Create JavaScript Functions

Function to Fetch Users

This function fetches users from an API (`jsonplaceholder.typicode.com`) and displays them.

```
async function fetchUsers() {
  const apiUrl = 'https://jsonplaceholder.typicode.com/users';
  const response = await fetch(apiUrl);

  if (response.ok) {
    const users = await response.json();
    displayUsers(users);
  } else {
    console.error('Error fetching users');
    document.getElementById('users-list').innerHTML = 'Failed to
load users.';
  }
}
```

Function to Display Users

This function renders the fetched users in the table.

```
function displayUsers(users) {
  const usersList =
document.getElementById('users-list').getElementsByTagName('tbody')[
0];
  usersList.innerHTML = '';

  users.forEach(user => {
    const row = document.createElement('tr');
    row.setAttribute('data-id', user.id);
    row.innerHTML = `
      <td>${user.name}</td>
      <td>${user.email}</td>
      <td>${user.phone}</td>
      <td><a href="http://${user.website}"
target="_blank">${user.website}</a></td>
      <td class="action-btns">
        <button class="edit-btn" onclick="editUser(${user.id},
'${user.name}', '${user.email}', '${user.phone}',
'${user.website}')">Edit</button>
        <button class="delete-btn"
onclick="deleteUser(${user.id})">Delete</button>
      </td>
    `;
    usersList.appendChild(row);
  });
}
```

Function to Add or Update User

This function adds a new user or updates an existing user.

```
function addOrUpdateUser() {
  const name = document.getElementById('user-name').value;
  const email = document.getElementById('user-email').value;
  const phone = document.getElementById('user-phone').value;
  const website = document.getElementById('user-website').value;
```

```
  if (!name || !email || !phone || !website) {
    alert('Please fill in all fields');
    return;
  }

  if (editingUserId) {
    const usersList =
document.getElementById('users-list').getElementsByTagName('tbody')[
0];
    const rows = usersList.getElementsByTagName('tr');
    for (const row of rows) {
      const userId = row.getAttribute('data-id');
      if (userId == editingUserId) {
        row.cells[0].textContent = name;
        row.cells[1].textContent = email;
        row.cells[2].textContent = phone;
        row.cells[3].innerHTML = `<a href="http://${website}"
target="_blank">${website}</a>`;
        break;
      }
    }
    editingUserId = null;
  } else {
    const newUser = {
      id: Date.now(),
      name: name,
      email: email,
      phone: phone,
      website: website
    };

    const usersList =
document.getElementById('users-list').getElementsByTagName('tbody')[
0];
    const row = document.createElement('tr');
    row.setAttribute('data-id', newUser.id);
    row.innerHTML = `
      <td>${newUser.name}</td>
      <td>${newUser.email}</td>
      <td>${newUser.phone}</td>
```

```
      <td><a href="http://${newUser.website}"
target="_blank">${newUser.website}</a></td>
      <td class="action-btns">
        <button class="edit-btn" onclick="editUser(${newUser.id},
'${newUser.name}', '${newUser.email}', '${newUser.phone}',
'${newUser.website}')">Edit</button>
        <button class="delete-btn"
onclick="deleteUser(${newUser.id})">Delete</button>
      </td>
    `;
    usersList.appendChild(row);
  }

  document.getElementById('user-name').value = '';
  document.getElementById('user-email').value = '';
  document.getElementById('user-phone').value = '';
  document.getElementById('user-website').value = '';
}
```

Function to Edit User

This function pre-fills the form with a user's details for editing.

```
function editUser(id, name, email, phone, website) {
  document.getElementById('user-name').value = name;
  document.getElementById('user-email').value = email;
  document.getElementById('user-phone').value = phone;
  document.getElementById('user-website').value = website;

  editingUserId = id;
}
```

Function to Delete User

This function removes a user from the list.

```
function deleteUser(id) {
  const usersList =
document.getElementById('users-list').getElementsByTagName('tbody')[
0];
```

```
  const rows = usersList.getElementsByTagName('tr');
  for (const row of rows) {
    const userId = row.getAttribute('data-id');
    if (userId == id) {
      usersList.removeChild(row);
      break;
    }
  }
}
```

Step 5: Test the Application

- Load the HTML file in a browser.
- The page will fetch a list of users from the JSONPlaceholder API and display them.
- You can add, edit, or delete users using the form and buttons.

Conclusion:

You have now created a User Management Portal with features to view, add, update, and delete users. This application interacts with a mock API (JSONPlaceholder) for the users' data and is a good starting point for building more complex user management systems.