Task 8: Advanced Form Handling with AJAX

Objective:

Implement AJAX to submit forms without reloading the page, handle server responses, and perform real-time form validation. This task aims to enhance your skills in asynchronous form handling and real-time validation using JavaScript.

Pre-requisites:

- Basic understanding of HTML, CSS, and JavaScript
- Familiarity with a code editor like Visual Studio Code
- Basic knowledge of AJAX and asynchronous JavaScript

Concepts Covered:

- AJAX Form Submission
- Server Response Handling
- Real-time Form Validation

Concepts:

1. **AJAX Form Submission:**

   Implement AJAX to submit forms without reloading the page. Use fetch or XMLHttpRequest to send form data to a server (simulation).

```html
<form id="contactForm">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" placeholder="Enter your
name" required>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" placeholder="Enter your
email" required>

    <button type="submit">Submit</button>
</form>
<div id="feedback"></div>
```

```javascript
document.getElementById('contactForm').addEventListener('submit', function(event)
{
    event.preventDefault();

    const formData = new FormData(this);
    fetch('https://jsonplaceholder.typicode.com/posts', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {
```

```
        document.getElementById('feedback').innerText = 'Form submitted
successfully!';
        document.getElementById('feedback').style.color = 'green';
    })
    .catch(error => {
        document.getElementById('feedback').innerText = 'Error submitting form.';
        document.getElementById('feedback').style.color = 'red';
    });
});
```

2. **Server Response Handling:**

   Simulate server responses and handle them in JavaScript. Display success or error messages based on the response.

```
fetch('https://jsonplaceholder.typicode.com/posts', {
    method: 'POST',
    body: formData
})
.then(response => response.json())
.then(data => {
    document.getElementById('feedback').innerText = 'Form submitted
successfully!';
    document.getElementById('feedback').style.color = 'green';
})
.catch(error => {
    document.getElementById('feedback').innerText = 'Error submitting form.';
    document.getElementById('feedback').style.color = 'red';
});
```

3. **Real-time Form Validation:**

   Validate user input in real-time as the user types. Provide immediate feedback on input validation.

```
<form id="contactForm">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" placeholder="Enter your
name" required>
    <span id="nameFeedback"></span>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" placeholder="Enter your
email" required>
    <span id="emailFeedback"></span>

    <button type="submit">Submit</button>
</form>
<div id="feedback"></div>
```

```
document.getElementById('name').addEventListener('input', function() {
    const name = this.value;
    if (name.length < 3) {
        document.getElementById('nameFeedback').innerText = 'Name must be at
least 3 characters long.';
        document.getElementById('nameFeedback').style.color = 'red';
    } else {
        document.getElementById('nameFeedback').innerText = 'Name looks good!';
        document.getElementById('nameFeedback').style.color = 'green';
    }
});

document.getElementById('email').addEventListener('input', function() {
    const email = this.value;
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!emailRegex.test(email)) {
        document.getElementById('emailFeedback').innerText = 'Invalid
email address.';
        document.getElementById('emailFeedback').style.color = 'red';
    } else {
        document.getElementById('emailFeedback').innerText = 'Email looks good!';
        document.getElementById('emailFeedback').style.color = 'green';
    }
});
```

Setup:

1. **Install Visual Studio Code (VS Code):**

   Download and install VS Code from Visual Studio Code.

2. **Web Browsers:**

   Use Google Chrome or Mozilla Firefox for viewing your webpage and utilizing their developer tools for debugging.

Tasks:

1. **AJAX Form Submission (10 minutes):**
   - Implement AJAX to submit forms without reloading the page.
   - Use fetch or XMLHttpRequest to send form data to a server (simulation).
   - Example:

```html
<form id="contactForm">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" placeholder="Enter your
name" required>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" placeholder="Enter your
email" required>

    <button type="submit">Submit</button>
</form>
<div id="feedback"></div>
```

```javascript
document.getElementById('contactForm').addEventListener('submit',
function(event) {
    event.preventDefault();

    const formData = new FormData(this);
    fetch('https://jsonplaceholder.typicode.com/posts', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById('feedback').innerText = 'Form submitted
successfully!';
        document.getElementById('feedback').style.color = 'green';
    })
    .catch(error => {
        document.getElementById('feedback').innerText = 'Error
submitting form.';
        document.getElementById('feedback').style.color = 'red';
    });
});
```

2. **Server Response Handling (10 minutes):**
    - Simulate server responses and handle them in JavaScript.
    - Display success or error messages based on the response.
    - Example:

```javascript
fetch('https://jsonplaceholder.typicode.com/posts', {
    method: 'POST',
    body: formData
})
.then(response => response.json())
.then(data => {
    document.getElementById('feedback').innerText = 'Form submitted
successfully!';
```

```
    document.getElementById('feedback').style.color = 'green';
})
.catch(error => {
    document.getElementById('feedback').innerText = 'Error submitting form.';
    document.getElementById('feedback').style.color = 'red';
});
```

3. **Real-time Form Validation (10 minutes):**
   - Validate user input in real-time as the user types.
   - Provide immediate feedback on input validation.
   - Example:

```
<form id="contactForm">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" placeholder="Enter your
name" required>
    <span id="nameFeedback"></span>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" placeholder="Enter your
email" required>
    <span id="emailFeedback"></span>

    <button type="submit">Submit</button>
</form>
<div id="feedback"></div>
```

```
document.getElementById('name').addEventListener('input', function() {
    const name = this.value;
    if (name.length < 3) {
        document.getElementById('nameFeedback').innerText = 'Name must be at
least 3 characters long.';
        document.getElementById('nameFeedback').style.color = 'red';
    } else {
        document.getElementById('nameFeedback').innerText = 'Name
looks good!';
        document.getElementById('nameFeedback').style.color = 'green';
    }
});

document.getElementById('email').addEventListener('input', function() {
    const email = this.value;
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!emailRegex.test(email)) {
        document.getElementById('emailFeedback').innerText = 'Invalid
email address.';
        document.getElementById('emailFeedback').style.color = 'red';
    } else {
```

```
        document.getElementById('emailFeedback').innerText = 'Email
looks good!';
        document.getElementById('emailFeedback').style.color = 'green';
    }
});
```

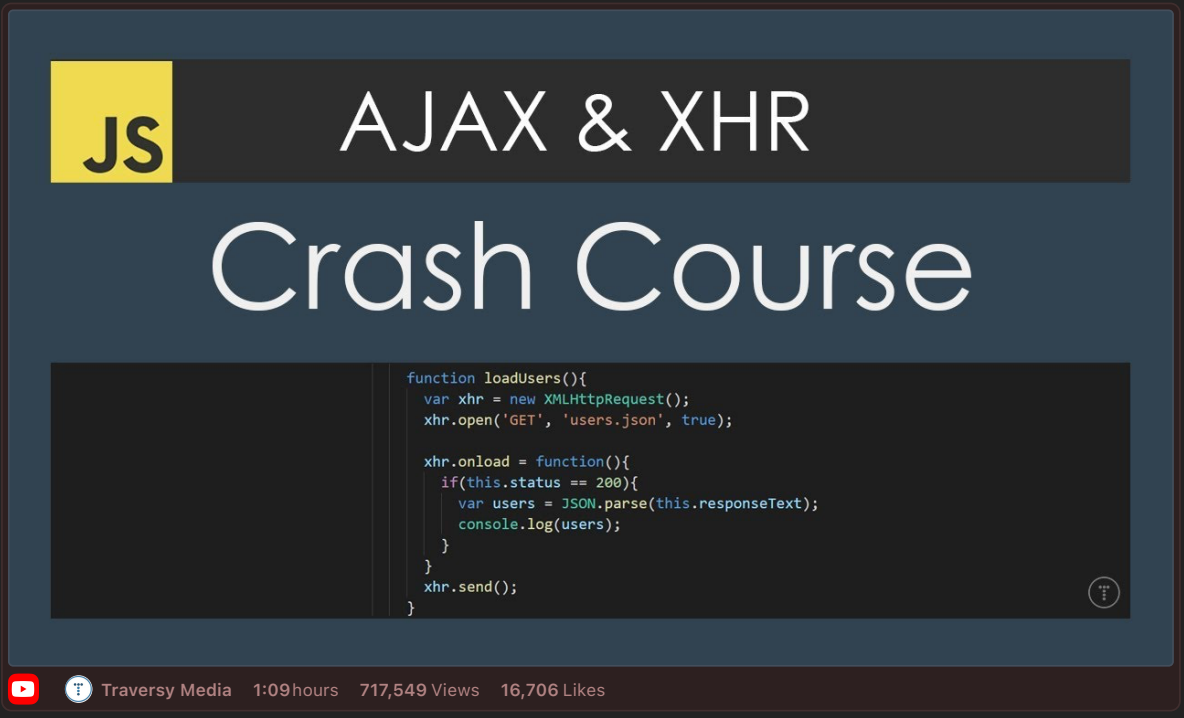
Instructions:

1. Write the required code in `index.html` and `script.js`.
2. Open the `index.html` file in your web browser to ensure the code displays correctly.
3. Use the browser's developer tools to debug and inspect the elements.

Resources:

- **Fetch API - Web APIs | MDN**
  The Fetch API provides an interface for fetching resources (including across the network). I...
  https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
- **FormData - Web APIs | MDN**
  The FormData interface provides a way to construct a set of key/value pairs representing f...
  https://developer.mozilla.org/en-US/docs/Web/API/FormData
- **Client-side form validation - Learn web development | MDN**
  Client-side form validation sometimes requires JavaScript if you want to customize styling...
  https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation
- **Documentation for Visual Studio Code**
  Find out how to set-up and get the most from Visual Studio Code. Optimized for building and d...
  https://code.visualstudio.com/docs

Videos:

- 

GitHub Instructions:

1. **Open in Visual Studio Code:**

   After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, VSCode will open the repository directly. If prompted, select "Open" or "Allow" to open the repository in VSCode.

2. **Open the Terminal in VSCode:**

   In VSCode, open a terminal by selecting Terminal > New Terminal from the top menu.

3. **Complete the Task:**

   In VSCode, write your solution in the `index.html` and `script.js` files.

4. **Run and Test Your Code:**

   Open your `index.html` file in a web browser to ensure it works correctly. Use the following command:

   ```
   open index.html
   ```

5. **Commit Your Changes:**

   In the VSCode terminal, add your changes to git:

   ```
   git add index.html script.js
   ```

   Commit your changes with a meaningful message:

```
git commit -m "Completed task 19"
```

6. **Push Your Changes to Your Repository:**
   Push your changes to your forked repository:

```
git push origin main
```

7. **Create a Pull Request:**
   Go to your repository on GitHub.
   Click on the "Pull Requests" tab.
   Click the "New Pull Request" button.
   Ensure the base repository is the original template repository and the base branch is `main`.
   Ensure the head repository is your forked repository and the compare branch is `main`.
   Click "Create Pull Request".
   Add a title and description for your pull request and submit it.

Summary of Commands:

```
# Open in Visual Studio Code

# Open terminal in VSCode

# Complete the task by editing index.html and script.js

# Navigate to the directory containing index.html
cd path/to/your/index.html

# Run your code
open index.html

# Add, commit, and push your changes
git add index.html script.js
git commit -m "Completed task 8"
git push origin main

# Create a pull request on GitHub
```