

Task 7: Implementing User Authentication

Objective:

Create login and registration forms, simulate authentication logic using JavaScript, and implement form validation and error handling. This task aims to enhance your skills in user authentication and form handling.

Pre-requisites:

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

Concepts Covered:

- Creating Login and Registration Forms
- Simulating Authentication Logic
- Implementing Form Validation and Error Handling

Concepts:

1. Creating Login and Registration Forms:

Dynamically generate forms for user login and registration. Include fields for username, password (and email, name for registration).

```
<form id="loginForm">
  <h2>Login</h2>
  <label for="loginUsername">Username:</label>
  <input type="text" id="loginUsername" name="username" placeholder="Enter your
username" required>

  <label for="loginPassword">Password:</label>
  <input type="password" id="loginPassword" name="password" placeholder="Enter
your password" required>

  <button type="submit">Login</button>
</form>

<form id="registerForm">
  <h2>Register</h2>
  <label for="registerName">Name:</label>
  <input type="text" id="registerName" name="name" placeholder="Enter your
name" required>

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

  <label for="registerUsername">Username:</label>
  <input type="text" id="registerUsername" name="username" placeholder="Enter
your username" required>
```

```

        <label for="registerPassword">Password:</label>
        <input type="password" id="registerPassword" name="password"
placeholder="Enter your password" required>

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

```

2. Simulating Authentication Logic:

Write JavaScript functions to simulate user authentication. Store and retrieve user data from browser local storage (for simulation purposes).

```

function registerUser(name, email, username, password) {
    const users = JSON.parse(localStorage.getItem('users')) || [];
    users.push({ name, email, username, password });
    localStorage.setItem('users', JSON.stringify(users));
}

function authenticateUser(username, password) {
    const users = JSON.parse(localStorage.getItem('users')) || [];
    return users.find(user => user.username === username && user.password
=== password);
}

```

3. Implementing Form Validation and Error Handling:

Validate user input for login and registration forms. Display error messages for invalid inputs or authentication failures.

```

document.getElementById('loginForm').addEventListener('submit', function(event) {
    event.preventDefault();

    const username = document.getElementById('loginUsername').value;
    const password = document.getElementById('loginPassword').value;

    if (authenticateUser(username, password)) {
        document.getElementById('feedback').innerHTML = 'Login successful!';
        document.getElementById('feedback').style.color = 'green';
    } else {
        document.getElementById('feedback').innerHTML = 'Invalid username or
password.';
        document.getElementById('feedback').style.color = 'red';
    }
});

document.getElementById('registerForm').addEventListener('submit',
function(event) {
    event.preventDefault();

```

```

const name = document.getElementById('registerName').value;
const email = document.getElementById('registerEmail').value;
const username = document.getElementById('registerUsername').value;
const password = document.getElementById('registerPassword').value;

if (name && email && username && password) {
    registerUser(name, email, username, password);
    document.getElementById('feedback').innerHTML = 'Registration
successful!';
    document.getElementById('feedback').style.color = 'green';
} else {
    document.getElementById('feedback').innerHTML = 'Please fill in
all fields.';
    document.getElementById('feedback').style.color = 'red';
}
});

```

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. Create Login and Registration Forms (10 minutes):

- Dynamically generate forms for user login and registration.
- Include fields for username, password (and email, name for registration).
- Example:

```

<form id="loginForm">
    <h2>Login</h2>
    <label for="loginUsername">Username:</label>
    <input type="text" id="loginUsername" name="username" placeholder="Enter
your username" required>

    <label for="loginPassword">Password:</label>
    <input type="password" id="loginPassword" name="password"
placeholder="Enter your password" required>

    <button type="submit">Login</button>
</form>

<form id="registerForm">
    <h2>Register</h2>

```

```

    <label for="registerName">Name:</label>
    <input type="text" id="registerName" name="name" placeholder="Enter your
name" required>

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

    <label for="registerUsername">Username:</label>
    <input type="text" id="registerUsername" name="username"
placeholder="Enter your username" required>

    <label for="registerPassword">Password:</label>
    <input type="password" id="registerPassword" name="password"
placeholder="Enter your password" required>

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

```

2. Simulate Authentication Logic (10 minutes):

- Write JavaScript functions to simulate user authentication.
- Store and retrieve user data from browser local storage (for simulation purposes).
- Example:

```

function registerUser(name, email, username, password) {
    const users = JSON.parse(localStorage.getItem('users')) || [];
    users.push({ name, email, username, password });
    localStorage.setItem('users', JSON.stringify(users));
}

function authenticateUser(username, password) {
    const users = JSON.parse(localStorage.getItem('users')) || [];
    return users.find(user => user.username === username && user.password
=== password);
}

```

3. Implement Form Validation and Error Handling (10 minutes):

- Validate user input for login and registration forms.
- Display error messages for invalid inputs or authentication failures.
- Example:

```

document.getElementById('loginForm').addEventListener('submit',
function(event) {
    event.preventDefault();

    const username = document.getElementById('loginUsername').value;
    const password = document.getElementById('loginPassword').value;

    if (authenticateUser(username, password)) {
        document.getElementById('feedback').innerHTML = 'Login successful!';
        document.getElementById('feedback').style.color = 'green';
    } else {
        document.getElementById('feedback').innerHTML = 'Invalid username or
password.';
        document.getElementById('feedback').style.color = 'red';
    }
});

document.getElementById('registerForm').addEventListener('submit',
function(event) {
    event.preventDefault();

    const name = document.getElementById('registerName').value;
    const email = document.getElementById('registerEmail').value;
    const username = document.getElementById('registerUsername').value;
    const password = document.getElementById('registerPassword').value;





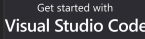

    if (name && email && username && password) {
        registerUser(name, email, username, password);
        document.getElementById('feedback').innerHTML = 'Registration
successful!';
        document.getElementById('feedback').style.color = 'green';
    } else {
        document.getElementById('feedback').innerHTML = 'Please fill in
all fields.';
        document.getElementById('feedback').style.color = 'red';
    }
});

```

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:

- 
Window: localStorage property - Web APIs | MDN
 The localStorage read-only property of the window interface allows you to access a Storag...
 <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>
- 
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:

- 

Dev Drawer 40:56min 34,797 Views 481 Likes

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 17"
```

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 6"
```

```
git push origin main
```

```
# Create a pull request on GitHub
```