Forms and User Input

Web Development Essentials - Session 12

Session Overview

- **Learning Goals for Today**:
  - Learn how to handle forms with JavaScript.
  - Understand how to validate user input.
  - Dynamically display content based on user interactions.

Introduction to Forms in HTML

What is a Form?

- A form is an HTML element that allows users to submit data (e.g., text inputs, radio buttons, checkboxes).

Form Elements:

- **Text inputs**: `<input type="text">`
- **Radio buttons**: `<input type="radio">`
- **Checkboxes**: `<input type="checkbox">`
- **Buttons**: `<button>Submit</button>`
- **Select dropdown**: `<select>`

```
<form>
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
  <button type="submit">Submit</button>
</form>
```

Handling Forms with JavaScript

Why Handle Forms with JavaScript?

- JavaScript allows you to validate form data, handle form submissions without reloading the page, and update the UI dynamically based on user input.

How to Access Form Values:

- Use `document.getElementById()` or `document.querySelector()` to access form fields.

```
let nameInput = document.getElementById("name");
console.log(nameInput.value); // Retrieves the value entered in the input field
```

Preventing Form Default Behavior

Form Default Behavior:

- When a form is submitted, the default behavior is to reload the page.

Preventing Default Form Submission:

- Use `event.preventDefault()` to stop the page from reloading.

```
let form = document.querySelector("form");
form.addEventListener("submit", function(event) {
  event.preventDefault(); // Prevents page from refreshing
  console.log("Form submitted!");
});
```

Validating User Input

What is Validation?

- Validation ensures that the data entered by the user meets certain criteria before being submitted or processed.

Types of Validation:

1. **Client-side Validation** (using JavaScript): Ensures validation happens before data is sent to the server.
2. **Server-side Validation**: Validation occurs on the server after the form is submitted.

```
let nameInput = document.getElementById("name");
if (nameInput.value === "") {
  alert("Name field cannot be empty.");
}
```

Common Form Validations

Required Fields: Ensure the user fills out necessary fields.

Email Format: Ensure the email follows a valid format (e.g., `name@example.com`).

Password Strength: Check if the password meets security criteria (e.g., length, special characters).

```
if (nameInput.value.trim() === "") {
  alert("Please enter your name.");
}

let emailInput = document.getElementById("email");
let emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z]+\.[a-z]{2,}$/;

if (!emailPattern.test(emailInput.value)) {
  alert("Please enter a valid email address.");
}
```

Handling Form Submission with Validation

Putting It All Together:

- Use JavaScript to validate inputs and prevent form submission if validation fails.

```
let form = document.querySelector("form");

form.addEventListener("submit", function(event) {
  event.preventDefault();

  let nameInput = document.getElementById("name");
  let emailInput = document.getElementById("email");

  if (nameInput.value.trim() === "") {
    alert("Name is required.");
    return;
  }

  if (!emailPattern.test(emailInput.value)) {
    alert("Please enter a valid email.");
    return;
  }

  alert("Form submitted successfully!");
});
```

Dynamically Displaying Content Based on User Input

Why Use Dynamic Content?

- Dynamically updating the content of a webpage based on user input enhances interactivity and improves the user experience.

Example: Displaying a Welcome Message:

```
let form = document.querySelector("form");

form.addEventListener("submit", function(event) {
  event.preventDefault();

  let nameInput = document.getElementById("name");
  let message = document.getElementById("message");

  message.textContent = "Welcome, " + nameInput.value + "!";
});
```

Real-time Validation with Event Listeners

Real-time Validation:

- You can validate inputs as the user types by adding event listeners to form fields (e.g., `input`, `change` events).

```
let emailInput = document.getElementById("email");

emailInput.addEventListener("input", function() {
  let emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z]+\.[a-z]{2,}$/;

  if (!emailPattern.test(emailInput.value)) {
    emailInput.style.borderColor = "red";
  } else {
    emailInput.style.borderColor = "green";
  }
});
```

Hands-On Activity

Goal: Create a form that validates user input and displays a success message when the form is submitted correctly.

Instructions:

1. Create a form with fields for name and email.
2. Validate the fields to ensure they are filled out correctly.
3. Display a success message without reloading the page.

```html
<form id="userForm">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name"><br>

  <label for="email">Email:</label>
  <input type="email" id="email" name="email"><br>

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

<p id="message"></p>
```

```html
<script>
  let form = document.getElementById("userForm");

  form.addEventListener("submit", function(event) {
    event.preventDefault();

    let nameInput = document.getElementById("name");
    let emailInput = document.getElementById("email");
    let message = document.getElementById("message");

    if (nameInput.value.trim() === "" || emailInput.value.trim() === "") {
      message.textContent = "Please fill out all fields.";
      return;
    }

    message.textContent = "Thank you, " + nameInput.value + "!";
  });
</script>
```

Common Mistakes in Form Handling

Forgetting `event.preventDefault()`: This will cause the form to reload the page, losing any dynamic behavior.

Not Handling Edge Cases: Ensure all possible user inputs are considered, including empty fields, incorrect formats, and malicious inputs (e.g., XSS attacks).

Summary

What We Learned Today:

- How to handle forms with JavaScript.
- Validating user input to ensure correct data is submitted.
- Dynamically updating the page based on user interactions.

Questions?

Q&A Session

- Any questions before we wrap up?

Thank You & See You in the Next Class!