JavaScript

Module 4 - Functions and Event Handling

Chapter 9 - Form Validations

Aitrich

Form Validation ??

- Form validation is the process of ensuring that the data entered into a web form meets specific requirements and criteria before it's submitted to the server. It's a crucial aspect of web development.

There are two main types of form validation:

- **Client-side validation**
- **Server-side validation**

Form Validation

Client-side validation:

This happens directly in the user's browser, using JavaScript, before any data is sent to the server. It provides immediate feedback and reduces server load, but it can be bypassed by users who disable JavaScript.

Examples: Checking email format, ensuring minimum password length, requiring mandatory fields.

Form Validation

Server-side validation:

This takes place on the server after receiving data from the user's browser. It offers better security and can handle complex checks, but it takes longer to provide feedback and increases server load.

Examples: Checking database uniqueness for username, verifying age against legal requirements, performing complex calculations on input data.

Form Validation - Benefits

- **Faster feedback:** Users receive immediate error messages for invalid input, improving the user experience.
- **Reduced server load:** Only valid data is sent to the server, optimizing performance.
- **Basic checks:** Suitable for simple validations like required fields, format, and range.
- **Security guarantee:** Ensures data integrity even if client-side validation is bypassed.
- **Complex checks:** Can handle advanced validations like database lookups, business logic, and data consistency.
- **Reliable data:** Guarantees consistency of data stored in databases or used for other purposes.

Form Validation - Attributes

- Validation attributes are special properties that can be added directly to HTML form elements to define basic validation rules without any JavaScript code.

- They provide a built-in mechanism within HTML5 to enforce data integrity and guide users towards correct input.

Form Validation - Attributes

Common validation attributes:

- **required:** Makes a field mandatory to fill in before submission.
- **minlength and maxlength:** Sets minimum and maximum lengths for text fields.
- **pattern:** Specifies a regular expression pattern for the input to match.
- **type:** Some input types have built-in validation, such as email, number, tel, etc.
- **min and max:** Specifies minimum and maximum values for numerical fields.
- **Read-only:** Making an input field read-only to prevent user modification.
- **Disabled:** Disabling an input field to prevent user interaction.

Form Validation - Attributes

```html
<!-- Required Attribute -->
<label for="username">Username:</label>
<input type="text" id="username" name="username" required>

<!-- Maxlength and Minlength Attributes -->
<label for="password">Password:</label>
<input type="password" id="password" name="password" maxlength="20" minlength="8">

<!-- Pattern Attribute -->
<label for="email">Email:</label>
<input type="email" id="email" name="email" pattern="[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}">

<!-- Type Attribute for Numeric Input -->
<label for="quantity">Quantity:</label>
<input type="number" id="quantity" name="quantity" min="1" max="100">

<!-- Type Attribute for Date Input -->
<label for="birthdate">Birthdate:</label>
<input type="date" id="birthdate" name="birthdate">

<!-- Pattern Attribute for Custom Validation -->
<label for="customInput">Custom Input (e.g., AAA-999):</label>
<input type="text" id="customInput" name="customInput" pattern="[A-Z]{3}-[0-9]{3}">

<!-- Readonly Attribute -->
<label for="readOnlyInput">Read-Only Input:</label>
<input type="text" id="readOnlyInput" name="readOnlyInput" value="Initial Value" readonly>

<!-- Disabled Attribute -->
<label for="disabledInput">Disabled Input:</label>
<input type="text" id="disabledInput" name="disabledInput" value="Cannot be edited" disabled>
```

Regular Expressions for Advanced Validation

Regular expressions are a powerful tool for defining intricate patterns of text to match, search, and manipulate strings.

Examples For : Email Validation

```
const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

// Example usage:
const email = "user@example.com";
if (emailRegex.test(email)) {
    console.log("Valid email address");
} else {
    console.log("Invalid email address");
}
```

Regular Expressions for Advanced Validation

Phone Number Validation:

```
// Example: Validates US phone numbers with optional country code
const phoneRegex = /^(\+\d{1,2}\s?)?(\(\d{3}\)|\d{3})([-.\s]?)\d{3}([-.\s]?)\d{4}$/;

// Example usage:
const phoneNumber = "+1 (123) 456-7890";
if (phoneRegex.test(phoneNumber)) {
    console.log("Valid phone number");
} else {
    console.log("Invalid phone number");
}
```

Regular Expressions for Advanced Validation

Password Complexity Rules:

```
const passwordRegex = /^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/;

// Example usage:
const password = "Passw0rd!";
if (passwordRegex.test(password)) {
    console.log("Valid password");
} else {
    console.log("Invalid password");
}
```

Thank You