# **Name: Sharvayu Zade**

# **PRN: 23070521135**

# **Sec:B(B2)**

# 

# **Practical 08**

# **Aim: Form Handling & Validation: Access and validate HTML form fields. Implement events such as onsubmit, onblur, onchange. Build a gym admission form with live validation of inputs like name, email, and age.**

# 

# 

# **Introduction:**

# Form handling and validation are essential in web development to ensure users submit correct and complete data. JavaScript provides events and validation techniques to check form input before sending it to the server.

# **JavaScript Form Properties**

A <form> element in HTML can be accessed in JavaScript as:

let form = document.forms["myForm"];

or

let form = document.getElementById("formId");

Sample of form validation:

// General Structure

function validateForm() {

let field = document.forms["myForm"]["username"].value;

if (field == "") {

alert("Field must be filled out");

return false; // prevents form submission

}

}

<form name="myForm" onsubmit="return validateForm()">

<input type="text" name="username">

<input type="submit" value="Submit">

</form>

## **1. Basic Form Properties**

| **Property** | **Description** | **Example** |
| --- | --- | --- |
| **action** | The URL where the form data will be sent. | form.action = "submit.php"; |
| **method** | The HTTP method used (GET or POST). | form.method = "post"; |
| **target** | Where to display the response (\_self, \_blank, \_parent, \_top). | form.target = "\_blank"; |
| **name** | The name of the form. | form.name |
| **id** | The ID of the form (via HTML attribute). | form.id |
| **length** | Number of form elements. | form.length |

## **2. Accessing Form Elements**

| **Property** | **Description** | **Example** |
| --- | --- | --- |
| **elements** | Collection of all form controls inside the form. | form.elements[0] OR form.elements["username"] |
| **value** | The value entered in a form control. | form.elements["username"].value |
| **type** | Type of the input element (text, password, radio, checkbox, submit, etc.). | form.elements["email"].type |
| **checked** | Boolean – whether a checkbox or radio is selected. | form.elements["gender"].checked |
| **disabled** | Boolean – whether the input is disabled. | form.elements["age"].disabled = true; |
| **readOnly** | Boolean – makes a field read-only. | form.elements["username"].readOnly = true; |
| **required** | Boolean – field must be filled out. | form.elements["email"].required = true; |

## 

## 

## **3. Form Control Specific Properties**

### **Text Fields (<input type="text">, <textarea>)**

* .value → The text entered.
* .maxLength → Maximum number of characters.
* .placeholder → Placeholder text.
* .selectionStart, .selectionEnd → Position of cursor/selection.

### **Password Fields (<input type="password">)**

* .value → Entered password.
* .maxLength → Maximum length allowed.

### **Checkboxes & Radio Buttons**

* .checked → Whether selected or not.
* .value → The value assigned to that option.

### **Select/Dropdown (<select>)**

* .options → All <option> elements.
* .selectedIndex → Index of selected option.
* .value → Value of selected option.
* .length → Number of options.

### **File Input (<input type="file">)**

* .files → List of uploaded files.
* .value → Path of the chosen file (may be restricted for security).

## 

## **4. Form Validation Properties (HTML5)**

| **Property** | **Description** | **Example** |
| --- | --- | --- |
| **validity** | Returns a ValidityState object with validation results (like valid, typeMismatch, etc.). | input.validity.valid |
| **checkValidity()** | Returns true if form is valid, otherwise false. | form.checkValidity() |
| **reportValidity()** | Shows validation messages to user. | form.reportValidity() |
| **willValidate** | Returns true if element is subject to validation. | input.willValidate |
| **setCustomValidity(message)** | Sets a custom error message. | input.setCustomValidity("Invalid input"); |

## **5. Form Collection Properties**

The document.forms object stores all forms on a page:

document.forms.length // Number of forms

document.forms[0] // First form

document.forms["loginForm"] // Form by name

# **JavaScript Form Events**

## **Form-Level Events**

| **Event** | **Description** |
| --- | --- |
| **onsubmit** | Fires when the form is submitted. |
| **onreset** | Fires when the form is reset. |
| **onforminput** | Fires when an element inside the form receives input. |
| **onformchange** | Fires when an element inside the form changes. |

## **Input & Field Events**

| **Event** | **Description** |
| --- | --- |
| **oninput** | Fires every time the value of an <input>, <textarea>, or <select> element changes while typing. |
| **onchange** | Fires when the element’s value changes and loses focus (for <input>, <textarea>, <select>). |
| **onfocus** | Fires when an input field gains focus. |
| **onblur** | Fires when an input field loses focus. |
| **onfocusin** | Similar to onfocus, but bubbles up (can be captured at form level). |
| **onfocusout** | Similar to onblur, but bubbles up. |

## **Keyboard Events in Forms**

| **Event** | **Description** |
| --- | --- |
| **onkeydown** | Fires when a key is pressed down. |
| **onkeypress** (deprecated) | Fires when a key is pressed and produces a character. |
| **onkeyup** | Fires when a key is released. |

## **Mouse Events on Form Elements**

| **Event** | **Description** |
| --- | --- |
| **onclick** | Fires when a user clicks on an input/button. |
| **ondblclick** | Fires on double click. |
| **onmousedown** | Fires when mouse button is pressed. |
| **onmouseup** | Fires when mouse button is released. |
| **onmousemove** | Fires when mouse moves over element. |
| **onmouseover** | Fires when mouse enters element. |
| **onmouseout** | Fires when mouse leaves element. |

## **Clipboard Events in Input Fields**

| **Event** | **Description** |
| --- | --- |
| **oncopy** | Fires when the user copies content. |
| **oncut** | Fires when the user cuts content. |
| **onpaste** | Fires when the user pastes content. |

## **Drag & Drop Events**

| **Event** | **Description** |
| --- | --- |
| **ondrag** | Fires when an element is being dragged. |
| **ondragstart** | Fires when dragging starts. |
| **ondragend** | Fires when dragging stops. |
| **ondragenter** | Fires when a dragged element enters a valid drop target. |
| **ondragover** | Fires when an element is being dragged over a valid drop target. |
| **ondragleave** | Fires when dragged element leaves a valid drop target. |
| **ondrop** | Fires when the dragged element is dropped. |

## **Other Useful Events**

| **Event** | **Description** |
| --- | --- |
| **oninvalid** | Fires when an element (like <input required>) fails validation. |
| **onselect** | Fires when text inside an <input> or <textarea> is selected. |
| **onwheel** | Fires when mouse wheel is used over an element. |

# **Form Validation Techniques**

## **1. Required Field Validation**

Ensures that a field is not left empty.

**HTML5 way:** <input type="text" name="username" required>

**JavaScript way:** if (username.trim() === "") {

alert("Username is required");

return false;

}

## **2. Data Type Validation**

Makes sure the entered value is of the correct type.

Example: checking if age is a number.

if (isNaN(age)) {

alert("Age must be a number");

}

## **3. Length Validation**

Checks minimum and maximum length of input (like password or name).

if (password.length < 6) {

alert("Password must be at least 6 characters");

}

## **4. Range Validation**

Ensures numeric values fall within a valid range.

Example: age should be between 18 and 60.

if (age < 18 || age > 60) {

alert("Age must be between 18 and 60");

}

## **5. Pattern / Regular Expression Validation**

Matches input against a pattern (regex).

Example: Email validation.

let pattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;

if (!email.match(pattern)) {

alert("Invalid email format");

}

## **6. Compare Fields Validation**

Ensures two fields match (like password & confirm password).

if (password !== confirmPassword) {

alert("Passwords do not match");

}

## **7. Numeric Validation**

Ensures only numbers are entered (e.g., phone number).

HTML5 way:  
  
 <input type="number" name="age">

JS way:  
  
 if (!/^[0-9]+$/.test(phone)) {

alert("Phone must contain only digits");

}

## **8. Email Validation**

Ensures valid email format.

let pattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;

if (!email.match(pattern)) {

alert("Invalid email address");

}

## **9. Password Strength Validation**

Check for uppercase, lowercase, number, special character, and length.

let pattern = /^(?=.\*[A-Z])(?=.\*[a-z])(?=.\*[0-9])(?=.\*[@#$%^&+=]).{8,}$/;

if (!password.match(pattern)) {

alert("Weak password. Use uppercase, lowercase, number & special char.");

}

## **10. Checkbox / Radio Validation**

Ensures the user selects an option.

if (!document.querySelector('input[name="gender"]:checked')) {

alert("Please select your gender");

}

## **11. Dropdown Validation**

Ensures a valid option is selected (not the default).

if (document.getElementById("city").value === "") {

alert("Please select a city");

}

## **12. Date Validation**

Ensures valid date format and logic (like DOB not in the future).

let dob = new Date(document.getElementById("dob").value);

let today = new Date();

if (dob > today) {

alert("Date of Birth cannot be in the future");

}

## **13. File Upload Validation**

Ensures file type, size, and extension are valid.

let file = document.getElementById("resume").files[0];

if (file.size > 2 \* 1024 \* 1024) { // 2MB

alert("File too large");

}

if (!file.name.endsWith(".pdf")) {

alert("Only PDF allowed");

}

## **14. Custom Validation using oninvalid**

HTML5 has built-in validation error messages. You can override them.

<input type="email" required oninvalid="this.setCustomValidity('Enter a valid email!')">

## 

## **15. Real-time (Live) Validation**

Use oninput or onkeyup to check values as the user types.

document.getElementById("username").oninput = function() {

if (this.value.length < 3) {

document.getElementById("error").innerText = "Must be at least 3 characters";

} else {

document.getElementById("error").innerText = "";

}

}

