Los formularios HTML son elementos fundamentales en la creación de páginas web interactivas. Permiten a los usuarios introducir datos, realizar selecciones y enviar información al servidor para su procesamiento. Sin embargo, los formularios básicos HTML pueden resultar limitados en cuanto a la funcionalidad y la experiencia del usuario. Es aquí donde JavaScript entra en juego, proporcionando herramientas para mejorar la interacción con los formularios y añadir características avanzadas.

Algunos usos de los formularios son:

- *Registro de usuarios*
- *Contacto*
- *Búsqueda*
- *Compras online*
- *Comentarios*

La estructura básica de un formulario es la siguiente:

```
<form name="registro">
        <label for="nombre">Nombre:</label>
        <input type="text" id="nombre" name="nombre">

        <label for="correo">Correo electrónico:</label>
        <input type="email" id="correo" name="correo">

        <button type="submit">Enviar</button>
</form>
```

Algunos elementos de los formularios son:

- **Campos de entrada** (`<input>`): Permiten a los usuarios introducir datos de texto, números, fechas, contraseñas, etc.
- **Casillas de verificación** (`<input type="checkbox">`): Permiten a los usuarios seleccionar múltiples opciones.
- **Botones de opción** (`<input type="radio">`): Permiten a los usuarios seleccionar una única opción de un grupo.
- **Menús desplegables** (`<select>`): Permiten a los usuarios seleccionar una opción de una lista.
- **Botones** (`<button>`): Inician una acción cuando se hace clic, como enviar el formulario o restablecer los campos.

- **Etiquetas** (`<label>`): Proporcionan texto descriptivo para los campos de entrada.

Algunos atributos de los formularios son:

- **id**: Un identificador único para el elemento.
- **name**: El nombre del elemento, utilizado para enviar los datos del formulario al servidor.
- **type**: El tipo de elemento, como text, email, checkbox, radio, etc.
- **value**: El valor inicial del elemento.
- **required**: Indica si el campo es obligatorio.
- **placeholder**: Texto de marcador que se muestra dentro del campo cuando está vacío.

Interacción con los formularios usando JavaScript

JavaScript permite interactuar con los formularios HTML de forma dinámica y agregar funcionalidades que no son posibles con HTML puro.

Acceso a los elementos del formulario

Para interactuar con los elementos de un formulario en JavaScript, primero hay que acceder a ellos. Esto se puede hacer utilizando diferentes métodos, como:

- `document.getElementById(id)` : Obtiene un elemento por su ID.
- `document.querySelector(selector)` : Obtiene un elemento que coincida con un selector CSS.
- `document.forms[nombre]` : Obtiene un formulario por su nombre.
- `form.elements[nombre]` : Obtiene un elemento dentro de un formulario por su nombre.

Ejemplo de acceso a elementos de un formulario:

```
<script>
  const nombreInput = document.getElementById('nombre');
  const emailInput = document.querySelector('input[name="correo"]');
  const formulario = document.forms['registro'];
  const botonEnviar = formulario.elements['enviar'];
</script>
```

Una vez que tenemos acceso a un elemento del formulario, podemos obtener su valor actual o modificarlo. Para obtener el valor, se utiliza la propiedad `value`. Para modificarlo, se asigna un nuevo valor a la propiedad `value`.

Gestión de eventos en formularios

Los eventos son acciones que ocurren en un elemento o en la página web. En el contexto de los formularios, algunos eventos comunes son:

- **submit**: Se dispara cuando se envía el formulario.
- **change**: Se dispara cuando se cambia el valor de un campo.
- **focus**: Se dispara cuando un campo recibe el foco.
- **blur**: Se dispara cuando un campo pierde el foco.

Para gestionar eventos en formularios, se utilizan los manejadores de eventos. Estos manejadores son funciones que se ejecutan cuando se produce el evento correspondiente. Se pueden asociar a los elementos del formulario utilizando diferentes métodos, como:

- **on**: agrega un manejador DOM.
- **addEventListener**: Agrega un manejador de eventos a un elemento.
- **removeEventListener**: Elimina un manejador de eventos de un elemento.

```
// 1. Evento 'change': Se dispara cuando cambia el valor de un campo
// En este caso, comprobamos el valor escrito en el campo 'nombre'
const nombreInput = document.getElementById('nombre');
nombreInput.addEventListener('change', function() {
    const nombre = nombreInput.value;
    console.log('El nombre ha cambiado a:', nombre);
});

// 2. Evento 'focus' y 'blur': Enfoque y desenfoque de un campo
// Podemos mostrar mensajes cuando el campo recibe o pierde el foco
nombreInput.addEventListener('focus', function() {
    console.log('El campo de nombre ha recibido el foco');
});

nombreInput.addEventListener('blur', function() {
    console.log('El campo de nombre ha perdido el foco');
});

// 3. Evento 'submit': Envío del formulario
// Se usa generalmente para validar y enviar datos
const formulario = document.getElementById('miFormulario');
formulario.addEventListener('submit', function(event) {
    event.preventDefault(); // Evita el envío del formulario por defecto
    // Validar los datos del formulario aquí
    if (nombreInput.value === '') {
        console.log('El campo de nombre está vacío');
        return; // Detener el proceso si el campo está vacío
    }
```

```
    // Si los datos son válidos, se podría enviar el formulario aquí
    console.log('Formulario enviado con éxito');
});
```

Submit en un formulario

```
// Manejador de evento submit para validación previa
formulario.addEventListener('submit', function(event) {
    event.preventDefault(); // Evita el envío por defecto
    // Validación de los datos
    if (nombreInput.value === '') {
        console.log('Por favor, complete el campo de nombre');
    } else {
        console.log('Datos del formulario válidos, listos para enviar');
    }
});
```

Validación de Formularios

La validación de formularios permite verificar que los datos ingresados por el usuario son correctos y completos. En JavaScript, podemos realizar validaciones básicas con métodos para cadenas y números, y validaciones más complejas usando **expresiones regulares (regex)**.

Las expresiones regulares son patrones para buscar y validar texto. Algunas aplicaciones incluyen:

- **Validación de datos**: Para formatos específicos como correos electrónicos y números de teléfono.
- **Búsqueda y extracción**: Encontrar coincidencias en texto.
- **Procesamiento de texto**: Transformar y manipular cadenas.

Ejemplo de validación de correo electrónico con regex:

```
// Función para validar un correo electrónico usando una expresión regular
function validarEmail(email) {
    const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/; // Patrón de regex básico para
correos
    return regex.test(email); // Devuelve true si el email es válido
}

// Evento para validar el correo al perder el foco en el campo de email
const emailInput = document.getElementById('email');
```

```
emailInput.addEventListener('blur', function() {
    const email = emailInput.value;
    if (!validarEmail(email)) {
        alert('El correo electrónico no es válido.');
    }
});
```

Ejemplo completo de validación de formulario

Este ejemplo valida:

- **Campo Nombre**: No vacío, al menos 5 caracteres.
- **Campo Email**: No vacío, con formato válido de email.
- Muestra errores si hay algún problema y solo envía el formulario cuando todos los datos son válidos.

```
// Validación del campo Nombre y Email
function validarFormulario() {
    let errores = false;

    // Validación del nombre
    const nombreInput = document.getElementById('nombre');
    const nombreError = document.getElementById('nombreError');
    if (nombreInput.value === '' || nombreInput.value.length < 5) {
        nombreError.style.display = 'block';
        nombreError.textContent = 'El nombre debe tener al menos 5
caracteres';
        errores = true;
    } else {
        nombreError.style.display = 'none';
    }

    // Validación del correo electrónico
    const emailInput = document.getElementById('email');
    const emailError = document.getElementById('emailError');
    if (emailInput.value === '' || !validarEmail(emailInput.value)) {
        emailError.style.display = 'block';
        emailError.textContent = 'Ingrese un correo electrónico válido';
        errores = true;
    } else {
        emailError.style.display = 'none';
    }

    // Si no hay errores, mostrar un mensaje de éxito
    if (!errores) {
```

```
        alert('Los datos se han enviado con éxito.');
    }
    return !errores; // Devuelve false si hay errores para evitar el envío
}

// Manejador del evento 'submit' del formulario
let formulario = document.getElementById('miFormulario');
formulario.addEventListener('submit', function(event) {
    event.preventDefault(); // Evita el envío del formulario si hay errores
    validarFormulario();
    //formulario.submit();
});
```

Detalles Importantes

1. **Divs de error**: Añadir debajo de cada campo un div oculto para mostrar mensajes de error.
2. **Validación en submit**: Realizar todas las validaciones antes de enviar.
3. **Manejo de Errores**: Mostrar mensajes de error específicos para cada campo inválido.

Controles de formularios

JavaScript permite manipular fácilmente los elementos de un formulario, como cuadros de texto, radio buttons, checkboxes, y selects. A continuación, se presentan ejemplos para cada tipo de control.

1. Cuadro de Texto y Textarea

Podemos obtener y establecer el valor de los cuadros de texto (`<input type="text">`) y áreas de texto (`<textarea>`) usando la propiedad `value` .

```
// Obtener valores de un input y un textarea
const valorText = document.getElementById("texto").value;
const valorTextArea = document.getElementById("parrafo").value;
```

Además, es posible definir campos como ocultos (`type="hidden"`) o de solo lectura (`readonly`).

2. Radio Button

Los radio buttons permiten seleccionar solo una opción de un grupo. Para verificar cuál de ellos está seleccionado, usamos la propiedad `checked` .

```
// Obtener el estado de selección de cada radio button
let elementos = document.getElementsByName("estadoCivil");
Array.from(elementos).forEach(ec => console.log(`${ec.value} es ${ec.checked ?
'seleccionado' : 'no seleccionado'}`));
```

3. Checkbox

Los checkboxes son similares a los radio buttons, pero permiten selecciones múltiples. Podemos comprobar si cada checkbox está seleccionado también con la propiedad `checked`.

```
// Verificar la selección de checkboxes
function verCondicionesPrivacidad() {
    const condiciones = document.getElementById("condiciones").checked ? " se
ha aceptado" : " no se ha aceptado";

    const privacidad = document.getElementById("privacidad").checked ? " se ha
aceptado" : " no se ha aceptado";

    console.log(`Condiciones: ${condiciones}, Privacidad: ${privacidad}`);
}
```

4. Select (Lista Desplegable)

Las listas desplegables (`<select>`) contienen una colección de opciones (`<option>`), y su valor puede obtenerse mediante la propiedad `value`.

```
// Obtener el valor seleccionado de un select
let lista = document.getElementById("lista");
console.log("Valor seleccionado:", lista.value);

lista.addEventListener("change", () => console.log("Nuevo valor:",
lista.value));
```

Para acceder a la posición o modificar la lista:

- `options` es un array con todas las opciones.
- `selectedIndex` indica el índice de la opción seleccionada.

```
// Acceder a una opción específica del select
let indiceSeleccionado = lista.selectedIndex;
let opcionSeleccionada = lista.options[indiceSeleccionado];
```

```
console.log("Texto seleccionado:", opcionSeleccionada.text);
console.log("Valor seleccionado:", opcionSeleccionada.value);
```

5. Añadir y Eliminar Elementos en un Select

Es posible agregar y eliminar opciones de un `<select>` de forma dinámica usando `add` y `remove`.

```
// Añadir una opción a la lista
function anyadirElementoLista() {
    let texto = prompt("Dime Texto");
    let valor = prompt("Dime Valor");
    let mioption = document.createElement("option");
    mioption.value = valor;
    mioption.text = texto;
    lista.appendChild(mioption);
}

// Eliminar una opción de la lista
function eliminarElementoLista(indice) {
    if (lista.options.length > 0) {
        lista.remove(indice);
    }
}
```

Ejemplo Completo: Gestión de Controles de Formulario

```
// Ejemplo de manipulación de controles en formulario
document.getElementById("submitBtn").addEventListener("click", function(event)
{
    event.preventDefault();

    // Cuadro de texto y textarea
    let nombre = document.getElementById("nombre").value;
    let mensaje = document.getElementById("mensaje").value;
    console.log("Nombre:", nombre, "Mensaje:", mensaje);

    // Radio Button
    let estadoCivilSeleccionado =
Array.from(document.getElementsByName("estadoCivil"))
        .find(rb => rb.checked);
    console.log("Estado Civil Seleccionado:", estadoCivilSeleccionado ?
estadoCivilSeleccionado.value : "Ninguno");
```

```
    // Checkbox
    let aceptaCondiciones = document.getElementById("condiciones").checked;
    console.log("Acepta Condiciones:", aceptaCondiciones);

    // Select
    let lista = document.getElementById("lista");
    let valorSeleccionado = lista.value;
    console.log("Valor Seleccionado de Lista:", valorSeleccionado);
});
```