Validación avanzada de formularios

- C.F.G.S. DAW
- 6 horas semanales
- Mes aprox. de impartición: Ene
- iPasen - cjaedia071@g.educaand.es

Carmelo José Jaén Díaz

Índice

OBJETIVO

- Identificar el concepto de BOM con los objetos que lo componen y saber utilizarlo en el desarrollo de aplicaciones web.

GLOSARIO

BOM (Browser Object Model). Objeto en el que se representa el navegador, no solo el documento. Mediante el BOM se puede acceder a datos como el historial de navegación, dimensiones de la ventana, etcétera.

DOM. Plataforma e interfaz de un documento que permite a los scripts y programas acceder y modificar dinámicamente su contenido, estructura y estilo.

Tag. Término en inglés que significa “etiqueta” y hace referencia a una palabra clave que sirve para describir un documento.

INTRODUCCIÓN

En las lecciones anteriores estudiamos cómo hacer una **validación básica de formularios utilizando solo Javascript**, y cómo hacer **una validación básica de formularios utilizando solamente HTML5**.

En esta lección veremos cómo aunar ambas cosas basándonos en la **API de Validación de Restricciones de Javascript**, que consta de un conjunto de **métodos y propiedades** disponibles en las interfaces DOM de elementos de formulario como `button`, `input`, `select`, etc. y que facilita enormemente este trabajo.

INTRODUCCIÓN

Recuerda: No es seguro validar un formulario únicamente con #Javascript o con #HTML. La validación en el lado del cliente sólo garantiza la experiencia de usuario. Hay que validar los datos en el servidor y en la base de datos para asegurarnos que son correctos.

INTRODUCCIÓN

Como hemos comentado anteriormente, el propósito de esta lección es mostrar cómo se validan los formularios en las aplicaciones web actuales. Estos implementan su estructura y restricciones en HTML5 y hacen uso de JavaScript para modificar los mensaje de validación, mejorando así la experiencia de usuario.

Para ejemplificar esto, se parte del formulario implementado en la lección anterior.

FORMULARIO

```
<h1>Formulario</h1>
<form action="procesar.php" method="post" id="miFormulario">
  <fieldset>
    <legend>Datos personales</legend>
    <p>Nombre: <input type="text" name="nombre" id="nombre" maxlength="15"
pattern="[A-Za-z ]{2,15}" title="Introduce entre 2 y 15 letras" required
/></p>
    <p>Edad: <input type="number" name="edad" id="edad" min="18" max="100"
required /></p>
    <p>Teléfono: <input type="tel" name="telefono" id="telefono"
pattern="[0-9]{3}-[0-9]{3}-[0-9]{3}" title="Número de 9 cifras separado por
guiones" required /></p>
  </fieldset>
```

FORMULARIO

```
  <p>
    <input type="submit" value="Enviar" id="enviar" />
    <input type="reset" value="Borrar" id="borrar" />
  </p>
</form>
```

FORMULARIO

En primer lugar, vinculamos el documento HTML con un fichero JS donde se implementará la lógica de validación del formulario mediante:

```
<script src="validacion.js"></script>
```

A continuación, vinculamos un fichero CSS externo e implementamos la clase `.error` que nos permite resaltar los campos del formulario que no cumplen los criterios de validación requeridos.

```
.error {
    border: solid 2px red;
}
```

FORMULARIO

Por último, implementamos un párrafo donde iremos mostrando los mensajes de error que vayan ocurriendo. Se ubica entre el formulario y los botones del mismo.

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

Recuerda que en W3SCHOOLS puedes encontrar ejemplos de formularios y cómo validar los campos (`input`) a través de sus atributos (`max`, `min`, `pattern`, `required`, `type`).

También, puedes encontrar la documentación relacionada con el método `checkValidity()`, como:

- Su mensaje de validación, validationMessage.
- Sus propiedades de validación:
    - `rangeOverflow`, se establece en `true` si el valor del elemento excede el valor del atributo `max`.
    - `toolong`, se establece en `true` si el valor del elemento excede el valor del atributo `maxLength`.

VALIDACIÓN DEL FORMULARIO

validacion.js

Es importante tener en cuenta que hasta que no se haya cargado (creado) la página web, no podremos acceder a los elementos de la misma, para ello podríamos implementar:

- El atributo `defer`, a la hora de vincular el fichero JS.
- Asignar una función al evento `window.onload`.

```
window.onload = iniciar; //Sin paréntesis ya que si no se ejecutaría automaticamente

//Asocia un evento a la pulsación del boton enviar
function iniciar() {
   document.getElementById("enviar").addEventListener('click', validar, false);}
```

VALIDACIÓN DEL FORMULARIO

validar()

La función `validar()` comprueba que todos los campos del formulario son correctos.

```
if (validaNombre() && validaEdad() && validaTelefono() && confirm("Pulsa
aceptar si deseas enviar el formulario")) {
        return true;
}
```

A continuación, se implementan cada una de las funciones incluidas en el `if`.

VALIDACIÓN DEL FORMULARIO

validaNombre()

La función `validaNombre()`:

- Accede al elemento correspondiente al nombre:

```
let elemento = document.getElementById("nombre");
```

- Comprobamos que la validación HTML5 es correcta, si no lo es invocamos la función `error()`:

```
if (!elemento.checkValidity()) //Comprobación de la validación HTML5
    error(elemento); //Análogo a la lección anterior
    return false;
```

- En caso contrario devolvemos `true`:

```
return true;
```

VALIDACIÓN DEL FORMULARIO

validaEdad()

La función `validaEdad()`:

- Implementa la misma lógica que `validaNombre()`:

```
let elemento = document.getElementById("edad");
if (!elemento.checkValidity()){ //Comprobación de la validación HTML5
    error(elemento); //Análogo a la lección anterior
    return false;
}
```

- En caso contrario devolvemos `true`:

```
return true;
}
```

VALIDACIÓN DEL FORMULARIO
validaTelefono()

La función `validaTelefono()`:

- Implementa la misma lógica que `validaNombre()`:

```
let elemento = document.getElementById("telefono");
if (!elemento.checkValidity()){ //Comprobación de la validación HTML5
   error(elemento); //Análogo a la lección anterior
   return false;
}
```

- En caso contrario devolvemos `true`:

```
return true;
}
```

VALIDACIÓN DEL FORMULARIO

validaTodo()

Ya que estas tres funciones implementan la misma lógica, la forma óptima sería integrarlas, evitando así la repetición de código. Por tanto podríamos implementar algo así:

```
function validaTodo(element) {
    let elemento = document.getElementById("element");
    if (!elemento.checkValidity()) {
        error(elemento)
        return false;
    }
        return true;
}
```

Sin embargo, no las integraremos debido a que vamos a seguir implementado funcionalidades para cada una de ellas.

VALIDACIÓN DEL FORMULARIO

validar()

La función `validar()` comprueba que todas las funciones de validación devuelven `true`, es importante el orden de comprobación para que los posibles mensajes de error aparezcan en el orden deseado. Adicionalmente, se implementa un cuadro de diálogo `confirm` para confirmar que el usuario está seguro de enviar el formulario.

```
function validar(e) {
    if (validaNombre() && validaTelefono() && validaFecha() && validaCheck()
    && confirm("Pulsa aceptar si deseas enviar el formulario")) {
        return true;
    }
```

VALIDACIÓN DEL FORMULARIO

validar()

En caso de que alguna función de validación devuelva `false` o bien el usuario pulse *cancelar* en el `confirm`, debemos evitar que se envíe el formulario.

Para ello, incluimos como parámetro (`e`), que hace referencia al objeto `Evento` creado automáticamente cuando se asocia una función a un evento. Y a este objeto le aplicamos el método `preventDefault()`, el cual cancela el evento por defecto asociado a la acción.

En este caso, la acción asociada al botón `submit`, es ir a la página `procesar.php`.

```
else {
    e.preventDefault();
    return false;
}
```

VALIDACIÓN DEL FORMULARIO

error()

Por último, vamos a implementar la función `error()` que modifica el elemento <p> con `id="mensajeError"`, sobreescribiendolo con el mensaje que tiene por defecto la validación `validationMessage`. Este atributo ya tiene predefinidos una serie de mensajes para cada uno de los idiomas del navegador.

Adicionalmente, modificaremos los campos erróneos, para que aparezcan resaltados con un borde rojo mediante la clase CSS definida anteriormente y le aplicaremos el foco para que el usuario no tenga dudas de a qué campo se refiere, mejorando así su experiencia de uso.

```
function error(elemento) {
    document.getElementById("mensajeError").innerHTML =
elemento.validationMessage;
    elemento.className = "error";
    elemento.focus();
}
```

VALIDACIÓN DEL FORMULARIO

borrarError()

Adicionalmente, implementamos la función `borrarError()` cuya finalidad es la de quitar todas las clases CSS `.error` de todos los elementos al enviar el formulario.

```
function borrarError() {
    let formulario = document.forms[0];

   //Recorremos todos los input del formulario
    for (let i = 0; i < formulario.elements.length; i++) {
        formulario.elements[i].className = "";
    }
}
```

Como queremos que se eliminen las clases al enviar el formulario, debemos invocar a dicha función al principio de la función `validar()`, ya que es la función invocada al hacer *click* en el botón del formulario.

VALIDACIÓN DEL FORMULARIO
Visualización del ejemplo

Puedes apreciar el comportamiento de este código en el siguiente link o en la imagen adjunta:

https://codepen.io/Carmelo-Jos-Ja-n-D-az/pen/RNbEjqy

VALIDACIÓN DEL FORMULARIO

Mejora

Como posible mejora, para cada validación realizada mediante `checkValidity()`, se va realizar la comprobación que permita conocer exactamente porqué el formato es erróneo. Mostrando un mensaje personalizado en vez del mensaje por defecto **`validationMessage`**.

Para ello, se emplean las propiedades de `validity`, como:

- `patternMismatch`: comprueba que el valor del elemento coincide con su patrón.
- `valueMissing`: comprueba que un elemento, con un atributo requerido, tiene un valor asignado.
- `rangeOverflow`: comprueba que el elemento no supere su atributo `max`.
- `rangeUnderflow`: comprueba que el elemento no supere su atributo `min`.

VALIDACIÓN DEL FORMULARIO
Mejora *validaNombre()*

En la validación realizada mediante `checkValidity()`, implementamos:

```
//Comprueba que el elemento tiene un valor asignado, el atributo required
if (elemento.validity.valueMissing) {
   //Implementamos una nueva función de error, a la que le pasamos como
   //argumentos el elemento y un mensaje de error
   error2(elemento, "El campo nombre es obligatorio")
}

//Comprueba que el elemento coincide con la RegEx
if (elemento.validity.patternMismatch) {
   error2(elemento, "El nombre debe tener entre 2 y 15 caracteres
   alfabéticos, mayúsculas o minúsculas");
}
```

VALIDACIÓN DEL FORMULARIO
Mejora *validaEdad()*

En la validación realizada mediante `checkValidity()`, implementamos:

```
//Comprueba que el elemento tiene un valor asignado, el atributo required
if (elemento.validity.valueMissing) {
   error2(elemento, "El campo edad es obligatorio")
}

//Comprueba que el elemento cumple el atributo max
if (elemento.validity.rangeOverflow) {
   error2(elemento, "El valor debe ser menor o igual de 100");
}

//Comprueba que el elemento cumple el atributo min
if (elemento.validity.rangeUnderflow) {
   error2(elemento, "El valor debe ser mayor o igual que 18");
}
```

VALIDACIÓN DEL FORMULARIO
Mejora *validaTelefono()*

En la validación realizada mediante `checkValidity()`, implementamos:

```
//Comprueba que el elemento tiene un valor asignado, el atributo required
if (elemento.validity.valueMissing) {
    //Implementamos una nueva función de error, a la que le pasamos como
    //argumentos el elemento y un mensaje de error
    error2(elemento, "El campo teléfono es obligatorio")
}

//Comprueba que el elemento coincide con la RegEx
if (elemento.validity.patternMismatch) {
    error2(elemento, "El teléfono debe tener 9 números, separados por
    guiones");
}
```

VALIDACIÓN DEL FORMULARIO

error2()

La función `error2()` mostrará el mensaje de error personalizado que se le pasa como argumento:

```
function error2(elemento, mensaje) {

    //Modificamos el párrafo encargado de mostrar el mensaje modificado
    document.getElementById("mensajeError").innerHTML = mensaje;

    //Le aplicamos la clase CSS .error
    elemento.className = "error";

    //Ponemos el foco en el elemento erróneo
    elemento.focus();
}
```

VALIDACIÓN DEL FORMULARIO
Visualización del ejemplo

Puedes apreciar el comportamiento de este código en el siguiente link o en la imagen adjunta:

https://codepen.io/Carmelo-Jos-Ja-n-D-az/pen/ByBMYoo

VALIDACIÓN DEL FORMULARIO
Propiedades de *validity*

Recuerda que puedes implementar tantas propiedades de `validity`, como necesites. A continuación, se listan las usadas en el ejemplo:

- `patternMismatch`: comprueba que el valor del elemento coincide con su patrón.
- `valueMissing`: comprueba que un elemento, con un atributo requerido, tiene un valor asignado.
- `rangeOverflow`: comprueba que el elemento no supere su atributo `max`.
- `rangeUnderflow`: comprueba que el elemento no supere su atributo `min`.