UD5
Interacción con el usuario. Formularios.

Formularios

Validación de formularios desde el lado del cliente

Validar formularios es una de las operaciones más comunes que se llevan a cabo en el desarrollo de páginas web. Para su tratamiento, tanto en HTML como en JavaScript han ido apareciendo elementos, etiquetas, etc, que facilitan el tratamiento de los formularios.

¿Qué es un formulario? Es un conjunto de campos en los que el usuario introduce datos que de alguna manera será procesada, bien por nuestro código en el lado cliente (JavaScript) o bien enviados y procesados en el servidor. La función principal de un formulario es la de **recoger información** para ser **procesada** posteriormente.

Estructura básica de un formulario

Una estructura básica puede verse en el siguiente código:

```
<form action="test.php" method="GET">
  <label>Introduce nombre:
    <input type="text" name="nombre"/></label>
  <br>
  <label>Introduce apellido:
    <input type="text" name="cognom"/>
  </label>
  <br>
  <input type="submit" value="enviar" />
</form>
```

Como puedes ver en este ejemplo hemos usado un objeto de tipo **form** el cual representa al formulario, y luego hemos seguido una serie de etiquetas **label** y una serie de campos de entrada de datos **input.**

Curiosidad: Mandando datos al servidor:

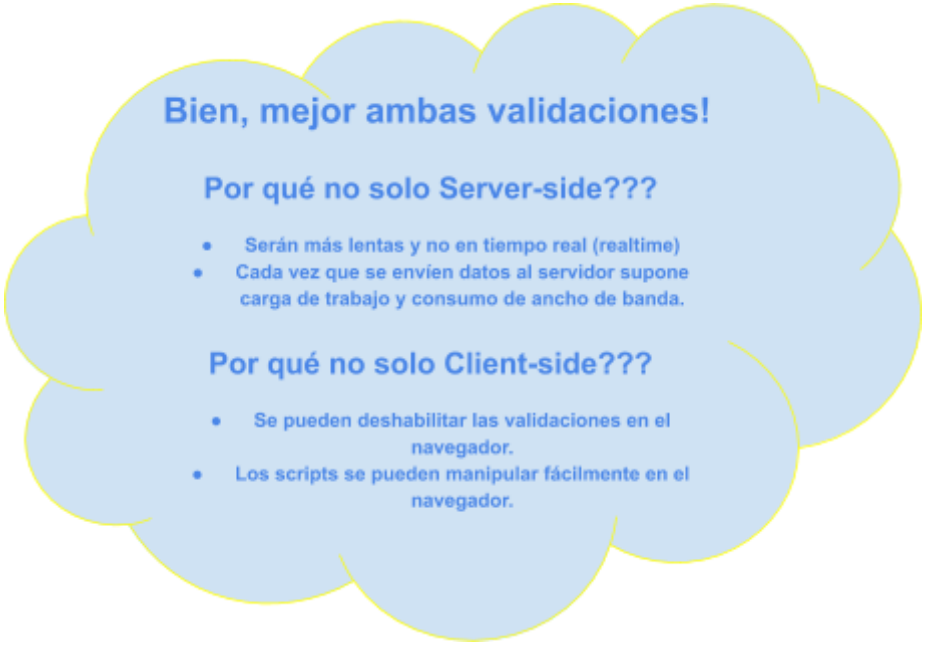
En el ejemplo anterior puedes ver en el elemento **form** los atributos **action="test.php"** y **method="GET"** . El primero indica que al pulsar el botón de envío, se debe pasar esta información en la página test.php , mientras que el segundo indica que el método que utilizaremos para pasar esta información es GET.

Cuando las páginas a las que dirigimos nuestras peticiones no existen, siempre se devuelve una página de error con el código 404. Esto puedes verlo en la herramienta de desarrollador de tu navegador, en la pestaña Network (Red en español).

Validación de formularios, ¿¿ en el cliente y/o en el servidor ??

La validación de datos se debe de hacer en ambos ámbitos. Toda información a validar frente a la base de datos se realizará en el servidor. Toda la validación en cuanto a campos vacíos, formatos, se pueden dejar solo en el lado cliente, y sin que el usuario llegue a clicar sobre "submit" o "enviar" se pueden ir realizando validaciones en tiempo real.

Los objetivos que se buscan son: que el cliente esté bien informado de los campos a introducir y posibles errores, minimizar llamadas al servidor innecesarias, y mantener la seguridad de nuestra sistema y usuarios respecto a la validación de datos.



Principales elementos de un formulario en HTML

Los elementos principales que se utilizan en un formulario son:

- **form:** elemento principal que forma un formulario; todos los datos definidos en elementos que se encuentren entre la etiqueta de apertura y cierre serán enviados al pulsar el botón de envío.
- **input:** este elemento permite que el usuario introduzca datos, y puede representarse de muchas maneras diferentes: *botones, cuadros de texto, selectores de colores, valores numéricos*,etc. Puedes ver todos los tipos de input en el siguiente enlace: https://www.w3schools.com/tags/tag_input.asp
- **textarea:** sirve únicamente para introducir textos. A diferencia de “input" de tipo text , el textarea es multilineal y se utiliza generalmente para la introducción de textos largos.
- **button:** este elemento crea un botón clicable que, por defecto, su comportamiento es enviar el formulario. Puede dar problemas en versiones de navegadores antiguos.
- **label:** para mejorar la estructura semántica del documento es recomendable utilizar este elemento para agregar las etiquetas correspondientes a cada elemento de entrada de datos.
- **datalist (HTML5):** este elemento nos permite añadir listas en nuestro código HTML que pueden ser utilizadas para otros elementos del formulario. Un ejemplo:

```
<html>  <body>
<form action="/action_page.php" method="get">
  <input list="browsers" name="browser">
  <datalist id="browsers">
      <option value="Internet Explorer">
      <option value="Firefox">
      <option value="Chrome">
      <option value="Opera">
```

```
      <option value="Safari">
  </datalist>
  <input type="submit">
</form>
</body>  </html>
```

¿Cómo accedemos al objeto “form” desde javascript?

Pues tendremos diversas formas de hacerlo. Vamos a partir de una línea HTML que defina el formulario:

```
<form action="procesar.php" method="post" id="miFormulario">
```

Y en las siguientes líneas de código podremos ver diversas formas de obtener dicho elemento desde nuestro código JavaScript:

```
<script>
//Si conocemos el id podemos hacerlo de estas 2 formas:
let formulario = document.getElementById("miFormulario");
let formulario2 = document.forms["miFormulario"];
//Si conocemos el número de formulario que es en la página podemos usar:
let formulario3 = document.getElementsByTagName("form")[0];  // 0 indica la posición
del formulario.
let formulario4 = document.forms[0];  // 0 indica la posición del formulario.
</script>
```

¿Podrías averiguar por qué no es muy recomendable utilizar las 2 últimas formas?

Solución: *En ambas líneas de código se selecciona el formulario basándonos en su posición. Imaginate que en la web se añadiera otro form previo a este. Este pasaría a ser el formulario en posición [1] creando dependencias y necesitando modificaciones del código.*

¿Cómo seleccionar elementos del formulario?

Una vez que tengamos el formulario, seguro que tendremos que seleccionar o trabajar con algunos de sus elementos. Para obtener acceso a algunos de sus elementos podemos usar varias estrategias:

1ª forma: **formulario.elements[ ]** ➜ Devuelve un array con todos los inputs del formulario.

2ª forma: **formulario.getElementById("id")** ➜ Devuelve el elemento en concreto con el id seleccionado.

3ª forma: **formulario.getElementsByTagName("tag")** ➜ Devuelve un array con todos los elementos que cumplan el tag (input, select, etc).

4ª forma: **formulario.getElementsByName("nombre")** ➜ Devuelve un array con todos los elementos que tienen el mismo nombre (por ejemplo, “radiobutton”).

5ª forma: **Métodos querySelector y querySelectorAll**

Importante: Es muy importante que aprendas las diversas formas de acceder a elementos del formulario. Además fíjate en lo que te devuelve cada una de los métodos empleados.

Importante: Diferencia entre usar la etiqueta "id" y "name"

- **Id:** El "id" se utiliza normalmente para identificar elementos HTML que vayan a ser utilizados en Javascript/CSS.
- **Name:** El atributo **"name"** es usado principalmente cuando un formulario se envía al servidor y necesitamos adaptar los campos del lado cliente al lado servidor. Diferentes controles responden de forma diferente. Por ejemplo: imagina que tenemos varios radiobuttons con **diferentes "id"** pero **igual "name"**; cuando se envía esa información se enviarán bajo la etiqueta "name" y solo tendrá un valor: el valor del radiobutton que hubieras seleccionado.

A la hora de construir el árbol DOM, es obligatorio que los **"id"** de los elementos sean únicos; no ocurre lo mismo que con la etiqueta "**name**".

Validación clásica de formularios: al ataquerrr!

Tenemos el siguiente formulario básico:

```
<form action="/login.php" id="formulario">
      <p>Usuario: <input type="text" name="usuario" id="usuario"></p>
      <p>Clave: <input type="text" name="clave" id="clave"></p>
      <input type="submit" value="Entrar">
</form>
```

Vamos a validarlo:

```
<script>
       window.onload = iniciar;
       function iniciar(){
           document.getElementById("formulario").addEventListener('submit', validarFormulario);
       }

       function validarNombre(){
           let usuario = document.getElementById('usuario').value;
               if(usuario.length == 0) {
                   alert('No has escrito nada en el usuario');
                   return false;
               }
               else
                   return true;
           }
       function validarClave(){
           let clave = document.getElementById('clave').value;
           if (clave.length < 6) {
               alert('La clave no es válida');
               return false;
           }else
```

```
            return true;
        }


        function validarFormulario( event )
        {
            if (validarNombre()  && validarClave())
                alert("Validación validada correctamente")
            else{
                event.preventDefault();    ///// IMPORTANTISIMO
                alert("Validación NO validada correctamente")
            }
        }
    </script>
```

Fíjate en la importancia que tiene la línea en la que se escribe: **event.preventDefault()**

Lo que hace este método es anular la acción por defecto que lleva una determinada acción/evento. En este caso, con formularios, la acción por defecto viene determinada por el **tag "action" del formulario**. Bien, pues con esta línea la anulamos. Para ver más claro su funcionamiento, haz que no valide el formulario y pulsa el botón submit con esa línea comentada y sin comentar.

Validación de formularios usando etiquetas de HTML5

Como ya habíamos mencionado anteriormente, el elemento principal que va dar forma a un formulario es el objeto de tipo **<form>**. Pero este, además, se nutre de un gran número de elementos construidos especialmente para la creación de formularios. La gran mayoría de estos elementos están bajo la etiqueta **<input>** (representa un campo de datos) y constan de una gran cantidad de tipos y etiquetas que nos van a ayudar a realizar validación desde el código HTML5. Podemos ver sus características en: https://www.w3schools.com/tags/tag_input.asp

Algunas de las principales **propiedades** que podemos añadir al elemento **<input>** y que nos ayudarán a realizar validación en línea son:

- **maxlength, minlength:** Representa el número máximo de caracteres que acepta un campo. (Tiene que ir en conjunto con el campo/propiedad "required").
- **max , min:** Representa el máximo o mínimo valor numérico que acepta un campo.
- **pattern:** Representa una expresión regular que validará o no los valores introducidos en el campo.
- **placeholder:** Representa el texto de ayuda que aparece semi transparente y que desaparece cuando el usuario escribe (hint).
- **required:** Hace que el campo sea obligatorio, y si no se completa, se invalidará el envío del formulario y se resaltan de rojo indicando que es obligatorio.
- **type:** Indica el tipo de campo. Tenemos: *button, checkbox, color, date, datetime-local, email, file, hidden, image, month, number, password, radio, range, reset, search, submit, tel, text, time, url, week*.

  Hay que tener en cuenta que no todos los navegadores soportan todos estos tipos de inputs. Podéis verlos mejor en el tutorial de la W3Schools:

  https://www.w3schools.com/html/html_form_input_types.asp

Attribute	Description
disabled	Specifies that the input element should be disabled
max	Specifies the maximum value of an input element
min	Specifies the minimum value of an input element
pattern	Specifies the value pattern of an input element
required	Specifies that the input field requires an element
type	Specifies the type of an input element

Validación avanzada: Form API

Javascript nos ofrece una API de métodos y propiedades que nos permite detectar si un determinado campo ha sido validado o no. En el caso de que no se haya validado, gracias a esta API podremos realizar una customización del manejo de errores.

Los métodos y propiedades de esta API quedan especificados en la siguiente web:

https://www.w3schools.com/js/js_validation_api.asp

Property	Description
customError	Set to true, if a custom validity message is set.
patternMismatch	Set to true, if an element's value does not match its pattern attribute.
rangeOverflow	Set to true, if an element's value is greater than its max attribute.
rangeUnderflow	Set to true, if an element's value is less than its min attribute.
stepMismatch	Set to true, if an element's value is invalid per its step attribute.
tooLong	Set to true, if an element's value exceeds its maxLength attribute.
typeMismatch	Set to true, if an element's value is invalid per its type attribute.
valueMissing	Set to true, if an element (with a required attribute) has no value.
valid	Set to true, if an element's value is valid.

(falta uno muy común que es el "tooShort" que es lo contrario del "tooLong")

A continuación trataremos de explicar la funcionalidad de algunos de ellos:

Método:

- **checkValidity():** Devuelve true si un determinado campo contiene un valor válido.
- **setCustomValidity()**: Nos permite customizar el mensaje de error si la validación no es positiva.

  !!!!!! Importantísimo !!!! Hay que volver a poner a cadena vacía el setCustomValidity para las próximas validaciones, si no, siempre el checkValidity() será false.

Propiedades:

- **validationMessage:** Nos muestra un mensaje del error producido.

- **customError:** Vale “true” si un mensaje customizado ha sido configurado.
- **patternMismatch:** Vale “true” si el valor de un campo no valida la expresión regular configurada.
- **rangeOverflow:** Vale “true” si el valor de un campo es mayor que el valor máximo configurado (atributo **max**).
- **rangeUnderflow:** Vale “true” si el valor de un campo es menor que el valor mínimo configurado (atributo **min**).
- **tooLong:** Vale “true” si el valor de un campo excede de la longitud marcada en su atributo “**maxLenght**”.
- **typeMismatch:** Vale “true” si el valor de un campo por el tipo de valor introducido es diferente al fijado en su atributo “**type**” .
- **valueMissing:** Vale “true” sin un campo no tiene valor y uno de sus atributos es “**required**”.
- **valid:** Vale “true” si el de valor de una variable es válido.

Para usar estos valores debemos de acceder a ellos a través de la variable **“validity”**.

```
<form action="/login.php" id="formulario">
        <p>Usuario: <input type="text" name="usuario" id="usuario" required
minlength="3"></p>
        <p>Clave: <input type="text" name="clave" id="clave" required minlength="5"></p>
        <input id="enviar" type="submit" value="Entrar">
    </form>

    <script>
        window.onload = iniciar;

        function iniciar() {
            document.getElementById("enviar").addEventListener('click',
validarFormulario);
        }

        function validarNombre() {
            let usuario = document.getElementById('usuario');
            if (!usuario.checkValidity()) {
                if (usuario.validity.valueMissing){
                    usuario.setCustomValidity( "No has escrito nada en el campo usuario");
                }else if (usuario.validity.tooShort)
                {
                        usuario.setCustomValidity("Nombre de usuario demasiado corto");
                }
                    return false;
                } else
                    return true;
        }

        function validarClave() {
            let clave = document.getElementById('clave');
            if (!clave.checkValidity()) {
                if (clave.validity.valueMissing)
                    clave.setCustomValidity('No has escrito nada en el campo clave');
                else if (clave.validity.tooShort)
                    clave.setCustomValidity("'Nombre de usuario demasiado clave'");

                return false;
            } else
                return true;
        }


        function validarFormulario(event) {
            if (validarNombre() && validarClave())
                alert("Validación validada correctamente")
```

```
            else {
                alert("Validación NO validada correctamente")
            }
        }
    </script>
```

Ejercicios

Ejercicio 1: Validación básica/clásica de un formulario mediante código javascript:

Dado el siguiente código html:

```
<!DOCTYPE html>
<html>
<head>
	<title>Formulario</title>
	<script src="validar.js"></script>
	<style>
	.error {
		border: solid 2px #FF0000;
	}
	</style>
</head>

<body>
	<h1>Formulario</h1>
	<form action="procesar.php" method="post" id="miFormulario">
	<table>
		<tr>
			<td>Nombre: </td>
			<td>
			<input type="text" name="nombre" id="nombre" />
			</td>
		</tr>
		<tr>
			<td>Telefono: </td>
			<td>
			<input type="text" name="telefono" id="telefono" />
			</td>
		</tr>
		<tr>
			<td>Fecha de nacimiento: </td>
			<td>
			<input type="text" name="dia" size="2" id="dia" />
			<input type="text" name="mes" size="2" id="mes" />
			<input type="text" name="ano" id="ano" size="4" />
			</td>
		</tr>
		<tr>
			<td>Sexo : </td>
			<td>
			<input type="radio" name="sexo" value="H" checked /> Hombre
			<input type="radio" name="sexo" value="M" /> Mujer
```

```
                </td>
            </tr>
            <tr>
                <td>Mayor de 18:</td>
                <td>
                <input type="checkbox" name="mayor" id="mayor" checked/>
                </td>
            </tr>
            </td>
            </tr>
      </table>
      <p>
            <input type="submit" value="Enviar" id="enviar" />
            <input type="reset" value="Borrar" id="borrar" />
      </p>
      </form>
</body>

</html>
```

Debemos de realizar una validación antes del envío del formulario mediante código javascript que realice las siguientes operaciones:

- Una método **iniciar()** el cuál se llame cuando el DOM se haya formado y que sea la encargada de añadir un evento “onclick” al botón “enviar” del formulario y llame a un método que nos determine la validación global del formulario.
- Un método **validar()** el cuál se encargue de comprobar que todos los campos validan. Si todos los campos están bien, antes de dar el OK, mostrar un mensaje de confirmación para que el usuario decida si enviar o no el formulario por última vez. Si algún campo no valida, debe de eliminar la acción del botón “submit”.
- **Métodos de validaciones de campos:**
    - Comprobar que el campo “nombre” no sea vacío.
    - Comprobar que el campo “teléfono” sea un número (da igual el formato).
    - Comprobar que los campos “día”, ”mes” y “ano” formen una fecha válida.
    - Comprobar que el checkbox de mayor de 18 años esté chequeado o no.

Ejercicio 2: Validación avanzada de un formulario mediante código javascript + etiquetas HTML5:

Dado el siguiente código html:

```
<html>
<head>
	<title>VALIDACIÓN DE FORMULARIOS CON HTML5</title>
</head>
<body>
	<h1>Formulario</h1>
	<form action="procesar.php" method="post" id="miFormulario">
	<table>
		<tr>
			<td>Nombre*: </td>
			<td>
		    <input type="text" name="nombre" id="nombre" required/>
			</td>
		 </tr>
		<tr>
			<td>Edad*: </td>
			<td>
			<input type="number" name="edad" id="edad" />
			</td>
		</tr>

		<tr>
			<td>Telefono*: </td>
			<td>
			<input type="text" name="telefono" id="telefono" />
			</td>
		</tr>
	</table>
	<p>
		<input type="submit" value="Enviar" id="enviar" />
		<input type="reset" value="Borrar" id="borrar" />
	</p>
	<p id="mensajeError"></p>
	</form>
</body>
</html>
```

1. Añade las etiquetas HTML5 necesarias para que el campo de texto **"nombre"** tenga una longitud máxima de 15 caracteres, y sea capaz de validar una expresión regular que acepte solo letras

(mayúsculas y/o minúsculas) y tenga una longitud de 2 a 15 letras. También, si el usuario sitúa el ratón encima del campo por unos segundos debe de aparecer el mensaje: "Introduce entre 2 y 15 letras". Además el campo debe de ser "obligatorio" de cumplimentar.

2. Añade la etiqueta HTML5 necesaria para que el campo numérico **"edad"** tenga un valor numérico mínimo de 18 y máximo de 100. También de ser "obligatorio" de cumplimentar.
3. Añade la etiqueta HTML5 necesaria para que el campo de texto **"teléfono"** sea capaz de validar una expresión regular en la que solo se puedan utilizar números y de una longitud de 9 dígitos. Si el usuario sitúa el ratón por unos segundos sobre el campo teléfono, aparecerá el mensaje "Número de 9 cifras". Además el campo debe de ser "obligatorio" de cumplimentar.
4. Mediante el uso del método `checkValidity()` (mira en los apuntes) y las distintas propiedades **"validity"** debemos de comprobar y capturar todos aquellos errores producidos por las validaciones HTML5 y personalizar los mensajes de error según el error producido. Los mensajes de error se mostrarán al final de la pantalla en la etiqueta con id **"`mensajeError`".**
5. Vamos a añadir 2 nuevos métodos: **mostrarError(elemt , mensajeError) y borrarError(elemt)**. Estos métodos reciben por parámetro un objeto al cual aplicarán o borrarán una clase de CSS que bordea el campo de rojo o lo elimina. La clase a aplicar es:

```
<style>
      .error {
                  border: solid 2px #FF0000;
      }
</style>
```

6. Investiga cómo añadirle al formulario un objeto de tipo "`<input type="range">`" y que tenga como valor inicial 50, el rango sea entre 1 y 100, y el paso (como cambian los valores) sea de 5 en 5.