

1. VALIDACIÓN DE DATOS EN PÁGINAS WEB

1.1. Introducción

Una vez visto, en la unidad formativa anterior, la funcionalidad de JavaScript, pasamos a una de las tareas donde JavaScript es más potente: **la validación de los datos introducidos por el usuario.**

Pensemos en los datos que un usuario introduce mediante un formulario cuando compramos un billete de avión, reservamos o compramos una entrada de teatro, cine..., hacemos la reserva de un hotel, etc, etc, los servidores pueden estar recibiendo, de forma simultánea, cientos de peticiones, por lo que es fundamental que los datos enviados sean correctos y evitar un ida y vuelta de datos inútiles entre cliente y servidor.

Es muy importante destacar que esta validación en el cliente es únicamente para evitar el tráfico de ida y vuelta con datos erróneos, la última validación de los datos enviados debe ser ejecutada en el propio servidor con lenguajes de programación más avanzados como: **PHP, ASP, JSP, PERL.**

La principal utilidad de JavaScript en el manejo de los formularios es la validación de los datos introducidos por los usuarios. Antes de enviar un formulario al servidor, se recomienda validar mediante JavaScript los datos insertados por el usuario. De esta forma, si el usuario ha cometido algún error al rellenar el formulario, se le puede notificar de forma instantánea, sin necesidad de esperar la respuesta del servidor.

Notificar los errores de forma inmediata mediante JavaScript mejora la satisfacción del usuario con la aplicación (lo que técnicamente se conoce como *"mejorar la experiencia de usuario"*) y ayuda a reducir la carga de procesamiento en el servidor.

Normalmente, la validación de un formulario consiste en llamar a una función de validación cuando el usuario pulsa sobre el botón de envío del formulario. En esta función, se comprueban si los valores que ha introducido el usuario cumplen las restricciones impuestas por la aplicación.

Aunque existen tantas posibles comprobaciones como elementos de formulario diferentes, algunas comprobaciones son muy habituales: que se rellene un campo obligatorio, que se seleccione el valor de una lista desplegable, que la

dirección de email indicada sea correcta, que la fecha introducida sea lógica, que se haya introducido un número donde así se requiere, etc.

1.2. Funciones De Validación

Normalmente, la validación de un formulario consiste en llamar a una función de validación cuando el usuario pulsa sobre el botón de envío del formulario. En esta función, se comprueban si los valores que ha introducido el usuario cumplen las restricciones impuestas por la aplicación.

Aunque existen tantas posibles comprobaciones como elementos de formulario diferentes, algunas comprobaciones son muy habituales: que se rellene un campo obligatorio, que se seleccione el valor de una lista desplegable, que la dirección de email indicada sea correcta, que la fecha introducida sea lógica, que se haya introducido un número donde así se requiere, etc.

A continuación se muestra el código JavaScript básico necesario para incorporar la validación a un formulario:

```
<form action="" method="" id="" name="" onsubmit="return validacion()">
```

```
...
```

```
</form>
```

Y el esquema de la función **validacion()** es el siguiente:

```
function validacion() {  
  
    if (condicion que debe cumplir el primer campo del formulario) {  
        // Si no se cumple la condicion...  
        alert('[ERROR] El campo debe tener un valor de...');  
        return false;  
    }  
  
    else if (condicion que debe cumplir el segundo campo del formulario) {  
        // Si no se cumple la condicion...  
        alert('[ERROR] El campo debe tener un valor de...');  
        return false;  
    }  
}
```

```

...
else if (condicion que debe cumplir el último campo del formulario) {
    // Si no se cumple la condicion...
    alert('[ERROR] El campo debe tener un valor de...');
    return false;
}

// Si el script ha llegado a este punto, todas las condiciones
// se han cumplido, por lo que se devuelve el valor true
return true;
}

```

El funcionamiento de esta técnica de validación se basa en el comportamiento del evento **onsubmit** de JavaScript. Al igual que otros eventos como **onclick** y **onkeypress**, el evento '**onsubmit**' varía su comportamiento en función del valor que se devuelve.

Así, si el evento **onsubmit** devuelve el valor **true**, el formulario se envía como lo haría normalmente. Sin embargo, si el evento **onsubmit** devuelve el valor **false**, el formulario no se envía. La clave de esta técnica consiste en comprobar todos y cada uno de los elementos del formulario. En cuando se encuentra un elemento incorrecto, se devuelve el valor **false**. Si no se encuentra ningún error, se devuelve el valor **true**.

Por lo tanto, en primer lugar se define el evento **onsubmit** del formulario como:

```
onsubmit="return validacion()"

```

Como el código JavaScript devuelve el valor resultante de la función **validacion()**, el formulario solamente se enviará al servidor si esa función devuelve **true**. En el caso de que la función **validacion()** devuelva **false**, el formulario permanecerá sin enviarse.

Dentro de la función **validacion()** se comprueban todas las condiciones impuestas por la aplicación. Cuando no se cumple una condición, se devuelve **false** y por tanto el formulario no se envía. Si se llega al final de la

función, todas las condiciones se han cumplido correctamente, por lo que se devuelve **true** y el formulario se envía.

La notificación de los errores cometidos depende del diseño de cada aplicación. En el código del ejemplo anterior simplemente se muestran mensajes mediante la función **alert()** indicando el error producido. Las aplicaciones web mejor diseñadas muestran cada mensaje de error al lado del elemento de formulario correspondiente y también suelen mostrar un mensaje principal indicando que el formulario contiene errores.

Una vez definido el esquema de la función **validacion()**, se debe añadir a esta función el código correspondiente a todas las comprobaciones que se realizan sobre los elementos del formulario. A continuación, se muestran algunas de las validaciones más habituales de los campos de formulario.

1.2.1. Validar un campo de texto obligatorio

Se trata de forzar al usuario a introducir un valor en un cuadro de texto o textarea en los que sea obligatorio. La condición en JavaScript se puede indicar como:

```
valor = document.getElementById("campo").value;  
if( valor == null || valor.length == 0 || /^\\s+$/.test(valor) ) {  
    return false;  
}
```

Para que se dé por completado un campo de texto obligatorio, se comprueba que el valor introducido sea válido, que el número de caracteres introducido sea mayor que cero y que no se hayan introducido sólo espacios en blanco.

La palabra reservada **null** es un valor especial que se utiliza para indicar *"ningún valor"*. Si el valor de una variable es **null**, la variable no contiene ningún valor de tipo objeto, array, numérico, cadena de texto o booleano.

La segunda parte de la condición obliga a que el texto introducido tenga una longitud superior a cero caracteres, esto es, que no sea un texto vacío.

Por último, la tercera parte de la condición (**/^\\s+\$/.test(valor)**) obliga a que el valor introducido por el usuario no sólo esté formado por espacios en blanco. Esta comprobación se basa en el uso de **"expresiones regulares"**, un recurso habitual en cualquier lenguaje de programación y de gran complejidad. Por lo

tanto, sólo es necesario copiar literalmente esta condición, poniendo especial cuidado en no modificar ningún carácter de la expresión.

1.2.2. Validar un campo de texto con valores numéricos

Se trata de obligar al usuario a introducir un valor numérico en un cuadro de texto. La condición JavaScript consiste en:

```
valor = document.getElementById("campo").value;  
if( isNaN(valor) ) {  
    return false;  
}
```

Si el contenido de la variable **valor** no es un número válido, no se cumple la condición. La ventaja de utilizar la función interna **isNaN()** es que simplifica las comprobaciones, ya que JavaScript se encarga de tener en cuenta los decimales, signos, etc.

A continuación se muestran algunos resultados de la función **isNaN()**:

```
isNaN(3);      // false  
isNaN("3");    // false  
isNaN(3.3545); // false  
isNaN(32323.345); // false  
isNaN(+23.2);  // false  
isNaN("-23.2"); // false  
isNaN("23a");  // true  
isNaN("23.43.54"); // true
```

1.2.3. Validar que se ha seleccionado una opción de una lista

Se trata de obligar al usuario a seleccionar un elemento de una lista desplegable. El siguiente código JavaScript permite conseguirlo:

```
indice = document.getElementById("opciones").selectedIndex;  
if( indice == null || indice == 0 ) {  
    return false;  
}
```

```
<select id="opciones" name="opciones">  
    <option value="">- Selecciona un valor -</option>  
    <option value="1">Primer valor</option>  
    <option value="2">Segundo valor</option>  
    <option value="3">Tercer valor</option>  
</select>
```

A partir de la propiedad **selectedIndex**, se comprueba si el índice de la opción seleccionada es válido y además es distinto de cero. La primera opción de la lista (**- Selecciona un valor -**) no es válida, por lo que no se permite el valor **0** para esta propiedad **selectedIndex**.

1.2.4. Validar una dirección de email

Se trata de obligar al usuario a introducir una dirección de email con un formato válido. Por tanto, lo que se comprueba es que la dirección *parezca* válida, ya que no se comprueba si se trata de una cuenta de correo electrónico real y operativa. La condición JavaScript consiste en:

```
valor = document.getElementById("campo").value;  
if( !(/\w+([-.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)/.test(valor)) )  
{  
    return false;  
}
```

La comprobación se realiza nuevamente mediante las expresiones regulares, ya que las direcciones de correo electrónico válidas pueden ser muy diferentes. Por otra parte, como el estándar que define el formato de las direcciones de correo electrónico es muy complejo, la expresión regular anterior es una simplificación.

Aunque esta regla valida la mayoría de direcciones de correo electrónico utilizadas por los usuarios, no soporta todos los diferentes formatos válidos de email.

1.2.5. Validar una fecha

Las fechas suelen ser los campos de formulario más complicados de validar por la multitud de formas diferentes en las que se pueden introducir. El siguiente código asume que de alguna forma se ha obtenido el año, el mes y el día introducidos por el usuario:

```
var ano = document.getElementById("ano").value;
var mes = document.getElementById("mes").value;
var dia = document.getElementById("dia").value;

valor = new Date(ano, mes, dia);

if( !isNaN(valor) ) {
    return false;
}
```

La función **Date(ano, mes, dia)** es una función interna de JavaScript que permite construir fechas a partir del año, el mes y el día de la fecha. Es muy importante tener en cuenta que **el número de mes se indica de 0 a 11, siendo 0 el mes de Enero y 11 el mes de Diciembre. Los días del mes siguen una numeración diferente, ya que el mínimo permitido es 1 y el máximo 31.**

La validación consiste en intentar construir una fecha con los datos proporcionados por el usuario. Si los datos del usuario no son correctos, la fecha no se puede construir correctamente y por tanto la validación del formulario no será correcta.

1.2.6. Validar un número de DNI

Se trata de comprobar que el número proporcionado por el usuario se corresponde con un número válido de Documento Nacional de Identidad o DNI. Aunque para cada país o región los requisitos del documento de identidad de las personas pueden variar, a continuación se muestra un ejemplo genérico fácilmente adaptable. La validación no sólo debe comprobar que el número esté formado por ocho cifras y una letra, sino que también es necesario comprobar que la letra indicada es correcta para el número introducido:

```
valor = document.getElementById("campo").value;  
  
var letras = ['T', 'R', 'W', 'A', 'G', 'M', 'Y', 'F', 'P', 'D', 'X', 'B', 'N', 'J', 'Z', 'S', 'Q', 'V', 'H', 'L', 'C', 'K', 'E', 'T'];  
  
if( !(/^\d{8}[A-Z]$/.test(valor)) ) {  
    return false;  
}  
  
if(valor.charAt(8) != letras[(valor.substring(0, 8))%23]) {  
    return false;  
}
```

La primera comprobación asegura que el formato del número introducido es el correcto, es decir, que está formado por 8 números seguidos y una letra. Si la letra está al principio de los números, la comprobación sería `/^[A-Z]\d{8}$/`. Si en vez de ocho números y una letra, se requieren diez números y dos letras, la comprobación sería `/^\d{10}[A-Z]{2}$/` y así sucesivamente.

La segunda comprobación aplica el algoritmo de cálculo de la letra del DNI y la compara con la letra proporcionada por el usuario. El algoritmo de cada documento de identificación es diferente, por lo que esta parte de la validación se debe adaptar convenientemente.

1.2.7. Validar un número de teléfono

Los números de teléfono pueden ser indicados de formas muy diferentes: con prefijo nacional, con prefijo internacional, agrupado por pares, separando los números con guiones, etc.

El siguiente script considera que un número de teléfono está formado por nueve dígitos consecutivos y sin espacios ni guiones entre las cifras:

```
valor = document.getElementById("campo").value;  
if( !(/^\\d{9}$/.test(valor)) ) {  
    return false;  
}
```

Una vez más, la condición de JavaScript se basa en el uso de expresiones regulares, que comprueban si el valor indicado es una sucesión de nueve números consecutivos. A continuación se muestran otras expresiones regulares que se pueden utilizar para otros formatos de número de teléfono:

Número	Expresión regular	Formato
900900900	<code>/^\d{9}\$/</code>	9 cifras seguidas
900-900-900	<code>/^\d{3}-\d{3}-\d{3}\$/</code>	9 cifras agrupadas de 3 en 3 y separadas por guiones
900 900900	<code>/^\d{3}\s\d{6}\$/</code>	9 cifras, las 3 primeras separadas por un espacio
900 90 09 00	<code>/^\d{3}\s\d{2}\s\d{2}\s\d{2}\$/</code>	9 cifras, las 3 primeras separadas por un espacio, las siguientes agrupadas de 2 en 2
(900) 900900	<code>/^\(\d{3}\)\s\d{6}\$/</code>	9 cifras, las 3 primeras encerradas por paréntesis y un espacio de separación respecto del resto
+34 900900900	<code>/^\+\d{2,3}\s\d{9}\$/</code>	Prefijo internacional (+ seguido de 2 o 3 cifras), espacio en blanco y 9 cifras consecutivas

1.2.8. Validar que un checkbox ha sido seleccionado

Si un elemento de tipo *checkbox* se debe seleccionar de forma obligatoria, JavaScript permite comprobarlo de forma muy sencilla:

```
elemento = document.getElementById("campo");  
  
if( !elemento.checked ) {  
  
    return false;  
  
}
```

Si se trata de comprobar que todos los *checkbox* del formulario han sido seleccionados, es más fácil utilizar un bucle:

```
formulario = document.getElementById("formulario");  
  
for(var i=0; i<formulario.elements.length; i++) {  
  
    var elemento = formulario.elements[i];  
  
    if(elemento.type == "checkbox") {  
  
        if(!elemento.checked) {  
  
            return false;  
  
        }  
  
    }  
  
}
```

1.2.9. Validar que un radiobutton ha sido seleccionado

Aunque se trata de un caso similar al de los *checkbox*, la validación de los *radiobutton* presenta una diferencia importante: en general, la comprobación que se realiza es que el usuario haya seleccionado algún *radiobutton* de los que forman un determinado grupo. Mediante JavaScript, es sencillo determinar si se ha seleccionado algún *radiobutton* de un grupo:

```
opciones = document.getElementsByName("opciones");

var seleccionado = false;

for(var i=0; i<opciones.length; i++) {

    if(opciones[i].checked) {

        seleccionado = true;

        break;

    }

}

if(!seleccionado) {

    return false;

}
```

El anterior ejemplo recorre todos los **radiobutton** que forman un grupo y comprueba elemento por elemento si ha sido seleccionado. Cuando se encuentra el primer **radiobutton** seleccionado, se sale del bucle y se indica que al menos uno ha sido seleccionado.

1.3. Ejemplos de validación de datos en JavaScript

1.3.1. Ejemplo Validación De Textos

En el siguiente ejemplo, la función valida que el campo "alias" tenga una longitud entre 5 y 10 caracteres y que además esté en mayúsculas.

```
<!doctype html>

<html>

<head>

<meta charset="iso-8859-1">

<title>validar texto</title>

<script language="javascript" type="text/javascript">

function validar() {

    var texto=document.getElementById("textoalias").value;

    if (texto.length<5 ||texto.length>10){
```

```

    alert ("longitud de la cadena invalida");

    document.getElementById("textoalias").value="";

    return;

}

if (texto!=texto.toUpperCase()) {

    alert ("el texto no está en mayúsculas");

    document.getElementById("textoalias").value="";

    return;

}

}

</script>

<style type="text/css">

#contenedor {

    background-color: #FF9;

    height: auto;

    width: 35%;

    margin-right: auto;

    margin-left: auto;

    padding: 10px;

    border-width: 2px;

    border-style: groove;

    -webkit-border-radius: 16px;

    -moz-border-radius: 16px;

    border-radius: 16px;

}

label {

    color: #903;

}

```

```

.input_text {
    -webkit-border-radius: 6px;
    -moz-border-radius: 6px;
    border-radius: 6px;
}
</style>
</head>
<body>
<h3 align="center">EJEMPLO ALIAS ENTRE 5 Y 10 CARACTERES Y SOLO
PERMITIR MAYÚSCULAS</h3>
<div id="contenedor">
<form name="formulariotexto" method="post" action="">
<table width="99%" border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="58%"><label for="textfield">ALIAS:</label></td>
<td width="42%"><input type="text" class="input_text"
id="textoalias" onblur="validar();"></td>
</tr>
<tr>
<td><label for="textfield2">NOMBRE:</label></td>
<td><input type="text" class="input_text"
id="textonombre"></td>
</tr>
<tr>
<td><label for="textfield3">APELLIDOS:</label></td>
<td><input type="text" class="input_text"
id="textoapellidos"></td>
</tr>
</table>

```

```
</div>
</body>
</html>
```

1.3.2. Ejemplo Validación De Números

El siguiente ejemplo muestra un documento con una caja de texto para introducir una edad entre 0 y 120, informando de error si el texto introducido no es un número o bien no está comprendido en el rango válido.

```
<!doctype html>
<html>
<head>
<meta charset="iso-8859-1">
<title>validar texto</title>
<script language="javascript" type="text/javascript">
function validar() {
    var edad=document.getElementById("textoedad").value;
    if (isNaN(edad)) {
        alert ("La edad introducida no es un numero");
        return;
    }
    if (edad<0 || edad >120) {
        alert ("Valor de edad erroneo");
        return;
    }
    alert ("Tu edad es: "+edad);
}
</script>
<style type="text/css">
```

```
background-color: #FF9;
height: auto;
width: 35%;
margin-right: auto;
margin-left: auto;
padding: 10px;
border-width: 2px;
border-style: groove;
-webkit-border-radius: 16px;
-moz-border-radius: 16px;
border-radius: 16px;
}

label {
    color: #903;
}

.input_text {
    -webkit-border-radius: 6px;
    -moz-border-radius: 6px;
    border-radius: 6px;
}

</style>
</head>
<body>
<h3 align="center">EJEMPLO INTRODUCIR EDAD ENTRE 0 Y 120</h3>
<div id="contenedor">
<form name="formulariotexto" method="post" action="">
```

<table width="99%" border="0" cellspacing="0" cellpadding="0">

```

<tr>

    <td width="58%"><label for="textfield2">NOMBRE:</label></td>

    <td width="42%"><input type="text" class="input_text"
id="textonombre"></td>

</tr>

<tr>

    <td><label for="textfield">EDAD:</label></td>

    <td><input type="text" class="input_text" id="textoedad"
onblur="validar();"></td>

</tr>

</table>

</form>

</div>

</body>

</html>

```

1.3.3. Ejemplo Validación De Contraseña

Para realizar esta validación tan detallada, se recorrerá el campo contraseña carácter a carácter y se almacenarán en distintos contadores la cantidad de elementos requeridos, dígitos, mayúsculas, etc.

En el siguiente ejemplo muestra la validación de una contraseña.

```

<!doctype html>

<html>

<head>

<meta charset="iso-8859-1">

<title>validar contraseña</title>

<script language="javascript" type="text/javascript">

function validar() {

    var CantMay=0; var CantMin=0; var CantNum=0;

```

```
var passw=document.getElementById("password").value;
```

```
        if (passw.length < 7 || passw.length> 12) {  
            alert ("Contraseña con longitud inválida");  
            return;  
        }  
        for (cont=0; cont<passw.length; cont++) {  
            if ( passw.charAt(cont)>="0" &&  
                passw.charAt(cont)<="9") CantNum++;  
            else {  
                if ( passw.charAt(cont)==passw.charAt(cont).toUpperCase())  
                    CantMay++;  
                if ( passw.charAt(cont)==passw.charAt(cont).toLowerCase())  
                    CantMin++;  
            }  
        }  
        if (CantMay==0 || CantMin==0 || CantNum<2)  
            alert ("Contraseña no cumple criterios de seguridad");  
    }  
</script>  
<style type="text/css">  
#contenedor {  
    background-color: #FF9;  
    height: auto;  
    width: 35%;  
    margin-right: auto;  
    margin-left: auto;  
    padding: 10px;  
    border-width: 2px;
```

border-style: groove;

```

        -webkit-border-radius: 16px;

        -moz-border-radius: 16px;

        border-radius: 16px;
    }

    label {
        color: #903;
    }

    .input_text {
        -webkit-border-radius: 6px;

        -moz-border-radius: 6px;

        border-radius: 6px;
    }
</style>
</head>
<body>

<h3 align="center">EJEMPLO INTRODUCIR CONTRASEÑA ENTRE
7 Y 12 CARACTERES,

<BR>CON ALGUNA MAYÚSCULA, ALGUNA MINÚSCULA Y
AL MENOS 2 DÍGITOS NUMÉRICOS </h3>

<div id="contenedor">

<form name="formulariotexto" method="post" action="">

<table width="99%" border="0" cellspacing="0" cellpadding="0">

    <tr>

        <td width="58%"><label for="textfield2">NOMBRE:</label></td>

        <td width="42%"><input type="text" class="input_text"
id="textonombre"></td>

    </tr>

    <tr>

        <td><label for="textfield">CONTRASEÑA:</label></td>

```

```
<td><input type="text" class="input_text" id="password"
onblur="validar();"></td>
```

```
</tr>
```

```
</table>
```

```
</form>
```

```
</div>
```

```
</body>
```

```
</html>
```