DWEC
TEMA 5

GESTIÓN DE EVENTOS Y FORMULARIOS EN JAVASCRIPT

P E G O
IES Enric Valor

GENERALITAT VALENCIANA
CONSELLERIA D'EDUCACIÓ, CULTURA I ESPORT

Unió Europea
Fons Social Europeu
El FSE inverteix en el teu futur

ÍNDICE

5.1 EL OBJETO FORM

La mayor parte de interactividad entre una página web y el usuario tiene lugar a través de un formulario. Es ahí donde nos vamos a encontrar con los campos de texto, botones, checkboxes, listas, etc. en los que el usuario introducirá los datos, que luego se enviarán al servidor.

JavaScript añade a los formularios dos características muy interesantes:

- ✓ JavaScript permite examinar y validar las entradas de usuario directamente, en el lado del cliente.
- ✓ JavaScript permite dar mensajes instantáneos, con información de la entrada del usuario.

5.1 EL OBJETO FORM

El objeto de JavaScript **Form**, es una propiedad del objeto **document**. Se corresponderá con la etiqueta **<FORM>** del HTML. Un formulario podrá ser enviado llamando al método **submit** de JavaScript, o bien haciendo click en el botón **submit** del formulario.

5.1 EL OBJETO FORM

El objeto de JavaScript **Form**, es una propiedad del objeto **document**. Se corresponderá con la etiqueta **<FORM>** del HTML. Un formulario podrá ser enviado llamando al método **submit** de JavaScript, o bien haciendo click en el botón **submit** del formulario.

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

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form

5.1 EL OBJETO FORM

Formas de selección del objeto Form.

Si partimos del siguiente ejemplo:

```
<div  id="menulateral">
     <form  id="contactar" name="contactar" action="...">...</form>
</div>
...
```

Tendremos los siguientes métodos de selección:

Método 1: A través del método **getElementById()** del DOM.

```
var formulario=document.getElementById("contactar");
```

Método 2: A través del método **getElementsByTagName()** del DOM.

```
var primerFormulario = document.getElementsByTagName("form")[0] ;
```

Método 3: Otro método puede ser a través de la colección **forms[]** del **objeto document**.

```
var miformulario = document.forms[0];     // primer formulario del documento
```

O bien:

```
var  miformulario  =  formularios["contactar"];  //  referenciamos  al  formulario  con  name
"contactar"
```

5.1 EL OBJETO FORM

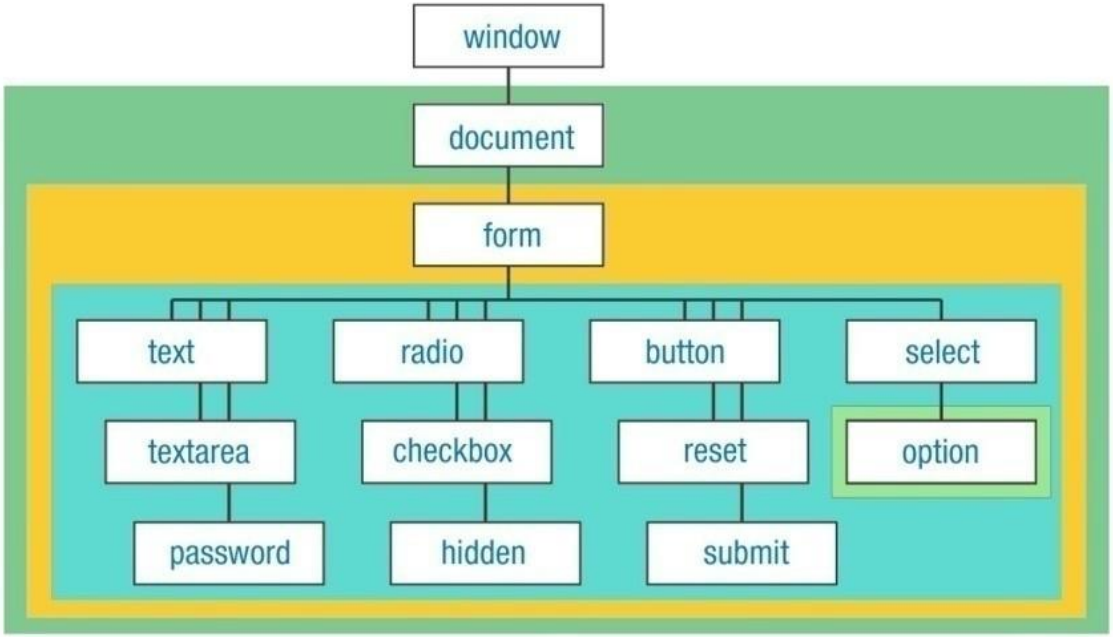
El formulario como objeto y contenedor.
Debido a que el DOM ha ido evolucionando con las nuevas versiones de JavaScript, nos encontramos con que el objeto **Form** está dentro de dos árboles al mismo tiempo.

En las nuevas definiciones del DOM, se especifica que **Form** es el padre de todos sus nodos hijos, incluidos objetos y textos,

Mientras que en las versiones antiguas **Form** sólo era padre de sus objetos (input, select, button y elementos textarea).

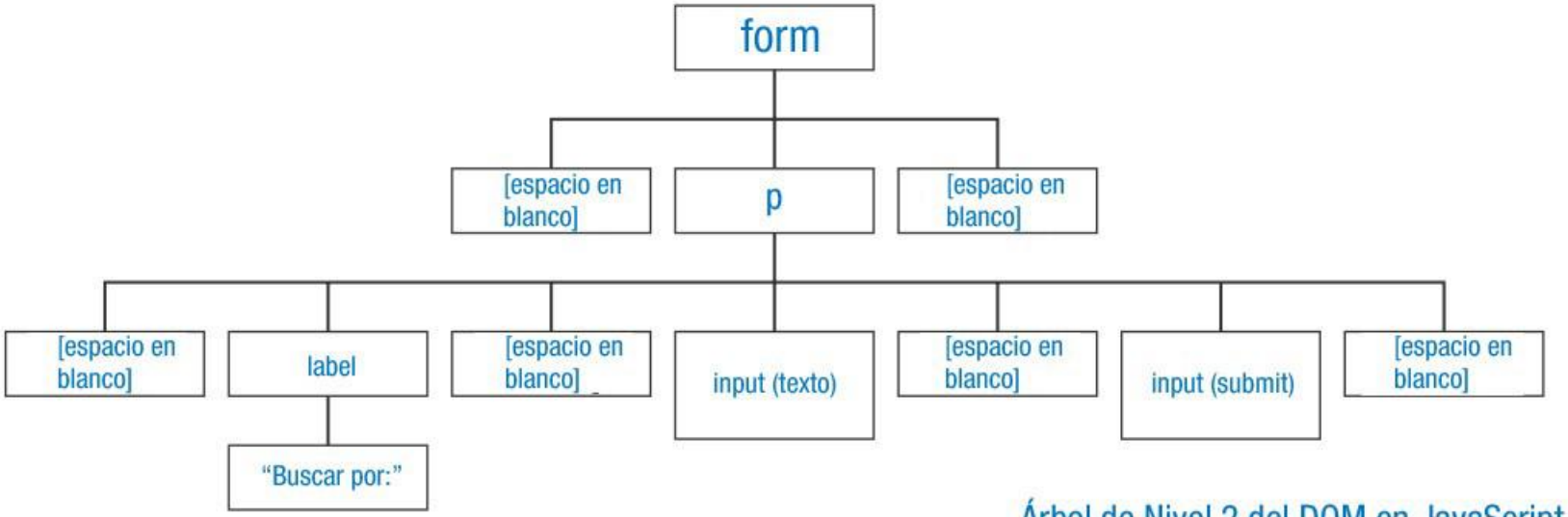
5.1 EL OBJETO FORM

El formulario como objeto y contenedor.
Jerarquía de nivel 0 del DOM para formularios y controles:



5.1 EL OBJETO FORM

El formulario como objeto y contenedor.
Ejemplo cómo sería el árbol de nivel 2 del DOM para un formulario típico:



Árbol de Nivel 2 del DOM en JavaScript

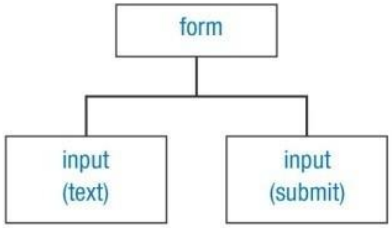
5.1 EL OBJETO FORM

El formulario como objeto y contenedor.

Partimos del siguiente código de ejemplo:

```
<form  action="buscar.php" name="elFormulario" id="miFormulario" method="post"> por
<p>
<label  for="busqueda">Buscar por:</label>
<input  id="busqueda"  name="busqueda"  type="text"  value="">
<input  id="submit"  type="submit"  value="Buscar">
</p>
</form>
```

El árbol del DOM de nivel 0, hace muchísimo más fácil leer y escribir los controles del formulario.




```
var elFormulario = document.getElementById("miFormulario");
var control = elFormulario.busqueda;
```

5.1 EL OBJETO FORM

Acceso a propiedades y métodos del formulario

Los formularios pueden ser creados a través de las etiquetas HTML, o utilizando JavaScript y métodos del DOM. En cualquier caso se pueden asignar atributos como **name**, **action**, **target** y **enctype**. Cada uno de estos atributos es una propiedad del objeto **Form**, a las que podemos acceder utilizando su nombre en minúsculas, por ejemplo:

```
var paginaDestino = objetoFormulario.action;
```

Para modificar una de estas propiedades lo haremos mediante asignaciones, por ejemplo:

```
objetoFormulario.action = "http://www.educacion.gob.es/recepcion.php";
```

5.1 EL OBJETO FORM

Propiedad form.elements[]

La propiedad **elements[]** de un formulario es una colección, que contiene todos los objetos **input** dentro de un formulario. Esta propiedad es otro array, con todos los campos **input** en el orden en el cual aparecen en el código fuente del documento.

Generalmente, es mucho más eficaz y rápido referenciar a un elemento individual usando su ID, pero a veces, los scripts necesitan recorrer cada elemento del formulario, para comprobar que se han introducido sus valores correctamente.

```
var miFormulario = document.getElementById("contactar");
// guardamos la referencia del formulario en una variable.

if (! miFormulario) return false; // Si no existe ese formulario devuelve false.

for (var i=0; i< miFormulario.elements.length; i++){
     if (miFormulario.elements[i].type == "text"){
          miFormulario.elements[i].value = "";
     }
}
```

5.2 OBJETOS RELACIONADOS CON FORMULARIOS

Para poder trabajar con los objetos de un formulario, lo primero que necesitas saber es, cómo referenciar a ese objeto.

Eso puedes hacerlo directamente a través de su ID, o bien con su nombre de etiqueta, empleando para ello los métodos del DOM nivel 2. O también se puede hacer usando la sintaxis del DOM nivel 0, y construyendo la jerarquía que comienza por **document**, luego el formulario y finalmente el control.

Lo mejor es identificar cada uno de los objetos con un atributo **id** que sea único, y que no se repita en el documento, así para acceder a cualquier objeto dentro de nuestro documento o formulario lo haremos con:

5.2 OBJETOS RELACIONADOS CON FORMULARIOS

Por ejemplo, si consideramos un ejemplo sencillo de formulario:

```
<form  id="formularioBusqueda"  action="cgi-bin/buscar.pl">
   <p>
      <input  type="text"  id="entrada"  name="cEntrada">
      <input  type="submit"  id="enviar"  name="enviar"  value="Buscar...">
   </p>
</form>
```

Las siguientes referencias al campo de texto entrada, serán todas válidas:

```
document.getElementById("entrada")
document.formularioBusqueda.cEntrada
document.formularioBusqueda.elements[0]
document.forms["formularioBusqueda"].elements["cEntrada"]
document.forms["formularioBusqueda"].cEntrada
```

NOTA:
A la hora de identificar los objetos en un formulario lo más recomendable es que el atributo id y el atributo name sean iguales y que no se repitan los id en todo el documento.

5.2 OBJETOS RELACIONADOS CON FORMULARIOS

Objeto input de tipo texto.

Cada uno de los 4 elementos de tipo texto de los formularios: **text**, **password**, **hidden** y elementos **textarea**, son un elemento dentro de la jerarquía de objetos. Todos los elementos, excepto los tipos **hidden**, se mostrarán en la página, permitiendo a los usuarios introducir texto y seleccionar opciones.

Cuando se envían los datos de un formulario a un programa en el lado del servidor, lo que en realidad se envía son los atributos **name**, junto con los valores (contenido del atributo **value**) de cada elemento

5.2 OBJETOS RELACIONADOS CON FORMULARIOS

Objeto input de tipo checkbox.
Un **checkbox** es también un objeto muy utilizado en los formularios.

```
<label  for="cantidad">Si desea recibir 20 Kg marque esta opción: </label>
<input  type="checkbox"  id="cantidad"  name="cantidad"  value="20 Kg">
```

Si chequeamos este **checkbox** y enviamos el formulario, el navegador enviará el par **name/value** "cantidad" y "20 Kg". Si el **checkbox** no está marcado, entonces este campo no será enviado en el formulario.
El texto del **label** se mostrará en la pantalla pero las etiquetas **label** no se envían al servidor. Para saber si un campo de tipo **checkbox** está o no marcado, disponemos de la propiedad **checked**.
Esta propiedad contiene un valor booleano: true si el campo está marcado o false si no está marcado. Con esta propiedad es realmente sencillo comprobar o ajustar la marca en un campo de tipo **checkbox**.

5.2 OBJETOS RELACIONADOS CON FORMULARIOS

Objeto input de tipo radio.
Para dejar que el navegador gestione un grupo de objetos de tipo radio, deberemos asignar el mismo atributo **name** a cada uno de los botones del grupo. Podemos tener múltiples grupos de botones de tipo radio en un formulario, pero cada miembro de cada grupo tendrá que tener el mismo atributo **name** que el resto de compañeros del grupo.
Por ejemplo, podemos ver cuántos botones hay en un grupo **radio**, consultando la propiedad **length** de ese grupo:

```
objetoFormulario.nombregrupo.length
```

Y si queremos acceder a la propiedad **checked** de un botón en particular, lo haremos accediendo a la posición del array y a la propiedad **checked**:

```
objetoFormulario.nombregrupo[0].checked      // Accedemos a la propiedad checked del primer botón del grupo
```

5.2 OBJETOS RELACIONADOS CON FORMULARIOS

Objeto select.
Uno de los controles más complejos que te puedes encontrar en formularios, es el objeto **select**. Un objeto **select** está compuesto realmente de un array de objetos **option**.
Algunas propiedades pertenecen al objeto **select** al completo, mientras que otras, por ejemplo, sólo se pueden aplicar a las opciones individuales dentro de ese objeto.
La propiedad más importante del objeto **select** es la propiedad **selectedIndex**, a la que puedes acceder de las siguientes formas:

```
objetoFormulario.nombreCampoSelect.selectedIndex
document.getElementById("objetoSelect").selectedIndex
```

Las opciones tienen dos propiedades accesibles que son **text** y **value**, y que te permitirán acceder al texto visible en la selección y a su valor interno para esa opción (ejemplo: **<option value="OU" >Ourense</option>**). Veamos las formas de acceso a esas propiedades:

```
objetoFormulario.nombreCampoSelect.options[n].text
objetoFormulario.nombreCampoSelect.options[n].value
```

5.3 EVENTOS

Modelo eventos W3C.
El W3C en la especificación del DOM de nivel 2, pone especial atención en los problemas del modelo tradicional de registro de eventos. En este caso ofrece una manera sencilla de registrar los eventos que queramos, sobre un objeto determinado.

Este método tiene tres **argumentos**: el **tipo de evento**, la **función a ejecutar** y un **valor** booleano (true o false), que se utiliza para indicar cuándo se debe capturar el evento: en la fase de captura (true) o de burbujeo (false).

```
elemento.addEventListener('evento', función, false|true)
```

También se pueden usar funciones anónimas (sin nombre de función), con el modelo W3C:

```
element.addEventListener('click', function () {
     this.style.backgroundColor = '#cc0000';
```

5.3 EVENTOS

Modelo eventos W3C.
El W3C en la especificación del DOM de nivel 2, pone especial atención en los problemas del modelo tradicional de registro de eventos. En este caso ofrece una manera sencilla de registrar los eventos que queramos, sobre un objeto determinado.
Este método tiene tres **argumentos**: el **tipo de evento**, la **función a ejecutar** y un **valor** booleano (true o false), que se utiliza para indicar cuándo se debe capturar el evento: en la fase de captura (true) o de burbujeo (false).

La ventaja de este método, es que podemos añadir tantos eventos como queramos. Por ejemplo:

```
document.getElementById("mienlace").addEventListener('click',alertar,false);
document.getElementById("mienlace").addEventListener('click',avisar,false);
document.getElementById("mienlace").addEventListener('click',chequear,false);
```

Para cancelar un evento, este modelo nos proporciona el método **preventDefault()**.

5.3 EVENTOS

Orden de disparo de los eventos.
Imagina que tenemos un elemento contenido dentro de otro elemento, y que tenemos programado el mismo tipo de evento para los dos (por ejemplo el evento click). ¿Cuál de ellos se disparará primero? Sorprendentemente, esto va a depender del tipo de navegador que tengamos.
El problema es muy simple. Imagina que tenemos el siguiente gráfico:

y ambos tienen programado el evento de click. Si el usuario hace click en el elemento2, provocará un click en ambos: elemento1 y elemento2. ¿Pero cuál de ellos se disparará primero?, ¿cuál es el orden de los eventos?

5.3 EVENTOS

Orden de disparo de los eventos.

Tenemos **dos Modelos propuestos** por Netscape y Microsoft en sus orígenes:

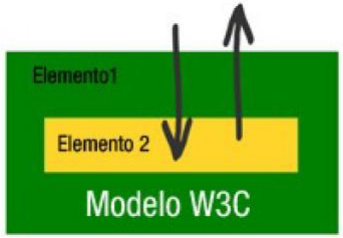
- Netscape dijo que, el evento en el elemento1 tendrá lugar primero. Es lo que se conoce como "**captura de eventos".**
- Microsoft mantuvo que, el evento en el elemento2 tendrá precedencia. Es lo que se conoce como **"burbujeo de eventos".**

5.3 EVENTOS

Orden de disparo de los eventos.

Modelo W3C

W3C decidió tomar una posición intermedia. Así supone que, cuando se produce un evento en su modelo de eventos, primero se producirá la fase de captura hasta llegar al elemento de destino, y luego se producirá la fase de burbujeo hacia arriba. Este modelo es el estándar, que todos los navegadores deberían seguir para ser compatibles entre sí.



Tú podrás decidir cuando quieres que se registre el evento: en la fase de captura o en la fase de burbujeo. El tercer parámetro de **addEventListener** te permitirá indicar si lo haces en la **fase de captura (true)**, o en la **fase de burbujeo(false)**.

5.4 ENVÍO Y VALIDACIÓN DE FORMULARIOS

La validación de un formulario es un proceso que consiste en chequear un formulario y comprobar que todos sus datos han sido introducidos correctamente. Por ejemplo, si tu formulario contiene un campo de texto en el que hay que escribir un e-mail, sería interesante comprobar si ese e-mail está escrito correctamente, antes de pasar al siguiente campo.

Hay tres métodos de validación de formularios:

- A partir de **HTML5** se pueden validar los formularios directamente en las etiquetas.
- En el **lado del servidor** (usando scripts CGI, PHP, ASP, etc.)
- En el **lado del cliente** (generalmente usando JavaScript).

NOTA: Se deben de utilizar los tres métodos de validación.

5.4 ENVÍO Y VALIDACIÓN DE FORMULARIOS

JavaScript añade a tus formularios dos características muy interesantes:

- JavaScript te permite examinar y validar las entradas de usuario directamente, en el lado del cliente.
- JavaScript te permite dar mensajes instantáneos, con información de la entrada del usuario.

La validación de datos del usuario en la entrada, generalmente suele fallar en alguna de las 3 siguientes categorías:

- **Existencia**: comprueba cuando existe o no un valor.
- **Numérica**: que la información contenga solamente valores numéricos.
- **Patrones**: comprueba que los datos sigan un determinado patrón, como el formato de un e-mail, una fecha, un número de teléfono, un número de la seguridad social, etc.

También se puede utilizar para modificar los elementos de un formulario, basándose en los datos introducidos por el usuario: tal como cubrir un campo de selección con una lista de nombres de ciudades, cuando una determinada provincia está seleccionada, etc.

5.4 ENVÍO Y VALIDACIÓN DE FORMULARIOS

Ejemplo formulario sin validación

```
<form action="procesar.php" method="post" id=elFormulari">
    <table>
        <tr>
            <td> Nom: </td>
            <td>
                <input type="text" name="nom" id="nom" />
            </td>
        </tr>
        <tr>
            <td> data neixement: </td>
            <td>
                <input type="text" name="neix" id="neix" />
            </td>
        </tr>
        <tr>
            <td> telèfon: </td>
            <td>
                <input type="text" name="tel" id="tel" />
            </td>
        </tr>
        <tr>
            <td> </td>
            <td>
            <input type="reset" name="limpiar" id="limpiar" value="Reset"/>
           <input type="submit" name="enviar" id="enviar" value="Enviar"/> </td>
        </tr>

    </table>
</form>
```

5.4 ENVÍO Y VALIDACIÓN DE FORMULARIOS

Ejemplo – 1er paso: **HTML5**

```
<form action="procesar.php" method="post" id=elFormulari">
    <table>
        <tr>
            <td> Nom: </td>
            <td>
                <input type="text" name="nom" id="nom" maxlength="15"
                 pattern="[A-Za-z]{2,15}" title="Introdueix entre 2 i 15 lletres" required/>
            </td>
        </tr>
        <tr>
            <td> data neixement: </td>
            <td>
                <input type="date" name="neix" id="neix" step="1"
                min="1900-01-01" max="2021-12-31" required/>
            </td>
        </tr>
        <tr>
            <td> telèfon: </td>
            <td>
                <input type="text" name="tel" id="tel" pattern="^\d{3}\s\d{3}\s\d{3}$"
                title="El telefono ha de ser amb el següent  format 999 999 999" required/>
            </td>
        </tr>
        <tr>
            <td> </td>
             <td>
            <input type="reset" name="limpiar" id="limpiar" value="Reset"/>
           <input type="submit" name="enviar" id="enviar" value="Enviar"/> </td>
        </tr>

    </table>

</form>
```

5.4 ENVÍO Y VALIDACIÓN DE FORMULARIOS

Ejemplo – 2n paso: **HTML5**

```
<form action="procesar.php" method="post" id=elFormulari">
    <table>
        <tr>
            <td> Nom: </td>
            <td>
                <input type="text" name="nom" id="nom" maxlength="15"
                 pattern="[A-Za-z]{2,15}" title="Introdueix entre 2 i 15 lletres" required/>
            </td>
        </tr>
        <tr>
            <td> data neixement: </td>
            <td>
                <input type="date" name="neix" id="neix" step="1"
                min="1900-01-01" max="2021-12-31" required/>
            </td>
        </tr>
        <tr>
            <td> telèfon: </td>
            <td>
                <input type="text" name="tel" id="tel" pattern="^\d{3}\s\d{3}\s\d{3}$"
                title="El telefono ha de ser amb el següent  format 999 999 999" required/>
            </td>
        </tr>
        <tr>
            <td> </td>
             <td>
            <input type="reset" name="limpiar" id="limpiar" value="Reset"/>
           <input type="submit" name="enviar" id="enviar" value="Enviar"/> </td>
        </tr>

    </table>

</form>
```

5.4 ENVÍO Y VALIDACIÓN DE FORMULARIOS

Ejemplo – 3r paso: **HTML5 + JS**

```
window.onload = iniciar;

function iniciar (){

    document.getElementById("enviar").addEventListener("click", validar,false);

}

function validarNom () {
      var element = document.getElementById("nom");
      if (!element.checkValidity()){
          if (element.validity.valueMissing){
              error2(element,"Deus d'introduïr un nom.");
          }
          if (element.validity.patternMismatch){
              error2(element, "El nom ha de tindre entre 2 i 14 caracters.");
          }
          //error(element);
          return false;
      }
      return true;

}
```

5.4 ENVÍO Y VALIDACIÓN DE FORMULARIOS

Ejemplo – 3r paso: **HTML5 + JS**

```
function validarNeix (){
    var element = document.getElementById("neix");
    if (!element.checkValidity()){
        if (element.validity.valueMissing){
            error2(element,"Deus d'introduïr una data.");
        }
        if (element.validity.rangeOverflow){
            error2(element, "La data mínima ha de ser superior al 01/01/1900.");
        }
        if (element.validity.rangeUnderflow){
            error2(element, "La data màxima ha de ser inferior al 31/12/2020.");
        }
        //error(element);
        return false;
    }
    return true;
}
```

5.4 ENVÍO Y VALIDACIÓN DE FORMULARIOS

Ejemplo – 3r paso: **HTML5 + JS**

```
function validarTel (){
    var element = document.getElementById("tel");
    if (!element.checkValidity()){
        if (element.validity.valueMissing){
            error2(element,"Deus d'introduïr un telèfon.");
        }
        if (element.validity.patternMismatch){
            error2(element,"El telèfon ha de tindre el format 999 999 999.");
        }
        //error(element);
        return false;
    }
    return true;
}
```

5.4 ENVÍO Y VALIDACIÓN DE FORMULARIOS

Ejemplo – 3r paso: **HTML5 + JS**

```
function validar (e) {
    esborrarError ();
    if (validarNom() && validarNeix() && validarTel() && confirm("Confirma si vols enviar el formulari") ){

        return true;

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

5.4 ENVÍO Y VALIDACIÓN DE FORMULARIOS

Ejemplo – 3r paso: **HTML5 + JS**

```
function error2 (element, missatge){
    document.getElementById("missatgeError").innerHTML = missatge;
    element.className ="error";
    element.focus();
}


function esborrarError (){
    var formulari = document.forms[0];
        for ( var i=0; i < formulari.elements.length; i++){
            formulari.elements[i].className="";
        }
}
```

5.5 EXPRESIONES REGULARES

LAS EXPRESIONES REGULARES SON PATRONES DE BÚSQUEDA, QUE SE PUEDEN UTILIZAR PARA ENCONTRAR TEXTO QUE COINCIDA CON EL PATRÓN ESPECIFICADO.

En JavaScript las expresiones regulares se gestionan a través del objeto **RegExp**. Para crear un literal del tipo **RegExp**, tendrás que usar la siguiente sintaxis:

```
var  expresion = /expresión regular/;
```

5.5 EXPRESIONES REGULARES

ENCONTRAR TEXTO QUE COINCIDA CON EL PATRÓN ESPECIFICADO.

Expresiones Regulares de Microsoft

Test de expresiones regulares Online

Expresiones Regulares de 3WSchool

5.5 EXPRESIONES REGULARES

Carácter	Coincidencias	Patrón	Ejemplo de cadena
^	Al inicio de una cadena	/^Esto/	Coincidencia en "Esto es...".
$	Al final de la cadena	/final$/	Coincidencia en "Esto es el final".
*	Coincide 0 o más veces	/se*/	Que la "e" aparezca 0 o más veces: "seeee" y también "se".
?	Coincide 0 o 1 vez	/ap?	Que la p aparezca 0 o 1 vez: "apple" y "and".
+	Coincide 1 o más veces	/ap+/	Que la "p" aparezca 1 o más veces: "apple" pero no "and".
{n}	Coincide exactamente n veces	/ap{2}/	Que la "p" aparezca exactamente 2 veces: "apple" pero no "apabullante".
{n,}	Coincide n o más veces	/ap{2,}/	Que la "p" aparezca 2 o más veces: "apple" y "appple" pero no en "apabullante".
{n,m}	Coincide al menos n, y máximo m veces	/ap{2,4}/	Que la "p" aparezca al menos 2 veces y como máximo 4 veces: "apppppple" (encontrará 4 "p").
.	Cualquier carácter excepto nueva línea	/a.e/	Que aparezca cualquier carácter, excepto nueva línea entre la a y la e: "ape" y "axe".
[...]	Cualquier carácter entre corchetes	/a[px]e/	Que aparezca alguno de los caracteres "p" o "x" entre la a y la e: "ape", "axe", pero no "ale".
[^...]	Cualquier carácter excepto los que están entre corchetes	/a[^px]/	Que aparezca cualquier carácter excepto la "p" o la "x" después de la letra a: "ale", pero no "axe" o "ape".
\b	Coincide con el inicio de una palabra	/\bno/	Que "no" esté al comienzo de una palabra: "novedad".

5.5 EXPRESIONES REGULARES

Carácter	Coincidencias	Patrón	Ejemplo de cadena
\B	Coincide al final de una palabra	/\Bno/	Que "no" esté al final de una palabra: "este invierno" ("no" de "invierno").
\d	Dígitos del 0 al 9	/\d{3}/	Que aparezcan exactamente 3 dígitos: "Ahora en 456".
\D	Cualquier carácter que no sea un dígito	/\D{2,4}/	Que aparezcan mínimo 2 y máximo 4 caracteres que no sean dígitos: encontrará la cadena "Ahor" en "Ahora en 456".
\w	Coincide con caracteres del tipo (letras, dígitos, subrayados)	/\w/	Que aparezca un carácter (letra, dígito o subrayado): "J" en "JavaScript".
\W	Coincide con caracteres que no sean (letras, dígitos, subrayados)	/\W/	Que aparezca un carácter (que no sea letra, dígito o subrayado): "%" en "100%".
\n	Coincide con una nueva línea		
\s	Coincide con un espacio en blanco		
\S	Coincide con un carácter que		
	no es un espacio en blanco		
\t	Un tabulador		
(x)	Capturando paréntesis		Recuerda los caracteres.
\r	Un retorno de carro		
?=n	Cualquier cadena que está seguida por la cadena n indicada después del igual.	/la(?= mundo)	Hola mundo mundial.

5.5 EXPRESIONES REGULARES

El objeto RegExp.

El objeto **RegExp** es tanto un literal como un objeto de JavaScript, por lo que también se podrá crear usando un constructor:

```
var expresionregular = new RegExp("Texto Expresión Regular");
```

Propiedades del objeto RegExp	
Propiedad	**Descripción**
global	Especifica que sea utilizado el modificador "**g**".
ignoreCase	Especifica que sea utilizado el modificador "**i**".
lastIndex	El índice donde comenzar la siguiente búsqueda.
multiline	Especifica si el modificador "**m**" es utilizado.
source	El texto de la expresión regular **RegExp**.

Métodos del objeto RegExp	
Método	**Descripción**
compile()	Compila una expresión regular.
exec()	Busca la coincidencia en una cadena. Devolverá la primera coincidencia.
test()	Busca la coincidencia en una cadena. Devolverá `true` o `false`.

5.5 EXPRESIONES REGULARES

Ejemplo Comprobación cadena.

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Expresiones 1</title>
    <script>
        // Se crea una expresión regular mediante el  correspondiente constructor.
        var expresion_1 = new RegExp (/\d\d\d/);
        // Entramos en un bucle que nos pedirà una cadena
        do{
            cadena = prompt("cadena", "");
            // Comprobamos si la cadena coincide
            if (expresion_1.test(cadena)){
                alert ("coincide");
            }else{
                alert("NO coincide");
            }
        }while (cadena != null);
    </script>
</head>
<body>
</body>
</html>
```

5.5 EXPRESIONES REGULARES

Ejemplo 1 Validación.

```
<body>
    <form name="formulario" id="formulario" method="post" action="">
        <p> <label for="nombre">Nombre;</label> <input type="text" name="nombre" id="nombre"> </p>
        <p> <label for="telefono">Teléfono:</label> <input type="text" name="telefono" id="telefono"> </p>
        <p> <label for="matricula">Matrícula coche:</label> <input type="text" name="matricula" id="matricula"> </p>
        <p> <input type="reset" name="button" id="button" value="Restablecer"> <input type="button" name="enviar"
                id="enviar" value="Enviar"> </p>

            <br />
    </form>
</body>
```

5.5 EXPRESIONES REGULARES

Ejemplo 1 Validación.

```
window.onload = iniciar;

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

function validar() {
    // Teléfono: 123456789
    var patronTelefono= /^\d{9}$/;
    //Matrícula: 1234 ABC
    var patronMatricula=/^\d{4} [A-Z]{3}$/;
    if (patronTelefono.test(document.getElementById("telefono").value)){
        if(patronMatricula.test(document.getElementById("matricula").value)){
            if(confirm("¿Desea enviar el formulario?")){
                document.getElementById("formulario").submit();
            }
         }else{
            alert("Error: campo matricula incorrecta");
            }
        }else{
            alert("Error: campo telefono incorrecto");
    }
}
```