

## PRÁCTICA 12: FORMULARIOS

Duración estimada: 50 min.

Objetivos:

- Validación de Formularios y expresiones regulares en JavaScript.

Bibliografía: [Internet](#)

### Formularios

#### Propiedades y métodos de formularios

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:

```
let paginaDestino = objetoFormulario.action;
```

- **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.

Por ejemplo, empleando la propiedad elements[], podemos hacer un bucle que recorra un formulario y si los campos son de tipo texto, pues que los ponga en blanco:

```
let miFormulario = document.getElementById("contactar");    //  
guardamos la referencia del formulario en una variable.  
  
if (! miFormulario) return false;        // Si no existe ese formulario  
devuelve false.  
for (let i=0; i< miFormulario.elements.length; i++) {  
    if (miFormulario.elements[i].type == "text") {
```

```
miFormulario.elements[i].value = "";
```

```
}
```

```
}
```

### *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 dos métodos principales de validación de formularios: en el lado del servidor (usando scripts CGI, PHP, ASP, etc.) y en el lado del cliente (generalmente usando JavaScript).

La validación en el lado del servidor es más segura, pero a veces también es más complicada de programar, mientras que la validación en el lado del cliente es más fácil y más rápida de hacer (el navegador no tiene que conectarse al servidor para validar el formulario, por lo que el usuario recibirá información al instante sobre posibles errores o fallos encontrados).

La idea general que se persigue al validar un formulario, es que cuando se envíen los datos al servidor, éstos vayan correctamente validados, sin ningún campo con valores incorrectos.

A la hora de programar la validación, podremos hacerlo a medida que vamos metiendo datos en el formulario, por ejemplo campo a campo, o cuando se pulse el botón de envío del formulario.

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.

JavaScript 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.

Una parte muy importante que no debes olvidar al usar JavaScript con formularios, es la posibilidad de que el usuario desactive JavaScript en su navegador, por lo que JavaScript no debería ser una dependencia en la acción de envío de datos desde un formulario.

**"Acuérdate de que JavaScript está para mejorar, no para reemplazar".**

La validación de un formulario en el lado del cliente puede ahorrar algunas idas y vueltas a la hora de enviar los datos, pero aún así, tendrás que realizar la validación de datos en el servidor, puesto que es allí realmente donde se van a almacenar esos datos y el origen de los mismos puede venir por cauces que no hemos programado.

### *Ejemplo de validación de un formulario*

```
<!DOCTYPE html>
<html>

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

    <title>DWEC05 - Validación de un Formulario</title>

    <script type="text/javascript">

      const validar = () => {

        let edad = document.getElementById('edad').value;

        if ( edad >= 18 && confirm("¿Deseas enviar el formulario?"))

          return true;

        else

        {

          alert('Lo siento, pero no eres mayor de edad.')

          this.event.preventDefault();

          return false;

        }

      }

    </script>

    <style type="text/css">

      label{

        width: 150px;

        float:left;

        margin-bottom:5px;

      }

    </style>

  </head>

</html>
```

```

    }

    input, select {

        width:150px;

        float:left;

        margin-bottom:5px;

    }

    fieldset{

        background:#CCFF99;

        width:350px;

    }

    .error{

        border: solid 2px #FF0000;

    }

}

</style>
</head>
<body>

    <fieldset>

        <legend>DWEC05 - Validación de un Campo -</legend>

        <form name="formulario" id="formulario" action="http://www.google.es" method="get">

            <label for="edad">Edad:</label>

            <input name="edad" type="text" id="edad" maxlength="3" />

            <input type="reset" name="limpiar" id="limpiar" value="Limpiar" />

            <input type="submit" name="enviar" id="enviar" value="Enviar" onClick="validar()"/>

        </form>

    </fieldset>

</body>
</html>

```

## Expresiones regulares y objetos RegExp

Las expresiones regulares son patrones de búsqueda, que se pueden utilizar para encontrar texto que coincida con el patrón especificado.

Ejemplo de búsqueda de una cadena de texto sin usar expresiones regulares:

```

let texto = "La linea de alta velocidad llegará pronto a toda España,";
let subcadena = "velocidad";
let indice = texto.indexOf(subcadena);    // devuelve 17, índice de donde
se encuentra la subcadena
if (indice != -1)
    // correcto, se ha encontrado la subcadena
....

```

**Cuando estamos buscando cadenas que cumplen un patrón en lugar de una cadena exacta, necesitaremos usar expresiones regulares.** Podrías intentar hacerlo con funciones de `String`, pero al final, es mucho más sencillo hacerlo con expresiones regulares, aunque la sintaxis de las mismas es un poco extraña y no necesariamente muy amigable.

**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:

```
let expresion = /expresión regular/;
```

La expresión regular está contenida entre la barras `/`, y fíjate que no lleva comillas. Las comillas sólo se pondrán en la expresión regular, cuando formen parte del patrón en si mismo.

Las expresiones regulares están hechas de caracteres, solos o en combinación con caracteres especiales, que se proporcionarán para búsquedas más complejas. Por ejemplo, lo siguiente es una expresión regular que realiza una búsqueda que contenga las palabras ***Aloe Vera***, en ese orden y separadas por uno o más espacios en medio:

```
let expresion = /Aloe\s+Vera/;
```

Los caracteres especiales en este ejemplo son, la barra invertida (`\`), que tiene dos efectos: ***o bien se utiliza con un carácter regular, para indicar que se trata de un carácter especial, o se usa con un carácter especial, tales como el signo más (+), para indicar que el carácter debe ser tratado literalmente.*** En este caso, la barra invertida se utiliza con "s", que transforma la letra s en un carácter especial indicando un espacio en blanco, un tabulador, un salto de línea, etc. El símbolo + indica que el carácter anterior puede aparecer una o más veces.

**Caracteres especiales utilizados en 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 "apple" 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".
\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		
(...)	Capturando paréntesis		Recuerda los caracteres.
\r	Un retorno de carro		
?=n	Cualquier cadena que está seguida por la cadena n indicada después del igual.	/!a(?! mundo)/	Hola mundo mundial.

inEva

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

***let expresionRegular = new RegExp("Texto Expresión Regular");***

## ¿Cuándo usar el literal o el objeto?

La expresión **RegExp** literal es compilada cuando se ejecuta el script, por lo tanto se recomienda usar el literal cuando sabemos que la expresión no cambiará. Una versión compilada es mucho más eficiente.

Usaremos el objeto, cuando sabemos que la expresión regular va a cambiar, o cuando vamos a proporcionarla en tiempo de ejecución.

Al igual que otros objetos en JavaScript, el objeto **RegExp** también tiene sus propiedades y métodos:

## Propiedades del objeto RegExp

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

## Métodos del objeto RegExp

Método	Descripción
<code>compile()</code>	Compila una expresión regular.
<code>exec()</code>	Busca la coincidencia en una cadena. Devolverá la primera coincidencia.
<code>test()</code>	Busca la coincidencia en una cadena. Devolverá true o false.

## Ejemplo de expresiones Regulares

### Validar un número de teléfono

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <title>Ejemplo de RegExp</title>
    <script type="text/javascript">
      const iniciar = () => {
        document.getElementById("comprobar").onclick =
comprobarTelefono;
      }
      const comprobarTelefono = () => {
        let telefono = document.getElementById("telefono").value;
        let patron = /^\d{9}$/;
        if (telefono.match(patron))
          alert('Teléfono Correcto!!!!');
        else
          alert('Teléfono INCORRECTO!!!!');
      }
      window.onload = iniciar;
    </script>
  </head>
  <body>
    <form name="formulario">
      <label for="telefono">Teleono:</label>
```

```
        <input type="text" name="telefono" id="telefono" />
        <input type="button" name="comprobar" id="comprobar"
value="Comprobar Formato" />
    </form>
</body>
</html>
```

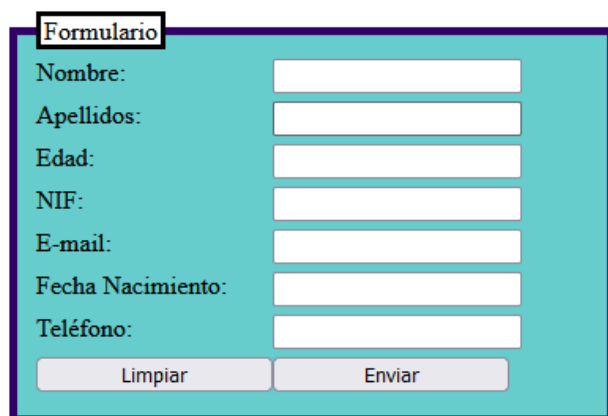
## Validación de un número de DNI

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <title>Ejemplo de RegExp</title>
    <script type="text/javascript">
      const iniciar = () => {
        document.getElementById("comprobar").onclick = comprobarDni;
      }
      const comprobarDni = () => {
        let dni = document.getElementById("dni").value;
        let patron = /^\\d{8}[A-Z]$/;
        if (dni.match(patron))
          alert('DNI Correcto!!!!');
        else
          alert('DNI INCORRECTO!!!!');
      }
      window.onload = iniciar;
    </script>
  </head>
  <body>
    <form name="formulario">
      <label for="dni">DNI:</label>
      <input type="text" name="dni" id="dni" />
      <input type="button" name="comprobar" id="comprobar"
value="Comprobar Formato" />
    </form>
  </body>
</html>
```



## Secuencia / Desarrollo:

### 1. Ejercicio 1: Realiza la validación del formulario, cumpliendo los siguientes requisitos



The image shows a web form titled "Formulario" with a light blue background and a dark blue border. It contains the following fields and labels:

- Nombre: [text input]
- Apellidos: [text input]
- Edad: [text input]
- NIF: [text input]
- E-mail: [text input]
- Fecha Nacimiento: [text input]
- Teléfono: [text input]

At the bottom of the form are two buttons: "Limpiar" (left) and "Enviar" (right).

- Realizar una función que valide los campos de texto **NOMBRE** y **APELLIDOS**. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en los campos correspondientes.
- Validar la **EDAD** que contenga solamente valores numéricos y que esté en el rango de 0 a 105. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo EDAD.
- Validar el **NIF**. Utilizar una expresión regular que permita solamente 8 números un guion y una letra. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo NIF. No es necesario validar que la letra sea correcta. Explicar las partes de la expresión regular mediante comentarios.
- Validar el **E-MAIL**. Utilizar una expresión regular que nos permita comprobar que el e-mail sigue un formato correcto. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo E-MAIL.
- Validar el campo **TELEFONO** utilizando una expresión regular. Debe permitir 9 dígitos obligatorios. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo **TELEFONO**.
- Pedir confirmación de envío del formulario. Si se confirma el envío realizará el envío de los datos; en otro caso cancelará el envío.