

A yellow square containing the letters 'JS' in a large, bold, black sans-serif font.

Desarrollo web en entorno cliente

Formularios

Contenidos



ÍNDICE

Formularios	3
Estructura de un formulario	3
Elementos de un formulario	4
Tipos de input	5
Acceder a elementos de formularios	8
text	8
Radio button	8
checkbox	8
select	9
Validación y envío de formularios	10
Validar un formulario	10
Deshabilitar enviar un formulario dos veces	11
Enviar un formulario desde código	11
Acceso a restricciones de HTML	11

1. FORMULARIOS

En esta unidad trataremos algunas de las operaciones más habituales en los formularios. En primer lugar veremos la estructura genérica de un formulario, para después tratar las operaciones de modificar apariencia, comportamiento, validación, etc

Un formulario web sirve para enviar, tratar y recuperar datos que son enviados y recibidos entre un cliente y un servidor web. Cada elemento del formulario almacena un tipo de dato o acciona una de sus funcionalidades. Los formularios disponen de una arquitectura, en este contexto están enmarcados en el lenguaje HTML.

☞ Al hacer una aplicación Web, el cliente debe validar la información introducida, pero ello no quita al servidor también de validarla.

1.1 Estructura de un formulario

Desde un punto de vista general, los formularios presentan esta estructura:

- Los formularios se definen con etiquetas. Dentro de cada etiqueta también podemos acceder a los atributos de la misma.
- La etiqueta principal es `<form>` `</form>`. Para que sea funcional, la etiqueta `<form>` necesita inicializar dos atributos:
 - **action** → Contiene la URL donde se redirigen los datos del formulario.
 - **method** → Indica el método por el cual el formulario envía los datos. Puede ser POST o GET.
- Además de los atributos principales anteriores, podemos encontrar otros atributos como:
 - **enctype** → define el tipo de codificación para enviar el formulario al servidor. Se usa cuando permite enviar archivos adjuntos.
 - **accept** → indica el tipo de fichero adjunto que acepta el servidor.

```
<body>

  <h3>Formulario</h3>

  <form action="www.Web.es/formulario.html" method="post">

    ...

  </form>

</body>
```

1.2 Elementos de un formulario

El elemento principal del formulario se denomina con la etiqueta **<input>**. Según su funcionalidad, los tipos de input se llaman:

- **Controles de formulario** → *Botones, etc...*
- **Campos de formulario** → *Contienen los datos que se envían a través del formulario.*

En una etiqueta input podemos encontrar los atributos:

Type → Indica el tipo de elemento que vamos a definir. De él dependen el resto de parámetros. Los valores posibles que acepta el atributo type son:

- **text** → *Cuadro de texto.*
- **password** → *Contraseña.*
- **checkbox** → *Casilla de verificación.*
- **radio** → *Opción de entre dos o más.*
- **submit** → *Botón de envío del formulario.*
- **reset** → *Botón de vaciado de campos.*
- **file** → *Botón para buscar ficheros.*
- **Hidden** → *Campo oculto para, el usuario no lo visualiza en el formulario.*
- **Image** → *Botón de imagen.*

Name → El atributo name asigna un nombre al elemento. Si no le asignamos nombre a un elemento, el servidor no podrá identificarlo y por tanto no podrá tener acceso al elemento.

Value → El atributo value inicializa el valor del elemento. Los valores dependerán del tipo de dato, en ocasiones los posibles valores a tomar serán verdadero o falso.

Size → Este atributo asigna el tamaño inicial del elemento. El tamaño se indica en píxeles. En los campos text y password hace referencia al número de caracteres.

Maxlength → Este atributo indica el número máximo de caracteres que pueden contener los elementos text y password. Es conveniente saber que el tamaño de los campos text y password es independiente del número de caracteres que acepta el campo.

Checked. Este atributo es exclusivo de los elementos checkbox y radio. En el definimos que opción por defecto queremos seleccionar.

Disable. Este atributo hace que el elemento aparezca deshabilitado. En este caso el dato no se envía al servidor.

Readonly. Este atributo sirve para bloquear el contenido del control, por tanto el valor del elemento no se podrá modificar.

src. Este atributo es exclusivo para asignar una URL a una imagen que ha sido establecida como botón del formulario.

Alt. El atributo alt, incluye una pequeña descripción del elemento. Habitualmente y si no lo hemos desactivado cuando posicionamos el ratón (sin pulsar ningún botón) encima del elemento, podemos visualizar la descripción del mismo.

1.3 Tipos de input

Cuadro de texto:

Este input muestra un cuadro de texto vacío en el que el usuario puede introducir un texto. Este es uno de los elementos más usados. La forma de indicar que es un campo de texto es:

```
<input type="text"
```



Nombre

Cuadro de contraseña:

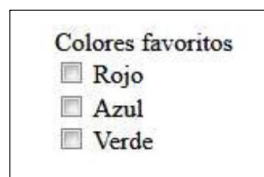
El cuadro de contraseña es como el cuadro de texto, con la diferencia que los caracteres que escribe el usuario no se ven en pantalla. En su lugar los navegadores muestran asteriscos o puntos:

```
<input type="password" name="contrasenia" />
```

Casilla de verificación

Estos elementos permiten al usuario activar o desactivar la selección de cada una de las casillas de forma individual.

```
<p>Colores favoritos</p>
<br><input name="rojo" type="checkbox" value="ro"/> Rojo
<br><input name="azul" type="checkbox" value="az"/> Azul
<br><input name="verde" type="checkbox" value="ve"/> Verde
```



Colores favoritos

☐ Rojo

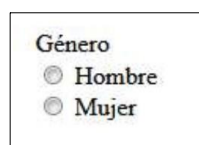
☐ Azul

☐ Verde

Opción de radio

Este tipo de elemento agrupa una serie de opciones excluyentes entre sí. De esta forma el usuario sólo puede coger una opción de entre todas las que tiene establecidas un grupo de botones radio.

```
Género
<br><input type="radio" name="género" value="M"> Hombre
<br><input type="radio" name="género" value="F"> Mujer
```



Género

☐ Hombre

☐ Mujer

Botón de envío

Este elemento es el encargado de enviar los datos del formulario al servidor. En este caso el type toma el valor submit. El valor del atributo value se mostrará en este caso en el botón generado.

```
<input type="submit" name="enviar" value="Enviar">
```



Enviar

Botón de reset:

Este elemento es un botón que establece el formulario a su estado original.



Ficheros adjuntos:

Este tipo de input permite adjuntar ficheros adjuntos. El elemento añade de forma automática un cuadro de texto que se dispondrá para almacenar la dirección del fichero adjunto seleccionado.

```
Fichero adjunto  
<input type="file" name="fichero"/>  
<form action="" method="post" enctype="multipart/form-  
data"> ... </form>
```



Campos ocultos

Los campos ocultos no son visibles en el formulario por el usuario. Estos elementos son útiles para enviar información de forma oculta que no tenga que ser tratada por el usuario.

```
<input type="hidden" name="campoOculto" value="cambiar"/>
```

Botón de imagen:

Este elemento es una personalización de un botón, cambiando el aspecto por defecto que tienen los botones de un formulario por una imagen.

```
<input type="image" name="enviar" src="imagen_mundo.jpg"/>
```



Botón:

Existe un elemento botón, al que podemos asociar diferentes funcionalidades. De esta forma no nos tenemos que ceñir los botones de submit o reset que nos ofrecen los formularios.

```
<input type="button" name="opcion" value="Opcion validar"/>
```

Ejemplo 1

```
<form action="pagina.html"
method="post" enctype="multipart/form-
data" /> <br/> Nombre:
<input type="text" name="nombre" value="" size="42" maxlength="30"/>
Apellidos:
<input type="text" name="ape" value="" size="40" maxlength="80"/>
DNI:
<input type="text" name="dni" value="" size="10" maxlength="9" />
Sexo:
<input type="radio" name="sexo" value="hombre" checked="checked"/>Hombre
<input type="radio" name="sexo" value="mujer" />Mujer
Incluir mi foto:
<input type="file" name="foto" /> <br/>
<input name="publ" type="checkbox" value="publicidad" checked="checked"/>
Enviar publicidad
<input type="submit" name="enviar" value="Guardar cambios" />
<input type="reset" name="limpiar" value="Borrar los datos introducidos"/>
</form>
```

Nombre:

Apellidos:

DNI:

Sexo:

☒ Hombre

☐ Mujer

Incluir mi foto:

☒ Enviar publicidad

2. ACCEDER A ELEMENTOS DE FORMULARIOS

2.1 text

Para acceder al valor de un input de tipo texto, simplemente debemos referenciar el **atributo “value”**.

```
<input type="text" id="miTexto">
```

```
var elemento=document.getElementById("miTexto");  
alert(elemento.value);
```

2.2 Radio button

Radio button son elementos del formulario, que ante varias entradas, te dejan seleccionar una de ellas. Se agrupan teniendo un “name” común.

Para acceder a ellos, se accede como un array, donde se tiene el atributo “value” y el atributo “checked” que es true si está seleccionado, false en caso contrario.

```
<input type="radio" id="preguntaSI" name="pregunta" value="si" />SI  
<input type="radio" id="preguntaNO" name="pregunta" value="no" />NO
```

```
var elementos=document.getElementsByName("pregunta");  
for(i=0;i<elementos.length;i++){  
    if(elementos[i].checked==true)  
        alert("Valor del elemento marcado "+elementos[i].value);  
}
```

2.3 checkbox

Similar a los radio button, salvo que puede haber más de uno marcado.

```
<input type="checkbox" id="preguntaAS" name="pregunta" value="asc"  
>Piso con ascensor  
<input type="radio" id="preguntaAM" name="pregunta" value="amb" />Piso  
amueblado
```

```
var elementos=document.getElementsByName("pregunta");  
for(i=0;i<elementos.length;i++){  
    if(elementos[i].checked==true){  
        alert("Valor del elemento marcado "+elementos[i].value);  
    }  
}
```


2.4 select

Elemento que muestra un desplegable y nos permite elegir una opción del mismo. Aquí destaca el atributo “options”, que es un atributo que contiene un array con las opciones disponibles y el atributo “selectedIndex” que contiene (y se puede modificar) la posición del array “options” seleccionada actualmente (o la primera si se permite multiselección) o -1 si no está seleccionada ninguna opción (o queremos des-seleccionarlas). Dentro de cada “options”, “value” almacena el valor y “text” el texto mostrado.

```
<select id="aprobar" >
  <option value="10">Saco 10 en DWEC</option>
  <option value="9">Saco 9 en DWEC</option>
  <option value="8">Saco 8 en DWEC</option>
</select>
```

```
var elemento=document.getElementById("aprobar");
for(i=0;i<elemento.options.length;i++){
    alert("Valor de la opcion "+elemento.options[i].value);
}

var sel=elemento.selectedIndex;
alert("El valor de la opcion seleccionada es
"+elemento.options[sel].value+" y el texto asociado es
"+elemento.options[sel].text);

// Cambiamos el indice seleccionado
elemento.selectedIndex=0;
```

Ejemplo 2

```
<form method="get" action="index51.html" id="formulario1">
  Ingrese clave:
  <input type="password" id="clave1" name="clave1" size="20" required>
  <br> Repita clave:
  <input type="password" id="clave2" name="clave2" size="20" required>
  <br>
  <input type="submit" id="confirmar" name="confirmar" value="Confirmar">
</form>

<script>
  document.getElementById("formulario1").addEventListener('submit', validar);

  function validar(evt) {
    let cla1 = document.getElementById("clave1").value;
    let cla2 = document.getElementById("clave2").value;
    if (cla1 != cla2) {
      alert('Las claves ingresadas son distintas');
      evt.preventDefault();
    }
  }
</script>
```

Ingrese clave:

Repita clave:

Confirmar

Esta página dice

Las claves ingresadas son distintas

Aceptar

3. VALIDACIÓN Y ENVÍO DE FORMULARIOS

El usuario puede cometer errores al rellenar un formulario. Si por ejemplo se espera un código postal y se introduce el nombre de una ciudad, se producirá un error. Para controlar estas situaciones se pueden usar las validaciones. Este tipo de validaciones se suelen realizar llamando a una función que analice si el dato cumple con las restricciones establecidas.

3.1 Validar un formulario

Si un manejador de un evento devuelve true (o no devuelve nada), se realiza el evento asociado. Si el manejador devuelve false, se cancela el evento. Existe un evento asociado a un formulario completo llamado "onsubmit".

Aprovechando esto, podemos a nuestro antojo permitir el envío de información al servidor mediante el formulario devolviendo true o cancelarlo devolviendo false.

La estructura típica es la siguiente:

```
<form onsubmit="return validarFecha();">
```

Si la función validar devuelve true, se realiza el envío. Si devuelve false, se cancela.

En la función validar podemos hacer las validaciones que estimemos convenientes. Es decir, para crear funciones de validación, se debe devolver un booleano:

- **true** → *Sí se lanza el evento "submit"*
- **false** → *No se lanza el evento "submit"*

```
script type="text/javascript">

    function validaFecha() {

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

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

        if( !isNaN(fecha) ) {

            return false;

        }

        return true;

    }

}
```

```
<form action="" onsubmit="return validacion()" method="" id="" name="" >
    ...
</form>
<script>
    function validacion(){
        valor=document.getElementById("label1").value;
        if (!isNaN(valor)){
            console.log(valor);
            return true;
        }
        return false;
    }
</script>
```

3.2 Deshabilitar enviar un formulario dos veces

A veces un usuario pulsa enviar un formulario mas de una vez por error. Si queremos evitar esto, podemos usar la propiedad “**disabled**”;

```
document.getElementById("idSubmit").disabled = true;
```

3.3 Enviar un formulario desde código

En algunas aplicaciones por motivos estéticos o de funcionalidad es deseable que el “enviar un formulario” no se haga desde un botón “submit”, sino desde cualquier otro evento que permita la ejecución de código. Esto se puede hacer recogiendo el elemento del formulario y aplicándole el método submit();
Un ejemplo de esto en Javascript.

```
var elemento=document.getElementById("formulario");  
elemento.submit();
```

4. ACCESO A RESTRICCIONES DE HTML

Mediante Javascript tenemos acceso a todos los campos del formulario por lo que podemos hacer la validación como queramos, pero es una tarea pesada, repetitiva y que provoca código spaghetti difícil de leer y mantener más adelante.

Para hacerla más simple podemos usar la [API de validación de formularios](#) de HTML5:

Esta API nos proporciona estas propiedades y métodos:

- **checkValidity()** → método que nos dice si el campo al que se aplica es o no válido. También se puede aplicar al formulario para saber si es válido o no
- **validationMessage** → en caso de que un campo no sea válido esta propiedad contiene el texto del error de validación proporcionado por el navegador
- **validity** → es un objeto que tiene propiedades booleanas para saber qué requisito del campo es el que falla:
 - **valueMissing**: indica si no se cumple la restricción *required* (es decir, valdrá *true* si el campo tiene el atributo *required* pero no se ha introducido nada en él)
 - **typeMismatch**: indica si el contenido del campo no cumple con su atributo *type* (ej. *type*="email")
 - **patternMismatch**: indica si no se cumple con el *pattern* indicado
 - **tooShort / tooLong**: indica si no se cumple el atributo *minlength/maxlength*
 - **rangeUnderflow / rangeOverflow**: indica si no se cumple el atributo *min / max*
 - **stepMismatch**: indica si no se cumple el atributo *step* del campo
 - **customError**: indica al campo se le ha puesto un error personalizado con *setCustomValidity*
 - **valid**: indica si es campo es válido
 - ...
- **setCustomValidity(mensaje)** → añade un error personalizado al campo (que ahora ya NO será válido) con el mensaje pasado como parámetro. Nos permite personalizar el mensaje del navegador. Para quitar este error se hace *setCustomValidity("")*

5. EXPRESIONES REGULARES

La potencia de las expresiones regulares es que podemos usar patrones para construir la expresión. Para definir una expresión regular, tenemos dos caminos:

Constructores	
<code>new RegExp(expr, flags)</code>	<i>Crea una nueva expresión regular a partir de r con los flags indicados.</i>
<code>/ expr / flags</code>	<i>Simplemente, la expresión regular r entre barras /. Notación preferida.</i>

		// DEFINICIÓN COMO LITERAL
<code>let r = /a/g;</code>	<code>let r = /expr/flag</code>	
		// DEFINICIÓN COMO OBJETO
<code>let r = new RegExp('a', 'g')</code>	<code>let r = new RegExp(expr, flag)</code>	
<hr/>		
<code>r.test('cadena_a_comprobar')</code>		
<hr/>		
<code>var reg_exp = /\d{8}[a-zA-Z]\$/; // 8 cifras numéricas + carácter alfabético.</code>		
<code>if(reg_exp.test(formulario.dni.value)===false) { // indicar el error }</code>		

Patrón	Descripción	
flags		
i	.ignoreCase	Ignorar mayúsculas y minúsculas
g	.global	Búsqueda global, no se para al encontrar primera ocurrencia
m	.multiline	Multilínea, permite a ^ \$ tratar los finales de línea \n
u	.unicode	La codificación es unicode
y	.sticky	Busca solo desde la posición indicada
Métodos		
test (str)	Comprueba si la expresión regular «casa» con el texto str pasado por parámetro. Boolean	
Reglas		
^	Principio:	/^ag/→ comienza por ag
\$	Final:	/\$in/→ termina en in
[]	Conjunto:	/[a-z]/ /[aeiou]/
[^]	Conjunto negado: No exista cualquiera de los caracteres entre corchetes	
{x}	Cuantificadores: /d{4,7}/→ de 4 a 7 dígitos /d{4}/→ 4 dígitos /d{4,}/→ mas de 4 dígitos	
	Alternancia:	/a b/→ El valor a ó b
.	Cualquier carácter menos el salto de línea	
?	Cero o ninguna ocurrencia : {0,1}	
*	Cero o más ocurrencias: {0,}	
+	Una o mas ocurrencias: {1,}	
\d	Dígito:	[0-9]
\w	Caracter:	[a-zA-Z0-9]
\s	Espacio en blanco	

```
const r = /.a.o/i;
```

```
r.test("gato");
```

```
// true
```

```
r.test("pato");
```

```
// true
```

```
r.test("perro");
```

```
// false
```

```
r.test("DATO");
```

```
// true (el flag i permite mayús/minús)
```

Ejemplos

```
<!DOCTYPE html>
<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=2.0">
  <title>TMP</title>
</head>
<body>
Introduce tus hobbies preferidos<br>
<form id="miFormulario" action="" method="post" onsubmit="return valida(this)">
<p>
  Hobbie 1: <input type="text" id="marca1" name="hobbie"/>
<br />
  Hobbie 2: <input type="text" name="hobbie2" />
  <!-- Hobbie 2: <input type="text" id="marca2" />
<br />
  Hobbie 3: <input type="text" name="hobbie3" />
<br />
<input type="submit" value="Comprobar" />
<input type="reset" value="Limpiar datos" />
</p>
</form>
<script>
function valida(f) {
  var ok = true;
  var msg = "Debes escribir contenido en los campos:\n";
  let cla1 = document.getElementById("marca1").value;
  if (cla1 == "")
  {
    msg += "- Hobbie 1\n";
    ok = false;
  }
  if(f.elements[1].value == "")
  {
    msg += "- Hobbie 2\n";
    ok = false;
  }
  if(f.hobbie3.value == "")
  {
    msg += "- Hobbie 3\n";
    ok = false;
  }
  if(ok == false)
    alert(msg);
  return ok;
}
</script>
</body>
</html>
```

Introduce tus hobbies preferidos

Hobbie 1:

Hobbie 2:

Hobbie 3:

Comprobar

Limpiar datos

Esta página dice

Debes escribir contenido en los campos:

- Hobbie 1

- Hobbie 2

- Hobbie 3

Aceptar


```

<script>
//Comproba si una palabra termina con una vocal.
    let palabra = prompt('Ingresa una palabra');
    let patron = /^[aeiouAEIOUáéíóú]$/;
    if (patron.test(palabra))
        alert('La palabra no finaliza con vocal');
    else
        alert('La palabra finaliza con vocal');
//Ingresar una matricula de un auto de italia y luego mostrar un mensaje si es correcto o no su valor.
// Toda patente está constituida por 2 letras, cuatro números y finalmente 2 letras (ej. ro1234oy )
    let patente = prompt('Ingresa una matricula');
    let patron = /[a-z]{2}[0-9]{4}[a-z]{2}/;
    if (patron.test(patente))
        alert('La matricula ingresada es correcta');
    else
        alert('La matricula ingresada no es correcta');
//si ingresamos: aa11cccd SE VERIFICA VERDADERO.
    let patente = prompt('Ingresa una patente');
    let patron = /^[a-z]{2}[0-9]{4}[a-z]{2}$/;
    if (patron.test(patente))
        alert('La matricula ingresada es correcta');
    else
        alert('La patente matricula ingresada no es correcta');
//Validar que se ingrese un número de 3 dígitos enteros, el carácter punto y 2 dígitos decimales.
    let valor = prompt('Ingresa un nro con el formato (999.99):');
    let patron = /^[0-9]{3}\.[0-9]{2}$/;
    if (patron.test(valor))
        document.write('El valor es correcto');
    else
        document.write('El valor no tiene el formato correcto');
//Verificar si un número tecleado tiene exactamente 9 dígitos.
    let valor = prompt('Ingresa un valor numérico de 9 dígitos');
    let patron = /^d{9}$/;
    if (patron.test(valor))
        document.write('El valor tiene 9 dígitos');
    else
        document.write('El valor no tiene 9 dígitos');
//Validar si un string contiene 4 números de 3 dígitos cada uno separados por coma.
    let cadena = prompt('Ingresa 2 números de 3 dígitos separados por coma');
    let patron = /^d{3}\,d{3}$/;
    if (patron.test(cadena))
        document.write('Numero correcto');
    else
        document.write('Numero incorrecto');
//Validar si una palabra comienza con los caracteres va o ba
    let palabra = prompt('Ingresa una palabra que comience con no o si:');
    let patron = /^[no|si]/;
    if (patron.test(palabra))
        document.write('La palabra comienza con no o si');
    else
        document.write('La palabra no comienza con no o si');
</script>

```

ANEXO

En la siguiente tabla resumimos diferencias entre el uso de get y post:

Aspecto	Con GET	Con POST
Los datos son visibles en la url	Sí	No
Los datos pueden permanecer en el historial del navegador	Sí	No
Una url puede ser guardada conteniendo parámetros de un envío de datos	Sí	No
Existen restricciones en la longitud de los datos enviados	Sí (no se puede superar la longitud máxima de una url)	No
Se considera preferible para envío de datos sensibles (datos privados como contraseñas, números de tarjeta bancaria, etc.)	No (los datos además de ser visibles pueden quedar almacenados en logs)	Sí (sin que esto signifique que por usar post haya seguridad asegurada)
Codificación en formularios	application/x-www-form-urlencoded	application/x-www-form-urlencoded ó multipart/form-data. Se usa multipart para envío de datos binarios, por ejemplo ficheros.
Restricciones de tipos de datos	Sí (sólo admite caracteres ASCII)	No (admite tanto texto como datos binarios p.ej. archivos)
Se considera preferible para disparar acciones	No (podría ser accedido por un robot que dispararía la acción)	Sí (sin que esto garantice que no pueda acceder un robot)
Riesgo de cacheado de datos recuperados en los navegadores	Sí	No
Posibles ataques e intentos de hackeo	Sí (con más facilidad)	Sí (con menos facilidad)

Por motivos de seguridad se recomienda usar GET a los efectos de únicamente **recuperar información** desde el servidor. Por el contrario, si se desea realizar operaciones como actualización, borrado, etc. no se recomienda usar GET.