DESARROLLO DE APLICACIONES WEB

16-1-2024

ROBERTO RODRÍGUEZ JIMÉNEZ

roberto.rodjim.1@educa.jcyl.es

Desarrollo Web en Entorno Cliente

Gestión de eventos y formularios en JavaScript

Tarea 05

Contenido

[Tarea online DWEC05 3](#_Toc158371108)

[1.- Descripción de la tarea 3](#_Toc158371109)

[¿Qué te pedimos que hagas? 3](#_Toc158371110)

[2.- Información de interés 4](#_Toc158371111)

[Recursos necesarios 4](#_Toc158371112)

[Consejos y recomendaciones 6](#_Toc158371113)

[3.- Evaluación de la tarea 6](#_Toc158371114)

[Criterios de evaluación implicados 6](#_Toc158371115)

[¿Cómo valoramos y puntuamos tu tarea? 6](#_Toc158371116)

[RESOLUCIÓN 7](#_Toc158371117)

[1. Programar el código de JavaScript en un fichero independiente. La única modificación que se permite realizar en el fichero .html es la de escribir la referencia al fichero de JavaScript. 7](#_Toc158371118)

[2. Almacenar en una cookie el número de intentos de envío del formulario que se van produciendo y mostrar un mensaje en el contenedor "intentos" similar a: "Intento de Envíos del formulario: X". Es decir, cada vez que le demos al botón de enviar tendrá que incrementar el valor de la cookie en 1 y mostrar su contenido en el div antes mencionado. Nota: para poder actualizar el contenido de un contenedor o div la propiedad que tenemos que modificar para ese objeto es innerHTML. 7](#_Toc158371119)

[3. Cada vez que los campos NOMBRE y APELLIDOS pierdan el foco, el contenido que se haya escrito en esos campos se convertirá a mayúsculas. 7](#_Toc158371120)

[4. 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. 7](#_Toc158371121)

[5. 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. 7](#_Toc158371122)

[6. Validar el NIF. Utilizar una expresión regular que permita solamente 8 números un guión 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 correct. Explicar las partes de la expresión regular mediante comentarios. 8](#_Toc158371123)

[7. 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. Explicar las partes de la expesión regular mediante comentarios. 8](#_Toc158371124)

[8. Validar que se haya seleccionado alguna de las PROVINCIAS. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo PROVINCIA. 8](#_Toc158371125)

[9. Validar el campo FECHA utilizando una expresión regular. Debe cumplir alguno de los siguientes formatos: dd/mm/aaaa o dd-mm-aaaa. No se pide validar que sea una fecha de calendario correcta. Si se produce algún error mostrar el mensaje en el contenedor "rrores" y poner el foco en el campo FECHA. Explicar las partes de la expresión regular mediante comentarios. 8](#_Toc158371126)

[10. 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. Explicar las partes de la expresión regular meiante comentarios. 8](#_Toc158371127)

[11. Validar el campo HORA utilizando una expresión regular. Debe seguir el patrón de hh:mm. No es necesario validar que sea una hora correcta. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo HORA. Explicar as partes de la expresión regular mediante comentarios. 8](#_Toc158371128)

[12. Pedir confirmación de envío del formulario. Si se confirma el envío realizará el envío de los datos; en otro caso cancelará el envío. 8](#_Toc158371129)

[Desarrollo 9](#_Toc158371130)

[Recursos 10](#_Toc158371131)

# Tarea online DWEC05

## 1.- Descripción de la tarea

**Caso práctico**

Antonio, afronta una de las partes más interesantes en su estudio de JavaScript. Se trata de la gestión de los eventos en JavaScript y cómo validar formularios.

Antonio estaba deseando llegar a esta sección, ya que su trabajo de desarrollo en el proyecto se va a centrar en muchas de las cosas que va a ver ahora. Va a tener que validar todos los formularios de la web antigua y tendrá que hacerlo con JavaScript, dando mensajes al usuario, sobre los posibles errores que se vaya encontrando durante la validación. La validación de un formulario es primordial, ya que lo que se busca es que cuando los datos sean enviados al servidor, vayan lo más coherentes posible.

Gracias a la gestión de eventos, Antonio podrá, por ejemplo, capturar acciones del usuario cuando pulse un botón o cuando pase el ratón por zonas del documento, al introducir datos, etc.

Juan está también muy ilusionado con los progresos de Antonio durante todo este tiempo, y le facilita toda la documentación y algunos ejemplos interesantes de validaciones de datos con JavaScript.

### ¿Qué te pedimos que hagas?

Realizar la validación del formulario facilitado en el enunciado, cumpliendo los siguientes requisitos:

1. Programar el código de JavaScript en un fichero independiente. La única modificación que se permite realizar en el fichero .html es la de escribir la referencia al fichero de JavaScript.
2. Almacenar en una cookie el número de intentos de envío del formulario que se van produciendo y mostrar un mensaje en el contenedor "intentos" similar a: "Intento de Envíos del formulario: X". Es decir cada vez que le demos al botón de enviar tendrá que incrementar el valor de la cookie en 1 y mostrar su contenido en el div antes mencionado. Nota: para poder actualizar el contenido de un contenedor o div la propiedad que tenemos que modificar para ese objeto es innerHTML.
3. Cada vez que los campos NOMBRE y APELLIDOS pierdan el foco, el contenido que se haya escrito en esos campos se convertirá a mayúsculas.
4. 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.
5. 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.
6. Validar el NIF. Utilizar una expresión regular que permita solamente 8 números un guión 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.
7. 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. Explicar las partes de la expresión regular mediante comentarios.
8. Validar que se haya seleccionado alguna de las PROVINCIAS. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo PROVINCIA.
9. Validar el campo FECHA utilizando una expresión regular. Debe cumplir alguno de los siguientes formatos: dd/mm/aaaa o dd-mm-aaaa. No se pide validar que sea una fecha de calendario correcta. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo FECHA. Explicar las partes de la expresión regular mediante comentarios.
10. 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. Explicar las partes de la expresión regular mediante comentarios.
11. Validar el campo HORA utilizando una expresión regular. Debe seguir el patrón de hh:mm. No es necesario validar que sea una hora correcta. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo HORA. Explicar las partes de la expresión regular mediante comentarios.
12. Pedir confirmación de envío del formulario. Si se confirma el envío realizará el envío de los datos; en otro caso cancelará el envío.

## 2.- Información de interés

### Recursos necesarios

Editor web para teclear el código de la aplicación y un navegador web para ejecutar y probar la aplicación.

Fichero .html con el formulario que hay que validar.

<!DOCTYPE html>

<htm xmlns="http://www.w3.org/1999/xhtml">

    <head>

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

        <title>DWEC05 - Solución Tarea</title>

        <script type="text/javascript" src="scriptSolucion.js"></script>

        <style type="text/css">

            label{

                width: 150px;

                float:left;

                margin-bottom:5px;

            }

            input,select {

                width:150px;

                float:left;

                margin-bottom:5px;

            }

            fieldset{

                background:#66CCCC;

                width:350px;

                border: thick solid #306;

            }

            legend{

                border-top-width: medium;

                border-right-width: medium;

                border-bottom-width: medium;

                border-left-width: medium;

                border-top-style: solid;

                border-right-style: solid;

                border-bottom-style: solid;

                border-left-style: solid;

                background-color: #FFF;

            }

            #mensajes{

                float: left;

                background:#33FF33;

                width: 325px;

            }

            #errores{

                float: left;

                background:#FF6633;

                width: 325px;

            }

            #intentos{

                float: left;

                background:#CCFF33;

                width: 325px;

            }

            .error{

                border: solid 2px #FF0000;

            }

        </style>

    </head>

    <body>

        <fieldset>

            <legend>DWEC05 - Solución Tarea</legend>

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

                <label for="nombre">Nombre:</label>

                <input type="text" name="nombre" id="nombre" />

                <label for="apellidos">Apellidos:</label>

                <input type="text" name="apellidos" id="apellidos" />

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

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

                <label for="nif">NIF:</label>

                <input name="nif" type="text" id="nif" />

                <label for="email">E-mail:</label>

                <input name="email" type="text" id="email" />

                <label for="provincia">Provincia:</label>

                <select name="provincia" id="provincia">

                    <option value="0" selected="selected">Seleccione Provincia</option>

                    <option value="C">A Coruña</option>

                    <option value="LU">Lugo</option>

                    <option value="OU">Ourense</option>

                    <option value="OU">Pontevedra</option>

                </select>

                <label for="fecha">Fecha Nacimiento:</label>

                <input name="fecha" type="text" id="fecha" />

                <label for="telefono">Teléfono:</label>

                <input name="telefono" type="text" id="telefono" maxlength="9"/>

                <label for="hora">Hora de visita:</label>

                <input name="hora" type="text" id="hora" />

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

                <input type="submit" name="enviar" id="enviar" value="Enviar" />

            </form>

            <div id="errores"></div>

            <div id="intentos"></div>

        </fieldset>

    </body>

</html>

### Consejos y recomendaciones

Se recomienda realizar una función para cada una de las validaciones de tal forma que se pueda llamar a cada una de forma independiente. Las funciones deberían devolver true si la validación ha sido correcta o false (y los mensajes de error solicitados) si la validación ha sido incorrecta.

## 3.- Evaluación de la tarea

### Criterios de evaluación implicados

a) Se han reconocido las posibilidades del lenguaje de marcas relativas a la captura de los eventos producidos.  
 b) Se han identificado las características del lenguaje de programación relativas a la gestión de los eventos.  
 c) Se han diferenciado los tipos de eventos que se pueden manejar.  
 d) Se ha creado un código que capture y utilice eventos.  
 e) Se han reconocido las capacidades del lenguaje relativas a la gestión de formularios Web.  
 f) Se han validado formularios web utilizando eventos.  
 g) Se han utilizado expresiones regulares para facilitar los procedimientos de validación.  
 h) Se ha probado y documentado el código.

### ¿Cómo valoramos y puntuamos tu tarea?

|  |  |
| --- | --- |
| **Rúbrica de la tarea** | |
| Apartado 1 | 0,5 puntos |
| Apartado 2 | 1 punto |
| Apartado 3 | 0,75 puntos |
| Apartado 4 | 0,5 puntos |
| Apartado 5 | 0,75 puntos |
| Apartado 6 | 1 punto |
| Apartado 7 | 1,5 puntos |
| Apartado 8 | 0,75 puntos |
| Apartado 9 | 0,75 puntos |
| Apartado 10 | 0,75 puntos |
| Apartado 11 | 0,5 puntos |
| Apartado 12 | 0,75 puntos |
| Claridad y presentación del código, indentación, etc. | 0,5 puntos |

## 

## RESOLUCIÓN

### Programar el código de JavaScript en un fichero independiente. La única modificación que se permite realizar en el fichero .html es la de escribir la referencia al fichero de JavaScript.

Captura de pantalla con letras blancas

Descripción generada automáticamente

Estructura de archivos

Se mantiene la referencia al fichero JavaScript, pero se coloca en un directorio *js*.  
Se modifica, por tanto, el enlace en la llamada al fichero desde *head*.

### Almacenar en una cookie el número de intentos de envío del formulario que se van produciendo y mostrar un mensaje en el contenedor "intentos" similar a: "Intento de Envíos del formulario: X". Es decir, cada vez que le demos al botón de enviar tendrá que incrementar el valor de la cookie en 1 y mostrar su contenido en el div antes mencionado. Nota: para poder actualizar el contenido de un contenedor o div la propiedad que tenemos que modificar para ese objeto es innerHTML.

Para actualizar el valor de la cookie ***intentos*** tenemos que hacerlo en dos pasos: comprobar si existe la cookie para obtener su valor y crearla o actualizarla.

Cargar el botón enviar y agregar el oyente, que ejecutará la función procesarFormulario

window.onload = () => {

    // Declaración de los componentes

    const BTN\_ENVIAR = document.getElementById('enviar');

    // Eventos

    BTN\_ENVIAR.addEventListener('click', procesarFormulario);

}

procesarFormulario establece la cookie y la modifica con setIntento(), posteriormente, escribe el mensaje en la página llamando a la función escribirIntentos().  
Evita que siga ejecutándose el script con preventDefault(), de manera que el formulario no se envía.

// Procesa el formulario

const procesarFormulario = () => {

    // Establece y actualiza la cookie de intentos de envío del formualario

    setIntento();

    // Escribe los intentos en la página

    escribirIntentos();

    event.preventDefault();

};

#### Función setIntento()

Esta función establece y modifica el valor de la cookie ***intentos****.*Para obtener el valor de la cookie, llama a la función getIntentos(), encargada de devolver el valor actual de la cookie.  
Después de obtener el valor, lo incrementa (puede ser 0), establece la duración en 1 día y el path en el directorio raíz.

Tanto el nombre como el valor son codificados para escapar los caracteres especiales con encodeURIComponent.

function setIntento(){

    // obtener los intentos e incrementarlos

    let intentos = getIntentos();

    intentos++;

    // Tiempo de vida de la cookie: 1 día

    let max\_time = 24\*60\*60;

    // Establecemos la cookie, útil durante 5 min.

    document.cookie = `${encodeURIComponent('intentos')}=${encodeURIComponent(intentos)};path=/;max-age=${max\_time};`;

}

#### Función getIntentos()

Función que devuelve el valor de cookie ***intentos***.  
Las cookies dependen de document.cookie (por lo que no pueden ser leídas desde otra ventana o pestaña) y se almacenan como una cadena de texto con el formato nombre1=valor1;nombre2=valor2;…;nombren=valorn;

Para encontrar la cookie ***intentos*** se declara el patrón /intentos=\d+/ en el que se busca la cadena *intentos=* segida de, al menos, un número.

El primer paso es buscar la cadena en el String que guarda las cookies y, si no aparece, devolvemos 0 sin hacer nada más.

function getIntentos(){

    // Patrón buscado

    let pattern = /intentos=\d+/;

    // Obtener las cookies decodificadas

    let cookies = document.cookie;

    // buscar el patrón entre la cadena de cookies

    if(cookies.match(pattern)){

. . .

    }else{

        return 0;

    }

}

En caso de que la búsqueda sea positiva, hay que procesar la cadena de texto con las cookies para obtener, primero, la cookie buscada y, posteriormente, su valor.

function getIntentos(){

    . . .

    // buscar el patrón entre la cadena de cookies

    if(cookies.match(pattern)){

        // tenemos las cookies con el formato nombre=valor;nombre=valor...

        let array\_cookies = cookies.split(';');

        // Hay que conseguir el registro intentos=\d y para ello usamos la función filter

        // que devuelve un array con las coincidencias

        const result = array\_cookies.filter((array\_cookies) => array\_cookies.match(pattern));

        // Obtenemos un array con las coincidencias, que solo debe ser una

        let intentos = decodeURIComponent(result[0].substring(result[0].indexOf('=') +1));

        return intentos;

    }else{

        return 0;

    }

}

#### Función escribirIntentos();

Es la función encargada de escribir el valor de la cookie ***intentos*** *en la página.*

function escribirIntentos(){

    document.getElementById('intentos').innerHTML = 'Intento de envíos del formualario: ' + getIntentos();

}

### Cada vez que los campos NOMBRE y APELLIDOS pierdan el foco, el contenido que se haya escrito en esos campos se convertirá a mayúsculas.

Para comprobar la pérdida del foco, habrá que estar a la escucha del evento onblur en los elementos.

    INP\_NOMBRE.addEventListener('blur', (e) => {

        toUpper(e);

    });

    INP\_APELLIDOS.addEventListener('blur', (e) => {

        toUpper(e);

    });

Se llama a la función toUpper() que cambia el valor de los inputs el mismo en mayúscuclas.

const toUpper = (e) => {

    e.target.value = e.target.value.toUpperCase();

}

**Nota:** el temario habla de pasar la referencia this como parámetro, pero la referencia que se pasa es la de la ventana.

### 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.

Interfaz de usuario gráfica

Descripción generada automáticamenteSe crea la función validarNombreApellido que recibe como parámetro el envento disparado.

const validarNombreApellido = (e) => {

. . .

}

En esta clase se especifica el patrón que va a evaluar el texto de los input.  
Son válidos nombres simples y nombres compuesto.  
Se permite un conjunto de preposición + artículo o artículo entre los dos nombres o apellidos.

Patrón completo

let pattern = /^([a-záéíóú]{3,12}){1}(( (de|la|del|de la|de los|y){1}){0,2} [a-záéíóú]{1,12})?$/gi;

**Primer grupo:**

^([a-záéíóú]{3,12}){1}

Se inicia con la cadena, aceptando letras entre a y z, además de las vocales con tilde.   
El *flag i* permite que no se distingan mayúsculas de minúsculas.

Cada una de estas letras debe aparecer entre 3 y 12 veces.  
El conjunto de las letras debe aparecer una única vez.

**Segundo grupo:**

(( (de|la|del|de la|de los|y){1}){0,1} [a-záéíóú]{1,12})?

El segundo grupo puede aparecer 0 o 1 vez (“?”).  
Está formado por un grupo de letras iguales al primer grupo, pero permite una única letra.

Opcionalmente, puede aparecer delante un conjunto de preposiciones o artículos para complementar el nombre.

**Final:**

$

Después del segundo grupo no puede haber nada más.

Antes de validar el texto, se limpian los espacios blancos que pueda haber delante y detrás:

let texto = e.target.value.trim();

**Captura de la coincidencia**

Si la búsqueda del patrón en la cadena de texto tiene éxito, se realizan las operaciones pendientes:

    if(!texto.match(pattern)){

        e.target.style.color = '#f00';

        let message = `Se ha producido un error en el campo ${e.target.id}`;

        let container = document.getElementById('errores');

        let p = document.createElement('p');

        let text = document.createTextNode(message);

        p.appendChild(text);

        container.appendChild(p);

    }

### 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.

La función validarEdad(e) recibe como argumento el evento y de él obtiene los datos.  
La función se limita a comprobar la validez de número y que este se mantenga en un rango.

En lugar de usar la función parseInt() se usa isFinite().  
El motivo es que parseInt() devuelve el valor entero de la cadena pasada como argumento hasta que encuentra un carácter no válido.

Number.isInteger() tampoco funciona porque el valor evaluado es de tipo texto, con lo que siempre dará error.

|  |  |  |  |
| --- | --- | --- | --- |
| **Valor** | **parseInt()** | **isFinite()** | **Number.isInteger()** |
| “35” | 35 | True | False |
| “35a” | 35 | False | False |
| “a35” | NaN | False | false |

El código es muy simple

const validarEdad = (e) => {

    let edad = e.target.value;

    let error = null;

    let message = [

        `${edad} no es un valor numérico`,

        `${edad} está fuera del rango entre 0 y 105 años`

    ];

    if(isFinite(edad)){

        if(edad < 0 || edad > 105){

            error = 1;

        }

    }else{

        error = 0;

    }

    if(error != null){

        e.target.style.color = '#f00';

        let container = document.getElementById('errores');

        let p = document.createElement('p');

        let text = document.createTextNode(message[error]);

        p.appendChild(text);

        container.appendChild(p);

    }

}

### Validar el NIF. Utilizar una expresión regular que permita solamente 8 números un guión 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.

Limpiamos el texto recibido del input y creamos un patrón para validar el NIF

// Limpiamos los caracteres blancos que pudiera haber al principo y al final

    let nif = e.target.value.trim();

    // Patrón buscado

    // Grupo de números: \d{8} indica que debe contener exactamente 8 dígitos

    // Grupo de letras: [a-z]{1} debe haber exactamente una letra, entre a y z.

    // Antes de los números no puede haber nada: ^

    // Después de la letra no puede haber nada: $

    // Entre los números y las letras debe haber un guión, sin espacios: -

    // El flag i indica que se no se distinguen mayúsculas y minúsculas.

    let pattern = /^\d{8}-[a-z]{1}$/gi;

Una vez evaluado, mostramos el error si lo hubiera.

if(!nif.match(pattern)){

        e.target.style.color = '#f00';

        let message = `El NIF ${nif} tiene un formato incorrecto`;

        let container = document.getElementById('errores');

        let p = document.createElement('p');

        let text = document.createTextNode(message);

        p.appendChild(text);

        container.appendChild(p);

    }

### 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. Explicar las partes de la expresión regular mediante comentarios.

    // Limpiamos los caracteres blancos que pudiera haber al principo y al final

    let email = e.target.value.trim();

    // Se divide el email en cuatro grupos: usuario, @ dominio y extensión de dominio.

    // Grupo 1: usuario

    //      Se compone de letras, mayúsculas o minúsculas, guión alto y bajo y punto.

    //      Debe aparecer, como mínimo, una vez.

    // Grupo 2: @

    //      Es obligatorio que aparezca

    // Grupo 3: Dominio

    //      Grupo de letras, mayúsculas y minúsculas, números y guiones alto y bajo.

    //      Solo aparece una vez.

    // Grupo 4: Niveles de dominio.

    //      Compuesto por un punto seguido por entre 2 y 4 letras munúsculas o mayúsculas.

    //      Este grupo puede aparecer 1 o 2 veces.

    // Antes y después de la cadena, no puede haber nada: ^ ... $

    // Patrón buscado

    let pattern = /^[a-zA-Z0-9-\_.][+@[a-zA-Z0-9-\_]+(\.[a-zA-Z]{2,4}){1,2}$/](mailto:+@[a-zA-Z0-9-_]+(\.%5ba-zA-Z%5d%7b2,4%7d)%7b1,2%7d$/);

### Validar que se haya seleccionado alguna de las PROVINCIAS. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo PROVINCIA.

Solamente se comprueba que el valor recibido sea diferente de 0.

    if(e.target.value == "0"){

        imprimirError("Debes seleccionar una provincia", e.target);

    }

### Validar el campo FECHA utilizando una expresión regular. Debe cumplir alguno de los siguientes formatos: dd/mm/aaaa o dd-mm-aaaa. No se pide validar que sea una fecha de calendario correcta. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo FECHA. Explicar las partes de la expresión regular mediante comentarios.

// Hay que tener en cuenta que se exige un 0 a la izquierda para cumplir la especificación.

// Grupo día:  ( 0[1-9] | [1-2]\d | 3[0-1] )

//      Días del 1 al 9: debe comenzar con un 0.

//      Días del 10 al 29: comienzan con 1 o 2 y siguen con otro número.

//      Días 30 y 31: comienzan con 3 y uno entre 0 y 1.

// Grupo mes:  ( 0[0-9] | 1[0-2] )

//      Meses del 1 al 9: 0 seguido de un número del 1 al 9.

//      Opciones 2, 3 y 4: los meses 10, 11 y 12.

// Grupo año:  ( [0-1][0-9]{3} | 20[0-9][0-5] )

//      Opción 1: admite desde el año 0 hasta 1999

//      Opción 2: admite desde el año 2000 hasta 2025

// El separado admite "/" o "-" y para elegirlo se incluye el mes en la opción.

let pattern = /^(0[1-9]|[1-2]\d|3[0-1])(\/(0[1-9]|1[0-2])\/|-(0[1-9]|1[0-2])-)([0-1]\d{3}|20\d[0-5])$/;

### 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. Explicar las partes de la expresión regular mediante comentarios.

El patrón es muy sencillo.

 // Se exigen 9 dígitos: \d{9}

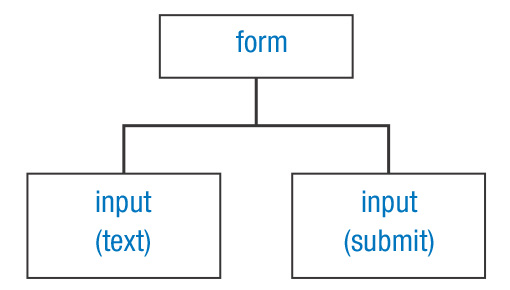
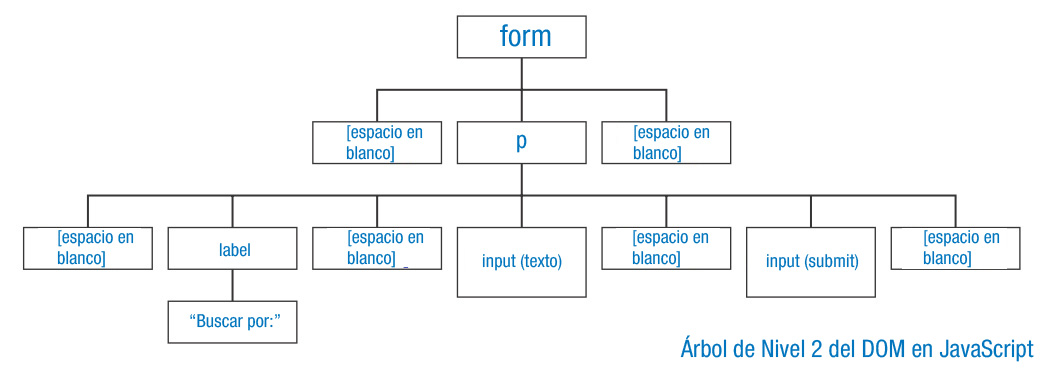
 // No puede haber nada delante (^) ni detrás($).

 let pattern = /^\d{9}$/;

### Validar el campo HORA utilizando una expresión regular. Debe seguir el patrón de hh:mm. No es necesario validar que sea una hora correcta. Si se produce algún error mostrar el mensaje en el contenedor "errores" y poner el foco en el campo HORA. Explicar las partes de la expresión regular mediante comentarios.

### Pedir confirmación de envío del formulario. Si se confirma el envío realizará el envío de los datos; en otro caso cancelará el envío.

## Desarrollo



Árbol de nivel 0 y árbol de nivel 2

Sintaxis:

**DOM nivel 0**: la jerarquía comienza con document seguido del componente y el control.

document.getElementById("id-del-control");

**DOM nivel 2**: se accede directamente al id del componente.

document.nombreFormulario.name-del-control;

El acceso a los componentes se hace empleando los métodos del **DOM nivel 2**.

El formulario tiene todos los campos de tipo texto, excepto *Provincia* que es tipo *select*.

En los elementos de tipo texto se envía el *name* y el *value*.

## Recursos

* Tipos de *input*

<https://www.htmlquick.com/es/reference/tags/input.html>

* Propiedades y métodos de los elementos HTML  
  <https://www.w3schools.com/jsref/dom_obj_all.asp>
* Cookies

Se hace referencia a encodeURIComponent  
<https://es.javascript.info/cookie>

* isFinite()

<https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/isFinite>