Formularios

Un **formulario** web sirve para enviar, tratar y recuperar datos que son enviados y recibidos entre un cliente y un servidor web.

Los formularios se definen con **etiquetas**. 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.

Ejemplo:

```
<html>
	<head><title>Ejemplo de formulario</title></head>
	<body>
		<h3>Formulario</h3>
		<form action="http://www.ceu.es" method="post">
		</form>
	</body>
</html>
```

Elementos de un formulario

El elemento principal del formulario se denomina con la etiqueta <**input**>

Atributos de la etiqueta input:

- **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** (cuadro de contraseña, los caracteres aparece ocultos tras asteriscos).
  - **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 en el formulario).
  - **button** (botón del formulario).

- **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 pixeles. 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.
- **Disabled**. 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.

Ejemplos de Tipos de input

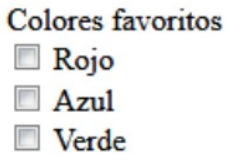
- **Cuadro de texto**: Este input muestra un cuadro de texto vacío en el que el usuario puede introducir un texto. Este es uno de los elementos más usados. La forma de indicar que es un campo de texto es: **type=”text”**

- **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" />
```

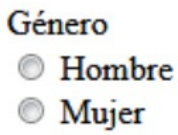
- **Casilla de verificación:** Estos elementos permiten al usuario activar o desactivar la selección de cada una de las casillas de forma individual.

```
<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
```



- **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="genero" value="M"> Hombre
</br><input type="radio" name="genero" value="F"> 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">
```

- **Botón de reset**: Este elemento es un botón que establece el formulario a su estado original.

```
<input type="reset" >
```

- **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"/>
```

- **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"/>

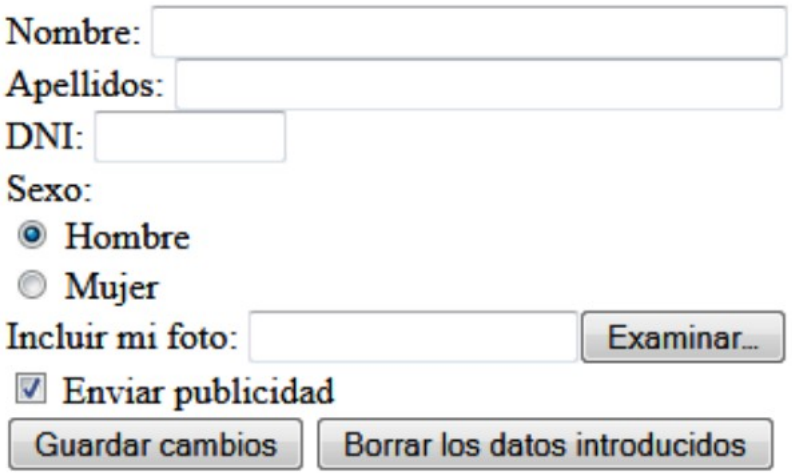
- **Botón**: Existe un elemento botón, al que podemos asociar diferentes funcionalidades. De esta forma no nos tenemos que ceñir los botones de submit o reset que nos ofrecen los formularios.

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

Ejemplo completo:

```
<form action="pagina.php" 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>
```



Formas de selección del objeto Form desde JS

Dentro de un documento html tendremos varias formas de selección de un formulario.

Si partimos del siguiente ejemplo:

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

Tendremos los siguientes métodos de selección del objeto Form en el documento:

Método 1: getElementById(id)

A través del método getElementById() del DOM, nos permite acceder a un objeto a través de su atributo **ID**. Tendremos que tener la precaución de asignar id únicos a nuestros objetos, para evitar que tengamos objetos con id repetidos.

Ejemplo:

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

Método 2: getElementsByTagName(tag)

A través del método getElementsByTagName() del DOM, el cual nos permite acceder a un objeto a través de la **etiqueta** HTML que queramos. **Devuelve un array de elementos**. Por ejemplo para acceder a los objetos con etiqueta form haremos:

```
let formularios = document.getElementsByTagName("form");
let primerFormulario = formularios[0]; // primer formulario del documento
```

o también todo en una única línea:

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

Otra posibilidad interesante que te permite el método anterior, es la de buscar objetos con un padre determinado, por ejemplo:

```
let menu=document.getElementById("menulateral");
let formularios=menu.getElementsByTagName("form"); // formularios contenidos en el menú lateral
let primerFormulario= formularios[0]; // primer formulario en el menú lateral
```

IMPORTANTE: getElementsByTagName() devuelve un array siempre

Método 3: forms[]

Otro método puede ser a través de la colección forms[] del objeto document. Esta colección es un

array, que contiene la referencia a todos los formularios que tenemos en nuestro documento.

Por ejemplo:

```
let formularios = document.forms; // la referencia a todos los formularios del documento
let miformulario = formularios[0]; // primer formulario del documento
```

o bien en una línea:

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

o bien:

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

IMPORTANTE: document.forms devuelve un array siempre

Acceso a propiedades y métodos del formulario desde JS

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 objetoFormulario = document.getElementeByd(id_form);
let paginaDestino = objetoFormulario.action;
```

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

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

Estas dos instrucciones las podemos recomponer usando referencias a objetos:

```
let paginaDestino = document.getElementByid("id").action;
document.forms[0].action = "http://www.educacion.gob.es/recepcion.php";
```

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

Identificación de objetos de un formulario

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.

- 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: **document.getElementById("id-del-control")**

- Otra forma es acceder a un objeto a través de su **atributo name:**

document.nombreFormulario.name-del-control

Por ejemplo, si consideramos un ejemplo sencillo de formulario:

```
<form id="formularioBusqueda" name=”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>
```

Todas las siguientes instrucciones obtienen el objeto entrada:

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

Objeto input de tipo text

- Cada uno de los 4 elementos de tipo texto de los formularios: text, password, hidden y 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.
- Para poder usar estos objetos dentro de nuestros scripts de JavaScript, simplemente será suficiente con asignar un atributo id, a cada uno de los elementos. Es recomendable asignar a cada objeto del formulario un atributo id único y que coincida con el name de ese objeto.
- *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*. Sin lugar a dudas, la propiedad más utilizada en un elemento de tipo texto es por lo tanto **value**. Un script podrá recuperar y ajustar el contenido de la propiedad value en cualquier momento.
- **IMPORTANTE**: el contenido de un value es siempre una cadena de texto, y quizás puedas necesitar realizar conversiones numéricas si quieres realizar operaciones matemáticas con esos textos. Se usan para definir constantes, valores que no pueden modificarse.
- Ejemplo: document.getElementById("nombre").value=”Buenas tardes”;

Propiedades del objeto INPUT de tipo texto

- **defaultValue** Ajusta o devuelve el valor por defecto de un campo de texto.
- **form** Devuelve la referencia al formulario que contiene ese campo de texto.
- **maxLength** Devuelve o ajusta la longitud máxima de caracteres permitidos en el campo
- **name** Ajusta o devuelve el valor del atributo name de un campo de texto. Sí
- **readOnly** Ajusta o devuelve si un campo es de sólo lectura, o no. Sí
- **size** Ajusta o devuelve el ancho de un campo de texto (en caracteres). Sí
- **type** Devuelve el tipo de un campo de texto. Sí
- **value** Ajusta o devuelve el contenido del atributo value de un campo de texto.

Métodos del objeto INPUT de tipo texto

- **select**() Selecciona el contenido de un campo de texto

Objeto input de tipo checkbox

- En un checkbox la propiedad **value** es un texto que está asociado al objeto. *Este texto no se mostrará en la página*, y su finalidad es la de asociar un valor con la opción actualmente seleccionada. Dicho valor será el que se enviará, cuando enviemos el formulario.

- 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.

- Ejemplo: document.getElementById("verano").checked=false;

Propiedades del objeto checkbox

Checked: Ajusta o devuelve el estado checked de un checkbox. Si está o no marcado.

DefaultChecked: Devuelve el valor por defecto del atributo checked. Si está o no marcado por defecto.

Form: Devuelve la referencia al formulario que contiene ese campo checkbox.

Name: Ajusta o devuelve el valor del atributo name de un checkbox.

Type: Nos indica que tipo de elemento de formulario es un checkbox.

Value: Ajusta o devuelve el valor del atributo value de un checkbox.

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.

Cuando le asignamos el mismo name a varios elementos en un formulario, el navegador lo que hace es crear un array con la lista de esos objetos que tienen el mismo name. El contenido del atributo name será el nombre del array. Algunas propiedades, se las podremos aplicar al grupo como un todo; otras en cambio, tendremos que aplicárselas a cada elemento del grupo y lo haremos a través del índice del array del grupo. Por ejemplo, podemos ver cuántos botones hay en un grupo radio, consultando la propiedad **length** de ese grupo:

```
objetoFormulario.nombregrupo.length
```

- **Ejemplo**: 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
  ```

- **Ejemplo de recorrido**: ¿Qué hace este bucle?

```
<script type="text/javascript">
    function mostrarDatos(){
    for (var i=0;i<document.formulario.actores.length; i++){
    if (document.formulario.actores[i].checked)
            alert(document.formulario.actores[i].value);
    }
}
</script>
<body>
    <h1>Trabajando con objetos input de tipo radio</h1>
    <form name="formulario" action="stooges.php">
        <fieldset>
            <legend>Selecciona tu actor favorito:</legend>
            <input type="radio" name="actores" id="actor-1" value="Walter Bruce Willis – 19 de Marzo de
                1955" checked>
            <label for="actor-1">Willis</label>
            <input type="radio" name="actores" id="actor-2" value="James Eugene Jim Carrey - 17 de
        Enero de 1962">
            <label for="actor-2">Carrey</label>
            <input type="radio" name="actores" id="actor-3" value="Luis Tosar - 13 de Octubre de 1971">
            <label for="actor-3">Tosar</label>
            <input type="button" id="consultar" name="consultar" value="Consultar Más Datos"
                onclick="mostrarDatos()">
        </fieldset>
    </form>
```

Objeto select

Un objeto select está compuesto realmente de un array de objetos **option**. El objeto select se suele mostrar como una lista desplegable en la que seleccionas una de las opciones, aunque también tienes la opción de selecciones múltiples, según definas el objeto en tu documento.

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. Si lo que quieres hacer es detectar la opción seleccionada por el usuario, y quieres usar JavaScript, tendrás que utilizar propiedades tanto de select, como de option.

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

El valor devuelto por esta propiedad, es el índice de la opción actualmente seleccionada. *Los índices comienzan en la posición 0.* Para conocer la longitud del objeto select, usamos el atributo **length**.

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.

Para conocer si una opción está marcada, se utiliza el atributo **selected: options[i].selected**

if(objetoSelect.options[i].selected)

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

Veamos las formas de acceso a esas propiedades:

objetoFormulario.nombreCampoSelect.**options**[n].text

objetoFormulario.nombreCampoSelect.**options**[n].value

Ejemplo:

```
<script type="text/javascript">
function consultar(){
    var objProvincias=document.getElementById("provincias");
    var texto=objProvincias.options[objProvincias.selectedIndex].text;
    var valor=objProvincias.options[objProvincias.selectedIndex].value;
    alert("Datos de la opción seleccionada:\n\nTexto: "+texto+"\nValor: "+valor);
}
</script>

<body>
    <h1>Trabajando con un objeto Select</h1>
    <form id="formulario" action="pagina.php">
        <p>
        <label for="provincias">Seleccione provincia: </label>
        <select name="provincias" id="provincias">
            <option value="C">La Coruña</option>
```

```
                    <option value="LU">Lugo</option>
                    <option value="OU">Ourense</option>
                    <option value="PO">Pontevedra</option>
                </select>
                </p>

        Selecciona una opción y pulsa el botón.
        <input type="button" name="boton" value="Consultar información de la opción"
        onclick="consultar()"/>
        </form>
</body>
```

Propiedad innerHTML: Permite poner código html en el elemento DOM:

```
<p id=”texto”></p>
document.getElementById(“texto”).innerHTML=”Texto para el párrafo”
```

Actualización de atributos

Podemos actualizar los valores de los atributos de elementos accediendo a ellos a través de un punto o bien con el método setAttribute().

Ejemplo:

```
<img src=”imagen.png” id =”i1”>
```

Forma 1:

```
document.getElementByd(“i1”). src=”nuevaImagen.png” ;
```

Forma 2:

```
let mi =  document.getElementByd(“i1”);
mi.setAttribute(“src” , “nuevaImagen.png”);
```

Estilos

Hay distintas formas de modificar estilos en javascript:

```
mi.style.backgroundColor = “red”;
mi.setAttribute(“class”, “rojo”);
mi.setAttribute(“style”, “background-color:red”);
```

Eventos

Los eventos son mecanismo que se accionan cuando el usuario realiza un cambio sobre una página web.

El encargado de crear la jerarquía de objetos que compone una página web es el DOM (Document Object Model). Por tanto es el DOM el encargado de gestionar los eventos.

Para poder controlar un evento se necesita un manejador. El manejador es la palabra reservada que indica la acción

que va a manejar. En el caso del evento click, el manejador sería onClick.

La especificación DOM define cuatro grupos de eventos dividiéndolos según su origen: Eventos del ratón, eventos del teclado, eventos HTML o eventos DOM

1.- Eventos del ratón

- Click. Este evento se produce cuando pulsamos sobre el botón izquierdo del ratón. El manejador de este evento es onclick.
- Dblclick. Este evento se acciona cuando hacemos un doble click sobre el botón izquierdo del ratón. El manejador de este evento es ondblclick.
- Mousedown. Este evento se produce cuando pulsamos un botón del ratón. El manejador de este evento es onmousedown.
- Mouseout. Este evento se produce cuando el puntero del ratón esta dentro de un elemento y este puntero es desplazado fuera del elemento. El manejador de este evento es onmouseout
- Mouseover. Este evento al revés que el anterior se produce cuando el puntero del ratón se encuentra fuera de un elemento, y este se desplaza hacia el interior. El manejador de este evento es onmouseover.
- Mouseup. Este evento se produce cuando soltamos un botón del ratón que previamente teníamos pulsado. El manejador de este evento es onmouseup.
- Mousemove. Se produce cuando el puntero del ratón se encuentra dentro de un elemento. Es importante señalar que este evento se producirá continuamente una vez tras otra mientras el puntero del ratón permanezca dentro del elemento. El manejador de este evento es onmousemove

2.- Eventos del teclado

- Keydown. Este evento se produce cuando pulsamos una tecla del teclado. Si mantenemos pulsada una tecla de forma continua, el evento se produce una y otra vez hasta que soltemos la misma. El manejador de este evento es **onkeydown**.
- Keypress. Este evento se produce si pulsamos una tecla de un carácter alfanumérico (El evento no se produce si pulsamos enter, la barra espaciadora, etc…). En el caso de mantener una tecla pulsada, el evento se produce de forma continuada. El manejador de este evento es **onkeypress**.
- Keyup. Este evento se produce cuando soltamos una tecla. El manejador de este evento es **onkeyup**.

3.- Eventos HTML

- Load. El evento load hace referencia a la carga de distintas partes de la página. Este se produce en el objeto Window cuando la página se ha cargado por completo. En el elemento <img> actúa cuando la imagen se ha cargado. En el elemento <object> se acciona al cargar el objeto completo. El manejador es **onload**.

- Unload. El evento unload actúa sobre el objeto Window cuando la pagina ha desaparecido por completo (por ejemplo, si pulsamos el aspa cerrando la ventana del navegador). También se acciona en el elemento <object> cuando desaparece el objeto. El manejador es **onunload**.
- Abort. Este evento se produce cuando el usuario detiene la descarga de un elemento antes de que haya terminado, actúa sobre un elemento <object>. El manejador es **onabort**
- Error. El evento error se produce en el objeto Window cuando se ha producido un error en JavaScript. En el elemento <img> cuando la imagen no se ha podido cargar por completo y en el elemento <object> en el caso de que un elemento no se haya cargado correctamente. El manejador es **onerror**.
- Select. Se acciona cuando seleccionamos texto de los cuadros de textos <input> y <textarea>. El manejador es **onselect**.
- Change. Este evento se produce cuando los cuadros de texto <input> y <textarea> pierden el foco y el contenido que tenían ha variado. También se producen cuando un elemento <select> cambia de valor. El manejador es **onchange**.
- Submit. Este evento se produce cuando pulsamos sobre un botón de tipo submit. El manejador es **onsubmit**.
- Reset. Este evento se produce cuando pulsamos sobre un botón de tipo reset. El manejador es **onreset**.
- Resize. Este evento se produce cuando redimensionamos el navegador, actúa sobre el objeto Window. El manejador es **onresize**.
- Scroll. Se produce cuando varía la posición de la barra de scroll en cualquier elemento que la tenga. El manejador es **onscroll**.
- Focus. Este evento se produce cuando un elemento obtiene el foco. El manejador es **onfocus**.
- Blur. Este evento se produce cuando un elemento pierde el foco. El manejador es **onblur**

4.- Eventos DOM

- DOMContentLoad. Se produce cuando el documento se ha cargado.
- DOMSubtreeModified. Este evento se produce cuando añadimos o eliminamos nodos en el subárbol de un elemento o documento.
- DOMNodeInserted. Este evento se produce cuando añadimos un nodo hijo a un nodo padre.
- DOMNodeRemoved. Este evento se produce cuando eliminamos un nodo que tiene nodo padre.
- DOMNodeRemovedFromDocument. Este evento se produce cuando eliminamos un nodo del documento.
- DOMNodeInsertedIntoDocument. Este evento se produce cuando añadimos un nodo al documento.

FORMA 1: Modelo de registro de eventos en línea.

- En el modelo de registro de eventos en línea (estandarizado por Netscape), el evento es añadido como un atributo más a la etiqueta HTML, como por ejemplo:

  <A href="pagina.html" onClick="alert('Has pulsado en el enlace')">Pulsa aqui

- Cuando hacemos click en el enlace, se llama al gestor de eventos onClick (al hacer click) y se ejecuta el script; que contiene en este caso una alerta de JavaScript. También se podría realizar lo mismo pero llamando a una función:

```
<A href="pagina.html" onClick="alertar()">Pulsa aqui</a>
function alertar(){
 alert("Has pulsado en el enlace");
}
```

- Este modelo no se recomienda, y aunque lo has visto en ejemplos que hemos utilizado hasta ahora, tiene el problema de que estamos mezclando la estructura de la página web con la programación de la misma, y lo que se intenta hoy en día es separar la programación en JavaScript, de la estructura HTML.

FORMA 2: Modelo de registro de eventos tradicional.

- En los navegadores antiguos, el modelo que se utilizaba era el modelo en línea. Con la llegada de DHTML, el modelo se extendió para ser más flexible. En este nuevo modelo el evento pasa a ser una propiedad del elemento, así que por ejemplo los navegadores modernos ya aceptan el siguiente código de JavaScript:

```
elemento.onclick = hacerAlgo; // cuando el usuario haga click en el objeto, se llamará a la
funcion hacerAlgo(){
---
}
```

- Esta forma de registro, no fue estandarizada por el W3C, pero debido a que fue ampliamente utilizada por Netscape y Microsoft, todavía es válida hoy en día. La ventaja de este modelo es que podremos asignar un evento a un objeto desde JavaScript, con lo que ya estamos separando el código de la estructura.
- Los nombres de los eventos sí que van siempre en minúsculas: elemento.onclick = hacerAlgo;
- Para eliminar un gestor de eventos de un elemento u objeto, le asignaremos null:

```
elemento.onclick = null;
```

- Otra gran ventaja es que, como el gestor de eventos es una función, podremos realizar una llamada directa a ese gestor, con lo que estamos disparando el evento de forma manual.

Por ejemplo:

elemento.onclick(); // Al hacer ésto estamos disparando el evento click de forma manual y se ejecutará la //función hacerAlgo()

SIN PARÉNTESIS

- En el registro del evento no usamos paréntesis (). El método onclick espera que se le asigne una función completa. Si haces: element.onclick = hacerAlgo(); la función será ejecutada y el resultado que devuelve esa función será asignado a onclick. Pero ésto no es lo que queremos que haga, queremos que se ejecute la función cuando se dispare el evento.

FORMA 3: Modelo de registro avanzado de eventos

- Desde JS se recupera el elemento que provoca el evento: por ejemplo el botón. Una vez recuperado el elemento, se le asocia una función a través del método **addEventListener**().
- 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)**

- Por ejemplo para registrar la función alertar:

```
document.getElementById("mienlace").addEventListener('click',alertar,false);
function alertar(){
 alert("Te conectaremos con la página: "+this.href);
}
```

This hace referencia al objeto que ha provocado el evento. En este caso un enlace.

- 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**()
- Para **eliminar** un evento de un elemento, usaremos el método **removeEventListener**():

```
elemento.removeEventListener();
```

Ejemplo de preventDefault:

```
<form id=”miFormulario”>
        <input type=”text” id=”n1” size=30 />
        <button id=”b1”>validar</button>
```

```
</form>
validar = (e) =>{
        e.preventDefault();
        alert(“Ahora el formulario no recarga, hemos anulado el comportamiento por defecto”);
}

let $boton = document.getElementById(“b1”);
$boton.addEventListener(“click”, validar);
```

Obteniendo información del evento. Importante

JavaScript permite obtener información sobre el ratón y el teclado mediante un objeto especial llamado **event.**

De forma automática, la función asociada al evento, tiene como argumento un parámetro que representa al evento.

Cuando se produce un evento, vamos a poder recuperar qué objeto ha provocado el evento así como todos sus atributos. Para ello, se utilizan **target y type.** Las funciones asociadas a los eventos, tienen como primer parámetro el evento

- **Target:** Elemento que ha generado el evento
- **Type**: El nombre del evento.
- Ejemplo:

Supongamos que tenemos un botón <button id="**mienlace**">pulsa</button>

Si le asociamos un evento al hacer ‘click’ de la siguiente forma:

document.getElementById("**mienlace**").addEventListener("click", **alertar**);

La siguiente función alertar podrá hacer lo siguiente:

```
alertar = (e) => {
 alert(e.type);
 alert(e.target);
 e.target.style.color = "blue";
};
```

¿Qué hace esta función?

Audio

Desde Javascript podemos reproducir sonidos y pausarlos:

<audio src=”pajaritos.mp3” type=”audio/mp3” >

Tu navegador no soporta el elemento de audio.

</audio>

```
<script>
     var audio = document.getElementById("myAudio");

    function playSound() {
       audio.play();      // reproduce
    }

    function stopSound() {
        audio.pause();     // para la reproducción
        audio.currentTime = 0;  // reinicia la reproducción
   }
</script>
```

NOTA: Para poder usar bootstrap en los ejercicios en head ponemos:

```
<link
    rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
  />
```

Ver estilos: https://getbootstrap.com/docs/3.4/css/

Ejercicios

1. Crear un formulario con un input de tipo fecha y 3 input de tipo text: nombre, apellido1 y apellido2. Asignarle un valor por defecto a los texto. Recuperar en javascript **todos** los nombres (atributo name)de los elementos del formulario y mostrarlos con alert. Después mostrar **con alert sólo los input de tipo text**.

   Añadir al ejercicio anterior, un alert para obtener el valor del nombre y del apellido 1 y apellido 2 juntos, a través del **id** de cada campo.

   Añadir al ejercicio anterior, un alert para obtener el valor del nombre, y del apellido 1 y apellido 2 juntos, a través del atributo **name** de cada campo.

2. Crear un formulario con una etiqueta label y un input type text para que escribáis vuestro nombre. Cuando hagáis **click** fuera del campo, tendréis que convertir el nombre a mayúsculas en el input. Debe quedarse señalado. Usar función flecha.

   El formulario también tendrá una etiqueta <audio > y dos botones: Reproducir sonido y Detener sonido. Buscar un mp3 y guardarlo en el proyecto. Cuando pulse ‘ Reproducir sonido’ debe escucharse el sonido y cuando se pulse ‘Detener sonido’, debe pararse. No debe reiniciarse el sonido cuando se pare, si luego le damos a reproducir, debe continuar por donde se quedó.

3. Crear un formulario con 6 checkbox y un botón ‘validar’ . Cuando pulsemos el botón, validará que hay 3 o más checkbox marcados. Mostrar por alert un mensaje que indique si hay o no 3 o más de 3 checkbox marcados.

Una vez realizado, crear otro checkbox independiente con dos botones nuevos: marcar y desmarcar. Cuando pulsemos el botón marcar, marcaremos el checkbox y cuando pulsemos el botón desmarcar, desmarcará el checkbox.

4. Crear una página html con un formulario y con un objeto select provincias con 4 provincias (option).

   Crear un botón tal que al pulsarlo muestre en un alert la información de la provincia seleccionada: índice del option seleccionado, descripción y valor.

   Añadir otro objeto select **multiple, provincias,** pero que permita la selección multiple, y otro botón. Al pulsarlo, mostrar por alert la descripción de las provincias seleccionadas.

5. Crear una página html con un campo de texto (input), poner vuestro nombre y al pulsar un botón, mostrareis el contenido del input debajo del botón.

   Pista: usar etiqueta div

   Añadir al ejercicio que después de pulsar el botón, limpie el cuadro de texto y ponga el foco en él.

   Añadir ahora que el nombre lo muestre en color azul y en negrita.

6. Crear un formulario con los componentes más usuales: editor de texto, área de texto, un conjunto de radio botones, un par de checkbox, y una lista desplegable. Colocaremos un botón, al hacer clic sobre él, recogeremos la información introducida por el usuario y la mostraremos en un div.

7. Crear una página html con dos input de tipo text. Cuando se escriba en ambos, el color del texto debe ser rojo y cuando pierda el foco, el color del texto será negro.

   Después, añadir un botón ‘Púlsame’, con un evento, tal que al hacer click ponga el color del texto azul.

   Añadir otro botón ‘Cambia’, de forma que al pasar por encima ponga el borde del botón de color verde y cuando salga del botón, lo ponga de color naranja.

8. Crear una página con dos botones: uno ‘Sumar’ y otro ‘Restar’. Por defecto se empezará a contar desde 5.

   Al pulsar el botón Sumar, se sumará uno, y al pulsar el botón restar, se restará uno.

   Cuando se llegue al valor 3, mostrar en el centro de la pantalla un mensaje que indique ‘Alcanzado valor 3’. Cuando vuelva a ser distinto de 3, el mensaje debe desaparecer.

9. Crear un formulario con:
   1 checkbox, que muestre un mensaje alert diciendo si está o no marcado cada vez que lo marque o desmarque.
   3 radios, que muestre si estan o no marcados cada vez que alguno cambie
   1 select, con 5 options, que muestre el valor del elemento que ha seleccionado.
   1 texto, que muestre el valor si cambia
   1 textarea, que muestre su valor si cambia.
   Utilizar para ello addEventListener.

10. Crear un estilo llamado boton que tenga una altura y anchura de 50px.
    El formulario tendrá 10 botones con ese estilo creado y con values del 0 al 9 respectivamente.
    Crear para todos ellos un evento tal que al pulsarlo muestre en una etiqueta div el valor del boton pulsado.

    NOTA: Para crear estilos: .boton {…..}

    Pista: crear los eventos con un bucle for.

11. Crear un formulario con 2 input de tipo de texto: Nombre y Apellidos. Y un botón validar.

    Cuando pulsemos el botón tendremos que validar:

    1. Si el nombre comienza por vocal.
    2. Si los apellidos tienen más de dos palabras.

    Poner debajo de cada input un mensaje en caso de no cumplir su validación.

    Una vez hecho, añadir que el mensaje aparezca con letra de color rojo.

12. Sobre el ejercicio anterior, modificar lo necesario para que una vez que se pulsa el botón, si se cumplen las dos validaciones, el formulario debe redirigir a la página de: https://ceu.es.

    En caso contrario, el comportamiento debe ser igual al ejercicio 11.

13. Dado el formulario:

```
<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="provincia">Provincia:</label>
    <select name="provincia" id="provincia">
        <option value="0" selected="selected">Seleccione Provincia</option>
            <option value="H">Huesca</option>
            <option value="ZA">Zaragoza</option>
            <option value="T">Teruel</option>
        </select>
<fieldset>
    <div id="resultado"></div>
    <input type="reset" name="limpiar" id="limpiar" value="Limpiar" />
    <input type="submit" name="enviar" id="enviar" value="Enviar" />
</fieldset>
</form>
```

Validar el formulario antes de enviarlo. Para ello se debe comprobar que el nombre, apellidos y edad tienen valor. La edad además debe ser un número y sus valores deben estar comprendidos entre 0 y 105. También se debe validar que se ha seleccionado alguna provincia.

Si se cumplen todas las validaciones, debe preguntarse al usuario con un mensaje si deseas enviar el formulario. En caso afirmativo, se enviará. En caso negativo no hará nada.

Cuando no se cumpla alguna validación debe aparecer el mensaje correspondiente en la etiqueta div (id=”resultado”) del final en color rojo.