4 . Interacción con el usuario

- Eventos y formularios -

Utilización de formularios desde código

Elementos de un formulario

```
<html>
<head>
    <title>Mi formulario</title>
</head>

<body>
    <h3>Formulario</h3>
    <form action="pagina.php" method="post">
        <!-- Elementos -->
    </form>
</body>
</html>
```

- `action`: **es la url de la página a la que redirige el formulario.**
- `method`: **puede ser POST o GET, dependiendo del enmascaramiento de los datos enviados.**

Utilización de formularios desde código

Elementos de un formulario

Propiedad	Descripción
acceptCharset	Ajusta o devuelve el valor del atributo accept-charset en un formulario.
action	Ajusta o devuelve el valor del atributo Action en un formulario.
enctype	Ajusta o devuelve el valor del atributo enctype en un formulario.
length	Devuelve el número de elementos de un formulario.
method	Ajusta o devuelve el valor del atributo method en un formulario.
name	Ajusta o devuelve el valor del atributo name en un formulario.
target	Ajusta o devuelve el valor del atributo target en un formulario.

Método	Descripción
reset()	Resetea un formulario.
submit()	Envía un formulario.

```
form.elements[]
```

- Devuelve un array con todos los input de un formulario.

Utilización de formularios desde código

Elementos de un formulario

Propiedad	Descripción
Type	Puede ser, a su vez: • Text (cuadro de texto). • Password (cuadro de 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 en el formulario). • Button (botón del formulario).
Name	Asigna un nombre al elemento. Es necesario para que el servidor pueda trabajar con él.
Value	Inicializa el valor del elemento.

Utilización de formularios desde código

Elementos de un formulario

Propiedad	Descripción
Size	Tamaño inicial. En campos text y password se refiere al número de caracteres.
Maxlength	Número máximo de caracteres que pueden contener text y password.
Checked	Para elementos checkbox y radio. Indica si el checkbox está marcado; o cuál de los botones de radio está pulsado.
Disabled	El elemento aparece deshabilitado. El dato no se envía al servidor.
Readonly	Bloquea el contenido de control: el valor del elemento no se puede modificar (solo lectura).
Src	Solo se utiliza para asignar la url de una imagen que se colocará en como botón.
Alt	Descripción del elemento, que se visualiza si nos ponemos con el ratón sobr él.
Required	Indica si el campo debe ser completado antes de enviar el formulario.

Método	Descripción
select()	Selecciona el contenido de un campo de texto.

Utilización de formularios desde código

Cuadro de texto. Ejemplo:

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

Cuadro de texto. Ejemplo:

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

Campo oculto. Ejemplo:

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

Utilización de formularios desde código

Casilla de verificación. Ejemplo:

```
<p>Frutas</p>
<input name="fresa" type="checkbox" value="fresa"/>Fresa<br/>
<input name="platano" type="checkbox" value="platano"/>Plátano <br/>
<input name="manzana" type="checkbox" value="manzana"/>Manzana <br/>
```

Radio button. Ejemplo:

```
<p>Frutas</p>
<input name="fruta" type="radio" value="fresa"/>Fresa<br/>
<input name="fruta" type="radio" value="platano"/>Plátano <br/>
<input name="fruta" type="radio" value="manzana"/>Manzana <br/>
```

Utilización de formularios desde código

Botón de envío. Ejemplo:

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

Botón de reset. Ejemplo:

```
<input name="enviar" type="reset" value="Borrar formulario"/>
```

Botón de imagen. Ejemplo:

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

Botón. Ejemplo:

```
<input name="enviar" type="button" value="Botón de prueba"/>
```

Selector de ficheros. Ejemplo. Fichero adjunto:

```
<input type="file" name="fichero" />
```

Utilización de formularios desde código

Elementos de un formulario

Propiedad	Descripción
for	Indica a qué otro elemento del formulario está asociada.

Label. Ejemplo:

```
<label for =“nombre”>Nombre</label>
<input type=“text” name=“nombre” />
```

Utilización de formularios desde código

Elementos de un formulario

Propiedad	Descripción
Name	Asigna un nombre al elemento. Es necesario para que el servidor pueda trabajar con él.
Rows	Numero de líneas del textarea.
Cols	Ancho del textarea.
Maxlength	Número máximo de caracteres que pueden contener text y password.
Checked	Para elementos checkbox y radio. Indica si el checkbox está marcado; o cuál de los botones de radio está pulsado.
Disabled	El elemento aparece deshabilitado. El dato no se envía al servidor.
Readonly	Bloquea el contenido de control: el valor del elemento no se puede modificar (solo lectura).
Required	Indica si el campo debe ser completado antes de enviar el formulario.

Textarea. Ejemplo:

```
<textarea name="comentarios" rows="4" cols="30" ></textarea>
```

Utilización de formularios desde código

Elementos de un formulario

Propiedad	Descripción
Name	Asigna un nombre al elemento. Es necesario para que el servidor pueda trabajar con él.
Disabled	El elemento aparece deshabilitado. El dato no se envía al servidor.
Multiple	Permite marcar más de una fila visible. Requiere del atributo size.
Size	En caso de que múltiple esté marcado, muestra cuántas filas estarán visibles.
Required	Indica si el campo debe ser completado antes de enviar el formulario.

Textarea. Ejemplo:

Dentro del elemento `<select>` se encuentra la etiqueta `<option>` con el campo value para almacenar el valor y en su interior lo que se visualizará.

Utilización de formularios desde código

Elementos de un formulario

Propiedad	Descripción
Name	(Fieldset) Asigna un nombre al elemento.
Align	(Legend) Permite elegir la alineación del texto del legend sobre el fieldset.

Fieldset se utiliza para agrupar elementos del formulario mediante una línea y permite asignar un título para esta agrupación gracias a la etiqueta `<legend>`

Fieldset. Ejemplo:

```
<form>
    <fieldset>
        <legend>Datos:</legend>
        Nombre: <input type="text" size="30"><br>
        Email: <input type="text" size="30"><br>
    </fieldset>
</form>
```

Utilización de formularios desde código

Desplegable. Ejemplo:

```
<p>Comida: </p>
<select name="comida”>
<option value=" pasta">Pasta</option>
    <option value="carne">Carne</option>
    <option value="pescado">Pescado</option>
    <option value="postre">Postre</option>
</select>
```

Desplegable con elección múltiple. Ejemplo:

```
<p>Comida: </p>
<select multiple size=“3” name="comida[]”>
<option value=" pasta">Pasta</option>
    <option value="carne">Carne</option>
    <option value="pescado">Pescado</option>
    <option value="postre">Postre</option>
</select>
```

Utilización de formularios desde código

```
<html>

<head>
    <title>Formulario</title>
</head>

<body>
    <h1>Formulario</h1>
    <form action="procesar.php" method="post">
        <table>
            <tr>
                <td>Nombre: </td>
                <td><input type="text" name="nombre" /></td>
            </tr>
            <tr>
                <td>Password: </td>
                <td><input type="password" name="psw" /></td>
            </tr>
            <tr>
                <td>Sexo : </td>
                <td>
                    <input type="radio" name="sexo" value="H" checked /> Hombre
                    <input type="radio" name="sexo" value="M" /> Mujer
                </td>
            </tr>
```

Utilización de formularios desde código

```
<tr>
                <td>Mayor de 18:</td>
                <td><input type="checkbox" name="mayor" checked /></td>
            </tr>
            <tr>
                <td>Fichero: </td>
                <td><input type="file" name="upfile" /></td>
            </tr>
            <tr>
                <td>Hidden: </td>
                <td><input type="hidden" name="escondido" value="sorpresa" /></td>
            </tr>
        </table>
<tr>
            <td>Comida: </td>
            <td>
                <select name="comida">
                    <option value="pasta">Pasta</option>
                    <option value="carne">Carne</option>
                    <option value="pescado">Pescado</option>
                    <option value="postre">Postre</option>
                </select>
            </td>
        </tr>
```

Utilización de formularios desde código

```
        <tr>
            <td>Bebidas: </td>
            <td>
                <select multiple size="3" name="bebida[]">
                    <option value="agua">Agua</option>
                    <option value="vino">Vino</option>
                    <option value="cerveza">Cerveza</option>
                </select>
            </td>
        </tr>
        </table>
        <p>
            <input type="submit" value="Enviar" />
            <input type="reset" value="Borrar" />
        </p>
    </form>
</body>

</html>
```

Utilización de formularios desde código

EJERCICIO: U4T1 - Formulario:

- Prepara un formulario diseñado para almacenar los discos con los que trabajamos en el ejercicio del tema anterior (U3T8). Recuerda qué elementos se almacenaban, y elije el componente de formulario más adecuado:
  - Nombre del disco.
  - Grupo de música o cantante.
  - Año de publicación.
  - Tipo de música (podrá ser “rock”, “pop”, “punk” o “indie”);
  - Localización: almacenará un número de estantería.
  - Prestado: almacenará un valor booleano. Por defecto será false.

Modificación de apariencia y comportamiento

Seleccionar un formulario
Disponemos del siguiente formulario:

```
<form id="miFormulario" name="miFormulario" action="siguiente.php">
```

getElementById()

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

getElementsByTagName()

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

O en una sola línea

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

Modificación de apariencia y comportamiento

Seleccionar un formulario
Disponemos del siguiente formulario:

```
<form id="miFormulario" name="miFormulario" action="siguiente.php">
```

forms[0]

```
let formularios = document.forms; // Devuelve un array de formularios
let primerform = formularios [0];
```

O en un sola línea

```
let formularios = document.forms[0];
```

Modificación de apariencia y comportamiento

Seleccionar un formulario
Disponemos del siguiente formulario:

```
<form id="miFormulario" name="miFormulario" action="siguiente.php">
```

forms["nombre"]

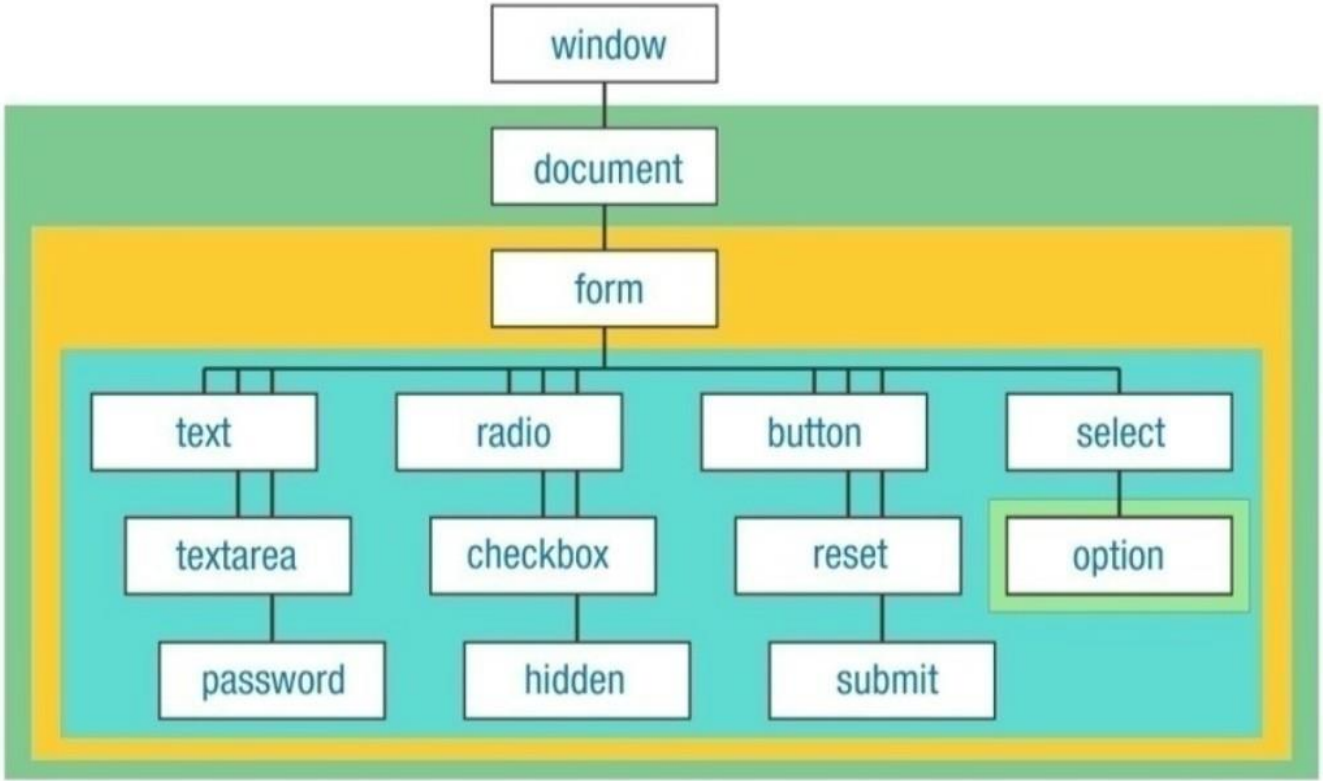
```
let formularios = document.forms; //Devuelve un array de formularios
let primerform = formularios ["miFormulario"];
```

O en un sola línea

```
let formularios = document.forms["miFormulario"];
```

Modificación de apariencia y comportamiento

El formulario como contenedor de objetos



Modificación de apariencia y comportamiento

Selección de objetos de un formulario

form.elements[] devuelve un array con todos los input de un formulario.

getElementByID() devuelve un elemento que tenga un id determinado.

getElementsByTagName() devuelve un array de elementos de un tipo.

Modificación de apariencia y comportamiento

Selección de objetos de un formulario

```
<!-- En este ejemplo: -->
<form id="formularioBusqueda" action="siguiente.php">
    <p>
        <input type="text" id="entrada" name="cEntrada">
        <input type="submit" id="enviar" name="enviar" value="Buscar...">
    </p>
</form>

// Para hacer referencia al input de tipo texto podemos hacerlo mediante:
document.getElementById("entrada")
document.formularioBusqueda.cEntrada
document.formularioBusqueda.elements[0]
document.forms["formularioBusqueda"].elements["cEntrada"]
document.forms["formularioBusqueda"].cEntrada
```

Modificación de apariencia y comportamiento

Objetos input de tipo texto

Es recomendable utilizar un `id` igual que el name para poder referirnos a ellos con `getElementById`
Podemos referirnos a cualquiera de sus propiedades para cambiarlas.

Ejemplo:

```
<input type="text" name="nombre" id="nombre"/>
document.getElementById("nombre").value="Valor nuevo"; //Cambia el valor del texto
```

Objetos input de tipo checkbox

La propiedad value es el texto asociado al objeto.
El formulario solamente envía los datos si el checkbox está marcado.

Ejemplo:

```
<input type="checkbox" id="cantidad" name="cantidad" value="100"/>
document.getElementById("cantidad").checked = true; //Marca el checked
```

Modificación de apariencia y comportamiento

Objetos input de tipo radio

Cuando utilizamos varios radiobutton con el mismo nombre, se crea un array de objetos input radio.
Podemos consultar el número de radio button con formulario.gruporadio.length;
Podemos consultar si un radio button está marcado con formulario.gruporadio[num].checked y devolverá true o false.
Ejemplo:

```
for (var i = 0; i < document.formulario.frutas.length; i++) {
  if (document.formulario.frutas[i].checked)   // Comprueba que es true
    alert(document.formulario.actores[i].value);
}
```

Objetos select

La propiedad selectedIndex devuelve el índice de la opción que ha sido seleccionada.
Cada opción tiene dos propiedades accesibles: text y value que indican el valor visible y el valor interno, respectivamente.
Ejemplo:

```
<option value="ZA">Zamora</option>
formulario.nombreCampoSelect.options[n].text; // Devuelve Zamora
formulario.nombreCampoSelect.options[n].value; // Devuelve ZA
```

Modificación de apariencia y comportamiento

Usando this

Utilizamos this para hacer referencia al objeto que llama a un evento.
Ejemplo:

```
<input type="text" name="nombre" id="nombre" value="Laura" onclick="identificar(this)" />

function identificar(objeto) {
  var nombre = objeto.name;
  var valor = objeto.value;
  var id = objeto.id;
  var tipo = objeto.type;
  alert(
    "Datos del objeto: " + nombre + " - " + valor + " - " + id + " - " + tipo
  );
}
```

Modelo de gestión de eventos

Evento

- Mecanismo que se acciona cuando el usuario realiza un cambio sobre la página web.
- Capturar un evento es programar una acción para que se realice una tarea.
- El encargado de gestionar los eventos es el DOM (Document Object Model).

Manejador (handler)

- Palabra reservada que indica la acción que se va a manejar. Por ejempo, en el evento click el manejador es onClick.

Modelo de gestión de eventos

Eventos de ratón

Evento	Manejador	Descripción
Click	onclick	Cuando pulsamos el botón derecho del ratón.
Dblclick	ondblclick	Cuando hacemos doble click el botón derecho del ratón.
Mousedown	onmousedown	Cuando pulsamos un botón del ratón.
Mouseup	onmouseup	Cuando soltamos un botón del ratón que previamente teníamos pulsado.
Mouseout	onmouseout	Cuando el puntero del ratón esta dentro de un elemento y este puntero es desplazado fuera del elemento.
Mouseover	onmouseover	Cuando el puntero del ratón se encuentra fuera de un elemento, y este se desplaza hacia el interior.
Mousemove	onmousemove	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.

Modelo de gestión de eventos

Eventos de teclado

Evento	Manejador	Descripción
Keydown	onkeydown	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
Keyup	onkeyup	Cuando soltamos una tecla.

Modelo de gestión de eventos

Eventos HTML

Evento	Manejador	Descripción
Load	onload	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.
Unload	onunload	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.
Abort	onabort	Este evento se produce cuando el usuario detiene la descarga de un elemento antes de que haya terminado, actúa sobre un elemento <object>.
Error	onerror	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.

Modelo de gestión de eventos

Eventos HTML

Evento	Manejador	Descripción
Change	onchange	Cuando los cuadros de texto <input> y <textarea> pierden el foco y el contenido que tenían ha variado. Cuando un elemento <select> cambia de valor.
Submit	onsubmit	Cuando pulsamos sobre un botón de tipo submit en un formulario
Reset	onreset	Cuando pulsamos sobre un botón de tipo reset en un formulario.
Error	onerror	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.

Modelo de gestión de eventos

Eventos HTML

Evento	Manejador	Descripción
Resize	onresize	Cuando redimensionamos el navegador, actúa sobre el objeto Window.
Scroll	onscroll	Cuando varía la posición de la barra de scroll en cualquier elemento que la tenga
Focus	onfocus	Cuando un elemento obtiene el foco. En elementos button, input, label, select, textarea, body.
Blur	onblur	Cuando un elemento pierde el foco. En elementos button, input, label, select, textarea, body.

Utilización de formularios desde código

Eventos DOM

Evento	Descripción
DOMSubtreeModified	cuando añadimos o eliminamos nodos en el subárbol de un elemento o documento.
DOMNodeInserted	cuando añadimos un nodo hijo a un nodo padre.
DOMNodeRemoved	cuando eliminamos un nodo que tiene nodo padre.
DOMNodeRemovedFromDocument	cuando eliminamos un nodo del documento.
DOMNodeInsertedIntoDocument	cuando añadimos un nodo al documento.

Modificación de apariencia y comportamiento

Eventos

Los eventos se ejecutan cuando el usuario interactúa con la página (haciendo clic, pulsando una tecla, moviendo el ratón, etc.) o bien se tratan de eventos no relacionados con acciones del usuario (como onload, que sucede cuando la página se carga).

Un poco de historia:

Incompatibilidades de navegadores: había que comprobar si los métodos eran soportados.

Modelo de registros de eventos en línea: se añade como atributo a la etiqueta html:
<a href="..." onClick=accion/>

Modelo de registro de eventos tradicional: los eventos son una propiedad de los elementos: elemento.onclick=accion;

Modelo avanzado de registro de eventos del W3C:
utiliza addListener();

Modificación de apariencia y comportamiento

Modelo de registro de eventos en línea

El evento se añade como atributo de una etiqueta html.
Se ejecuta primero el script y luego la acción por defecto.
Ejemplos:

1)
```
<a href="siguiente.html" onclick="alert('Has pulsado en el enlace')">Pulsa aquí</a>
```
2)
```
<a href="siguiente.html" onclick="alertar()">Pulsa aquí</a>
function alertar() {
  alert("Has pulsado el enlace");
}
```

Para evitar que el navegador ejecute acción por defecto:
```
<a href="siguiente.html" onclick="alertar(); return false;">Pulsa aquí</a>
```
De este modo no ejecutaría la acción por defecto: redirigir a la página siguiente.html.

Para preguntar si ejecutar o no la acción por defecto:
```
<a href="siguiente.html" onclick="return preguntar()">Pulsa aquí</a>
function preguntar() {
  return confirm("¿Desea realmente ir a esa dirección?");
}
```

Modificación de apariencia y comportamiento

¿Por qué no utilizar el modelo de registro de eventos en línea?

No separa la programación de la estructura.

Si la función redirige a una página no se ejecutan las demás tareas.

Al ejecutar la acción por defecto se pueden perder los objetos que hay en memoria (ej. Al cambiar de página).

Modificación de apariencia y comportamiento

Modelo de registro de eventos tradicional

El evento se convierte en una propiedad del elemento:
Ejemplos:

```
elemento.onclick = acción;
function accion() {…}
```

Para eliminar un gestor de eventos de un elemento:

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

Para llamar al gestor de eventos cuando queramos:

```
elemento.onclick();
```

No usamos paréntesis, porque no necesitamos que devuelva el resultado.

~~**elemento.onclick=accion();**~~ -> `elemento.onclick();`

Modificación de apariencia y comportamiento

Modelo de registro de eventos avanzado del W3C

Utiliza un evento addEventListener() con tres argumentos:

- **Tipo de evento.**
- **Función a ejecutar.**
- **Valor booleano para elegir la fase de captura (true) o burbujeo (false).**

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

Ejemplo:

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

¡¡Ojo!! No lleva "on" delante del nombre del evento.

Modificación de apariencia y comportamiento

Modelo de registro de eventos avanzado del W3C

Eliminar un evento de un elemento:

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

Crear más de un evento para el mismo elemento:

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

Crear un evento anónimo (no está asociado a ninguna función):

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

Modificación de apariencia y comportamiento

Modelo de registro de eventos avanzado según Microsoft

Utiliza un evento attachEvent() con dos argumentos:

- **Tipo de evento.**
- **Función a ejecutar.**

¡¡Ojo!! En este modelo de registro de eventos se utiliza "on" delante del nombre.

```
elemento.attachEvent('evento', función);
```

Ej:
```
document.getElementById("mienlace").attachEvent('onclick', alertar);
```

Eliminar un evento de un elemento:

```
elemento.detachEvent("onclick", accion);
```

Crear más de un evento para el mismo elemento:

```
elemento.attachEvent("onclick", hacerUnaCosa);
elemento.attachEvent("onclick", hacerOtraCosa);
```

Modificación de apariencia y comportamiento

Modelo de registro de eventos avanzado según Microsoft

EJERCICIO - U4T2:

- Vas a crear un juego que consiste en encontrar parejas en 12 cartas con 6 parejas de los personajes de los Simpson. El juego consistirá en lo siguiente:
  - La aplicación deberá tener una tabla con 3 filas y cuatro columnas de un color. Además habrá un cuadro de texto con el valor 0 pero no modificable.
  - Cuando el usuario haga clic sobre una celda, se mostrará una imagen.
  - Cuando el usuario haga clic sobre otra celda, se mostrará otra imagen.
  - Si las dos imágenes son iguales, se cambiará el color de la celda y se añadirá 1 al cuadro de texto.
  - Si las dos imágenes son diferentes, se ocultarán mostrando nuevamente el color inicial.
- En primer lugar, crea un esquema en una hoja con el árbol DOM del documento html y en otra indica los métodos que vas a crear asociados a qué evento.
- Trata de utilizar el menor número de variables posibles: utiliza las propiedades de los elementos para cambiar su comportamiento.

Envío y validación de formularios

Validación de formularios

Debemos indicar, en la etiqueta del formulario, que realice la validación:

```
<form action="URL" method="post" name="formValidado" onsubmit="return validar()">
    <!-- … -->
</form>
```

Podemos hacerlo con un AddEventListener (mejor):

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

Envío y validación de formularios

Validación de formularios

En primer lugar definimos una función “iniciar” que se va a cargar únicamente cuando el documento haya sido cargado completamente.

```
window.onload = iniciar;
function iniciar() {
  document.getElementById("enviar").addEventListener("click", validar, false);
}
```

La función validar recibe el evento por defecto asociado al botón enviar. Validamos cada apartado llamando a las funciones correspondientes:

```
function validar(eventopordefecto) {
  if (
    validarcampostexto(this) && validarMatricula() &&validarProvincia() &&
    confirm("¿Deseas enviar el formulario?")
  )
    return true;
  else {
    //Cancelamos el envento de envío por defecto asignado al botón
    eventopordefecto.preventDefault();
    return false; // Salimos de la funcioń devolviendo false.
  }
}
```

Envío y validación de formularios

Validación de un campo como obligatorio

Valida un campo de texto:

```
function validacion() {
  valor = document.getElementById("campo").value;
  if (valor == "" || valor.length == 0) {
    alert("El campo no puede ser  vacío");
    return false;
  }
  return true;
}
```

Envío y validación de formularios

Validación de un campo como obligatorio

Valida todos los campos de texto del formulario:

```
var formulario = objeto.form;
for (var i = 0; i < formulario.elements.length; i++) {
  if (
    formulario.elements[i].type == "text" &&
    formulario.elements[i].value == ""
  ) {
    alert(
      "El campo:  " + formulario.elements[i].name + " no puede  estar en blanco"
    );
    formulario.elements[i].className = "error";
    formulario.elements[i].focus();
    return false;
  }
}
```

Envío y validación de formularios

Validación de un campo de texto como numérico

Ejemplo 1 (teléfono):

```
function validaNum() {
  valor = document.getElementById("telefono").value;
  if (isNaN(valor)) {
    alert("El campo tiene que ser  numérico");
    return false;
  }
  return true;
}
```

Ejemplo 2(edad):

```
if (formulario.elements[i].id == "edad") {
  if (
    isNaN(formulario.elements[i].value) ||
    formulario.elements[i].value < 0 ||
    formulario.elements[i].value > 105
  ) {
    alert(
      "El campo: " +
        formulario.elements[i].name +
        "  posee valores incorrectos o la edad <0 o  >105"
    );
  }
}
```

Envío y validación de formularios

Validación de un campo de fecha

```
function validaFecha() {
  vardia = document.getElementById("dia").value;
  varmes = document.getElementById("mes").value;
  varano = document.getElementById("ano").value;
  fecha = new Date(ano, mes, dia);
  if (!isNaN(fecha)) {
    return false;
  }
  return true;
}
```

Envío y validación de formularios

Modificación de la apariencia de un campo erróneo

Podemos modificar la apariencia de un campo creando en css una clase error:

```
.error {
  border: solid 2px #ff0000;
}
```

Así, cuando un campo sea erróneo, le aplicamos dicha clase. Ejemplo:

```
if (
  formulario.elements[i].type == "text" &&
  formulario.elements[i].value == ""
) {
  alert(
    "El campo: " + formulario.elements[i].name + " no puede estar en blanco"
  );
  formulario.elements[i].className = "error";
  formulario.elements[i].focus();
  return false;
}
```

Envío y validación de formularios

Validación de un checkbox

Ejemplo:

```
function validaCheck() {
  elemento = document.getElementById("campoCondiciones");
  if (!elemento.checked) {
    return false;
  }
  return true;
}
```

Envío y validación de formularios

EJERCICIO: U4T3

- A partir del formulario creado en la tarea U4T1, realiza las funciones necesarias para validarlo teniendo en cuenta:
  - Nombre del disco: 20 caracteres, obligatorio.
  - Grupo de música o cantante: 20 caracteres, obligatorio.
  - Año de publicación: 4 caracteres numéricos.
  - Tipo de música (podrá ser “rock”, “pop”, “punk” o “indie”): sin comprobación.
  - Localización: almacenará un número de estantería: vacío o numérico.
  - Prestado: sin comprobación.
- Tendrás que tener en cuenta, además:
  - Que utilizarás un método window.onload.
  - Que la validación se asignará al formulario mediante un AddEventListener.
  - Que los campos nombre del disco y grupo de música se validarán en la misma función (campo20).

Envío y validación de formularios

EJERCICIO: U4T4:

- **Modifica las funciones de tal manera que, en caso de que se produzca un error en la validación, el campo implicado tenga el reborde rojo, y la etiqueta que lo acompaña también aparezca de color rojo.**
- **Para ellodeberás crear las clases css necesarias (puedes incluirlas en el propio html si lo deseas) para que se visualicecorrectamente.**
- **Ten en cuenta que si el usuario mete correctamente el nombre, deberá volver a su color habitual.**

Expresiones Regulares

Expresiones regulares

Se utiliza en datos que siguen un patrón concreto.

Ejemplos de expresiones regulares:

- Código postal.
- Teléfono (sobre todo con prefijos...)
- DNI.
- Tarjeta de extranjería.
- Código de cuenta del banco.
- Correo electrónico.
- Identificadores de usuario.

Expresiones Regulares

Caracteres especiales

Car.	Descripción	Detalle
^	Principio de entrada o línea.	Este carácter indica que las cadenas deberán comenzar por el siguiente carácter. Si este fuera una “a” minúscula como indicamos en el punto anterior la expresión regular seria sería, ^a.
$	Fin de entrada olínea.	Indica que la cadena debe terminar por el elemento precedido al dólar.
*	El carácter anterior 0 o más veces.	* El carácter anterior 0 o más veces. El asterisco indica que el carácter anterior se puede repetir en la cadena 0 o más veces.
+	El carácter anterior 1 o más veces.	El símbolo más indica que el carácter anterior se puede repetir en la cadena una o más veces.
?	El carácter anterior una vez como máximo.	El símbolo interrogación indica que el carácter anterior se puede repetir en la cadena cero o una vez
.	Cualquier carácter individual.	El símbolo punto indica que puede haber cualquier carácter individual salvo el de salto de línea.

Expresiones Regulares

Caracteres especiales

Car.	Descripción	Detalle
x|y	X ó y	La barra vertical indica que puede ser el carácter x o el y.
{n}	n veces el carácter anterior	El carácter anterior a las llaves tiene que aparecer exactamente n veces.
{n,m}	Entre n y m veces el carácter anterior	El carácter anterior a las llaves tiene que aparecer como mínimo n y como máximo m veces.
[abc]	Cualquier carácter de los corchetes.	En la cadena puede aparecer cualquier carácter que este incluido en los corchetes.
[^abc]	Un carácter que no esté en los corchetes	En la cadena pueden aparecer todos los caracteres que no estén incluidos en los corchetes.

Expresiones Regulares

Caracteres especiales

Car.	Descripción	Detalle
\b	Fin de palabra	Este símbolo indica que tiene que haber un fin de palabra o retorno de carro.
\B	No fin de palabra	El símbolo \B indica cualquiera que no sea un límite de palabra.
\d	Cualquier carácter dígito	Este símboloindica que puede haber cualquier carácter numérico, de 0 a 9.
\D	Carácter que no es dígito	Este símbolo indica que puede haber cualquier carácter siempre que no sea numérico.
\f	Salto de página	Este símbolo indica que tiene que haber un salto de página
\n	Salto de línea	Este símbolo indica que tiene que haber un salto de línea.
\s	Cualquier espacio en blanco	Este símbolo indica que tiene que haber un carácter individual de espacio en blanco: espacios, tabulaciones, saltos de página o saltos delínea.
\S	Cualquier carácter que no sea blanco	Este símbolo indica que tiene que haber cualquier carácter individual que no sea un espacio en blanco.
\t	Cualquier tabulación	Este símbolo indica que tiene que haber cualquier tabulación.
\w	Cualquier alfanumérico	Este símbolo indica que puede haber cualquier carácter alfanumérico.
\W	Cualquier carácter no alfanumérico	Este símbolo indica que puede haber cualquier carácter que no sea alfanumérico.

Expresiones Regulares

Validación de un DNI

```
function validaDNI() {
  valor = document.getElementById("dni").value;
  varletras = [
    "T","R","W","A","G","M","Y","F","P","D","X","B",
    "N","J","Z","S","Q","V","H","L","C","K","E","T",
  ];
  if (!/^\d{8}[A-Z]$/.test(valor)) {
    return false;
  }
  if (valor.charAt(8) != letras[valor.substring(0, 8) % 23]) {
    return false;
  }
  return true;
}
```

Expresiones Regulares

Validación de un correo electrónico

```
function validaEmail() {
  valor = document.getElementById("campo").value;
  if (!/\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)/.test(valor)) {
    return false;
  }
  return true;
}
```

Expresiones Regulares

Validación de un teléfono

```
function validaTelefono() {
  valor = document.getElementById("telefono").value;
  if (!/^\d{9}$/.test(valor)) {
    return false;
  }
  return true;
}
```

Expresiones Regulares

EJERCICIO - U4T5

Walter White y Jesse Pinkman tienen que controlar las bolsas de producto azul que cocinan para distribuir en Alburquerque. Para ello tienen que crear un formulario que almacene la información de cada bolsa y validarlo teniendo en cuenta lo siguiente:

- **Fecha de creación: obligatorio y con formato dd/mm/aaaa.**
- **Cocinero: será un nombre en clave formado por dos letras en mayúscula, un símbolo y cuatro dígitos (ej. WW$1234)**
- **Destinatario: estará formado por dos o tres letras mayúsculas correspondientes al estado, un guión bajo, el nombre de la ciudad en minúsculas, dos puntos, y el código de distrito de 4 digitos (ej. NM_alburquerque:1234).**
- **Gramos: será un número del 100 al 5000.**
- **Composición: estará formado por una cantidad en gramos seguida de dos conjuntos de una o dos letras seguidas o no de un número. (ej. 200gC3OH7)**
- **Número de cuenta de EEUU: ver diapositiva siguiente:**

Expresiones Regulares

EJERCICIO - U4T5

Supongamos que un número de cuenta estadounidense tiene el siguiente formato:

- **Dos letras: suponemos que el valor de cada letra es del 1 al 26 (no hay ñ ni ll).**
- **Dos dígitos: debe corresponderse con la suma de la primera letra y la segunda: en caso de que sea menor que 10 se pone el 0 delante.**
- **Un guión.**
- **Doce dígitos de cuenta.**
- **Un guión.**
- **Dos dígitos de control: los dos primeros deben ser la suma de los 6 primeros dígitos de la cuenta dividido entre 6 y extrayendo solamente su parte entera; y los dos últimos exactamente igual, pero con los 6 siguientes.**
- **Si el número está correcto se colocará en un campo de texto al lado del anterior, pero sin guiones: solamente los números y las letras.**

Cookies

Cookie

- Fichero de texto propio de cada navegador.
- Se almacena en diferentes lugares según el navegador y el sistema operativo.
- Se comenzaron a utilizar para guardar la información de los carritos de compra.
- Son específicas del dominio: un dominio no puede acceder a la cookie de otro dominio.
- Tienen una fecha de caducidad: como el número de cookies es limitado, el propio navegador se encargar de borrarlas.

Cookies

Formato de registro de una cookie

- Dominio del servidor que creó la cookie.
- Información de si es necesaria una conexión http segura para acceder a la cook
- Trayectoria de las URL que podrán acceder a la cookie.
- Fecha de caducidad de la cookie.
- Nombre de una entrada de datos.
- Cadena de texto asociada a ese nombre de entrada de datos.

Cookies

¿Dónde se almacenan?

En Windows 7 y Vista:
"\Usuarios*‹Tu_usuario›*\AppData\Roaming\Mozilla\Firefox\Profiles\serie_de_números_y_letras.default".

En Unix/Linux en la carpeta: "~/.mozilla/".

En Mac OS X: "~/Library/Mozilla/".

Almacenamiento: "cookies.sqlite" (a partir de la versión 3 de Firefox). También os encontraréis con los archivos "cookies.sqlite-shm", "cookies.sqlite-wal" y "cookies-nontor.xml" para poder explorar (darle formato visual) la base de datos de cookies.

En Windows vista o 7: "\Usuarios*‹Tu_usuario›*\AppData\L ocal\Google\Chrome\User Data\Default\".

En Mac OS X: "~/Library/Preferences/Opera Preferences/cookies4.dat".

En UNIX y distros Linux: "~/.config/google-chrome/Default".

Almacenamiento: "cookies" sin extensión en formato base de datos.

Cookies

¿Dónde se almacenan?

Windows Vista y 7: "\Usuarios*‹Tu_ usuario›*\AppData\R oaming\Opera\Opera\".

Mac OS X: "~/Library/Cookies".

Unix y otras distros Linux: "~/.opera/".

Almacenamiento: "cookies4.dat" en formato binario.

Windows Vista y 7: "\Usuarios*‹Tu_ usuario›*\AppData\R oaming\Apple Computer\Safari\Cookies".

Mac OS X: "~/Library/Cookies".

Unix y otras distros Linux: no hay una versión oficial del navegador de Apple para Linux.

Almacenamiento: "Cookies.binarycookies" en formato binario.

En Windows 7 y Vista:
"\Usuarios*‹Tu_ usuario›*\AppData\Roaming\Microsoft\Windows\Cookies".
Almacenamiento: en formato texto, en archivos codificados con extensión .txt.

Cookies

Opciones de utilización

- Mantener opciones de visualización
- Almacenamiento de variables
- Seguimiento de la actividad del usuario
- Autenticación de usuarios

Cookies

Grabar una cookie:

```
document.cookie = "nombreCookie = datosCookie"
[; expires=horaFormatoGMT] // Ej. expires=01 Jan 70 00:00:00 GMT;
[; path=ruta] // Ruta actual de nuestra página web.
[; domain=nombreDominio] // Por defecto, el que creó la cookie.
[; secure] // Si no se pone, puede ser accesible por cualquier programa que se  ajuste al dominio y path
```

Borrar una cookie:

Se le asigna una fecha de caducidad anterior a la actual.

Ejemplos:

```
document.cookie = "usuario=Laura; expires=2 Dec 2014, 23:59:59 GMT";
document.cookie = "contador=0";
```

Si la cookie no existe, la crea; si existe, la sobreescribe.

Cookies

Función para inicializar una cookie:

```
function setCookie(cname, cvalue, exdays) {
  let d = new Date(); // Se extrae la fecha actual
  //Se le suman los días  indicados por parámetro
  d.setTime(d.getTime() + exdays * 24 * 60 * 60 * 1000);
  let expires = "expires=" + d.toUTCString(); // Se pasa a formato GMT
  document.cookie = cname + "=" + cvalue + "; " + expires; //Se crea la cookie
}
```

Cookies

Leer una cookie:

```
// Devuelve toda la cadena con todas las variables;
let x = document.cookie;
```

Leer un valor concreto de la cookie:

```
function getCookie(cname) {
  var name = cname + "="; // Crea una variable con el texto del campo y un igual
  var ca = document.cookie.split(";"); // Divide la cookie en trozos separados por ;
  for (var i = 0; i < ca.length; i++) {
    // Recorre cada trozo de la cookie
    var c = ca[i];
    while (c.charAt(0) == " ") c = c.substring(1);
    if (c.indexOf(name) == 0)
      // Si encuentra el campo buscado
      return c.substring(name.length, c.length); // lo devuelve
  }
  return ""; // Sino devuelve vacío
}
```

Cookies

Comprobar si una cookie está inicializada:

```
function checkCookie() {
  var username = getCookie("username"); // Se extrae el valor (ver función  anterior)
  if (username != "") {
    // Si no está vacío
    alert("Bienvenido de nuevo, " + username); // Se le saluda
  } else {
    username = prompt("Introduzca sunombre:", ""); // Se le solicita con prompt
    if (username != "" && username != null) {
      setCookie("username", username, 365); // Se inicializa la variable
    }
  }
}
```

Cookies

Comprobar si una cookie está inicializada:

EJERCICIO – U4T6:

- **Crea un fichero que se llame "cookies.js" y que permita trabajar con las cookies de manera que puedas crear, borrar y consultar una cookie. Consulta los métodos tratados en clase y recuerda que el método borrar no ha sido creado aún así que deberás crearlo tú mismo.**
- **Modifica el archivo de la tarea anterior (U4T5) de la siguiente manera:**
  - **Cada vez que el usuario trate de enviar el formulario y haya algún error, una variable contador (almacenada en una cookie) se incrementará.**
  - **El resultado del número de intentos se reflejará en un campo de texto que se encontrará al final del formulario.**
  - **Si el usuario sale del programa y vuelve a entrar, el campo de texto mostrará el número almacenado en la cookie.**
  - **Junto al campo de texto habrá un botón que, al pulsarlo, permitirá reiniciar el valor de la cookie a 0.**

END

prof.jduran@iesalixar.org