

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN
CICLO: 02	<p style="text-align: center;"><i>GUIA DE LABORATORIO #7</i></p> Nombre de la Práctica: Control de eventos con JavaScript Lugar de Ejecución: Centro de Cómputo MATERIA: Desarrollo de Aplicaciones Web con Software Interpretado en el Cliente

I. OBJETIVOS

Que el estudiante:

1. Conozca el modelo básico de control de eventos en JavaScript
2. Realice vinculación de eventos al documento web a través de atributos HTML
3. Efectúe vinculación de eventos al documento web utilizando JavaScript
4. Utilice el modelo de eventos del DOM2 para captura de eventos en un documento

II. INTRODUCCION TEORICA

Control de eventos

Los eventos son controlados por un modelo de eventos. Este modelo de eventos es parte del Modelo de Objetos de Documento y se encarga de determinar la forma en que los scripts deben procesar dichos eventos cuando sucedan.

Los eventos pueden ser movimientos del puntero del ratón, pulsación de los botones, pulsaciones de teclas en el teclado o arrastrar un objeto por la pantalla.

En el control de los eventos a través de un navegador web estándar participan varias tecnologías:

El navegador: que determina qué tipo de eventos soporta y cómo debe gestionarlos.

El lenguaje HTML: que proporciona el marco de trabajo de los objetos del documento a los que están vinculados los eventos.

El lenguaje JavaScript: que se puede utilizar para establecer, manipular y responder a las acciones soportadas por el navegador.

Los modelos de eventos proporcionados por los navegadores han evolucionado de forma incompatible, de modo que el consorcio W3C incluyó en el DOM nivel 2 un modelo estándar de eventos.

Modelo básico de eventos

Podemos definir un **evento** como una acción destacada que ocurre dentro del navegador y a la que se puede responder con ejecutando un guión o *script*. Un **manejador de eventos** es código de JavaScript asociado a una parte específica del documento y a un evento en particular. El código de JavaScript se ejecuta si ocurre el evento y exactamente cuando este se produce en la parte del documento al que está asociado.

El proceso de control de eventos en el modelo básico de eventos, común a todos los navegadores, es directo. Los manejadores de eventos se asignan a partes del documento utilizando HTML o JavaScript y cuando ocurre un evento, el navegador revisa la parte pertinente de la jerarquía del documento en busca del manejador apropiado. Si existe uno, se ejecuta, si no, no sucede nada.

Los objetos en los que ocurren los eventos son, por lo general: enlaces, botones, formularios e imágenes. Sin embargo, también lo puede ser el documento mismo, o cualquier elemento HTML como pueden ser párrafos, listas o tablas. Hay que mencionar que como todo lo demás en JavaScript, no todos los modelos soportan todos los eventos para cada elemento de la página.

Vinculación de eventos en HTML

HTML soporta vinculación de eventos nucleares para la mayor parte de los elementos. Esta vinculación se realiza a través de atributos para los elementos, como *onclick*, *onmouseover*, *onmouseout*, *onload*, *onchange*, etc que toman como valor código de JavaScript que se ejecutará cuando el evento dado ocurra en ese objeto.

Veamos el siguiente ejemplo, en el que se realiza una vinculación a través de un manejador Click para un enlace:

```
<a href="http://www.w3c.org/" onclick="alert('Conectando con las oficinas centrales del DOM')">El DOM del consorcio W3C</a>
```

La mayor parte de los atributos de eventos HTML suponen una interacción simple del usuario, como hacer clic en un botón del ratón o pulsar una tecla. Otros elementos, como los campos de formularios tienen algunos eventos especiales asociados a ellos como indicar el campo que ha recibido el foco, el envío del formulario, el cambio de opción en un campo de menú de selección, etc.

La siguiente tabla muestra una lista de los eventos fundamentales del modelo básico de eventos:

Atributo de evento	Descripción de evento	Elementos que lo admiten
onblur	Ocurre cuando un elemento pierde el foco, lo que significa que el usuario ha movido el foco hacia otro elemento, generalmente haciendo clic con el ratón sobre él o moviéndose hasta él con el tabulador.	<code><a></code> , <code><area></code> , <code><button></code> , <code><input></code> , <code><label></code> , <code><select></code> , <code><textarea></code> , <code><div></code> , <code><embed></code> , <code><hr></code> , <code></code> , <code><marquee></code> , <code><object></code> , <code><applet></code> , <code></code> , <code><table></code> , <code><tr></code> , <code><td></code>
onchange	Indica que el campo de formulario ha perdido el foco del usuario y su valor se ha modificado durante el último acceso.	<code><input></code> , <code><select></code> , <code><textarea></code>
onclick	Indica que se ha hecho clic en el elemento.	La mayoría de los elementos de visualización: botones, enlaces, campos de formulario, en general.
ondblclick	Indica que se ha hecho doble clic en el elemento.	La mayoría de los elementos de visualización: botones, enlaces, campos de formulario, en general.
onfocus	Indica que el elemento ha recibido el foco; concretamente, que se ha seleccionado para manejarlo o para la entrada de datos.	<code><a></code> , <code><area></code> , <code><button></code> , <code><input></code> , <code><label></code> , <code><select></code> , <code><textarea></code> , <code><div></code> , <code><embed></code> , <code></code> , <code><hr></code> , <code><marquee></code> , <code><object></code> , <code></code> , <code><table></code> , <code><tr></code> , <code><td></code> , <code><body></code>
onkeydown	Indica que una tecla se está pulsando con foco en el formulario.	La mayoría de elementos de visualización: botones, enlaces, campos de formulario, en general.
onkeypress	Describe el evento de pulsar y liberar una tecla con foco en el elemento.	La mayoría de elementos de visualización: botones, enlaces, campos de formulario, en general.
onkeyup	Indica que una tecla se deja de pulsar estando el foco en el elemento.	La mayoría de elementos de visualización: botones, enlaces, campos de formulario, en general.
onload	Indica el evento de que una ventana o un conjunto de marcos terminen de cargar de un documento u otro fichero.	<code><body></code> , <code><frameset></code> , <code><embed></code> , <code><link></code> , <code><script></code> , <code><style></code> , <code><applet></code> , <code></code> , <code><layer></code>

onmousedown	Indica la pulsación de un botón del ratón con foco en el elemento.	La mayoría de los elementos de visualización: botones, enlaces, campos de formulario, etc.
onmousemove	Indica que el ratón se ha movido mientras estaba sobre el elemento.	La mayoría de los elementos de visualización: botones, enlaces, campos de formulario, etc.
onmouseout	Indica que el ratón se ha desplazado fuera de un elemento.	La mayoría de los elementos de visualización: botones, enlaces, campos de formulario, etc.
onmouseover	Indica que el ratón se ha desplazado sobre un elemento.	La mayoría de los elementos de visualización: botones, enlaces, campos de formulario, etc.
onmouseup	Indica que ha sido liberado un botón del ratón con foco en el elemento.	La mayoría de los elementos de visualización: botones, enlaces, campos de formulario, etc.
onreset	Indica que el formulario ha sido restablecido o borrado, posiblemente por la pulsación de un botón Restablecer (reset).	<form>
onselect	Indica la selección de texto por el usuario, generalmente resaltando el texto deseado.	<input>, <textarea>
onsubmit	Indica que el formulario está a punto de ser enviado, generalmente como resultado de activar un botón de envío.	<form>
onunload	Indica que el navegador está abandonando el documento actual y descargándolo de la ventana o marco.	<body>, <frameset>

El valor de un atributo de controlador de evento es una cadena arbitraria de código JavaScript, tomando en cuenta que si el controlador consiste de múltiples instrucciones JavaScript, estas deben separarse entre sí con punto y coma. Veamos algunos ejemplos:

```
<input type="button" value="Hazme clic" onclick="if(window.numclicks) numclicks++; else numclicks=1; this.value='Click #' + numclicks;" />
```

Cuando un controlador de eventos requiere múltiples instrucciones, resulta más conveniente definir las en el cuerpo de una función (o método si está utilizando un enfoque completamente orientado a objetos) y utilizar el atributo del controlador de eventos HTML para llamar a dicha función. Por ejemplo, para validar una entrada del usuario a un campo de formulario antes de enviar dicho formulario al servidor, se puede utilizar el atributo `onsubmit` del elemento FORM. Suponiendo que hemos creado una función de validación denominada `validateForm()` para realizar la validación:

```
<form name="myForm" id="myForm" action="validar.php" onsubmit="return validateForm();" >
  <!-- Campos del formulario -->
</form>
```

Vinculación de eventos en JavaScript

Muchas veces resulta más ventajoso utilizar JavaScript en lugar de vincular manejadores de eventos a partes de un documento utilizando atributos de eventos con HTML. Esto se debe a las características más avanzadas del modelo de eventos. Otra ventaja añadida de la utilización de JavaScript es la separación entre la estructura y la lógica del documento y su presentación.

Hay que tener en cuenta que en JavaScript se accede a los manejadores de eventos como métodos del objeto al que están vinculados.

Por ejemplo, para establecer el manejador clic en un botón de formulario se le asigna a su propiedad `onclick` el código que se desea activar:

```
<form name="myForm" id="myForm">
```

```
<input type="button" name="myButton" id="myButton" value="Hacer clic aquí">
</form>
<script language="JavaScript" type="text/javascript">
<!--
    document.myForm.myButton.onclick = new Function("alert('Gracias por pulsarme');");
//-->
</script>
```

Activación de eventos de forma manual

Se pueden invocar tanto manejadores de eventos, como eventos de forma manual. Debe tenerse cuidado de invocarlos de forma correcta respetando las letras mayúsculas o minúsculas. No hay que olvidar que JavaScript distingue entre mayúsculas y minúsculas y si se invoca una propiedad de instancia desconocida provocará un error en tiempo de ejecución.

Existen métodos para manejadores de eventos como `onmouseover()`, `onclick()` y también existen métodos para activar eventos como `submit()` y `reset()`.

Veamos el siguiente ejemplo:

```
<form name="myForm" id="myForm" onsubmit="return confirm('Fin en los datos del formulario')"
method="GET" action="http://www.pint.com">
<input type="button" value="Botón regular (tipo button) con método submit"
onclick="document.myform.submit()" "><br>
<input type="submit" value="Botón enviar (tipo submit) real">
</form>
```

La diferencia entre activar el método **submit()** con el evento `onclick` de un botón `button`, comparado con la utilización del botón tipo *submit* es que el primero efectúa el envío inmediatamente, en tanto que el segundo activa primero el evento *submit* y luego hace el envío propiamente dicho.

Otro aspecto importante que poseen los manejadores de eventos es que se pueden utilizar sus valores de retorno para modificar el comportamiento predeterminado del evento. Como se ve a continuación:

```
<a href="http://www.w3c.org" onclick="return confirm('Ir al sitio de la W3C')">W3C</a>
```

La siguiente tabla muestra valores comunes de retorno de eventos y sus efectos:

Evento	Valor de retorno	Efecto
Click	False	Botones de opción y casillas de verificación. Ninguno se establece. Botones de envío: el envío del formulario se cancela. Botones de restablecer, el formulario no se restablece. Enlaces, el enlace no se sigue.
DragDrop	False	Arrastrar y cargar se cancela.
KeyDown	False	Cancela los eventos de KeyPress que siguen (mientras el usuario mantiene la tecla pulsada).
KeyPress	False	Cancela el evento KeyPress.
MouseDown	False	Cancela la acción predeterminada (el comienzo de un arrastre, el comienzo del modo de selección o el montaje de un enlace).
MouseOver	True	Hace que cualquier cambio en las propiedades del estado (<i>status</i>) o del estado predeterminado (<i>defaultStatus</i>) de la ventana se refleje en el navegador.
MouseOver	False	Hace que algunos cambios del estado (<i>status</i>) o del estado predeterminado (<i>defaultStatus</i>) se ignoren hasta que ocurra <i>MouseOut</i>
Submit	False	Cancela el envío del formulario.

Modelo de eventos del DOM2

El consorcio W3C proporciona un modelo de eventos del DOM2 que intenta normalizar con un mínimo divergencias la interacción de los scripts con documentos estructurados de forma correcta, brindando una

interfaz independiente del lenguaje y de la plataforma. Para conseguir la funcionalidad de este modelo se requiere de un Modelo de Objetos de Documento que cumpla con la especificación del DOM de la W3C.

Los eventos empiezan su ciclo de vida en la parte superior de la jerarquía del documento y van abriéndose camino descendiendo hasta el objeto destino. Esto se conoce como la fase de captura. Durante su descenso, el evento puede ser preprocesado, controlado o dirigido por cualquier objeto participante. Una vez que ha alcanzado su objeto destino y el manejador lo haya ejecutado, el objeto comienza su camino de regreso ascendiendo por la jerarquía. Esto se conoce como fase de ascenso. Los eventos ascienden de forma predeterminada, pero se deben capturar explícitamente en su camino descendente de forma semejante a la de los navegadores Netscape 4+.

El DOM nivel 2 define un objeto *Event* que proporciona información relevante a los manejadores. Este objeto es creado automáticamente por el navegador y debe ser pasado a las funciones que van a constituirse en los manejadores de evento. Las diferencias con las que cada navegador incorpora propiedades y métodos para este objeto *event* deben ser consideradas por el desarrollador. La siguiente tabla muestra algunas propiedades relevantes de este objeto.

Propiedad	Descripción
altKey	Valor lógico que indica si la tecla <i>alt</i> ha sido pulsada
bubbles	Valor lógico que indica si el evento asciende
button	Valor numérico que indica el botón de ratón presionado (originalmente es 0, pero varía de sistema a sistema)
cancelable	Valor lógico que indica si el evento no debería ascender por la jerarquía
charCode	El valor ASCII de la tecla presionada durante eventos de pulsación de tecla
clientX	Valor numérico que indica la coordenada horizontal del evento
clienteY	Valor numérico que indica la coordenada vertical del evento
ctrlKey	Valor lógico que indica si la tecla CTRL se ha pulsado
currentTarget	Nodo al que está asignado el manejador de eventos
shiftKey	Valor lógico que indica si la tecla SHIFT se ha pulsado
target	Nodo en el cual ha ocurrido el evento
type	Cadena que contiene el tipo de evento (por ejemplo "click")

Vinculación de controladores de eventos de forma compatible

A pesar de las diferencias en el manejo de eventos entre navegadores que se apegan a DOM Nivel 2 y el Internet Explorer, versiones anteriores a la 9.0. Es posible desarrollar código de JavaScript portable.

```
if (document.addEventListener) {
    this.addEvent = function (elem, type, fn) {
        elem.addEventListener(type, fn, false);
        return fn;
    };
    this.removeEvent = function (elem, type, fn) {
        elem.removeEventListener(type, fn, false);
    };
}
else if (document.attachEvent) {
    this.addEvent = function (elem, type, fn) {
        var bound = function () {
            return fn.apply(elem, arguments);
        };
        elem.attachEvent("on" + type, bound);
        return bound;
    };
    this.removeEvent = function (elem, type, fn) {
        elem.detachEvent("on" + type, fn);
    };
}

/* Probar el controlador de eventos compatible */
addEvent(window, "load", function () {
    var elems = document.getElementsByTagName("div");
```

```
for (var i = 0; i < elems.length; i++)
  (function (elem) {
    var handler = addEvent(elem, "click", function () {
      this.style.backgroundColor =
        this.style.backgroundColor==' ' ? 'green' : ' ';
      removeEvent(elem, "click", handler);
    });
  })(elems[i]);
//Fin del for()
});
```

Módulos, interfaces y tipos de eventos

El DOM Nivel 2 está organizado en una serie de módulos, de modo que proporciona una estructura básica para el control de los eventos.

La admisión de determinados tipos de eventos se manifiesta en submódulos, proporcionando cada uno de ellos compatibilidad con una categoría de tipos de eventos relacionados y define un tipo de evento que se pasa a los controladores de eventos.

En la siguiente tabla se muestra una lista de módulos, interfaces y tipos de eventos admitidos por cada módulo:

Módulo	Interfaz	Tipos de eventos
HTMLEvents	Event	abort, blur, change, error, focus, load, reset, resize, scroll, select, submit, unload.
MouseEvents	MouseEvent	click, mousedown, mousemove, mouseout, mouseover, mouseup
UIEvents	UIEvent	DOMActivate, DOMFocusIn, DOMFocusOut

Gestión del objeto Event

El objeto Event es un objeto que crea automáticamente el navegador cuando ocurren eventos del teclado y del ratón. El objeto es pasado también de forma automática a controlador del evento definido como una función u objeto que lo gestiona.

El manejo de este objeto es incompatible entre los navegadores, sobre todo en el Internet Explorer 8.0 e inferiores. De modo que el tratamiento de este objeto debe hacerse de formas particulares para obtener la máxima compatibilidad y portabilidad.

El objeto Event contiene una serie de propiedades que son de utilidad para realizar acciones cuando se producen eventos del ratón o del teclado. Estos eventos son los de mayor relevancia cuando se produce la interacción del usuario con los elementos de la página web.

Ejemplo de manejo del objeto Event en un evento onclick

```
function eventoClick(ev) {
  var texto = "Evento onclick en el navegador " + navigator.appName + "\n";
  //Si existe el objeto event lo utiliza, en caso contrario,
  //usa un objeto event pasado por parámetro
  if(window.event){
    texto += "Usa el objeto event\n";
    for(i in event){
      texto += i + " = " + ev[i] + "\n";
    }
  }
  else {
    texto += "Usa objeto ev pasado por parámetro\n";
    for(i in ev){
      texto += i + " = " + ev[i] + "\n";
    }
  }
  alert(texto);
}
document.onclick = eventoClick;
```

Eventos del ratón y del teclado

En navegadores Firefox, Chrome, Safari, Opera y las nuevas versiones de Internet Explorer a partir de la 9.0 se utilizan las propiedades `pageX` y `pageY` para seguir la posición del ratón.

En Internet Explorer 8.0 y anteriores se utilizan las propiedades `offsetX` y `offsetY`.

Los valores de coordenadas son relativas a la propia ventana por lo que son relativas a la esquina superior izquierda del elemento BODY.

```
document.onmouseover = obtenerPosRaton;
var pX = 0;
var pY = 0;
function obtenerPosRaton(e) {
    if(document.all){
        //Detecta que el navegador es Internet Explorer 8.0 y ant.
        pX = event.offsetX;
        pY = event.offsetY;
    }
    else {
        //Para el resto de navegadores
        pX = e.pageX;
        pY = e.pageY;
    }
    //Actualizar barra de título y de estado de la ventana
    document.title = "Coordenadas del ratón (x:" + pX + ", y:" + pY + ")";
    window.status = "Coordenadas del ratón (x:" + pX + ", y:" + pY + ")";
}
```

Además del ratón (mouse), el otro dispositivo de entrada, más importante, es el teclado.

JavaScript proporciona formas de manipular eventos del teclado.

Lastimosamente, los eventos y los controladores de eventos para el teclado son más incompatibles que el resto de eventos vistos anteriormente.

A pesar de esta dificultad, es posible escribir secuencias de comando que funcionen correctamente para los navegadores más utilizados.

Los eventos de teclado *keydown*, *keypress* y *keyup* se corresponden con los controladores de los eventos *onkeydown*, *onkeypress* y *onkeyup*.

Una pulsación normal de tecla por parte del usuario genera tres eventos, que son, en orden: *keydown*, cuando se comienza a presionar la tecla, *keypress*, cuando se mantiene presionada la tecla, y *keyup*, cuando se está soltando la tecla.

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Requerimiento	Cantidad
1	Guía de práctica #7: Control de eventos con JavaScript	1
2	Computadora con editor HTML instalado y navegadores	1
3	Memoria USB.	1

IV. PROCEDIMIENTO

Ejercicio #1: El siguiente ejemplo muestra cómo crear un visor de imágenes que permite ir pasando fotos con eventos de presión de teclas específicas del teclado. Una para avanzar y otra para retroceder. El código es portable con la mayor parte de navegadores modernos.

Guión 1: imageshow.html

```
<!DOCTYPE html>
<html lang="es">
<head>
```

```

<title>Visor de imágenes</title>
<meta charset="utf-8" />
<link rel="stylesheet" href="css/fonts.css" />
<!-- CSS only -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaIlGyVh/UjpbCx/TYkiZhlZB6+fzT" crossorigin="anonymous">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.9.1/font/bootstrap-icons.css">
<script src="js/eventosteclas.js"></script>
</head>
<body>
<section class="container pt-5 ">
<article class="d-flex justify-content-center">
<div class="bg-light p-3 border border rounded w-50">

    <h1 class="text-center">Visor de imagenes</h1>
    

    <p class="text-center">
        Pulse tecla [<-] para retroceder,<br />
        Pulse tecla [->] para avanzar.
    </p>
</div>
</article>
</section>
</body>
</html>

```

Guión 2: eventosteclas.js

```

document.onkeydown = keyHit;
var thisPic = 0;

function keyHit(evt){
    var myPix = new Array(
        "img/lion.jpg",
        "img/tigger.jpg",
        "img/puma.jpg",
        "img/leopard.jpg",
        "img/lince.jpg"
    );
    var imgCt = myPix.length - 1;
    //37 y 39 son los códigos de las teclas que representan
    //la presión de las teclas de cursor izquierda y derecha
    //respectivamente
    var ltArrow = 37;
    var rtArrow = 39;
    //Manejo del objeto para controlar los eventos del teclado
    //de forma uniforme sin importar el navegador
    var thisKey = (evt) ? evt.which : window.event.keyCode;
    if(thisKey == ltArrow){
        chgSlide(-1);
    }
    else if(thisKey == rtArrow){
        chgSlide(1);
    }
    return false;

    function chgSlide(direction){
        thisPic = thisPic + direction;
        if(thisPic > imgCt){
            thisPic = 0;
        }
    }
}

```



```

    }
    if(thisPic < 0){
        thisPic = imgCt;
    }
    document.getElementById("myPicture").src = myPix[thisPic];
}
}
}

```

Resultado:



Ejemplo #2: El siguiente ejemplo muestra cómo crear un área de texto editable en un formulario controlada con eventos del teclado, mediante la pulsación combinada de las teclas CTRL+E para editar y CTRL+S para guardar el texto ingresado en un elemento div. Al inicio se muestra este elemento div y al pulsar las teclas CTRL+E se oculta y se cambia por el área de texto. Por otro lado, al presionar CTRL+S se oculta el área de texto y se vuelve a mostrar el elemento div con el texto ingresado.

Guión 1: comentarios.html

```

<!DOCTYPE html>
<html lang="es">

<head>
    <title>Área de texto editable</title>
    <meta charset="utf-8" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
    integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOMLASjC" crossorigin="anonymous">
    <script src="js/modernizr.custom.js"></script>
    <script src="js/keydown.js"></script>
</head>

<body class="container">
    <header>
        <h1 class="text-center">Área de edición de texto</h1>
    </header>
    <section>
        <article>
            <div id="wrap">
                <h1 class="text-center">Déjanos tus comentarios</h1>
                <div id='form_wrap' class="d-flex justify-content-center">
                    <form id="contact-form" class="bg-light p-3 border border rounded w-50"
                        action="javascript:alert('¡Comentario enviado!');">
                        <p id="formstatus"></p>
                        <p>
                            Instrucciones: CTRL+E para Editar y CTRL+S para Guardar,
                            ESC para Cancelar.
                        </p>
                        <div class="form-group">
                            <label for="email">Tu mensaje: </label>
                            <div id="view"></div>

```

```

        <textarea name="message" class="form-control" id="area"
value="Tu mensaje" id="message"></textarea>
    </div>
    <div class="form-group">
        <label for="name">Nombre: </label>
        <input type="text" name="name" class="form-control" value=""
id="name" />
    </div>
    <div class="form-group">
        <label for="email">Correo electrónico: </label>
        <input type="text" name="email" class="form-control" value=""
id="email" />
    </div>
    <br>
    <div class="form-group">
        <input type="submit" name="submit" class="btn btn-info form-
control" value="Enviar" />
    </div>
</form>
</div>
</div>
<div id="fact">
</div>
</div>
</article>
</section>
</body>

</html>

```

Guión 2: keydown.js

```

window.onload = init;

function init() {
    var view = document.getElementById('view');
    var area = document.getElementById('area');

    view.onclick = edit;

    document.onkeydown = function(e) {
        e = e || event;
        // Escape
        if(e.keyCode == 27) {
            cancel();
            return false;
        }
        if ((e.ctrlKey && e.keyCode == 'E'.charCodeAt(0)) && !area.offsetHeight) {
            edit();
            return false;
        }
        if((e.ctrlKey && e.keyCode == 'S'.charCodeAt(0)) && area.offsetHeight) {
            save();
            return false;
        }
    }

    function edit() {
        //Ocultar el elemento div
        view.style.display = 'none';
        //Dibujar el campo textarea y ponerle estilos
        area.value = view.innerHTML;
        area.style.display = 'block';
        area.style.height = '80px';
        area.style.padding = '6px';
    }
}

```

```

        area.style.width = '444px';
        area.focus();
    }

    function save() {
        area.style.display = 'none';
        view.innerHTML = area.value;
        view.style.display = 'block';
        view.style.letterSpacing = '1.2px';
    }

    function cancel() {
        area.style.display = 'none';
        view.style.display = 'block';
    }
}

```

El resultado al cargar la página en un navegador reciente sería el siguiente:

Área de edición de texto

Déjanos tus comentarios

Instrucciones: CTRL+E para Editar y CTRL+S para Guardar, ESC para Cancelar.

Tu mensaje:

Nombre:

Correo electrónico:

Enviar

Al editar con CTRL+E

Instrucciones: CTRL+E para Editar y CTRL+S para Guardar, ESC para Cancelar.

Tu mensaje:

Yo os las entrego tales como son, en su frescor de carne y de rosa. Sólo existe un método honrado y lógico de traducción: la «literalidad», una literalidad

Al presionar CTRL+S

Instrucciones: CTRL+E para Editar y CTRL+S para Guardar, ESC para Cancelar.

Tu mensaje:

Yo os las entrego tales como son, en su frescor de carne y de rosa. Sólo existe un método honrado y lógico de traducción: la «literalidad», una literalidad impersonal, apenas atenuada por un leve parpadeo y una ligera sonrisa del traductor. Ella crea, sugestiva, la más grande potencia literaria. Ella produce el placer de la evocación. Ella es la garantía de la verdad. Ella es firme e inmutable, en su desnudez de piedra. Ella cautiva el aroma primitivo y lo cristaliza. Ella separa y desata... Ella fija.

Ejercicio #3: El siguiente ejemplo, muestra cómo hacer captura de todo tipo de eventos relacionados con campos de formulario. Los eventos capturados son mostrados en un campo de área de texto justo en el momento que se producen, sin sobrescribir los que ya han capturados previamente.

Guión 1: camposformulario.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Controladores de eventos para elementos de formulario</title>
  <meta charset="utf-8" />
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
  integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWFspD3yD65VohhpUuCOMLASjC" crossorigin="anonymous">
</head>
<body class="container">
  <h1 class="text-center">Evetos de controles formulario</h1>
  <!-- Formulario con todos los tipos de campos de formulario -->
  <div id="formulario" class="d-flex justify-content-center">
    <form name="everything" class="bg-light p-3 border border rounded w-50">
      <ul class="inputelements">
        <li>
          [1]<input type="text" name="username" placeholder="Usuario"
class="form-control" />
        </li>
        <li>
          [2]<input type="password" name="password" placeholder="Contraseña"
class="form-control" />
        </li>
        <li>
          [3]<input type="file" name="file" placeholder="Elija un archivo"
class="form-control" />
        </li>
      </ul>
      <ul class="chkelements">
        <li>
          <label for="extras">Periféricos de la computadora:</label>
          [4]<input type="checkbox" name="extras" value="quemador" class="form-
check-input" />
          <label>Quemador de CD/DVD</label>
        </li>
        <li>
          [4]<input type="checkbox" name="extras" value="impresor" class="form-
check-input" />
          <label>Impresor</label>
        </li>
        <li>
          [4]<input type="checkbox" name="extras" value="parlantes" class="form-
check-input" />
          <label>Parlantes</label>
        </li>
        <li>
          [4]<input type="checkbox" name="extras" value="inalambrica"
class="form-check-input" />
          <label>Tarjeta de red inalámbrica</label>
        </li>
        <li>
          [4]<input type="checkbox" name="extras" value="mouse optico"
class="form-check-input" />
          <label>Mouse óptico</label>
        </li>
      </ul>
    </form>
  </div>
</body>
</html>
```

```

        </ul>
        <ul class="radelements">

            <label for="browser">Mi navegador preferido:</label>

            <li>
                [5]<input type="radio" name="browser" class="form-check-input"
value="firefox">
                <label>Mozilla firefox</label>
            </li>
            <li>
                [5]<input type="radio" name="browser" class="form-check-input"
value="iexplorer">
                <label>Internet Explorer</label>
            </li>
            <li>
                [5]<input type="radio" name="browser" class="form-check-input"
value="netscape">
                <label>Maxthon</label>
            </li>
            <li>
                [5]<input type="radio" class="form-check-input" name="browser"
value="opera">
                <label>Opera</label>
            </li>
            <li>
                [5]<input type="radio" class="form-check-input" name="browser"
value="safari">
                <label>Safari</label>
            </li>
            <li>
                [5]<input type="radio" class="form-check-input" name="browser"
value="chrome">
                <label>Chrome</label>
            </li>
        </ul>
        <ul class="hobbies">
            <label for="hobbies">Mis pasatiempos</label>

            <li>
                [6]<select name="hobbies" class="form-control" size="6" multiple>
JavaScript</option>
                <option value="programar" selected="selected">Programar con
                <option value="musica">Escuchar música</option>
                <option value="cine">Ir al cine</option>
                <option value="playa">Ir a la playa</option>
                <option value="leer">Leer obras</option>
                <option value="surfear">Surfear</option>
                <option value="bailar">Ir a bailar</option>
                <option value="viajar">Ir de viaje</option>
            </select>
        </li>
        </ul>
        <ul class="colors">
            <label for="color">Color favorito:</label>
            <li>
                [7]<select name="color" class="form-control">
                <option value="red" selected="selected">Rojo</option>
                <option value="blue">Azul</option>
                <option value="yellow">Amarillo</option>
                <option value="purple">Morado</option>
                <option value="orange">Naranja</option>
                <option value="pink">Rosado</option>
            </li>
        </ul>
    
```

```

        <option value="green">Verde</option>
        <option value="gray">Gris</option>
        <option value="black">Negro</option>
        <option value="white">Blanco</option>
    </select>
</li>
</ul>
<ul class="events">
    <li>
        [8]<textarea      name="textarea"      class="form-control      h-100"
placeholder="Eventos capturados"></textarea>
    </li>
</ul>
<ul class="buttons">
    <li>
        [9] <input type="button" value="Limpiar" class="btn btn-info form-
control" name="clearbutton" />
    </li>
    <br>
    <li>
        [10]<input type="submit" value="Enviar" class="btn btn-info form-
control" name="submitbutton" />
    </li>
    <br>
    <li>
        [11]<input type="reset" value="Restablecer" class="btn btn-info form-
control" name="resetbutton" />
    </li>
</ul>
</form>
</div>
<br>
<!-- Tabla con la descripción de los campos -->
<div id="tabla">
    <table class="table table-dark">
        <tr >
            <th colspan="6" class="text-center">Elementos de formulario</th>
        </tr>
        <tr>
            <td>[1]Text</td>
            <td>[2]Password</td>
            <td>[3]File</td>
            <td>[4]Checkbox</td>
            <td>[5]Radio</td>
        </tr>
        <tr>
            <td>[6]Select list</td>
            <td>[7]Select menu</td>
            <td>[8]Textarea</td>
            <td>[9]Button</td>
            <td>[10]Submit</td>
        </tr>
        <tr>
            <td colspan="5" class="text-center">[11]Reset</td>
        </tr>
    </table>
</div>
<script type="text/javascript" src="js/eventreport.js"></script>
</body>
</html>

```

Guión 2: eventreport.js

```
//Esta función añade detalles de un controlador al elemento
//Textarea del formulario. Se llama desde diversos
//controladores de eventos.
function report(element, event){
    if((element.type == "select-one") || (element.type == "select-multiple")){
        value = "";
        for(var i=0; i<element.options.length; i++){
            if(element.options[i].selected){
                value += element.options[i].value + " ";
            }
        }
    }
    else if(element.type == "textarea") value = "...";
    else value = element.value;
    var msg = event + ": " + element.name + " (" + value + ")\n";
    var t = element.form.textarea;
    t.value += msg;
}

function addHandlers(f){
    //Recorrer con un lazo todos los controles del formulario
    for(var i=0; i<f.elements.length; i++){
        var e = f.elements[i];
        e.onclick = function(){report(this, 'Click');}
        e.onchange = function(){report(this, 'Change');}
        e.onfocus = function(){report(this, 'Focus');}
        e.onblur = function(){report(this, 'Blur');}
        e.onselect = function(){report(this, 'Select')}
    }

    //Definir algún controlador de evento especial para tres botones
    f.clearbutton.onclick = function(){
        this.form.textarea.value = "";
        report(this, 'Click');
    }
    f.submitbutton.onclick = function(){
        report(this, 'Click');
        return false;
    }
    f.resetbutton.onclick = function(){
        this.form.reset();
        report(this, 'Click');
        return false;
    }
}

//Activar el formulario añadiendo todos los controladores de eventos
addHandlers(document.everything);
```

Resultado:

Evetos de controles formulario

- [1]
- [2]
- [3]
- Periféricos de la computadora:
 - [4] ☐ Quemador de CD/DVD
 - [4] ☐ Impresor
 - [4] ☐ Parlantes
 - [4] ☐ Tarjeta de red inalámbrica
 - [4] ☐ Mouse óptico
- Mi navegador preferido:
 - [5] ☐ Mozilla firefox
 - [5] ☐ Internet Explorer
 - [5] ☐ Maxthon
 - [5] ☐ Opera
 - [5] ☐ Safari
 - [5] ☐ Chrome
- Mis pasatiempos
 - [6]

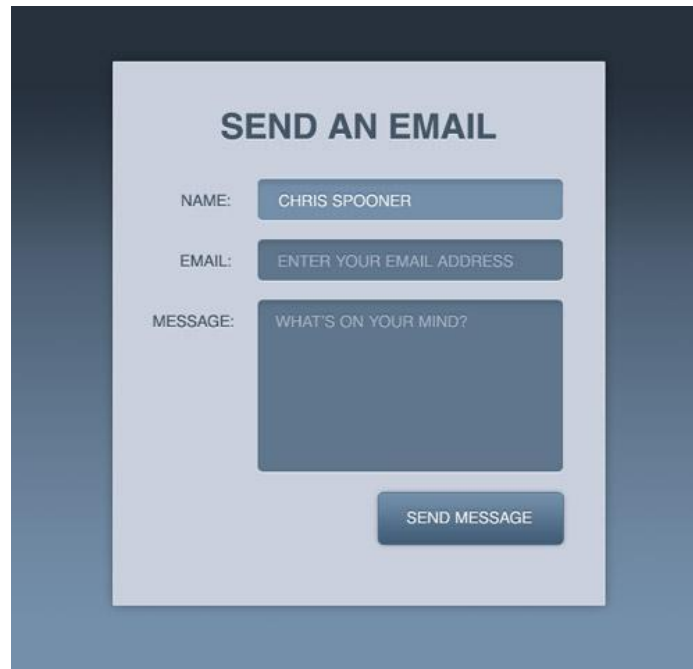
Programar con JavaScript

Escuchar música

Ir al cine

V. DISCUSION DE RESULTADOS

1. Realice un visor de imágenes con imágenes propias (utilice 10 imágenes) que haga uso tanto del método de avance y retroceso con botones de formulario (adelante y atrás) y con teclado (avanzar y retroceder con las teclas de flecha de cursor). Al avanzar hacia adelante en la secuencia de imágenes cuando se llegue a la última imagen de la secuencia el comportamiento debe ser mostrar la primera y al retroceder en la secuencia, al llegar a la primera, el comportamiento debe ser mostrar la última. Esto es lo que se hace en el ejemplo de avance de fotos con teclado, pero ahora debe implementarlo también en el avance y retroceso con botones de formulario.
2. Realice un formulario de contacto como el que se muestra en la imagen al final del enunciado de este ejercicio, y con JavaScript no obstrusivo (unobtrusive) realice el código necesario con para validar cada uno de los campos de entrada. En el campo nombre, únicamente deben permitirse caracteres alfabéticos, en el campo correo electrónico caracteres que puedan formar una dirección de correo electrónico válido, tomando en cuenta que el signo de arroba (@) solamente puede ingresarse una sola vez y el campo mensaje, pueden ingresarse distintos tipos de caracteres, incluyendo símbolos gramaticales.



The image shows a web form titled "SEND AN EMAIL" in a bold, dark blue font. The form is set against a light blue background with a dark blue border. It contains three input fields: "NAME:" with the text "CHRIS SPOONER", "EMAIL:" with the placeholder "ENTER YOUR EMAIL ADDRESS", and "MESSAGE:" with the placeholder "WHAT'S ON YOUR MIND?". Below these fields is a "SEND MESSAGE" button. The form is centered on a dark blue background.

VI. BIBLIOGRAFIA

- Flanagan, David. JavaScript La Guía Definitiva. 1ª Edición. Editorial ANAYA Multimedia. 2007. Madrid, España.
- Terry McNavage. JavaScript Edición 2012 Manual Imprescindible. 1ª Edición. Editorial ANAYA Multimedia / Apress. 2012. Madrid, España.
- Deitel, Paul / Deitel, Harvey / Deitel, Abbey. Internet & World Wide Web. Cómo programar. 5ª Edición. Editorial Pearson. 2014. México D.F.
- Tom Negrito / Dori Smith. JavaScript y AJAX para diseño web. Editorial Pearson Prentice Hall. Madrid, España. 2007.
- Powell, Thomas / Schneider, Fritz. JavaScript Manual de Referencia. 1ra Edición. Editorial McGraw-Hill. 2002. Madrid, España.