Materia: Tecnologías para la Web.
Tema: Modelo de objetos de JavaScript.

1. Introducción

- Los eventos de JavaScript permiten que las secuencias de comandos respondan a las interacciones del usuario y modifiquen la página según sea necesario.
- Los eventos y el manejo de eventos ayudan a que las aplicaciones Web sean más responsivas, dinámicas e interactivas.

2. Repaso del evento `load`

- Las funciones que manejan eventos se llaman manejadores de eventos. Al proceso de asignar un manejador de eventos a un evento en un nodo DOM se le conoce como registrar un manejador de eventos.
- El evento `load` se dispara cada vez que un elemento termina de cargarse con éxito.
- Si una secuencia de comandos en el elemento `head` intenta obtener un nodo DOM para un elemento de HTML5 en el elemento `body`, `getElementById` devuelve `null` debido a que el elemento `body` aún no se ha cargado.
- El método `addEventListener` puede invocarse varias veces en un nodo DOM para registrar más de un método de manejo de eventos para un evento.
- Es posible quitar un componente de escucha de eventos mediante una llamada a `removeEventListener`, con los mismos argumentos que se le pasaron a `addEventListener` para registrar ese manejador de eventos.
- El modelo en línea del registro de eventos coloca las llamadas a las funciones de JavaScript directamente en el código de HTML.
- El modelo tradicional de registro de eventos usa una propiedad de un objeto para especificar un manejador de eventos.

EJEMPLO 1.

Archivo load.html:

```
<!-- Demostración del evento load. -->
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Evento load</title>
		<link rel = "stylesheet" type = "text/css" href = "estilo.css">
		<script src = "load.js" ></script>
	</head>
	<body>
		<p>Segundos que ha estado viendo esta p&aacute;gina hasta ahora:
		<span id = "hastaAhora">0</span></p>
	</body>
</html>
```

Archivo load.js:

```
// Secuencia de comandos para demostrar el evento load.
var segundos = 0;
// se invoca al cargarse la página para iniciar el temporizador
function iniciarTemporizador(){
	window.setInterval( "actualizarTiempo()", 1000 );
} // fin de la función iniciarTemporizador
// se invoca cada 1000 ms para actualizar el temporizador
function actualizarTiempo(){
	++segundos;
	document.getElementById( "hastaAhora" ).innerHTML = segundos;
} // fin de la función actualizarTiempo
window.addEventListener( "load", iniciarTemporizador, false );
```

3. El evento `mousemove` y el objeto `event`

- El `mousemove event` se dispara cada vez que el usuario mueve el ratón.
- El objeto `event` almacena información sobre el evento que llamó a la función manejadora de eventos.
- La propiedad `ctrlKey` del objeto `event` contiene un valor booleano que refleja si se presionó la tecla Ctrl durante el evento.
- La propiedad `shifKey` del objeto `event` refleja si se presionó la tecla Mayús durante el evento.
- En una función manejadora de eventos, `this` hace referencia al objeto DOM en el que ocurrió el evento.
- El objeto `event` almacena en su propiedad `target` el nodo en el que ocurrió la acción.

Propiedad	Descripción
`altKey`	Este valor es `true` si se presionó la tecla Alt cuando se disparó el evento.
`cancelBubble`	Se establece en `true` para evitar que burbujee el evento. El valor predeterminado es `false`.
`clientX` `clientY`	Las coordenadas del cursor del ratón dentro del área cliente (es decir, el área activa en donde se visualiza la página Web, excluyendo barras de desplazamiento, botones de navegación, etc.).
`ctrlKey`	Este valor es `true` si se presionó la tecla Ctrl cuando se disparó el evento.
`keyCode`	El código ASCII de la tecla presionada en el evento de teclado.
`screenX`, `screenY`	Las coordenadas del cursor del ratón en el sistema de coordenadas de la pantalla.
`shiftKey`	Este valor es `true` si se presionó la tecla Mayús cuando se disparó el evento.
`target`	El objeto del DOM que recibió el evento.
`type`	El nombre del evento que se disparó.

EJEMPLO 2.

Archivo dibujar.html:

```
<!-- Programa simple de dibujo. -->
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Programa simple de dibujo</title>
		<link rel = "stylesheet" type = "text/css" href = "estilo.css">
		<script src = "dibujar.js"></script>
	</head>
	<body>
		<table id = "lienzo">
			<caption>Mantenga presionada <em>Ctrl</em> (o <em>Control</em>) para dibujar en azul.
				Mantenga presionada <em>May&uacute;s</em> para dibujar en rojo.</caption>
			<tbody id = "cuerpotabla"></tbody>
		</table>
	</body>
</html>
```

Archivo dibujar.js:

```
// Programa simple de dibujo.
// función de inicialización para insertar celdas en la tabla
function crearLienzo(){
	var lado = 100;
	var tbody = document.getElementById( "cuerpotabla" );
	for ( var i = 0; i < lado; ++i ){
		var fila = document.createElement( "tr" );
```

```
            for ( var j = 0; j < lado; ++j ){
                  var celda = document.createElement( "td" );
                  fila.appendChild( celda );
            } // fin de for
            tbody.appendChild( fila );
      } // fin de for
      // registrar componente de escucha de mousemove para la tabla
      document.getElementById( "lienzo" ).addEventListener(
            "mousemove", procesarMouseMove, false );
} // fin de la función crearLienzo
// procesa el evento onmousemove
function procesarMouseMove( e ){
      if ( e.target.tagName.toLowerCase() == "td" ){
            // colorea la celda de azul si está presionada la tecla Ctrl
            if ( e.ctrlKey ){
                  e.target.setAttribute( "class", "blue" );
            } // fin de if
            // colorea la celda de rojo si está presionada la tecla Mayús
            if ( e.shiftKey ){
                  e.target.setAttribute( "class", "red" );
            } // fin de if
      } // fin de if
} // fin de la función procesarMouseMove
window.addEventListener( "load", crearLienzo, false );
```

Archivo estilo.css:

```
/* CSS para dibujar.html. */
#canvas { width: 400px;
          border: 1px solid #999999;
          border-collapse: collapse }
td      { width: 4px; height: 4px; margin: 0px; padding: 0px; }
.blue   { background-color: blue; }
.red    { background-color: red; }
```

4. Sustituciones con `mouseover` y `mouseout`

- Cuando el cursor del ratón entra en un elemento, ocurre un evento `mouseover` para ese elemento. Cuando el cursor del ratón sale del elemento, ocurre un evento `mouseout` para ese elemento.
- Al crear un objeto `Image` y establecer su propiedad `src` se realiza una carga previa de la imagen.

EJEMPLO 3.

Archivo mouseoverout.html:

```
<!-- Los eventos mouseover y mouseout. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title>Eventos mouseover y mouseout</title>
            <link rel = "stylesheet" type = "text/css" href = "estilo.css">
```

```
            <script src = "mouseoverout.js" ></script>
      </head>
      <body>
            <h1><img src = "encabezado1.png" id = "encabezado"
                  alt = "Imagen de encabezado"></h1>
            <p>&iquest;Es posible distinguir un color a partir de su c&oacute;digo
                  RGB hexadecimal? Vea el c&oacute;digo hex y adivine su color. Para ver
                  a qu&eacute; color corresponde, mueva el rat&oacute;n sobre el
                  c&oacute;digo hex. Si quita el rat&oacute;n de la celda de la tabla de
                  c&oacute;digo hex se mostrar&aacute; el nombre del color.</p>
            <div>
                  <ul>
                        <li id = "Black">#000000</li>
                        <li id = "Blue">#0000FF</li>
                        <li id = "Magenta">#FF00FF</li>
                        <li id = "Gray">#808080</li>
                        <li id = "Green">#008000</li>
                        <li id = "Lime">#00FF00</li>
                        <li id = "Maroon">#800000</li>
                        <li id = "Navy">#000080</li >
                        <li id = "Olive">#808000</li>
                        <li id = "Purple">#800080</li>
                        <li id = "Red">#FF0000</li>
                        <li id = "Silver">#C0C0C0</li>
                        <li id = "Cyan">#00FFFF</li>
                        <li id = "Teal">#008080</li>
                        <li id = "Yellow">#FFFF00</li>
                        <li id = "White">#FFFFFF</li>
                  </ul>
            </div>
      </body>
</html>
```

Archivo mouseoverout.js:

```
// Los eventos mouseover y mouseout.
imagen1 = new Image();
imagen1.src = "encabezado1.png";
imagen2 = new Image();
imagen2.src = "encabezado2.png";
function mouseOver( e ){
      // intercambia la imagen cuando se mueve el ratón sobre ella
      if ( e.target.getAttribute( "id" ) == "encabezado" ){
            e.target.setAttribute( "src", imagen2.getAttribute( "src" ) );
      } // fin de if
      // si el elemento es li, asigna su id a su color para
      // cambiar el texto del código hex al color correspondiente
      if ( e.target.tagName.toLowerCase() == "li" ){
            e.target.setAttribute( "style",
            "color: " + e.target.getAttribute( "id" ) );
      } // fin de if
} // fin de la función mouseOver
function mouseOut( e ){
      // regresar la imagen original al quitar el ratón
      if ( e.target.getAttribute( "id" ) == "encabezado" ){
            e.target.setAttribute( "src", imagen1.getAttribute( "src" ) );
      } // fin de if
      // si el elemento es li, asigna su Id a innerHTML
```

```
      // para mostrar el nombre del color
      if ( e.target.tagName.toLowerCase() = "li" ){
            e.target.innerHTML = e.target.getAttribute( "id" );
      } // fin de if
} // fin de la función mouseOut
document.addEventListener( "mouseover", mouseOver, false );
document.addEventListener( "mouseout", mouseOut, false );
```

Archivo estilo.css:

```
/* CSS for onmouseoverout.html. */
body  { background-color: wheat }
div   { border-style: groove;
            text-align: center;
            font-family: monospace;
            font-weight: bold;
            width: 34em; }
ul    { margin-left: -40px; }
li    { display: inline-block; width: 8em; }
```

Mover el mouse sobre cada código de color.

5. Procesamiento de formularios con `focus` y `blur`

- El evento `focus` se dispara cuando un elemento obtiene el enfoque (es decir, cuando el usuario hace clic en un campo de formulario o usa la tecla Tab para moverse entre un elemento y otro del formulario).
- `blur` se dispara cuando un elemento pierde el enfoque, lo que ocurre cuando otro control obtiene el enfoque.

EJEMPLO 4.

Archivo focusblur.html:

```
<!DOCTYPE html>
<!-- Demonstración de los eventos focus y blur. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>Formulario con focus y blur</title>
      <link rel = "stylesheet" type = "text/css" href = "estilo.css">
      <script src = "focusblur.js"></script>
   </head>
   <body>
      <form id = "myForm" action = "">
         <p><label class = "fixed" for = "name">Nombre:</label>
            <input type = "text" id =  "name"
               placeholder = "Ingresar nombre"></p>
         <p><label class = "fixed" for = "email">E-mail:</label>
            <input type = "email" id = "email"
               placeholder = "Ingresar e-mail"></p>
         <p><label>Habilitar si le gusta este sitio
            <input type = "checkbox" id =  "like"></label></p>
         <p><label for = "comments">Comentarios:</p>
```

```
         <textarea id =  "comments"
            placeholder = "Ingresar commentarios"></textarea>
         <p><input id = "submit" type = "submit">
            <input id = "reset" type = "reset"></p>
      </form>
      <p id = "helpText"></p>
   </body>
</html>
```

Archivo focusblur.js:

```
// Demonstración de los eventos focus y blur.
var helpArray = [ "Ingresar su nombre aqu&iacute;.",
  "Ingresar su e-mail con el formato usuario@dominio.",
  "Habilitar esta caja si le gusta este sitio.",
  "Ingresar comentarios acerca de este sitio.",
  "Este bot&oacute;n env&iacute;a el formulario al script del servidor web.",
  "Este bot&oacute;n limpia los campos del formulario.", "" ];
var helpText;

// initialize helpTextDiv and register event handlers
function init(){
   helpText = document.getElementById( "helpText" );
   // register listeners
   registerListeners(document.getElementById( "name" ), 0 );
   registerListeners(document.getElementById( "email" ), 1 );
   registerListeners(document.getElementById( "like" ), 2 );
   registerListeners(document.getElementById( "comments" ), 3 );
   registerListeners(document.getElementById( "submit" ), 4 );
   registerListeners(document.getElementById( "reset" ), 5 );
} // end function init
// utility function to help register events
function registerListeners( object, messageNumber ){
   object.addEventListener( "focus",
      function() { helpText.innerHTML = helpArray[ messageNumber ]; },
      false );
   object.addEventListener( "blur",
      function() { helpText.innerHTML = helpArray[ 6 ]; }, false );
} // end function registerListener
window.addEventListener( "load", init, false );
```

Archivo estilo.css:

```
/* CSS for onfocusblur.html. */
textarea    { width: 300px; height: 100px; }
label.fixed { width: 4em; float: left; }
#helpText   { font-family: sans-serif; color: blue; margin-top: 10px; }
p           { margin: 0; }
```

Campos vacíos y sin mensaje.

Algunos campos llenos y mensaje del campo comentarios.

6. **Más procesamiento de formularios con `submit` y `reset`**

- Los eventos `submit` y `reset` se disparan cuando se envía o reinicia un formulario, respectivamente.
- Una función anónima es una función que se define sin nombre; se crea casi de la misma forma que cualquier otra función, sólo que sin identificador después de la palabra clave `function`.
- Las funciones anónimas son útiles al crear una función con el único propósito de asignarla a un manejador de eventos.
- El método `confirm` hace una pregunta a los usuarios, les presenta un botón Aceptar y un botón Cancelar. Si el usuario hace clic en Aceptar, `confirm` devuelve `true`; de lo contrario `confirm` devuelve `false`.
- Al devolver `true` o `false`, los manejadores de eventos indican si se va a realizar la acción predeterminada para el evento.
- Si un manejador de eventos devuelve `true` o no devuelve un valor, la acción predeterminada se realiza una vez que el manejador de eventos termina su ejecución.

EJEMPLO 5.
Utilizar los archivos focusblur.html y estilo.css del EJEMPLO 4.
Archivo submitreset.js:

```
// Demostración de los eventos focus y blur.
var arregloAyuda = [ "Escriba su nombre en este cuadro de entrada.",
      "Escriba su direcci&oacute;n de e-mail en el formato usuario@dominio.",
      "Marque esta casilla si le gust&oacute; nuestro sitio.",
      "Escriba aqu&iacute; los comentarlos para que nosotros los leamos.",
      "Este bot&oacute;n env&iacute;a el formulario a la secuencia de comandos del lado del
servidor.",
      "Este bot&oacute;n borra el formulario.", "" ];
var textoAyuda;
      // inicializar textoAyudaDiv y registrar los manejadores de eventos
function inic(){
      textoAyuda = document.getElementById( "textoAyuda" );
      // registrar componentes de escucha
      registrarEscuchas(document.getElementById( "nombre" ), 0 );
      registrarEscuchas(document.getElementById( "email" ), 1 );
      registrarEscuchas(document.getElementById( "gusta" ), 2 );
      registrarEscuchas(document.getElementById( "comentarios" ), 3 );
      registrarEscuchas(document.getElementById( "enviar" ), 4 );
      registrarEscuchas(document.getElementById( "reiniciar" ), 5 );
      var miFormulario = document.getElementById( "miFormulario" );
      miFormulario.addEventListener( "submit",
            function(){
                  return confirm("¿Seguro que desea enviar el formulario?");
            }, // fin de la función anónima
            false );
      miFormulario.addEventListener( "reset",
            function(){
                  return confirm("¿Seguro que desea reiniciar el formulario?");
            }, // fin de la función anónima
            false );
} // fin de la función inic
// función utilitaria para ayudar a registrar eventos
function registrarEscuchas( objeto, numeroMensaje ){
      objeto.addEventListener( "focus",
            function() { textoAyuda.innerHTML = arregloAyuda[ numeroMensaje ]; },
            false);
      objeto.addEventListener( "blur",
            function(){ textoAyuda.innerHTML = arregloAyuda[ 6 ]; }, false );
} // fin de la función registrarEscuchas
window.addEventListener( "load", inic, false );
```

7. Burbujeo de eventos

- El burbujeo de eventos es el proceso en el que los eventos disparados en elementos hijos "burbujean" hacia sus elementos padres. Cuando se dispara un evento en un elemento, primero se entrega al manejador de eventos del elemento (si lo hay) y después al manejador de eventos del elemento padre (si lo hay).
- Si usted intenta manejar un evento sólo en un elemento hijo, debe cancelar el burbujeo del evento en el código de manejo de eventos del elemento hijo mediante el uso de la propiedad `cancelBubble` del objeto `event`.

Evento	Descripción
`abort`	Se dispara cuando el usuario interrumpe una transferencia de imagen.
`change`	Se dispara cuando se realera una nueva elección en un elemento `select` o cuando cambia la entrada de texto y el elemento pierde el enfoque.
`click`	Se dispara cuando el usuario hace clic en el ratón.
`dblclick`	Se dispara cuando el usuario hace doble clic en el ratón.
`focus`	Se dispara cuando un elemento de formulario obtiene el enfoque.
`keydown`	Se dispara cuando el usuario presiona una tecla.
`keypress`	Se dispara cuando el usuario presiona y suelta una tecla.
`keyup`	Se dispara cuando el usuario suelta una tecla.
`load`	Se dispara cuando se cargan: un elemento y todos sus hijos.
`mousedown`	Se dispara al presionar un botón del ratón.
`mousemove`	Se dispara al mover el ratón.
`mouseout`	Se dispara cuando el ratón sale de un elemento.
`mouseover`	Se dispara cuando el ratón entra a un elemento.
`mouseup`	Se dispara al soltar un botón del ratón.
`reset`	Se dispara al reiniciar un formulario (es decir, cuando el usuario hace clic en un botón para reiniciar).
`resize`	Se dispara cuando cambia el tamaño de un objeto (es decir, cuando el usuario cambia el tamaño de una ventana o un marco).
`select`	Se dispara al comenzar una selección de texto (se aplica a `input` o `textarea`).
`submit`	Se dispara al enviar un formulario.
`unload`	Se dispara cuando una página está a punto de descargarse.

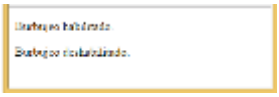
EJEMPLO 6.

Archivo burbujeo.html:

```
<!-- Cancelación del burbujeo de eventos. -->
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Burbujeo de eventos</title>
		<script src = "burbujeo.js"></script>
	</head>
	<body>hola
		<p id = "burbujeo">Burbujeo habilitado.</p>
		<p id = "sinBurbujeo">Burbujeo deshabilitado.</p>
	</body>
</html>
```

Archivo burbujeo.js:

```
// Cancelación del burbujeo de eventos.
function clicDocumento(){
      alert( "Hizo clic en el documento." );
} // fin de la función clicDocumento
function burbujeo( e ){
      alert( "Esto va a burbujear." );
      e.cancelBubble = false;
} // fin de la función burbujeo
function sinBurbujeo( e ){
      alert( "Esto no va a burbujear." );
      e.cancelBubble = true;
} // fin de la función sinBurbujeo
function registrarEventos(){
      document.addEventListener( "click", clicDocumento, false );
      document.getElementById( "burbujeo" ).addEventListener(
            "click", burbujeo, false );
      document.getElementById( "sinBurbujeo" ).addEventListener(
            "click", sinBurbujeo, false );
} // fin de la función registrarEventos
window.addEventListener( "load", registrarEventos, false );
```



(1) Digitar en el primer párrafo.

(2) Clic en Aceptar.

(3) Clic en Aceptar.

(4) Digitar en el segundo párrafo.

(5) Clic en Aceptar.