1. Eventos

En la programación tradicional, las aplicaciones se ejecutan secuencialmente de principio a fin para producir sus resultados. Sin embargo, existen lenguajes de programación que permiten la programación basada en eventos. Los scripts y programas esperan sin realizar ninguna tarea hasta que se produzca un evento. Una vez producido, ejecutan alguna tarea asociada a la aparición de ese evento y cuando concluye, el script o programa vuelve al estado de espera.

Los eventos permiten la interacción entre las aplicaciones JavaScript y los usuarios, cada vez que se pulsa un botón se produce un evento, cada vez que se pulsa una tecla, cada vez que se carga una página…

1.1. Tipos de eventos

Cada elemento HTML tiene definida su propia lista de posibles eventos que se le pueden asignar. Un mismo tipo de evento (por ejemplo, pinchar el botón izquierdo del ratón) puede estar definido para varios elementos HTML y un mismo elemento HTML puede tener asociados diferentes eventos.

El nombre de los eventos se construye mediante el prefijo `on`, seguido del nombre en inglés de la acción asociada al evento. Así, el evento de pinchar un elemento con el ratón se denomina `onclick` y el evento asociado a la acción de mover el ratón se denomina `onmousemove`.

La siguiente tabla resume los eventos más importantes definidos por JavaScript:

Evento	Descripción	Elementos para los que está definido
onblur	Un elemento pierde el foco	<button>, <input>, <label>, <select>, <textarea>, <body>
onchange	Un elemento ha sido modificado	<input>, <select>, <textarea>
onclick	Pulsar y soltar el ratón	Todos los elementos
ondblclick	Pulsar dos veces seguidas con el ratón	Todos los elementos
onfocus	Un elemento obtiene el foco	<button>, <input>, <label>, <select>, <textarea>, <body>
onkeydown	Pulsar una tecla y no soltarla	Elementos de formulario y <body>
onkeypress	Pulsar una tecla	Elementos de formulario y <body>
onkeyup	Soltar una tecla pulsada	Elementos de formulario y <body>
onload	Página cargada completamente	<body>
onmousedown	Pulsar un botón del ratón y no soltarlo	Todos los elementos
onmousemove	Mover el ratón	Todos los elementos
onmouseout	El ratón "sale" del elemento	Todos los elementos
onmouseover	El ratón "entra" en el elemento	Todos los elementos
onmouseup	Soltar el botón del ratón	Todos los elementos
onreset	Inicializar el formulario	<form>
onresize	Modificar el tamaño de la ventana	<body>
onselect	Seleccionar un texto	<input>, <textarea>
onsubmit	Enviar el formulario	<form>
onunload	Se abandona la página, por ejemplo al cerrar el navegador	<body>

Los eventos más utilizados en las aplicaciones web tradicionales son `onload` para esperar a que se cargue la página por completo, los eventos `onclick`, `onmouseover`, `onmouseout` para controlar el ratón y `onsubmit` para controlar el envío de los formularios.

1.2. Manejadores de eventos

Para que un evento de JavaScript resulte útil, se debe de asociar un código (script) a cada evento.

1.2.1. Manejadores como atributos HTML

Es el método más sencillo, se introduce un atributo en el propio elemento HTML.

Ejemplo

```
<input type="button" value="Púlsame" onclick="window.alert('Gracias por pul
  sar');" />
```

Realiza un div en html del tamaño que quieras que al pasar el ratón por encima aparezca en el navegador en un cuadro de diálogo el texto, "estás sobrevolando territorio protegido"

1.2.2. Manejadores de eventos y variable this

JavaScript define una variable especial llamada ***this*** que se crea automáticamente y que se emplea en algunas técnicas avanzadas de programación. En los eventos, se puede utilizar la variable this para referirse al elemento HTML que ha provocado el evento. Esta variable es muy útil para ejemplos como el siguiente:

Cuando el usuario pasa el ratón por encima del , el color del borde se muestra de color negro. Cuando el ratón sale del , se vuelve a mostrar el borde con el color gris claro original.

```
<div id="contenidos" style="width:150px; height:60px; border:thin solid
  silver">
  Sección de contenidos...
</div>
```

Si no se utiliza la variable `this`, el código necesario para modificar el color de los bordes, sería el siguiente:

```
<div id="contenidos" style="width:150px; height:60px; border:thin solid
  silver"
  onmouseover="document.getElementById('contenidos').style.borderColor='bl
  ack';"
  onmouseout="document.getElementById('contenidos').style.borderColor='sil
  ver';">
    Sección de contenidos...
</div>
```

El código anterior es demasiado largo y demasiado propenso a cometer errores. Dentro del código de un evento, JavaScript crea automáticamente la variable `this`, que hace referencia al elemento HTML que ha provocado el evento. Así, el ejemplo anterior se puede reescribir de la siguiente manera:

```
<div id="contenidos" style="width:150px; height:60px; border:thin solid
  silver" onmouseover="this.style.borderColor='black';"
  onmouseout="this.style.borderColor='silver';">
```

Sección de contenidos...

El código anterior es mucho más compacto, más fácil de leer y de escribir y sigue funcionando correctamente, aunque se modifique el valor del atributo id del .

Realiza un programa html que cuando pase el ratón sobre un div, éste cambie el color de fondo a rosa y cuando el ratón este fuera vuelva a su color normal. (Ayuda la propiedad es backgroundColor) ¿Puedes hacer que cambie de tamaño?

1.3. Manejadores de eventos como funciones externas

Cuando el código de la función manejadora es más complejo, como por ejemplo la validación de un formulario, es aconsejable agrupar todo el código JavaScript en una función externa que se invoca desde el código XHTML cuando se produce el evento.

De esta forma, el siguiente ejemplo:

```
<input type="button" value="Púlsame" onclick="window.alert('Gracias por pul
  sar');" />
```

Se puede transformar en:

```
function muestraMensaje() {
  window.alert('Gracias por pulsar');
}
<input type="button" value="Púlsame" onclick="muestraMensaje()" />
```

En las funciones externas no es posible utilizar la variable `this` de la misma forma que en los manejadores insertados en los atributos XHTML. Por tanto, es necesario pasar la variable `this` como parámetro a la función manejadora:

Es decir:

```
   <script>
            function cambiarColor(elemento) {
                elemento.style.backgroundColor = "red";
            }

            function cambiarColor2(elemento) {
                elemento.style.backgroundColor = "white";
            }
        </script>
        <div onmouseover="cambiarColor(this)" onmouseout="cambiarColor2(thi
  s)">Hola mundo
        </div>
```

1.3.1. Manejadores de eventos semánticos

Como hemos visto en la primera evaluación, al crear páginas web se recomienda separar los contenidos (HTML) de la presentación (CSS). También se recomienda separar los contenidos (HTML) de la programación (JavaScript) para no complicar excesivamente el código fuente de la página lo que dificulta su mantenimiento, así el primer ejemplo del botón se puede transformar en:

```
<input id="boton" type="button" value="Púlsame" />


<script>
    function muestraMensaje() {
        window.alert('Gracias por pulsar');
    }
    document.getElementById("boton").onclick = muestraMensaje;
</script>
```

El código HTML resultante es muy "limpio" porque no se mezcla con el código JavaScript. La técnica de los manejadores semánticos consiste en:

1. Asignar un identificador único al elemento HTML mediante el atributo id.
2. Crear una función de JavaScript encargada de manejar el evento.
3. Asignar la función a un evento concreto del elemento HTML mediante DOM.

El problema de utilizar este método es que tenemos que asegurarnos de que la página se ha cargado completamente o no se ejecutará, para ello podemos hacer uso de otro evento, onload.

```
window.onload = function() {
  document.getElementById("boton").onclick = muestraMensaje;
}
```