Tratamiento de los eventos en Javascript

1. Eventos definidos directamente en la marca HTML.

Esta metodología está en desuso aunque podemos seguir encontrando scripts que la usan.

Problema

Implementar un formulario que solicite la carga del nombre de usuario y su clave. Mostrar un mensaje si no se ingresan datos en alguno de los dos controles.

```
<html>
<head>

<script type="text/javascript">

  function validar()
  {
     var usu=document.getElementById("usuario").value;
     var cla=document.getElementById("clave").value;
     if (usu.length==0 || cla.length==0)
     {
       alert('El nombre de usuario o clave está vacío');
       return false;
     }
     else
       return true;
  }

</script>

</head>
<body>

<form method="post" action="procesar.php" onsubmit=" return validar();"
id="formulario1">
Ingrese nombre:
<input type="text" id="usuario" name="usuario" size="20">
<br>
Ingrese clave:
<input type="password" id="clave" name="clave" size="20">
<br>
<input type="submit" id="confirmar" name="confirmar" value="Confirmar">
</form>

</body>
</html>
```

El formulario usa la propiedad onsubmit y le asigna el nombre de la función JavaScript que debe llamarse previo a que el navegador empaquete todos los datos del formulario y los envíe al servidor para ser procesados.

Si la función retorna true los datos del formulario son enviados por al servidor.

Ejercicio

1. Implementar un formulario que solicite la carga del nombre, la edad y el mail de una persona. Cuando el control input tome foco borrar el contenido actual, al abandonar el mismo, mostrar un mensaje de alerta si el input está vacío. Mostrar en las propiedades value de los controles text los mensajes "nombre",edad y "mail" respectivamente.

   Un vez conseguido esto validar el formulario usando una función Javascript

2. Eventos definidos accediendo a propiedades del objeto.

Ejemplo

Implementar un formulario que solicite la carga del nombre de usuario y su clave. Mostrar un mensaje si no se ingresan datos en alguno de los dos controles.

```
<html>
<head>

<script type="text/javascript">

  window.onload=inicio;

  function inicio()
  {
    document.getElementById("formulario1").onsubmit=validar;
  }

  function validar()
  {
     var usu=document.getElementById("usuario").value;
     var cla=document.getElementById("clave").value;
     if (usu.length==0 || cla.length==0)
     {
       alert('El nombre de usuario o clave está vacío');
       return false;
     }
     else
       return true;
  }

</script>

</head>
<body>

<form method="post" action="procesar.php" id="formulario1">
Ingrese nombre:
<input type="text" id="usuario" name="usuario" size="20">
<br>
Ingrese clave:
<input type="password" id="clave" name="clave" size="20">
<br>
<input type="submit" id="confirmar" name="confirmar" value="Confirmar">
</form>

</body>
</html>
```

Con esta segunda metodología vemos que el código HTML queda limpio de llamadas a funciones JavaScript y todo el código queda dentro del bloque del script (pudiendo luego llevar todo este bloque a un archivo externo *.js)

Lo primero que vemos es inicializar la propiedad onload del objeto window con el nombre de la función que se ejecutará cuando finalice la carga completa de la página, es importante notar que a la propiedad onload le asignamos el nombre de la función y NO debemos disponer los paréntesis abiertos y cerrados (ya que no se está llamando a la función sino le estamos pasando la dirección o referencia de la misma)

```
  window.onload=inicio;
```

La misma metodología pero **utilizando funciones anónimas** para cada evento el código ahora queda condensado:

```
<html>
<head>

<script type="text/javascript">

  window.onload=function() {
    document.getElementById("formulario1").onsubmit=function () {
      var usu=document.getElementById("usuario").value;
      var cla=document.getElementById("clave").value;
      if (usu.length==0 || cla.length==0)
      {
        alert('El nombre de usuario o clave está vacío');
        return false;
      }
      else
        return true;
    }
  }

</script>

</head>
<body>

<form method="post" action="procesar.php" id="formulario1">
Ingrese nombre:
<input type="text" id="usuario" name="usuario" size="20">
<br>
Ingrese clave:
<input type="password" id="clave" name="clave" size="20">
<br>
<input type="submit" id="confirmar" name="confirmar" value="Confirmar">
</form>

</body>
</html>
```

Ejercicio

2. Crear una tabla cuyas filas cambien de color cuando se pasa sobre ellas

3 . Modelo de eventos definidos por W3C (World Wide Web Consortium)

Este modelo de eventos se basa en la implementación de un método para todos los objetos que nos permite registrar eventos. La sintaxis del método es:

```
addEventListener(evento, método a ejecutar,[fase]);
```

Veamos como implementamos el problema anterior utilizando este nuevo modelo de eventos:

```
<html>
<head>

<script type="text/javascript">

  window.addEventListener('load',inicio);

  function inicio()
  {
     document.getElementById("formulario1").addEventListener('submit',val
     idar,false);
  }

  function validar(evt)
  {
     var usu=document.getElementById("usuario").value;
     var cla=document.getElementById("clave").value;
     if (usu.length==0 || cla.length==0)
     {
       alert('El nombre de usuario o clave está vacío');
       evt.preventDefault();
     }
  }

</script>

</head>
<body>

<form method="post" action="procesar.php" id="formulario1">
Ingrese nombre:
<input type="text" id="usuario" name="usuario" size="20">
<br>
Ingrese clave:
<input type="password" id="clave" name="clave" size="20">
<br>
<input type="submit" id="confirmar" name="confirmar" value="Confirmar">
</form>

</body>
</html>
```

Lo primero que vemos es que en vez de inicializar la propiedad onload procedemos a llamar al método addEventListener:

```
  window.addEventListener('load',inicio);
```

El primer parámetro es un string con el nombre del evento a inicializar, el segundo parámetro es el nombre de la función a ejecutar y el tercer parámetro (opcional) normalmente es el valor false.

Ejemplo

1. Realizar un script que muestre en un div el número 2. Cada vez que se hace doble clic sobre el mismo duplicar el valor contenido en el div.
2. Script que muestre un cuadrado, el mismo mostrará los bordes redondeados cuando introduzcamos la flecha del mouse en su interior y volverá al estado anterior cuando retiremos la flecha.