IES de Teis
Avda. de Galicia, 101
36216 – Vigo

Tfno: 886 12 04 64
e-mail: ies.teis@edu.xunta.es
http://www.iesteis.es

PRÁCTICA 4: INTERFACES GRAFICAS DE USUARIO – JAVASCRIPT AVANZADO

Duración estimada: 150 min.

Objetivos:

- **Consolidar los conocimientos de javaScript.**
- **Trabajar con objetos y métodos (Objeto window, form, navigation, location, …)**
- **Eventos en JavaScript.**

Introducción:

La orientación de objetos se centra en identificar los objetos procedentes del dominio de la aplicación que se va a estudiar y en especificar lo que es y lo que hace más que en especificar la forma en que se utiliza.

Un objeto es un concepto o una vista abstracta de una entidad. Es como una variable en algunos aspectos, su objetivo es representar una cosa, por ejemplo un objeto podría ser un botón o un campo de entrada de un documento HTML. Cualquier objeto posee unas características que lo define, estas características se llaman propiedades, cada propiedad tiene un valor de algún tipo unido a él. Con los objetos podemos hacer cosas, necesitamos saber los nombres de las cosas que podemos hacer y los parámetros que necesitamos para hacerlas. Cada una de estas cosas se llama método.

Objeto window:

Se trata del objeto más alto en la jerarquía del navegador (navigator es un objeto independiente de todos en la jerarquía), pues todos los componentes de una página web están situados dentro de una ventana. El objeto window hace referencia a la ventana actual. Veamos a continuación sus propiedades y sus métodos.

Propiedades:

- **closed**: Es un booleano que nos dice si la ventana está cerrada (closed = true) o no (closed = false).
- **defaultStatus**: Cadena que contiene el texto por defecto que aparece en la barra de estado (status bar) del navegador.
- **Frames**: Es un array: cada elemento de este array (frames[0], frames[1], ...) es uno de los frames que contiene la ventana. Su orden se asigna según se definen en el documento HTML.
- **History**: Se trata de un array que representa las URLS visitadas por la ventana (están almacenadas en su historial).
- **Length**: Variable que nos indica cuántos frames tiene la ventana actual.
- **Location**: Cadena con la URL de la barra de dirección.

IES de Teis
Avda. de Galicia, 101
36216 – Vigo
Tfno: 886 12 04 64
e-mail: ies.teis@edu.xunta.es
http://www.iesteis.es

- **Name**: Contiene el nombre de la ventana, o del frame actual.
- **Opener**: Es una referencia al objeto window que lo abrió, si la ventana fue abierta usando el método open() que veremos cuando estudiemos los métodos.
- **Parent**: Referencia al objeto window que contiene el frameset.
- **Self**: Es un nombre alternativo del window actual.
- **Status**: String con el que tiene la barra de estado.
- **Top**:Nombre alternativo de la ventana del nivel superior.
- **Window**:Igual que self: nombre alternativo del objeto window actual.

Métodos:

- **alert(mensaje):** Muestra el mensaje 'mensaje' en un cuadro de diálogo
- **blur():** Elimina el foco del objeto window actual.
- **clearInterval(id):** Elimina el intervalo referenciado por 'id' (ver el método setInterval(), también del objeto window).
- **clearTimeout(nombre):** Cancela el intervalo referenciado por 'nombre' (ver el método setTimeout(), también del objeto window).
- **close()**: Cierra el objeto window actual.
- **confirm(mensaje):** Muestra un cuadro de diálogo con el mensaje 'mensaje' y dos botones, uno de aceptar y otro de cancelar. Devuelve true si se pulsa aceptar y devuelve false si se pulsa cancelar.
- **focus():** Captura el foco del ratón sobre el objeto window actual.
- **moveBy(x,y):** Mueve el objeto window actual el número de pixels especificados por (x,y).
- **moveTo(x,y):** Mueve el objeto window actual a las coordenadas (x,y).
- **open(URL,nombre,caracteristicas):** Abre la URL que le pasemos como primer parámetro en una ventana de nombre 'nombre'. Si esta ventana no existe, abrirá una ventana nueva en la que mostrará el contenido con las características especificadas. Las características que podemos elegir para la ventana que queramos abrir son las siguientes:
    - **toolbar** = [yes|no|1|0]. Nos dice si la ventana tendrá barra de herramientas (yes,1) o no la tendrá (no,0).
    - **location** = [yes|no|1|0]. Nos dice si la ventana tendrá campo de localización o no.
    - **directories** = [yes|no|1|0]. Nos dice si la nueva ventana tendrá botones de dirección o no.
    - **status** = [yes|no|1|0]. Nos dice si la nueva ventana tendrá barra de estado o no.
    - **menubar** = [yes|no|1|0]. Nos dice si la nueva ventana tendrá barra de menús o no.
    - **scrollbars** = [yes|no|1|0]. Nos dice si la nueva ventana tendrá barras de desplazamiento o no.
    - **resizable** = [yes|no|1|0]. Nos dice si la nueva ventana podrá ser cambiada de tamaño (con el ratón) o no.
    - **width** = px. Nos dice el ancho de la ventana en pixels.
    - **height** = px. Nos dice el alto de la ventana en pixels.
    - **outerWidth** = px. Nos dice el ancho *total* de la ventana en pixels. A partir de NS 4.
    - **outerHeight** = px. Nos dice el alto *total* de la ventana el pixels. A partir de NS 4

IES de Teis
Avda. de Galicia, 101
36216 – Vigo
Tfno: 886 12 04 64
e-mail: ies.teis@edu.xunta.es
http://www.iesteis.es

- **left** = px. Nos dice la distancia en pixels desde el lado izquierdo de la pantalla a la que se debe colocar la ventana.
- **top** = px. Nos dice la distancia en pixels desde el lado superior de la pantalla a la que se debe colocar la ventana.

- **prompt(mensaje,respuesta_por_defecto):** Muestra un cuadro de diálogo que contiene una caja de texto en la cual podremos escribir una respuesta a lo que nos pregunte en 'mensaje'. El parámetro 'respuesta_por_defecto' es opcional, y mostrará la respuesta por defecto indicada al abrirse el cuadro de diálogo. El método retorna una cadena de caracteres con la respuesta introducida.
- **scroll(x,y):** Desplaza el objeto window actual a las coordenadas especificadas por (x,y).
- **scrollBy(x,y):** Desplaza el objeto window actual el número de pixels especificado por (x,y).
- **scrollTo(x,y):** Desplaza el objeto window actual a las coordenadas especificadas por (x,y).
- **setInterval(expresion,tiempo):** Evalua la expresión especificada después de que hayan pasado el número de milisegundos especificados en tiempo. Devuelve un valor que puede ser usado como identificativo por clearInterval().
- **setTimeout(expresion,tiempo):** Evalua la expresión especificada después de que hayan pasado el número de milisegundos especificados en tiempo. Devuelve un valor que puede ser usado como identificativo por clearTimeout().

NOTA: Estas dos funciones permiten ejecutar una función pasado un cierto intervalo de tiempo. La única diferencia entre ellas es que "setInterval" se ejecutará una y otra vez en intervalos de x segundos, en cambio, setTimeout se ejecutará una sola vez pasados x segundos.

Estas funciones se utilizan sobre todo para realizar animaciones o por ejemplo para redireccionar una página transcurridos x segundos.

Ambos se pueden cancelar mediante clearTimeout(temporizador) y clearInterval(intervalo).

Por ejemplo, si quisiéramos programar un reloj en la barra de estado la función más idónea sería setInterval. Teniendo una función llamada "reloj()" que imprime en la barra de estado la hora actual utilizaríamos la función setInterval de esta forma:

```
setInterval("reloj()",1000);
```

El primer parámetro es la función a ejecutar, el segundo es el intervalo en milisegundos, es decir, 1000 milisegundos = 1 segundo. De esta forma, cada segundo se ejecutará la función "reloj()" y por lo tanto se mostrará un reloj con la hora actual que irá cambiando segundo a segundo.

Ejemplo1: Crear una ventana.

```
var ventana=null;
```

IES de Teis
Avda. de Galicia, 101
36216 – Vigo
Tfno: 886 12 04 64
e-mail: ies.teis@edu.xunta.es
http://www.iesteis.es


```
ventana = window.open ("","",opciones);
ventana.document.write ("Esta es la ventana que se ha creado");
```

Ejemplo2: Cerrar una ventana.

```
if (ventana && ventana.closed)
        ventana.close(); // cierra la ventana creada
window.close(); // cierra la ventana actual.
```

Todos los objetos **window** y **frame** tienen asociados un objeto **location**. Este objeto permite modificar la URL de la página para cambiar a una nueva página.

E*jemplo3:*

```
nuevo=window.open("","",opciones);
nuevo.location.href="prueba.htm";
```

Introducción a los eventos:

En JavaScript, la interacción con el usuario se consigue mediante la captura de los eventos que éste produce. Un **evento** es una acción del usuario ante la cual puede realizarse algún proceso (por ejemplo, el cambio del valor de un formulario, o la pulsación de un enlace).

Los eventos se capturan mediante los manejadores de eventos. El proceso a realizar se programa mediante funciones JavaScript llamadas por los manejadores de eventos.

Existen varias formas diferentes de manejar eventos en Javascript.

Forma	Ejemplo
Mediante atributos HTML	<button onClick="..."></button>
Mediante propiedades Javascript	.onclick = function() { ... }
Mediante addEventListener()	addEventListener("click", ...)

Cada una de estas opciones se puede utilizar para gestionar eventos en Javascript de forma equivalente, pero cada una de ellas tiene sus ventajas y sus desventajas. En los siguientes apartados veremos detalladamente sus características, pero por norma general, lo aconsejable es utilizar la última, los listeners, ya que son las más potentes y flexibles.

IES de Teis
Avda. de Galicia, 101
36216 – Vigo
Tfno: 886 12 04 64
e-mail: ies.teis@edu.xunta.es
http://www.iesteis.es

1. **Mediante atributos HTML:** Es la opción menos aconsejable ya que suele ser buena práctica **no** incluir llamadas a funciones Javascript en nuestro código .html. No se debe mezclar código html con javascript.

   Ejemplo:

```
<script>
  function saludar() {
    alert("Hola que tal estas!");
  }
</script>

<button onClick="saludar()">Saludar</button>
```

2. **Mediante propiedades Javascript:** En esta ocasión haremos uso de una propiedad Javascript, a la que le asignaremos la función con el código asociado.

```
<button>Saludar</button>
<script>
const boton = document.querySelector("button");
//Este método devuelve el primer botón. A partir de ese momento se almacena en la constante boton y nos permite acceder a sus métodos

button.onclick = function() {
  alert("Hola que tal estas!");
}
</script>
```

NOTA. Método querySelector y querySelectorAll

- El método **querySelector()** recibe entre paréntesis y comillas los selectores de la misma forma que se indican en CSS. Esto quiere decir que para id se antepone numeral(#) y para clase se antepone punto (.).
  Por ejemplo, si tenemos un párrafo cuya clase es primero y se encuentra dentro de un , cuya id es caja, podríamos escribir un código como el siguiente:
  const parrafo_primero = document.querySelector("#caja p.primero");
  Este método devuelve el primer elemento que coincida con los selectores dados. Para el ejemplo anterior, queda guardado en la constante parrafo_primero.
  A continuación podríamos realizar el trabajo que necesitemos con JavaScript. Por ejemplo, si deseamos cambiar el color a verde, usaríamos un código como el que sigue:

```
parrafo_primero.style.color = "green";
```

- **querySelectorAll**(): El uso de querySelectorAll() es similar a querySelector(). La diferencia esencial es que en este caso no recibimos solo el primer elemento que coincida, sino el conjunto de todos los que coincidan.

Siguiendo el ejemplo planteado, veamos si ahora deseamos obtener todos los párrafos, en ese caso indicamos:

const parrafos = document.querySelector("#caja p");

Como el resultado que queda guardado en párrafos es un array de JavaScript, para utilizarlo debemos recorrerlo. Esto lo podemos realizar con un for. Veamos el ejemplo para poner el color de fondo en amarillo a todos los párrafos que están dentro del con id caja:

for(i=0 ; i < parrafos.length ; i++){

parrafos[i].style.backgroundColor = "yellow";

3. **Mediante addEventListener():**

- Con **.addEventListener()** se pueden añadir fácilmente varias funcionalidades.
- Con **.removeEventListener()** se puede eliminar una funcionalidad previamente añadida.
- Con **.addEventListener()** se pueden indicar ciertos comportamientos especiales.

El método addEventListener() permite añadir una escucha del evento indicado (primer parámetro) y en el caso de que ocurra, se ejecutará la función asociada (segundo parámetro)

Método	Descripción
.addEventListener(event,func)	Escucha el evento event, y si ocurre, ejecuta func.
.addEventListener(event,func,options)	Idem, pasándole ciertas opciones.

Para comprobar su funcionamiento, utilizaremos el mismo ejemplo que en los casos anteriores.

```
const boton = document.querySelector("button");

button.addEventListener("click", function() {
  alert("Hola que tal estas!");
});
```

Podemos utilizar funciones para que el código quede mucho más legible.

```
const boton = document.querySelector("button");

const saludar = () => alert("hola que tal estas!");

button.addEventListener("click", saludar);
```

Una de las características más importantes es que nos permite añadir múltiples listeners y así asociar varias funciones a un mismo evento

```
<button>Saludar</button>

<style>
  .red { background: red }
</style>

<script>
const boton = document.querySelector("button");
const saludar = () => alert("Hola que tal estas!");
```

IES de Teis
Avda. de Galicia, 101
36216 – Vigo
Tfno: 886 12 04 64
e-mail: ies.teis@edu.xunta.es
http://www.iesteis.es


```
const cambiar = () => boton.classList.toggle("red"); //Añade o quita el color rojo del botón

boton.addEventListener("click", saludar);        // Hello message
boton.addEventListener("click", cambiar);        // Add/remove red CSS
</script>
```

Podemos trabajar con varias opciones y eliminar la escucha. Esto se tratará más adelante.

Ejemplo:

```
<HTML>
<HEAD><TITLE>Eventos</TITLE>
<SCRIPT>
<!--
function Reacciona(campo) {
        alert("¡Introduzca un valor!")
        campo.focus()
}
//-->
</SCRIPT></HEAD>
<BODY>
<FORM METHOD=POST>
<INPUT TYPE=text NAME=campo onFocus="Reacciona(this)">
</FORM>
 </BODY>
 </HTML>
```

Nota: El evento onFocus no funciona con Mozilla.

Eventos onLoad y onUnload:

El evento onload de Javascript se activa cuando se termina de cargar la página. Se ha de colocar en la etiqueta <body>, aunque en versiones modernas de Javascript, también lo aceptan otros elementos como las imágenes.

Con el evento onload podemos ejecutar acciones justo cuando se han terminado de cargar todos los elementos de la página. Es un evento bastante utilizado pues es muy habitual que se deseen llevar a cabo acciones en ese preciso instante.

Ejemplo:

<BODY onLoad="Hola()" onUnload="Adios()">

La función Hola() se ejecutará antes de que se cargue la página y la función Adios() al abandonarla.

Ej: Pasar el foco a un elemento del formulario.

Nombre

IES de Teis
Avda. de Galicia, 101
36216 – Vigo
Tfno: 886 12 04 64
e-mail: ies.teis@edu.xunta.es
http://www.iesteis.es

Apellidos

Secuencia y desarrollo:

1. ***Ejemplo1: Implementa el siguiente código javaScript donde se crea una nueva ventana, la cual se va moviendo aleatoriamente.***

```
<HTML>
<HEAD>
        <title>Ejemplo de JavaScript</title>
        <script LANGUAGE="JavaScript">
<!--
function moverVentana()
   {
      mi_ventana.moveBy(5,5);
      i++;
      if (i<20)
            setTimeout('moverVentana()',100);
      else mi_ventana.close();
   } //-->
</script>
</HEAD>
<BODY>
<script LANGUAGE="JavaScript">
<!--
   var opciones="left=100,top=100,width=250,height=150", i= 0;
  mi_ventana = window.open("","",opciones);
  mi_ventana.document.write("Una prueba de abrir ventanas");
  mi_ventana.moveTo(400,100);
  moverVentana();
//-->
</script>
</BODY>
</HTML>
```

2. **Ejercicio2:** Implementar la siguiente página, donde se creará una ventana con un ancho y un alto que se piden por teclado. Al hacer clic en el botón Crear Ventana se creará la ventana con las especificaciones dadas y aparecerá un texto, al hacer click en el botón Cerrar Ventana se cerrará la ventana actual y la ventana creada. Por defecto, la nueva ventana que se crea se visualiza sin barras

de menú, de herramientas, de dirección, de status, etc. Se pueden especificar ciertas características para variar las opciones por defecto.