Programación en JavaScript

2. Fundamentos de JavaScript

Sintaxis Básica de JavaScript

Saltos de Línea

Es un código que indica un movimiento a la siguiente línea de texto.

sintaxis:

```
<script>
    alert ( "Mensaje\nal\nUsuario usando\nJavascript" );
</script>
```

Comentarios

Son anotaciones legibles al programador en el código fuente de un Programa.

sintaxis:

```
<script>
    // Esto es un comentario
</script>
```

La combinación de de caracteres /* y */, sirve para comentar varias líneas, formando un bloque de comentarios.

```
<script>
    /* Esto es un comentario de varias líneas */
</script>
```

Método Confirm( )

confirm( ) es un método predefinido de Javascript que muestra un cuadro de diálogo con un mensaje y dos botones, Aceptar y Cancelar. Devuelve un valor booleano

sintaxis:

```
confirm ( "Mensaje de confirmación al Usuario usando Javascript" );
```

Ejemplo en Javascript

nombre: confirmar.html

sintaxis:

```
<script>
    confirm ( "Mensaje de confirmación al Usuario usando Javascript" );
</script>
```

Método document.write( )

document.write( ) es un método predefinido de Javascript que permite escribir en una página texto, el resultado de una función o ambos.

sintaxis:

```
document.write ( "Texto insertado usando Javascript" );
```

Ejemplo en Javascript

nombre: ejemplos.html

sintaxis:

```
<script>
    document.write ( "<br><br><hr>Ejemplos realizados en<br> <b>
    <i>Javascript</i></b><hr>" );
</script>
```

Método document.getElementById( )

document.getElementById( ) usa referencias para acceder a los elementos HTML desde Javascript por el valor de su atributo id.

sintaxis:

```
document.getElementById('ID').innerHTML = "Texto insertado
usando Javascript";
```

Ejemplo en Javascript

nombre: ejemplos.html

sintaxis:

```
<p id="ejemplo"></p>
<script>
    document.getElementById('ejemplo').innerHTML = "Ejemplo de Texto
    Javascript en <b>HTML</b><hr>";
</script>
```

2.4 Funciones

Una función es básicamente un bloque de código identificado por un nombre. Declarar el código dentro de una función en lugar de hacerlo en el espacio global ofrece varias ventajas y la principal es que aunque se declaren, no se ejecuta hasta que son llamadas por su nombre.

sintaxis:

```
<script>
        function nombredelafuncion()
        {
                bloque de instrucciones;
        }
</script>
```

Ejemplo en Javascript

nombre: confirmar2.html

sintaxis:

```
<script>
        // funcion confirmar
        function funconfirmar()
        {
                confirm ( "Mensaje de confirmación al Usuario usando Javascript" );
        }
</script>
```

Javascript desde un archivo externo

Javascript permite insertar uno o más archivos externos, y posteriormente llamarlos mediante el atributo src en todos los documentos del mismo sitio web y reutilizar el código en cualquier momento que se desee.

sintaxis:

```
<script src="nombredelarchivo.js"></script>
```

Ejemplo en Javascript

nombre: funciones.js

sintaxis:

```
// funcion confirmar
function funconfirmar()
	{
		confirm ( "Mensaje de confirmación al Usuario usando Javascript" );
	}
```

nombre: confirmar3.html

sintaxis:

```
<script>
	funconfirmar();
</script>
```

3.2. Eventos y manejo de eventos

En Javascript, las acciones del usuario se denominan eventos. Cuando el usuario realiza una acción, como hacer click con el ratón o pulsar una tecla, se activa un evento que es especifico para cada acción. Además de los eventos producidos por un usuario, también hay eventos activados por sistema.

Evento onClick

nombre: ejemplos.html

sintaxis:

```
<script>
	<button onclick="alert('Mensaje\nal\nUsuario usando eventos
	en\nJavascript')">Evento Pulsa Aquí</button>
</script>
```

sintaxis:

```
<script>
	<p onclick="document.write('Mensaje\nal\nUsuario usando eventos
	en\nJavascript')">Evento Pulsa Aquí</p>
</script>
```

sintaxis:

```
<script>
	<p onclick="document.getElementById('mensaje').innerHTML =
	'Mensaje\nal\nUsuario usando eventos en\nJavascript'">Evento Pulsa
	Aquí</p>
	<p id="mensaje"></p>
</script>
```

2.3 Estructuras de Control

Javascript ofrece un total de cuatro declaraciones para procesar código predeterminadas: if, switch, for, while.

Sentencia: if

La instrucción *if* evalúa una determinada condición y, en caso de ser verdadera, se ejecuta un bloque de instrucciones. Si dicha condición no se cumple, ninguna de las instrucciones es ejecutada.

sintaxis:

```
if ( condición )
{
        bloque de instrucciones;
}
```

La sentencia *if* presenta una opción para indicar otro bloque de instrucciones que se ejecutará en caso de no cumplirse la condición especificada (*else*):

sintaxis:

```
if ( condición )
{
        bloque de instrucciones caso verdadero;
}
else
{
        bloque de instrucciones caso falso;
}
```

Ejemplo en Javascript

sintaxis:

```
<script>
    function funifelse()
    {
        var resp = confirm ( "Acepta los términos y condiciones" );
        if ( resp == true )
        {
            alert ("Términos y condiciones aceptados");
        }else
        {
            alert ("Términos y condiciones NO aceptados");
        }
    }
</script>
```

Método Prompt( )

prompt( ) es un método predefinido de Javascript que muestra un cuadro de diálogo pidiendo una entrada al usuario y, opcionalmente, un valor de defecto. Devuelve un objeto String con el contenido introducido por el usuario, o null si se pulsa la tecla cancelar.

sintaxis:

```
prompt( "Información ingresada por el usuario usando Javascript", "texto por defecto" );
```

Sentencia: switch

La sentencia *switch* permite la ejecución de un bloque de instrucciones en función del valor que tome una expresión.

sintaxis:

```
switch ( expresión )
{
    case resultado1:
        bloque de instrucciones resultado1;
    break;
    case resultado2:
        bloque de instrucciones resultado2;
    break;
    ...
    default:
        bloque de instrucciones por defecto;
}
```

Ejemplo en Javascript

sintaxis:

```
<script>
    function funswitch()
    {
        var resp = prompt( "Módulo que cursa actualmente en el diplomado", "1" );
        switch ( resp )
        {
            case '1':
                alert ("Su Módulo es HTML y CSS");
            break;
            case '2':
                alert ("Su Módulo es Javascript y JQuery");
            break;
```

```
                case '3':
                        alert ("Su Módulo es XML y AJAX");
                break;
                case '4':
                        alert ("Su Módulo es BASE DE DATOS");
                break;
                default:
                        alert ("El Módulo "+resp+" No pertenece a Webmaster");
        }
    }
</script>
```

Sentencia: for

El bucle *for* tiene como objetivo repetir un bloque de instrucciones mientras se cumpla una condición preestablecida.

sintaxis:

```
for ( inicialización ; condición ; inc o dec )
{
        bloque de instrucciones;
}
```

Ejemplo en Javascript

sintaxis:

```
<script>
    function funfor()
    {
        var inicio = prompt("Ingrese el número para iniciar el conteo");
        // Conversión a Númerico
        inicio= Number(inicio);
        var fin = prompt("Ingrese el número donde finalizar el conteo");
        // Conversión a Númerico
        fin = Number(fin);
        if ( inicio < fin )
        {
            for ( var i= inicio ; i<= fin ; i++ )
            {
                alert ("Número : "+ i);
            }
        }
        else
        {
            if ( inicio > fin )
            {
```

```
                        for ( var i= inicio ; i>= fin ; i-- )
                        {
                                alert ("Número : "+ i);
                        }
                }
                else
                {
                        alert ("Los números son iguales, no hay conteo.");
                }
        }
}
</script>
```

Sentencia: while

El bucle *while* permite la repetición de un bloque de instrucciones un número indeterminado de veces. Es posible que las sentencias del bucle no se lleguen a ejecutar nunca, ya que antes de proceder a interpretar la primera instrucción se evalúa la condición, y si ésta resulta ser falsa, no entrará en las instrucciones del bloque.

sintaxis:

```
while ( condición )
{
        bloque de instrucciones;
}
```

Ejemplo en Javascript

sintaxis:

```
<script>
        function funwhile()
        {
                var num = prompt("Ingrese el número entre 0-50");
                // Conversión a Númerico
                num= Number(num);
                        if ( num < 0 || num > 50 )
                        {
                                alert ("Número fuera del rango 0-50");
                        }
                        else
                        {
                                while ( num < 50 )
                                {
                                        num++;
                                        alert ("Número : "+ num);
                                }
```

```
                }
        }
</script>
```

Sentencia: do-while

El bucle *do-while* es similar al bucle generado con la instrucción while, con la diferencia de que la condición de ejecución se comprueba tras la ejecución del bloque de sentencias. Esto implica que existe siempre la garantía de que las instrucciones del bloque se ejecutarán al menos una vez.

sintaxis:

```
do
{
        bloque de instrucciones;
}while ( condición );
```

Ejemplo en Javascript

sintaxis:

```
<script>
        function fundowhile()
        {
                do
                {
                        var resp = confirm ( "Acepta los términos y condiciones" );
                        if ( resp == true )
                        {
                                alert ("Términos y condiciones aceptados");
                        }else
                        {
                                alert ("Términos y condiciones NO aceptados");
                        }
                }while( resp == false);
        }
</script>
```