TEMA 2. Introducción a JavaScript

Desarrollo Web en Entorno Cliente.

Profesor: Juan José Gallego García

Índice :

- Operadores:
  - Operadores de asignación
  - Operadores matemáticos
  - Operadores lógicos o booleanos
  - Operadores relacionales
  - Operadores matemáticos especiales
  - Operadores matemáticos reducidos
  - Orden de ejecución de los operadores
- Caracteres especiales en JavaScript
- Bibliografía

¿Qué es un Script?

La definición depende mucho del contexto, pero dentro de la informática se puede afirmar lo siguiente: Un script es un archivo de texto que contiene órdenes (o comandos) que se deben ejecutar secuencialmente en un ordenador. Los lenguajes de script son aquellos que no son compilados, son interpretados por la aplicación que los hace funcionar, en nuestro caso el navegador web (de hay el nombre intérprete web).

¿Qué es JavaScript?

JavaScript (abreviado a JS) es un lenguaje de programación en entorno cliente. Es ejecutado dentro del navegador web, por lo que si se pierde conexión con el servidor o con la red, seguirá funcionando correctamente.

Este lenguaje se iba a llamar inicialmente LiveScript, pero cuando salió al mercado laboral (1995), se cambió el nombre debido a que en aquella época la palabra Java estaba de moda en el mundo informático, en otras palabras, fue por marketing. Actualmente **ECMAScript** es la especificación estándar de **JavaScript**.

Estándares ECMAScript**.**

Las 4 cualidades de JavaScript son:

- **Imperativo**, esto significa que ejecuta secuencialmente una serie sentencias (órdenes) para la realización de una tarea. Para separar una sentencia de la siguiente usaremos un carácter especial, el punto y coma (;).
- **Orientado a Objetos**, es una evolución en la programación imperativa que agrupa el estado y las posibles modificaciones.

- **Débilmente tipado**, no es necesario declarar el tipo de una variable y además permite el cambio de tipos de variable durante la ejecución (tipos dinámicos).
- **Dinámico**, permite la modificación de la información durante la ejecución de las sentencias. En estos tiempos, esta propiedad parece evidente, pero hace no mucho tiempo fue una revolución la posibilidad de modificaciones en tiempo de ejecución.

¿Cómo usar JavaScript?

JavaScript debe incluirse dentro de un archivo HTML, pero existen 3 formas de utilizarlo (algo en común con CSS):

- En la cabecera del HTML,
- Incrustado en el HTML,
- O referenciado en un archivo externo.

Para utilizarlo en la cabecera del HTML, lo guardaremos dentro de las etiquetas <script type=”text/javascript”> y </script>. ó sin type. Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>
<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```

Dentro de un HTML pueden existir multitud de estas etiquetas (nunca anidadas), e incluso pueden estar dentro del body (no respetando ciertos estándares).

Al utilizarlo incrustado, debemos ayudarnos de los eventos que se producen en un HTML, los cuales los veremos más adelante, pero al menos veamos un ejemplo:

```
<!DOCTYPE html>
<html>
<head>
</head>
<body onload="javascript:alert('Hola');">
</body>
</html>
```

El texto ***javascript*:** se utiliza para indicar que se ejecuta código de este tipo, la mayoría de las veces no es necesario (el navegador lo detecta automáticamente).

Para utilizar un archivo externo, utilizamos la misma etiqueta, añadiendo un nuevo atributo **src** (de source traducido como fuente u origen) que indica la posición del archivo. Ejemplo:

```html
<!DOCTYPE html>
<html>
<head>
    <script charset="UTF-8" src="miScript.js"></script>

</head>
<body onload="alerta();">
</body>
</html>
```

```javascript
//miScript.js
function alerta() {
    alert ("Hola mundo");
    console.log("Hola mundo");
}
```

Palabras reservadas en JavaScript.

En todos los lenguajes de programación existen una serie de palabras que no se deben de usar porque tienen un sentido propio, en JS son:

break, case, catch, continue, default, delete, do, else, finally, for, function, if, in, instanceof, new, null, return, switch, this, throw, try, typeof, var, void, while, with.

JavaScript no se tratan los espacios en blanco, saltos de línea y tabuladores, de forma similar a cómo los tratan los navegadores web. En JavaScript se distinguen entre las mayúsculas y las minúsculas, las palabras “HOLA” y “hola” son palabras diferentes.

Tipos usados en JavaScript.

Hemos dicho que JavaScript es un lenguaje de programación débilmente tipado, eso no quiere decir que no tenga tipos. Estos tipos son soportados por una serie de datos. Un tipo datos es una restricción de valores dada, por ejemplo: un entero sólo puede tener valores de números positivos, negativos o cero… sin parte decimal.

En JavaScript tenemos los siguientes tipos principales:

- **Numérico**: que admite todos los posibles valores de números (0, 12, -23.4, etc.),
- **Booleano**: que puede tomar valor SÍ (true) o NO (false).

- **Carácter**: que guarda caracteres (¿, $, w, c, 8, etc.) para identificarlos se pone una comilla simple al principio y una comilla simple al final, los ejemplos anteriores serían: '¿', '$', 'w', 'c', '8'.
- **String o cadena de caracteres**: son agrupaciones de caracteres que pueden formar palabras, frases, etc. Para distinguir que es una cadena de caracteres se antepone unas comillas dobles y otras que van al final. Nota: JavaScript permite utilizar comillas simples aunque lo más correcto es usar las comillas dobles. Ejemplo: "Esto es una cadena de caracteres".
- **Arrays**: son agrupaciones de objetos, se verán y explicarán más adelante.
- **Definidos por el usuario**: son los que crean los programadores con estructuras de POO (objetos).

Ej: foo = "Hola"; ---> Valor como cadena.

foo = 3 ; ---> Ahora como numérico.

Muchos objetos creados en programación, inicialmente (antes de que se les asigne un valor), toman el valor **undefined**. Posteriormente podemos usar **null** para asignar “nada” a una variable.La diferencia entre **undefined** y **null**, es que el primero no necesita asignación y JS lo crea automáticamente, el segundo es una variable que podemos asignar y usar como variable vacía. Esto nos permite hacer asignaciones para anular, comparaciones, etc.

Comentarios

En JavaScript tenemos dos formas de poner comentarios:

- **En una sola línea** anteponiendo los caracteres //.
- **En múltiples líneas** al principio pondremos los caracteres /* y al final los caracteres */.

```
<script type="text/javascript">

    /* Esto es un comentario de
       multiples lineas */

    // Esto es un comentario de una sola linea

</script>
```

Funciones básicas: alert

La función **alert** muestra un mensaje de alerta, sólo tiene un argumento que el texto que queremos que muestre, se suele utilizar para mostrar información al usuario. Ejemplo:

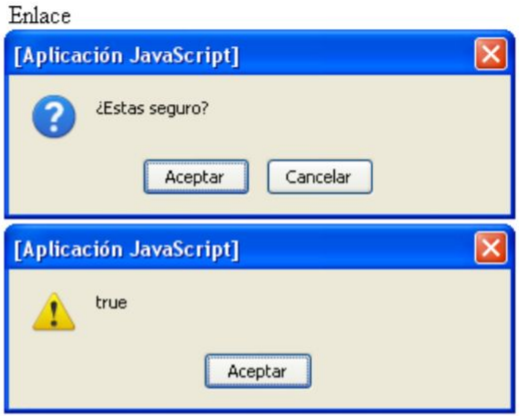
```
<body>
  <a href="www.iluro7.tk"
     onmouseover="javascript:
          alert('¡Haz clic en este enlace!');">
    Enlace
  </a>
</body>
```

Funciones básicas: confirm

La función **confirm** muestra un mensaje (el mensaje lo recibe como argumento), también muestra un par de botones de confirmación (aceptar y cancelar), esta función devuelve un valor booleanos (true si se pulsó aceptar o false si se pulsó cancelar).

Ejemplo:

```
<a onclick="javascript:
    var res=
      confirm(
        '¿Estas seguro?'
      );
    alert(res);">
  Enlace
</a>
```



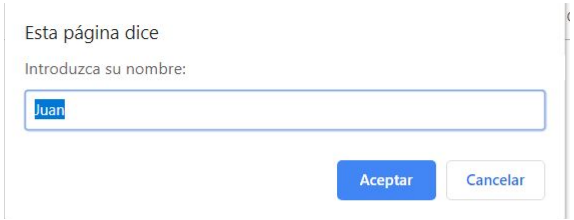
Funciones básicas: prompt

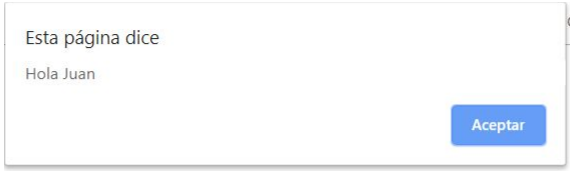
La función **prompt** muestra un mensaje recibido como parámetro y solicita al usuario la introducción de un texto. Permite aceptarlo (envía ese texto al código) o cancelarlo (envía null). Se puede sobrecargar la función introduciendo un valor predefinido para la respuesta. Ejemplo:

```
function alerta()
{
var mensaje;
var opcion = prompt("Introduzca su nombre:", "Juan");

if (opcion == null || opcion == "") {
        mensaje ="Has cancelado o introducido el nombre vacío";

        } else {
            mensaje = "Hola " + opcion;
            }
            alert( mensaje);
}
```





Variables y Constantes

Para crear variables y constantes debemos utilizar las palabras reservadas: var/let y const respectivamente. Se comportan igual que en los lenguajes en los que se basó JavaScript: Java y C. Ejemplo:

```
<script type="text/javascript">
    var coste = 0;
    const PI = 3.14159;
</script>
```

Operadores de asignación

El operador de asignación es el igual = Ejemplo:

```
<script type="text/javascript">
    var coste = 13;
    alert(coste);
</script>
```

Operadores matemáticos

Las operaciones matemáticas se realizan con los siguientes símbolos:

- Suma +
- Resta -
- Multiplicación *
- División /
- Resto de la división entera %

```
<script type="text/javascript">
    var sum = 10 + 3;
    var res = 10 - 3;
    var mul = 10 * 3;
    var div = 10 / 3;
    var res = 10 % 3;
</script>
```

Operadores lógicos o booleanos

A nivel del tipo booleano:

- **Negación**: se utiliza el cierre de la exclamación: !false equivale a true
- **AND lógico**: se utilizan 2 ampersand (&&) y será true, si ambas entradas son true: true && false equivale a false
- **OR lógico**: se utilizan 2 barras verticales (||) y será true, si alguna entrada es true: true || false equivale a true

A nivel de bits:

- **AND**: se utiliza 1 ampersan &
- **OR**: se utiliza 1 barra vertical |
- **XOR**: se utiliza el acento circunflejo ^
- **NOT**: se utiliza la virgulilla ~ en modo prefijo
- **Desplazamiento a la izquierda**: se utilizan 2 menores que <<
- **Desplazamiento a la derecha** (propagando el valor más significativo): se utilizan 2 mayores que >>
- **Desplazamiento a la derecha** (**rellenando de ceros**): se utilizan 3 mayores que >>>

Operadores relacionales

Los operadores relacionales son los utilizados para comparar dos expresiones: mayor que (>), menor que (<), mayor o igual que (>=), menor o igual que (<=), igual que (= =) y distinto de (!=). Ejemplos:

```
<script type="text/javascript">
    var r1 =  1 > 0;    // devuelve true
    var r2 = 20 <= 0;   // devuelve false
    var r3 = 12 == 3*4; // devuelve true
    var r4 =  0 != 0;   // devuelve false
</script>
```

Operadores matemáticos especiales

Para realizar incrementos o decrementos (en una unidad) usamos los símbolos: ++ y -- antes (prefijo) o después (sufijo) de la expresión que queramos incrementar o decrementar en uno:

```
<script type="text/javascript">
    var num = 1;
    num++;      // Incrementa y guarda 2 en num
    ++num;      // Incrementa y guarda 3 en num
    num--;      // Decrementa y guarda 2 en num
    --num;      // Decrementa y guarda 1 en num
</script>
```

La diferencia entre incrementar/decrementar antes o después con '++' ó '--' podemos apreciarla con los siguientes ejemplos: **alert(num++) / alert (++num)**

Operadores matemáticos reducidos

En algunos lenguajes de programación (JavaScript, PHP, Java, etc.) se permite reducir los operadores básicos (suma, resta, multiplicación, división y módulo) siempre y cuando el primero de los argumentos sea la variable origen y destino, veamos las opciones existentes junto con sus traducciones:

Formato Reducido	Formato Equivalente
peso += 3;	peso = peso + 3;
peso -= 3;	peso = peso - 3;
peso *= 3;	peso = peso * 3;
peso /= 3;	peso = peso / 3;
peso %= 3;	peso = peso % 3;

Orden de ejecución de los operadores

Orden	Operaciones evaluadas
1º	Paréntesis
2º	Negación (!), incrementos y decrementos previos (con el ++ o el -- antes de la expresión)
3º	Multiplicaciones, divisiones y módulos
4º	Sumas y restas
5º	Comparaciones de ordenación (<, >, <= y >=)
6º	Comparaciones de igualdad (== y !=)
7º	AND lógico
8º	OR lógico
9º	Asignaciones (=, +=, -=, *=, /= y %=)
10º	Incrementos y decrementos posteriores (con el ++ o el -- después de la expresión)

Caracteres especiales en JavaScript

En los lenguajes de programación existen caracteres que, o bien no se pueden representar, o bien son caracteres especiales que tienen un significado especial. Esto nos obligaría a no poder utilizarlos si no tuviéramos una forma de representarlos. Estos son:

Carácter	Nombre	Observaciones
\t	Tabulador	Carácter de difícil representación
\n	Salto de línea	Carácter de difícil representación
\'	Comillas simples	Usado para indicar texto
\"	Comillas dobles	Usado para indicar texto
\\	Barra de raíz	Usado para crear caracteres especiales

Bibliografía

- LibrosWeb
- W3Schools
- Artículo sobre JavaScript de la wikipedia
- Manual de desarrollo de la fundación Mozilla