FUNDAMENTOS DE JAVASCRIPT

Unidad didáctica 2

Fundamentos de JavaScript

- Comentarios
- Punto y coma
- Palabras reservadas
- Mayúsculas, minúsculas
- Secuencias de escape
- Variables
  - Globales
  - Locales
- Tipos de datos
  - Números
  - Cadenas de texto
  - Valores booleanos
  - Otros
  - Conversiones
- Operadores
  - Comparación
  - Aritméticos
  - Asignación
  - Booleanos
  - Bit a bit
  - De Objeto
  - Misceláneos

Comentarios

```
// Este es un comentario de una línea
var nombre="Marta" // comentario después del código
// Podemos dejar por ejemplo
//
// una línea en medio en blanco

// el salto de línea anterior no se tiene en cuenta
```

Comentarios

```
/* Ésta es una sección
de comentarios
en el código de JavaScript */
/* Autor: Alejandro Ruiz Lameiro
   e-mail: alex.ruiz@edu.xunta.gal
   Licencia: CC-BY
   Comentarios: Si encuentras algún error o
                realizas alguna mejora en el código
                me gustaría que me lo comunicaras
*/
```

Punto y coma

```
// obligado el uso del punto y coma
// si la línea contiene más de una instrucción.
var a=5; var b=6;


// cuando es una sóla instrucción no hay por qué usar ;
var a=5
var b=6

// mejor acostumbrase a usarlos siempre
```

Palabras reservadas

abstract	boolean	break	byte
case	catch	char	class
const	continue	deault	do
double	else	extends	false
final	finally	float	for
function	goto	int	implements
input	in	instanceof	interface
long	native	new	null
package	private	protected	public
return	short	static	super
switch	synchronized	this	throw
throws	transient	true	try
var	val	while	with

Uso de mayúsculas y minúsculas

Hola ≠ HOLA ≠ hola ≠ hOLa

// el código siguiente es perfectamente válido
var Var=VAR;

Case sensitive = Sensible a las mayúsculas y minúsculas

¡¡¡ MUCHA ATENCIÓN !!!

Secuencias de escape

Secuencia	Resultado
\\	\
\’	’
\”	”
\n	Salto de línea
\t	Tabulación horizontal
\v	Tabulación vertical
\f	Salto de página
\r	Retorno de carro
\b	Retroceso

Variables

- Zonas de memoria que almacenan datos.
- Nombres: caracteres alfanuméricos y el carácter _
- Declaración

```
var mi_variable_1;
var otra_variable;
var una_variable, dos_variable;
```

- Inicialización

```
var mi_variable_1=1;
var otra_variable="Pedro";
var una_variable=otra_variable;
var variable1=5, variable2=6;
var dos_variable=prompt("Inicializa la variable tu mismo");
```

~~Palabras reservadas~~
Nombres descriptivos
No hay que declarar el tipo

Seguir siempre el mismo criterio

Variables: ámbito (*"scope"*)

- Ámbito global (cualquier punto del programa).
  - Declaradas

```
<script>
    var variable_global;
    function unaFuncion () {
       // resto del código de la función
    }
</script>
```

  - No declaradas

```
<script>
    function unaFuncion () {
      variable_global=8;
      // resto del código de la función
    }
</script>
```

Variables: ámbito

- Ámbito local.
  - Declaradas (siempre)

```
<script>
    function unaFuncion () {
      var variable_local_función;
      // resto del código de la función
    }

    function otraFuncion () {
      variable_global=8;
      if condicion {
        var variable_local_otraFuncion;
        // resto del código del if
      }
      // resto
    }
</script>
```

Nombre_Variable_Global ≠ Nombre_variable_local (dentro del ámbito local)

Variables: ámbito

- Ámbito local de bloque.

```
<script>
    function unaFuncion () {
      var variable_local_función;
      // resto del código de la función
    }

    function otraFuncion () {
      variable_global=8;
      if condicion {
        let variable_local_if;
        // resto del código del if
      }
      // resto del código de la función
    }
</script>
```

Sintaxis y ejemplos let

Tipos de datos

Tipo	Descripción	Ejemplo
string	Una serie de caracteres dentro de comillas dobles o simples.	"Hola mundo" 'adiós mundo'
number	Un número con o sin decimales, con o sin signo.	9.45 -9
boolean	Un valor verdadero o falso.	true false
null	Sin contenido, simplemente es un valor null.	null
undefined	Sin definir.	
object	Cualquier objeto software definido por sus propiedades y métodos (los arrays también son objetos).	
function	La definición de una función.	

Conversiones entre tipos de datos

```
4 + 5          // resultado = 9
4 + "5"        // resultado = "45"
4 + 5 + "6"    // resultado = "96"
+"5"           // resultado = 5
-"5"           // resultado = -5
```

Orden de evaluación de las operaciones

DE	A	Función	Ejemplo	Resultado
Cadena	Número entero	parseInt	parseInt("34")	34
			parseInt("89.76")	89
	Número	parseFloat	parseFloat("34")	34
			parseFloat("89.76")	89.76
			4 + 5 + parseInt("6")	15
Número	Cadena		(" " + 3400)	"3400"
			(" " + 3400).length	4

Sintaxis y ejemplos de parseInt

Operadores

Tipo	Operadores	Función
Comparación	== != === !== > >= < <=	Compara el contenido de dos operandos
Aritméticos	+ - * / % ++ – ** +valor -valor	Calcula un nuevo valor como resultado de una operación aritmética
Asignación	= += -= *= /= %= **= <<= >>= >>>= &= |= ^=	Asigna el valor a la derecha del operador a la variable que esté a la izquierda de operador
Boolean	&& || !	Realiza una operación Y, O, NO
Bit a bit	& | ^ ~ << >> >>>	Realiza operaciones aritméticas o de desplazamiento de columna en las representaciones binarias de operandos
Objeto	. [ ] ( ) delete in instanceOf new this	Ayudan a los scripts a evaluar la herencia y capacidades de un objeto particular antes de que tengamos que invocar al objeto y sus propiedades o métodos
Misceláneos	, ?: typeof void	Comportamiento especial

Operadores de comparación

Operador	Comparación que realiza	Ejemplo	Resultado
==	Igualdad	30 == 30 30 == 30.0 "Marta" == "marta“ "123" == 123 parseInt("123") == 123	true true false true true
!=	Desigualdad	5 != 8	true
===	Igualdad estricta	3 === "3"	false
!==	Desigualdad estricta	3 !== "123"	true
>	Mayor que	"Marta" > "marta“ 9 > 13	false
>=	Mayor o igual que	5 > 3	true
<	Menor que	"Mark" < "Marta"	true
<=	Menor o igual que	7.29 <= 7.28	false

Admite todo tipo de operandos. Devuelve siempre un resultado booleano (true o false)

Operadores aritméticos

```
a = 5;
b = 10;
d = ”5”;
```

Operador	Operación que realiza	Tipos de operandos que admite	Resultado c
+	Suma (*)	c = a + b;	15
-	Resta	c = a - b;	-5
*****	Multiplicación	c = a * b;	50
/	División	c = b / a;	2
%	Módulo (resto)	c = b % a;	0
++	Incremento	c = ++a;	6
		c = a++;	5 (a=6)
--	Decremento	c = --a;	4
		c = a--;	5 (a=4)
******	Exponenciación	c = a**2;	25

+valor	Positivo (*)	c = +a;
		c = +d; (*)
		c = +true;
			5
			5
			1

-valor	Negativo (*)	c = -a;	-5

Todos admiten operandos de tipo integer y float y devuelve un resultado integer o float. (*) admite también string.

Operadores de asignación

Operador	Operación que realiza	Ejemplo	Equivalente
=	Asignación	c = a;	c = a;
+=	Suma y asigna	c += a;	c = c + a;
-=	Resta y asigna	c -= a;	c = c – a;
*=	Multiplica y asigna	c *= a;	c = c * a;
/=	Divide y asigna	c /= a;	c = c / a;
%=	Calcula módulo y asigna	c %= a;	c = c % a;
**=	Calcula la potencia y asigna	c **= a;	c = c ** a;
<<=	Desplaza a la izquierda de los bits de ‘c' tantas posiciones como indique ‘a‘ y asigna. Se rellena por la derecha con ceros.	c <<= a;	c = c <<a;
>>=	Desplaza a la derecha de los bits de ‘c’ tantas posiciones como indique ‘a‘ y asigna. Se conserva el signo.	c >>= a;	c = c >> a;
>>>=	Desplaza a la derecha de los bits de ‘c' tantas posiciones como indique ‘a‘ y asigna. Se rellena por la izquierda con ceros.	c >>>= a;	c = c >>> a;
&=	Realiza operación AND bit a bit y asigna	c &= a;	c = c & a;
|=	Realiza operación OR bit a bit y asigna	c |= a	c = c | a;
^=	Realiza operación XOR bit a bit y asigna	c ^= a	c = c ^ a;
~	Realiza operación NOT bit a bit y asigna	~a	

Operadores Booleanos

Operador	Operación que realiza
&&	Y lógico (AND) Conjunción
||	O lógico (OR) Disyunción
!	NO lógico (NOT) Negación

a	b	a&&b
true	true	true
true	false	false
false	true	false
false	false	false

a	b	a||b
true	true	true
true	false	true
false	true	true
false	false	false

a	!a
true	false
false	true

- Todos los operandos son Boolean y el resultado de las operaciones también

Operadores bit a bit

Operador	Operación que realiza	Ejemplo	Resultado
&	Desplazamiento AND	8 & 3	0
|	Desplazamiento OR	8 | 3	11
^	Desplazamiento XOR	8 ^ 3	11
~	Desplazamiento NOT	~5	-5-1=-6

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

a	b	a ^ b
0	0	0
1	0	1
0	1	1
1	1	0

- Todos los operandos son enteros cuya representación binaria tiene 32 bits de longitud (4bytes)

Operadores bit a bit

Operador	Operación que realiza	Ejemplo	Resultado
<<	Desplazamiento a la izquierda	8 << 3	64
>>	Desplazamiento a la derecha	8 >> 3	1
>>>	Desplazamiento a la derecha rellenando con ceros	8 >>> 3	1

- Todos los operandos son enteros cuya representación binaria tiene 32 bits de longitud (4bytes)

Operadores de objeto

Operador	Operación que realiza	Ejemplo
.	Permite acceder a las propiedades o métodos de un objeto	objeto.propiedad objeto.metodo()
[ ]	Permite enumerar miembros de un objeto	a[1], a[2]
delete	Elimina un elemento de una colección	delete a[1]
in	Inspecciona propiedades o métodos de un objeto	“write” in document
instanceof	Comprueba si un objeto es una instancia de un objeto nativo de JavaScript	a instanceof Array
new	Accede a constructor de objetos incorporados en el núcleo de JavaScript	var hoy = new Date();
this	Accede a la referencia del propio objeto	this.value

Operadores misceláneos

Operador	Operación que realiza	Ejemplo
,	Indicar una serie de expresiones que van a ser evaluadas en secuencia, de izquierda a derecha	var a, b, c;
? :	Operador condicional. Forma reducida del if ...else condición ? expresión_si_verdadera : expresión_si_falsa	a>b?true:false
typeof	Devuelve el tipo de valor de una variable o expresión (number, string, boolean, object, function, undefined)	typeof miVar == "number"

OPERADORES

PRECEDENCIA DE OPERADORES

	Aritméticos	Relacionales	Lógicos	Bit a bit	Otros
Alta					()(función) [](vector) .
	++ -- * / % + -		!	~	
				<< >>	
		< <= > >=			
		!= ==			
				&	
				^	
				|	
			&&		
			||		
					?:
				<<= >>=	= += -= *=
				^= |= &=	/= %=
Baja					,

Autoevaluación

- Si sumamos 9+4+"10" en JavaScript obtendremos:
  a) “23”
  b) 23
  c) “1310”

¿Cuál es el resultado de las siguientes operaciones?

a) !true
b) !(10 > 5)
c) !(“gato” == “pato”)
d) 5 > 1 && 50 >10

Fundamentos de JavaScript

- Condiciones y bucles
  - Estructuras de control
    - if
    - if-else
  - Bucles
    - for
    - while
    - do-while

Sangría correcta del código

Condicionales

Sintaxis 1

```
if (condición)
{
   //sentencias
}
```

```
if (condición) {
  //sentencias
}
```

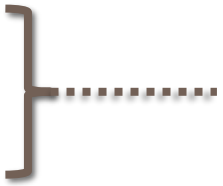
Sintaxis 2

```
if (condición)
{
   //sentencias
}
else
{
   //sentencias
}
```

```
if (condición) {
   //sentencias
} else {
   //sentencias
}
```

Sintaxis 3 (anidamiento)

```
if (condición) {
   //sentencias
} else if (condición) {
   //sentencias
} else [if (condición) {
   //sentencias
} else ]{
   //sentencias
}
```



Condicionales (forma abreviada)

(condición) ? sentencias_verdad : sentencias_falso

Equivalente a:

```
if (condición) {
//sentencias_verdad
} else {
//sentencias_falso
}
```

Condicionales (otra sintaxis)

```
if (condición) sentencia
else  sentencia;
```

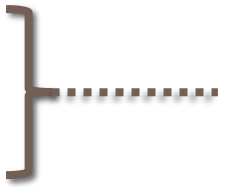
```
///o bien...
if (condición)
sentencia;
else
sentencia;
```

solo en el caso de que haya una sola sentencia se pueden omitir las llaves de apertura y cierre

Condicionales

Para evitar un anidamiento excesivo de if ... else if ...

```
switch (expresión) {
  case valor1:
     //sentencias_si expresión=valor1
     break;
  case valor2:
     //sentencias_si expresión=valor2
     break;
 [case valor??:
     //sentencias_si expresión=valor??
     break;]
  default:
     //sentencias a ejecutar en cualquier otro caso
}
```



Bucles: FOR

```
for (expresión_inicial; condición; incremento)
{
    // Instrucciones a ejecutar dentro del bucle.
}
```

- Mientras **condición** sea true se ejecuta el cuerpo del bucle.
- Al final del bucle se realiza el **incremento** del contador

OTRA FORMA DE ESCRIBIRLO:

```
for (expresión_inicial; condición; incremento) {
    // Instrucciones a ejecutar dentro del bucle.
}
```

Bucles: FOR

OTRA FORMA DE ESCRIBIRLO:

```
for (expresión_inicial; condición; incremento)
    sentencia;
```

solo en el caso de que haya una sola sentencia se pueden omitir las llaves de apertura y cierre

Bucles: FOR (ejemplos)

```
for (var i=5; i<8; i++) {
    document.write("la variable i vale ahora "+i+"<br />");
}

var i=5;
for (; i<8; i++) {  //omitimos el primer argumento del for
    document.write("la variable i vale ahora "+i+"<br />");
}

var i=5;
for (; i<8;) {  //omitimos el primer y el último argumento del for
    document.write("la variable i vale ahora "+(i++)+"<br />");
}
```

Bucles: FOR (para objetos y colecciones)

```
for (var variable in objeto) {
    //sentencias que trabajan con variable
}
```

solo en el caso de que haya una sola sentencia se pueden omitir las llaves de apertura y cierre

variable toma el valor de cada propiedad del objeto

Ejemplo:

```
var persona = {
        nombre: “Pedro”,
        apellidos: “García López”,
        edad:40
    }
for (var dato in persona) {
   document.write(dato+” = “ + persona[dato]+”<br />”);
}
```

Bucles: WHILE

```
while (condición)
{
    // Instrucciones a ejecutar dentro del bucle.
}
```

solo en el caso de que haya una sola sentencia se pueden omitir las llaves de apertura y cierre

- Mientras **condición** sea true se ejecuta el cuerpo del bucle.
- Si **condición** no se cumple la primera vez ya no entra.

OTRA FORMA DE ESCRIBIRLO:

```
while (condición) {
    // Instrucciones a ejecutar dentro del bucle.
}
```

Bucles: DO WHILE

```
do
{
    // Instrucciones a ejecutar dentro del bucle.
} while (condición)
```

solo en el caso de que haya una sola sentencia se pueden omitir las llaves de apertura y cierre

- El bucle se ejecuta al menos una vez-
- Mientras **condición** sea true se ejecuta de nuevo el cuerpo del bucle.

OTRA FORMA DE ESCRIBIRLO:

```
do {
    // Instrucciones a ejecutar dentro del bucle.
}  while (condición)
```

break y continue

break; //para terminar la ejecución de un bucle

/* se emplea en el switch siempre para evitar que cuando entra en un caso siga con las instrucciones de los demás */

continue; // para pasar a la siguiente iteración de un bucle

```
for (var i=5; i<=40; i++) {
    if ((i%5)!=0) {
       continue;
    } else {
       document.write(i+” es múltiplo de 5<br />”);
    }
}
```

```
for (var i=5; i<=40; i++) {
    if ((i%5)==0) {
       document.write(i+” es múltiplo de 5<br />”);
    }
}    //esto es más correcto
```

Ejercicio

- Utilizando la función de JavaScript alert() que vale para visualizar una ventana conteniendo un mensaje (de texto o el resultado de la ejecución de alguna operación) y de la cual se muestra un ejemplo a continuación, se pide:
  - Visualizar en la página: I'm a student of DAW2
  - El título de la página será: Probando los caracteres de escape.
  - Debe ser válido según el validador W3C.

Ejemplo:

alert("Hola");

El resultado de la ejecución sería►

Enlaces recomendados

ECMA

Tutorial JavaScript

Desplazamiento de bits

Operador a nivel de bits (Wikipedia)

Ejemplos de operadores