ESTRUCTURA DEL LENJUAJE JAVASCRIPT

Unidad 2

Lenguaje en Entorno cliente

1.1. ¿Qué es JavaScript?

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.

Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java.

1.2. Cómo incluir JavaScript en documentos XHTML

La integración de JavaScript y XHTML es muy flexible, ya que existen al menos tres formas para incluir código JavaScript en las páginas web.

- **Incluir JavaScript en el mismo documento XHTML.**

  El código JavaScript se encierra entre **etiquetas <script>** y se incluye en cualquier parte del documento. Aunque es correcto incluir cualquier bloque de código en cualquier zona de la página, se recomienda definir el código JavaScript dentro de la cabecera del documento (dentro de la etiqueta <head>):

- **Definir JavaScript en un archivo externo.**

  Las instrucciones JavaScript se pueden incluir en un archivo externo de tipo JavaScript que los documentos XHTML enlazan mediante la etiqueta <script>.

  <script type="text/javascript" src="/js/codigo.js"></script>

- **Incluir JavaScript en los elementos XHTML.**

  Este último método es el menos utilizado, ya que consiste en incluir trozos de JavaScript dentro del código XHTML de la página:.

1.3. Sintaxis

- **No se tienen en cuenta los espacios en blanco y las nuevas líneas**.
- **Se distinguen las mayúsculas y minúsculas**:
- **No se define el tipo de las variables**: al crear una variable, no es necesario indicar el tipo de dato que almacenará.
- **No es necesario terminar cada sentencia con el carácter de punto y coma (;)**: Aunque JavaScript no obliga a hacerlo, es conveniente.
- **Se pueden incluir comentarios**:

  El comentario de una sola línea: se definen añadiendo dos barras oblicuas (//) al principio de la línea.

  ```
  // a continuación se muestra un mensaje
  alert("mensaje de prueba");
  ```

  Los comentarios multilínea se definen encerrando el texto del comentario entre los símbolos /* y */.

1.2. Variables

- Las variables en JavaScript se crean mediante la palabra reservada **var.**
- La palabra reservada var solamente se debe indicar al definir por primera vez la variable, lo que se denomina **declarar** una variable.
- Si cuando se declara una variable se le asigna también un valor, se dice que la variable ha sido **inicializada.**

```
var numero_1;
var numero_2;
numero_1 = 3;
numero_2 = 1;
var resultado = numero_1 + numero_2
```

- Se pueden utilizar variables que no se han definido pero se recomienda declarar todas las variables que se vayan a utilizar

anteriormente mediante la palabra reservada `var`. El ejemplo anterior también es correcto en JavaScript de la siguiente forma:

```
var numero_1 = 3;
var numero_2 = 1;
resultado = numero_1 + numero_2
```

El nombre de una variable también se conoce como **identificador** y debe cumplir las siguientes normas:

- Sólo puede estar formado por letras, números y los símbolos $ (dólar) y _ (guión bajo).
- El primer carácter no puede ser un número.

1.3. Tipos de datos

Tipos de datos soportados por JavaScript		
Tipo	**Ejemplo**	**Descripción**
Cadena.	"Hola mundo"	Una serie de caracteres dentro de comillas dobles.
Número.	9.45	Un número sin comillas dobles.
Boolean.	true.	Un valor verdadero o falso.
Null.	null.	Desprovisto de contenido, simplemente es un valor null.
Object.		Es un objeto software que se define por sus propiedades y métodos (los arrays también son objetos).
Function.		La definición de una función.

1.3.1.- Conversiones de tipos de datos.

- Para convertir cadenas a números dispones de las funciones: **parseInt()** y **parseFloat()**.

  Por ejemplo:

  parseInt("34") // resultado = 34

  parseInt("89.76") // resultado = 89

  parseFloat devolverá un entero o un número real según el caso:

  parseFloat("34") // resultado = 34

  parseFloat("89.76") // resultado = 89.76

  4 + 5 + parseInt("6") // resultado = 15

- Si lo que deseas es realizar la conversión de números a cadenas, es mucho más sencillo, ya que

simplemente tendrás que concatenar una cadena vacía al principio, y de esta forma el número será

convertido a su cadena equivalente:

("" + 3400) // resultado = "3400"

("" + 3400).length // resultado = 4

1.4. Operadores

Categorías de operadores en JavaScript	
Tipo	**Qué realizan**
Comparación.	Comparan los valores de 2 operandos, devolviendo un resultado de true o false (se usan extensivamente en sentencias condicionales como `if...` `else` y en instrucciones `loop`). `==` `!=` `===` `!==` `>` `>=` `<` `<=`
Aritméticos.	Unen dos operandos para producir un único valor que es el resultado de una operación aritmética u otra operación sobre ambos operandos. `+` `-` `*` `/` `%` `++` `- -` `+valor` `-valor`
Asignación.	Asigna el valor a la derecha de la expresión a la variable que está a la izquierda. `=` `+=` `-=` `*=` `/=` `%=` `<<=` `>=` `>>=` `>>>=` `&=` `|=` `^=` `[]`
Boolean.	Realizan operaciones booleanas aritméticas sobre uno o dos operandos booleanos. `&&` `||` `!`
Bit a Bit.	Realizan operaciones aritméticas o de desplazamiento de columna en las representaciones binarias de dos operandos. `&` `|` `^` `~` `<<` `>>` `>>>`
Objeto.	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. `.` `[]` `()` `delete` `in` `instanceOf` `new` `this`
Misceláneos.	Operadores que tienen un comportamiento especial. `,` `?:` `typeof` `void`

1.4.1.- Operadores de comparación.

Operadores de comparación en JavaScript			
Sintaxis	**Nombre**	**Tipos de operandos**	**Resultados**
==	Igualdad.	Todos.	Boolean.
!=	Distinto.	Todos.	Boolean.
===	Igualdad estricta.	Todos.	Boolean.
!==	Desigualdad estricta.	Todos.	Boolean.
>	Mayor que .	Todos.	Boolean.
>=	Mayor o igual que.	Todos.	Boolean.
<	Menor que.	Todos.	Boolean.
<=	Menor o igual que.	Todos.	Boolean.

1.4.2.- Operadores aritméticos.

Operadores aritméticos en JavaScript			
Sintaxis	**Nombre**	**Tipos de Operando**	**Resultados**
+	Más.	integer, float, string.	integer, float, string.
-	Menos.	integer, float.	integer, float.
*	Multiplicación.	integer, float.	integer, float.
/	División.	integer, float.	integer, float.
%	Módulo.	integer, float.	integer, float.
++	Incremento.	integer, float.	integer, float.
--	Decremento.	integer, float.	integer, float.
+valor	Positivo.	integer, float, string.	integer, float.
-valor	Negativo.	integer, float, string.	integer, float.

1.4.3.- Operadores de asignación.

Operadores de asignación en JavaScript			
Sintaxis	**Nombre**	**Ejemplo**	**Significado**
=	Asignación.	x = y	x = y
+=	Sumar un valor.	x += y	x = x + y
-=	Substraer un valor.	x -= y	x = x - y
***=**	Multiplicar un valor.	x *= y	x = x * y
/=	Dividir un valor.	x /= y	x = x / y
%=	Módulo de un valor.	x %= y	x = x % y
<<=	Desplazar bits a la izquierda.	x <<= y	x = x << y
>=	Desplazar bits a la derecha.	x >= y	x = x > y
>>=	Desplazar bits a la derecha rellenando con 0.	x >>= y	x = x >> y
>>>=	Desplazar bits a la derecha.	x >>>= y	x = x >>> y
&=	Operación AND bit a bit.	x &= y	x = x & y
|=	Operación OR bit a bit.	x |= y	x = x | y
^=	Operación XOR bit a bit.	x ^ = y	x = x ^ y
[]=	Desestructurando asignaciones.	[a,b]=[c,d]	a=c, b=d

1.4.4.- Operadores booleanos.

Operadores de boolean en JavaScript			
Sintaxis	**Nombre**	**Operandos**	**Resultados**
&&	AND.	Boolean.	Boolean.
||	OR.	Boolean.	Boolean.
!	Not.	Boolean.	Boolean.

Tabla de valores de verdad del operador AND			
Operando Izquierdo	**Operador AND**	**Operando Derecho**	**Resultado**
True	&&	True	True
True	&&	False	False
False	&&	True	False
False	&&	False	False

Tabla de valores de verdad del operador OR			
Operando Izquierdo	**Operador OR**	**Operando Derecho**	**Resultado**
True	||	True	True
True	||	False	True
False	||	True	True
False	||	False	False

1.4.5.- Operadores bit a bit.

Tabla de operador Bit a Bit en JavaScript			
Operador	**Nombre**	**Operando izquierdo**	**Operando derecho**
&	Desplazamiento AND.	Valor integer.	Valor integer.
|	Desplazamiento OR.	Valor integer.	Valor integer.
^	Desplazamiento XOR.	Valor integer.	Valor integer.
~	Desplazamiento NOT.	(Ninguno).	Valor integer.
<<	Desplazamiento a la izquierda.	Valor integer.	Cantidad a desplazar.
>>	Desplazamiento a la derecha.	Valor integer.	Cantidad a desplazar.
>>>	Desplazamiento derecha rellenando con 0.	Valor integer.	Cantidad a desplazar.

1.4.6.- Operadores de objeto.

El siguiente grupo de operadores se relaciona directamente con objetos y tipos de datos. La mayor parte de ellos fueron implementados a partir de las primeras versiones de JavaScript, por lo que puede haber algún tipo de incompatibilidad con navegadores antiguos.

- ***. (punto)*** El operador punto, indica que el objeto a su izquierda tiene o contiene el recurso a su derecha, como por ejemplo: **objeto.propiedad** y **objeto.método()**.

```
var s = new String('rafa');
var longitud = s.length;
var pos = s.indexOf("fa");        // resultado: pos = 2
```

- ***[] (corchetes para enumerar miembros de un objeto).***

  var a =["Santiago","Coruña", "Lugo"];

  a[1] = "Coruña";

- ***Delete (para eliminar un elemento de una colección).***

  var oceanos = new Array("Atlantico", "Pacifico", "Indico","Artico");

  Podríamos hacer:

  delete oceanos[2];

- *In (para inspeccionar métodos o propiedades de un objeto)*
- *instanceof (para comprobar si un objeto es una instancia de un objeto nativo de JavaScript).*

```
a = new Array(1,2,3);
a instanceof Array; // devolverá true.
new (para acceder a los constructores de objetos incorporados en el núcleo de JavaScript).
```

1.4.7.- Operadores misceláneos.

- ***El operador coma ,***

  Ejemplo:

  var nombre, direccion, apellidos, edad;

- ***? : (operador condicional)***

Este operador condicional es la forma reducida de la expresión **if …. else.**

Si usamos esta expresión con un operador de asignación:

Var = condicion ? expresión si se cumple la condición: expresión si no se cumple;

- ***typeof (devuelve el tipo de valor de una variable o expresión).***

Este operador unario se usa para identificar cuando una variable o expresión es de alguno de los siguientes tipos: **number**, **string**, **boolean**, **object**, **function** o **undefined**.

Ejemplo:if (typeof miVariable == "number")

{miVariable = parseInt(miVariable);}

1.5.- Condiciones y bucles.

1.5.1.- Estructuras de control

Construcción if

```
if  (condición)     // entre paréntesis irá la condición que se evaluará a true o false.
 {
     // instrucciones a ejecutar si se cumple la condición
 }
```

Construcción if … else

```
if  (condición)     // entre paréntesis irá la condición que se evaluará a true o false.
{
     // instrucciones a ejecutar si se cumple la condición
}
else
{
     // instrucciones a ejecutar si no se cumple la condición
}
```

1.5.2.- Bucles.

Bucle for.

Este tipo de bucle te deja repetir un bloque de instrucciones un número limitado de veces.

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

Bucle while().

Este tipo de bucles se utilizan cuando queremos repetir la ejecución de unas sen na condición.

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

Bucle do … while().

El tipo de bucle **do…while** es la última de las estructuras para implementar repeticiones de las que dispone JavaScript, y es una variación del bucle **while()** visto anteriormente. Se utiliza generalmente, cuando no sabemos el número de veces que se habrá de ejecutar el bucle. Es prácticamente igual que el bucle **while()**, con la utará una vez.

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