Tema 2. Estructura del lenguaje javascript.

# 1.- Fundamentos de javascript.

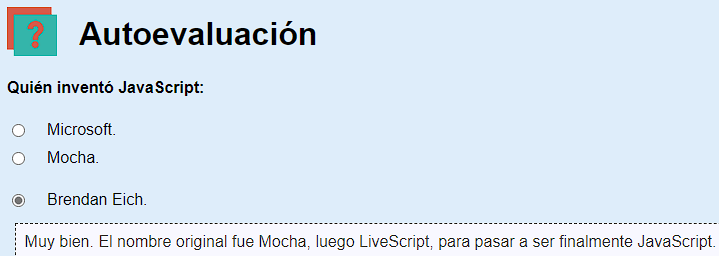
JavaScript es el lenguaje que estudiaremos. Seguro has oído otros nombres similares como JScript (que es el nombre que le dio Microsoft a este lenguaje).

Microsoft le dio el nombre de JScript para evitar problemas relacionados con la marca, pero no pudo evitar otros problemas surgidos por las incompatibilidades que su versión de JavaScript tenía con múltiples navegadores.

Para evitar esas incompatibilidades, **el W3C, diseñó el DOM** (Modelo de Objetos del Documento). A partir de 1997 **este lenguaje se rige por un estándar denominado ECMA**, y en concreto, por el documento de especificaciones ECMA-262. Tanto JavaScript como Script son compatibles con el estándar ECMA-262.

**JavaScript se diseñó con una sintaxis similar al lenguaje C** y aunque adopta nombres y convenciones del lenguaje **Java**, éste último **no tiene relación con JavaScript ya que tienen semánticas y propósitos diferentes**.

JavaScript fue desarrollado originariamente por Brendan Eich, de Netscape, con el nombre de **Mocha**, el cual se renombró posteriormente a **LiveScript** y quedó finalmente como **JavaScript**.

****: [Brendan Eich - Wikiwand](https://www.wikiwand.com/es/Brendan_Eich)

## 1.1.- Comentarios en el código.

A la hora de programar en cualquier lenguaje de programación, es importante **comentar tu código**. Los comentarios son ignorados por el intérprete de JavaScript, con la ventaja de que **permiten a los desarrolladores dejar notas sobre cómo funcionan las cosas en sus scripts**.

Cuando alguien se descargue vuestro código, necesitará más o menos tiempo, dependiendo del tamaño de vuestro fichero (en este sentido, los comentarios también influyen). A pesar de ello, los comentarios proporcionan muchas más ventajas que inconvenientes. **Comenta lo máximo posible (si es necesario)**.

JavaScript permite **dos estilos de comentarios**:

* **Para comentarios de una sola línea**: doble barra.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **Comentarios más largos**: /\* y \*/

Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Reflexiona**:

* Podemos usar comentarios para desactivar bloques de código (y que JavaScript no los ejecute) sin tener que borrarlos.
* Además, podemos dar nuestra información de contacto para que otros programadores contacten contigo.

## 1.2.- Variables.

* Para trabajar con datos, necesitamos variables. Una variable **es como un cajón que almacena información**.
* **El tiempo que dicha información permanecerá almacenada, dependerá de muchos factores**. En el momento que el navegador limpia la ventana o marco, cualquier variable conocida será eliminada.
* **Las variables en JavaScript pueden almacenar diferentes tipos de valores**, y no tenemos que decirle de qué tipo es una variable u otra.
  + Por ello, a la hora de dar nombres a las variables, **tendremos que poner nombres que realmente describan el contenido de la variable**.
* **Respecto a los nombres de las variables**:
  + No podemos usar palabras reservadas (return, for, etc.).
  + No podemos usar símbolos de puntuación en medio de la variable.
  + No puede contener espacios en blanco.
  + Han de construirse con caracteres alfanuméricos y el carácter subrayado (\_).
    - El nombre de la variable no puede comenzar por un número.
  + Están prohibidos caracteres raros como +, %, $, etc.
* **Hay varias maneras de crear variables en JavaScript**:

|  |  |
| --- | --- |
| **Con la palabra reservada var** | * **Definirla solamente**: var nombreVariable; * **Crearla e inicializarla**: var nombreVariable = valor * **Definir varias variables de seguido**: var nombreVar1, nombreVar2;   + var nombre, edad, teléfono, dni; |
| **Let** | Nos permite definir variables cuyo ámbito será el bloque de instrucciones en el que estén definidas, de forma que, **por ejemplo** el siguiente código produce un error: |
| **Const** | Para definir constantes |
| **Nada** | Si no se pone nada, es decir, si se escribe directamente el nombre de la variable y a continuación, se le asigna su valor, **por defecto, se crea una variable tipo var**, con las consecuencias de visibilidad que ello conlleva. |

* Las palabras reservadas **set y const fueron introducidos en la edición 6 del estándar ECMA-262** (en adelante ES6).

**Para saber más**: información y ejemplos de variables: [JavaScript Variables (w3schools.com)](https://www.w3schools.com/js/js_variables.asp)

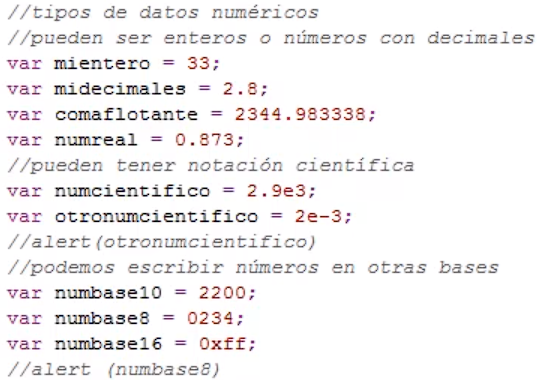
## 1.3.- Tipos de datos.

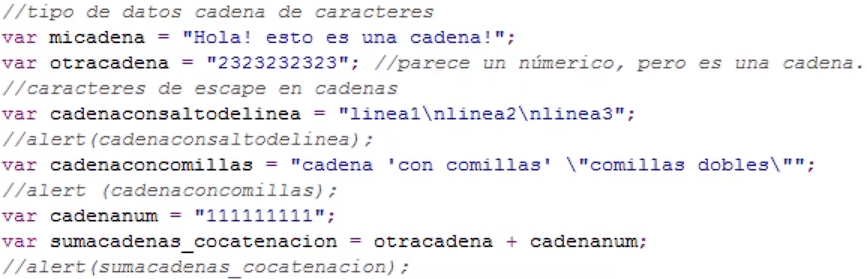
Tipos de datos soportados en JavaScript:

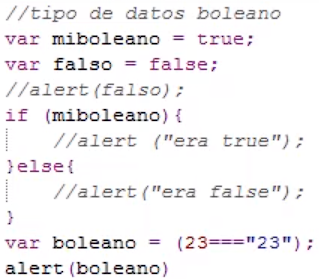
| **Tipos de datos soportados por JavaScript** | | |
| --- | --- | --- |
| **Tipo** | **Ejemplo** | **Descripción** |
| Cadena. | "Hola mundo". | Una serie de caracteres dentro de comillas dobles (o comillas simples). |
| Número. | 9.45 | Un número sin comillas dobles. |
| **Boolean**. | **true**. | 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. |

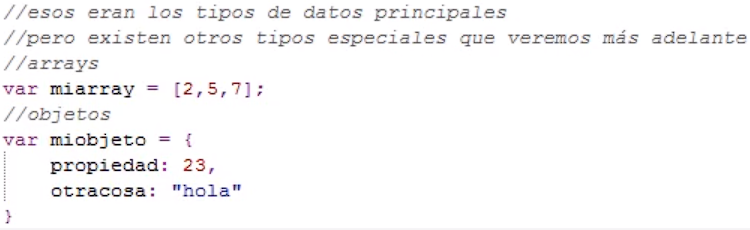
El contener solamente este tipo de datos, simplifica mucho las tareas de programación, especialmente aquellas que abarcan tipos incompatibles entre números.

**Debes conocer**: videotutorial sobre los tipos de datos en JavaScript: [Videotutorial de Javascript parte 8: Tipos de datos en Javascript - YouTube](https://www.youtube.com/watch?v=cw5sFqlDBtE&feature=emb_logo)









### 1.3.1.- Conversiones de tipos de datos.

Aunque los tipos en JavaScript son muy sencillos, a veces puede suceder que las operaciones no se realizan como uno quiere, debido a la conversión de tipos de datos automática que hace.

**Por ejemplo**:

* 4 + "5" // resultado = "45"
* 4 + 5 + "6" // resultado = "96"

Estos resultados son obtenidos de esta forma porque las expresiones se evalúan de izquierda a derecha. Primero se suma el 4 y el 5, y su resultado (9), se concatena con la cadena “6”, dando como resultado la cadena “96”.

Vamos a ver las **conversiones explícitas**:

* **Convertir cadenas a números:** parseInt() y parseFloat()
  + parseInt() convierte a número solo la parte entera.
  + parseFloat() convierte a número también la parte decimal (si la hubiera).

Texto

Descripción generada automáticamente

* **Conversión de números a cadenas**: basta con concatenar una cadena vacía al principio, para que el nº sea convertido a su cadena equivalente.
  + **En el segundo de los ejemplos que se muestran a continuación**, se muestra la potencia de los paréntesis, ya que nos permiten acceder a las propiedades típicas de cualquier string, con tan solo la expresión que la forma.



## 1.4.- Operadores.

Los operadores son símbolos y palabras que realizan operaciones sobre uno o varios valores, para obtener un nuevo valor. Tipos de operadores para usar en nuestras expresiones:

* **Operadores unarios**: contiene un operando y un operador. Ejemplo: b++
* **Operadores binarios**: contiene dos operandos separados por un operador. Ejemplo: a + b.

|  |  |
| --- | --- |
| **De comparación** | ==: comprueba si dos variables son iguales en contenido |
| !=: comprueba si dos variables son distintas en contenido. |
| ===: comprueba si dos variables son iguales en contenido y tipo. |
| !==: comprueba si dos variables son distintas en contenido y tipo. Aclaración: Si es igual en contenido, pero distinto en tipo, devolverá true. |
| >: comprueba si una variable es mayor que otra. |
| >=: comprueba si una variable es mayor o igual que otra. |
| <: comprueba si una variable es menor que otra. |
| <=: comprueba si una variable es menor o igual que otra. |
| * Para comparar cadenas, JavaScript convierte cada carácter de la cadena en su correspondiente valor ASCII. * Cada letra, comenzando con la primera del operando de la izquierda, se compara con su correspondiente letra en el operando de la derecha. * Los valores ASDCII de las letras mayúsculas, son más pequeños que sus correspondientes en minúsculas, por eso, “Marta” no es mayor que “marta”.     funciona como , es decir, internamente, se convierte la cadena “123 a número para poder comparar los datos. |
| **Aritméticos** | +, -, \*, /, %, ++, --, +valor, -valor      Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico  Descripción generada automáticamente |
| **Asignación** | Le asigna a la variable (de la izquierda) el valor de la derecha de la expresión.  =, +=, -=, \*=, /=, %=, <<=, >=, >>=, >>>=, &=, |=, ˆ =, [] |
| **Aclaraciones**:   * **Desplazar bits a la izquierda**: x <<= y es equivalente a x = x << y * **Desplazar bits a la derecha**: x >= y es equivalente a x = x > y * **Desplazar bits a la derecha rellenando con 0**: x >>= y es equivalente a x = x >> y * **Desplazar bits a la derecha**: x >>>= y es equivalente a x >>> y * **Operación AND bit a bit**: x &= y es equivalente a x = x & y * **Operación OR bit a bit**: x |= x | y * **Operación XOR bit a bit**: x ^= y es equivalente a x = x ^ y * **Desestructurando asignaciones**: []=.   + **Por ejemplo**: [a,b] = [c,d] es equivalente a a = c y b = d. |
| **Boolean** | Realiza operaciones booleanas aritméticas sobre uno o dos operandos booleanos.  &&, ||, ! |
| **Bit a bit** | Realiza operaciones aritméticas o de desplazamiento de columna en las representaciones binarias de dos operandos.  **&, |, ˆ, ∼, <<, >>, >>>** |
| Este tipo de operaciones son un tema avanzado para cuestiones del lado del cliente, solo si trabajas en procesos externos de aplicaciones del lado del servidor, los necesitarás. |
| * **Bit a bit AND (&)**: 1 si ambos dígitos son 1. * **Bit a bit OR (|)**: 1 si cualquiera de los dos dígitos es 1. * **Bit a bit XOR (^)**: 1 si solo un dígito es 1. * **Desplazamiento a la izquierda <<**: especifica el nº de bits a desplazar hacia la izquierda. * **Desplazamiento a la derecha >>**: especifica el nº de bits a desplazar a la derecha. * **Desplazamiento derecha rellenando con 0 >>>**: duda. |
| **Por ejemplo**:  4 << 2 // resultado = 16  El número 4 en binario es 00000100, al decirle que desplace el 1 a la izquierda dos posiciones, tenemos que el número es 00010000, el cual se corresponde en valor decimal al 16. |
| **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** |
| **Ejemplo con el punto**: |
| **Corchetes**: sirven para enumerar miembros de un objeto:   * **Crear un array**: var a =["Santiago","Coruña", "Lugo"]; * **Enumerar un elemento de un array**: a[1] = "Coruña"; * **Enumerar una propiedad de un objeto**: a["color"]= "azul"; |
| **Delete**: sirve para eliminar un elemento de una colección.  **Por ejemplo**, si consideramos:  var oceanos = new Array("Atlantico", "Pacifico", "Indico","Artico");  **Podríamos hacer**:  delete oceanos[2];  // **Esto eliminaría el tercer elemento del array ("Indico"), pero la longitud del array no cambiaría. Si intentamos referenciar esa posición oceanos[2]**  **obtendríamos undefined**. |
| **In**: sirve para inspeccionar métodos o propiedades de un objeto. Si el objeto conoce la propiedad o método, la expresión devolverá true. **Ejemplos**: |
| **Instanceof**: para comprobar si un objeto es una instancia de un objeto nativo de JavaScript. **Ejemplo**:  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. **Ejemplo**:  var hoy = new Date();  **// creará el objeto hoy de tipo Date() empleando el constructor por defecto de dicho objeto.** |
| **This**: para hacer referencia al propio objeto en el que estamos localizados.  **Ejemplo**: nombre.onchange = validateInput; |
| **Misceláneos** | Operadores que tienen un comportamiento especial.  La coma, ?:, typeof |
| **La coma**: sirve para indicar una serie de expresiones que van a ser evaluadas en secuencia, de izquierda a derecha. **Por ejemplo**, se usa para declarar múltiples variables a la vez:    También dentro de los bucles for, para definir varios contadores: |
| **? (operador condicional)**: es igual que el if…else.  **Su sintaxis es**: condicion ? expresión si se cumple la condición: expresión si no se cumple;  Si usamos esta expresión con un operador de asignación (que es como normalmente utilizaremos este operador), sería así:    **Ejemplo**: |
| **Typeof**: devuelve el tipo de valor de una variable o expresión.  Devuelve uno de los siguientes tipos: number, string, boolean, object, function o undefined.  **Ejemplo**: |

**Para saber más**: [información y ejemplos de operadores](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions_and_Operators). (Mozilla)

**Para saber más**: [información y ejemplos de operadores](https://www.w3schools.com/js/js_operators.asp). (W3Schools)

**Para saber más**: [operaciones a nivel de bits](http://www.mundoprogramacion.com/net/dotnet/operar_con_bits.aspx). (No funciona el enlace)

## 1.5.- Condiciones y bucles.

Las condiciones sirven para tomar decisiones en función de valores. Los bucles sirven para repetir acciones en función de si se cumple cierta condición o no.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

### 1.5.1.- Estructuras de control.

|  |  |
| --- | --- |
| **If** |  |
| **Ejemplo**: |
| **If … else** |  |
| **Ejemplo**: |

**Para saber más**: [de if…else](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Control_flow_and_error_handling#Sentencia_if...else).

### 1.5.2.- Bucles.

|  |  |
| --- | --- |
| **For** |  |
| **Ejemplo**: |
| **While** |  |
| **Ejemplo**: |
| **Do… while()** |  |
| **Ejemplo**: |

### 1.5.3.- Ejemplo sencillo con JavaScript.

Aquí se muestra el código fuente, de un pequeño ejemplo de una aplicación en JavaScript.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>Ejemplo Básico de JavaScript</title>

</head>

<body>

<h2>Código fuente de un ejemplo básico con JavaScript

</h2>

<script type="text/javascript">

// Definimos variables.

var nombre, apellidos, edad, hermanos;

nombre="Rafa";

apellidos="Martinez Díaz";

edad=38;

hermanos=3;

// Imprimo el nombre y los apellidos.

document.write("Hola " + nombre + " " + apellidos);

// Imprimo un salto de línea.

document.write("<br/>");

// Imprime la edad y el número de hermanos.

document.write(nombre + " tienes " + edad + " años y además tienes " + hermanos + " hermanos.<br/>");

// Fíjate en la diferencia entre las dos siguientes lineas.

document.write("Dentro de 15 años tu edad será " + edad + 15 + ".<br/>");

document.write("Dentro de 15 años tu edad será " + (edad+15) + ".<br/>");

// Tu nombre escrito 50 veces.

for (i=1; i<=50; i++)

{

document.write(nombre + ",");

}

</script>

</body>

</html>

# Ejemplo de las tutorías:

* let nombre = prompt(‘Escribe tu nombre’) -> permite crear una ventana que solicite un dato en una caja de texto.
* document.write() -> permite escribir texto en el documento HTML.
* Console.log() -> escribe mensajes por la consola de JavaScript.
* Etiqueta br: añade retorno de carro.
* Etiqueta p: añade retorno de carro (separación de párrafos)
* [Can I use](https://caniuse.com/): página que permite saber que etiquetas JavaScript podemos usar sin perder compatibilidad con navegadores.
* Tipos de cadenas string:



Es equivalente a:



* **Conversión de tipos**:
  + parseInt y number sirven para pasar cadenas a números.
    - También sirve poner un + delante de la variable cadena que queramos convertir.