# JavaScript

Editores de Código:

* Notepad++ (<http://notepad-plus-plus.org/>)
* Aptana (<http://www.aptana.com/products> )
* Kompozer (<http://kompozer.net>) Se trata de un software integrado en la categoría deIDE (*Integrated Development Environment*)
* Adobe Dreamweaver es otro generador de código IDE

**Input:** Declare variables and functions

**Process:** Use the variables and functions

**Output:** Save and reload the browser

El código JavaScipt, almacenado como archivos en un servidor web, es accesible, a través del protocolo HTTP, por el navegador del “lado del cliente”, usando una llamada del usuario a través de la introducción de una URL o con un click en un enlace de hipertexto El código JavaScript es no compilado y es interpretado por un motor dedicado, incorporado al navegador.

|  |  |  |
| --- | --- | --- |
| JavaScript VS JAVA | JavaScript | JAVA |
| Autor | Desarrollado por Netscape Communications | Desarrollado por Sun Microsystems |
| Formato | Interpretado | Compilado como de bte-code |
| Almacenamiento | Código insertado en la página web | Applet descargado como un elemento de la página web |
| Uso | Se usa para “dinamizar” las páginas web | Se usa para desarrollar todo tipo de aplicaciones |
| Ejecución | Se ejecuta por el navegador o por Node JS (motor V8) | Se ejecuta en la JVM (*Java Virtual Machine*) del navegador |
| Programación orientada a objetos (ON) | Manipula objetos | Orientado a objetos |
| Tipado | Sin control de tipo (débilmente tipado: POO con prototype) | Fuertemente tipado (POO con clases) |

JavaScript es un código Web Interpretado: hay que ejecutarlo (refresh). JavaScript no es un código fuertemente tipado porque no es necesario especificar el tipo de dato que contiene la variable, sino que se determina a posteriori: “lenguaje con relajación de tipos”.

Las secuencias de código JavaScript son, la mayor parte de las veces, para animar los scripts HTML. HTML describe el contenido, JavaScript gestiona la interactividad.

JavaScript está orientado al “lado cliente”. Los lenguajes Back actúan desde el “lado del servidor” preguntando a las bases de datos y generando sobre la marcha el código HTML. Posteriormente, se envía al navegador del “lado cliente” usando el protocolo HTTP.

Un algoritmo es la descripción de las operaciones necesarias para obtener un resultado a partir de valores de entrada: “los datos”. Un programa es un algoritmo escrito en un lenguaje concreto. La programación va a ser el medio de definir y especificar al ordenador el conjunto de operaciones necesarias para resolver un problema. Se trata de la traducción a un lenguaje comprensible por la máquina de las secuencias de acciones que se deben aplicar a los datos, en el marco de un procesamiento informático.

**<!DOCTYPE html>  
<html>  
<head>  
 <title>JavaScript Project</title>** //¿versions antiguas? **<meta http-equiv=”Conteny-Type” content=”text/html” >  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />**

**</head>  
<body>**

**<script>**

Indicar que no hay que guardar la página en el caché del navegador (la página no se guardará en los archivos temporales del usuario), con la palabra clave **pragma**

**<meta http-equiv=”pragma” content=”no-cache” />**

Indicar que, pasada una fecha de expiración, la página a no estará disponible, con la palabra clave  **expires:**

**<meta http-equiv=”expires” content=”Tue, 20 Aug 218 14:25:26 GMT” />**

Requerir la actualización de la página cada n segundos, con la palabra clave **refresh**. Ej: 15 seg:

**<meta http-equiv=”refresh” content=”15” />**

Dirigir al internauta a otra página después de n segundos, con la palabra clave **refresh**

**<meta http-equiv=” refresh” content=”20;URL=https://www.google.es/” />**

## Variables

Durante el análisis de un problema, a menudo tenemos que descomponer este en subproblemas, que se deben resolver en un orden secuencial concreto. Cada subproblema genera resultados que se pueden utilizar en los subproblemas que siguen. Por lo tanto, para poder manipular estos resultados, conviene asignarles un nombre. Estas **variables de memoria** se suelen escribir con mayúsculas por convenio. La nomenclatura se hace a través de un **identificador.** A cada variable se le asocia un identificador (nombre) y el **tipo** (conjunto al que pertenece el valor que designa)

Las variables numéricas requieren cuatro bites, mientras que las booleanas solo requiEren 1. El dominio de una variable es el tipo de dato que contiene:

1. Números
   * Enteros ( int ): **var myInteger = 3;**
   * Racionales ( float ): **var** **myFloat** **=** **3.14**;
2. Cadenas ( string ): **var** **myString** **=** **"Hello World"**;
3. Lógicos ( boolean ): **var** **myBoolean** **=** **true**;
4. **null**
5. **undefined**

Las variables primero se declaran (INPUT) y después se llaman (PROCESS). El valor de una variable se puede inicializar en la declaración, en el process, o también se puede inicializar introduciendo su valor con el teclado mediante el método **prompt:**

**variableName = prompt( “variableValue” )**

## Operadores

Una operación es una acción que ocurre entre dos operandos a través de un operador (por ejemplo, una concatenación, una suma o una comparación). Una expresión es un conjunto de una o más operaciones, cuyos operandos pueden ser a su vez expresiones (por ejemplo, concatenar un string con el resultado de una suma entre paréntesis).

El operador de asignación es **=** pero el de comparación es **==**

* Aritméticos
  + **+** Addition
  + **-** Substraction
  + **\*** Multiplication
  + **\*\*** Potentiation
  + **/** Division
  + **%** Modulus (Division remainder)
  + **++** Increment
  + **–** Decrement

Si el operador ++ se indica como prefijo del identificador de la variable, su valor se incrementa **antes** de realizar cualquier otra operación. Si el operador ++ se indica como sufijo del identificador de la variable, su valor se incrementa **después** de ejecutar la sentencia en la que aparece.

* String Interpolation : dos opciones: con el **+** o con bacticks (``) y envolviendo variables
  + **“string”** **+** **var console.log(`String ${var1}. Continue string ${var2}.`)**
* De Comparación *(es recomendable poner primer el valor, luego el operador de comparación y después la variable u objeto)*
  + **==** equal to 🡪 (true regardless the type)
  + **===** equal value and equal type
  + **!=** not equal
  + **!==** not equal value or not equal type
  + **>** greater than
  + **<** less than
  + **>=** greater than or equal to
  + **<=** less than or equal to
* Lógicos
  + Negación: **!** Conjunción: **&&** Disyunción: **||**

## Estructuras de control de flujo

Un condicional es una ECF que dirige el flujo de ejecución según el resultado de evaluar una condición (generalmente una comparación lógica entre dos elementos).

Switch: es un condicional múltiple que evita la pirámide de la muerte (pyramid of mood). Es una forma de abreviar los if else, al igual que el for es una forma de abreviar el while. El switch se utiliza para cosas que son mutuamente excluyentes utilizando el break. El último case de switch no se llama case sino default. Switch case: switch ( selectedMonth ) {blocks of code}

**switch( variable )  
{**

**case 0:   
outputMonth = months[0];  
break**;

**case 1:  
outputMonth = months[1];  
break;**

**default:  
outputMonth = "ERROR";  
break;**

**}**

Un bucle es una ECF que repite iterativamente el flujo de ejecución, y se compone de: una situación inicial, una condición de parada (generalmente una comparación lógica) y un cambio de dicha situación.

Según se definan estos tres elementos, se presentan bucles de tipo iterador (una variable entera que se incrementar o decrementa) o de tipo cerrojo (una variable booleana que invierte su valor).

Todo bucle tiene que tener tres cosas:

* Un iterador con un valor incial
* Una condición de parada
* Un incremento del iterador

El bucle puede definirse con condición de parada o con una variable cerrojo: definimos la variable fuera del bucle, y cuando, dentro del bucle, la variable cambia su valor, el bucle termina.

El bucle **for** se utiliza cuando es conocido el número de iteraciones. Cuando este número de iteraciones (**i**) no es conocido se emplea el bucle **while** (realiza una lógica de acciones mientras que con una comprobación al inicio de la estructura) o el **do while** (realiza una lógica de acciones mientras que con una comprobación al final de la iteración).

* **if (hombre == true) If (hombre)**
* **if (hombre == false) if (hombre != true) if(!hombre)**

## Arrays (tablas)

Un array de dimensión única es una variable en memoria “compuesta”, en la que es posible almacenar datos independientes, de diferentes tipos, con un número: **índice**. El acceso a cada dato de la tabla se hará con el valor de índice (en JavaScript se empieza a contar desde 0).

La declaración se puede hacer en sintaxis literal u con notación orientada a objetos:

**var dias = [“Lunes”, “Martes”, “Miércoles”, “Jueves, “Viernes”, “Sábado”, “Domingo”]; dias [6] = "Sábado"; dias [7] = "Domingo";**

**var días = new Array(“Lunes”, “Martes, “Miércoles”, “Jueves”, “Viernes”);**

Los Arrays o tablas tienen la ventaja de su gran flexibilidad:

* No tienen dimensionamiento a priori (Siempre es posible extender el tamaño)
* Posibilidad de mezclar en un mismo array datos de distintos tipos
* Posibilidad de utilizar array asociativo (cuyos índices se sustituyen por valores de texto)

Con la notación orientada a objetos disponemos de diversos métodos para insertar, eliminar y encontrar elementos en una tabla. También existe métodos de ordenación (**sort**, **reverse**).

It is also possible to access individual characters in a string. This works because JavaScript stores strings in a similar way that it stores arrays.

You can also change elements of an array using their indices.

**.push(***[separator[, limit]]***)** 🡪This method allows us to add items (arguments separated by, commas) to the end of an array.  
**.pop() 🡪**This method removes the last item of an array.

The **.split()** splits a string object into an array of strings by separating the string into substrings, using a specified separator string to determine where to make each split. If an empty string ("") is used as the separator, the string is split between each character.

**array.splice(***start*, *deleteCount*, *item1*, *item2*, ...**)**

***Start*** 🡪 Índice donde se comenzara a cambiar el array (con 0 como origen). Si es mayor a la longitud del array, el punto inicial será la longitud del array. Si es negativo, empezará esa cantidad de elementos contando desde el final.

**deleteCount** 🡪 Si es igual a 0, no se eliminará ningún elemento. En casos como este, se debe especificar, como mínimo, un nuevo elemento. Si **deleteCount** es mayor que el número de elementos restantes desde el índice indicado en **start**, se eliminarán todos los elementos hasta el final del array. Si **deleteCount** se omite, será igual a (arr.length - start).

**item1, item2, …** 🡪 Los elementos que se agregarán al array.

Si no se especifica ningún elemento, **splice()** solamente eliminará elementos del array.

**array.shift()** 🡪 elimina el primer elemento del array y devuelve dicho elemento, de manera que se modifica la longitud del array.

**array.unshift(***element1, … , elementN***)** 🡪 agrega elementos al principio del array y devuelve la nueva longitud longitud del array.

***var*** **nuevoArray = viejoArray.concat(valor1[, valor2[, ...[, valorN]]])**

El método **.concat** crea un nuevo array que consta de los elementos del objeto que lo llama, seguido, en orden de ingreso, por los elementos de cada parámetro (en caso de que el parámetro sea un array), o el parámetro mismo (en caso de que no sea un array). No se aplica de forma recursiva a parámetros con arreglos anidados.

El método **.concat** no altera el array original, ni ninguno de los que fueron ingresados como parámetros, sino que devuelve una copia superficial que contiene copias de los mismos elementos de los arrays originales combinados. Los elementos de los arrays originales son copiados en el nuevo array de la siguiente manera:

* Referencias a Objetos (no el objeto real): **.concat** copia las referencias de objetos en el nuevo array. Ambos, el array original y el nuevo refieren al mismo objeto. Es decir, si un objeto referenciado es modificado, los cambios serán visibles tanto en el array nuevo como en el antiguo.

**.join()**method joins all elements of an array (or an array-like object) into a string and returns this string.

***var* elements = ['Fire', 'Wind', 'Rain'];  
*console.*log(elements.join());** // expected output: Fire,Wind,Rain  
***console.*log(elements.join(''));** // expected output: FireWindRain  
***console.*log(elements.join('-'));** // expected output: Fire-Wind-Rain

***var* a = ['Viento', 'Lluvia', 'Fuego'];**

***var* miVar1 = a.join();** // asigna 'Viento,Lluvia,Fuego' a miVar1

***var* miVar2 = a. join (', ');** // asigna 'Viento, Lluvia, Fuego' a miVar2

***var* miVar3 = a. join (' + ');** // asigna 'Viento + Lluvia + Fuego' a miVar3

***var* miVar4 = a. join ('');** // asigna 'VientoLluviaFuego' a miVar4

**.slice(**beginIndex, endIndex**)** 🡪 This method extracts a section of a string and returns it as a new string.

***beginIndex*** The zero-based index at which to begin extraction. If negative, it is treated as strLength + beginIndex

array**.toString()**; nos saca un string pero con las comas que separan cada elemento

**parseInt(***String, Base***)**

***String****:* Una cadena que representa el valor que se desea convertir.  
***Base****:* Un entero que representa la base de la mencionada cadena.

**parseInt()** es una función de alto nivel y no está asociada a ningún objeto. Tiene la misma funcionalidad que el método ***Number*.parseInt()**

La función **parseInt()** comprueba el primer argumento, una cadena, e intenta devolver un entero de la base especificada. Por ejemplo, una base de 10 indica una conversión a número decimal, 8 octal, 16 hexadecimal, y así sucesivamente. Para bases superiores a 10, las letras del alfabeto indican numerales mayores que 9. Por ejemplo, para números hexadecimales (base 16), se utiliza de la A hasta la F.

Si **parseInt()** encuentra un carácter que no es un numeral de la base especificada, lo ignora a él y a todos los caracteres correctos siguientes, devolviendo el valor entero obtenido hasta ese punto. **parseInt()** trunca los números en valores enteros. Se permiten espacios anteriores y posteriores. Si no se especifica la base o se especifica como 0, JavaScript asume lo siguiente:

* Si el parámetro ***string*** comienza por "0x", la base es 16 (hexadecimal).
* Si el parámetro ***string*** comienza por "0", la base es de 8 (octal).
* Si el parámetro cadena comienza por cualquier otro valor, la base es 10 (decimal).

**.forEach()** method will execute the same code on each element of an array

**arrayName.forEach(*function*(arrayItem) {…});**

Because **.forEach()** is an iterator method, every element in the **array** will be passed to this function as an **argument** in place of **arrayItem**. Syntactically, the name of the parameter does not matter. However, it is a best practice to give parameters descriptive names so that other developers who read your code can easily understand what it does.

**arrayName.forEach(arrayItem** **=>** ***console***.**log(‘I want‘** **+ arrayItem));**

### Arrays de dimensiones múltiples (matriz)

Declaración de la matriz: **var matriz = new Array(mFilas);**Declaración de la primera fila: **matriz [0] = new Array(nColumnas);**

Inicialización de las 4 columnas de la primera fila:

**matriz [0] [0] = “1-1”**

**matriz [0] [1] = “1-2”**

**matriz [0] [2] = “1-3”**

**matriz [0] [3] = “1-4”**

Declaración de la primera fila: **matriz [0] = new Array();**

**matriz [1] [0] = “2-1”**

**matriz [1] [1] = “2-2”**

**matriz [1] [2] = “2-3”**

**matriz [1] [3] = “2-4”**

Instance methods, by definition, require that you create an instance before you can use them. Libraries contain methods that you can call without creating an instance.**.random()** method returns a random number between 0 and 1 **.floor()** method rounds the number down to the nearest whole number.

Both methods belong to the **Math** library:

Math.floor(Math.random()\*50);

Continue: (mirar qué es).

**Evaluación perezosa:**

**Apache Cordova:** Aplicación para que las PWA´s (Progresive Web Apps) sean visibles en dispositivos móviles. Adobe tiene una copia de pago que se llama Phonegap.

Android es un sublenguaje de Java y Objective-C (Swift) es para iOS (i-phone). Sin embargo se puede picar el código con las tecnologías Frontend: HTML, CSS y JavaScript y a continuación envolverlo en Apache Cordova para que sea legible por los dispositivos móviles.

Igual que Apache Cordova está orientado a dispositivos móviles debe haber otra aplicación para convertir a Mac, Windows…

Este proceso se denomina creación de aplicaciones híbridas.

Si directamente se pica el código en el lenguaje propio del dispositivo (p. ej Swift) sería crear una aplicación nativa. Hoy en día Js está desarrollando NativeScript. **Tatatachaaan!!**

**Comando alert:** Muestra un mensaje en forma de ventana modal

**alert(“Mensaje de la ventana modal”);**

**<br/><br/><br/>**

**<center>**

**<a href="JavaScript:window.location=' view-source: '** URL**",>Código fuente</a>**

**</center>**

***document*.write();** es una sintaxis orientada a objetos, en la que el método **write** se aplica al objeto ***document***. Lo mismo ocurre con ***console*.log();**

El método **charAt**, con el argumento **0**, extrae el primer carácter de la variable u objeto al que se le aplica (en una cadena de caracteres, la numeración empieza por cero **object.charAt(arg);** El método **substring**, necesita dos argumentos: el primero indica la posición del primer carácter a extraer, y el segundo indica el número de caracteres a extraer: **object.substring(arg1, arg2);**

The split() method is used to split a string into an array of substrings, and returns the new Array.

**Tip:** If an empty string ("") is used as the separator, the string is split between each character. Si se omite el separador, devolverá un sólo elemento con la cadena completa.

**Note:** The split() method does not change the original string.

Split a string into an array of substrings:

**var str = "How are you doing today?";  
var res = str.split(" ");**

***str*.split(“ “).join(); //**Sustituye los espacios por comas

### Methods

**.toUpperCase(); //** is a built-in method, so it can be appended to the string

**.StartsWith(); .trim();**

In this case: ***Math*.floor(*Math*.random() \* 50);**

1. ***Math*.random** generates a random number between 0 and 1.
2. We then multiply that number **\* 50**, so now we have a number between 0 and 50.
3. Then ***Math*.floor**, rounds the number down to the nearest whole number.

***Math*.floor();**

***Math*.ceil();**

***Math*.round();**

***Number*.isInteger();**

<https://www.aprenderaprogramar.es/index.php?option=com_content&view=category&id=78&Itemid=206>

<https://www.aprenderaprogramar.es/index.php?option=com_content&view=article&id=57&Itemid=86>

<https://www.arkaitzgarro.com/javascript/capitulo-18.html>