**Resumen de Java Script**

**------------- o -------------**

**Cyntia Fernández Marenco**

Tabla de contenido

[Windows Open() 5](#_Toc410983805)

[String toLowerCase() 5](#_Toc410983806)

[String toUpperCase() 5](#_Toc410983807)

[Prompt() 5](#_Toc410983808)

[Reserved Keywords (Palabras Reservadas) 6](#_Toc410983809)

[Confirm() 6](#_Toc410983810)

[Undefined 6](#_Toc410983811)

[Null 6](#_Toc410983812)

[TypeOf 6](#_Toc410983813)

[Web Browse Engine / Motor de renderizado 6](#_Toc410983814)

[Lista de Motores EcmaScript 7](#_Toc410983815)

[Motor EcmaScript 7](#_Toc410983816)

[V8 7](#_Toc410983817)

[Scripting Lenguaje 7](#_Toc410983818)

[Dynamic Typing 7](#_Toc410983819)

[Onunload 7](#_Toc410983820)

[Onload 7](#_Toc410983821)

[Delete 8](#_Toc410983822)

[OOP – Programación Orientada a Objetos 8](#_Toc410983823)

[Constructor de Objetos 8](#_Toc410983824)

[Diferencia entre Document.ready y Onunload 8](#_Toc410983825)

[== Igual Igual 8](#_Toc410983826)

[=== Igual Igual Igual 8](#_Toc410983827)

[Diferencia entre == o === 8](#_Toc410983828)

[Capitulo 2-Gramática 8](#_Toc410983829)

[El espacio en blanco 8](#_Toc410983830)

[Nombres 9](#_Toc410983831)

[Números 9](#_Toc410983832)

[String 9](#_Toc410983833)

[Declaraciones 9](#_Toc410983834)

[Do 9](#_Toc410983835)

[Try 9](#_Toc410983836)

[Throw 10](#_Toc410983837)

[Return 10](#_Toc410983838)

[Expresiones 10](#_Toc410983839)

[Los literales 10](#_Toc410983840)

[Funciones 10](#_Toc410983841)

[Capitulo 3-Objetos 10](#_Toc410983842)

[Los literales de objetos 10](#_Toc410983843)

[Recuperación 10](#_Toc410983844)

[Actualización 11](#_Toc410983845)

[Referencia 11](#_Toc410983846)

[Prototype / Prototipo 11](#_Toc410983847)

[Reflexión 11](#_Toc410983848)

[Enumeración 11](#_Toc410983849)

[Delete / Borrar 11](#_Toc410983850)

[Apendice A 11](#_Toc410983851)

[Variables globales 11](#_Toc410983852)

[Scope 12](#_Toc410983853)

[Semicolon Insertion 12](#_Toc410983854)

[Unicode 12](#_Toc410983855)

[ParseInt 12](#_Toc410983856)

[Operador + 12](#_Toc410983857)

[Floating Point 13](#_Toc410983858)

[NaN 13](#_Toc410983859)

[Phony Arrays 13](#_Toc410983860)

[Falsy Values 13](#_Toc410983861)

[hasOwnProperty 13](#_Toc410983862)

[Objects 13](#_Toc410983863)

[JavaScript Tipo de conversión 14](#_Toc410983864)

[Tipos de datos JavaScript 14](#_Toc410983865)

[El operador typeof 14](#_Toc410983866)

[La propiedad constructor 15](#_Toc410983867)

[La conversión de números a cadenas 15](#_Toc410983868)

[La conversión de los Booleanos para Cuerdas 16](#_Toc410983869)

[Conversión de Fechas para Cuerdas 16](#_Toc410983870)

[La conversión de cadenas en números 16](#_Toc410983871)

[Conversión Booleans a Números 16](#_Toc410983872)

[La conversión de fechas a Números 17](#_Toc410983873)

[Tipo de conversión automática 17](#_Toc410983874)

[Conversión automática de Cuerdas 17](#_Toc410983875)

[JavaScript Izar 17](#_Toc410983876)

[IndexOf() 17](#_Toc410983877)

# Windows Open()

Abre una nueva ventana.

Close() Cierra una ventana.

Ejemplo:

SYNTAXIS:

window.open(*URL,name,specs,replace*)

Eval()

Evalúa o realiza un argumento que se le pase

Ejemplo:

SYNTAXIS:

eval(*string*)

# String toLowerCase()

Convierte un String a letras minúsculas

Ejemplo:

SYNTAXIS:

*string*.toLowerCase()

# String toUpperCase()

Convierte un String a letras mayúsculas.

Ejemplo:

SYNTAXIS:

*string*.toUpperCase()

# Prompt()

Cuadros de diálogos que se le muestran al visitante a la hora de entrada

Ejemplo:

SYNTAXIS:

prompt(*text,defaultText*)

# Reserved Keywords (Palabras Reservadas)

Las siguientes palabras están reservadas en JavaScript:

abstract boolean break byte case catch char class const continue debugger default delete do double else enum export extends false final finally float for function goto if implements import in instanceof int interface long native new null package private protected public return short static super switch synchronized this throw throws transient true try typeof var volatile void while with

La mayoría de estas palabras no se usan en el lenguaje. No pueden ser utilizados para las variables o parámetros de nombres. Cuando las palabras reservadas se utilizan como claves en objetos literales, deben ser citados. Ellos no se pueden utilizar con la notación de puntos, por lo que es necesario a veces utilizar la notación de soporte en su lugar.

# Confirm()

Es una caja de dialogo con la que se especifica un mensaje a lo largo con un Ok o un Cancelar.

Una caja de Confirm es a menudo usada para darle al usuario una aceptación o verificación algunas veces.

Ejemplo:

SYNTAXIS:

confirm(*if, alert, else*)

# Undefined

Son las variables que no han sido definidas por lo tanto no existen o que han sido definidas sin asignarles un valor.

# Null

Este en Javascript es un valor, especial. El valor null es en si mismo un calor, pero un valor que indica la ausencia de contenido, el valor vacio.

# TypeOf

Devuelve una cadena que indica el tipo del operando a evaluar.

# Web Browse Engine / Motor de renderizado

Conocido también como el motor del navegador web, es un software de componente que tiene marcado contenido HtML, XtMl y otros.

Este permite dibujar el contenido en una ventana, para así mostrar el código ya formateado.

# Lista de Motores EcmaScript

# Motor EcmaScript

Es un programa que ejecuta código fuente escrito en una versión del EcmaScript.

Lenguaje estándar:

V8(Utilizado por google Chrome)

Carakan

Chakra

Javacript core, entre otros.

# V8

Es un motor de código abierto para javascript creado por google.

Esta escrito en C++(Lenguaje de programación que permite la manipulación de objetos) y es usado por Google Chrome. Además de que tienen integrado el Sistema Operativo Android 2.2. Puede funcionar solo o incorporado a cualquier aplicación C++.

# Scripting Lenguaje

Es un lenguaje de programación que admite secuencias de comandos, es un lenguaje de alto nivel que esta interpretado por otro programa en tiempo de ejecución que en lugar de compilarlo lo interpreta y automatiza.

# Dynamic Typing

Lenguaje de programación mecanografiado dinámico. En el cual necesariamente deben definirse las variables antes de ser utilizadas.

PHP y PHYTON son lenguajes de programación con lenguaje dinámico.

# Onunload

Este evento se produce cuando una página se ha descargado o la ventana del navegador se ha cerrado, o también cuando el usuario se desplaza fuera de la página o carga la página.

# Onload

Carga el DOM y todas las imágenes y hasta que esté completamente cargado se ejecuta.

# Delete

Borra un objeto o elemento de una variable.

# OOP – Programación Orientada a Objetos

Programa y refleja lo que queremos resolver en la vida cotidiana.

# Constructor de Objetos

Nos ayuda a devolver una instancia. Es una función que esta dentro de una clase.

# Diferencia entre Document.ready y Onunload

Uno carga únicamente el DOM.

# == Igual Igual

Compara solo el valor

# === Igual Igual Igual

Compara valor y tipo.

# Diferencia entre == o ===

Que uno compara si el valor y el tipo son iguales.

# Capitulo 2-Gramática

# El espacio en blanco

El espacio en blanco puede tomar la forma de caracteres de formato o comentarios. Es necesario el uso de los espacios en blanco para separar las secuencias de caracteres que de otro modo se combinan en un único token.

JavaScript ofrece dos formas de comentarios, comentarios en bloque formados con / \* \* / y los comentarios de fin de línea comenzando con //. Los comentarios deben ser usados libremente para mejorar la legibilidad de los programas. Tenga cuidado de que los comentarios siempre describen con precisión el código. Comentarios obsoletos son peores que no tiene comentarios.

# Nombres

Es una carta seguido opcionalmente por una o más letras, dígitos o guiones bajos. Un nombre no puede ser una de estas palabras reservadas:

abstract boolean byte descanso clase Char capturas caso const continuar depurador predeterminado eliminar hacer doble

La mayoría de las palabras reservadas en esta lista no se utilizan en el lenguaje. La lista no incluye algunas palabras que deberían haberse reservado pero no lo fueron, como no definido, NaN, y el Infinito. No está permitido nombrar una variable o parámetro con una palabra reservada.

# Números

JavaScript tiene un solo tipo de número. Internamente, se representa como punto flotante de 64 bits, la misma que la de doble Java. A diferencia de la mayoría de otros lenguajes de programación, no hay ningún tipo de entero por separado, por lo que 1 y 1.0 son el mismo valor.

# String

Cuerdas una cadena literal se pueden envolver en comillas simples o dobles. Puede contener cero o más caracteres. El \ (barra invertida) es el carácter de escape.

# Declaraciones

Es una unidad de compilación que contiene un conjunto de instrucciones ejecutables. En los navegadores web, cada etiqueta <script> entrega una unidad de compilación que se compila y ejecuta inmediatamente. Al carecer de un enlazador, JavaScript todas lanza juntos en un espacio de nombres global común.

Las declaraciones tienden a ser ejecutados en orden de arriba a abajo.

# Do

La instrucción do es como la instrucción while excepto que la expresión se ensaya después el bloque se ejecuta en lugar de antes. Eso significa que el bloque siempre se ejecutará al menos una vez.

# Try

La sentencia try ejecuta un bloque y las capturas de las excepciones que fueron arrojados por el bloque. La cláusula catch define una nueva variable que recibirá el objeto de excepción.

# Throw

La sentencia throw lanza una excepción. Si la sentencia throw está en un bloque try, el control continúa en la cláusula catch. De lo contrario, la invocación de la función es abandonado, y el control pasa a la cláusula catch de la prueba en la función que llama.

# Return

La sentencia return hace que el pronto retorno de una función. También puede especificar el valor a devolver. Si no se especifica una expresión de retorno, entonces el valor de retorno será indefinido.

# Expresiones

Las expresiones más simples son un valor literal (tal como una cadena o un número), una variable, un valor integrado entre otros.

# Los literales

Los literales de objetos son una notación conveniente para especificar los objetos nuevos. Los nombres de las propiedades se pueden especificar como nombres o como cadenas.

# Funciones

Una literal función define un valor de función. Puede tener un nombre opcional que puede usar para llamar a sí mismo de forma recursiva. Puede especificar una lista de parámetros que actuarán como variables incluye definiciones y declaraciones.

# Capitulo 3-Objetos

# Los literales de objetos

Los literales de objetos proporcionan una notación muy conveniente para la creación de nuevos valores de objeto.

Un objeto literal es un par de llaves que rodean cero o más pares nombre / valor.

# Recuperación

Los valores se pueden recuperar de un objeto envolviendo una expresión de cadena en un [ ] sufijo. Si la expresión de cadena es una constante, y si es un nombre legal JavaScript y no una palabra reservada, entonces la notación se puede utilizar en su lugar. Se prefiere la notación porque es más compacto y se lee mejor.

# Actualización

Un valor de un objeto puede ser actualizado por asignación. Si el nombre de la propiedad ya existe en el objeto, el valor de la propiedad se sustituye.

# Referencia

Objetos se pasan alrededor por referencia. Nunca se copian.

# Prototype / Prototipo

Es cada objeto que está vinculado a un objeto prototipo de la que puede heredar propiedades. Todos los objetos creados a partir de objetos literales están vinculados a **Object.prototype,** un objeto que viene de serie con JavaScript. Cuando usted hace un nuevo objeto, puede seleccionar el objeto que debe ser su prototipo.

# Reflexión

Es para inspeccionar de manera fácil un objeto para determinar qué propiedades tiene al tratar de recuperar las propiedades y el examen de los valores obtenidos. El operador typeof puede ser muy útil para determinar el tipo de una propiedad.

# Enumeración

Es donde se incluirán todas las propiedades, incluyendo funciones y propiedades de prototipo que podría no estar interesados en lo que es necesario para filtrar los valores que no desea. Los filtros más comunes son el método hasOwnProperty y el uso de typeof para excluir funciones.

# Delete / Borrar

Se puede utilizar para eliminar una propiedad de un objeto. Se eliminará una propiedad del objeto, si lo tiene. No va a tocar cualquiera de los objetos en el vínculo prototipo.

# Apendice A

# Variables globales

La peor de todas las malas características de JavaScript es su dependencia de las variables globales.

Una variable global es una variable que es visible en todos los ámbitos. Las variables globales pueden ser una conveniencia en programas muy pequeños, pero convertido rápidamente en difícil de manejar como los programas se hacen más grandes. Debido a que una variable global se puede cambiar por cualquier parte del programa en cualquier momento, pueden complicar significativamente el comportamiento del programa.

# Scope

La sintaxis de JavaScript viene de C. En todos los demás lenguajes de programación como C, un bloque (un conjunto de estados envueltos entre llaves) crea un ámbito.

# Semicolon Insertion

JavaScript tiene un mecanismo que intenta corregir los programas defectuosos por punto y coma insertar automáticamente. No dependen de esto. Esto puede enmascarar los errores más graves. A veces inserta un punto y coma en lugares donde no son bienvenidos.

# Unicode

JavaScript fue diseñado en un momento en que se espera Unicode tener como máximo 65.536 caracteres. Desde entonces ha crecido para tener una capacidad de más de 1 millón de caracteres.

# ParseInt

ParseInt es una función que convierte una cadena en un número entero. Se detiene cuando ve un nondigit, por lo parseInt ("16") y parseInt ("16 toneladas") producen el mismo resultado.

# Operador +

El operador + puede agregar o concatenar. Cuál lo hace depende de los tipos de los parámetros. Si alguno de los operandos es una cadena vacía, produce el otro operando convierte en una cadena. Si ambos operandos son números, que produce la suma. De lo contrario, convierte ambos operandos de cadenas y las concatena. Este comportamiento complicado es una fuente común de errores. Si tiene la intención + añadir, asegúrese de que ambos operandos son números.

# 

# Floating Point

Números de punto flotante binarios son ineptos en el manejo de las fracciones decimales, por lo que 0.1 + 0.2 no es igual a 0,3. Este es el error más frecuentemente reportado en JavaScript, y es una consecuencia intencional de haber adoptado el estándar IEEE para aritmética binaria FloatingPoint (IEEE 754). Esta norma es muy adecuado para muchas aplicaciones, pero viola la mayoría de las cosas que ha aprendido acerca de los números en la escuela media.

# NaN

El valor NaN es una cantidad especial definido por IEEE 754. Representa no es un número, a pesar de que:

typeof "número" NaN === // true

El valor puede ser producido por el intento de convertir una cadena en un número cuando la cadena no está en la forma de un número.

# Phony Arrays

JavaScript no tiene matrices reales. Eso no es del todo malo. Matrices de JavaScript son muy fácil de usar. No hay necesidad de darles una dimensión, y nunca generar errores outof-grada. Sin embargo, su rendimiento puede ser considerablemente peor que las matrices reales.

# Falsy Values

JavaScript tiene un sorprendente conjunto de valores falsy. Estos valores son todos falsy, pero no son intercambiables.

# hasOwnProperty

El método hasOwnProperty se ofrece como un filtro para evitar un problema con el en el comunicado. Desafortunadamente, hasOwnProperty es un método, no un operador, por lo que en cualquier objeto que pueda ser reemplazado con una función diferente o incluso un valor que no es una función.

# Objects

Objetos de JavaScript nunca son verdaderamente vacío, ya que pueden recoger a los miembros de la cadena de prototipo.

# JavaScript Tipo de conversión

Número () convierte a un Number, String () convierte a String, Boolean ()

convierte a un booleano.

# Tipos de datos JavaScript

En JavaScript hay 5 tipos de datos diferentes que pueden contener valores:

● cadena

● número

● boolean

● objeto

● función

Hay 3 tipos de objetos:

● Objeto

● Fecha

● Colección

Y 2 tipos de datos que no pueden contener valores:

● nulo

● indefinido

# El operador typeof

Puede utilizar el typeof operador para encontrar el tipo de datos de una variable

de JavaScript.

**Ejemplo**

typeof "John" // Returns string

typeof 3.14 // Returns number

typeof NaN // Returns number

typeof false // Returns boolean

typeof [ 1 , 2 , 3 , 4 ] // Returns object

typeof {name: 'John' , age:34} // Returns object

typeof new Date() // Returns object

typeof function () {} // Returns function

typeof myCar // Returns undefined (if myCar is not

declared)

typeof null // Returns object

**I**

**nténtelo usted mismo »**

Tenga en cuenta:

● El tipo de datos de NaN es el número

● El tipo de datos de una matriz es objeto

● El tipo de datos de una fecha es objeto

● El tipo de datos de nula es objeto

● El tipo de datos de una variable no definida es indefinido

No se puede usar typeof para definir si un objeto es un array JavaScript o una

Fecha de JavaScript.

# La propiedad constructor

El constructor propiedad devuelve la función constructora para todas las variables

de JavaScript.

**Ejemplo**

"John" .constructor // Returns function String() { [native

code] }

( 3.14 ).constructor // Returns function Number() { [native

code] }

false.constructor // Returns function Boolean() { [native

code] }

[ 1 , 2 , 3 , 4 ].constructor // Returns function Array() { [native

code] }

{name: 'John' , age:34}.constructor // Returns function Object() { [native

code] }

new Date().constructor // Returns function Date() { [native

code] }

function () {}.constructor // Returns function Function(){ [native

code] }

**Inténtelo usted mismo »**

Usted puede comprobar la propiedad constructor para averiguar si un objeto es

un array (contiene la palabra "arrays"):

# La conversión de números a cadenas

El método global String () puede convertir números a cadenas.

Se puede utilizar en cualquier tipo de números, literales, variables o expresiones:

**Ejemplo**

String(x) // returns a string from a number variable x

String( 123 ) // returns a string from a number literal 123

String( 100 + 23 ) // returns a string from a number from an expression

# La conversión de los Booleanos para Cuerdas

El método global String () puede convertir booleanos para cuerdas.

String( false ) // returns "false"

String( true ) // returns "true"

El método booleano toString () hace lo mismo.

false.toString() // returns "false"

true.toString() // returns "true"

# Conversión de Fechas para Cuerdas

El método global String () puede convertir fechas para cuerdas.

String(Date()) // returns Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe

Daylight Time)

El método Fecha toString () hace lo mismo.

Ejemplo

Date().toString() // returns Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe

Daylight Time

# La conversión de cadenas en números

El método global Number () puede convertir cadenas a números.

Las cadenas que contengan números (como "3.14") se convierten en números

(como 3.14).

Las cadenas vacías se convierten en 0.

Todo lo demás se convierte en NaN (no es un número).

Number( "3.14" ) // returns 3.14

Number( " " ) // returns 0

Number( "" ) // returns 0

Number( "99 88" ) // returns NaN

# Conversión Booleans a Números

El método global Number () también puede convertir booleans a números.

Number( false ) // returns 0

Number( true ) // returns 1

# La conversión de fechas a Números

El método global Number () puede ser usado para convertir las fechas para

números.

d = new Date();

Number(d) // returns 1404568027739

El método de la fecha getTime () hace lo mismo.

d = new Date();

d.getTime() // returns 1404568027739

# Tipo de conversión automática

Cuando JavaScript intenta operar en un tipo de datos "mal", que tratará de

convertir el valor a un tipo de "derecho".

El resultado no siempre es lo que esperas:

5 + null // returns 5 because null is converted to 0

"5" + null // returns "5null" because null is converted to "null"

"5" + 1 // returns "51" because 1 is converted to "1"

"5" - 1 // returns 4 because "5" is converted to 5

# Conversión automática de Cuerdas

JavaScript llama automáticamente a la función toString de la variable () cuando

intenta "salida" de un objeto o una variable:

document.getElementById( "demo" ).innerHTML = myVar

# JavaScript Izar

Izar es JavaScript comportamiento predeterminado 's de las declaraciones de pasar a la parte superior.

# IndexOf()

El **indexOf()** método devuelve el índice, dentro del objeto String que realiza la llamada, de

la primera ocurrencia del valor especificado, comenzando la búsqueda desde

indiceBusqueda ; o 1

si no se encuentra dicho valor.