**JavaScript Best Practices.**

Evitar y minimizar las variables globales , todo data type, objetos y funciones.

Siempre declarar variables locales.

Todas las variables deben ser declaradas como Locales.

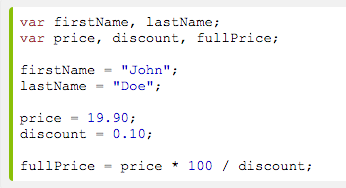
Las variables locales se declaran con var.

**Declaraciones sobre el tope.**

Poner todas las declaraciones en la parte superior de ada SCRIPT o FUN.

Mas fácil evitar variables globales no deseadas y se reduce la posibilidad de re-declaraciones no deseadas.

Las variables deben ser los primeros scripts y funciiones , tambien en loop.



**Nunca declarar numbers , strigns o booleans como objetos.**

Usar num , string o booleans como valores primitivos.

Cmo objetos se relentiza la velocidad de ejecucion y produce efectos secundarios desagradables.

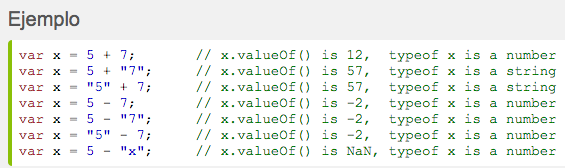
* Use {} en lugar de new Object ()
* Utilice "" en lugar de new String ()
* Utilice 0 en lugar de nueva Number ()
* Utilice falsa en lugar de new Boolean ()
* Utilice [] en lugar de new Array ()
* Utilice / () / en lugar de nuevo RegExp ()
* Utilice function () {} en lugar de la nueva función ()

## Tener cuidado con Automatic Type Conversions.

Ten en cuenta que los números accidentalmente pueden ser convertidos a cadenas o NaN (Not a Number).

Una variable puede contener diferentes tipos de datos, y una variable puede cambiar su tipo de datos:

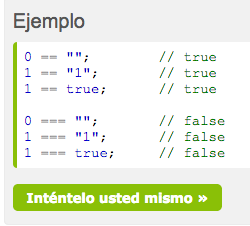
Al hacer operaciones matemáticas, JavaScript puede convertir números a cadenas.



Restando una cadena de una cadena, no genera un error pero devuelve NaN (Not a Number)

## Utilice === Comparación.

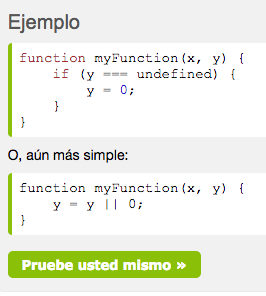
Siempre convierte (to matching types) antes de la comparación.



## Utilice Parameter Defaults.

Si llama una función con un argumento que falta, el argumento que falta se establece en indefinido.

Los valores no definidos pueden romper el código. Es un buen hábito para asignar valores predeterminados a los argumentos.



## Evitar el uso de eval ()

La función eval () se utiliza para ejecutar el texto como código. En casi todos los casos, no debería ser necesario usarlo.

Porque permite código arbitrario ejecutado, también representa un problema de seguridad.

## Introducción

Los objetos son entidades que tienen un determinado *estado*, *comportamiento (método)* e *identidad*:

* El *estado* está compuesto de datos o informaciones; serán uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
* El comportamiento está definido por los [métodos](http://es.wikipedia.org/wiki/M%C3%A9todo_%28inform%C3%A1tica%29) o mensajes a los que sabe responder dicho objeto, es decir, qué operaciones se pueden realizar con él.
* La identidad es una propiedad de un objeto que lo diferencia del resto; dicho con otras palabras, es su identificador (concepto análogo al de identificador de una [variable](http://es.wikipedia.org/wiki/Variable_%28programaci%C3%B3n%29) o una [constante](http://es.wikipedia.org/wiki/Constante_%28programaci%C3%B3n%29)).

**Objeto**

Contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. Disponen de mecanismos de interacción llamados [métodos](http://es.wikipedia.org/wiki/M%C3%A9todo_%28programaci%C3%B3n_orientada_a_objetos%29) , favorece a su vez el cambio de estado en los propios objetos como unidades indivisibles , no se separa el estado y el comportamiento.

**Los métodos (comportamiento) y atributos (estado)**

Están relacionados por la propiedad de conjunto. Destaca que una clase requiere de métodos para tratar los atributos con los que cuenta. El [programador](http://es.wikipedia.org/wiki/Programador) debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a alguno de ellos.

**La POO**

Difiere de la [programación estructurada](http://es.wikipedia.org/wiki/Programaci%C3%B3n_estructurada) tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan.

En la programación estructurada solo se escriben funciones que procesan datos. Los programadores que emplean Programación Orientada a Objetos, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

## 

[Origen y Conceptos Fundamentales](https://docs.google.com/document/d/11-dF74OCo-JjwiRzwAePvfbPDYuPuRvkYoO80mqJ2dY/edit#heading=h.gl5c9zqw9o86).

Un **motor del navegador web** ( **motor de diseño** o **motor de renderizado)** es un software de [componente](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Component-based_software_engineering&usg=ALkJrhgdA58hazAQHDPGBtAnAdhkaeyi2A) que tiene [marcado](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Markup_language&usg=ALkJrhg2SUl3jKYNVzHzVdXQgA0rR9ufRA) contenido (como [HTML](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/HTML&usg=ALkJrhizPrvfOLclTppzVcKR7mC59txWPg) , [XML](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/XML&usg=ALkJrhg92oas4mAarYFnkjqACY2J-GScTw) , [img](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Digital_image&usg=ALkJrhgw5GxXu9bpfLmpGDOD_h7kCc7cjw) de archivos, etc.) y la información de formato (como [CSS](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Cascading_Style_Sheets&usg=ALkJrhh4oIdyVpX6DGBo8ybQwdn6NDMxnQ) , [XSL](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Extensible_Stylesheet_Language&usg=ALkJrhhOYsGqZcp0KsiYkgwDamJDRMTDNw) ) y muestra el contenido formateado en la pantalla. Se basa en el área de contenido de una ventana, que se muestra en un [monitor de](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Computer_monitor&usg=ALkJrhg_GG8R2z2sYMLOEJVEYohSqCaYmg) o una [impresora](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Computer_printer&usg=ALkJrhghW69amYpfXivqlocrMIwABhlSWQ) . Un motor de diseño es típicamente incrustado en [los navegadores web](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Web_browser&usg=ALkJrhjhVCqIiV-WmX2xxkyZs2pZVlePWQ) , [clientes de correo electrónico](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/E-mail_client&usg=ALkJrhjnPcC1oLHhpv1A5ISSUQA5-_4cvg) , [lectores de libros electrónicos](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/E-book_readers&usg=ALkJrhgL0X61xkxazuThzVpkmybsbNy3Kw) , en línea los sistemas de ayuda u otras aplicaciones que requieren la visualización (y edición) de contenido web. Los motores pueden esperar a que todos los datos que se reciban antes de renderizar una página, o pueden comenzar prestación antes de recibir todos los datos. Esto puede resultar en páginas cambiando a medida que se recibe más datos, como imágenes que se están rellenados o un [destello de contenido sin estilo](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Flash_of_unstyled_content&usg=ALkJrhjv1IKZNvVIJZwwNBz2o6RaXMrbEQ) si la representación comienza antes de formatear la información que se reciba.

## Server-side and client-side programming

El desarrollo web es todo acerca de la comunicación. En este caso, la comunicación entre dos (2) partes, a través del protocolo HTTP:

El servidor - Esta parte es responsable de servir páginas.

El cliente - Esta parte solicita páginas desde el servidor, y los muestra al usuario. En la mayoría de los casos, el cliente es un navegador web.

El Usuario - El usuario utiliza el cliente con el fin de navegar por la web, rellenar formularios, ver videos en línea, etc.

La programación de cada lado, se refiere al código que se ejecuta en la máquina específica, del servidor o el cliente de.

**Ejemplo**

* El usuario abre su navegador web (el cliente).
* El Usuario navega a http://google.com.
* El cliente (en el nombre del usuario), envía una petición a http://google.com (el servidor), por su página de inicio.
* El servidor reconoce la solicitud y responde al cliente con algunos meta-datos (calledheaders), seguido de la fuente de la página.
* El cliente recibe entonces fuente de la página, y lo hace en un sitio web visible humano.
* Los tipos de usuario desbordamiento de pila en la barra de búsqueda, y presione la tecla Intro
* El cliente envía los datos al servidor.
* El servidor procesa esos datos, y responde con una página de búsqueda de los resultados de búsqueda.
* El Cliente, una vez más, hace que esa página para el usuario que desea ver.

# **Programación**

## Programación del lado del servidor

Programación del lado del servidor, es el nombre general para los tipos de programas que se ejecutan en el servidor.

Usos

1. Proceso de entrada del usuario.
2. Páginas de pantalla.
3. Aplicaciones web Estructura.
4. Interactuar con el almacenamiento permanente (SQL, archivos).

Ejemplo Idiomas

1. PHP
2. ASP.Net en C #, C ++ o Visual Basic.
3. Casi cualquier idioma (C ++, C #, Java). Estos no fueron diseñados específicamente para la tarea, pero ahora se utilizan a menudo para los servicios web a nivel de aplicación.

## La programación del lado del cliente

Al igual que el del lado del servidor, la programación del lado del cliente es el nombre de todos los programas que se ejecutan en el cliente.

Usos

1. Hacer páginas web interactivas.
2. Haga cosas suceda de forma dinámica en la página web.
3. Interactuar con el almacenamiento temporal y el almacenamiento local (cookies, localStorage).
4. Enviar solicitudes al servidor, y recuperar datos de él.
5. Proporcionar un servicio remoto para aplicaciones del lado del cliente, como el registro de software, la entrega de contenido, o juegos con varios jugadores a distancia.

Ejemplo idiomas

1. JavaScript (principalmente)
2. HTML \*
3. CSS \*
4. Cualquier lenguaje que se ejecuta en un dispositivo cliente que interactúa con un servicio remoto es un lenguaje del lado del cliente.

\* HTML y CSS no están realmente "lenguajes de programación" per-se. Están Sintaxis del marcado por el que el cliente hace que la página para el usuario.

## Lista de los motores de ECMAScript.

Un **motor de ECMAScript** es un programa que ejecuta el código fuente escrito en una versión de la [ECMAScript](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/ECMAScript&usg=ALkJrhhE653Dt2PMZBu3vJdbkRKN--97cA) lenguaje estándar, por ejemplo, [JavaScript](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/JavaScript&usg=ALkJrhgniKJ8GEWQWHEUACnZSzu8NQAmHQ) .

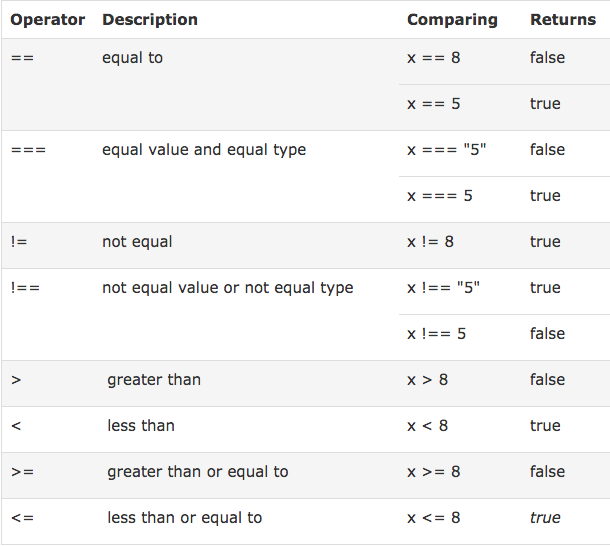
Estos son los motores nuevos de ECMAScript generación de navegadores web, todo implementación [justo a tiempo de compilación](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Just-in-time_compilation&usg=ALkJrhikCGfQ7oVF8sQrLK7xqNOyOIhFHw)(JIT) o variaciones de esa idea. Los beneficios de rendimiento para la compilación justo a tiempo hacen que sea mucho más adecuado para aplicaciones web escritas en [JavaScript](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/JavaScript&usg=ALkJrhgniKJ8GEWQWHEUACnZSzu8NQAmHQ) .

[V8](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/V8_(JavaScript_engine)&usg=ALkJrhifymE-XTlQHbd3fOxNcxePvObmIQ) : Un motor de JavaScript se utiliza en [Google Chrome](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Google_Chrome&usg=ALkJrhjN2zBt8Gpr6rVgvwlU4hvy__AcBQ) , [Node.js](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://en.wikipedia.org/wiki/Node.js&usg=ALkJrhiWu9z4IEuPgThxta1d--DawQvM-g) y [V8 .NET](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://v8dotnet.codeplex.com/&usg=ALkJrhjKqLbeaghKGmHX-OiofNA2F9zrHQ) .

## Operadores de comparación.

Los operadores de comparación se utilizan en los estados lógicos para determinar la igualdad o la diferencia entre las variables o valores.

Dado que x = 5, la siguiente tabla se explican los operadores de comparación:



## ¿Cómo puede ser usado.

Los operadores de comparación se pueden utilizar en las sentencias condicionales para comparar valores y actuar en función del resultado:

if (age < 18) text = "Too young";

## Operadores lógicos.

Los operadores lógicos se utilizan para determinar la lógica entre las variables o valores.

Dado que x = 6 ey = 3, la siguiente tabla se explican los operadores lógicos:



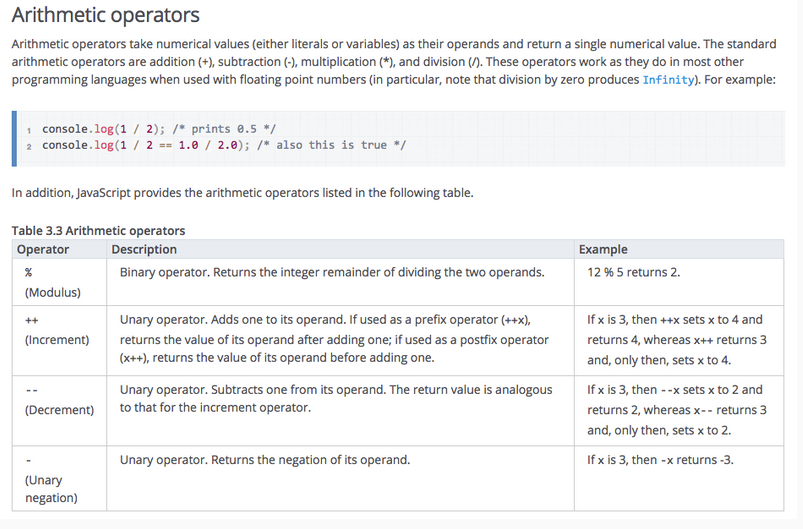
## Operador condicional.

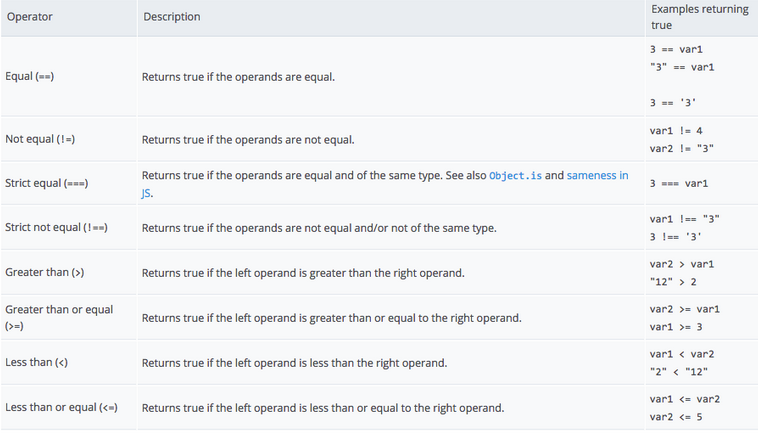
JavaScript también contiene un operador condicional que asigna un valor a una variable basado en alguna condición.

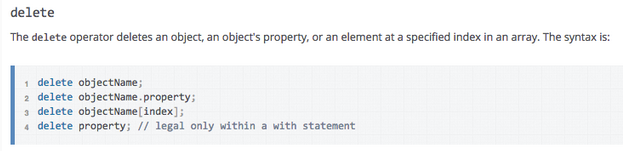
Si la variable edad es un valor por debajo de 18, el valor de la variable de voteable será "demasiado joven", de lo contrario el valor de voteable será "bastante mayor".

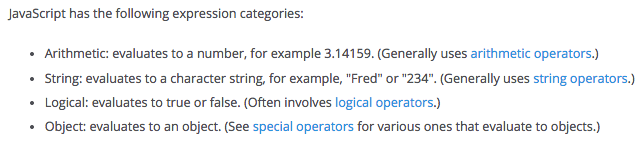
*variablename* = ( *condition* ) ? *value1* : *value2*

*voteable = (age < 18) ? "Too young":"Old enough";*

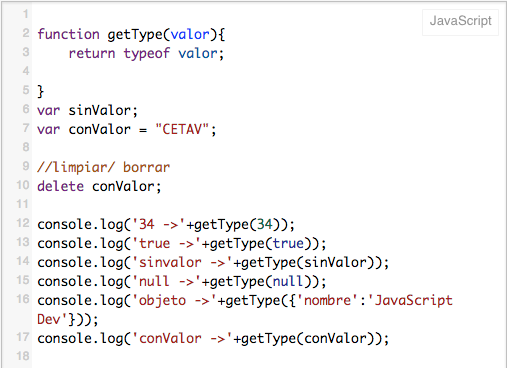




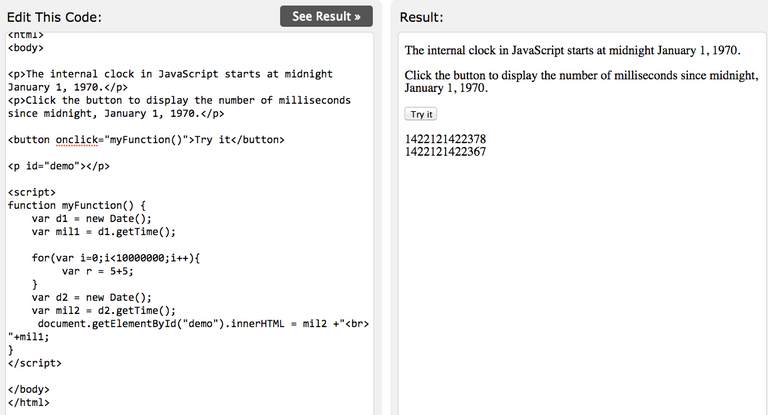












## 

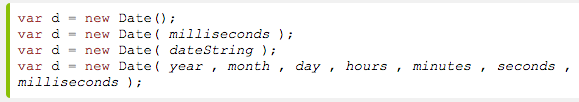
## 

## Date Object

El objeto Date se utiliza para trabajar con fechas y horas.

Fecha objetos son creados con new Date ().

Hay cuatro formas de crear instancias de una fecha:



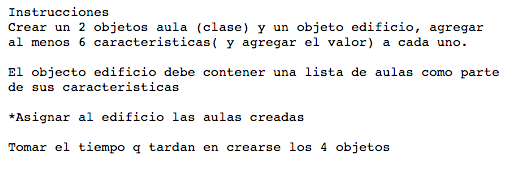
## Date Object Properties

|  |  |
| --- | --- |
| Property | Description |
| [constructor](http://www.w3schools.com/jsref/jsref_constructor_date.asp) | Returns the function that created the Date object's prototype |
| [prototype](http://www.w3schools.com/jsref/jsref_prototype_date.asp) | Allows you to add properties and methods to an object |

[Date Object Methods](http://www.w3schools.com/jsref/jsref_obj_date.asp)

[Operator Precedence](http://matricula.parquelalibertad.org/mod/resource/view.php?id=279)

Instrucciones  
Crear un 2 objetos aula (clase) y un objeto edificio, agregar   
al menos 6 características( y agregar el valor) a cada uno.  
El objeto edificio debe contener una lista de aulas como parte   
de sus características  
\*Asignar al edificio las aulas creadas  
Tomar el tiempo q tardan en crearse los 4 objetos.



[Ejercicio 1 Semana 4](http://jsfiddle.net/4n3qz72p/)



[Ejercicio 2 Semana 4](http://jsfiddle.net/y7k9fxf6/)

## Los valores booleanos

Booleano JavaScript representa uno de dos valores: True o False.

Muy a menudo, en la programación, usted necesitará un tipo de datos que sólo puede tener uno de dos valores, como

* SI / NO
* ON / OFF
* TRUE / FALSE

Para esto, JavaScript tiene un tipo de datos Boolean. Sólo puede tomar los valores verdadero o falso.

## El Boolean () Función

Usted puede utilizar la función Boolean () para averiguar si una expresión (o una variable) es verdadera:

Boolean(10 > 9) // returns true

(10 > 9) // also returns true

10 > 9 // also returns true

## 

## Comparaciones y Condiciones

Comparaciones El capítulo JS da una visión completa de los operadores de comparación.

Las condiciones capítulo JS da una visión completa de las sentencias condicionales.

He aquí algunos ejemplos:

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| == | equal to | if (day == "Monday") |
| > | greater than | if (salary > 9000) |
| < | less than | if (age < 18) |

## Todo con un valor real es True

[True Value](http://www.w3schools.com/js/tryit.asp?filename=tryjs_boolean)

## Todo sin un verdadero valor es False

El valor booleano de 0 (cero) es falsa:

var x = 0;

Boolean(x); // returns false

El valor booleano de -0 (menos cero) es falsa:

var x = -0;

Boolean(x); // returns false

El valor booleano de "" (cadena vacía) es falsa:

var x = "";

Boolean(x); // returns false

El valor booleano de indefinido es falsa:

var x;

Boolean(x); // returns false

El valor booleano de nula es falsa:

var x = null;

Boolean(x); // returns false

El valor booleano de falsa es (adivinaste) falsa:

var x = false;

Boolean(x); // returns false

El valor booleano de NaN es falsa:

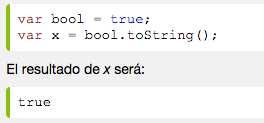
var x = 10 / "H";

Boolean(x); // returns false

## JavaScript Boolean toString() Method

Ejemplo

Convertir un valor booleano en una cadena:



## Definición y uso

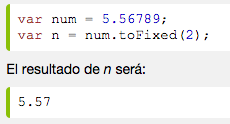
El método toString () devuelve un valor booleano como una cadena.

Nota: Este método es llamado por JavaScript automáticamente cada vez que se utiliza un valor lógico en una operación de cadena.

## JavaScript toFixed () Método

Ejemplo

Convertir un número en una cadena, manteniendo sólo dos decimales:



## Definición y uso

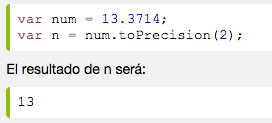
El método toFixed () convierte un número en una cadena, manteniendo un número determinado de decimales.

Nota: si el número deseado de decimales son más altos que el número real, se añaden los nulos para crear la longitud decimal deseada.

## JavaScript toPrecision () Método

Ejemplo

Formatear un número en una longitud especificada:



## Definición y uso

El método toPrecision () formatea un número a una longitud especificada.

Se añaden un punto decimal y los nulos (si es necesario), para crear la longitud especificada.

## String.prototype.indexOf()

Resumen

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.

Sintaxis

*cadena*.indexOf(*valorBusqueda*[, *indiceDesde*])

Parámetros

valorBusqueda

Una cadena que representa el valor de búsqueda.

indiceDesde

La localización dentro de la cadena llamada desde la que empezará la búsqueda. Puede ser un entero entre 0 y la longitud de la cadena. El valor predeterminado es 0.

## Descripción

Los caracteres de una cadena se indexan de izquierda a derecha. El índice del primer carácter es 0, y el índice del último carácter de una cadena llamada nombreCadena es nombreCadena.length - 1.



El método indexOf es sensible a mayúsculas. Por ejemplo, la siguiente expresión devuelve -1:

"Ballena Azul".indexOf("azul")

## Ejemplos

Ejemplo: Usando indexOf y lastIndexOf

El siguiente ejemplo utiliza indexOf y lastIndexOf para localizar valores dentro de la cadena "Brave new world".

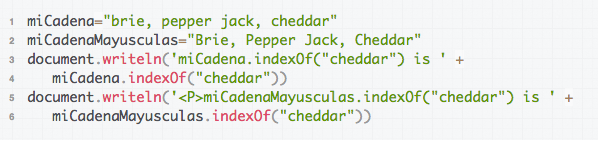


### 

Ejemplo:

### indexOf y sensibilidad a mayúsculas

El siguiente ejemplo define dos variables de tipo cadena. Las variables contienen la misma cadena excepto que la segunda cadena contienen letras en mayúscula. El primer método writeln muestra 19. Pero a que el métodoindexOf es sensible a mayúsculas, no se encuentra la cadena "cheddar" en miCadenaMayusculas, así que el segundo método writeln muestra -1.



Ejemplo:

### Usando indexOf para contar ocurrencias de una letra en una cadena

El siguiente ejemplo establece cuenta como el número de ocurrencias de la letra x dentro de la cadenamiCadena:



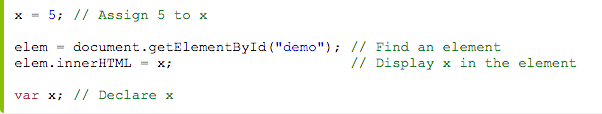
# **JavaScript Hoisting**

## JavaScript Declarations are Hoisted

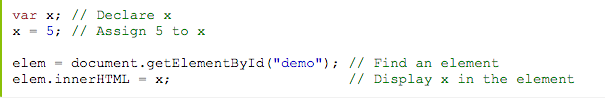
En JavaScript, una variable puede ser declarado después de que se ha utilizado.

En otras palabras; una variable puede ser utilizado antes de que haya sido declarada.

Ejemplo 1



Ejemplo 2



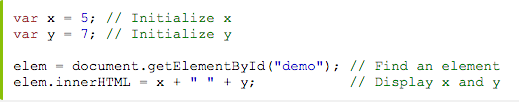
Para entender esto, hay que entender el término "hoisting".

Hoisting es el comportamiento predeterminado de JavaScript de mover todas las declaraciones de la parte superior del ámbito actual (a la parte superior de la secuencia de comandos actual o la función actual).

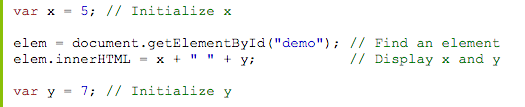
## Inicializaciones JavaScript no son Hoisted

JavaScript sólo iza declaraciones, no inicializaciones.

Ejemplo 1

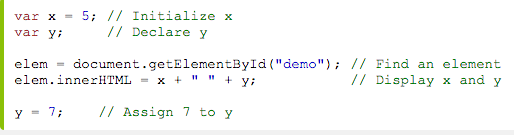


Ejemplo 2



Esto es porque sólo la declaración (var y), no la inicialización (= 7) se iza a la parte superior.

Debido a elevación, y se ha declarado antes de que se utiliza, sino porque inicializaciones no se izan, el valor de y es indefinido.



## Declarar las variables en la cima!

Izar es (para muchos desarrolladores) un comportamiento desconocido o ignorado de JavaScript.

Si un desarrollador no entiende de elevación, los programas pueden contener bugs (errores).

Para evitar errores, siempre declarar todas las variables al comienzo de cada ámbito.

Dado que esta es la forma en JavaScript interpreta el código, siempre es una buena regla.

# **JavaScript uso estricto**

"Use strict"; Define que el código JavaScript debe ser ejecutado en el "modo estricto"

El propósito de "use strict" es para indicar que el código debe ser ejecutado en el "modo estricto".

Con el modo estricto, no se puede, por ejemplo, utilizar variables no declaradas.

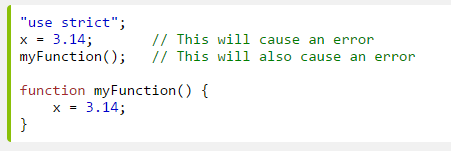
# **Declarando Modo estricto**

El modo estricto se declara mediante la adición de "uso estricto"; al principio de un archivo JavaScript, o una función de JavaScript.

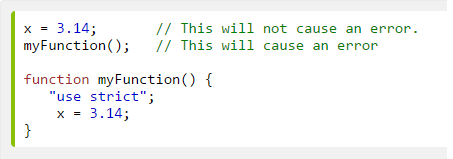
Declarado en el comienzo de un archivo JavaScript, tiene un alcance global (todo el código seejecutará en modo estricto).

Declarado dentro de una función, tiene alcance local (sólo el código dentro de la función es en modo estricto).

***Declaración global:***



***Declaración local:***



# **¿Por qué Modo estricto?**

El modo estricto hace que sea más fácil escribir "seguro" de JavaScript.

Cambios de modo terminante aceptada previamente "mala sintaxis" en errores reales.

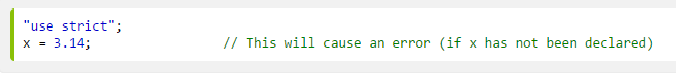
Como ejemplo, en JavaScript normal, mistyping un nombre de variable crea una nueva variable global. En modo estricto, esto generará un error, por lo que es imposible crear accidentalmente una variable global.

En JavaScript normal, un desarrollador no recibirá ninguna notificación de errores asignación de valores a las propiedades no modificables.

En modo estricto, cualquier asignación a una propiedad no se puede escribir, una propiedad-comprador único, un no-existente propiedad, una variable que no existe, o un objeto no existente, se generará un error.

# **No se permite en modo estricto**

El uso de una variable (propiedad u objeto) sin declararlo, no está permitido:



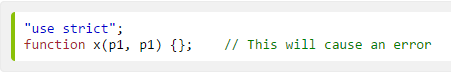
Eliminación de una variable, una función o un argumento, no está permitido.



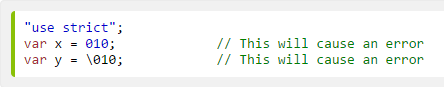
Definición de una propiedad más de una vez, no está permitido:



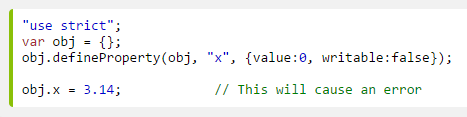
Duplicación de un nombre de parámetro no está permitido:



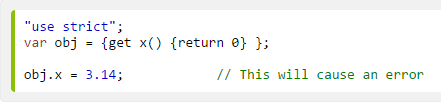
Octal literales numéricos y caracteres de escape no están permitidos:



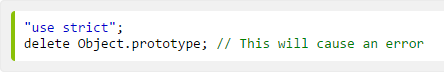
Escribiendo a una propiedad de sólo lectura no está permitido:



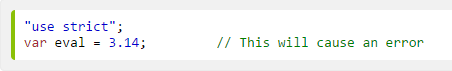
Escribiendo a una propiedad-get sólo no está permitido:



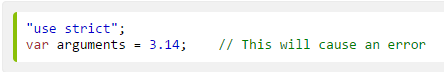
Eliminación de una propiedad imborrable no está permitido:



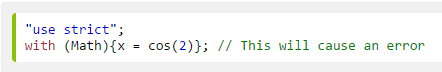
La cadena "eval" no puede ser utilizado como una variable:



Los "argumentos" de cadena no se pueden usar como una variable:



No se permite la sentencia with:



Por razones de seguridad, eval () no pueden crear variables en el ámbito desde el que se llama:



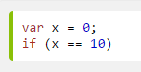
JavaScript Common Mistakes  
<http://www.w3schools.com/js/js_mistakes.asp>

# **Errores comunes de JavaScript**

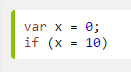
## Accidentalmente Usando el operador de asignación

Programas JavaScript pueden generar resultados inesperados si un programador utiliza accidentalmente un operador de asignación (=), en lugar de un operador de comparación (==) en una sentencia if.

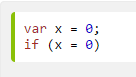
Esta sentencia if devuelve false (como se esperaba), ya que x no es igual a 10:



Esta sentencia if devuelve true (quizás no tan esperado), porque 10 es verdadero:

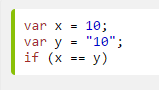


Esta sentencia if devuelve false (quizás no tan esperado), porque 0 es falsa:

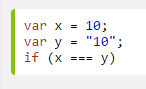


# **Esperando Libremente Comparación**

En comparación regular, tipo de datos no importa. Esta sentencia if devuelve true:

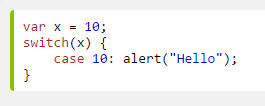


En comparación estricta, tipo de datos es importante. Esta sentencia if devuelve false:

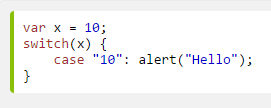


Es un error común olvidar que las sentencias switch utilizan comparación estricta:

Este interruptor caso mostrará una alerta:



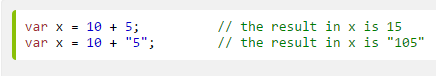
Este interruptor caso no mostrará una alerta:



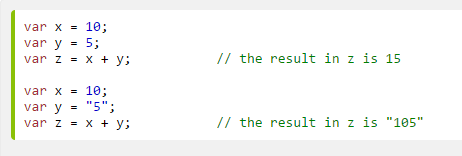
# **Confundir Adición y concatenación**

Además se trata de la adición de números. La concatenación es sobre la adición de cuerdas. En JavaScript ambas operaciones utilizan el mismo operador +.

Debido a esto, cuando se añade un número como un número, producirá un resultado diferente de la adición de un número como una cadena:



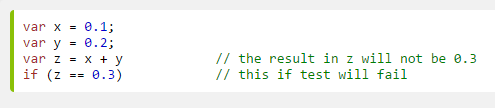
Cuando la adición de dos variables, puede ser difícil anticipar el resultado:



# **Incomprensión Flota**

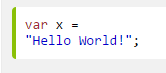
Todos los números en JavaScript se almacenan como 64-bits de números de punto flotante (flotadores).

Todos los lenguajes de programación, incluyendo JavaScript, tienen dificultades con valores de coma flotante precisas:

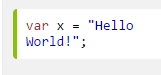


# **Romper una cadena JavaScript**

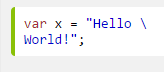
JavaScript se permitirá romper una declaración en dos líneas:



Pero, rompiendo un comunicado en el medio de una cadena no funcionará:

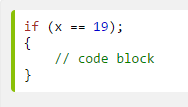


Se debe utilizar una "barra invertida" si tiene que romper una declaración en una cadena:



# **Extraviar Punto y coma**

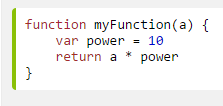
Debido a un punto y coma fuera de lugar, este bloque de código se ejecutará independientemente del valor de x:

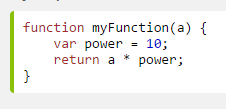


# **Rompiendo una sentencia return**

Se trata de un comportamiento por defecto JavaScript para cerrar un comunicado automáticamente al final de una línea.

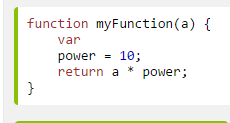
Debido a esto, estos dos ejemplos se devolverá el mismo resultado:



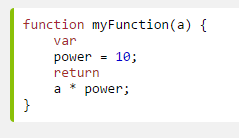


JavaScript también permitirá romper una declaración en dos líneas.

Debido a esto, el ejemplo 3 también devolverá el mismo resultado:

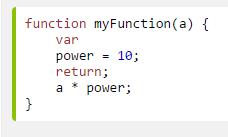


Pero, ¿qué pasará si se rompe la sentencia return en dos líneas como esta:



La función devolverá indefinido!

¿Por qué? Debido JavaScript piensa que quisiste decir:



# **Explicación**

Si una declaración es incompleta como:

var

JavaScript tratará de completar la declaración mediante la lectura de la siguiente línea:

power = 10;

Pero ya que esta declaración es completa:

return

JavaScript automáticamente cerrará así:

return;

Esto sucede porque el cierre (final) declaraciones con punto y coma es opcional en JavaScript.

JavaScript será cerrar la instrucción de retorno al final de la línea, ya que es una declaración completa.

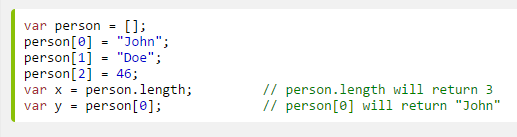
# **Acceso a matrices con índices con nombre**

Muchos lenguajes de programación soportan matrices con índices con nombre.

Las matrices con índices nombrados se llaman matrices asociativas (o hashes).

JavaScript no admite matrices con índices con nombre.

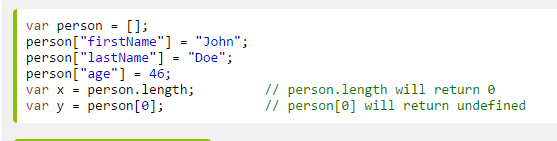
En JavaScript, arrays utilizan índices numerados:



En JavaScript, objetos utilizan named índices.

Si utiliza un índice llamado, cuando se accede a una matriz, JavaScript redefinirá la matriz a un objeto estándar.

Después de la redefinición automática, métodos y propiedades de la matriz producirán resultados indefinidos o incorrectas:



# **Finalización de una definición de la matriz con una coma**

## Incorrecto:

points = [40, 100, 1, 5, 25, 10,];

Algunos motores de JSON y JavaScript fracasarán, o comportarse de forma inesperada.

## Correcto:

points = [40, 100, 1, 5, 25, 10];

# **Poner fin a una definición de objeto con una coma**

## Incorrecto:

person = {firstName:"John", lastName:"Doe", age:46,}

Algunos motores de JSON y JavaScript fracasarán, o comportarse de forma inesperada.

## Correcto:

person = {firstName:"John", lastName:"Doe", age:46}

# **Indefinido no es nulo**

Con JavaScript, null es para objetos, indefinido es para variables, propiedades y métodos.

Para ser nula, un objeto tiene que ser definido, de lo contrario será indefinido.

Si desea probar si existe un objeto, esto generará un error si el objeto no está definido:

## Incorrecto:

if (myObj !== null && typeof myObj !== "undefined")

Debido a esto, debe probar typeof () primero:

## Correcto:

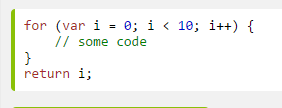
if (typeof myObj !== "undefined" && myObj !== null)

# **Esperando Ámbito de nivel de bloque**

JavaScript no crea un nuevo ámbito para cada bloque de código.

Es cierto en muchos lenguajes de programación, pero no es cierto en JavaScript.

Es un error común, entre los nuevos desarrolladores de JavaScript, para creer que este código devuelve undefined:



JavaScript Style Guide and Coding Conventions  
<http://www.w3schools.com/js/js_conventions.asp>

# **Guía de estilo JavaScript**

# **Convenciones de codificación**

## *Convenciones de codificación de JavaScript*

Son las directrices de estilo de programación.

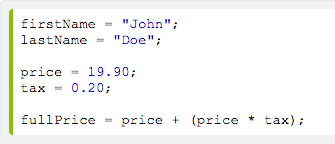
* Reglas de nomenclatura y de declaración de variables y funciones.
* Reglas para el uso de espacios en blanco, la sangría, y los comentarios.
* Programación de las prácticas y los principios

Coding conventions secure quality:

* Mejora la legibilidad del código
* Hacer el mantenimiento del código más fácil.

# Nombres de variables

Todos los nombres comienzan con una letra.



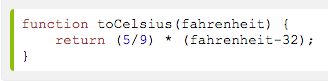
# Espacios alrededor de Operadores

Espacios alrededor de los operadores (= + - \* /), y después de las comas.

Screen Shot 2015-03-28 at 11.18.24 AM.png

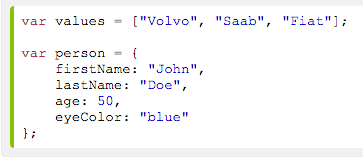
# Sangrado de código

Dar 4 espacios para el sangrado de bloques de código:

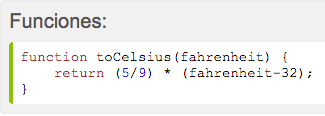
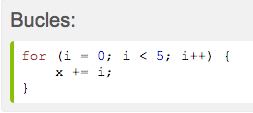


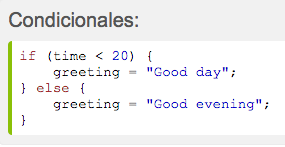
# Reglas Declaración

* Siempre terminar declaración con un punto y coma.



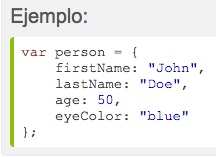
* Ponga el soporte de abertura en el extremo de la primera línea.
* Utilice un espacio antes del corchete de apertura.
* Ponga el soporte de cierre en una nueva línea, sin espacios iniciales.
* No finalice declaración complejo con un punto y coma.



# Reglas generales para definir objetos

* Coloque el soporte de la apertura en la misma línea que el nombre del objeto.
* Utilice dos puntos más un espacio entre cada propiedad y su valor.
* Use comillas valores de cadena, no en torno a valores numéricos.
* No añada una coma después de la última pareja propiedad-valor.
* Coloque el soporte de cierre, en una nueva línea, sin espacios iniciales.
* Siempre termine una definición de objeto con un punto y coma.



Objetos cortos pueden escribirse comprimidos, en una sola línea, así:

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

## Longitud de la línea <80

Evite líneas de más de 80 caracteres.

Si una sentencia JavaScript no cabe en una línea, el mejor lugar para romperlo, es después de un operador o una coma.

document.getElementById("demo").innerHTML =

"Hello Dolly."; hyp-hens, camelCase, or under\_scores

# Convenciones de nomenclatura

Usar el mismo convenio de denominación para todo el código.

* Nombres de variables y funciones escritas como camelCase
* Variable global escrito en MAYÚSCULAS
* Constantes (como PI) escritas en MAYÚSCULAS

En caso de utilizar HYP-hens, camelCase o under\_scores en nombres de variables?

Los guiones en HTML y CSS:

HTML5 atributos pueden empezar de datos (cantidad de datos, los datos de los precios).

CSS utiliza guiones en propiedad-nombres (font-size).

***Subraya:***

Muchos programadores prefieren usar subrayado (Fecha\_Nacimiento), sobre todo en las bases de datos SQL.

Subrayados son de uso frecuente en la documentación de PHP.

***PascalCase:***

PascalCase es a menudo preferido por los programadores de C.

***camelCase:***

camelCase se utiliza por sí JavaScript, por jQuery, y otras bibliotecas JavaScript.

# Cargando JavaScript en HTML

## <script src="myscript.js">

# Acceso a elementos HTML

Una consecuencia de la utilización de "desordenados" estilos HTML, podría dar lugar a errores de JavaScript.

Estas dos sentencias JavaScript producirán diferentes resultados:

var obj = getElementById("Demo")

var obj = getElementById("demo")

# Extensiones de archivo

Archivos HTML deben tener una extensión .html (no .htm).

Archivos CSS deben tener una extensión .css.

Archivos JavaScript debe tener una extensión .js.

# Utilice nombres de archivo en minúscula

La mayoría de los servidores web (Apache, Unix) son sensibles acerca de los nombres de archivo:

london.jpg no se puede acceder como London.jpg.

Otros servidores web (Microsoft, IIS) no distinguen entre mayúsculas y minúsculas:

london.jpg se puede acceder como London.jpg o london.jpg.

1. Si utiliza una mezcla de mayúsculas y minúsculas, lo que tienes que ser muy consistente.
2. Si se muda de un caso insensible, a un servidor de mayúsculas y minúsculas, incluso los pequeños errores se romperán su web.
3. Para evitar estos problemas, utilice siempre los nombres de archivo en minúsculas (si es posible).

# Rendimiento

Las convenciones de codificación no son utilizados por las computadoras. La mayoría de las reglas tienen poco impacto en la ejecución de los programas.

Sangría y espacios adicionales no son significativas en pequeños scripts.

Para el código en el desarrollo, se debe preferir la legibilidad. Grandes guiones de producción deben minified.

======== <http://www.w3schools.com/js/js_conventions.asp> ==========

# **JavaScript JSON**

JSON es un formato para almacenar y transportar datos.

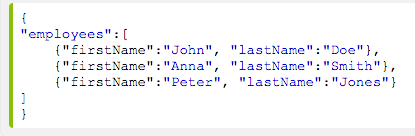
JSON se utiliza a menudo cuando se envían datos desde un servidor a una página web.

## ¿Qué es JSON?

* JSON significa J ava S cripta O bject N otación
* JSON es un formato de intercambio de datos ligero
* JSON es independiente del idioma \*
* JSON es "auto-descripción" y fácil de entender

## Ejemplo JSON

Una matriz de 3 registros de empleados (objetos):



## El formato JSON Evalúa a objetos JavaScript

El formato es sintácticamente idéntico al código para la creación de objetos de JavaScript.

Debido a esta similitud, un programa de JavaScript puede convertirse fácilmente a datos JSON en objetos nativos de JavaScript.

## Reglas de sintaxis JSON

* Los datos son en pares nombre / valor
* Los datos se separan por comas
* Las llaves contienen objetos
* Los corchetes tienen matrices

## Los datos JSON - Un nombre y un valor

Se escribe como pares nombre / valor, al igual que las propiedades de objetos de JavaScript.

Un par nombre / valor consiste en un nombre de campo (entre comillas), seguido de dos puntos, seguido de un valor:

"firstName":"John"

## Objetos JSON

Objetos JSON se escriben entre llaves.

Al igual que en JavaScript, los objetos pueden contener varios pares nombre / valor:

{"firstName":"John", "lastName":"Doe"}

## JSON Arrays

Arrays JSON se escriben entre corchetes.

Al igual que en JavaScript, un array puede contener objetos:

|  |
| --- |
| "employees":[  {"firstName":"John", "lastName":"Doe"},  {"firstName":"Anna", "lastName":"Smith"},  {"firstName":"Peter", "lastName":"Jones"}  ] |

En el ejemplo anterior, el objeto "empleados" es una matriz. Contiene tres objetos.

Cada objeto es un registro de una persona (con un nombre y un apellido).

## La conversión de un texto JSON a un objeto JavaScript

Un uso común de JSON es

1. Leer datos de un servidor web
2. Mostrar los datos en una página web.

## Crear una cadena de JavaScript que contiene sintaxis JSON

|  |
| --- |
| var text = '{ "employees" : [' +  '{ "firstName":"John" , "lastName":"Doe" },' +  '{ "firstName":"Anna" , "lastName":"Smith" },' +  '{ "firstName":"Peter" , "lastName":"Jones" } ]}'; |

Utilice el código JavaScript integrado ***JSON.parse*** () para convertir la cadena en un objeto de JavaScript

var obj = JSON.parse(text);

|  |
| --- |
| var text = '{"employees":[' +  '{"firstName":"John","lastName":"Doe" },' +  '{"firstName":"Anna","lastName":"Smith" },' +  '{"firstName":"Peter","lastName":"Jones" }]}'; |

|  |
| --- |
| obj = JSON.parse(text);  document.getElementById("demo").innerHTML =  obj.employees[1].firstName + " " + obj.employees[1].lastName; |

=================================================

JavaScript JSON  
<http://www.w3schools.com/js/js_json.asp>

# **JavaScript Performance**

Cómo acelerar su código JavaScript.

## Reducir la actividad en Loops

Cada declaración en un bucle, incluyendo la instrucción for, se ejecuta para cada iteración del bucle.

Búsqueda de declaraciones o asignaciones que se pueden colocar fuera del bucle.

## Better Code:

l = arr.length;

for (i = 0; i < l; i++) {

## Reduce DOM Access

El cceso al DOM HTML es muy lento, en comparación con otras sentencias de JavaScript.

Si usted espera para acceder a un elemento DOM varias veces, acceder a él una vez, y lo utilizan como una variable local:

|  |
| --- |
| <script>  var obj;  obj = document.getElementById("demo");  obj.innerHTML = " Hello";  </script> |

## 

## 

## Reduce DOM Size

Mantenga el número de elementos en el DOM HTML pequeño.

Esto siempre mejorará la carga de la página, y la velocidad de renderizado (visualización de la página), sobre todo en los dispositivos más pequeños.

Todo intento de buscar el DOM (como getElementsByTagName) se beneficiará de un DOM menor.

## Evite variables innecesarias

No crear nuevas variables si no planea guardar valores.

A menudo se puede reemplazar código como este:

var fullName = firstName + " " + lastName;

document.getElementById("demo").innerHTML = fullName;

Con este:

document.getElementById("demo").innerHTML = firstName + " " + lastName

## Delay JavaScript Loading

Poner guiones en la parte inferior del cuerpo de la página, permite que el navegador carge la página por primera vez.

Mientras que un script se está descargando, el navegador no iniciará ningún otra descargas.Además toda la actividad de análisis y representación podría ser bloqueado.

Una alternativa es utilizar aplazar = "true" en la etiqueta de script. El atributo aplazar especifica que el guión debe ser ejecutado después de la página ha terminado de analizar, pero sólo funciona para los scripts externos.

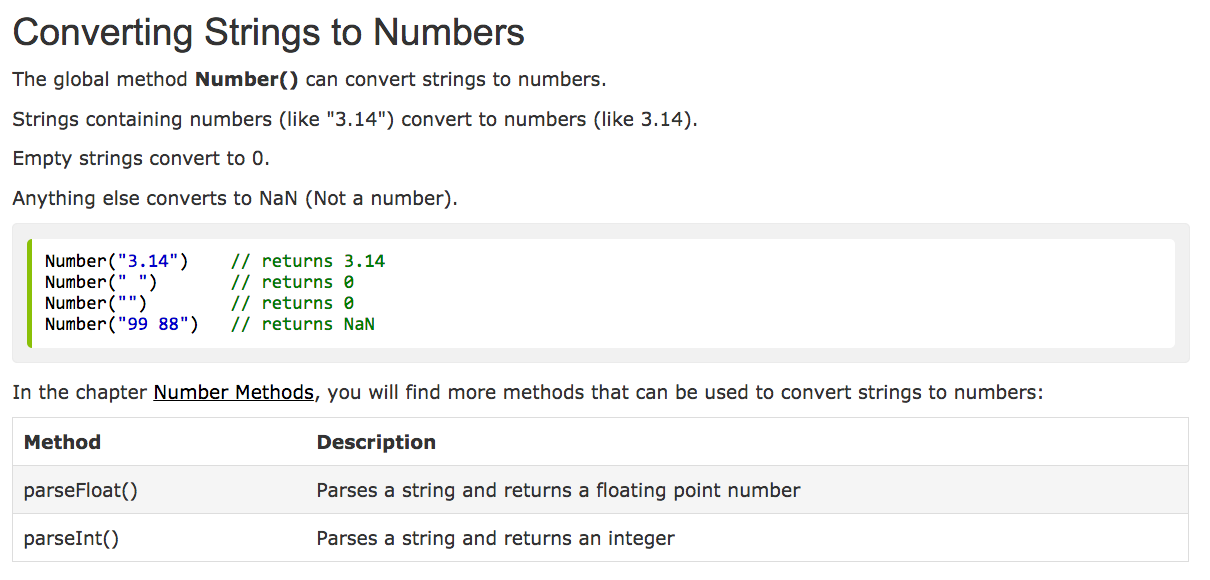
Si es posible, usted puede añadir su guión a la página con el código, después de que la página se ha cargado:

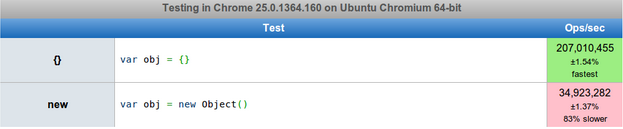
|  |
| --- |
| window.onload = downScripts;  function downScripts() {  var element = document.createElement("script");  element.src = "myScript.js";  document.body.appendChild(element);  } |

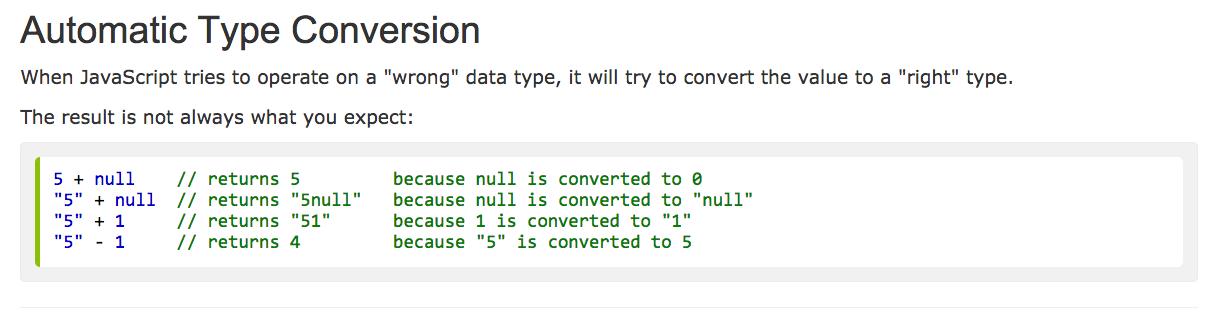
## Evitar el uso with

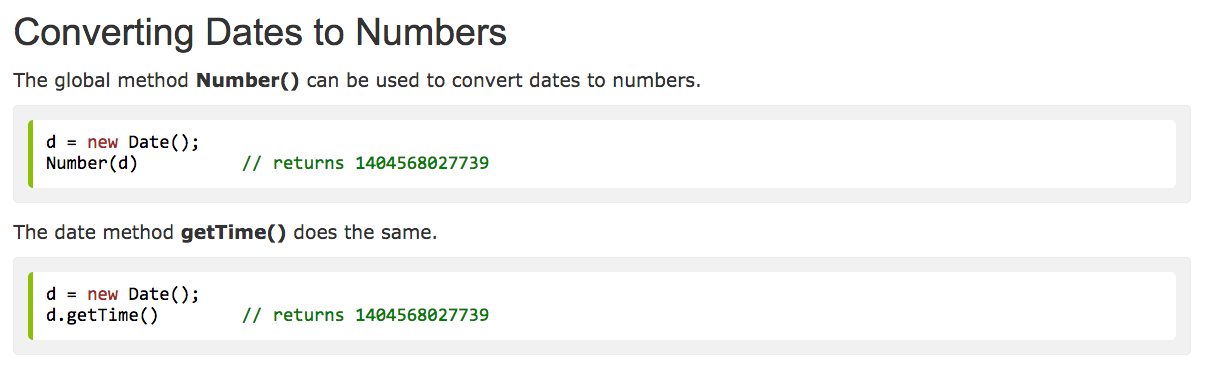
Evite el uso de la palabra clave with. Tiene un efecto negativo en la velocidad. También clutters hasta ámbitos de JavaScript.

El con la palabra clave no está permitido en modo estricto.









Exceptions (Acá pueden encontrar las instrucciones para la tarea programada #1)  
<http://jsfiddle.net/Lhxxc3yz/2/>



Funciones - patrones de invocación y arguments  
<http://jsfiddle.net/zoshtbm0/1/>

|  |
| --- |
| var app = {};  app.usuario = {  nombre : "Edison",  apellido : "Lizano",  ordenes : 0,  incrementarOrdenes : function(){ //Patron de invocacion METODO  this.ordenes++;  }  };  console.log(app.usuario.ordenes);  app.usuario.incrementarOrdenes();  console.log(app.usuario.ordenes);  function suma(a,b){//Patron de invocacion FUNCION  return a+b;  }  var Libro = function (nombre){ //Patron de invocacion CONSTRUCTOR  this.nombre=nombre;  };  var Persepolis = new Libro("Persepolis");  //ejemplo de arguments  suma(4,7,9,68);  function suma(a, b, c, d){  //no es un arreglo  for(var i=0; i<arguments.length; i++){  // console.log(arguments[i]);  }  } |

Augmenting y Cascade   
<http://jsfiddle.net/2pk3jn7f/1/>

|  |
| --- |
| <label id="lbl\_1">Clase #5 Funciones</label>  //Augmenting  Function.prototype.method = function(name, func){    if(!this.prototype[name]){  this.prototype[name] = func;  }  return this;  };  //MAL  function changeElementColor(c){  document.getElementById("lbl\_1").style.color = c;  }  //changeElementColor("Blue");  var elElemento = document.getElementById('lbl\_1');  //Augment  Element.method('cambiarColor', function(c){  this.style.color = c;  return this;  });  //Augment again  Element.method('agregarEsteHTML', function(s){  this.innerHTML = s;  return this;  });  //elElemento.cambiarColor('Red');  //elElemento.agregarEsteHTML('Clase #5 ya casi nos vamos');  elElemento.cambiarColor('Green').agregarEsteHTML('Con esto prodrian hacer su propio jQuery'); |

Timing  
<http://www.w3schools.com/js/js_timing.asp>

# **JavaScript Timing Events**

## 

## 

JavaScript Performance  
http://www.w3schools.com/js/js\_performance.asp

