Resumen JavaScript

Buenas Prácticas

- Evitar usar variables globales.
- Declarar siempre variables locales.
- Declarar las variables de primero.
- No declarar nunca números, strings o booleanos como objetos.
- No usar el new Object(), usar solo { } o “ ” …
- Tenga cuidado de que no se le cambie el tipo de variable, cuando le cambia el valor.
- Usar === en vez de ==.
- Asignar valores al parámetro por defecto.
- Evite el uso de eval()

Intro POO.

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 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 o una constante).

Conceptos fundamentales

La programación orientada a objetos es una forma de programar que trata de encontrar una solución a estos problemas. Introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

- ***Clase***: Definiciones de las propiedades y comportamiento de un tipo de objeto concreto.
- ***Herencia***: (Por ejemplo, herencia de la clase C a la clase D) es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D
- ***Objeto***: Instancia de una clase
- ***Método***: Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje".
- ***Evento*****:** Es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto).
- ***Atributos*****:** Características que tiene la clase.
- ***Mensaje*****:** Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

- ***Propiedad o atributo*:** Contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.
- ***Estado interno:*** Es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto
- ***Componentes de un objeto:*** Atributos, identidad, relaciones y métodos.
- ***Identificación de un objeto:*** Un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

Constructor

En programación orientada a objetos (POO), un constructor es una subrutina cuya misión es inicializar un objeto de una clase. En el constructor se asignan los valores iniciales del nuevo objeto.

Qué es JavaScript?

JavaScript es un lenguaje de scripting multiplataforma, orientado a objetos. JavaScript es un pequeño y liviano lenguaje; no es útil como lenguaje independiente, pero esta diseñado para ser fácilmente embebido en otros productos y aplicaciones, como ser web browsers.

JavaScript y Java

Javascript y Java son similares en algunos puntos, pero fundamentalmente diferentes en otros. El lenguaje Javascript se parece al de Java pero no tiene el tipado estático y comprobación de tipos fuerte. Javascript tiene sintaxis, convenciones de nombres y controles básicos de flujo parecidos a los de Java, por eso es que se le cambió el nombre de Livescript a javascript.

Investigaciones

- ***LowerCase:*** Convierte un String a mayúsculas, pero sin cambiar el String original.
- ***Eval():*** Evalúa una cadena de código JavaScript sin referenciar a un objeto en particular. Su sintaxis es eval(*cadena* [, *objeto*]) y recibe por parámetros una cadena o un objeto.
- ***Prompt***: Sirve para la entrada de datos por teclad. Cada vez que necesitamos ingresar un dato con esta función, aparece una ventana donde cargamos el valor.

Palabras Reservadas

- ***Break***: Interrumpe un bloque de instrucciones saltando a la primera instrucción que sigue al bloque que contiene el "break". Un uso apropiado evitará la formación de loop sin salida.
- ***Case*** – Switch: Nos permite evaluar el valor de una expresión para tomar una decisión en el flujo del ejecución. Es muy parecida a la estructura de control i**f/else** pero con **switch/case** podemos evaluar más de dos casos.
- ***Const***: Constantes en JavaScript.
- ***Continue***: Indica que se continue un bloque de instrucciones, pero interrumpiendo la iteración en ese punto y volviendo a comenzar desde el inicio del bloque.
- ***Do*** – ***While***: Crea un bucle que ejecuta una sentencia especificada, hasta que la condición de comprobación se evalúa como falsa. La condición se evalúa después de ejecutar la sentencia, dando como resultado que la sentencia especificada se ejecute al menos una vez.
- ***Export***: Permite a un script firmado proporcionar propiedades, funciones, y objetos a otro script firmado o no.
- ***For***: Crea un bucle que consiste en tres expresiones opcionales, encerradas en paréntesis y separadas por puntos y comas, seguidas de una sentencia ejecutada en un bucle.
- ***Function***: Declara una función con los parámetros especificados. Puede también definir funciones usando el constructor Function y el function (expresión function).
- ***If – Else:*** Ejecuta una sentencia si una condición específicada es evaluada como verdadera. Si la condición es evaluada como falsa, otra sentencia puede ser ejecutada.
- ***Instanceof***: Devuelve verdadero si el objeto especificado es del tipo especificado.
- ***Let***: Controla el ámbito de una variable y controlar mejor su propagación.
- ***New***: Crea una instancia de un tipo de objeto a partir de una función constructora nativa ó definida por el usuario.
- ***Return***: Especifica el valor devuelto por una función.
- ***This***: Su valor está determinado por cómo se llama a la función. No puede ser establecida por una asignación en tiempo de ejecución, y esto puede ser diferente cada vez que la función es llamada.
- ***Throw***: Lanza una excepción definida por el usuario.
- ***Try – Catch – Finally:*** Con try especificamos una serie de sentencias Javascript que vamos a tratar de ejecutar. Con catch especificamos lo que queremos realizar si es que se ha cazado un error en el bloque try. Finally se ejecuta después de try y catch. Se ejecuta siempre.
- ***Typeof***: Devuelve una cadena que indica el tipo de variable, cadena, palabra clave u objeto sin evaluarlo.
- ***Var***: Declaración de una variable, opcionalmente inicializada a un valor.

- ***Void***: Específica una expresión que se evalúa sin devolver un valor.
- ***Yield***: Se utiliza para hacer una pausa y reanudar una función generadora

Motor de renderizado

- Es software que toma contenido marcado e información de formateo y luego muestra el contenido ya formateado en la pantalla de aplicaciones.
- Se usan típicamente en navegadores web, clientes de correo electrónico.
- Todos los navegadores web incluyen necesariamente algún tipo de motor de renderizado.

Algunos de los motores de renderizado más notables son:

- Gecko, utilizado en Mozilla Suite
- Trident, el motor de Internet Explorer para Windows.
- Tasman, el motor de Internet Explorer para Mac.
- WebKit, el motor de Epiphany, Safari.
- Blink, el nuevo motor de Google Chrome y Opera

V8 (motor JavaScript)

- Es un motor de código abierto para JavaScript creado por Google.
- Está escrito en C++.
- Es usado en Google Chrome.
- Está integrado en el navegador de internet del sistema operativo Android 2.2 “Froyo”.

Programación del lado del cliente y del lado del servidor

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

- Servidor: Provee el servicio

- Cliente: Solicita el servicio.
- El Usuario: Utiliza el cliente con el fin de navegar por la web, rellenar formularios, ver videos en línea, etc.

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

Ejemplo de lenguajes:

- PHP
- ASP.Net en C #, C ++ o Visual Basic.

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
Hacer páginas web interactivas.
Haga cosas suceda de forma dinámica en la página web.
Interactuar con el almacenamiento temporal y el almacenamiento local (cookies, localStorage).
Enviar solicitudes al servidor, y recuperar datos de él.

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 :

- JavaScript
- HTML *
- CSS *

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 son realmente "lenguajes de programación"

Lenguaje Script

Un lenguaje de programación o lenguaje de script es un lenguaje de programación que soporta scripts, programas escritos para un ambiente especial en tiempo de ejecución que puede interpretar (en lugar de compilar) y automatizar la ejecución de tareas que, alternativamente, podría ser ejecutado de una en una por un ser humano operador.

Lista de motores de ECMAScript

Un motor de ECMAScript es un programa que ejecuta el código fuente escrito en una versión de la norma del lenguaje ECMAScript, por ejemplo, JavaScript.

Carakan: Un motor de JavaScript desarrollado por Opera
Chakra: Un motor de JScript utilizarse en Internet Explorer.
SpiderMonkey: Un motor de JavaScript en aplicaciones de Mozilla Geck

SquirrelFish: El motor JavaScript de WebKit de Apple Inc. También conocido como Nitro.
Tamarin: Un motor de ActionScript y ECMAScript usado en Adobe Flash.
V8: Un motor de JavaScript se utiliza en Google Chrome.
JavaScriptCore: Un intérprete de JavaScript derivado originalmente de RV. Se utiliza en el proyecto WebKit y aplicaciones como Safari.
Nashorn: Un motor de JavaScript se utiliza en Oracle Java Development Kit (JDK).

Investigaciones

- Diferencia entre Window.Onload y Document.Ready

El Document.Ready ocurre después que el HTML ha sido cargado, mientras que el Window.Onload ocurre cuando el contenido (por ejemplo las imágenes) han sido cargadas. Document.Ready es un evento específico de Jquery, Window.Onload es un evento estándar del DOM.

- Diferencia entre Window.Onload y Window.Onunload

Onload es aquel que se produce cuando un navegador carga un documento HTML o una imagen.
Onunload tiene como misión ejecutar un script cuando la página web actual se descarga, ya sea porque se accede a otra página o porque se pulsan los botones de retroceder y avanzar.

- ¿Para qué sirve NoScript?

Muestra un mensaje al usuario cuando su navegador no puede ejecutar JavaScript.

- Diferencia entre Null y Undefined

    - Undefined: para Javascript, **no existe**. O bien no ha sido declarada o jamás se le asignó un valor.
    - Null: para Javascript, **la variable existe**. En algún momento, explícitamente, la variable se estableció a null.

La precedencia de operadores

La precedencia de operadores determina el orden en que se evalúan los operadores. Los operadores con mayor precedencia se evalúan primero.

Asociatividad (ARREGLAR!)

La asociatividad determina el orden en que se procesan los operadores con la misma precedencia. Por ejemplo, considere una expresión:
una OP OP b c
Asociatividad por la izquierda (de izquierda a derecha) significa que se procesa como (a OP b) c OP, mientras asociatividad por la derecha (de derecha a izquierda) significa que se interpreta como una OP (b OP c). Operadores de asignación son asociativo por la derecha, por lo que puede escribir:
a = b = 5;
con el resultado esperado que ayb obtener el valor 5. Esto se debe a que el operador de asignación devuelve el valor que se le asigna. En primer lugar, b se establece en 5. A continuación, la una se establece en el valor de b.

Good parts

Porque JavaScript?

Hay 2 respuestas por las cuales se usa JS. La primera es que no tenemos otra opción ya que es el único lenguaje que se encuentra en todos los navegadores. Y la otra es que a pesar de sus deficiencias JS es realmente bueno. Es ligero y expresivo.

Gramática

- Whitespace: Para comentar un bloque se utiliza /* */. Y para comentar al final de una línea usamos //.
- Names: No usar como nombre una palabra reservada

Converting Strings to Numbers

The global method **Number()** can convert strings to numbers.

Strings containing numbers (like "3.14") convert to numbers (like 3.14).

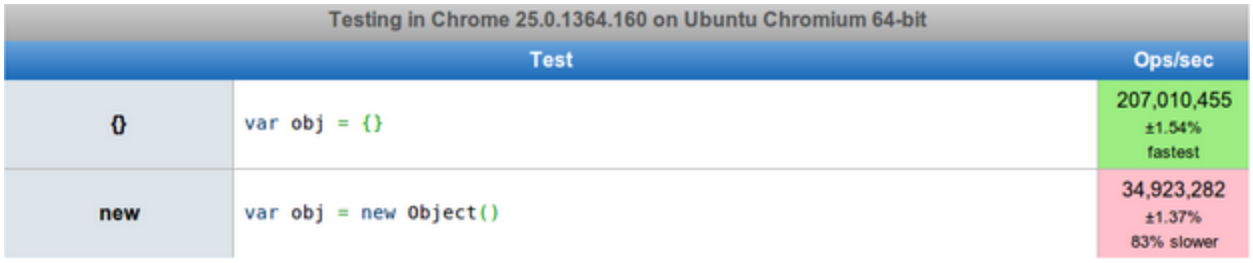
Empty strings convert to 0.

Anything else converts to NaN (Not a number).

```
Number("3.14")    // returns 3.14
Number(" ")       // returns 0
Number("")        // returns 0
Number("99 88")   // returns NaN
```

In the chapter Number Methods, you will find more methods that can be used to convert strings to numbers:

Method	Description
parseFloat()	Parses a string and returns a floating point number
parseInt()	Parses a string and returns an integer



Converting Dates to Numbers

The global method **Number()** can be used to convert dates to numbers.

```
d = new Date();
Number(d)          // returns 1404568027739
```

The date method **getTime()** does the same.

```
d = new Date();
d.getTime()        // returns 1404568027739
```

Automatic Type Conversion

When JavaScript tries to operate on a "wrong" data type, it will try to convert the value to a "right" type.

The result is not always what you expect:

```
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
```

String.prototype.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.

Sintaxis:

```
cadena.indexOf(valorBuqueda[, indiceDesde])
```

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`.

JavaScript Hoisting

Hoisting es el comportamiento predeterminado de JavaScript de las declaraciones de pasar a la parte superior.

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.

JavaScript Use Strict

No es una declaración, sino una expresión literal, ignorado por las versiones anteriores de JavaScript.

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 se ejecutará en modo estricto).

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

Declaración global:

```
"use strict";
x = 3.14;       // This will cause an error
myFunction();   // This will also cause an error

function myFunction() {
      x = 3.14;
}
```

Declaración local:

```
x = 3.14;       // This will not cause an error.
myFunction();   // This will cause an error

function myFunction() {
   "use strict";
      x = 3.14;
}
```

Errores comunes de JavaScript

Accidentalmente Usando el operador de asignación
Programas JavaScript puede 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:

```
var x = 0;
if (x == 10)
```

```
Esta sentencia if devuelve true (quizás no tan esperado), porque 10 es
verdadero:
var x = 0;
if (x = 10)
```

```
Esta sentencia if devuelve false (quizás no tan esperado), porque 0 es
falsa:
var x = 0;
if (x = 0)
```

Guía de estilo JavaScript y convenciones de codificación

Convenciones de codificación de JavaScript
Las convenciones de codificación son las directrices de estilo de programación. Por lo general se refieren a:
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
Las convenciones de codificación de calidad segura:

Mejora la legibilidad del código
Hacer el mantenimiento del código más fácil
Las convenciones de codificación se pueden documentar las reglas para los equipos a seguir, o simplemente ser su práctica de codificación individual.

Nombres de variables

En W3schools utilizamos camelCase de nombres de identificadores (variables y funciones).
Todos los nombres comienzan con una letra.
En la parte inferior de esta página, usted encontrará una discusión más amplia sobre las reglas de nombres.

```
firstName = "John";
lastName = "Doe";

price = 19.90;
tax = 0.20;

fullPrice = price + (price * tax);
```

Espacios alrededor de Operadores

Siempre ponga espacios alrededor de los operadores (= + / *), y después de las comas:

```
var x = y + z;
var values = ["Volvo", "Saab", "Fiat"];
```