**Resumen JavaScript**

**Kenia Montano Granados**

Contenido

[¿Qué es JavaScript? 2](#_Toc410244406)

[JavaScript y Java 3](#_Toc410244407)

[Palabras Reservadas en JavaScript 3](#_Toc410244408)

[Futuras Palabras Reservadas en JavaScript. 4](#_Toc410244409)

[toLowerCase() 5](#_Toc410244410)

[toUpperCase() 5](#_Toc410244411)

[alert() 5](#_Toc410244412)

[prompt() 7](#_Toc410244413)

[eval() 7](#_Toc410244414)

[confirm() 8](#_Toc410244415)

[open() 9](#_Toc410244416)

[close() 9](#_Toc410244417)

[Programación Orientada a Objeto 9](#_Toc410244418)

[Conceptos 9](#_Toc410244419)

[¿Se Puede Guardar una función en una variable? 10](#_Toc410244420)

[Diferencia entre $(document).ready() y window. onload 10](#_Toc410244421)

[Onunload 11](#_Toc410244422)

[<noscript> 11](#_Toc410244423)

[Typeof 11](#_Toc410244424)

[Null 12](#_Toc410244425)

[Undefined 12](#_Toc410244426)

[Diferencia entre == y === 12](#_Toc410244427)

[Date() 13](#_Toc410244428)

[JavaScript The Good Parts 15](#_Toc410244429)

[Capitulo 1-Good Parts 15](#_Toc410244430)

[Capítulo 2-Grammar 15](#_Toc410244431)

[Capítulo 3-Objects 16](#_Toc410244432)

# ¿Qué es JavaScript?

Es un lenguaje de programación orientada a objetos, basada en prototipos que permite mejoras en el interfaz del usuario y páginas web dinámicas.

Una página dinámica se refiere al movimiento de textos, imágenes, animaciones o acciones que se realicen al presionar un botón o al hacer clic en la página.

Es un lenguaje de programación que permite a los desarrolladores crear acciones en su página web.

Los programas escritos por JavaScript se pueden probar desde el navegador sin necesidad de procesos intermedios.

# JavaScript y Java

JavaScript y Java son similares en algunos puntos. JavaScript tiene sintaxis, convenciones de nombres y controles básicos de flujo parecidos a los de Java, por eso se le cambio el nombre de Livescript a JavaScript. JavaScript soporta un sistema en tiempo de ejecución basado en un pequeño número de tipos de datos que representan valores numéricos, booleanos y cadenas. Tiene un modelo de objetos basado en prototipos este provee herencia dinámica.

# Palabras Reservadas en JavaScript

Las palabras reservadas son aquellas que no se pueden utilizar como identificadores ya que tienen un lenguaje específico utilizado para crear una función las cuales son:

break: permite terminar de forma abrupta un bucle.

case: pertenece al switch y este hace referencia a cada opción(caso) que se debe evaluar hasta que se cumpla alguna opción(caso) y se realizaría la función correspondiente.

catch: permite manejar el error.

continue: permite saltarse algunas repeticiones del bucle.

default: permiten parámetros formales para inicializar con valores por defecto si no hay valor o indefinido se pasa

delete:  retira una propiedad de un objeto.

do: es una variante del bucle while. Este bucle se ejecutará el bloque de código una vez, antes de comprobar si la condición es verdadera, entonces se repetirá el bucle mientras la condición es verdadera.

else: para especificar un bloque de código que se ejecutará, si la misma condición es falsa.

finally: permite ejecutar código, después de tratar de atrapar, sin importar el resultado.

for: crea un bucle que consiste en tres expresiones opcionales.

function: cada función en JavaScript es actualmente un objeto Function.

if: sirve para especificar un bloque de código que se ejecutará, si la condición especificada sea verdadera.

in: retorna true si la propiedad especificada está en el objeto especificado.

instanceof: devuelve verdadero si el objeto especificado es del tipo especificado.

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.

switch: sirve para especificar muchos bloques alternativos de código para ser ejecutado.

this: el valor de this 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.

try: permite probar un bloque de código para los errores.

throw: permite crear errores personalizados.

typeof: devuelve una cadena que indica el tipo del operando sin evaluarlo.

var: declaración de una variable, opcionalmente inicializada a un valor.

void:  especifica una expresión que se evalúa sin devolver un valor.

while: crea un bucle que ejecuta una sentencia especificada mientras cierta condición se evalúe como verdadera. Dicha condición es evaluada antes de ejecutar la sentencia.

with: se extiende la cadena de ámbito de un comunicado.

# Futuras Palabras Reservadas en JavaScript.

Estas palabras no se deben utilizar como identificadores, aunque no posean ahora una función ya que en el futuro se utilizaran. Estas palabras son:

* class
* enum
* export
* extends
* import
* super

# toLowerCase()

Pasa una palabra en mayúscula a minúscula.

**Ejemplo:**

var str = "Hello World!";  
var res = str.toLowerCase();

El resultado de res será: hello world!

**Explicación:**

Lo que hace es que crea una variable (str) con alguna palabra escrita en mayúscula y en otra variable (res) se utiliza la variable de la palabra (str) y el toLowerCase() y esto hace que la palabra almacenada en str cambie a minúscula en la variable res.

# toUpperCase()

Hace lo contrario a toLowerCase(), es decir, pasa de minúscula a mayúscula.

**Ejemplo:**

var str = "Hello World!";  
var res = str.toUpperCase();

El resultado de res será: HELLO WORLD!

**Explicación:**

En este ejemplo pasa exactamente lo mismo al ejemplo anterior con la variante que la palabra va a pasar a mayúscula y no minúscula.

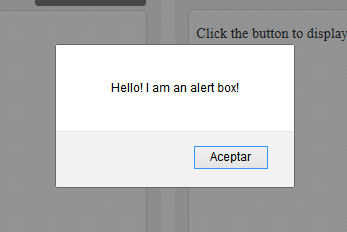
# alert()

Esta muestra un cuadro de alerta con el mensaje que este dentro de los paréntesis. Los mensajes pueden ser un string o una variable que contenga información.

**Ejemplo 1:**

alert("Hello! I am an alert box!!");

El resultado de este alert sera:



**Explicación:**

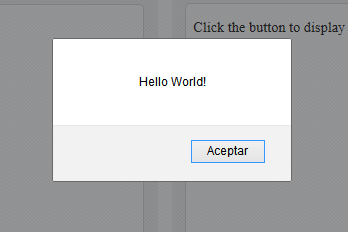
Es muy sencillo solamente aparece el texto que esta dentro del paréntesis.

**Ejemplo 2:**

var str = "Hello World!";

alert(str);

El resultado de este alert será:



**Explicación:**

Es lo mismo al ejemplo anterior solamente que utilizamos una variable para almacenar el mensaje.

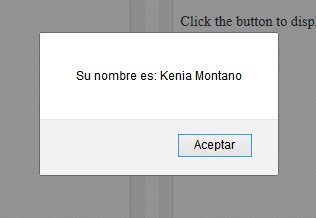
También podemos mezclar ambos ejemplos.

**Ejemplo 3:**

var str = "Kenia Montano";

alert(“Su nombre es: ”+ str);

El resultado sera:



**Explicación:**

Lo que hacemos es concatenar el texto del paréntesis con una variable (str) ya definida con un “+”.

# prompt()

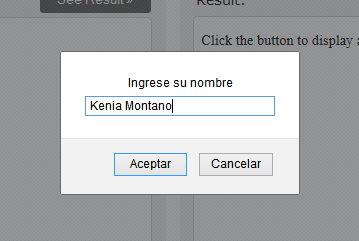
Este muestra un mensaje pidiendo el usuario que ingrese un dato (nombre, edad, nacionalidad, color, etc.) al ingresar lo que se pide, saldrá un mensaje, este mensaje dependerá de la información suministrada.

Este se puede almacenar en una variable para luego se utilice la variable en el mensaje, esta variable tendrá el dato introducido por el usuario.

**Ejemplo:**

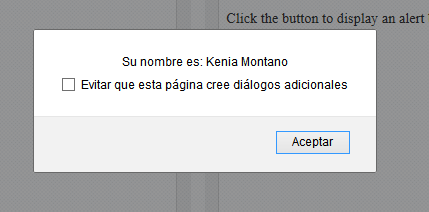
var str = prompt(“Ingrese su nombre”);

El resultado del prompt será:



alert(“Su nombre es: “+str);

El resultado del alert será:

**Explicación:**

El prompt() solicita el nombre y lo almacena en la variable luego por medio de un alert() se imprime el mensaje con el texto del alert mas el dato almacenado en la variable que fue ingresado por el usuario.

# eval()

Evalúa o ejecuta la expresión que se encuentre dentro del paréntesis.

**Ejemplo:**

var x = 10;  
var y = 20;  
var a = eval("x \* y") + "<br>";  
var b = eval("2 + 2") + "<br>";  
var c = eval("x + 17") + "<br>";  
  
var res = a + b + c;

El resultado será:

a=200  
b=4  
c=27

res=231

**Explicación:**

Se crean primero dos variables (x,y) donde se les da un valor (10,20) y estas mismas variables se utilizan para realizar una multiplicación en la variable a (10 \* 20=200), en la variable b solamente da los números y la operación se realiza con eval() y en variable c (10 + 17=27) todas estas operaciones la realiza la función eval(). En res solamente se suma a + b +c (200+4+27=231).

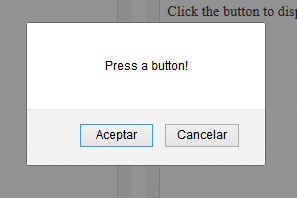
# confirm()

Este muestra un mensaje de confirmación con dos botone ok y cancelar, es utilizado para que el usuario verifique o acepte alguna función.

**Ejemplo:**

confirm("Press a button!");

El resultado de confirm será:



**Explicacion:**

El confirm solamente tira este mensaje con ok y con un cancel para que el usuario escoja.

# open()

Abre una nueva ventana en el navegador para que el usuario realice alguna función opcionalmente se le puede dar una url para que esta se dirija al página.

# close()

Hace lo contrario a open, es decir, cierra la ventana actual.

# Programación Orientada a Objeto

Un objeto posee estado, comportamiento e identidad.

**Estado:** Datos e información del objeto.

**Comportamiento**: Es la respuesta a los métodos o mensajes que se le den, es decir, lo que el objeto puede realizar.

**Identidad**: Es la propiedad que diferencia al objeto de los demás.

## Conceptos

**Clase:** Define las propiedades y comportamiento de un objeto.

**Herencia:** Cuando un objeto hereda atributos y operaciones de otro.

**Objeto:** Este puede poseer datos y métodos que reaccionen a eventos. Pueden asociarse a objetos reales o objetos internos del sistema.

**Método:** Lo que el objeto puede hacer.

**Evento:** La reacción que desencadena un objeto.

**Atributos:** Características.

**Mensaje:** Es el medio en que se comunican los objetos donde se ordena alguna ejecución.

**Propiedad o Atributo:** Contiene los datos asociados a un 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:** El constructor es llamado en el momento de la creación de la instancia (el momento en que se crea la instancia del objeto). El constructor es un método de la clase. El constructor se usa para establecer las propiedades del objeto o para llamar a los métodos para preparar el objeto para su uso.

# ¿Se Puede Guardar una función en una variable?

Si se puede guardar una función en una variable para luego utilizar el valor retornado por la función, si no se guarda en una variable el valor retornado de la función se pierde, es decir, no es utilizado.

**Ejemplo:**

function calculaPrecioTotal(precio, porcentajeImpuestos) {

var gastosEnvio = 10;

var precioConImpuestos = (1 + porcentajeImpuestos/100) \* precio;

var precioTotal = precioConImpuestos + gastosEnvio;

return precioTotal;

}

var precioTotal = calculaPrecioTotal(23.34, 16);

**Explicación:**

Lo primero que hace es crear una función que introduce por parámetros precio, porcentajeImpuestos y dentro de la función realiza las operaciones al final retorna (return) el valor que se encuentra en la variable precioTotal y ese va a ser el resultado de la función, fuera de la función se crea otra variable precioTotal donde se almacena la función que le da los valores a los parámetros calculaPrecioTotal(23.34, 16). Despues de esto podríamos utilizar sin ningún problema la variable precioTotal ya que esta contiene el resultado de la función.

# Diferencia entre $(document).ready() y window. onload

**$(document).ready:**

El .ready() cuando DOM(Document Obect Model) ha cargado.

Como este evento se da cuando se carga todo el documento es un buen momento para que se carguen los eventos y funciones de jQuery.

**window. onload:**

Este ejecuta un script cuando la página web carga completamente el contenido incluyendo imágenes, archivos de comandos, archivos CSS, etc.

El onload también se utiliza para comprobar el tipo y versión de navegador del visitante para mostrar la información correcta y se puede utilizar para hacer frente a los cookies.

# Onunload

Este ocurre cuando el usuario se desplaza a otra página por ejemplo al hacer click a un enlace, envía un formulario, cierra la ventana del navegador.

Sintaxis en HTML:

<*element* onunload="*myScript*">

Sintaxis en JavaScript:

object.onunload=function(){*myScript*};

# <noscript>

Define un contenido alternative para los usuarios tienen problemas con los comandos del navegador o que el navegador no admita script.

Es un elemento que se puede usar en el <head> y <body>.

Este se activara siempre que el navegador no admita scipt o que se desactiven.

# Typeof

Da el tipo de variable, es decir, según su valor da si es de string (texto), number (numero), boolean(true o false) , function(función) u object.

El typeof solo retornara el tipo de variable que corresponda.

**Ejemplo:**

typeof "John"                // Returns string   
typeof 3.14                  // Returns number  
typeof false                 // Returns boolean  
typeof [1,2,3,4]             // Returns object  
typeof {name:'John', age:34} // Returns object

typeof function(){}  // Returns function

**Explicación:**

Es muy sencillo a cada dato se le antepone un typeof y en la parte de comentario (//) solamente dice que clase de variable es.

# Null

Null en JavaScript es nada. Es algo que no existe pero aun asi corresponde en JavaScript a un objeto.

**Ejemplo:**

var person = null;    // Value is null, but type is still an object

**Explicación:**

Como pueden ver el valor es nulo pero el tipo es un object.

# Undefined

Es cuando una variable no posee valor. Y el typeof es undefined.

**Ejemplo:**

var person;              // Value is undefined, type is undefined

# Diferencia entre == y ===

**==**

Cuando se igualan dos datos y estos no poseen el mismo tipo pero si el mismo dato.

**Ejemplo:**

var num1 = “25”;

var num2 = 25;

if(num1==num2){

alert(“True”)

} else{

alert(“False”)

}

**Explicación:**

En este caso el mensaje que saldrá es “True” porque ambos poseen el mismo valor aunque no el mismo tipo.

**===**

Cuando se igualan dos datos y estos deben poseer el mismo tipo y el mismo dato.

**Ejemplo:**

var num1 = “25”;

var num2 = 25;

if(num1==num2){

alert(“True”)

} else{

alert(“False”)

}

**Explicación:**

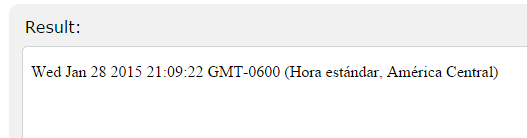
En este caso el mensaje que saldrá es “False” porque no poseen el mismo tipo aunque ambos posean el mismo valor.

# Date()

Permite trabajar con fechas (años, meses, días, minutos, segundos, milisegundos).

**Ejemplo:**

<p id="demo"></p>  
  
<script>  
document.getElementById("demo").innerHTML = Date();  
</script>



**Explicación:**

En este ejemplo aparecerá la fecha exacta en el momento que el código se corra.

Fecha objetos se crean con el **new Date ()** constructor.

Hay **4 maneras** de iniciar una fecha:

new Date()  
new Date(milliseconds)  
new Date(dateString)  
new Date(year, month, day, hours, minutes, seconds, milliseconds)

Al usar new Date() crea un objeto que da la hora y la fecha actual, al crear new Date(Fecha especificada) da la fecha que dice en los paréntesis y al crear new Date(numero) da el momento cero mas el numero.

## Propiedades Date

**Constructor:** Devuelve la función que creó el prototipo del objeto Date.

**Prototype:** Le permite agregar propiedades y métodos a un objeto.

Algunos métodos son:

|  |  |
| --- | --- |
| Método | Función |
| getDate(), getDay(), getMonth(), getFullYear() | Devuelve el día del mes, el día de la semana, el mes y el año respectivamente |
| getHours(), getMinutes(), getSeconds(), getMilliseconds() | Devuelve la hora, los minutos, los segundos y los milisegundos respectivamente |
| getTime() | Devuelve el número de milisegundos desde la medianoche del 01 de enero 1970 |
| getTimezoneOffset() | Devuelve la diferencia horaria entre la hora UTC y la hora local, en minutos |

# JavaScript The Good Parts

## Capitulo 1-Good Parts

JavaScript es un lenguaje importante porque es el idioma del navegador web. JavaScript se basa en algunas ideas muy buenas y algunas muy malas.

Funciones de JavaScript son objetos de primera clase con ámbito léxico. Tiene más en común con Lisp(Es con piel de c) y Scheme que con Java.

## Capítulo 2-Grammar

**Whitespace:** Puede tomar la forma de caracteres del formato o comentarios.

**Name:** es una carta seguida opcionalmente por una o más letras, dígitos o guiones bajos. Se utilizan para declaraciones, variables, parámetros, nombres de propiedades, operadores y etiquetas.

**Number:** JavaScript tiene un solo tipo de número. 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. El valor NaN es un valor numérico que es el resultado de una operación que no puede producir un resultado normal.

**String:** es una cadena literal que se pueden envolver en comillas simples o dobles. El \ (barra invertida) es el carácter de escape.

**Statements:** Las declaraciones tienden a ser ejecutados en orden de arriba a abajo. La secuencia de ejecución puede ser alterado por if(cambia el flujo del programa basado en el valor de la expresión), switch(realiza una rama de múltiples vías.), while(realiza un bucle simple), for(es una declaración de bucle más complicado), do(es como la instrucción while excepto que la expresión se ensaya después el bloque se ejecuta en lugar de antes), break(la salida de una sentencia de bucle o de una sentencia switch), return(hace que el pronto retorno de una función) , throw(lanza una excepción) y por la invocación de la función. Un bloque es un conjunto de sentencias envueltos entre llaves.

**Expressions:** Las expresiones más simples son un valor literal, una variable, un valor integrado, una expresión envuelto en paréntesis, una expresión precedida por un operador de prefijo. Los paréntesis se pueden utilizar para alterar la prioridad normal.Los valores producidos por typeof son number, string, boolean, undefined, function y object.

* Si el operando es null el resultado será object.
* El operador + añade o concatena. Si lo que desea es sumar asegurese de que ambos operandos sean números.
* El operador / puede producir un resultado no entero, incluso si ambos operandos son enteros.
* El operador && produce el valor de su primer operando si el primer operando es false, de lo contrario, se produce el valor del segundo operando.
* El operador || produce el valor de su primer operando si el primer operando es true, de lo contrario, se produce el valor del segundo operando.

**Literals:** 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. Literales de conjunto es una notación conveniente para especificar nuevos arrays.

**Functions:** . Puede tener un nombre opcional, especificar una lista de parámetros y en su cuerpo puede incluir definiciones y declaraciones.

## Capítulo 3-Objects

Los tipos simples de JavaScript son number, string, boolean (true y false), null y undefined. Todos los demás valores son objects. Un objeto es un contenedor de propiedades, donde una propiedad tiene un nombre y un valor. Son útiles para la recopilación y organización de datos.

**Object Literals:** Un objeto literal es un par de llaves que rodean cero o más pares nombre / valor.El nombre de una propiedad puede ser cualquier cadena, incluyendo la cadena vacía. El valor de una propiedad puede ser obtenida de cualquier expresión, incluyendo otro objeto literal.

**Retrieval:** Los valores se pueden recuperar de un objeto envolviendo una expresión de cadena en un [] sufijo.

**Update:** 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.

**Reference:** Objetos se pasan alrededor por referencia.

**Prototype:** Cada objeto 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. El enlace prototipo se utiliza sólo en la recuperación.

**Reflection:** Es fácil de inspeccionar 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.

**Enumeration:** La enumeración incluirá todas las propiedades, incluyendo funciones y propiedades de prototipo que podría no estar interesado 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:** El operador de eliminación se puede utilizar para eliminar una propiedad de un objeto. Se eliminará una propiedad del objeto, si lo tiene.

**Global Abatement:** JavaScript hace que sea fácil de definir variables globales que pueden contener todos los activos de su aplicación. Desafortunadamente, las variables globales debilitan la resistencia de programas y deben evitarse.