## JavaScript

Kimberly Quirós Jiménez

v Prompt: Muestra un diálogo de campo de formulario con botones Ok y Cancel, un texto definido por el primer parámetro enviado a la función y un input de texto con valor predeterminado definido por el segundo parámetro.

Código:

Var name = prompt(“Please enter your name:”);

v Alert: Crea una caja con un texto que será mostrado al usuario con un botón de aceptar.

Código:

Alert(“This a custom alert box”);

v toLowerCase: Pasa los una cadena de caracteres en minúscula.

Codigo:

v Eval: Evalúa una cadena de código.

v Concatenar (números o letras): Se pueden concatenar cadenas de varias formas en JavaScript.

Lo normal es usar el operador (+).

Var name = “Kimberly”;

Var lastname = “Quiros”;

Var result = name + lastname

Se puede concatenar las cadenas de texto que sean necesarias con el operador (+).

También se puede usar la asignación (+=) para concatenar varias cadenas.

|  |  |  |
| --- | --- | --- |
| var oldValue = "creando"; | |  |
|  | oldValue += " un tutorial"; | |
|  |  | |
|  |  |  |

Palabras Reservadas

Futuras Palabras Reservadas

Buenas Prácticas:

Avoid Global Variables:

Minimiza el uso de las Variables Globales. Las variables globales y las funciones pueden ser sobrescritas por otros Script.

Siempre declarar variables locales:

Todas las variables en una función podrían ser declaradas como variable local.

Declaraciones arriba:

Esto es una buena práctica al poner todas las declaraciones arriba de cada string o función. Esto hace que sea más fácil evitar variables globales no deseadas.

var num1 , num2;

var resultado;

num1 = 54;

num2= 3;

resultado = num1 + num2;

Nunca declare String, Number o Booleans como Object:

Siempre trate números, cadenas, o booleanos como valores primitivos. No como objetos.

var x = "John";

var y = new String("John");

(x === y)

v Web browser engine:

Un **Web browser engine**  es software que toma contenido marcado (como HTML, XML, archivos de imágenes, etc.) e información de formateo (como CSS, XSL, etc.) y luego muestra el contenido ya formateado en la pantalla de aplicaciones.

Algunos de los motores de renderizado más notables son:

·  [Gecko](http://es.wikipedia.org/wiki/Gecko_(motor_de_navegaci%C3%B3n)), utilizado en Mozilla Suite, y otros navegadores como [Galeon](http://es.wikipedia.org/wiki/Galeon).

·  [Trident](http://es.wikipedia.org/wiki/Trident_(motor_de_navegaci%C3%B3n)), el motor de Internet Explorer para Windows.

·  [KHTML](http://es.wikipedia.org/wiki/KHTML)/[WebCore](http://es.wikipedia.org/wiki/WebCore), el motor de [Konqueror](http://es.wikipedia.org/wiki/Konqueror). Antecesor del [WebKit](http://es.wikipedia.org/wiki/WebKit).

·  [Presto](http://es.wikipedia.org/wiki/Presto_(motor_de_renderizado)), el antiguo motor de Opera.

·  [Tasman](http://es.wikipedia.org/wiki/Tasman), el motor de Internet Explorer para Mac.

·  [gzilla](http://es.wikipedia.org/wiki/Gzilla), el motor de [Dillo](http://es.wikipedia.org/wiki/Dillo).

·  [GtkHTML](http://es.wikipedia.org/w/index.php?title=GtkHTML&action=edit&redlink=1), el motor de Links.

·  [WebKit](http://es.wikipedia.org/wiki/WebKit), el motor de [Epiphany](http://es.wikipedia.org/wiki/Epiphany), Safari.

·  [Blink](http://es.wikipedia.org/wiki/Blink), el nuevo motor de Google Chrome y [Opera](http://es.wikipedia.org/wiki/Opera_(navegador)) (se trata de un [fork](http://es.wikipedia.org/wiki/Fork) de WebKit).

·  [Servo](http://es.wikipedia.org/wiki/Servo_(motor_de_renderizado)), nuevo motor en desarrollo por parte de Mozilla (con el apoyo de Samsung), está siendo optimizado para la arquitectura ARM y la plataforma Android.

El término motor de renderizado también puede referirse a motores de renderizado de texto como Pango o [Uniscribe](http://es.wikipedia.org/w/index.php?title=Uniscribe&action=edit&redlink=1) los cuales hacen presentables a los textos plurilingües, teniendo en cuenta los textos bidireccionales, combinaciones de "caracteres básicos" con acentos, y otras complicaciones del texto plurilingüe.

v V8: Es un motor de código abierto para JavaScript creado por Google. Está escrito en C++ y es usado en Google Chrome. V8 puede funcionar de manera individual (standalone) o incorporada a cualquier aplicación C++.

v Client-side: Cualquier lenguaje que se ejecuta en un dispositivo cliente que interactúa con un servicio remoto es un lenguaje del lado del cliente.

v Server-side: es el nombre general para los tipos de programas que se ejecutan en el servidor.

v Scripting Language: es un lenguaje de programación que soporta las secuencias de comandos, los programas escritos para un entorno especial de tiempo de ejecución que puede interpretar (en lugar de compilar) y automatizar la ejecución de tareas que, alternativamente podría ser ejecutados de uno en uno por un operador humano.

v List ECMAScript engine: es el lenguaje de programación estandarizado por Ecma Internacional en la especificación ECMA-262 e ISO / IEC 16262. El lenguaje se usa ampliamente para client-side scripting en la web, en la forma de varias implementaciones conocidas como JavaScript, JScript y ActionScript.

Estos son los motores nuevos de ECMAScript generación de navegadores web, todo implementación justo a tiempo de compilación (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.

Carakan: Un motor de JavaScript desarrollado por Opera Software ASA, incluye en la Comunicado del navegador web Opera 10.50, hasta el cambio a V8 con Opera 15 (lanzado en 2013).

Chakra: Un motor de JScript utilizarse en Internet Explorer. Se fue visto de antemano primero en MIX 10 como parte de Internet Explorer 9 Platform Preview.

SpiderMonkey: Un motor de JavaScript en aplicaciones de Mozilla Gecko, incluyendo Firefox. El motor incluye actualmente los compiladores IonMonkey y JägerMonkey, ha incluido previamente el compilador TraceMonkey (javascript JIT), y está previsto incluir el próximo compilador OdinMonkey.

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, Node.js y V8.NET.

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).

Los siguientes motores utilizan intérpretes de tiempo de ejecución, que no se compilan en código máquina nativo y por lo general se ejecutan más lentamente.

Continuum: Un auto-intérprete que soporta las últimas versiones del proyecto de ECMAScript 6 especificación Únicamente, el motor se implementa en ECMAScript 3, lo que hace posible ejecutar ES6 en los navegadores tan antiguos como IE6

Futhark: El motor ECMAScript de las versiones del navegador web Opera 9,50 a 10,10.

Inscript: Una biblioteca propietaria obsoleto utilizado para iCab 2 y 3

JScript: El motor que se utiliza en Internet Explorer para versiones hasta IE9, y uno de los componentes del motor de renderizado Trident.

RV: El motor utilizado en Konqueror, y uno de los componentes de KHTML, un predecesor de JavaScriptCore.

Lineal B: El motor ECMAScript de las versiones del navegador web Opera 7,0-9,50, exclusivo.

Narciso: JavaScript implementado en JavaScript (un evaluador meta-circular), destinado para funcionar en otro motor de JavaScript, de carácter teórico y educativo solamente.

JS-Intérprete un peso ligero intérprete de JavaScript implementado en JavaScript con la ejecución paso a paso.

QtScript: Originalmente desarrollado por Trolltech, ahora propiedad de Digia. Proporciona integración QObject con JavaScriptCore.

Rhino: Uno de los varios motores de JavaScript de Mozilla, utilizando la plataforma Java.

YAJI:. Un motor ECMAScript sobre la base de la FESI aplicación por Jean-Marc Lugrin en 1999, el uso de la plataforma Java, actualmente en desarrollo para apoyar a los últimos estándares (Spec 262, v5.1)

Duktape: una pequeña huella, motor fácilmente integrable ECMAScript E5 / E5.1.

La Plataforma Kinoma, un entorno de ejecución de ECMAScript 5 y el marco

Jsish: un intérprete de Javascript con sqlite orden interna, json, WebSocket y zvfs apoyo.

Websocket.js: motor de Javascript integrable con HTTP / WebSocket apoyo.

Espruino: una muy pequeña huella intérprete específicamente para microcontroladores. Puede funcionar en menos de 8kB de RAM mediante la ejecución de la fuente (en lugar de Bytecode)

MuJS: una biblioteca intérprete ECMAScript ligero, diseñado para encajar en otro software para extenderlos con capacidades de scripting. Originalmente desarrollado para MuPDF.

Summary:

El operador precedencia determina el orden en que el operador esta evaluado. El valor con gran altura de precedencia será evaluado de primero.

3 + 4 \* 5 // returns 23

Windows.onload:

El evento load dispara el evento al final del proceso de carga del documento. En este punto, todos los objetos del documento son DOM, y todas las imágenes y sub-frames han terminado de cargarse.

window.onload = funcRef;

FuncRef es la función tipo handler function a ser llamada cuando el evento load de window se dispara.

Onload:

El atributo onload incendios cuando un objeto se ha cargado.

onload es la más utilizada en el elemento <body> para ejecutar un script una vez a la página web se ha cargado completamente todo el contenido.

El atributo onload se puede utilizar para comprobar el tipo de navegador y versión del navegador del visitante, y cargar la versión correcta de la página web en base a la información.

<!DOCTYPE html>

<html>

<body>

<img src="w3html.gif" onload="loadImage()" width="100" height="132">

<script>

function loadImage() {

alert("Image is loaded");

}

</script>

</body>

</html>

Onunload:

El onunload atributo incendios una vez al perfil ha descargado (o la ventana del navegador se ha cerrado ).

onunload se produce cuando el usuario se desplaza fuera de la página (haciendo clic en un enlace, envía un formulario, cierre la ventana del navegador , etc. ).

<!DOCTYPE html>

<html>

<body onunload="myFunction()">

<h1>Welcome to my Home Page</h1>

<p>Close this window or press F5 to reload the page.</p>

<p><strong>Note:</strong> Due to different browser settings, this event may not always work as expected.</p>

<script>

function myFunction() {

alert("Thank you for visiting W3Schools!");

}

</script>

</body>

</html>

TypeOf:

El operador typeof devuelve una cadena que indica el tipo del operando sin evaluarlo. Operando es la cadena, variable, palabra clave u objeto para el que se devolverá su tipo. Los paréntesis son opcionales.

El operador typeof devuelve los siguientes resultados operator returns the following results for these variables:

typeof miFuncion == 'function'

typeof forma == 'string'

typeof tamano == 'number'

typeof hoy == 'object'

typeof noExiste == 'undefined'

typeof true == 'boolean'

typeof null == 'object'

Null:

El valor nulo es un valor literal JavaScript en representación nula o un "vacío", es decir, sin valor de objeto está presente. Es uno de los valores primitivos de JavaScript.

### Difference between null and undefined

typeof null

typeof undefined

null === undefined

null == undefined

Undefined:

La propiedad de valor global undefined representa el valor undefined . Es uno de los tipos primitivos de JavaScript.

Logica de operadores

La lógica de operadores son típicamente usadas con el valor de boolean.

&& à Cuando se usa con un valor boolean && retorna true si ambos o perandos son verdaderos; de lo contrario, devuelve false.

|| à Cuando se usa con un valor boolean && retorna true si cualquiera de los operandos son verdaderos; si ambos son falsos, devuelve false.

! à Retorna false si su único operando es verdadero, de lo contrario, devolver false

Delete:

El operador delete elimina una propiedad de un objeto.

Si el operador delete tiene éxito, se elimina la propiedad del objeto por completo. Sin embargo, si existe una propiedad con el mismo nombre en la cadena de prototipo del objeto, el objeto heredará esa propiedad del prototipo.

### Funciones parseInt and parseFloat

Las dos funciones "convertidoras", [parseInt](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Obsolete_Pages/Gu%C3%ADa_JavaScript_1.5/es/Referencia_de_JavaScript_1.5/Funciones_globales/parseInt) y [parseFloat](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Obsolete_Pages/Gu%C3%ADa_JavaScript_1.5/es/Referencia_de_JavaScript_1.5/Funciones_globales/parseFloat), retornan un valor numérico cuando se pasa una cadena como un argumento.

ParseFloat:

Convierte (parsea) un argumento de tipo cadena y devuelve un número de punto flotante.

parseFloat es una función de alto nivel y no está asociada a ningún objeto.

parseFloat convierte su argumento, una cadena, y devuelve un número de punto flotante. Si encuentra un carácter diferente al signo (+ o -), numerales (0-9), un punto decimal o un exponente, devuelve el valor hasta ese punto e ignora ese carácter y todos los correctos siguientes. Se permiten espacios anteriores y posteriores.

Si el primer carácter no se puede convertir a número, parseFloat devuelve NaN.

Ejemplo:

parseFloat("3.14");

parseFloat("314e-2");

parseFloat("0.0314E+2");

var cadena = "3.14"; parseFloat(cadena);

parseFloat("3.14más caracteres no dígitos");

ParseInt:

ParseInt convierte su primer argumento, la cadena cadena, e intenta retornar un entero en una base de raíz especificada, indicada por el segundo parámetro opcional, radix. Por ejemplo, una raíz de 10 indica convertir a número decimal, ocho a octal, dieciseis a hexadecimal y así sucesivamente. Para raíces mayores a diez, las letras del alfabeto indican números mayores a nueve. Por ejemplo, para números hexadecimales (base 16), son utilizadas desde la A hasta la F.

Si parseInt encuentra un caracter que no es un número en la raíz especificada, lo ignora y todos los caracteres sucesivos y retorna el valor entero convertido hasta el punto. Si el primer caracter no puede ser convertido a un número en una raíz especificada, retorna "NaN." La función parseInt trunca la cadena a valores enteros.

Ejemplo:

Código:

var a = parseInt("10") + "<br>";

var b = parseInt("10.00") + "<br>";

var c = parseInt("10.33") + "<br>";

var d = parseInt("34 45 66") + "<br>";

var e = parseInt(" 60 ") + "<br>";

var f = parseInt("40 years") + "<br>";

var g = parseInt("He was 40") + "<br>";

var h = parseInt("10",10)+ "<br>";

var i = parseInt("010")+ "<br>";

var j = parseInt("10",8)+ "<br>";

var k = parseInt("0x10")+ "<br>";

var l = parseInt("10",16)+ "<br>";

var n = a + b + c + d + e + f + g + "<br>" + h + i + j + k +l;

en el browser:

10

10

10

34

60

40

NaN

10

10

8

16

16

Timing

Es posible ejecutar algún código en intervalos de tiempo especificados. Esto se llama eventos de tiempo .  
  
Es muy fácil para los acontecimientos del tiempo en JavaScript . Los dos métodos principales que se utilizan son :  
  
 setInterval () - ejecuta una función , una y otra vez , a intervalos de tiempo especificados  
 setTimeout () - ejecuta una función , una vez , después de esperar un número especificado de milisegundos  
  
Nota : El setInterval () y setTimeout () son los dos métodos del objeto HTML DOM ventana.

var myVar=setInterval(function () {myTimer()}, 1000);

function myTimer() {

var d = new Date();

document.getElementById("demo").innerHTML = d.toLocaleTimeString();

}

JS Errors

La sentencia try le permite probar un bloque de código para los errores.  
  
La sentencia catch le permite manejar el error.  
  
La sentencia throw permite crear errores personalizados .  
  
La sentencia finally permite ejecutar código , después de tratar de atrapar , sin importar el resultado.

Cuando se ejecuta el código JavaScript , pueden ocurrir errores diferentes .

function myFunction() {

var message, x;

message = document.getElementById("message");

message.innerHTML = "";

x = document.getElementById("demo").value;

try {

x = Number(x);

if(x == "") throw "is empty";

if(isNaN(x)) throw "is not a number";

if(x > 10) throw "is too high";

if(x < 5) throw "is too low";

}

catch(err) {

message.innerHTML = "Error: " + err + ".";

}

finally {

document.getElementById("demo").value = "";

}

}

Patrón Constructor Fábrica

Este patrón es especial, ya que no utiliza "new".

The object is created by a simple function call, similar to *Python-style*:

|  |
| --- |
| var animal = Animal ("fox") |

var rabbit = Rabbit ("rab")

Declaración  
  
El constructor se define como una función que devuelve un nuevo objeto:

function Animal(name) {

return {

run: function() {

alert(name + " is running!")

}

}

}

herencia

Rabbit se hace mediante la creación de un animal y , a continuación, la mutación es :

function Rabbit(name) {

var rabbit = Animal(name) // make animal

rabbit.bounce = function() { // mutate

this.run()

alert(name + " bounces to the skies! :)")

}

return rabbit // return the result

}

var rabbit = Rabbit("rab")

rabbit.bounce()