Glosario JS Glab-Academy

Este glosario de palabras claves te ayudará a entender la funcionalidad de sentencias en **JavaScript.**

Nombre	Definición	Sintaxis
Break	sentencia break se utiliza para salir de las declaraciones sentencia switch o bucle.	**break [etiqueta];**
Class	La declaración class crea una nueva clase con el nombre proporcionado utilizando la herencia basada en prototipo	**class name [extends] {** **// Contenido de la clase** **}**
Const	Esta declaración crea una constante cuyo alcance puede ser global o local para el bloque en el que se declara.	**const varname1 = value1 [, varname2 = value2 [, varname3 = value3 [, ... [, varnameN = valueN]]]];**
Continue	Termina la ejecución de las sentencias de la iteración actual del bucle actual o la etiqueta y continua la ejecución del bucle con la próxima iteración.	**continue [etiqueta];**
Debugger	La sentencia debugger invoca cualquier funcionalidad de depuración disponible, tiene la misma función que un breakpoint. Si la funcionalidad de	**debugger;**

	depuración no está disponible, esta sentencia no tiene efecto alguno.	
do..While	La condición se evalúa después de ejecutar la sentencia, dando como resultado que la sentencia especificada se ejecute al menos una vez.	**do** **Sentencia** **while (condición);**
Empty	Un empty statement o sentencia vacía es usada para no proveer una sentencia, incluso si la sintaxis JavaScript esperase una.	**;**
Export	La declaración export se utiliza al crear módulos de JavaScript para exportar funciones, objetos o tipos de dato primitivos del módulo para que puedan ser utilizados por otros programas con la sentencia importante.	**export { name1, name2, …, nameN };** **export { variable1 as name1, variable2 as name2, …, nameN };** **export let name1 = …, name2 = …, …, nameN; // también var, const**
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.	**for ([expresion-inicial]; [condicion]; [expresion-final])sentencia**
For await of	La sentencia for await...of crea un bucle iterando tanto sobre objetos iterables asincrónicos como sincrónicos	**for await (variable of iterable) {** **sentencia** **}**
For…in	La instrucción for-in itera sobre todas las	

	propiedades enumerables de un objeto que está codificado por cadenas (ignorando los codificados por Símbolos, incluidas las propiedades enumerables heredadas.	**for (variable in objeto)** **instrucción**
For…for	La sentencia sentencia for...of ejecuta un bloque de código para cada elemento de un objeto iterable.	**for (variable of iterable) {** **statement** **}**
Function	Declara una función con los patrones especificados	**function nombre([parametro1] [,parametro2] [..., parametroN]) {sentencias}**
Function*	La declaración function* (la palabra clave function seguida de un asterisco) define una *función generadora*, que devuelve un objeto.	**function* nombre([param[, param[, ... param]]]) {** **instrucciones** **}**
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.	**if (condición) sentencia1 [else sentencia2]**
Import	La sentencia import se usa para importar funciones que han sido exportadas desde un módulo externo.	**import * as name from "module-name";** **mport "module-name";** **import { export1 , export2 } from "module-name";**
label	Proporciona una sentencia con un identificador al que se puede referir al usar las sentencias	**etiqueta :sentencia**

let	La instrucción let declara una variable de alcance local con ámbito de bloque(blockscope), la cual, opcionalmente, puede ser inicializada con algún valor.	**let var1 [= valor1] [, var2 [= valor2]] [, ..., varN [= valorN]];**
return	La sentencia return finaliza la ejecución de la función y especifica un valor para ser devuelto a quien llama a la función.	**return [[expresion]];**
Switch	La declaración switch evalúa una expresión, comparando el valor de esa expresión con una instancia case, y ejecuta declaraciones asociadas a ese case, así como las declaraciones en los case que siguen.	**switch (expresión) {** **case valor1:** **//Declaraciones ejecutadas cuando el resultado de expresión coincide con el valor1** **[break;]**
Throw	Lanza una excepcion definida por el usuario.	**throw expresion;**
Var	La sentencia var declara una variable, opcionalmente inicializándola con un valor.	**var nombreDeVariable1 [= valor1] [, nombreDeVariable2 [= valor2] ... [, nombreDeVariableN [=valorN]]];**

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.	**while (condicion)** **sentencia**