JavaScript CONVERSIONES AUTOMÁTICAS Y FORZADAS / COMPARACIONES DE IGUALDAD : *RESUMEN DE PROPIEDADES NOTABLES* (-> : promociona a)

↦	null	undefined	Boolean	Number	BigInt	String	Symbol	Object
null	`typeof null === "object"` `(null).propiedad` -> TypeError `(null)?.propiedad` -> undefined			`null` -> 0	-> TypeError			

undefined		`typeof undefined === 'undefined'`
		`(undefined).propiedad` -> TypeError
		`(undefined)?.propiedad`
		-> undefined
			`Boolean(undefined) === false`
			`Boolean(!undefined) === true`
				`undefined` -> NaN
				`undefined` --a entero 32bits-> 0
					-> TypeError			

Boolean			`typeof Boolean() === "boolean"`
			`Boolean(false) === false`
			`Boolean(true) === true`
			`new Boolean(false) == false`
			`new Boolean(true) == true`
			`new Boolean(false) !== true`
			`new Boolean(true) !== true`
			`if (new Boolean(false))` -> true
			`if (new Boolean(true))` -> true
				`true` -> 1
				`false` -> 0
				
				`+true` es 1
				`-true` es -1
				`+false` es 0
				`-false` es -0
					`true` -> 1n
					`false` -> 0n			

Number			Convertir entero en booleano:
			`!!(35)` -> true
			`!!(35.45)` -> true
			`!!(0)` -> false
			`!!(0.00)` -> false
			`!!(+0)` -> false
			`!!(-0)` -> false
			`!!(NaN)` -> false
			
			`Boolean(35)` -> true
			`Boolean(0)` -> false
			`Boolean(+0)` -> false
			`Boolean(-0)` -> false
			`Boolean(NaN)` -> false
				`typeof Number() === 'number'`
				`typeof (22343) === 'number'`
				`typeof (1.344) === 'number'`
				`Number + String === 'string'`
				`String + Number === 'string'`
				`String -/*% Number === 'number'`
				`Number -/*% String === 'number'`
				Forzar aritmética entera:
				`((x * 3 / 9)>>>0)` *positivos*
				`((x * -3 / 9)|0)` *si hay negativos*
				`+x` coerciona `x` a Number
				`(34).toExponential(3)`;
				comprobar si x es `NaN`: `x !== x`
				`~~(-0)` y `-0 << 2 >> 2`
				`* / +` con `-0`
				falsy `&&` `x` devuleve el falsy (no `false`)
				truthy `&&` `x` devuleve `x`
				`~x` equivale a `-x - 1`
					-> TypeError			

BigInt			*Igual que* Number	BigInt -> TypeError `Number(BigInt)` -> Number (con posible pérdida de precisión)	para entero grande pasarlo como string : `BigInt("-1234567890987654321")`			

String			`!!""` -> false
			`!""` -> true
			`""` -> false
			`[]` -> "" -> false
				`"sdfd"` -> NaN (error de análisis)
				`"Infinity"` -> Infinity
				`"-34.34"` -> -34.34
				`""` -> 0
				
				`+'1.1'` devuelve 1.1
				`Number.parseFloat(string)`
				`Number.parseInt(string)`
					`"34434342"` -> 34434342n			

Symbol				-> TypeError	-> TypeError			

Object			-> true	1º) `[@@toPrimitive]('number')`
				2º) `valueOf()`
				3º) `toString()`
				4º) primitivo resultante a Number
					1º) `[@@toPrimitive]('number')`
					2º) `valueOf()`
					3º) `toString()`
					4º) primitivo resultante a BigInt
						`({}).toString() === '[object Object]'`
						`([]).toString() === ''`
						`[]` -> ''		

		Tipos primitivos				Objetos	
		true	false	Boolean(true)	Boolean(false)	new Boolean(true)	new Boolean(false)
==	true	true	false	true	false	true	false
	false		true	false	true	false	true
	Boolean(true)			true	false	true	false
	Boolean(false)				true	false	true
	new Boolean(true)					false	false
	new Boolean(false)						false
===	true	true	false	true	false	false	false
	false		true	false	true	false	false
	Boolean(true)			true	false	false	false
	Boolean(false)				true	false	false
	new Boolean(true)					false	false
	new Boolean(false)						false

	==		===	
	true	false	true	false
"" *(cadena vacía)*	false	true	false	false
"no vacía"	false	false	false	false
0	false	true	false	false
+0	false	true	false	false
-0	false	true	false	false
+0.0	false	true	false	false
-0.0	false	true	false	false
+Infinity	false	false	false	false
-Infinity	false	false	false	false
NaN	false	false	false	false

Valor original	--> Number	-->String	-->Boolean
false	0	"false"	false
true	1	"true"	true
0	0	"0"	false
1	1	"1"	true
"0"	0	"0"	true
"000"	0	"000"	true
"1"	1	"1"	true
NaN	NaN	"NaN"	false
Infinity -Infinity	Infinity -Infinity	"Infinity" "-Infinity"	true
""	0	""	false
"20"	20	"20"	true
"veinte"	NaN	"veinte"	true
[ ]	0	""	true
[20]	20	"20"	true
[10,20]	NaN	"10,20"	true
["veinte"] ["diez","veinte"]	NaN NaN	"veinte" "diez,veinte"	true
function(){}	NaN	"function(){}"	true
{ }	NaN	"[object Object]"	true
null	0	"null"	false
undefined	NaN	"undefined"	false

X	y	==	===	Objeto.is	SameValueZero
undefined	undefined	true	true	true	true
null	null	true	true	true	true
true	true	true	true	true	true
false	false	true	true	true	true
'foo'	'foo'	true	true	true	true
0	0	true	true	true	true
+0	-0	true	true	false	true
+0	0	true	true	true	true
-0	0	true	true	false	true
0n	-0n	true	true	true	true
0	false	true	false	false	false
""	false	true	false	false	false
""	0	true	false	false	false
'0'	0	true	false	false	false
'17'	17	true	false	false	false
[ ]	false	true	false	false	false
[1, 2]	'1,2'	true	false	false	false
[1, 2]	'1, 2'	false	false	false	false
new String('foo')	'foo'	true	false	false	false
null	undefined	true	false	false	false
null	false	false	false	false	false
undefined	false	false	false	false	false
{ foo: 'barra' }	{ foo: 'barra' }	false	false	false	false
new String('foo')	new String('foo')	false	false	false	false
0	null	false	false	false	false
0	NaN	false	false	false	false
'foo'	NaN	false	false	false	false
NaN	NaN	false	false	true	true

VALORES falsy (cualquier otro valor será truthy)

Valor	Descripción
false	La palabra clave false .
0	El Número cero (entonces, también 0.0 , etc., y 0x0).
-0	El Número menos cero (entonces, también -0.0 , etc., y -0x0).
0n	El BigInt cero (entonces, también 0x0n). Tenga en cuenta que no hay un cero negativo de BigInt : la negación de 0n es 0n .
"" '' ``	Valor de cadena vacío.
null	null — la ausencia de referencia a objeto.
undefined	**undefined** — el valor primitivo.
NaN	NaN: no es un número.
document.all	Los objetos son falsos si y solo si tienen la ranura interna **[[IsHTMLDDA]]**. Dicha ranura solo existe en document.all y no se puede configurar mediante JavaScript.

```
consola.log( false && "perro" );
// devuleve:  false
consola.log( 0 && "perro" );
// devuelve: 0
```

```
true && "perro"
// devuelve "perro"
[] && "perro"
// devuelve "perro"
```