TIPOS DE DATOS

[**Tipo Boolean**](https://developer.mozilla.org/es/docs/Web/JavaScript/Data_structures#tipo_boolean)

Boolean representa una entidad lógica y puede tener dos valores: true y false. Consulta [Boolean](https://developer.mozilla.org/es/docs/Glossary/Boolean) y [Boolean](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Boolean) para obtener más detalles.

[**Tipo Null**](https://developer.mozilla.org/es/docs/Web/JavaScript/Data_structures#tipo_null)

El tipo Null tiene exactamente un valor: null. Consulta [null](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/null) y [Null](https://developer.mozilla.org/es/docs/Glossary/Null) para obtener más detalles.

[**Tipo Undefined**](https://developer.mozilla.org/es/docs/Web/JavaScript/Data_structures#tipo_undefined)

Una variable a la que no se le ha asignado un valor tiene el valor undefined. Consulta [undefined](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/undefined) y [Undefined](https://developer.mozilla.org/es/docs/Glossary/undefined) para obtener más detalles.

[**Tipo Number**](https://developer.mozilla.org/es/docs/Web/JavaScript/Data_structures#tipo_number)

ECMAScript tiene dos tipos numéricos integrados: **Number** y **BigInt** (ve más abajo).

El tipo Number es un [valor en formato binario de 64 bits de doble precisión IEEE 754](https://es.wikipedia.org/wiki/Formato_en_coma_flotante_de_doble_precisi%C3%B3n) (números entre -(253 - 1) y 253 - 1). Además de representar números de punto flotante, el tipo Number tiene tres valores simbólicos: +Infinity, -Infinity y [NaN](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/NaN) ("**N**ot a **N**umber" o No es un número).

El tipo BigInt es un primitivo numérico en JavaScript que puede representar números enteros con precisión arbitraria. Con BigInts, puedes almacenar y operar de forma segura en números enteros grandes incluso más allá del límite seguro de enteros para Numbers.

Un BigInt se crea agregando n al final de un número entero o llamando al constructor.

Puedes obtener el valor más seguro que se puede incrementar con Numbers utilizando la constante [Number.MAX\_SAFE\_INTEGER](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Number/MAX_SAFE_INTEGER). Con la introducción de BigInts, puedes operar con números más allá de [Number.MAX\_SAFE\_INTEGER](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Number/MAX_SAFE_INTEGER).

Este ejemplo demuestra, dónde, incrementando el [Number.MAX\_SAFE\_INTEGER](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Number/MAX_SAFE_INTEGER) devuelve el resultado esperado:

> const x = 2n \*\* 53n;

9007199254740992n

> const y = x + 1n;

9007199254740993n

**let** te permite declarar variables limitando su alcance (scope) al bloque, declaración, o expresión donde se está **usando**. a diferencia de la palabra clave **var** la cual define una variable global o local en una función sin importar el ámbito del bloque.

QUE ES VAR EN JAVASCRIT

El ámbito de una variable declarada con la palabra reservada **var** es su contexto de ejecución en curso, que puede ser la función que la contiene o, para las variables declaradas afuera de cualquier función, un ámbito global. Si re-declaras una variable **Javascript**, esta no perderá su valor.

QUE ES CONST EN JAVASCRIT

 Esta declaración crea una constante cuyo alcance puede ser global o local para el bloque en el que se declara. Es necesario inicializar la constante, es decir, se debe especificar su valor en la misma sentencia en la que se declara, lo que tiene sentido, dado que no se puede cambiar posteriormente.

QUE ES LET EN JAVASCRIT

**let**te permite declarar variables limitando su alcance (scope) al bloque, declaración, o expresión donde se está usando.a diferencia de la palabra clave var la cual define una variable global o local en una función sin importar el ámbito del bloque. La otra diferencia entre var y let es que este último se inicializa a un valor sólo cuando un analizador lo evalúa (ver abajo).