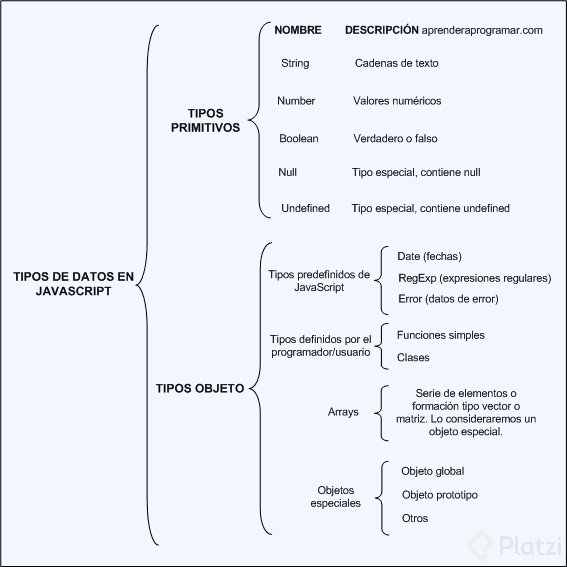
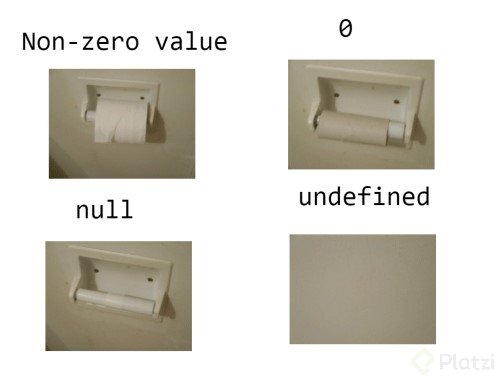
**APUNTES**  
1.- JavaScript tiene una **comunidad enorme** de desarrolladores que te pueden ir ayudando a generar diferentes cosas.

1. Si solo estuvieras interesado en trabajar **aplicaciones web** tienes muchos frameworks y librerías construidas en JavaScript que te van a ayudar a hacer proyectos de forma mucho mas rápida, eficiente y robusta (Angular, View, React,entre otros)
2. Si no quieres trabajar solo en aplicaciones Web puedes utilizar JavaScript con un framework que se llama React Native para poder **construir aplicaciones nativas** como Android y IOS.
3. Puedes construir **aplicaciones de escritorio** con JavaScript, usando un framework llamado Electron, pueden correr en Mac o Windows.
4. También puedes trabajar en la parte del **Back-end** o \*\*IOT \*\*(Internet Od Things) es un concepto que se refiere a una interconexion digital de objetos cotidianos con Internet. Esto con un Framework llamado NodeJS, el cual es un entorno de ejecución de JavaScript que corre directamente en el Back-end.

* WebAssembly: es un nuevo tipo de código que puede ser ejecutado en navegadores modernos es un lenguaje de bajo nivel, similar al lenguaje ensamblador, con un formato binario compacto que se ejecuta con rendimiento casi nativo y provee un objetivo de compilación para lenguajes como C/C++ y Rust que les permite correr en la web. También está diseñado para correr a la par de JavaScript, permitiendo que ambos trabajen juntos.



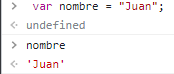


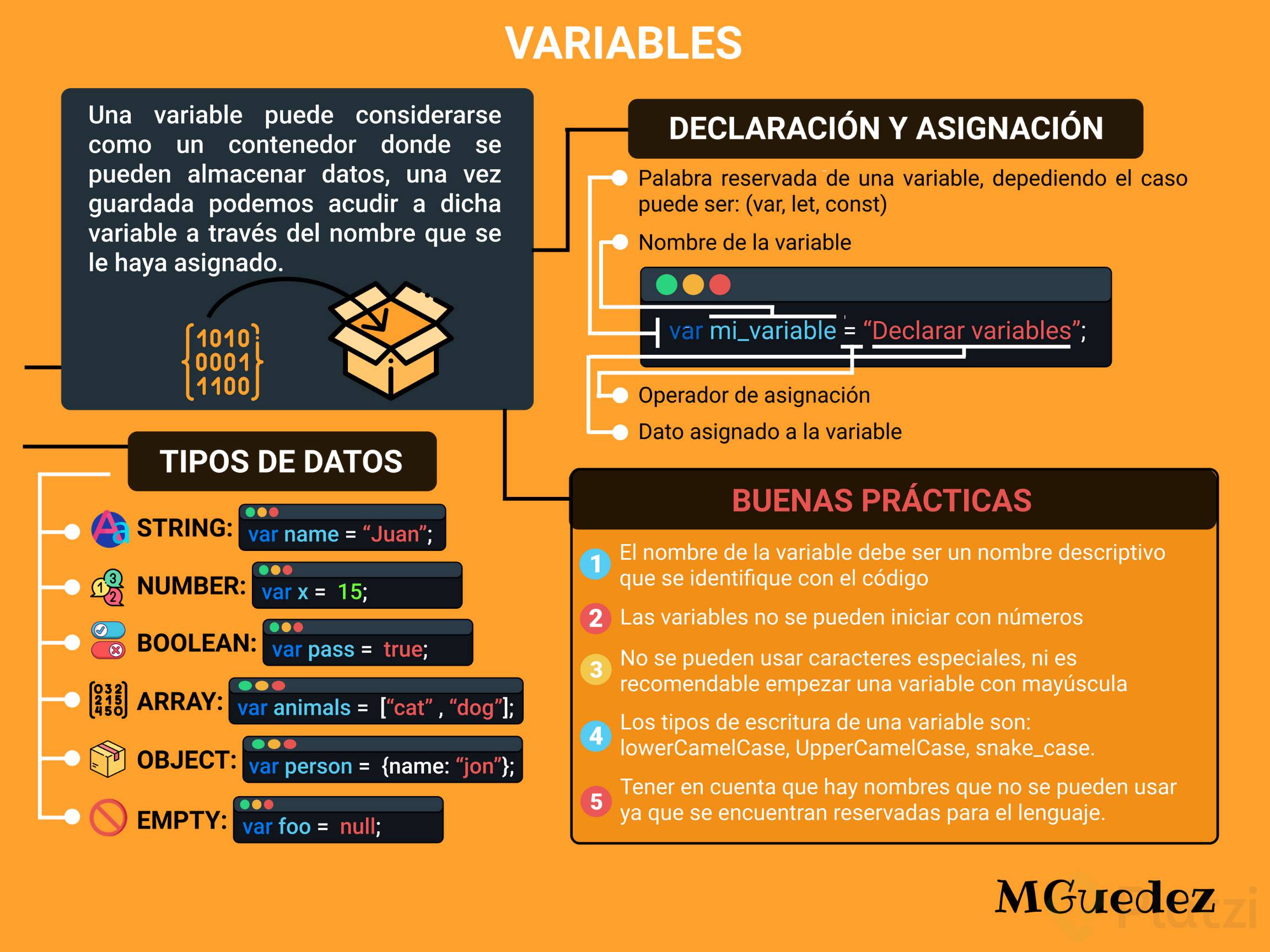


Typeof es una palabra reservada para saber el tipo de variable de algo



Variables





Dentro de JavaScript tenemos tres formas de declarar una variable las cuales son: **var**, **const** y **let.**

1. **Var:** Era la forma en que se declaraban las variables hasta ECMAScript 5. Casi ya no se usa porque es de forma global y tiene las siguientes características:

**o Se puede reinicializar:** osea todas las variables se inicializan, por ejemplo:  
Var pokemonType = ‘electric’ entonces reinicializar es:  
Var pokemonType = ‘grass’ osea la misma variable con diferentes datos el último dato predomina.  
**o Se puede reasignar:** osea la variable ya inicializada le reasignamos otro valor por ejemplo: inicializamos la variable: Var pokemonType = ‘electric’ ahora la reasignamos pokemonType = ‘grass’ ya no va var  
**o Su alcance es función global:** osea inicializamos la variable, pero la podemos llamar desde cualquier bloque (una llave abierta y una cerrada {}) pero hay que tener mucho cuidado con ello ya que puede haber peligro, no es recomendable usar VAR.

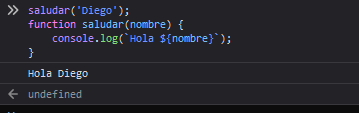
**const y let es la forma en que se declaran las variables a partir de ECMAScript 6,**

1. **const:** sirve para declarar variables que nunca van a ser modificadas:  
   **o No se puede reinicilizar:** es una const única no puede haber otra inicializada con el mismo nombre. const pokemonType = ‘electric’ no puede haber:  
   const pokemonType = ‘grass’  
   o **No se pude re asignar:** una vez que la hayamos inicializado no la podemos reasignar solo con su nombre: const pokemonType = ‘electric’ no puede ejecutarse:  
   pokemonType = ‘grass’  
   o **No es inmutable:** osea no puede cambiar con objetos:
2. **Let:** Son variables que pueden ser modificadas, se pueden cambiar:  
   **o No se puede reinicilizar:** es una const única no puede haber otra inicializada con el mismo nombre. let pokemonType = ‘electric’ no puede haber:  
   let pokemonType = ‘grass’  
   o **Se puede reasignar:** Osea la variable ya inicializada le reasignamos otro valor por ejemplo: inicializamos la variable: let pokemonType = ‘electric’ ahora la reasignamos pokemonType = ‘grass’  
   o **Su contexto de es bloque:** Solo funciona dentro de un bloque {}, fuera de ello no.

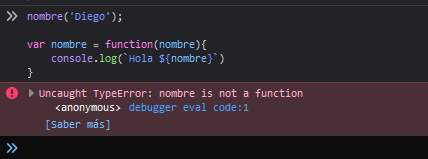


Cuando se refiere a que el **hoisting** afecta a una y a la otra no:

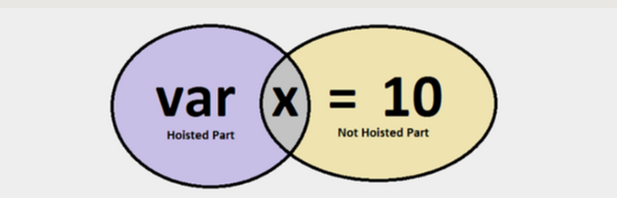
1. si intentamos usar primero una **función declarativa** antes de ser declarada:



si intentamos usar primero una **expresión de función** antes de ser declarada:

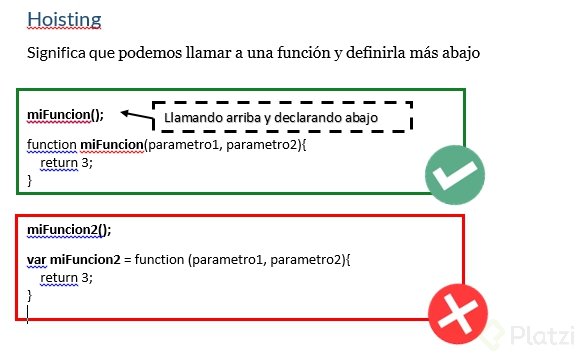


Y esto básicamente se debe a que:



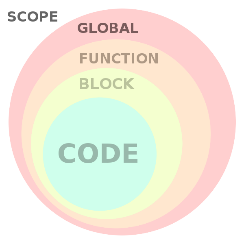
**hoisting** detecta primero **declaraciones** de **variables** y **funciones** para poder ser utilizadas, pero no su **definición (valor)**.

Y la **expresión de función** al ser una función **declarada** dentro de una variable, detecta solamente **var nombre**; (sin valor) por lo tanto la función **NO** es detectada en un principio.



Scope: Es el alcance que tienen las variables

**Scope Global** : todo el mundo.  
**Scope Local** : Las Vegas, y lo que pasa en las vegas, se queda en Las Vegas.



Lo que esta en local puede acceder a lo que esta en global, pero lo que esta en global no va a poder acceder a lo que se tenga en local.

