

**var, let, const**



A QUÉ BLOQUES DE CÓDIGO VA A PODER ACCEDER UNA VARIABLE



DEV.F

# El Scope de las variables en JavaScript

Antes de empezar a analizar las nuevas maneras de declarar variables (**let** y **const**) es necesario comprender el ámbito de las mismas en JavaScript.



## Tipos de variables

1. **Variables Locales:** una variable local es aquella que está declarada dentro de un bloque de código, es decir (una función, un ciclo o condicionales) se caracteriza por que se declara usando la palabra reservada **var** o **let**.
2. **Variables Globales:** una variable global es toda aquella que está definida fuera de todo bloque de código.

## ¿Qué sucede si declaramos una variable sin var, let o const?

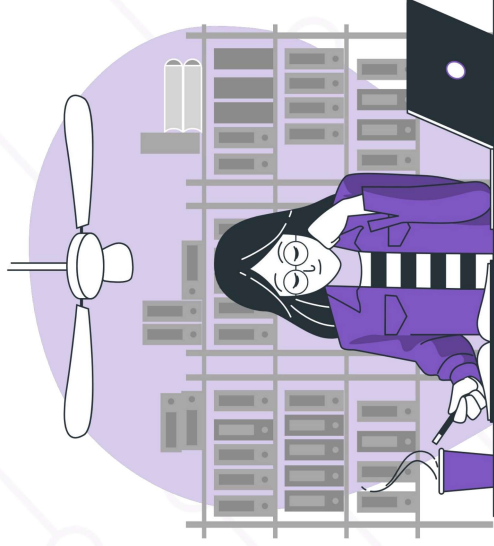
Si declaramos una variable dentro o fuera de un bloque de código sin usar ninguna palabra reservada, esta automáticamente se transforma en una variable global.

```
> function saludar(){  
  nombre = "Gabriela";  
}  
saludar()  
console.log(nombre);  
Gabriela
```



DEV.F

En un comienzo cuando se utilizaba javascript la manera de declarar una variable era usando **var**



con var podemos **re inicializar** una variable

```
nombre;  
console.log(nombre)  
var nombre = "juanito";  
juanito
```

con var podemos **re asignar** una variable

```
var edad = 12;  
edad = 13;
```

Recuerda muy bien qué **var** tiene **alcance** de función  
(**global o local**)

# var, let, const

## SON LAS MANERAS DE PODER INICIAR UNA VARIABLE

¿Qué es una variable?

Es una pequeña porción de memoria donde guardamos cierta información.

```
var ataque = 100;  
let pokemonType= 'electrico';  
const pokemonName = 'pikachu';
```



DEV.F

```
function Combate( ) {  
  var ataque = 16;  
  if ( ataque > 15) {  
    return "GANASTE LA BATALLA POKEMON"  
  }else if (ataque < 15 ) {  
    return "PERDISTE LA BATALLA POKEMON"  
  }  
}  
Combate()
```



Ningún pokemón fue dañado  
durante la explicación :)



```
var ataque = 60;  
var pokemon = 'PIKACHU';  
  
function Combate() {  
  if ( ataque > 15 ) {  
    return pokemon + " GANASTE LA BATALLA POKEMON"  
  }else if (ataque < 15 ) {  
    return pokemon + " PERDISTE LA BATALLA POKEMON"  
  }  
}  
Combate()
```





# ¿Qué es let? 🤔

let y const existen desde ECMAS6 y surgieron por el año 2015/16

let es una nueva forma de declarar variables, permite al programador proteger el valor de una variable **dentro del bloque donde se la asigna.**

```
function Multiplicar(numero){  
  for( let index=1; index<=10; index++){  
    console.log(numero + " * " + index + " = " +  
      (numero*index))  
  }  
}  
Multiplicar(5)
```



DEV.E

# Diferencias entre var y let

var

```
var edad = 25;

if(edad >= 18){
  var mensaje = "Es adulto"
}

console.log(mensaje);
//salida:
//"Es adulto"
```

let

```
let edad = 25;

if(edad >= 18){
  let mensaje = "Es adulto"
}

console.log(mensaje);
//salida:
//Uncaught ReferenceError: mensaje is not defined
```

DEV.E

¿Qué sucedió acá? 😞

¿Aparentemente hicimos lo mismo no?  
Solo cambiamos **let** por **var**. Aquí radica  
la principal diferencia entre ambas  
palabras reservadas y la explicación es  
bastante sencilla



DEV.F

# Diferencias entre var y let

var

```
var edad = 25;

if(edad >= 18){
  var mensaje = "Es adulto"
}

console.log(mensaje);
//salida:
//"Es adulto"
```

Declaramos una variable **edad** con **var** y luego verificamos que **edad** sea mayor a 18, de ser así imprimimos Es adulto FUERA DEL BLOQUE IF.

let

```
let edad = 25;

if(edad >= 18){
  let mensaje = "Es adulto"
}

console.log(mensaje);
//salida:
//Uncaught ReferenceError: mensaje is not defined
```

**let** encapsula a la variable dentro de un bloque (en este caso un if) por ende, al pretender usar la variable fuera de ese bloque salta el error **mensaje is not defined**, evitando así que las mismas se sobre escriban en un futuro.

DEV.E

## ¿Qué es const?

`const` permite declarar constantes, ósea, espacios en memoria donde sus valores no cambian en el tiempo.

```
const PI = Math.PI  
console.log(PI)  
3.141592653589793
```

*Por convenciones se acostumbra declarar las constantes con mayúsculas*

Una **constante** no permite que su valor se pueda alterar una vez declarada, el siguiente ejemplo devolvería un error:

```
const fruta = "Manzana";  
let fruta = "Zandia";  
console.log(fruta);  
//salida: Uncaught SyntaxError: redeclaration of const fruta
```

## Conclusiones

- Las variables locales deben ser declaradas con las palabras reservadas **var**, **let** o **const**, de lo contrario el intérprete de JavaScript las convierte automáticamente en un variable global.
- **let** encapsula una variable dentro de un bloque, porque no podrá ser usada fuera del mismo.
- **let** evita reescribir el valor de una variable, cosa que **var** no hace.
- **var** y **let** comparten el mismo scope.
- Usa **let** o **const** de forma correcta para tener un mejor código y no ocupar mayor espacio de memoria, **var** es una forma anticuada de declarar variables pero aún se siguen utilizando.