

Variáveis Javascript

É possível declarar variáveis de diferentes formas, utilizando **var**, **let** e **const**

Ciclo de vida de uma variável

Declaração

Inicialização

Atribuição

- **Declaração:** O nome da variável é registrado no contexto de execução, também conhecido como escopo, da função
- **Inicialização:** A variável é inicializada com o valor undefined
- **Atribuição:** Um valor é atribuído para a variável

Ao utilizar var, a variável é declarada e inicializada no escopo da função, **não respeitando bloco e permitindo a redeclaração e reatribuição**

```
//VAR
var precoDoProduto = 29.72;

var nomeDoProfessor = "Eduardo Henrique";

var acessoLiberado = true;

var idadeDoEdu = 30;; // primeiro declare a variável atribua
console.log(idadeDoEdu); // com console.log sempre vai dar undefined

var idadeDoEdu; // redeclaração
idadeDoEdu = 50; // reatribuição

if (true) {
    var banana = 12;
}
console.log(banana); // com var é permitido
```

Se você declarar o **var** dentro de um IF você consegue alterar os valores fora do IF. Consegue redeclarar e reatribuir

Ao utilizar **let**, a variável é declarada no escopo da função mas só é inicializada posteriormente, **respeitando bloco e permitindo reatribuição mas não a redeclaração**

```
//LET

let precoDoCarro = 89000;
console.log(precoDoCarro);

precoDoCarro = 10000; // reatribuição permitida
let precoDoCarro; // redeclaração não permitida

console.clear();

if (true) {
  let abacaxi = 44;
  console.log(abacaxi) // com let dentro do bloco funciona
}
console.log(abacaxi); // com let não é permitido
```

Se você declarar o **let** dentro de um IF você NÃO consegue alterar os valores fora do IF, nem redeclarar. Você reatribui apenas dentro do if.

Ao utilizar `const`, a variável é declarada no escopo da função mas só é inicializada posteriormente, **respeitando bloco e não permitindo reatribuição nem redeclaração**

```
//CONST

const precoDaPrancha = 111111;
console.log(precoDaPrancha);

precoDaPrancha = 10000; // reatribuição não permitida

const precoDaPrancha; // redeclaração não permitida
```

Ao declarar uma variável sem `var`, `let` ou `const` **ela é criada no escopo global**

```
function func1(a) {
  var kiwi = arguments[0];
  console.log(kiwi);
  pera = 3456;
}
func1(344);

console.log(pera); //se não declarar pelo menos var, vai pra global
//acrescente var na variável pera e teste novamente em novo navegador
```

Nunca declare variáveis sem `VAR`, `LET` ou `CONST`.

Evite `VAR` sempre que puder.

Um identificador válido deve começar com [a-zA-Z_\$] seguido por [a-zA-Z0-9_\$]

REGEX – Começar com letra de a à z minúsculo, letra de A à Z maiúsculo, underline ou cifrão.

Seguidos de letra de a à z minúsculo, letra de A à Z maiúsculo, números de 0 à 9 , underline ou cifrão.

```
// nomes de identificadores
let codigo;
let Codigo;
let _codigo;
let $codigo;

let codigo123;
let Codigo123;
let _codigo123;
let $codigo123;

let 123codigo; // não aceitos
let @Codigo; // não aceitos
```

Operadores Javascript

Operadores aritméticos

+, -, *, / e %

```
// Aritméticos

umMaisUm = 1 + 1;
//Adição +

umMenosUm = 1 - 1;
//Subtração -

//Multiplicação *
tresVezesDois = 3 * 2;

//Divisão /
quatroDivididoPorDois = 4 / 2;

//Módulo %
seisModuloCinco = 6 % 5; //usado apenas em numeros
```

```
//concatenação de strings
var s1 = " Marcelo ";
var s2 = " ";
var s3 = " Martins ";

var s4 = s1 + s2 + s3;

console.log(s4);
```

Os operadores *, / e % estão no mesmo nível de prioridade e os operadores + e - estão em um nível mais baixo de precedência.

Operadores de atribuição

`+=`, `-=`, `*=`, `/=` e `%=`

```
// Operadores de atribuição
```

```
// Simples =
```

```
let valor = 1;
```

```
// valor = 1
```

```
// Incremental +=
```

```
valor += 2;
```

```
// valor = 3
```

```
// Decremental -=
```

```
valor -= 1;
```

```
// valor = 2
```

```
// Multiplicativa *=
```

```
valor *= 6;
```

```
// valor = 12
```

```
// Divisória /=
```

```
valor /= 3;
```

```
// valor = 4
```

```
// Modular %=
```

```
valor %= 3;
```

```
// valor = 1
```

Operador de incremento e decremento **`++`** e **`--`**

```
// Operadores de Incremento
```

```
let valor2 = 10;
```

```
// Incremento ++
```

```
valor2 ++;
```

```
// valor = 10 é mostrado pois o ++ é pós visualização
```

```
valor2 // agora mostra 11
```

```
// Decremento --
```

```
valor2 --;
```

```
// valor = 10 é mostrado pois o -- é pós visualização
```

```
valor2 // agora mostra 10
```

Operadores relacionais

==, !=, <, <=, >, >=

```
// Operadores Relacionais

var valor = 2;
var t = false ;

// Igualdade ==
t = ( valor == 2);      // t = true

// Diferença !=
t = ( valor != 2);      // t = false

// Menor <
t = ( valor < 2);       // t = false

// Menor ou igual <=
t = ( valor <= 2);      // t = true

// Maior >
t = ( valor > 1);       // t = true

// Maior ou igual >=
t = ( valor >= 1);      // t = true
```


Operador binários

|, &, ^, ~, <<, >> e >>>

OR, AND, XOR, NOT, SHIFT (deslocamento esquerda / direita / direita com mudança de sinal)

```
// | ("OU")
var a = Math.random();
var b = Math.random();
console.log(a);
console.log(b);

alert (a > 0.2 || b < 0.8) ;
```

a > 0.2	b < 0.8	a > 0.2 b < 0.8
V	V	V
V	F	V
F	V	V
F	F	F

```
// && ("AND")
var c = Math.random();
var d = Math.random();
console.log(c);
console.log(d);

alert (c > 0.2 && d < 0.8) ;
```

a > 0.2	b < 0.8	a > 0.2 && b < 0.8
V	V	V
V	F	F
F	V	F
F	F	F

```
// ^ ("XOR")  
var nb = 5^9 // = 12  
console.log(nb);
```

```
// ~ ("NOT")  
var nb = ~2 // = -3  
console.log(nb);
```