DigitalHouse>
Coding School

Certified Tech Developer

The Ultimate Degree

Cheatsheet JavaScript

Variáveis

var

Declara uma variável global ou no escopo de uma **função.**

```
var varGlobal= 'valor inicial';

function name() {
  // escopo da função
  console.log(varGlobal); // -> 'valor inicial'

  if (condicao) {
    // escopo do bloco
    var varFuncao= 'Estou declarada em uma função';
    console.log(varGlobal); // -> 'valor inicial'

    varFuncao = 'Contínuo dentro da Função';
  }

  console.log(varFuncao); // -> 'Eu continuo dentro da Função' }

console.log(varGlobal); // -> 'outro valor'
console.log(varFuncao); // -> erro: não está declarada
```

let

Declara uma variável dentro do escopo de um **bloco**.

```
let varGlobal = 'valor inicial';

function name() {
  // escopo da função
  console.log(varGlobal); // -> 'valor inicial'

  if (condicao) {
    // escopo de um bloco
    let varFuncao = 'Estou declarada em uma função?';
    console.log(varGlobal); // -> 'valor inicial'
    varGlobal = 'outro valor';
  }

  console.log(varFuncao); // -> error não está declarada
  // let faz que a variável esteja disponível somente dentro do if }

console.log(varGlobal); // -> 'outro Valor'
console.log(varFuncao); // -> error não está declarada
```

const

Declara uma **constante** no escopo de um bloco.

```
const varGlobal = 'valor inicial';

function name() {
  // escopo da função
  console.log(varGlobal); // -> 'valor inicial'
```

```
  if (condicao) {
    // escopo do bloco
    const varFuncao = 'Estou declarada em uma função?';
    console.log(varGlobal); // -> 'valor inicial'

    varGlobal = 'otro Valor'; // erro a const não pode ser modificada }

  console.log(varFuncao); // -> erro no está declarada
  // const se comporta igual ao let em termos de alcance
}

console.log(varGlobal); // -> 'valor inicial'
console.log(varFuncao); // -> error no está declarada
```

Tipos

```
let myVariable = 'Hello wold'; // é uma string
let myVariable1 = 22; // é number
let myVariable2 = false; // é boolean
let myVariable3; // É undefined
let myVariable4 = { nome: 'meu nome" } ; // É um objeto
let myVariable5 = null; // É um objeto (É um tipo de objeto especial)
let myVariable6 = function() { let doSomething; }; // é uma function

// Você pode verificar esses tipos usando typeOf typeOf myVariable // -> number
```

Estruturas de controle

If

Permite executar um bloco somente se uma determinada condição for atendida.

```
if (condicao) {
   // se verdadeiro, executa este bloco de código
   }
```

If... else

Permite avaliar uma condição e executar um bloco de código ou outro.

```
if (condition) {
   //se verdadeiro, executa este bloco de código
} else {
  //se falso, executa este bloco de código
}
```

Switch

Permite executar diferentes ações dependendo do valor de uma variável.

```
switch (variavel) {
   case 1:
      // code if variavel == 1;
   break;

   case 2:
```

```
        // code if variavel == 2;
    break;

    default:
        // Executar se nenhuma condição anterior for encontrada
    break;
}
```

Laços

For

Permite que você execute repetidamente um bloco de código.

```
let n = 4;
for(var i = 0; i < n; i++) {
    // código para executar n vezes (4)
}
```

for(*inicial* ; *condição* ; *final* **) { }**

inicial: Este código é executado no início único do loop, normalmente uma variável é declarada conforme mostrado no exemplo.

condição: Sempre que o bloco de código fechado termina, esta condição é verificada, e se for verdadeira, o loop termina e o código abaixo continua.

final: Uma ação a ser executada a cada vez que o bloco a ser repetido é finalizado, a variável usada na condição é comumente modificada.

While

Seu comportamento é semelhante a um loop 'for', mas o bloco continuará executando indefinidamente enquanto a condição for verdadeira.

```
let n = 1;
while (n < 3) {
  // código que será executado
}
```

while(*condicao*) { }

condição: condição a ser avaliada antes de cada execução do bloco, se nunca se tornar falsa, o loop permanecerá em execução indefinidamente.

Do... while

Permite que você execute um bloco de código, desde que uma condição seja verdadeira. Ao contrário de while, a condição é avaliada no final de cada execução, que resulta na execução do bloco fechado pelo menos uma vez.

```
do {
  // código que será executado
} while (n < 3);
```