DigitalHouse>
Coding School

Certified Tech Developer

The Ultimate Degree

Cheatsheet JavaScript

Variables

var

Declara una variable global o en el ámbito de una **función**.

```
var varGlobal= 'valor inicial';

function name() {
  // ámbito de función
  console.log(varGlobal); // -> 'valor inicial'

  if (condicion) {
    // ámbito de bloque
    var varFuncion= 'Estoy declarada en una función';
    console.log(varGlobal); // -> 'valor inicial'

    varFuncion = 'sigo en la funcion';
  }

  console.log(varFuncion); // ->

'sigo en la funcion' }

console.log(varGlobal); // -> 'otro Valor'
console.log(varFuncion); // -> error no esta declarada
```

let

Declara una variable en el ámbito de un **bloque**.

```
let varGlobal= 'valor inicial';

function name() {
  // ámbito de función
  console.log(varGlobal); // -> 'valor inicial'

  if (condicion) {
    // ámbito de bloque
    let varFuncion= 'Estoy declarada en
    una función?';
    console.log(varGlobal); // ->
    'valor inicial'
    varGlobal = 'otro Valor';
  }

  console.log(varFuncion); // -> error no esta declarada
  // let hace que la variable solo este
disponible dentro de if }

console.log(varGlobal); // -> 'otro Valor'
console.log(varFuncion); // -> error no esta declarada
```

const

Declara una **constante** en el ámbito de un bloque.

```
const varGlobal= 'valor inicial';

function name() {
  // ámbito de función
  console.log(varGlobal); // -> 'valor inicial'

  if (condicion) {
    // ámbito de bloque
    const varFuncion= 'Estoy declarada en una función?';
    console.log(varGlobal); // -> 'valor inicial'

    varGlobal = 'otro Valor'; // error const no
  puede ser modificado }

  console.log(varFuncion); // -> error no esta declarada
  // const se comporta igual que let en cuanto alcance
}

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

Tipos

```
let myVariable = 'Hello wold'; // es un string
let myVariable1 = 22; // es un number
let myVariable2 = false; // es un boolean
let myVariable3; // es un undefined
let myVariable4 = { nombre: 'mi nombre' } ; // es un objeto let
myVariable5 = null; // es un objeto (es un tipo de objeto especial) let
myVariable6 = function() { let doSomething; }; // es una function

// Se pueden comprobar estos tipos mediante el uso de
typeOf typeOf myVariable // -> number
```

Estructuras de control

If

Permite ejecutar un bloque solo si se cumple una condición dada.

```
if (condicion) {
    // code...
}
```

If... else

Permite evaluar una condición y ejecutar un bloque de código u otro.

```
if (condicion) {
    // code if true
```

```
} else {
  // code if false
}
```

Switch

Permite ejecutar diferentes acciones dependiendo del valor de una variable.

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

    case 2:
        // code if variable == 2;
    break;

    default:
        // a ejecutar si no se cumple ninguna coincidencia anterior
    break;
}
```

Bucles

For

Permite ejecutar repetitivamente un bloque de código.

```
let n = 4;
for(var i = 0; i < n; i++) {
     // código a ejecutar n veces (4)
}
```

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

inicial: este código se ejecuta al iniciar el bucle por única vez, comúnmente se declara una variable como se muestra en el ejemplo.

condición: Por cada vez que termina el bloque de código encerrado, se comprueba esta condición, si se vuelve verdadera, el bucle finaliza y se continúa con el código debajo.

final: Una acción a ejecutar por cada vez que se termine el bloque a repetir, comúnmente se modifica la variable utilizada en la condición.

While

Su comportamiento es similar a un bucle 'for', pero el bloque seguirá ejecutando indefinidamente mientras la condición sea verdadera.

```
let n = 1;
while (n < 3) {
  // código a ejecutar
}
```

while(*condición* **) { }**

condición: condición a evaluar antes de cada ejecución del bloque, si esta nunca se hace falsa, el bucle quedará corriendo indefinidamente.

Do... while

Permite ejecutar un bloque de código mientras una condición sea verdadera. A diferencia de while la condición se evalúa al final de cada ejecución, esto se traduce en que el bloque encerrado se ejecuta al menos una vez.

```
do {
   // código a ejecutar
} while (n < 3);
```