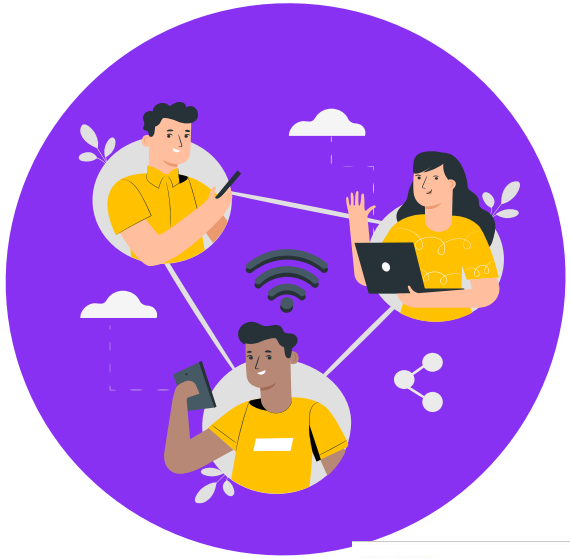
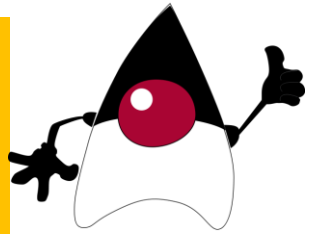
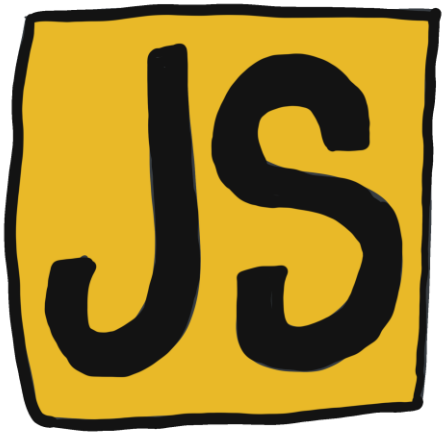
Maurisandev @MauriDeveloper maurisan4011@gmail.com

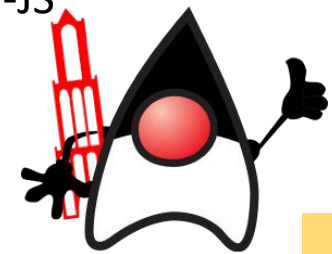


Curso Java FullStack

Codo a Codo
4.0
Clase-15
BOOTSTRAP-JS



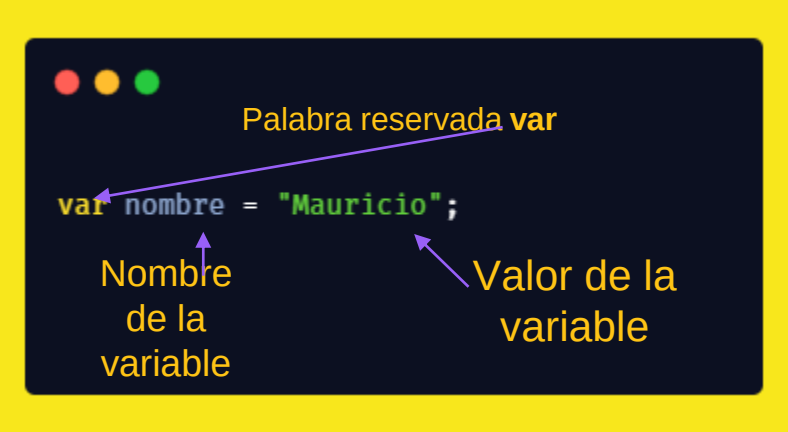




VARIABLES

Variables

Una variable es un contenedor para almacenar un valor para utilizarlo múltiples veces en el código. Ocupa un espacio en la memoria de nuestra máquina.



Variables

Las variables se usan como nombres simbólicos para valores en nuestra aplicación. Los nombres de las variables se rigen por ciertas reglas: tienen que comenzar por una letra, un guion bajo o el símbolo de $, los valores subsiguientes pueden ser números, JavaScript diferencia entre mayúsculas y minúsculas, por lo tanto las letras incluyen desde la "A a la "Z" y desde la "a" a la "z".

Variables

Para asignarle un nombre a las variables o constantes (llamados también identificadores), deben cumplir las siguientes reglas:

- ✓ El nombre debe contener solo letras, dígitos o los símbolos $y _.
- ✓ El primer carácter no debe ser un número.

Nombres reservados:

- ✓ ver acá

Reglas de nombres

No podemos declarar una variable que comience con:

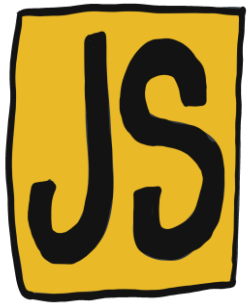
- Números
- Caracteres especiales
- Palabras reservadas de JS

```
> var 3test = 1;
Uncaught SyntaxError: Invalid or unexpected token
```

```
> var @miVariable = "Juan";
Uncaught SyntaxError: Invalid or unexpected token
```

Debemos hacerlo con:

- Guión bajo
- Signo dólar
- Cualquier palabra que no rompa las dos primeras reglas



Variables

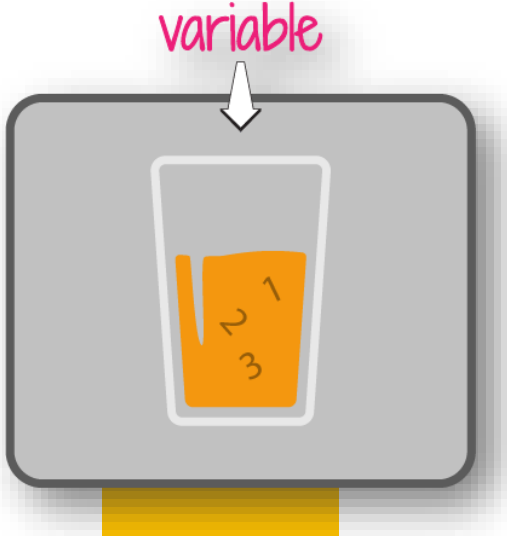
Hay 3 tipos de variables en JavaScript:

- ✓ **var:** declara una variable, iniciándola opcionalmente a un valor. Podrá cambiar el mismo y su scope es global o de función.
- ✓ **let:** declara una variable en un bloque de ámbito, iniciándola opcionalmente a un valor. Podrá cambiar su valor.
- ✓ **const:** declara una variable de sólo lectura en un bloque de ámbito. No será posible cambiar su valor mediante la asignación.

Var

var: las variables se hacen visibles en el ámbito global, es decir, que sin importar donde se definan, puede ser accedida desde cualquier parte del documento y permite que su valor pueda ser reasignado. El uso de ésta, puede dar a resultados inesperados, por eso, hay que tener cuidado de cómo se usa.

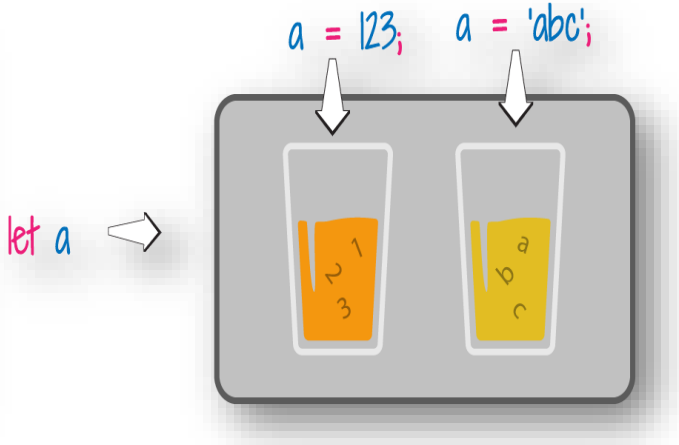
```
var nombreVariable;
var a;
var b;
```



Let

let: el alcance de estas variables, es que solo pueden ser accedidas dentro del bloque donde se definen. También, permiten que su valor pueda ser reasignado.

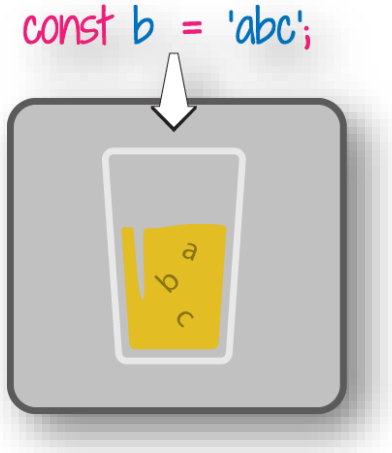
```
let nombreVariable = 'texto';
let a = 'abc';
a = 123;
let b = 1;
b = 5;
```



Const

const: estas variables (al igual que "let") solo pueden ser accedidas dentro del bloque donde están definidas, pero no permite que su valor sea reasignado, es decir, la variable se vuelve inmutable.

```
const nombreVariable = 'texto';
const a = 'Hola Mundo';
const b = 'abc';
const c = 123;
```



Ámbito de una variable

Cuando declaramos una variable fuera de una función se la denomina variable global. Cuando declaramos una variable dentro de una función se la denomina variable local, porque está disponible solo dentro de esa función donde fue creada.

Las variables en JavaScript pueden hacer referencia a una variable declarada más tarde. Este concepto se lo conoce como **hoisting**. Las variables son "elevadas" a la parte superior de la función, las variables que no se han inicializado todavía devolverán un valor **undefined**.

Tipos de Datos

- ✓ **String:** Secuencia de caracteres que representan un valor.
- ✓ **Number:** Valor numérico, entero, decimal, etc.
- ✓ **Boolean:** Valores true o false.
- ✓ **Null:** Valor nulo.
- ✓ **Undefined:** Valor sin definir.
- ✓ **Symbol:** Tipo de dato cuyos casos son únicos e inmutables.

CONTROL DE FLUJOS

Condicional: definición

Cuando en programación hablamos de condicionales, hablamos de una **estructura sintáctica** que sirve para tomar una **decisión** a partir de una **condición**.

Si <condición> entonces <operación>

JS

JS – Condicionales

Una sentencia condicional es un conjunto de comandos que se ejecutan si una condición es verdadera. JavaScript soporta dos sentencias condicionales: if...else y switch

```
var edadFacundo = 18;
var edadJuana = 25;

if (edadFacundo > edadJuana) {
    console.log("Facundo es mayor que Juana");
} else {
    console.log("Juana tiene la misma edad o es mayor que Facundo");
}
```

Se utiliza la sentencia **if** para comprobar si la condición lógica es verdadera. Se utiliza la opción **else** para ejecutar una sentencia si la condición es falsa.

JS – Condicionales

También se pueden armar sentencias más complejas usando **else if** para tener múltiples condiciones, como se muestra a continuación:

```
if (edadFacundo > edadJuana) {
    console.log("Facundo es mayor que Juana");
} else if (edadJuana > edadFacundo) {
    console.log("Juana es mayor que Facundo");
} else {
    console.log("Juana y Facundo tienen la misma edad");
}
```

JS – Condicionales

Una sentencia switch permite a un programa evaluar una expresión e intentar igualar el valor de dicha expresión a una etiqueta de caso (case). Si se encuentra una coincidencia, el programa ejecuta la sentencia asociada. Una sentencia switch se describe como se muestra a continuación:

```
var dia = "viernes";

switch (dia) {
   case 'lunes':
       console.log("Hoy es lunes. A arrancar.");
       break;
   case 'martes':
       console.log("Hoy es martes. A tomar envion");
       break;
   case 'miercoles':
       console.log("Hoy es miercoles. Aprendo a programar");
       break;
    default:
       //sentencia por defecto
       console.log("No es ni lunes, ni martes, ni miercoles");
       break;
}
```

Condicional if: sintaxis

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

```
var mostrarMensaje =
true;

if(mostrarMensaje) {
 alert("Hola Mundo");
}
```

If ... Else

```
if(condicion) {
  ...
}
else {
  ...
}
```

```
var nombre = "";

if(nombre == "") {
 alert("Aún no nos has dicho tu nombre");
}
else {
 alert("Hemos guardado tu nombre");
}
```

Condicionales Anidadas

```
if(edad < 12) {
 alert("Todavía eres muy pequeño");
}
else if(edad < 19) {
 alert("Eres un adolescente");
}
else if(edad < 35) {
 alert("Aun sigues siendo joven");
}
else {
 alert("Piensa en cuidarte un poco más");
}
```

OPERADORES EN JS

OPERADORES LÓGICOS Y RELACIONALES	DESCRIPCIÓN	EJEMPLO
==	Es igual	a == b
===	Es estrictamente igual	a === b
!=	Es distinto	a != b
!==	Es estrictamente distinto	a !== b
<, <=, >, >=	Menor, menor o igual, mayor, mayor o igual	a <=b
&&	Operador and (y)	a && b
||	Operador or (o)	a || b
!	Operador not (no)	!a

COMBINACIÓN DE OPERADORES

Ante una combinación de operadores AND **será requisito** para cumplir esa condición que **todas** las condiciones individuales se **cumplan**.

En caso de utilizar OR, la evaluación es parcial sobre cada condición y **con que una se cumpla**, se dará por válida la condición general.

COMBINACIÓN DE OPERADORES

Ya que las expresiones lógicas son evaluadas de izquierda a derecha, a veces es necesario agrupar las operaciones para asegurar que se cumplan como uno lo desea:

No es lo mismo:

```
if( ( mostrado && usuarioPermiteMensajes ) || texto != "test1") {
```

que:

```
if(  mostrado && usuarioPermiteMensajes  || texto != "test1") {
```

challenge

https://www.bitdegree.org/learn/javascript-console-log

JS

Maurisandev @MauriDeveloper maurisan4011@gmail.com

MUCHAS GRACIAS!

****

NOS VEMOS EL mañana!!

<codoa codo/>