



Nuestro compromiso es con el *futuro*.

Introducción a la programación

Repaso

Variables

- Una variable un espacio de memoria **asociado** a un valor.
- Es una forma de **almacenar** el valor de algo para usarlo más adelante.
- **Anatomía** de una variable
palabra clave nombre = valor

```

1  var name = "Juan";
2  let apellido = "Pérez";
3  const peliculaPreferida = "Titanic";
4
5  console.log(nombre)
6  console.log(apellido)
7  console.log(peliculaPreferida)
8
9  console.log(variableNoDeclarada);

```

Variables

- Var
- Let
- Const

```
1  var name = "Juan";
2  let apellido = "Pérez";
3  const peliculaPreferida = "Titanic";
4
5  console.log(nombre)
6  console.log(apellido)
7  console.log(peliculaPreferida)
8
9  console.log(variableNoDeclarada);
```

Tipos de dato

- String: es una **cadena de caracteres** encerrada por comillas dobles o simples.
- Number
- Boolean: sólo expresa valores de **verdadero** o **falso**
- undefined: hay una variable, pero **no tiene valor asignado**.
- null: es una palabra reservada para configurar un valor **vacío** a propósito.

```

1  // STRINGS
2  var transporte = "colectivo";
3  var saludo = "Hola, mi nombre es Juan";
4
5  // NUMBERS
6  var numero = 13;
7  var real = 13.2;
8  var negativo = -13;
9
10 // BOOLEAN
11 var entiendoJavascript = true;
12 var estoyAburrido = false;
13
14 // null y undefined
15 var edad;
16 console.log(edad); // undefined
17
18 var age = null;

```

Operaciones

- El módulo es el **resto** que queda luego de efectuada una división.

`dividendo % divisor = resto`

- JavaScript entiende de **precedencia de operadores**

```
1  var x = 3;
2  var y = 5;
3  var z = 10;
4
5  console.log(x * y - z)
```

```
1  // Operadores matemáticos
2
3  // + - * / ==
4
5  1 + 1 == 2
6  2 * 2 == 4
7  2 - 2 == 0
8  2 / 2 == 1
9
10 // % -> módulo
11 21 % 5 == 1;
12 21 % 6 == 3;
13 21 % 7 == 0;
```

Operaciones con strings

- Operando con strings, el operador de suma actúa **concatenando** las cadenas de caracteres.

```
1  var saludo = "¡Hola!, mi nombre es";  
2  var nombre = "Marcos";  
3  var emoticon = ":)"  
4  
5  console.log(saludo + " " + nombre + " " + emoticon)
```


Operaciones con strings

- Con la función `.length`, podemos obtener el largo del string, es decir, la cantidad de caracteres que posee la cadena.

```
7 // .length
8 console.log(saludo.length) // 20
9 console.log(nombre.length) // 6
10 console.log(emoticon.length) // 2
```

Clase 2

Funciones

Una función es una forma de encapsular ciertas instrucciones para no repetir las mismas una y otra vez y al mismo tiempo poder darle parámetros para que utilice dentro de dichas instrucciones.

```
function funcionAleatoria () {  
  | return // something  
}
```

Funciones

Los parámetros son datos que podemos “pasarle” a una función para que ésta los utilice dentro de su ejecución y de esa manera **ahorrarnos el escribir múltiples veces una misma instrucción con pequeñas variaciones**. También nos permite de esta manera encapsular estas instrucciones para lograr una mayor claridad a la hora de realizar estos procesos.

```
function sumarDosNumeros (numero1, numero2) {  
  return numero1 + numero2  
}
```

Funciones

Una función:

1. Puede recibir algo (**parámetros**)
2. Ejecutan algo (**instrucciones**)
3. Devuelve algo (un **valor**)

```
1 // declaración de una función
2 function saludar(nombre, ciudad) {
3     return "¡Hola!, mi nombre es " + nombre + " y vivo en " + ciudad;
4 }
```

Funciones

- Para invocar una función, es necesario llamarla por su **nombre**, seguido de los argumentos que requiera.
- También es posible **guardar** el valor de retorno de una función a través de la palabra reservada **return**

```

1  function mostrarNombre(nombre) {
2      |   console.log(nombre);
3      |
4      |
5      mostrarNombre("Marcos"); // Marcos
6      mostrarNombre("Juan"); // Juan
7      mostrarNombre("Mica"); // Mica

```

```

16  function sumar(numero1, numero2) {
17      |   return numero1 + numero2;
18      |
19      |
20  var resultado = sumar(4, 5)
21  console.log(resultado) // 9

```

Funciones

Veremos otra forma más moderna y compacta de escribir funciones, que nos permitirá acortar mucho el código y simplificar en gran medida la sintaxis, sobre todo en lo que respecta a utilizar las funciones como parte de otras estructuras más complejas.

A esas funciones más modernas se las conoce como Arrow Functions.

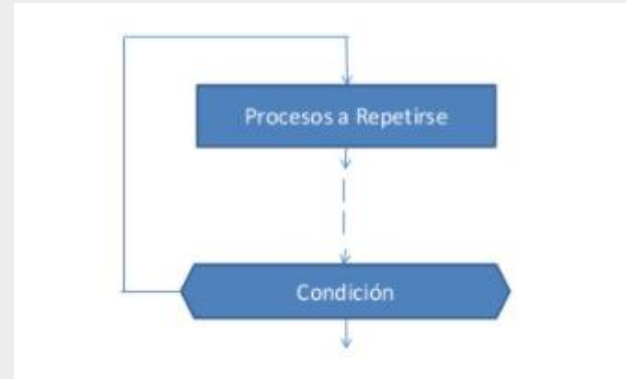
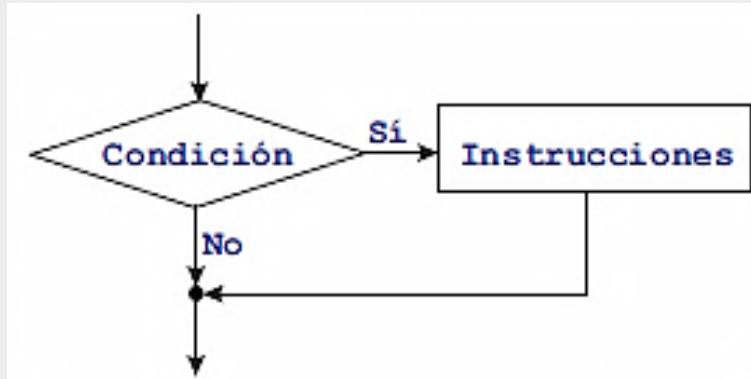
Funciones

Y se ven más o menos así

```
1  function restar(numero1, numero2) {  
2    |   return numero1 - numero2;  
3  }  
4  
5  const restar = function (numero1, numero2) {  
6    |   return numero1 - numero2;  
7  }  
8  
9  const restar = (numero1, numero2) => numero1 - numero2;
```


Estructuras de control

Las estructuras de control son formas que tenemos de tomar decisiones dentro de un algoritmo. Son una forma de guiar el flujo de los datos según algún criterio.



Operadores lógicos

Establecen **una relación** entre dos operandos

- > → mayor
- >= → mayor o igual
- < → menor
- <= → menor o igual
- == → igualdad simple
- === → igualdad estricta
- != → distinto

```

1  // Operadores lógicos
2
3  5 > 4    // true
4  5 >= 5   // true
5  5 == 0   // false
6  5 < 4    // false
7  5 <= 4   // false
8  5 != 0   // true
9
10 5 === "5" // false
11 5 == "5"  // true
    
```

Estructuras condicionales

IF

```
if (condition) {  
    // something  
}
```

IF - ELSE

```
if (condition) {  
    // something  
} else if (anotherCondition) {  
    // something else  
}
```

Estructuras selectivas

SWITCH

```
switch (variable) {  
    case value:  
        // do something  
        break;  
  
    case value:  
        // do something  
        break;  
  
    case value:  
        // do something  
        break;  
  
    default:  
        // do something  
        break;  
}
```

Estructuras repetitivas

FOR

```
for (let i = 0; i < array.length; i++) {  
  // do something  
}
```

Estructuras repetitivas

WHILE

```
while (condition) {  
    // something  
}
```

DO WHILE

```
do {  
    // something  
} while (condition);
```

Arrays

Se le denomina vector, formación, matriz a una **zona de almacenamiento contiguo** que contiene una serie de elementos (del mismo tipo o no), los elementos de la matriz.

Desde el punto de vista lógico una matriz se puede ver como un conjunto de elementos ordenados en fila (o filas y columnas, si tuviera dos dimensiones).

[illegible]

En definitiva, el primer elemento tiene lo que se llama **índice** 0, que no es más que su posición dentro del array; el segundo elemento tiene un **índice** de 1, etc.

[illegible]

Arrays

Como toda lista, vamos a poder manipular sus elementos: vamos a poder agregarlos, eliminarlos, contarlos, ordenarlos, etc.

pop()

push()

toString()

join()

splice()

sort()

shift()

unshift()

reverse()

concat()

slice()

filter()

find()

forEach()

map()

reduce()

every()

some()

Módulos

¿Qué son los módulos en JavaScript / Nodejs?

Son una manera que tenemos de compartir código entre varios archivos, de esta manera podemos literalmente “modularizar” nuestra aplicación, dividiendo los contenidos en fragmentos o componentes más simples de leer y comprender, y poder también poder reutilizarlos en varios lugares sin tener que repetir muchas veces el mismo código.

Módulos

En Nodejs vamos a utilizar un tipo de declaración de módulos que se llama CommonJS, que si bien no es la forma más moderna, es la que encontraremos casi siempre, debido a la compatibilidad que necesita tener Nodejs con los servidores que existen en la actualidad.

Más adelante aprenderemos también cómo utilizar la forma más moderna, llamada ES2015 Modules o ESM, para acortar.

```
const express = require('express')
```

```
import express from 'express'
```

Muchas gracias!



ICARO Asociación Civil
CUIT 30716564815
info@icaro.org.ar
www.icaro.org.ar

Vamos al código!