



**Certified Tech  
Developer**

The Ultimate Degree

## Programación Imperativa

# Funciones y procedimientos

Sabemos que una función busca concentrar en un único espacio una cantidad de pasos a realizar bajo un nombre que describa el propósito de dichos pasos, por ejemplo: Poner(Rojo), Mover(Norte), sumar(2, 2), console.log("hola Mundo"). Practicaremos la creación y uso de esta herramienta en el lenguaje de JavaScript, en el entorno de Node.js, con el editor de texto VS Code.

## Objetivo

Dominar las herramientas de desarrollo, ordenar nuestro código en carpetas y archivos, definir e invocar funciones en JS, darles nombres descriptivos y combinarlas entre ellas para un fin más complejo.

## Ejemplo: Math

Como vimos, JS ya trae incluido en su código una serie de herramientas, entre ellas, unas cuantas sobre matemáticas. Si escribimos en un archivo .js el procedimiento `Math.random()` ([link](#)) y lo ejecutamos, parece que nada sucede, sin embargo, sí sucede, el problema es que no le pedimos a esa función que lo que haga nos lo muestre por la consola.

Sería así: `console.log( Math.random() );`, fijémonos cómo el log se encarga de mostrar en la consola aquello que tiene entre sus paréntesis, que en este caso es eso



que hace `Math.random()` y que no habíamos podido ver. ¿Salió algo ahora en la consola?

Si lo ejecutamos varias veces a ese `archivo.js`, ¿siempre da un resultado diferente? Esto es porque `random` quiere decir aleatorio y lo que hace es entregar un número aleatorio entre 0 y 1.

Ahora bien, estas 2 funciones, `console.log()` y `Math.random()` no las creamos, sino que ya vienen con el lenguaje, sólo las invocamos o llamamos. Se les dice así cuando queremos ejecutarlas. Nos toca crear nuestras propias funciones.

## Definición e invocación de una función

Definir y crear una función es lo mismo, en este caso, definamos una que muestre en la consola un saludo. Luego, para poder ejecutarla, debemos invocarla o llamarla. Que sea algo así:

```
function saludar() {  
  console.log("hola, tanto tiempo sin verte.");  
}  
  
saludar()
```

Muy bien, ahora deberemos modificar la función para que tome por parámetro un nombre y salude a esa persona cuando el lenguaje la ejecute.



## Un caso extraño

El operador matemático **+** en algunos lenguajes y en JS funciona para unir dos cadenas de texto o Strings. Por ejemplo, si queremos unir un saludo a un nombre haríamos algo así:

```
function saludarA(nombre) {  
  const mensaje = "hola, tanto tiempo sin verte "  
  console.log(mensaje + nombre);  
}  
  
saludarA("Ezequiel")  
saludarA("Diego")
```

Como podemos ver, esta función no retorna o devuelve ningún valor, solo muestra algo en la consola. Veamos eso de retornar, ¿cómo es?

## Funciones que retornan valor

Como contó la profesora del video, una función es una máquina, entra un valor y devuelve un resultado. En JS ese ejemplo se escribe así:

```
function multiplicarPorDosYSumarleCinco(x) {  
  return 2 * x + 5  
}  
multiplicarPorDosYSumarleCinco(1)  
multiplicarPorDosYSumarleCinco(5)  
multiplicarPorDosYSumarleCinco(pepe) //pepe no es un número  
const pepe = 5  
multiplicarPorDosYSumarleCinco(pepe) //Ahora si
```



Pero acá falta algo, ¿se ve el resultado en la consola? ¿Qué debemos agregar?

Claro, el procedimiento o la función que se encarga de mostrar valores en la consola.

```
console.log(multiplicarPorDosYSumarleCinco(pepe));
```

Pero que largo escribir todo eso, ¿se podrá guardar en una variable aquello que retorna de una función? ¿Y se podrá, entonces, pasar esa variable a la consola?

```
let valor = multiplicarPorDosYSumarleCinco(pepe)  
console.log(valor);
```

¡Atención! No olvidemos que el signo = en programación se usa para asignar un valor a una variable, es distinto a cómo se usa en matemáticas. Ahora ya tenemos todo lo necesario para hacer los ejercicios de las mesas de trabajo. ¡Mucha suerte!