# Clase 4

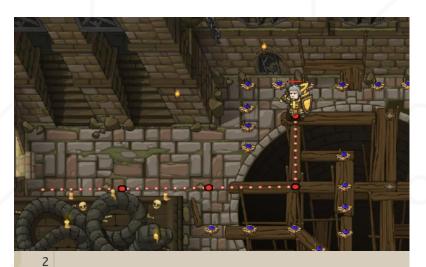
**Funciones** 

DEV.F.

### **FUNCIONES**

Una función en JavaScript es similar a un procedimiento.

Un conjunto de instrucciones que realiza una tarea o calcula un valor



```
3 hero.moveRight(3);
4 hero.moveUp();
5
```



## **FUNCIONES**

Recibe elementos de entrada por ejemplo x, y

Entrega un valor de salida

Podemos pasar fórmulas matemáticas para obtener cierto valor

$$f(x,y) = x^2 + y^2$$





### RECETA PARA TRABAJAR CON FUNCIONES

- 1.- DECLARAR MI FUNCIÓN.
- 2.- PODEMOS ESTABLECER LOS PARÁMETROS O BIEN PEDIRLE AL USUARIO QUE INGRESE ESOS PARÁMETROS.
- 3.- INGRESAR LAS INSTRUCCIONES A MI FUNCIÓN, ES DECIR, EL CÓDIGO QUE VA A EJECUTAR.
- 4.- LLAMAR LA FUNCIÓN QUE DECLARAMOS PREVIAMENTE.





# **DECLARAR NUESTRA FUNCIÓN**

Nombre de mi función

function MyFunction ( ) {

}



# **PARÁMETROS**

```
function MyFunction( ) {
   function MyFunction( params ) {
function MyFunction (num1, nombre, edad){
```

### **INSTRUCCIONES**

```
function Suma ( num1, num2 ) {
       Instrucciones | bloque de
código
   function Suma ( num1, num2 ) {
         let total = num1 + num2;
         return "LA SUMA ES: " + total;
```



# **LLAMAR MI FUNCIÓN**

```
function Suma ( num1, num2 ) {
     let total = num1 + num2;
     return "LA SUMA ES: " + total;
Suma(2,2)
"LA SUMA ES: 4"
```



### **FUNCIONES ANIDADAS**

```
EJEMPLO DE FUNCIÓN ANIDADA
function ObtenerGoles() {
        var partido1 = 3;
        var partido2 = 3;
 function Agregar() {
    var nombre = "juanito";
    return nombre + " anoto "
    + (partido1 + partido2) + " goles";
 return Agregar()
        ObtenerGoles()
```





# Console.log vs return

console.log("some text here")

Imprime lo que sigue dentro de los paréntesis

return "some text here" + variable

devuelve el valor de una operación. Esto sirve para luego asignar ese valor a una variable.

El procesador realiza las operaciones de la función, pero sólo va a guardar su resultado en memoria si le indicamos return. Y una vez guardado en memoria podemos rescatar ese valor para asignarlo a una variable y poder seguir haciendo operaciones con ese valor.



# Tipos de funciones

#### Declaración de función:

Consta de la palabra clave **function**, seguida de:

- El nombre de la función.
- Una lista de parámetros de la función, entre paréntesis y separados por comas.
- Las declaraciones de JavaScript que definen la función, encerradas entre llaves, { ... }.

```
function square(number) {
  return number * number;
}
```

```
return number * number;
```



# Tipos de funciones

#### Expresión de función:

Esta función puede ser **anónima**; no tiene por qué tener un nombre. Por ejemplo, la función square se podría haber definido como:

```
const square = function(number) { return number * number }
var x = square(4) // x obtiene el valor 16
```



# Tipos de funciones

#### **Funciones Flecha:**

Una expresión de función flecha tiene una sintaxis más corta en comparación con las expresiones de función y no tiene su propio **this**. Las funciones flecha siempre son anónimas y disponen de retorno implícito.

```
const sumarDos = (num1, num2) => {
  console.log(num1 + num2)
}
sumarDos(10, 50)
```



#### Modo 1: Función declarada

```
function hacerSonido (){
  console.log("Pling!");
}
hacerSonido();
```

#### Modo 2: Expresión de Función

```
const hacerSonido = function() {
  console.log("Pling!");
};
hacerSonido();
// → Pling!
```



#### Modo 3: Función Flecha o Arrow Function

```
const hacerSonido = () => {
    console.log("Pling!");
}
hacerSonido();
```



```
const sumarDos = (num1, num2) => {
  console.log(num1 + num2)
sumarDos(10, 50)
const sumarDos = num1 => {
  console.log(num1)
sumarDos(10)
const sumarDos = num1 => {
  return num1
console.log(sumarDos(10))
const sumarDos = num1 => num1
```

console.log(sumarDos(10))

```
POSIBILIDADES DE ARROW FUNCTION
```

