



**DigitalHouse** >  
Coding School

# Funciones



**Certified Tech  
Developer**

The Ultimate Degree

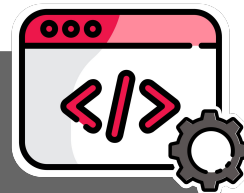
# Índice

1. [Retomando funciones](#)
2. [Arrow functions](#)
3. [Funciones como parámetros](#)

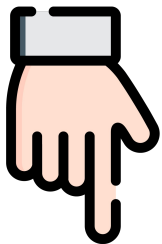
# 1 | Retomando funciones

# ¿Qué es una **función**?

```
function sumar(par1, par2) {  
  
    let valor = par1, par2;  
  
    return valor;  
  
}  
  
sumar(1, 3);
```



JavaScript cuenta con estructuras denominadas funciones, estas no son más que fragmentos de código, los cuales **no** se ejecutan hasta que, en algún momento, lo “invoquemos” o llamemos mediante un nombre o identificador.



```
function sumar(par1, par2) {  
    let valor = par1, par2;  
  
    return valor;  
  
}  
  
sumar(1, 3);
```

Aquí utilizamos la palabra clave **function** para indicar una declaración de una función denominada **suma** la cual recibe dos parámetros **par1** y **par2**.



```
function sumar(par1, par2) {
```

```
    let valor = par1, par2;
```

```
    return valor;
```

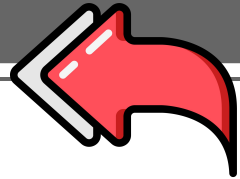
```
}
```

```
sumar(1, 3);
```

Definimos qué es lo que hace la función, podría ser cualquier algoritmo que nos sea útil.

```
function sumar(par1, par2) {  
  
    let valor = par1, par2;  
  
    return valor;  
  
}  
  
sumar(1, 3);
```

Mediante **return**, salimos de la función y volvemos a la línea donde fue llamada. Además devolvemos podemos devolver un **valor** o bien simplemente no hacerlo mediante el uso de **return;** solo.



```
function sumar(par1, par2) {  
  
    let valor = par1, par2;  
  
    return valor;  
  
}
```

```
sumar(1, 3);
```



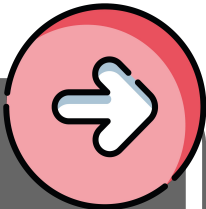
Llamamos o invocamos a la función **sumar** con los parámetros **1** y **3**.



# 2 | Arrow functions

# Arrow functions

```
let sumar = (a, b) => {  
    let valor = a + b;  
    return valor;  
}  
  
sumar(1, 3);
```



Existe otra forma de declarar una función mediante la notación de '**arrow function (=>)**'. Esta función se comporta de la misma manera que las funciones normales de Javascript e incluso se invocan de la misma manera.

# Algunas particularidades

En el fragmento de código siguiente podemos ver una arrow function que realiza exactamente lo mismo que en los anteriores ejemplos

## Nombre de función

En este caso la función se denomina como la variable a la cual se le asigna.

## Return rápido

Esta forma de declarar una función permite devolver sin utilizar la palabra return, esto sucede cuando no usamos llaves para encerrar nuestro algoritmo. Esto generalmente solo es utilizado cuando se puede expresar en una única línea.

JS

```
let sumar = (a, b) => a + b;
```

Parámetros

3

## Funciones cómo parámetros

# Funciones **cómo parámetros**

JavaScript nos permite utilizar una función **como parámetro de otra**, esto es de gran utilidad para nuestro código:



```
{  
  function ejecutor(func) {  
    // código de la función  
    func(1, 2);  
    // código de la función  
  }  
  function sumar(a, b) {  
    return a + b;  
  }  
  ejecutor(sumar);  
}
```



En el ejemplo, se puede ver que tenemos una función **ejecutador**, esta función realiza todo su algoritmo y, cuando lo necesita, ejecuta la función **func** pasada como parámetro, que corresponde a la función **suma**. Se suele decir en estos casos, que la función **ejecutador** es la responsable de ejecutar la función **sumar** (o cualquier otra que le pasemos).

DigitalHouse>  
Coding School