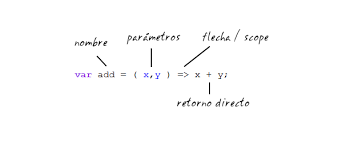
¿Qué es función flecha?

Una expresión de **función flecha** es una alternativa compacta a una expresión de **función** tradicional, pero es limitada y no se puede utilizar en todas las situaciones. Diferencias y limitaciones: No tiene sus propios enlaces a this o super y no se debe usar como métodos.

¿Cuándo usar funciones Flecha JavaScript?



**Con las funciones flecha, buscamos simplificar todo lo anterior de un modo mucho más directo y declarativo:**

1. Eliminamos la palabra reservada function y nos limitamos a recoger los parámetros mediante los paréntesis tradicionales.
2. Podemos eliminar las llaves que delimitan el scope abriéndolo con una **flecha**.

# Funciones Flecha

Una **expresión de función flecha** es una alternativa compacta a una expresión de función tradicional, pero es limitada y no se puede utilizar en todas las situaciones.

**Diferencias y limitaciones:**

* No tiene sus propios enlaces a this o super y no se debe usar como [métodos](https://developer.mozilla.org/es/docs/Glossary/Method).
* No tiene argumentos o palabras clave new.target.
* No apta para los métodos call, apply y bind, que generalmente se basan en establecer un [ámbito o alcance](https://developer.mozilla.org/es/docs/Glossary/Scope)
* No se puede utilizar como [constructor](https://developer.mozilla.org/es/docs/Glossary/Constructor).
* No se puede utilizar yield dentro de su cuerpo.

### [Comparación de funciones tradicionales con funciones flecha](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow_functions#comparaci%C3%B3n_de_funciones_tradicionales_con_funciones_flecha)

Observa, paso a paso, la descomposición de una "función tradicional" hasta la "función flecha" más simple:  
Nota: Cada paso a lo largo del camino es una "función flecha" válida

// Función tradicionalfunction (a){

return a + 100;

}

// Desglose de la función flecha

// 1. Elimina la palabra "function" y coloca la flecha entre el argumento y el corchete de apertura.

(a) => {

return a + 100;

}

// 2. Quita los corchetes del cuerpo y la palabra "return" — el return está implícito.

(a) => a + 100;

// 3. Suprime los paréntesis de los argumentos

a => a + 100;

Como se muestra arriba, los { corchetes }, ( paréntesis ) y "return" son opcionales, pero pueden ser obligatorios.

Por ejemplo, si tienes **varios argumentos** o **ningún argumento**, deberás volver a introducir paréntesis alrededor de los argumentos:

// Función tradicional

function (a, b){

return a + b + 100;

}

// Función flecha

(a, b) => a + b + 100;

// Función tradicional (sin argumentos)

let a = 4;

let b = 2;

function (){ return a + b + 100;

}

// Función flecha (sin argumentos)

let a = 4;

let b = 2;() => a + b + 100;

Del mismo modo, si el cuerpo requiere **líneas de procesamiento adicionales**, deberás volver a introducir los corchetes **Más el "return"** (las funciones flecha no adivinan mágicamente qué o cuándo quieres "volver"):

// Función tradicional

function (a, b){

let chuck = 42;

return a + b + chuck;

}

// Función flecha

(a, b) => {

let chuck = 42; return a + b + chuck;

}

Y finalmente, en las **funciones con nombre** tratamos las expresiones de flecha como variables

// Función tradicional

function bob (a){ return a + 100;}

// Función flechalet bob = a => a + 100;

## [Sintaxis](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow_functions#sintaxis)

### [Sintaxis básica](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow_functions#sintaxis_b%C3%A1sica)

Un parámetro. Con una expresión simple no se necesita return:

param => expression

Varios parámetros requieren paréntesis. Con una expresión simple no se necesita return:

(param1, paramN) => expression

Las declaraciones de varias líneas requieren corchetes y return:

param => { let a = 1; return a + b;}

Varios parámetros requieren paréntesis. Las declaraciones de varias líneas requieren corchetes y return:

(param1, paramN) => { let a = 1; return a + b;}

### [Sintaxis avanzada](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow_functions#sintaxis_avanzada)

Para devolver una expresión de objeto literal, se requieren paréntesis alrededor de la expresión:

params => ({foo: "a"}) // devuelve el objeto {foo: "a"}

Los parámetros rest son compatibles:

(a, b, ...r) => expression

Se admiten los parámetros predeterminados:

(a=400, b=20, c) => expression

Desestructuración dentro de los parámetros admitidos:

([a, b] = [10, 20]) => a + b;

// el resultado es 30

({ a, b } = { a: 10, b: 20 }) => a + b;

// resultado es 30