teoría complementaria arrow func

Una **expresión de función flecha** es una alternativa compacta a una [expresión de función](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/function) tradicional, pero es limitada y ***no se puede utilizar en todas las situaciones***.

## [**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 + param; } |
| --- |

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](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/rest_parameters) son compatibles:

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

Se admiten los [parámetros predeterminados](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Default_parameters):

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

[Desestructuración](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment) 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 |
| --- |

### [**Saltos de línea**](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow_functions#saltos_de_l%C3%ADnea)

Una función flecha no puede contener un salto de línea entre sus parámetros y su flecha.

| var func = (a, b, c)  => 1; // SyntaxError: expresión esperada, obtuve '=>' |
| --- |

Sin embargo, esto se puede modificar colocando el salto de línea después de la flecha o usando paréntesis/llaves como se ve a continuación para garantizar que el código se mantenga bonito y esponjoso. También puedes poner saltos de línea entre argumentos.

| var func = (a, b, c) =>  1;  var func = (a, b, c) => {  return 1 };  var func = (  a,  b,  c ) => 1; |
| --- |

// no se lanza SyntaxError

### [**Orden de procesamiento**](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow_functions#orden_de_procesamiento)

Aunque la flecha en una función flecha no es un operador, las funciones flecha tienen reglas de procesamiento especiales que interactúan de manera diferente con [prioridad de operadores](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Operator_Precedence) en comparación con las funciones regulares.

| let callback;  callback = callback || function() {}; // ok  callback = callback || () => {}; // SyntaxError: argumentos de función flecha no válidos  callback = callback || (() => {}); // bien |
| --- |

## [**Ejemplos**](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow_functions#ejemplos)

### [**Uso básico**](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow_functions#uso_b%C3%A1sico)

| // Una función flecha vacía devuelve undefinided let empty = () => {};  (() => 'foobar')(); // Devuelve "foobar" // (esta es una expresión de función invocada inmediatamente)  var simple = a => a > 15 ? 15 : a; simple(16); // 15 simple(10); // 10  let max = (a, b) => a > b ? a : b;  // Fácil filtrado de arreglos, mapeo, ...  var arr = [5, 6, 13, 0, 1, 18, 23];  var sum = arr.reduce((a, b) => a + b); // 66  var even = arr.filter(v => v % 2 == 0); // [6, 0, 18]  var double = arr.map(v => v \* 2); // [10, 12, 26, 0, 2, 36, 46]  // Cadenas de promesas más concisas promise.then(a => {  // ... }).then(b => {  // ... });  // Funciones flecha sin parámetros que son visualmente más fáciles de procesar setTimeout( () => {  console.log('sucederá antes');  setTimeout( () => {  // código más profundo  console.log ('Sucederá más tarde');  }, 1); }, 1); |
| --- |

ref: [Funciones Flecha - JavaScript | MDN](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow_functions)