
ES6
ROCKS

ARROW FUNCTIONS

ARROW FUNCTIONS

Las arrow functions son una nueva forma de crear funciones

```
// forma clásica
function sumar(a, b) {
 return a + b;
}
console.log( sumar(1, 3) );
// 4
```

```
// ES6 arrow function
const sumar = (a, b) => a + b;

console.log( sumar(1, 3) );
// 4
```

ARROW FUNCTIONS: SINTAXIS

Las arrow functions se crean utilizando el token **=>**. Son anónimas. Del lado izquierdo de la flecha, están los parámetros. Del lado derecho, la/s acción/es a ejecutar.

```
const sumar = (a, b) => {
  return a + b;
}
console.log( sumar(2, 3) ); // 5
console.log( sumar(7, 14) ); // 21
console.log( sumar(4, 2) ); // 6
```

ARROW FUNCTIONS: SINTAXIS

Las arrow functions tienen el return implícito siempre que la función tenga una sola expresión (sin las llaves), por lo que podemos evitar usar la palabra **return** y las llaves

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


console.log( sumar(2, 3) ); // 5
console.log( sumar(7, 14) ); // 21
console.log( sumar(4, 2) ); // 6
```

ARROW FUNCTIONS: SINTAXIS

Cuando una arrow function tiene un solo parámetro, los paréntesis son opcionales

```
function saludar0 (nombre) {
 return `¡Hola ${nombre}!`;
}
const saludar1 = (nombre) => {
 return `¡Hola ${nombre}!`;
}
const saludar2 = nombre => {
 return `¡Hola ${nombre}!`;
}
const saludar3 = nombre => `¡Hola ${nombre}!`;
```

```
console.log(saludar0('Ada'));
// ¡Hola Ada!
console.log(saludar1('Grace'));
// ¡Hola Grace!
console.log(saludar2('Hedy'));
// ¡Hola Hedy!
console.log(saludar3('Mary'));
// ¡Hola Mary!
```

ARROW FUNCTIONS: SINTAXIS

Cuando una arrow function no tiene parámetros, necesitamos poner los paréntesis.

```
const mostrarSaludo = () => {
 console.log('¡Hola!');
}

const mostrarSaludoMinima = () => console.log('¡Hola!');

mostrarSaludo(); // muestra por consola ¡Hola!
mostrarSaludoMinima(); // muestra por consola ¡Hola!
```

ARROW FUNCTIONS: SINTAXIS

Siempre que una arrow function tenga más de una expresión entre sus acciones, es obligatorio poner las llaves (y también necesitaríamos usar return si vamos a retornar algo)

```
const saludar = nombre => {
 if (nombre) {
   return `¡Hola, ${nombre}!`;
 } else {
   return `¡Hola, anónimo!`;
 }
}
console.log( saludar() ); // ¡Hola, anónimo!
console.log( saludar('Ada') ); // ¡Hola, Ada!
```

ARROW FUNCTIONS: VENTAJAS

Las ventajas más importantes que nos suma arrow functions son 2:

1. Una forma de escribir funciones más reducida que utilizando las funciones típicas creadas con **function**
2. No hace el binding de **this** (veremos esto mas adelante!)

ARROW FUNCTIONS: EJERCICIOS

1. Convertí las siguientes funciones a arrow functions

```
var sumarLosTres = function (num1, num2, num3) {
 return num1 + num2 + num3;
}
let cuadrado = function (num) {
 return num * num;
}
const decirHola = function () {
 console.log('¡Hola!');
}
```

ARROW FUNCTIONS: EJERCICIOS

2. Convertí la siguiente función a una arrow functions

```
function saludar (nombre) {
 if (nombre === undefined) {
   return "hola anónimo";
 } else {
   return "hola " + nombre;
 }
}
```

ARROW FUNCTIONS: EJERCICIOS

3. Arreglá las siguientes arrow functions

```
const nombreCompleto = nombre, apellido => {
 return `${nombre} ${apellido}`;
}
const exclamar = str => {
 `¡${str}!`
}
const mayusculas = (str) => return str.toUpperCase()
const minusculas = (str) => str.toLowerCase()
console.log( nombreCompleto('Ada', 'Lovelace') ); // Ada Lovelace
console.log( exclamar('Hedy') ); // ¡Hedy!
console.log( mayusculas('grace') ); // GRACE
console.log( minusculas('SHERYL') ); // sheryl
```

PARÁMETROS POR DEFECTO

PARÁMETROS POR DEFAULT

Los parámetros por default permiten que los parámetros de una función puedan ser definidos con un valor inicial.
Partiendo del siguiente ejemplo:

```
const suma = (a, b) => {
 return a + b;
}
console.log( suma() );
// retorna NaN (not a number)
```

Cómo ejecuta la función **suma** sin parámetros, **a** y **b** son *undefined*

PARÁMETROS POR DEFAULT: SINTAXIS

Para asignarle un valor por defecto a cualquier de los parámetros, tenemos que agregar (después del parámetro) ***= valor***

```
const suma = (a = 0, b = 0) => {
 return a + b;
}
console.log( suma() );
// retorna 0
```

Cómo ejecuta la función **suma** sin parámetros, **a** y **b** ahora toman el valor por defecto

PARÁMETROS POR DEFAULT: SINTAXIS

Podemos asignar un valor por default a uno, varios o ningún parámetro. En el ejemplo anterior le asignamos un valor a los 2 parámetros. Pero podemos hacerlo solo con **b**

```
const suma = (a, b = 0) => {
 return a + b;
}
console.log( suma() );
// retorna NaN (not a number)
console.log( suma(3) );
// retorna 3
```

Como ejecuta la función **suma** sin parámetros, **a es undefined** y **b** toma el valor por defecto

PARÁMETROS POR DEFAULT: EJERCICIOS

1. Modificá la siguiente función, agregando parámetros por default, para que al ejecutar los casos de prueba no tire ningún error.

```
const longitudDelNombre = (nombre) =>
nombre.length


console.log( longitudDelNombre() ); // 0
console.log( longitudDelNombre('Ada') ); // 3
```

PARÁMETROS POR DEFAULT: EJERCICIOS

2. ¿Cómo modificarías la siguiente función para seguir obteniendo el mismo resultado, pero con un código más reducido?

```
const saludarVisitanteWeb = (nombreUsuario) => {
 if (nombreUsuario === undefined) {
   return '¡Hola anónimo!';
 } else {
   return `¡Hola ${nombreUsuario}!`;
 }
}
console.log( saludarVisitanteWeb() ); // ¡Hola anónimo!
console.log( saludarVisitanteWeb('Hedy') ); // ¡Hola Hedy!
```

¡Gracias!