Tarea Independiente 07/08/2025

David Núñez Franco

August 10, 2025

Inventario de Conceptos Claves

- Evolución y popularidad de lenguajes
- TIOBE y popularidad de lenguajes
- Abstracción en lenguajes
- Modelo pirámide
- Multiparadigma
- FP-OOP
- FP como un nivel de abstracción superior
- FP primer principio: función como objeto
- Arrow (flecha gorda =¿) en JS
- Punto-y-coma debate DRY o no DRY
- beta reducción

Una pequeña introducción

Antes de desarrollar los ejercicios, veamos la estructura a seguir entre una función clásica y una arrow function:

```
function saludar(nombre) {
    return "Hola, " + nombre + "!";
}

console.log(saludar("Nunez")); // Salida: Hola, Nunez!
```

Listing 1: Classic Function en JavaScript

```
const saludarFlecha = (nombre) => {
    return `Hola, ${nombre}!`;
};

console.log(saludarFlecha("Nunez")); // Salida: Hola, Nunez!
```

Listing 2: Arrow Function en JavaScript

Algunas diferencias entre ambas funciones (Ejercicio 4)

Sintaxis
- **Clásica**: `function nombre(param) { ... }`
- **Flecha**: `(param) => { ... }` (más corta y expresiva).

`this`
- **Clásica**: Tiene su propio `this`, depende de cómo se llame la función.
- **Flecha**: No tiene su propio `this`, hereda el del contexto donde se define.

Uso como método de objeto
- **Clásica**: Adecuada para métodos que necesiten su propio `this`.
- **Flecha**: No recomendable si el método debe usar `this` del objeto.

Constructores (`new`)
- **Clásica**: Puede usarse como constructor con `new`.
- **Flecha**: No puede ser usada como constructor.

`arguments`
- **Clásica**: Tiene acceso al objeto `arguments` (lista de todos los parámetros).
- **Flecha**: No tiene `arguments`, hay que usar parámetros *rest* (`...args`).

Ejercicio 1

Escriba una función en JS choose(p, f, g) que retorne una función de x que retorna f(x) si p(x) es verdadero y retorna g(x) en caso contrario Ejemplos choose(x =¿ x ¿ 0, x =¿ x**2, x =¿ x + 1)(5) // retorna 25 choose(x =¿ x ¿ 0, x =¿ x**2, x =¿ x + 1)(-5) // retorna -4

Solución al planteamiento

Se solicita una función choose(p, f, g) que:

- p: es una función predicado (retorna true o false al evaluar x)
- f: es una función que se aplica si el predicado es verdadero
- g: es una función que se aplica si el predicado es falso

Función choose retorna otra función que reciba x, y luego:

- Si p(x) es true, retorna f(x)
- Si p(x) es false, retorna g(x)

```
function choose(p, f, g) {
    return function (x) {
        if (p(x)) {
            return f(x);
        } else {
            return g(x);
        }
    };
}

console.log(
choose(
(x) => x > 0,
(x) => x ** 2,
(x) => x + 1
)(5)
); // 25
console.log(
choose(
(x) => x > 0,
(x) => x ** 2,
(x) => x + 1
)(-5)
); // -4
```

Listing 3: classic function

```
const choose = (p, f, g) => (x) => p(x) ? f(x) : g(x);

console.log(
choose(
(x) => x > 0,
(x) => x ** 2,
(x) => x + 1
)(5)
); // 25
console.log(
choose(
(x) => x > 0,
(x) => x ** 2,
(x) => x + 1
)(-5)
); // -4
```

Listing 4: arrow function

Ejercicio 2

Escriba id la función identidad: es decir, para cualquier x se cumple id(x) = x.

Solución al planteamiento

Esta función devuelve exactamente lo que recibe. No lo transforma, procesa o modifica.

```
function id(x) {
    return x;
}

console.log(id(5)); // 5
console.log(id("Hola")); // Hola
console.log(id(true)); // true
console.log(id([1, 2, 3])); // [1, 2, 3]
```

Listing 5: classic function

```
const id = (x) => x;

console.log(id(5)); // 5
console.log(id("Hola")); // Hola
```

Listing 6: arrow function

Ejercicio 3

Escriba una función repeat(n, f) que retorne una función de x que calcula f**n(x) = f(f(...f(x)...)) donde f está aplicada n veces sobre x (este ejercicios es "peludillo") Ejemplos repeat(0, id)(666) // retorna 666 repeat(5, x =¿ 2 * x)(1) // retorna 32

Solución al planteamiento

Queremos una función repeat(f, n) que retorne otra función de x, y que aplique f exactamente n veces sobre x. Pero, hay casos especiales:

- Si n = 0 no se aplica f (se devuelve el valor original)
- Si n ¿ 0 aplicar f las n veces

NOTA: USAMOS let CUANDO SE NECESITE UNA VARIABLE QUE PUEDA CAMBIAR SU VALOR, PERO LIMITADA AL BLOQUE DONDE SE DECLARE.

```
const repeat = (n, f) => (x) => {
    let result = x;

    for (let i = 0; i < n; i++) {
        result = f(result);
    }

    return result;
};

console.log(repeat(0, (x) => x)(666)); // 666
console.log(repeat(5, (x) => 2 * x)(1)); // 32, porque 2*1 =
    2, 2*2 = 4, 2*4 = 8, 2*8 = 16, 2*16 = 32 (5 veces en
   total)
```

Listing 7: arrow function

```
const repeat = (n, f) => (x) => n === 0 ? x : repeat(n - 1,
   f)(f(x));

console.log(repeat(0, (x) => x)(666));
console.log(repeat(5, (x) => 2 * x)(1));
```

Listing 8: recursive (ChatGPT helped me)