Tarea Independiente 25/08/2025

David Núñez Franco

August 27, 2025

Inventario de Conceptos Claves

- Faceta declarativo (faceta de expresiones de lenguaje)
- Faceta operativa (faceta de verbos imperativos de lenguaje)
- Expresiones versus estatuto
- FP Principio de declaratividad: preferir la faceta declarativa sobre la operacional imperativa
- nodemon: Forma correcta de uso de node, para monitorear la carpeta actual de un archivo digamos main.mjs
  - nodemon -w . main.mjs
- Nociones módulos en JS, estándar MJS (export/import)
- for -of: Variante del for basado en iterador (similar a for(var e:a)... de Java)
- Operador spread(...)
- Predicado: Función que retorna algo booleano
- Patrón de autómata que representa un típico ciclo de estados
- COmbinadores map, filter, reduce de arrays (Se llama combinador a una función de funciones, usualmente. Eso viene de lógica combinatoria, que es otra forma de ver el cálculo lambda).

Ejercicio 1

1) Sea a es un array cualquiera y sean f y g funciones cualesquiera. ¿Bajo qué condiciones razonables que se pongan sobre f y/o g (según el caso) se cumplen las siguientes identidades?

a) a.map(f) es lo mismo que a b) a.map(f).map(g) tiene el mismo valor que a.map(g).map(f) c) Escriba una expresión equivalente pero más simple que a.filter(f).filter(g) asumiendo f y g son predicados (i.e. funciones que retornan true o false)

1.a

Premisa

a.map(f) es lo mismo que a

Se cumple si y sólo si $f(x) = x$ para todo $x \in a$, es decir, cuando f es la función identidad. Caso especial: si a es vacío, la igualdad se cumple para cualquier f.

1.b

Premisa

a.map(f).map(g) tiene el mismo valor que a.map(g).map(f)

Se cumple si y sólo si $g(f(x)) = f(g(x))$ para todo $x \in a$. Casos triviales: $f = id$ o $g = id$.

1.c

Premisa

Escriba una expresión equivalente pero más simple que a.filter(f).filter(g) asumiendo f y g son predicados (i.e. funciones que retornan true o false)

Equivalente:

```
1        a.filter(x => f(x) && g(x))
```

Listing 1: Equivalente

Ejercicio 2

`Sea un array de k lambdas [f_1,...,f_k]. Se quiere calcular la función g que cumpla g(x) = f_1(f_2(...f_k(x)...)) usando FP. Asuma que para cada f_i el dominio y el codominio son iguales entre sí y e iguales entre todos los i.`

Solución

Para aplicar composición de funciones, haremos uso de **reduceRight**, el cual aplica una función acumuladora recorriendo el arreglo de derecha a izquierda.

```
1        arr.reduceRight((acumulador elemento) => ..., valorInicial)
```

Listing 2: Base de reduceRight

```
// DICHA DEFINICION SERA UNA COMPOSICION DE VARIAS FUNCIONES
const composeAll = (fs) =>
fs.reduceRight(
(g, f) => (x) => f(g(x)),
(x) => x
);

// ALGUNAS LAMBDAS QUE NOS SERVIRAN COMO PRUEBA
const f1 = (x) => x + 1;
const f2 = (x) => x * 2;
const f3 = (x) => x * x;

// CREAMOS EL ARRAY DE FUNCIONES
const fs = [f1, f2, f3];

// APLICAMOS g(x) = f1(f2(f3(x)))
const g = composeAll(fs);

// Pruebas
console.log(`g(2) = ${g(2)}`); // Deberia ser f1(f2(f3(2)))
   = ( (2^2)*2 ) + 1 = 9
console.log(`g(3) = ${g(3)}`); // Deberia ser f1(f2(f3(3)))
   = ( (3^2)*2 ) + 1 = 19
```

Listing 3: g(x) = f1(f2(...fk(x)...)) con FP