

Practica 13

Luisfelipe Rodrigo Mamani Arosquipa

June 2022

Índice

| | |
|----------------|---|
| 1. Ejercicio 1 | 3 |
| 2. Ejercicio 2 | 3 |
| 3. Ejercicio 3 | 3 |
| 4. Ejercicio 4 | 3 |
| 5. Ejercicio 5 | 4 |
| 6. Ejercicio 6 | 4 |

1. Ejercicio 1

Investigue el concepto de first class en Javascript y muestre una pequeña definición seguida ejemplos. (2 puntos)

Una función es de primera clase si se puede almacenar en una variable, pasar como argumento a una función, devolverla desde otra función y tratarla como un objeto de primera clase con sus propias propiedades. También significa que estas funciones de nivel superior admiten todo lo que pueden hacer otros objetos de JavaScript.

Ejemplo:

```
function sumar(x, y) {  
    return x + y;  
}  
let sum = sumar;
```

2. Ejercicio 2

Describa la diferencia entre Currying and Partial Application. Incluya ejemplos. (2 puntos)

La principal diferencia entre Currying y Partial Application, es que en Currying permite llamar una función y dividirla en múltiples llamadas con un argumento por llamada, y en Partial Application son múltiples argumentos para cada llamada. Ejemplo:

```
function sumar(x, y) {  
    return x + y;  
}  
let sum = sumar;  
  
let sumarcurling = (x) =>{  
    return (y) =>{  
        return x+y;  
    }  
}
```

3. Ejercicio 3

Implemente una función que calcule el volumen de un cilindro. Incluya la versión normal y una aplicando Currying. (2 puntos)

```
let cilindrevolume = (r,h) => r*r*h*Math.PI;  
  
let cilindrvolumeclurry = (r) =>{  
    return (h) =>{  
        return r*r*h*Math.PI  
    }  
}  
console.log(cilindrevolume(8,9))  
  
console.log(cilindrvolumeclurry(8)(9))
```

4. Ejercicio 4

Cree una función joinWords que una varios parámetros de tipo string. (3 puntos)

```
result = joinWords ('Hello ') () ;  
console .log ( result ); // Hello  
result = joinWords ('There ')('is ')('no ')('spoon .') () ;  
console .log ( result ); // There is no spoon .  
  
//Cree una función joinWords que una varios parámetros de tipo string.  
function joinWords(...args){
```

```

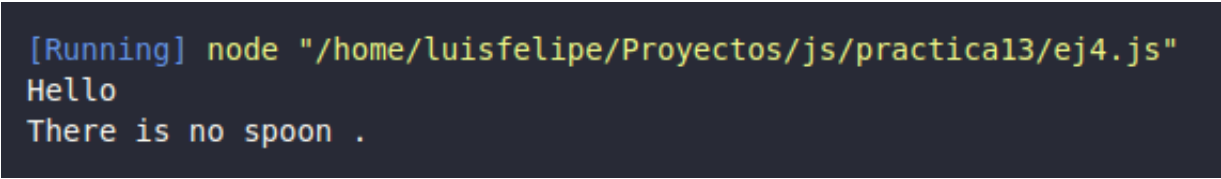
const result = args.join('');
const joinW = (...innerargs)=>{
  if(innerargs.length===0) return result;
  return joinWords(...args, innerargs)
}
return joinW;
}

```

```

result = joinWords ('Hello ') () ;
console .log ( result ); // Hello
result = joinWords ('There ')( 'is ')( 'no ')( 'spoon .' ) () ;
console .log ( result ); // There is no spoon .

```



```

[Running] node "/home/luisfelipe/Proyectos/js/practical13/ej4.js"
Hello
There is no spoon .

```

5. Ejercicio 5

Implemente una función `delayInvoc` que en cada invocación incremente la variable total con el valor enviado como parámetro. (3 puntos)

```

var total = 0;
var delayInvoc = function (a) {
  // your code here
};
delayInvoc (4) (5)
console .log ( total ); //9
delayInvoc (4) (5) (8)
console .log ( total ); // 26

//Implemente una función delayInvoc que en cada invocación incremente la variable total con el
//valor enviado como parámetro.
let total = 0;

function delayInvoc(...args) {
  let result= args.reduce((r,v)=> r+v);

  const sum = (...innerargs)=>{
    if (innerargs.length === 0) return result;
    return delayInvoc(...args, ...innerargs)
  }
  return sum;
};

let total1 = delayInvoc(4)(5)();
console.log ( total1); //9
total1 = delayInvoc(4)(5)(8)();
console.log ( total1 ); // 26

```

6. Ejercicio 6

Implemente una función `curry` que tome como argumento cualquier función `f` y retorne la versión curried de `f`. (4 puntos)

```
|
[Running] node "/home/luisfelipe/Proyectos/js/practical13/tempCodeRunnerFile.js"
9
17

[Done] exited with code=0 in 0.13 seconds
```

```
function abc (a, b, c){
return a+b+c;
}
function curry (f) {
// your code here
}
var curriedAbc = curry ( abc );
console .log ( curriedAbc (2) (3) (4) ); // 9
console .log ( curriedAbc (2 ,3) (4) ); // 9
console .log ( curriedAbc (2) (3 ,4) ); // 9
console .log ( curriedAbc (2 ,3 ,4) ); // 9

//Implemente una funcion curry que tome como argumento cualquier funcion f y retorne la version
//curried de f.
function abc(a, b, c) {
    return a + b + c;
}

function curry (f) {
    return suma = (...args) => {
        if (f.length !== args.length) return suma.bind(null, ...args);
        return f(...args);
    };
}

var curriedAbc = curry(abc);
console.log(curriedAbc(2)(3)(4)); // 9
console.log(curriedAbc(2, 3)(4)); // 9
console.log(curriedAbc(2)(3, 4)); // 9
console.log(curriedAbc(2, 3, 4)); // 9
```

```
[Running] node "/home/luisfelipe/Proyectos/js/practical13/ej6.js"
9
9
9
9

[Done] exited with code=0 in 0.095 seconds
```