Laboratorio lunes 3 de Mayo

Programación Funcional

Objetivo

El objetivo de este laboratorio es introducir con un ejercicio simple el paradigma funcional.

Problema

Vamos a hacer un simple contador que sume o reste. ***Aquí pueden ver un video de la aplicación***.

Es necesario destacar que el paradigma funcional no permite realizar una aplicación pura. Esto se debe a que si o si tendrán que guardar el estado del contador. Esto es aceptable, ya que al desarrollar una aplicación con el paradigma funcional lo que realmente se espera es que la mínima cantidad de código sea impuro, en general se apunta a un 80/20. Este laboratorio es introductorio y por lo tanto será guiado por pasos.

Librerías útiles

Node es asíncrono, por lo que una librería que les puede ser útil para manejar la consola de manera síncrona es ***prompt-sync***. **Notar la opción sigint: true al importar que permite cerrar el programa con Ctrl+C**.

Paso 1: Función para mostrar la vista

Programe una **función pura** que reciba como parámetro el valor actual del contador, *function view(counter)*. Esta función se encarga de **retornar** (**Ojo!** NO IMPRIMIR) lo que se muestra en consola, es decir un string con el formato de lo que se muestra en el video. Esta función es pura, ya que su resultado solo depende de los parámetros que ella recibe y lo retorna (no cambia el estado de una variable externa).

Paso 2: Función para actualizar la vista

Programe una **función pura** que reciba como parámetro el string ingresado por el usuario ("+" ó "-") y el valor actual del contador, *function update(msg, counter)*. Esta función se encarga de **retornar** el valor del contador actualizado, es decir, si *msg* = "+", debería retornar *counter + 1*, si *msg* = "-" debería retornar *counter - 1* y *counter* en caso de cualquier otro caracter. Esta función es pura, ya que su resultado solo depende de los parámetros que ella recibe y lo retorna (no cambia el estado de una variable externa).

Paso 3: Función aplicación

Esta función será el *cerebro* de su aplicación. Es por definición impura, ya que se encarga de llevar el estado actual del contador (impuro), leer el input del usuario(impuro) y llamar a las funciones anteriores en un loop infinito hasta que la aplicación se cierre.

Programe una función que recibe como parámetro el contador, *function app(counter)*. Esta función contiene un ciclo infinito (puede usar un while(true)) en el cual se debe:

1. Obtener el valor de la vista actual, llamando a la función *view(counter)*. Ej. *const currentView = view(counter)*.
2. Limpiar la consola con *console.clear()*. De esta manera se genera el efecto de actualizar la vista.
3. Imprimir la vista actual.
4. Preguntar y guardar el input del usuario en *const msg*.
5. Actualizar el valor del contador con la función *update(msg, counter)*. Ej. *counter = updated(msg, counter)*.
6. Luego el ciclo se repite.

Paso 4: Incialización

Al final de su documento debe colocar el comando *app(0)*, el cual iniciará su aplicación con el contador en 0.

Paso 5: Prueba

Corra su aplicación con el comando *node <nombre de su app>.js* en una consola.

Entrega

Suba su proyecto a Github antes del lunes 10 a las 12:00 PM, para esto:

Debes crear tu aplicación dentro del repositorio previamente clonado de Github Classroom en https://classroom.github.com/a/cmYJeZfZ.

El proceso para entregar es el mismo que el del laboratorio 0: debes realizar un commit, que es una foto al estado actual de tu proyecto y luego subirlo a github, con los siguientes comandos desde una consola:

```
$ git add .
$ git commit -m "tu mensaje"
$ git push origin master
```