Trabajo Nº 1

Lunes 30 - 2pm Todos los grupos
Presentación de lo que consideren más interesante (Zoom) en máximo 10 minutos

Objetivo

El objetivo es desarrollar una solución funcional en JavaScript a un problema de cierta complejidad. Se espera que utilicen el máximo de elementos funcionales posibles. Por ejemplo:

- Combinators
- Iterators y Generators
- La librería Lodash
- Currying y partial evaluation
- Composición y Pipes
- Promesas y funciones Async
- Chaining

Trabajo a Desarrollar

El trabajo es grupal y se espera que cada grupo

- Presente su trabajo en clases 10 minutos por grupo.
- Deje todo su material: Láminas, video disponibilizarlo en Google Docs y el Código compartirlo en GitHub

Evaluación

La evaluación considera

- calidad del ejemplo desarrollado (belleza del código, uso de técnicas) 60%
- efectividad de la presentación (explicaciones claras, didáctica) 40%

Temas Propuestos

Hay dos desafíos propuestos y dividiremos los grupos de la siguiente forma
Grupos 1 al 4 Tema 1
Grupos 5 al 8 Tema 2

1. Transformación de archivo de texto en otro formateado de acuerdo a diferentes especificaciones
    a. Cada frase debe comenzar con *n* espacios en blanco (después de un punto seguido)
    b. Cada párrafo debe estar separado por *n* líneas (después de un punto aparte)
    c. El ancho del texto debe ser a lo más *n* (sin cortar palabras)
    d. Cada párrafo debe tener *n* espacios de sangría
    e. Se ignoran los párrafos que tienen menos de *n* frases
    f. Se ignoran los párrafos que tienen más de *n* frases
    g. Cada frase debe aparecer en párrafo aparte
    h. Solo las primeras *n* frases de cada párrafo

    Estas transformaciones deben poder combinarse o encadenarse de cualquier forma.

2. Transformar un archivo de texto en otro de acuerdo a lo siguiente
    a. Transformar un archivo de texto en formato de markup en un archivo de texto que emule aproximadamente la forma en que se vería ese archivo al verlo con un programa capaz de entender el markup.
    b. Transformar un archivo de texto en formato de markup en un archivo html que se vea lo más parecido posible a la forma en que se vería ese archivo al verlo con un programa capaz de entender el markup.

    Nota: No es necesario que incluyan TODOS los elementos de markup.

Recuerden que, como siempre, es importante que el programa haga lo que se pide, pero en este caso es incluso más importante el **mantenerse dentro del paradigma funcional.**