

FUNCTIONAL JAVASCRIPT



Problemática

En un juego de dardos, los jugadores toman turnos para lanzar 3 dardos a un blanco

Cada jugador parte con un puntaje base de 501 puntos. Debemos calcular el puntaje de cada uno e identificar al ganador, quien es el primero en llegar a 0 puntos



Pipeline

```
const checkResults = (dartsResults) => dartsResults.map(item => cleanResults(item))

const sumResults = (checkedResults) => checkedResults.reduce((x,y)=> x+y)

const pipe = functions => data => functions.reduce((value, func) => func(value), data)

const resultsPipeline = pipe([checkResults, sumResults])
```



Map & Reduce

```
const checkResults = (dartsResults) => dartsResults.map(item => cleanResults(item))  
const sumResults = (checkedResults) => checkedResults.reduce((x,y)=> x+y)  
const pipe = functions => data => functions.reduce((value, func) => func(value), data)  
const resultsPipeline = pipe([checkResults, sumResults])
```



Operador ternario

```
const cleanResults = (item) => (item === 'SB' ? 25 :  
  item === 'DB' ? 50 :  
  item === null ? 0 :  
  item[0] * item[1])
```



Librería LODASH



1 Generar funciones modulares

2 Crear métodos compuestos

3 Más sencillo iterar sobre arrays, objetos y strings



Lodash

```
const playGame = (players) => _ .transform(players,  
  (result, player) => (  
    result[player] = 501))
```

