JavaScript Funcional

Grupo 4
Felipe Campbell
Florencia Valdés
Sebastián Montoya

Args

Args

```
const initGame = (...args) => args.map((name) => ({ name, points: 501
}));
const playGame = (...args) => {
  const players = initGame(...args);
  console.log(welcomeLog(players));
  makeTurn(players);
};

playGame('Pepe', 'Lucía', 'María');
```

FlowRight

Composición de funciones

```
const square = (n) => n * n;
const add = (a, b) => a + b;

const addSquare = _.flowRight([square,
add]);
addSquare(1, 2) // => 9
```

Funciones Interesantes

```
const playerNames = (players) => players.map((player) => player.name);

const stringPlayerNames = (names) => names.reduce((a, b, i, array) => a + (i <
array.length - 1 ? ', ' : ' y ') + b);

const playerNamesLog = _.flowRight([stringPlayerNames, playerNames]);
```

Funciones Interesantes

```
const applyMove = (score, move) => (score === 0 ? 0 : Math.abs(score - move));

const insertMoves = (player, moves) => {
  const mapMovesToValues = moves.map((move) => moveValue(move));
  const currentScore = player.points;
  const finalValue = mapMovesToValues.reduce((score, move) => applyMove(score,
move), currentScore);
  return finalValue;
};
```

¿Por qué utilizarlo?

- Mantener el principio de responsabilidad única de las funciones
- Reutilizar funciones

Bajo Acoplamiento

- Mayor facilidad de testeo para funciones.
- Lectura de código más amigable.
- Mejora en capacidad de mantenibilidad

¡Gracias!