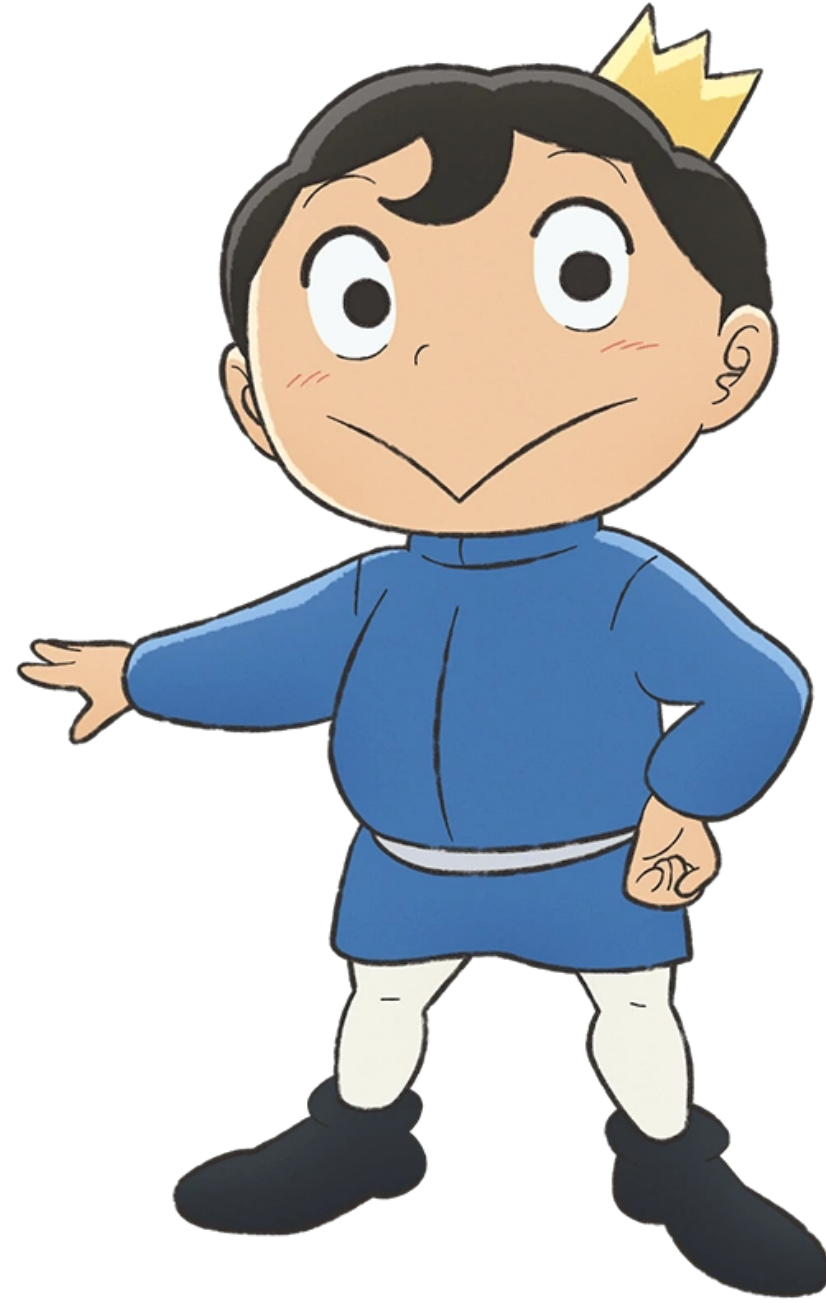


# T2: Reactive JS

Javiera Inostroza, Elías Sabja, Samuel Zúñiga

# PACMAN



# INSTRUCCIONES

El juego consiste en recolectar coronas y esquivar serpientes. Cuando los jugadores hayan recolectado todas las coronas, ganan el juego. Si son atrapados por alguna serpiente, pierden.



**DEMO**

# PROGRAMACIÓN REACTIVA

```
interface Observable {  
    subscribe(observer: Observer): Subscription  
}
```

```
interface Observer {  
    next(v: any): void;  
    error(e: Error): void;  
    complete(): void;  
}
```

```
interface Subscription {  
    unsubscribe(): void;  
}
```

# COMIENZO DEL JUEGO



```
1 document.addEventListener("DOMContentLoaded", async () => {  
2   fillBoard();  
3   drawCrowns(crowns);  
4 });
```

# MOVIMIENTO DE PERSONAJES



```
1 let inputPJ1$ = Rx.Observable.fromEvent(document, 'keydown').scan((lastMove, newMove) => {  
2   const nextMove = getMovement(lastMove, newMove.keyCode);  
3   return legalMove(lastMove, nextMove);  
4 }, { x: getRandomPos(0, board.width - PLAYER_WIDTH), y: getRandomPos(0, board.height - PLAYER_HEIGHT), dir: 0, pj: 'pj1'});
```



```
1 const movePJ1$ = inputPJ1$.subscribe(newPos => movePJ(newPos));
```



```
1  const getMovement = (lastMove, keyCode) => {
2    let pj = 0;
3    if (Object.values(KEYMAP_PJ1).includes(keyCode)) {pj = 'pj1';}
4    else if (Object.values(KEYMAP_PJ2).includes(keyCode)) {pj = 'pj2';}
5
6    if (keyCode == KEYMAP_PJ1.left || keyCode == KEYMAP_PJ2.left){
7      return {x: -VELOCITY, y: 0, dir: 'left', pj};
8    } else if (keyCode == KEYMAP_PJ1.right || keyCode == KEYMAP_PJ2.right){
9      return {x: VELOCITY, y: 0, dir: 'right', pj};
10   } else if (keyCode == KEYMAP_PJ1.up || keyCode == KEYMAP_PJ2.up){
11     return {x: 0, y: -VELOCITY, dir: 'up', pj};
12   } else if (keyCode == KEYMAP_PJ1.down || keyCode == KEYMAP_PJ2.down){
13     return {x: 0, y: +VELOCITY, dir: 'down', pj};
14   }
15   return {x: 0, y: 0, dir: 0, pj};
16 };
```



# ENEMIGO



```
1  const enemiesMove = Rx.Observable.timer(1000, 1000).subscribe(event => {
2    enemies.forEach(enemy => {
3      const newMove = moveEnemy(enemy);
4      enemy.x = newMove.x ;
5      enemy.y = newMove.y;
6      enemy.dir = newMove.dir;
7    });
8  });
```

# CORONAS



```
1  const getCrown$ = new Rx.Observable.create(sub => {  
2    checkCollision$.subscribe((data) => {  
3      const crown = data[0]; const newPos = data[1];  
4      ctx.fillRect(crown.x, crown.y, CROWN_SIZE, CROWN_SIZE);  
5      movePJ(newPos);  
6      sub.next(crowns.length);  
7    });  
8  }).subscribe(subject);
```

# FIN DEL JUEGO

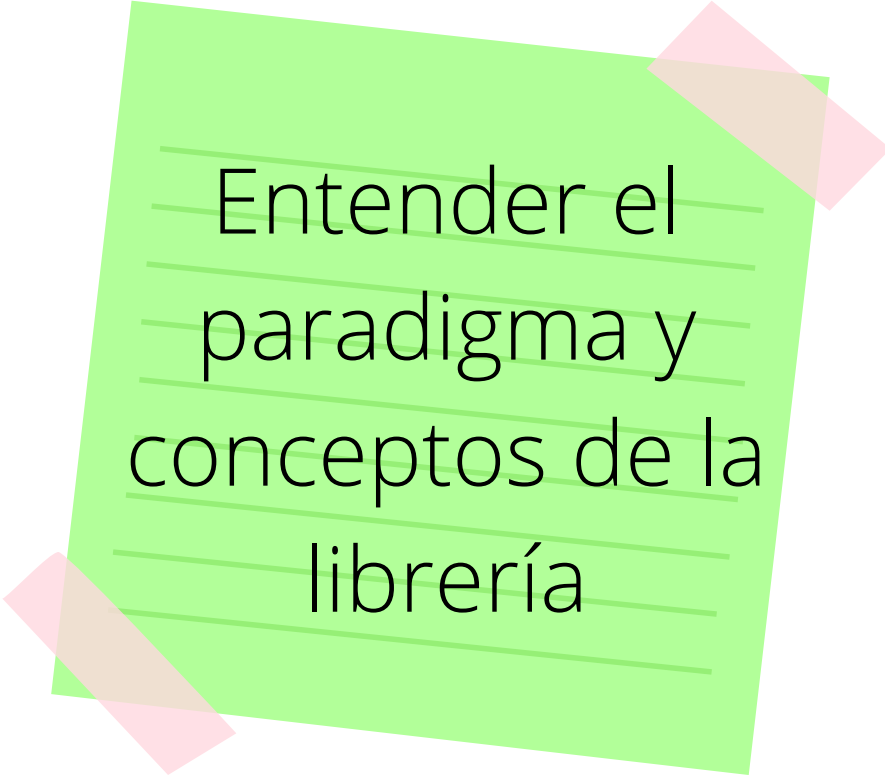


```
1  const checkCollision$ = new Rx.Observable.create(sub => {
2    Rx.Observable.merge(inputPJ1$, inputPJ2$).subscribe((newPos) => {
3      if (collision(enemies, newPos)) gameOver("Game Over", "red");
4
5      crowns.forEach(crown => {
6        if ((newPos.x + PLAYER_WIDTH > crown.x && newPos.x - PLAYER_WIDTH < crown.x) &&
7          (newPos.y + PLAYER_HEIGHT > crown.y && newPos.y - PLAYER_HEIGHT < crown.y))
8        {
9          const id = crowns.indexOf(crown);
10         crowns.splice(id, 1);
11         sub.next([crown, newPos]);
12       }
13     });
14   });
15 });
```




```
1  const winGame = Rx.Observable.combineLatest(subject, score$).subscribe([crownsLeft, score]) => {  
2    if (crownsLeft == 0)  
3    {  
4      gameOver("You won!", "white");  
5    }  
6  });
```

# DIFICULTADES



Entender el  
paradigma y  
conceptos de la  
librería



Aumento de  
complejidad  
en las  
funciones

# CONCLUSIÓN

# T2: Reactive JS

Javiera Inostroza, Elías Sabja, Samuel Zúñiga