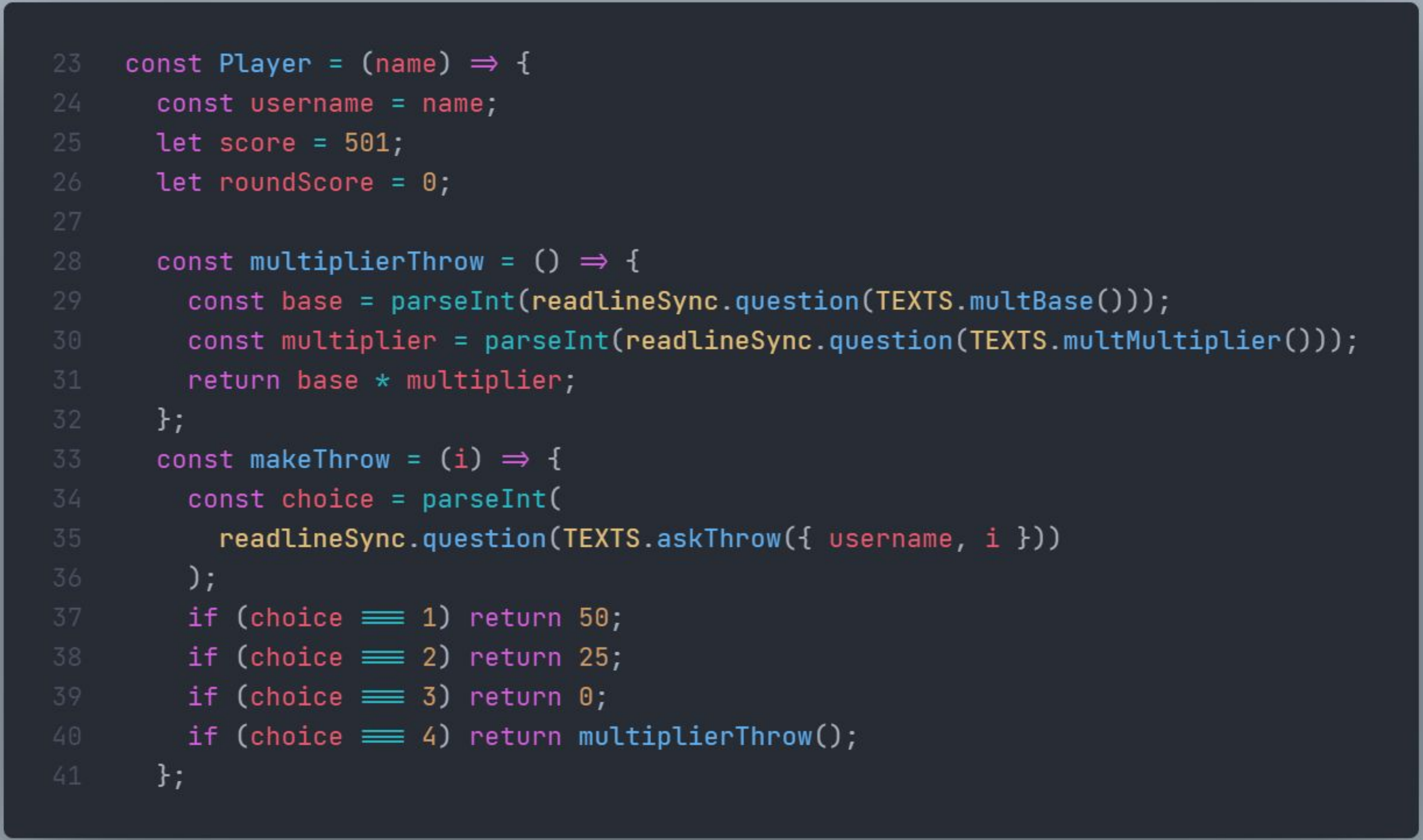
T1 - Dardos

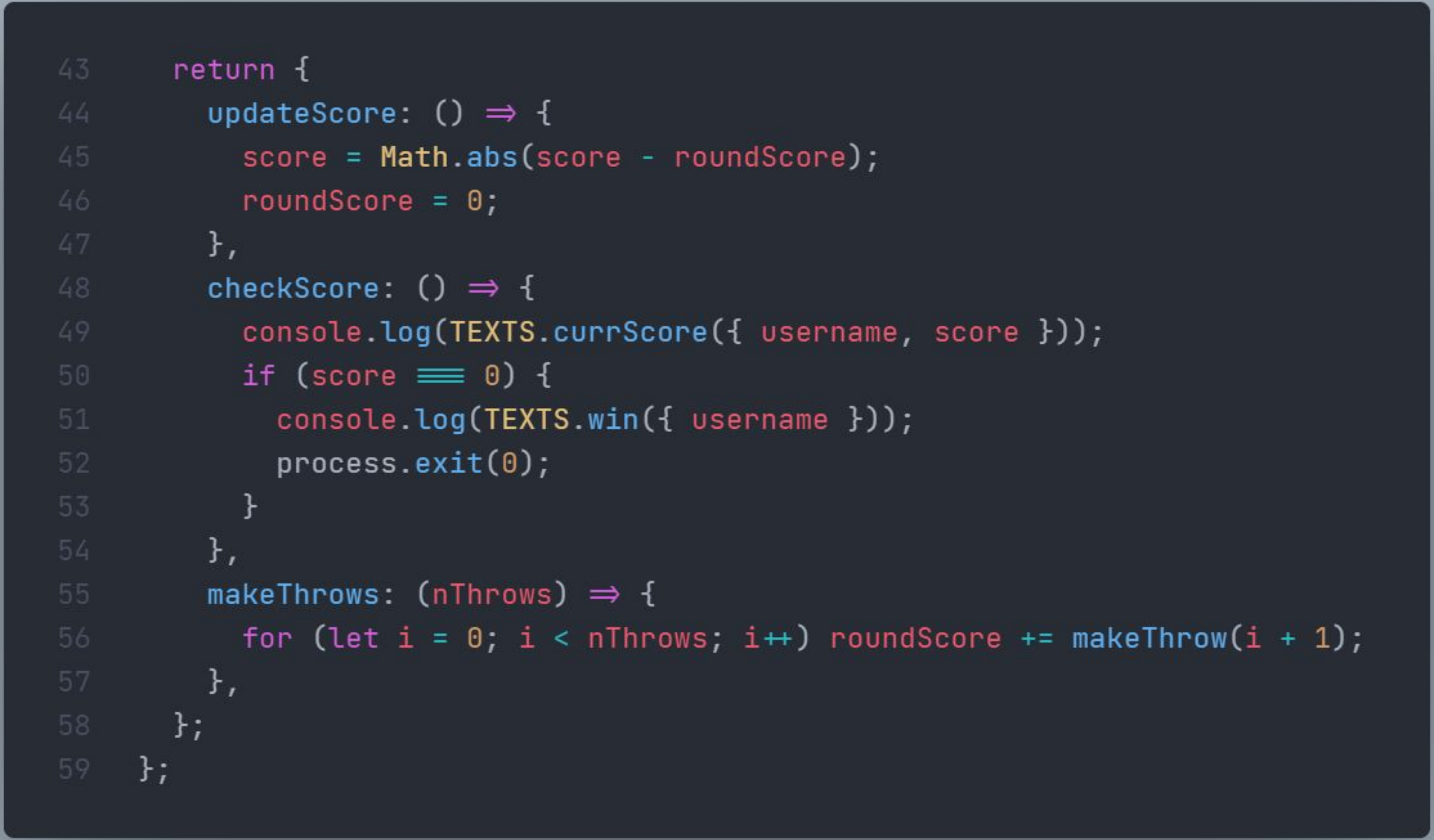
Jorge Araneda, Vicente Calisto,
Tristan Heuer

Primera Iteración

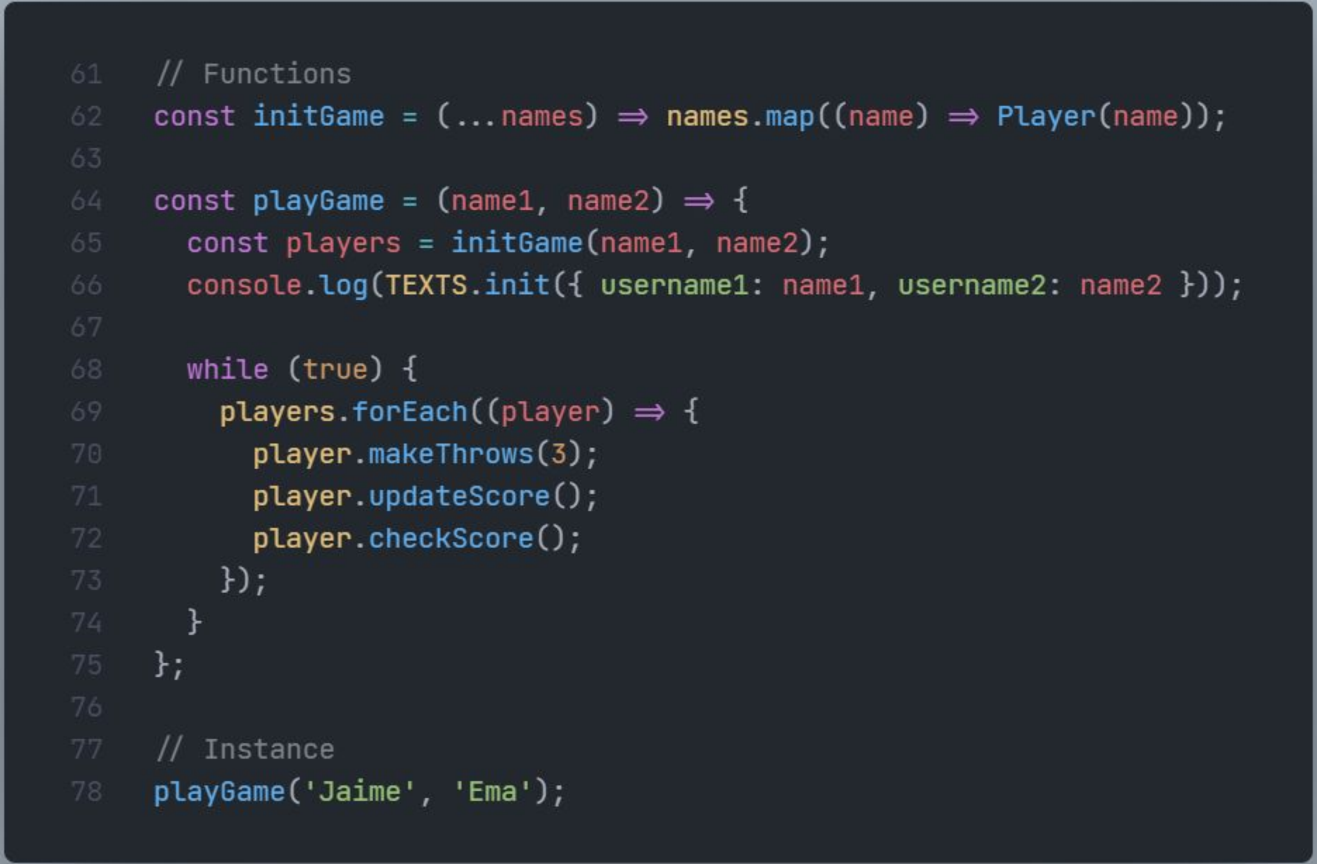
Funcional Encapsulado



```javascript
const Player = (name) => {
  const username = name;
  let score = 501;
  let roundScore = 0;

  const multiplierThrow = () => {
    const base = parseInt(readlineSync.question(TEXTS.multBase()));
    const multiplier = parseInt(readlineSync.question(TEXTS.multMultiplier()));
    return base * multiplier;
  };
  const makeThrow = (i) => {
    const choice = parseInt(
      readlineSync.question(TEXTS.askThrow({ username, i }))
    );
    if (choice === 1) return 50;
    if (choice === 2) return 25;
    if (choice === 3) return 0;
    if (choice === 4) return multiplierThrow();
  };
```



```javascript
  return {
    updateScore: () => {
      score = Math.abs(score - roundScore);
      roundScore = 0;
    },
    checkScore: () => {
      console.log(TEXTS.currScore({ username, score }));
      if (score === 0) {
        console.log(TEXTS.win({ username }));
        process.exit(0);
      }
    },
    makeThrows: (nThrows) => {
      for (let i = 0; i < nThrows; i++) roundScore += makeThrow(i + 1);
    },
  };
};
```

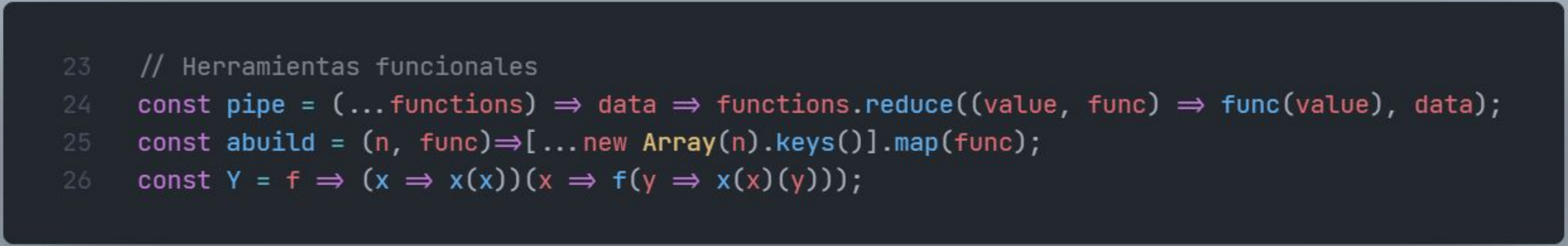


```javascript
// Functions
const initGame = (...names) => names.map((name) => Player(name));

const playGame = (name1, name2) => {
  const players = initGame(name1, name2);
  console.log(TEXTS.init({ username1: name1, username2: name2 }));

  while (true) {
    players.forEach((player) => {
      player.makeThrows(3);
      player.updateScore();
      player.checkScore();
    });
  }
};

// Instance
playGame('Jaime', 'Ema');
```

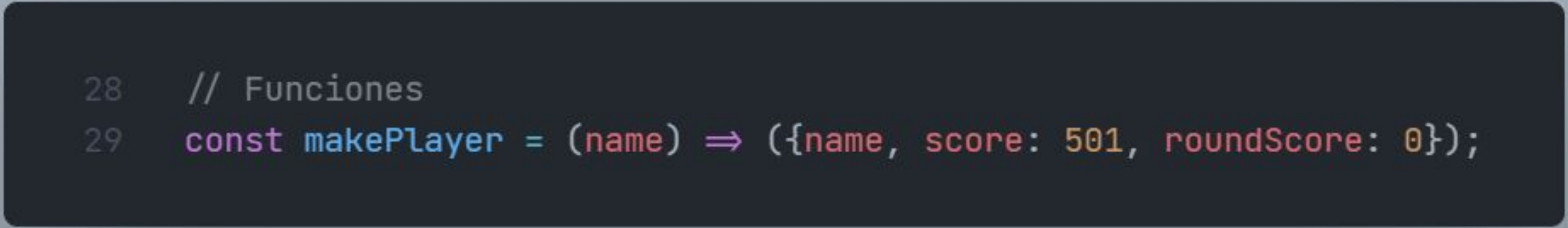
```javascript
// TEXTS
const TEXTS = {
    init: (names) => `Juego inicializado con jugadores ${names.join(' y ')}`,
    askThrow: (name, i) =>
        [
            `Ingrese el lanzamiento N°${i} de ${name}`,
            '1. Double Bull (DB)',
            '2. Single Bull (SB)',
            '3. Null',
            '4. Otro\n',
        ].join('\n'),
    multBase: () => 'Ingrese el puntaje base (1-20)\n',
    multMultiplier: () => 'Ingrese el multiplicador (1-3)\n',
    badChoice: () => 'Opción inválida, por favor inténtelo nuevamente\n',
    currScore: ({ name, score }) => `¡${name} queda con ${score} puntos!`,
    win: ({ name }) => `¡¡¡EL JUGADOR ${name} HA GANADO!!!`,
};
```

Segunda Iteración

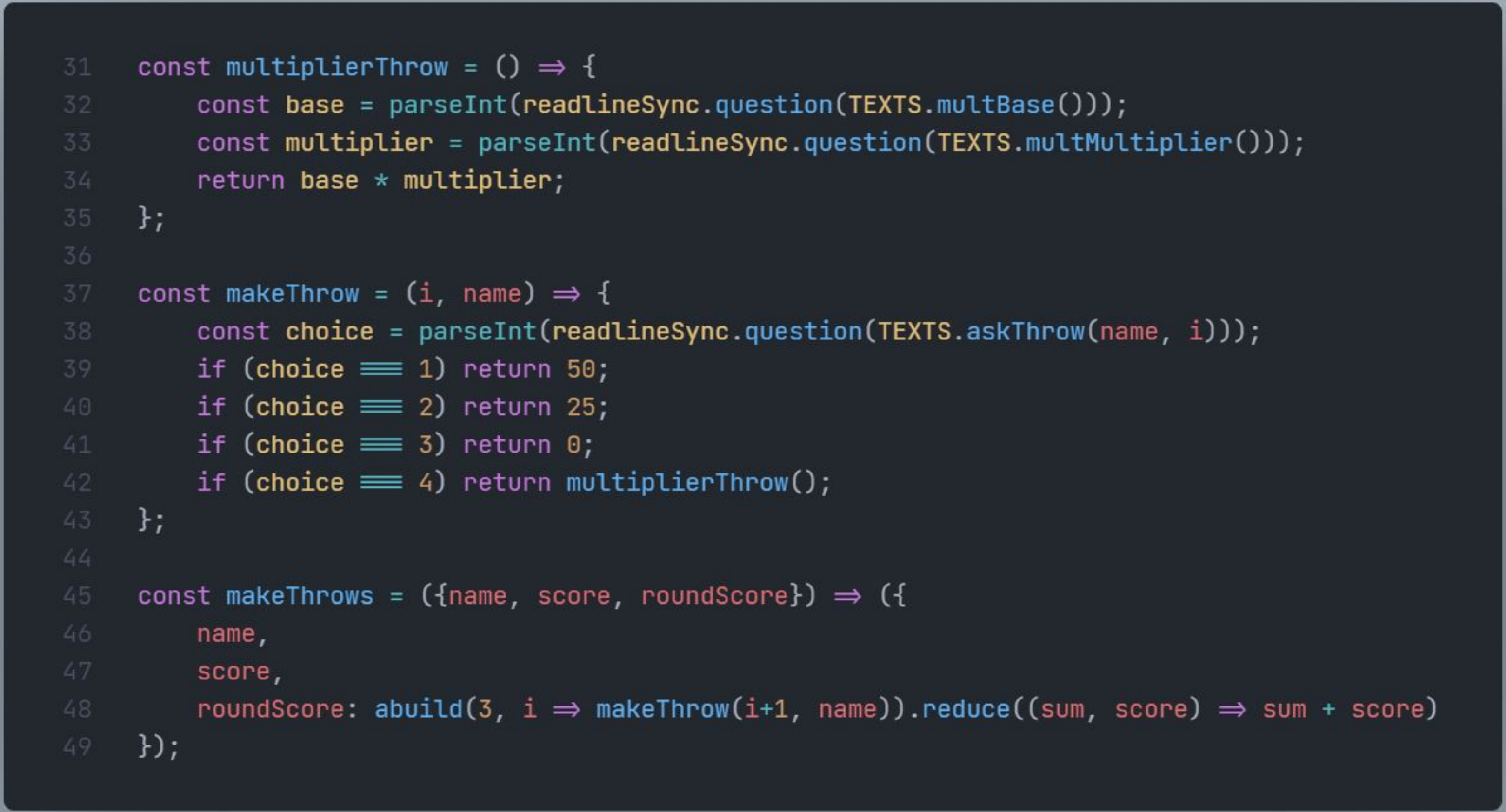
Funcional++



```javascript
// Herramientas funcionales
const pipe = (...functions) => data => functions.reduce((value, func) => func(value), data);
const abuild = (n, func)=>[...new Array(n).keys()].map(func);
const Y = f => (x => x(x))(x => f(y => x(x)(y)));
```

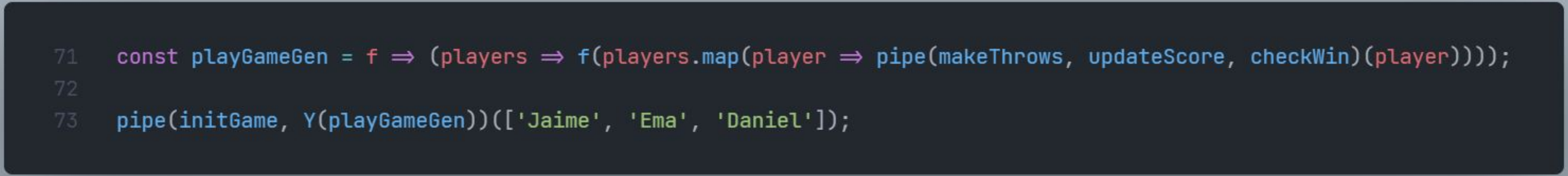


```javascript
// Funciones
const makePlayer = (name) => ({name, score: 501, roundScore: 0});
```



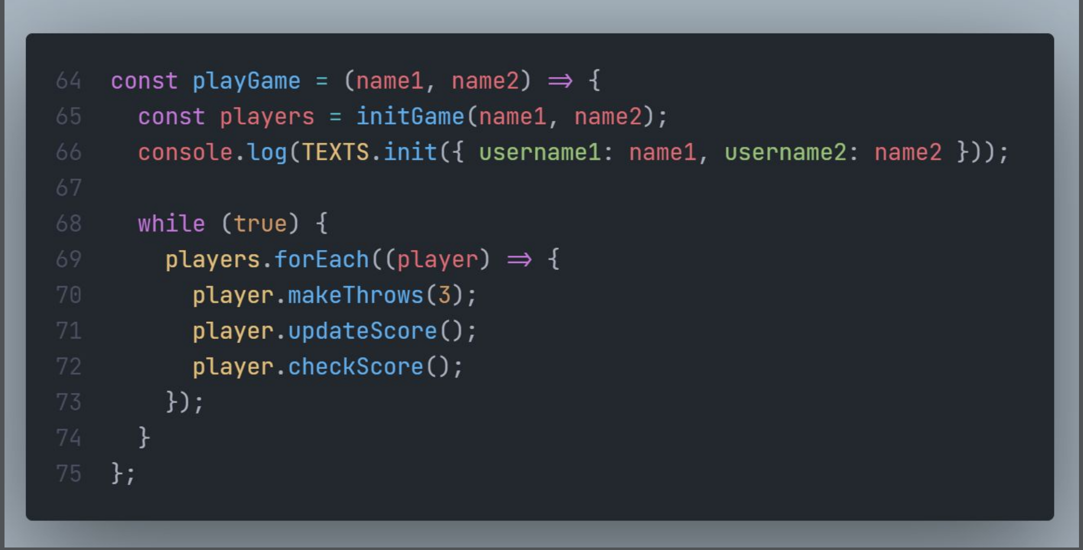
```javascript
const multiplierThrow = () => {
    const base = parseInt(readlineSync.question(TEXTS.multBase()));
    const multiplier = parseInt(readlineSync.question(TEXTS.multMultiplier()));
    return base * multiplier;
};

const makeThrow = (i, name) => {
    const choice = parseInt(readlineSync.question(TEXTS.askThrow(name, i)));
    if (choice === 1) return 50;
    if (choice === 2) return 25;
    if (choice === 3) return 0;
    if (choice === 4) return multiplierThrow();
};

const makeThrows = ({name, score, roundScore}) => ({
    name,
    score,
    roundScore: abuild(3, i => makeThrow(i+1, name)).reduce((sum, score) => sum + score)
});
```

```javascript
const updateScore = ({name, score, roundScore}) => ({
    name,
    score: Math.abs(score - roundScore),
    roundScore: 0
});

const checkWin = (player) => {
    console.log(TEXTS.currScore(player));
    if (player.score === 0) {
        console.log(TEXTS.win(player));
        process.exit(0);
    }
    return player;
};

const initGame = (names) => {
    console.log(TEXTS.init(names));
    return names.map((name) => makePlayer(name));
}
```

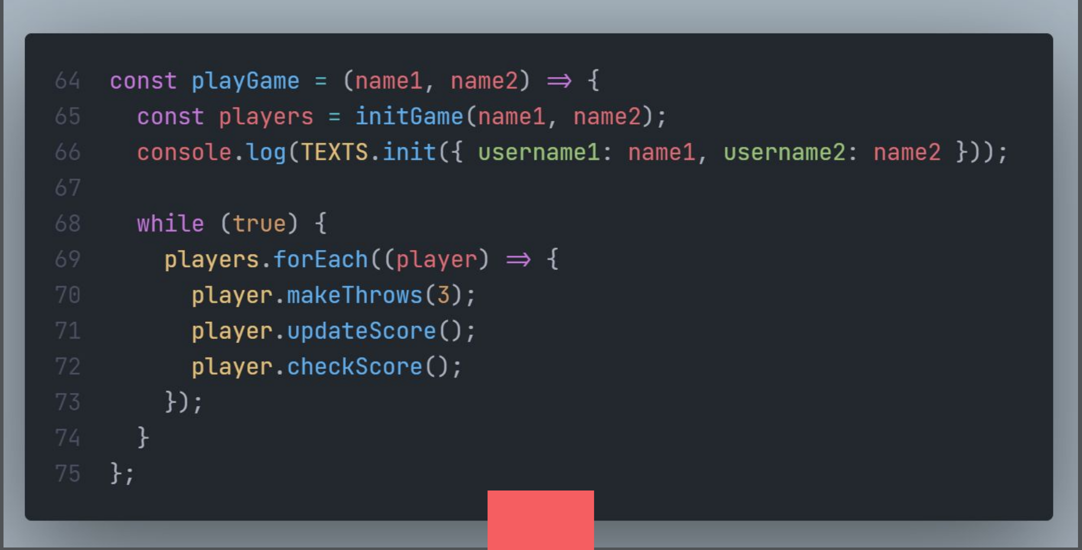


```javascript
const playGameGen = f => (players => f(players.map(player => pipe(makeThrows, updateScore, checkWin)(player))));

pipe(initGame, Y(playGameGen))(['Jaime', 'Ema', 'Daniel']);
```

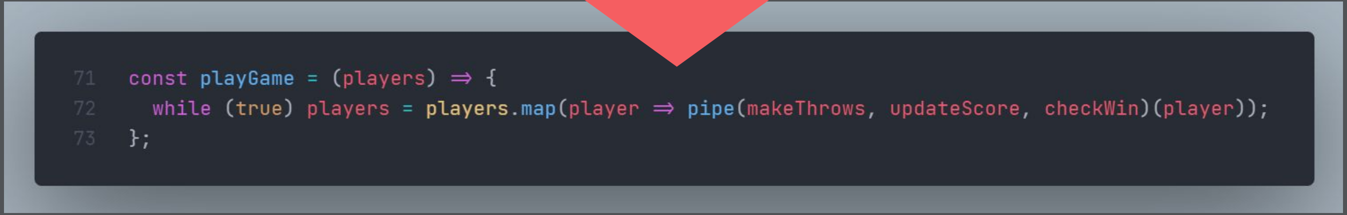
Y Combinator



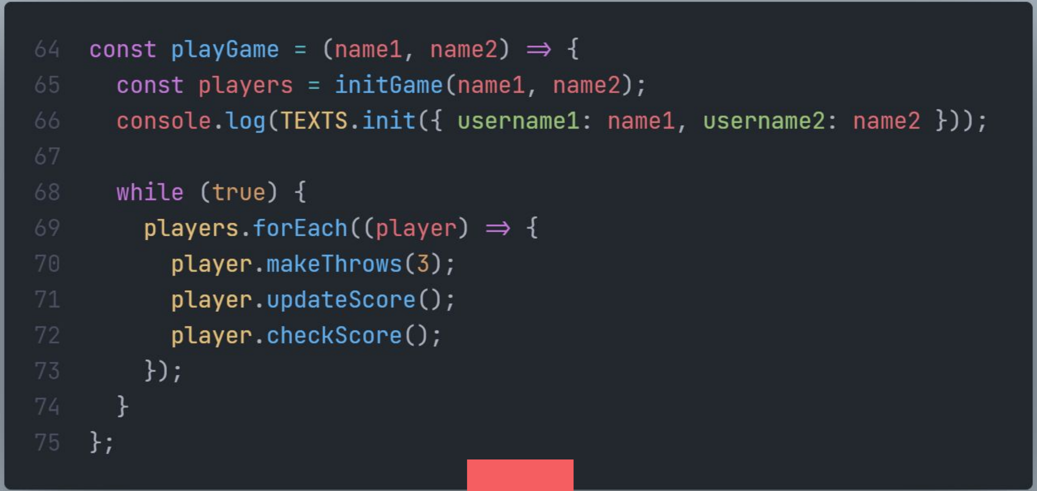
```javascript
const playGame = (name1, name2) => {
  const players = initGame(name1, name2);
  console.log(TEXTS.init({ username1: name1, username2: name2 }));

  while (true) {
    players.forEach((player) => {
      player.makeThrows(3);
      player.updateScore();
      player.checkScore();
    });
  }
};
```



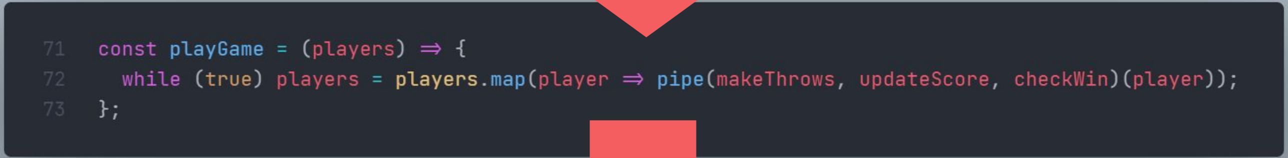
```
const playGame = (name1, name2) => {
  const players = initGame(name1, name2);
  console.log(TEXTS.init({ username1: name1, username2: name2 }));

  while (true) {
    players.forEach((player) => {
      player.makeThrows(3);
      player.updateScore();
      player.checkScore();
    });
  }
};
```



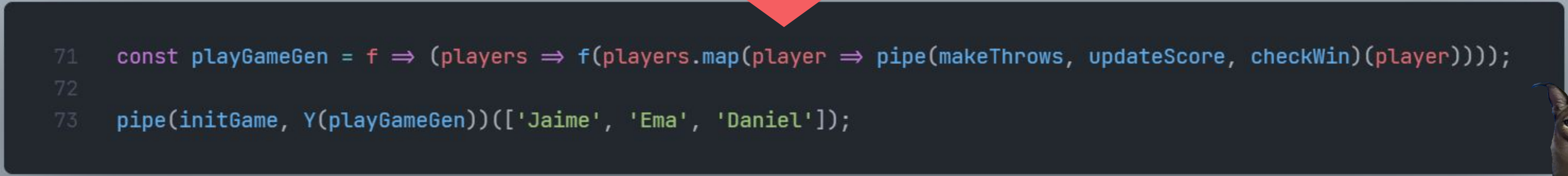
```
const playGame = (players) => {
  while (true) players = players.map(player => pipe(makeThrows, updateScore, checkWin)(player));
};
```



```javascript
const playGame = (name1, name2) => {
  const players = initGame(name1, name2);
  console.log(TEXTS.init({ username1: name1, username2: name2 }));

  while (true) {
    players.forEach((player) => {
      player.makeThrows(3);
      player.updateScore();
      player.checkScore();
    });
  }
};
```



```javascript
const playGame = (players) => {
  while (true) players = players.map(player => pipe(makeThrows, updateScore, checkWin)(player));
};
```



```javascript
const playGameGen = f => (players => f(players.map(player => pipe(makeThrows, updateScore, checkWin)(player))));

pipe(initGame, Y(playGameGen))(['Jaime', 'Ema', 'Daniel']);
```