EAIE

Esports as Accelerator for Social Inclusive Education

**LESSON**

# THE REACTIVE MAZE (REACT HOOKS GAME)

**AUTHOR**

SERGIO ELVIRA PEREZ

CODEARTS SOLUTIONS

# THE REACTIVE MAZE (REACT HOOKS GAME)

The Reactive Maze uses a broken React application as a sequence of puzzles. Learners repair state, side effects and references with Hooks, unlocking each level only when the interface behaves correctly and no new rendering problems are introduced.

### Subject and level

VET front-end learners with JavaScript and basic React component knowledge who are beginning to manage state and side effects using functional components and Hooks.

### Connection to curriculum

The exercise reinforces dynamic user-interface development, state management, asynchronous behaviour, component lifecycle reasoning and debugging. It links React syntax with observable application behaviour and performance considerations.

### Didactic and pedagogic considerations

An escape-room structure encourages hypothesis, testing and reflection. Bugs are selected to expose common misconceptions about state updates and dependency arrays, while teacher clues are deliberately limited so that students practise reading errors and discussing alternatives.

### Time estimate

2 hours

# *THE REACTIVE MAZE (REACT HOOKS GAME)*

**Group size**

15-25 students

**Overall introduction to exercise**

Teams navigate three branches of a React repository, each representing a maze level. A level is complete only when the specified behaviour works and the team can explain why the chosen Hook configuration is correct.

By turning bugs into puzzles, the exercise normalises debugging and helps learners see re-rendering as something they can reason about. The game develops persistence, technical vocabulary and confidence with tools such as the browser console and React Developer Tools.

# THE REACTIVE MAZE (REACT HOOKS GAME)

**Exercise as presented to the students**

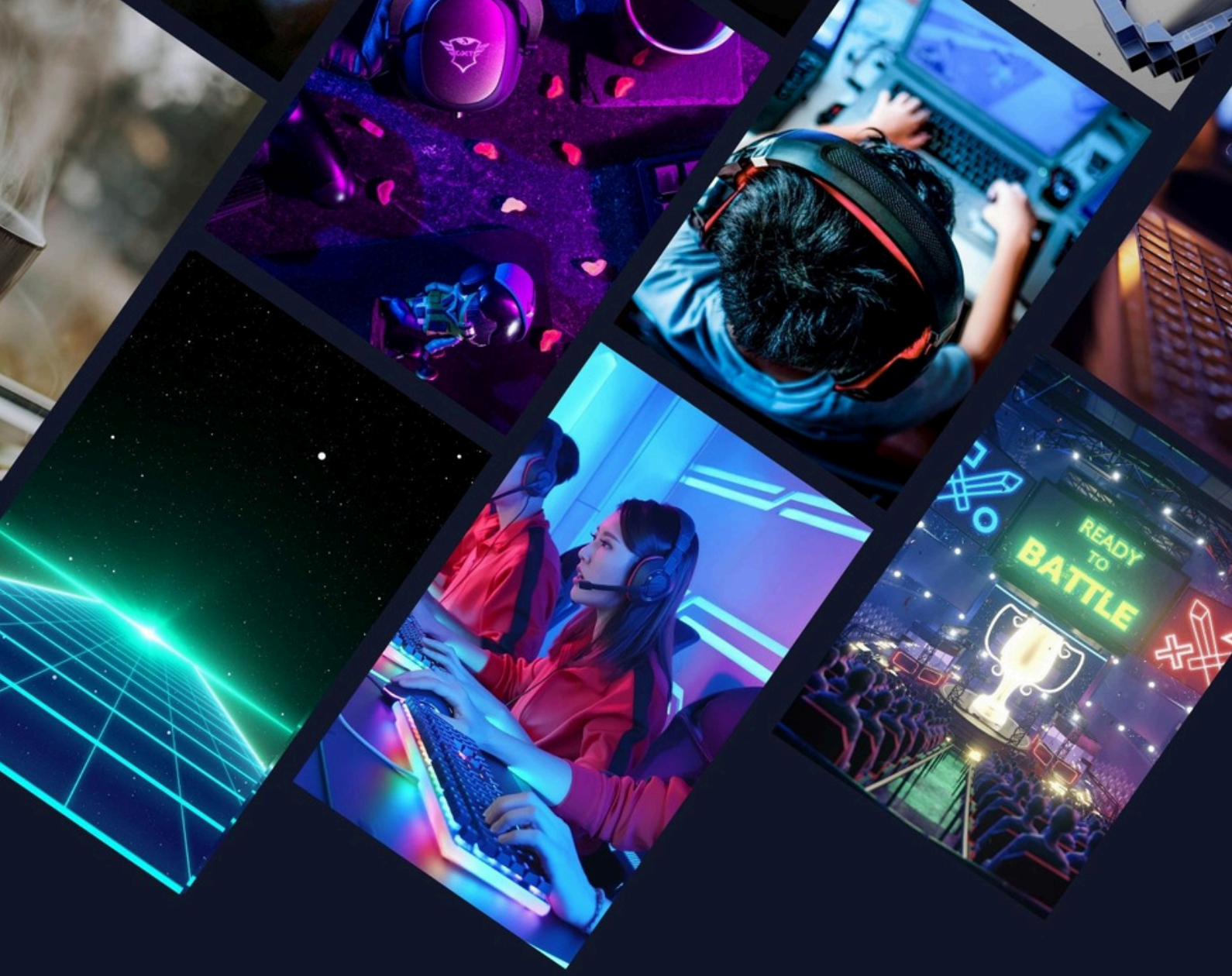
Students entered the session through a short story: a React application had become a digital maze and every broken component represented a locked door. After a concise live demonstration of useState, useEffect and useRef, pairs cloned the starter repository and opened the first level.

Level one contained state errors: buttons updated the wrong values and one counter used stale data. Teams had to restore the intended behaviour and write a one-sentence explanation beside each fix. Level two focused on useEffect, including a missing dependency and an intentional infinite request loop.

The third level introduced a small data-fetching component and a focus interaction requiring useRef. Students used the browser console, Network panel and React Developer Tools to inspect renders and requests. The teacher acted as game master, offering only clue cards such as 'check when this effect should run' rather than giving code.

A team could unlock the Exit Node only after another pair tested the repaired application. If a fix created unnecessary renders or failed after navigation, the reviewers returned it with a clue. This peer-testing stage prompted useful discussions about solutions that appear to work but are not robust.

The closing walkthrough compared different fixes and asked students to predict what would happen if a dependency changed. Learners recorded one Hook rule they now understood better and one debugging technique they would reuse. The maze therefore converted common React frustrations into a structured practice session with visible progress and shared problem-solving.




EASIE
Esports as Accelerator for Social Inclusive Education

# E-SPORT AS ACCELERATOR FOR SOCIAL INCLUSIVE EDUCATION

WWW.EASIE.GG

Co-funded by
the European Union

FUNDED BY THE EUROPEAN UNION. VIEWS AND OPINIONS EXPRESSED ARE HOWEVER THOSE OF THE AUTHOR(S) ONLY AND DO NOT NECESSARILY REFLECT THOSE OF THE EUROPEAN UNION OR THE EUROPEAN EDUCATION AND CULTURE EXECUTIVE AGENCY (EACEA). NEITHER THE EUROPEAN UNION NOR EACEA CAN BE HELD RESPONSIBLE FOR THEM.