All practice for this Part is done into the usePopcorn App – 06

React Hooks

# What are React Hooks?

* Special built-in function that allow us to ‘hook’ into React Internals
  + Create or access **State** from the Fiber Tree
  + Registering **Side Effects** in the Fiber Tree
  + Manual **DOM Selection**
* Always start with the word ‘**use’ (**useState, useEffect, etc)
* We can create **custom Hooks** 
  + The Custom Hooks will also start with ‘use’

# Most Used Hooks

* useState
* useEffect
* useReducer
* useContext

# Less Used Hooks

* useRef
* useCallback
* useMemo
* useTransition
* useDefferedValue
* useLayoutEffect
* useDebugValue
* useImperativeHandle
* useId

# Only for Library Authors

* useSyncExternalStore
* useInsertionEffect

# The Rules of Hooks

## Can only be called at the top level

* Cannot be called inside conditionals (if’s), loops, nested components or after an early return
* This is necessary to ensure that hooks are always called **IN THE SAME ORDER**

## Hooks can only be called from React Functions

* Only call hooks inside **function components** or a custom hook

# useState Summary

* Create state:
  + Simple 🡺 useState(23)
  + Callback Function 🡺 useState(()=>someFunction())
* Update state:
  + Simple 🡺 setCount(100)
  + Based on current State 🡺 setCount((prevState)=> prevState+1)
    - Never mutate objects or arrays
    - Create a new array

# useRef hook

* We use this hook to create a **REF**
* A **REF** is a reference
  + It’s like a box where we can put any data that will be preserved between renders
* React will give us an object with the **.current**
* **We can write any** data and read from it
* The **.current property**  is MUTABLE
* The .current property is PERSISTENT DURING RENDERS

### Usecase

* Create a variable that stays the same between renders
  + Previous state
  + setTimeout ID
* Selecting and storind **DOM ELEMENTs**
* The data is mutable inside a useEffect HOOK
* **DO NOT mutate data** in the render logic
* A screenshot of a computer

  Description automatically generated
* Changing the REF will **NOT RERENDER THE COMPONENT**

### useRef structure

1. We define the variable that will store the ref
   1. Const element = useRef(null)
2. We define the element that we want to be stored in the ref (in JSX)
   1. Ref={element}
3. To use the Ref
   1. In a useEffect (for every render), we set the focus of the element

A screen shot of a computer program

Description automatically generated

## useRef as a STORAGE VARIABLE

* That will not cause the app to RERENDER