useEffect Best Practices

# Every state variable, prop used inside the effect MUST be included in the dependency array

* Any context value must also be included
* All reactive values
  + State
  + Props
  + Context
  + Any other value that references a reactive value
* **NEVER IGNORE** the ‘exhaustive-deps’ ESLint rule
* Do not use OBJECTS or ARRAYS as dependencies
  + These are recreated on each render
  + So the Effect will run at every render

# Removing Unnecessary dependencies

* Move the function into the Effect
* Memoize the function with useCallback
* If the function doesn’t reference any reactive values, move it out of the component
* Don’t include the entire object, only the properties you need (primitive values)
* If there are multiple related reactive values as dependencies, try using a reducer
* setState and Dispatch 🡺 NO NEED TO INCLUDE THEM as a dependency
  + React Guarantees these are Stable

# When not to use an effect

* Effects should be used **as a last resort**
  + When no other solution makes sense
  + React calls them ‘escape hatch’

## Responding to a user event

* An Event handler function should be used instead

## Fetching data on a component mount

* It’s ok to use it in small apps
* But in real-world apps, a library like React Query should be used

## Syncronizing state changes with one another

* Try to use derived state and Event Handlers

# Closures and Stale Closures in Effects

### Closure = the variables and their values at the time a effect or a function is called

### The reason for **DEPENDENCY ARRAY** is to avoid using STALE CLOSURES. With the dependency array, we make sure the effect is always running with the latest values for the variables.