ACADEMIND

What is an “Effect” (or a “Side Effect”)?

Main Job: Render UI & React to User Input

Evaluate & Render JSX
Manage State & Props
React to (User) Events & Input
Re-evaluate Component upon State & Prop Changes

This all is “baked into” React via the “tools” and features covered in this course (i.e. useState() Hook, Props etc).

Side Effects: Anything Else

Store Data in Browser Storage
Send Http Requests to Backend Servers
Set & Manage Timers
...

These tasks **must happen outside of the normal component evaluation** and render cycle – especially since they might block/ delay rendering (e.g. Http requests)

ACADEMIND

Handling Side Effects with the useEffect() Hook

```
useEffect(() => { … }, [ dependencies ]);
```

A function that should be executed AFTER every component evaluation IF the specified dependencies changed

Dependencies of this effect – the function only runs if the dependencies changed

Your side effect code goes into this function.

Specify your dependencies of your function here

ACADEMIND

Introducing useReducer() for State Management

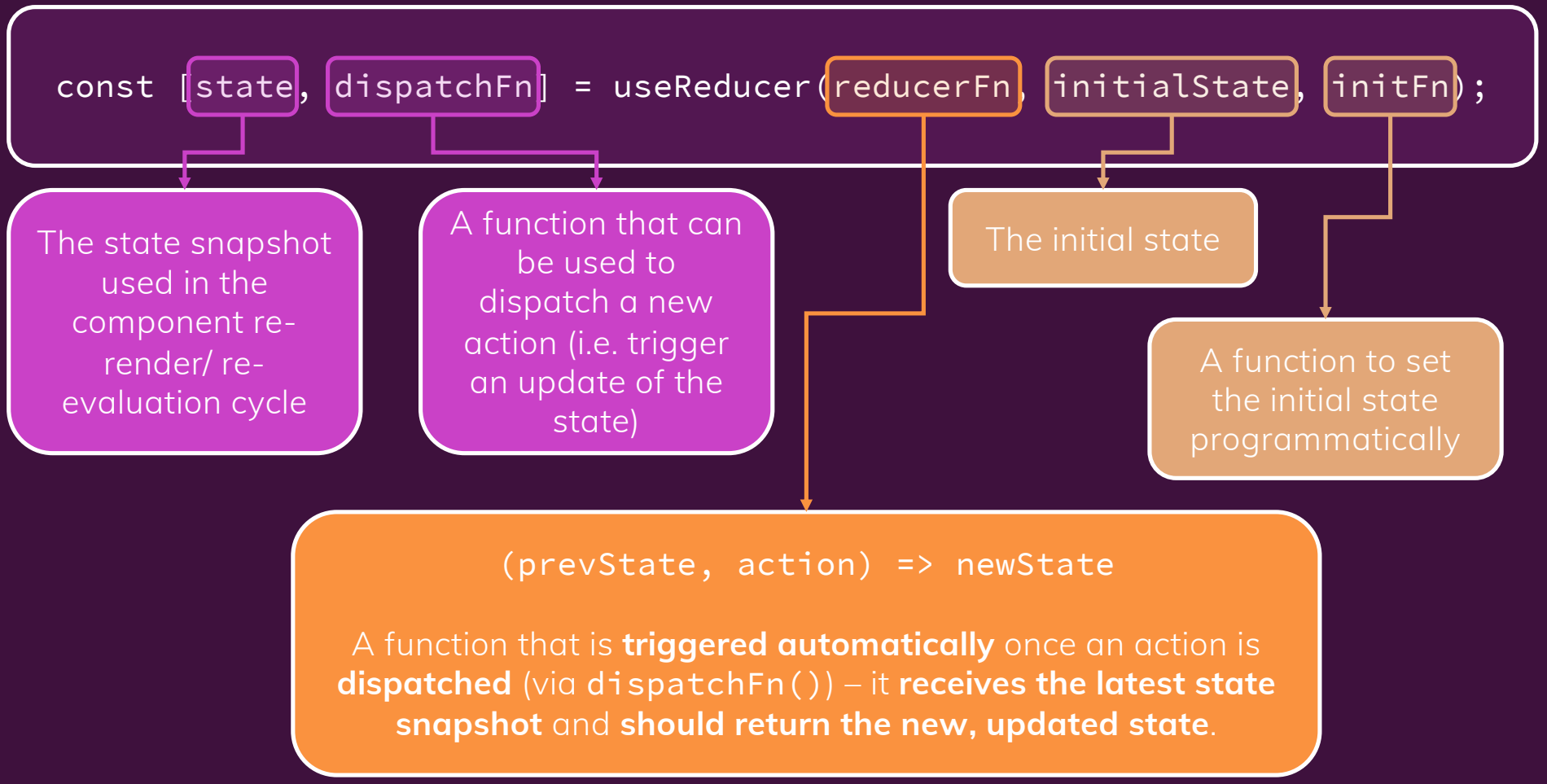
Sometimes, you have **more complex state** – for example if it got **multiple states**, **multiple ways of changing** it or **dependencies** to other states

useState() then often **becomes hard or error-prone to use** – it's easy to write bad, inefficient or buggy code in such scenarios

useReducer() can be used as a **replacement** for useState() if you need "**more powerful state management**"

ACADEMIND

Understanding useReducer()



ACADEMIND

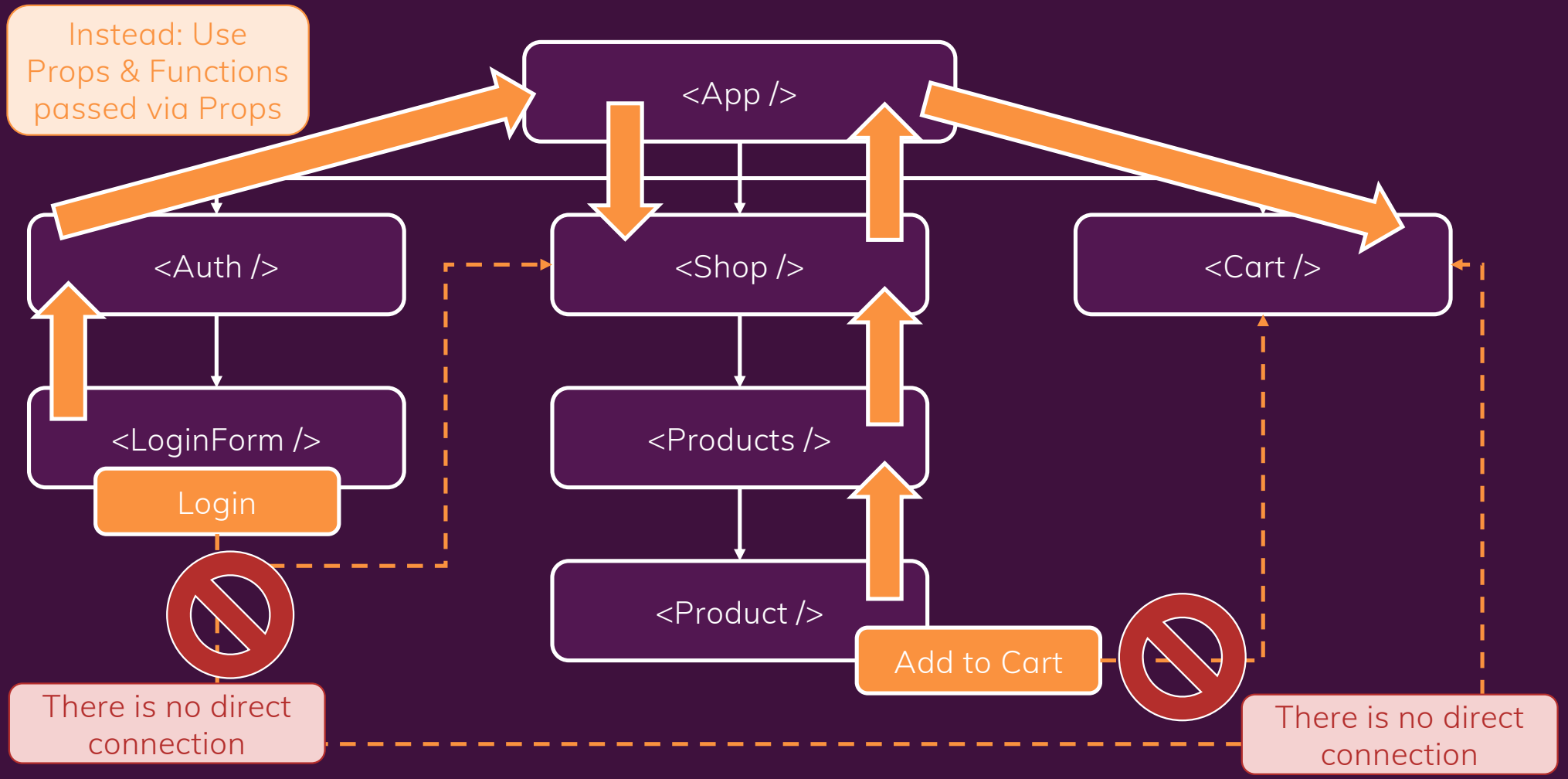
useState() vs useReducer()

Generally, you'll know when you need useReducer() (➔ when using useState() becomes cumbersome or you're getting a lot of bugs/ unintended behaviors)

useState()	useReducer()
The main state management "tool"	Great if you need "more power"
Great for independent pieces of state/ data	Should be considered if you have related pieces of state/ data
Great if state updates are easy and limited to a few kinds of updates	Can be helpful if you have more complex state updates

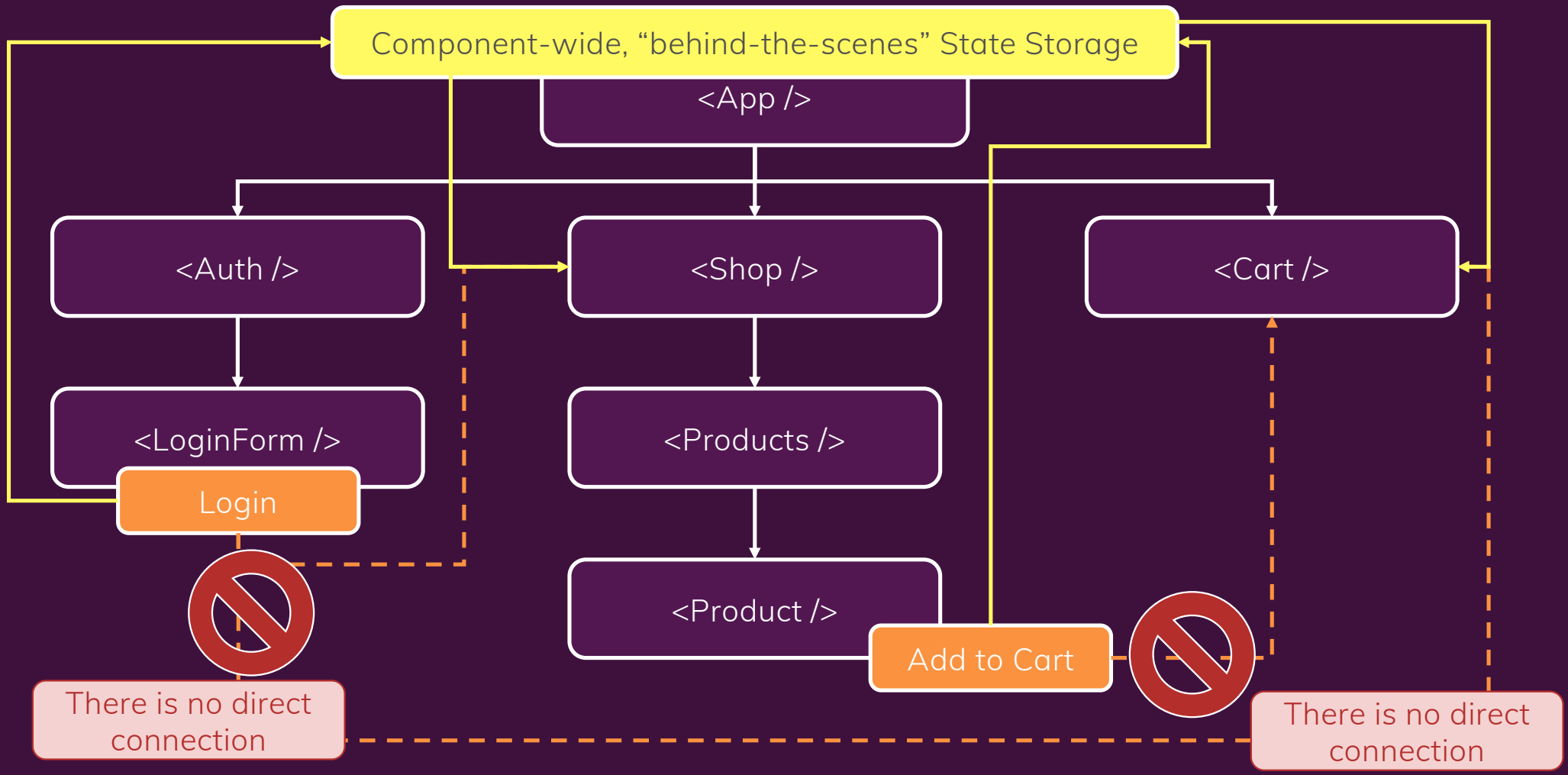
ACADEMIND

Component Trees & Component Dependencies



ACADEMIND

Context to the Rescue!



ACADEMIND

Context Limitations

React Context is **NOT optimized** for high frequency changes!

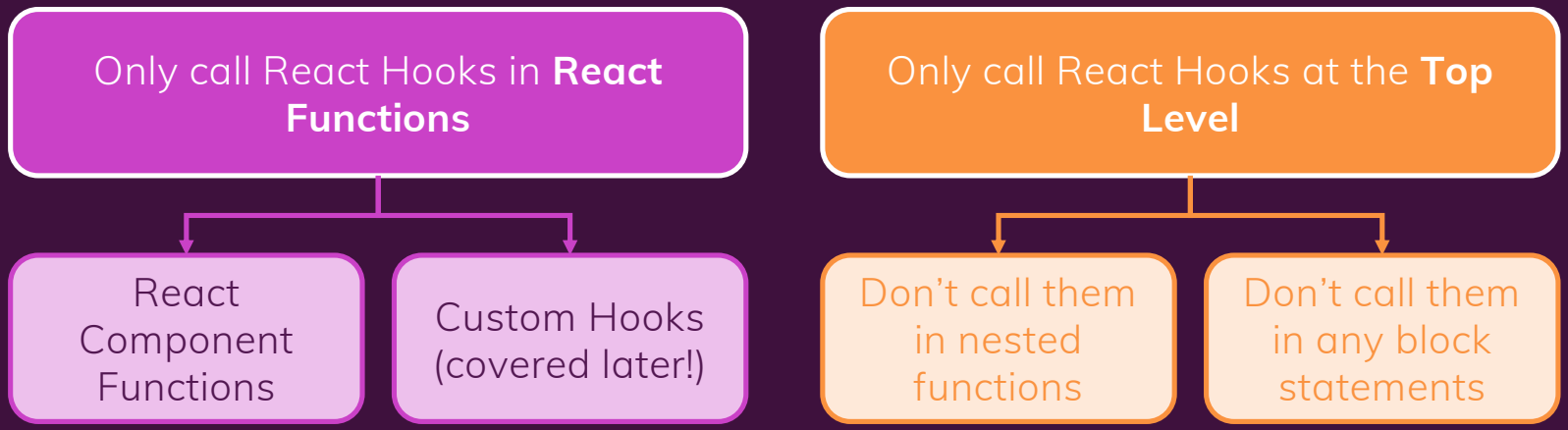
We'll explore a better tool for that, later

React Context also **shouldn't be used to replace ALL** component communications and props

Component should still be configurable via props and short "prop chains" might not need any replacement

ACADEMIND

Rules of Hooks



+ extra, unofficial Rule for **useEffect()**: ALWAYS add everything you refer to inside of useEffect() as a dependency!