Effects, Reducers & Context

Advanced, yet super-important Features!

Module Content

Working with (Side) Effects

Managing more Complex State with Reducers

Managing App-Wide or Component-Wide State with Context

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
...

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

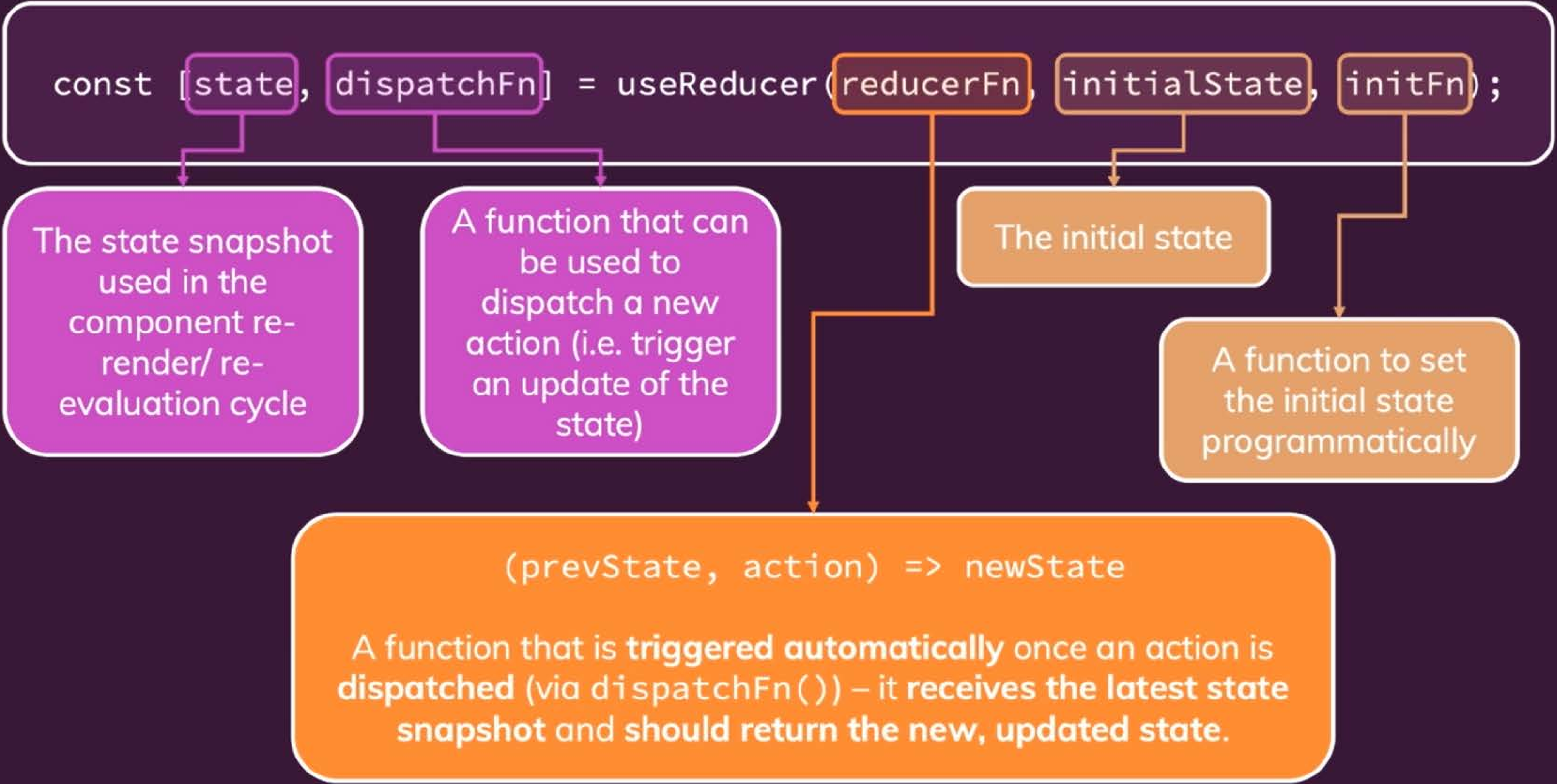
Introducing useReducer() for State Management

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"**

Understanding useReducer()



```
const [state, dispatchFn] = useReducer(reducerFn, initialState, initFn);
```

state: The state snapshot used in the component re-render/ re-evaluation cycle

dispatchFn: A function that can be used to dispatch a new action (i.e. trigger an update of the state)

initialState: The initial state

initFn: A function to set the initial state programmatically

reducerFn:

```
(prevState, action) => newState
```

A function that is **triggered automatically** once an action is **dispatched** (via `dispatchFn()`) – it **receives the latest state snapshot** and **should return the new, updated state**.

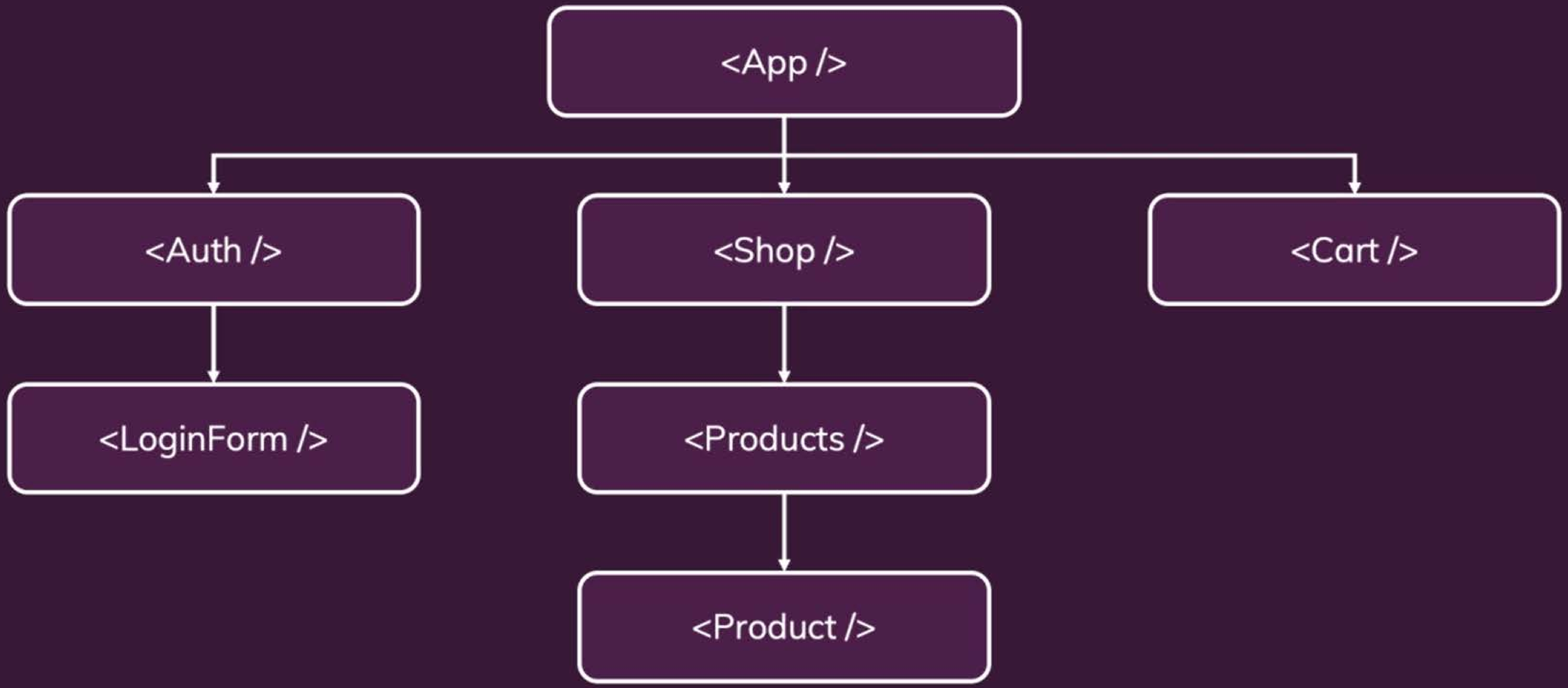
ACADE MIND

useState() vs useReducer()

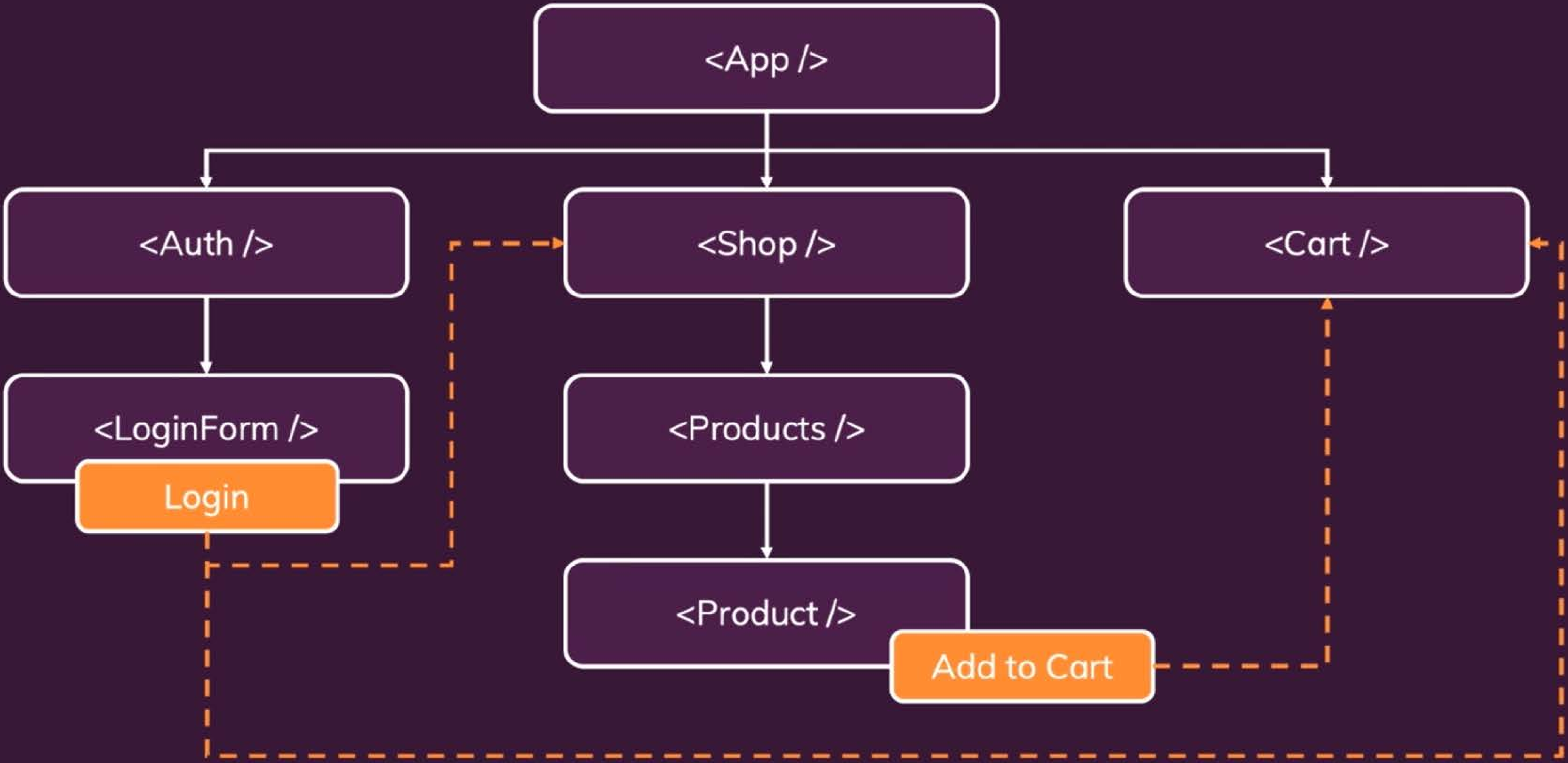
Generally, you'll know when you need useReducer() (➔ when using useState() becomes cumbersome or you're getting a lot of bugs/ unintended behaviors)

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

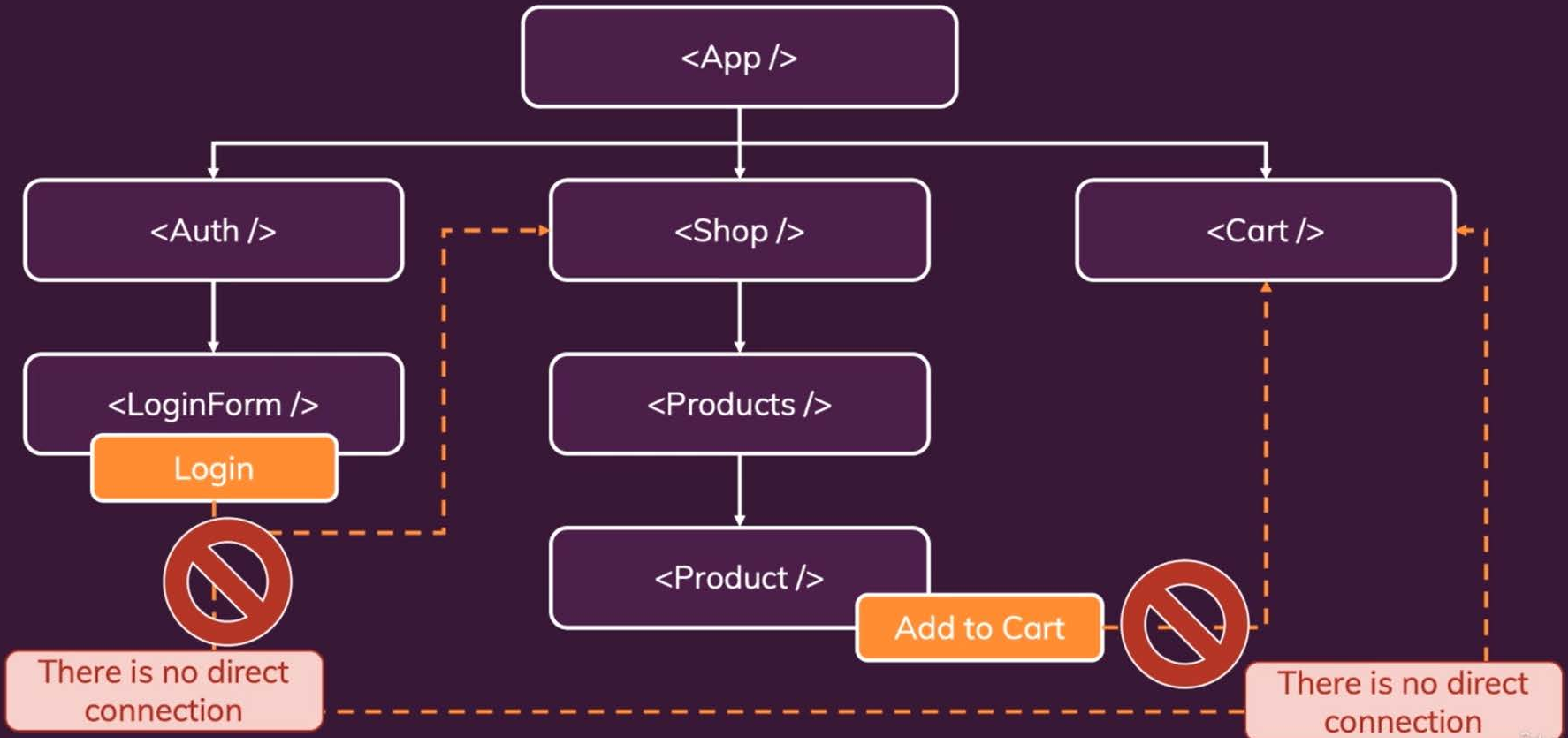
Component Trees & Component Dependencies



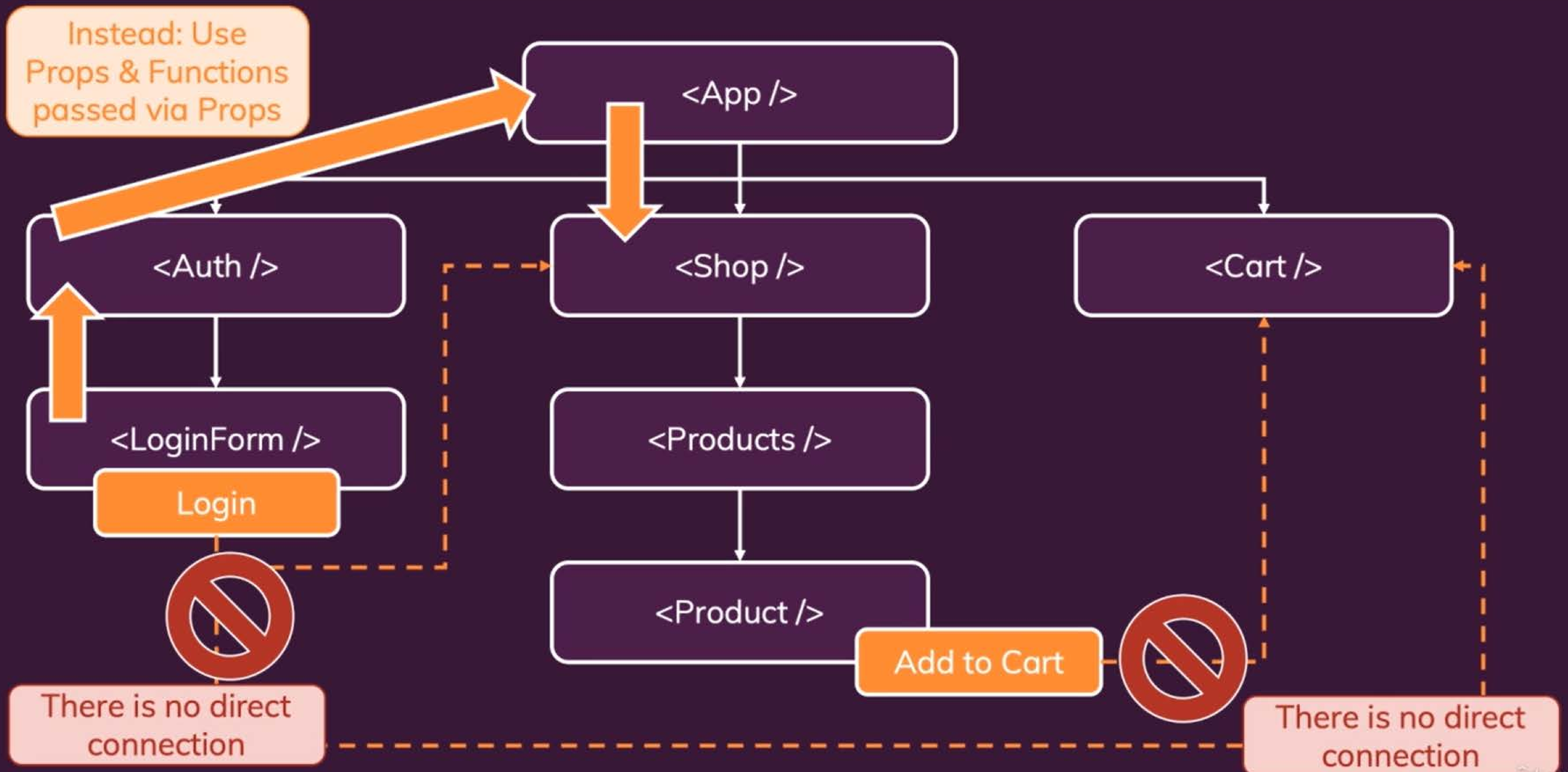
Component Trees & Component Dependencies



Component Trees & Component Dependencies

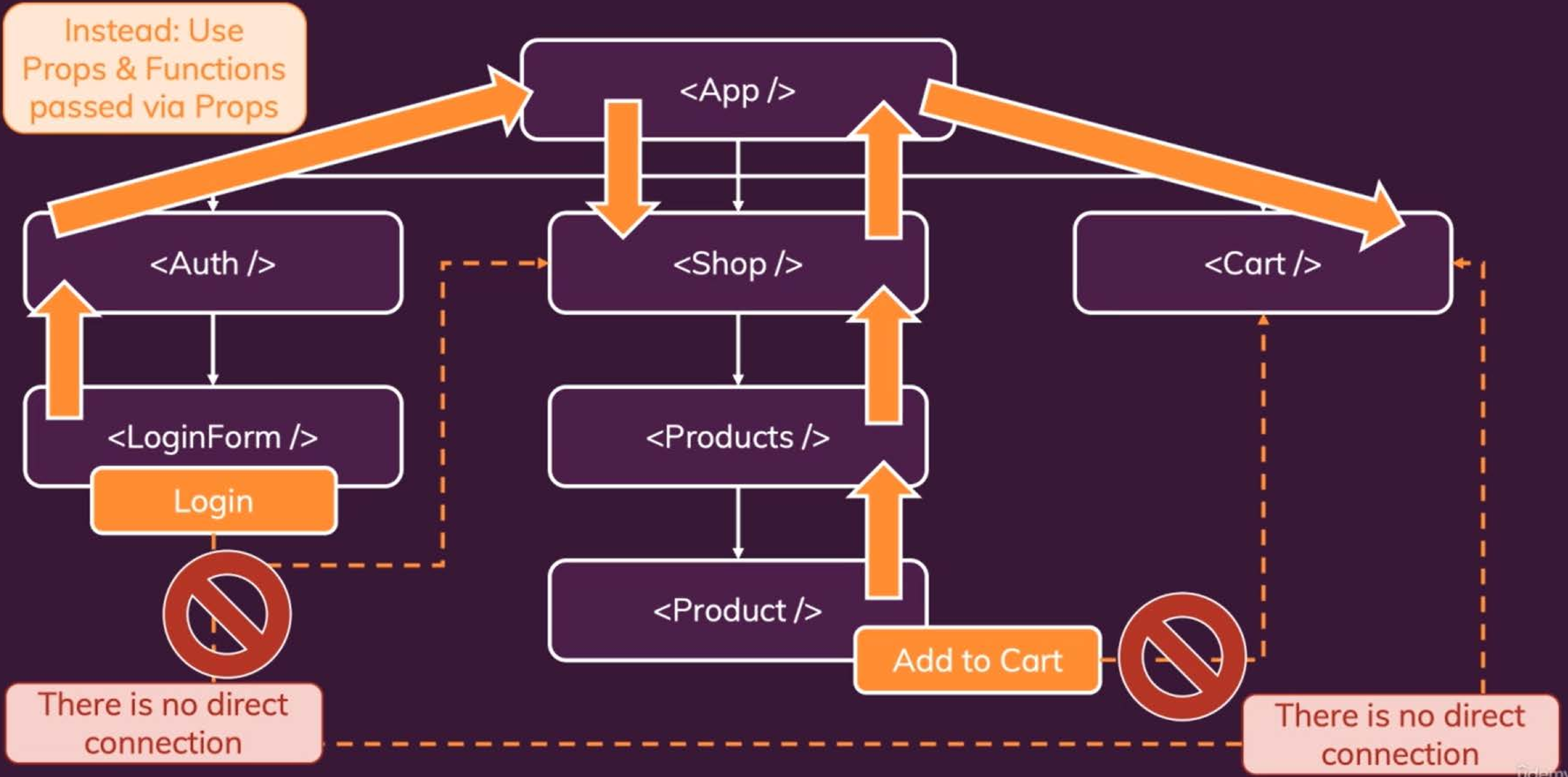


Component Trees & Component Dependencies



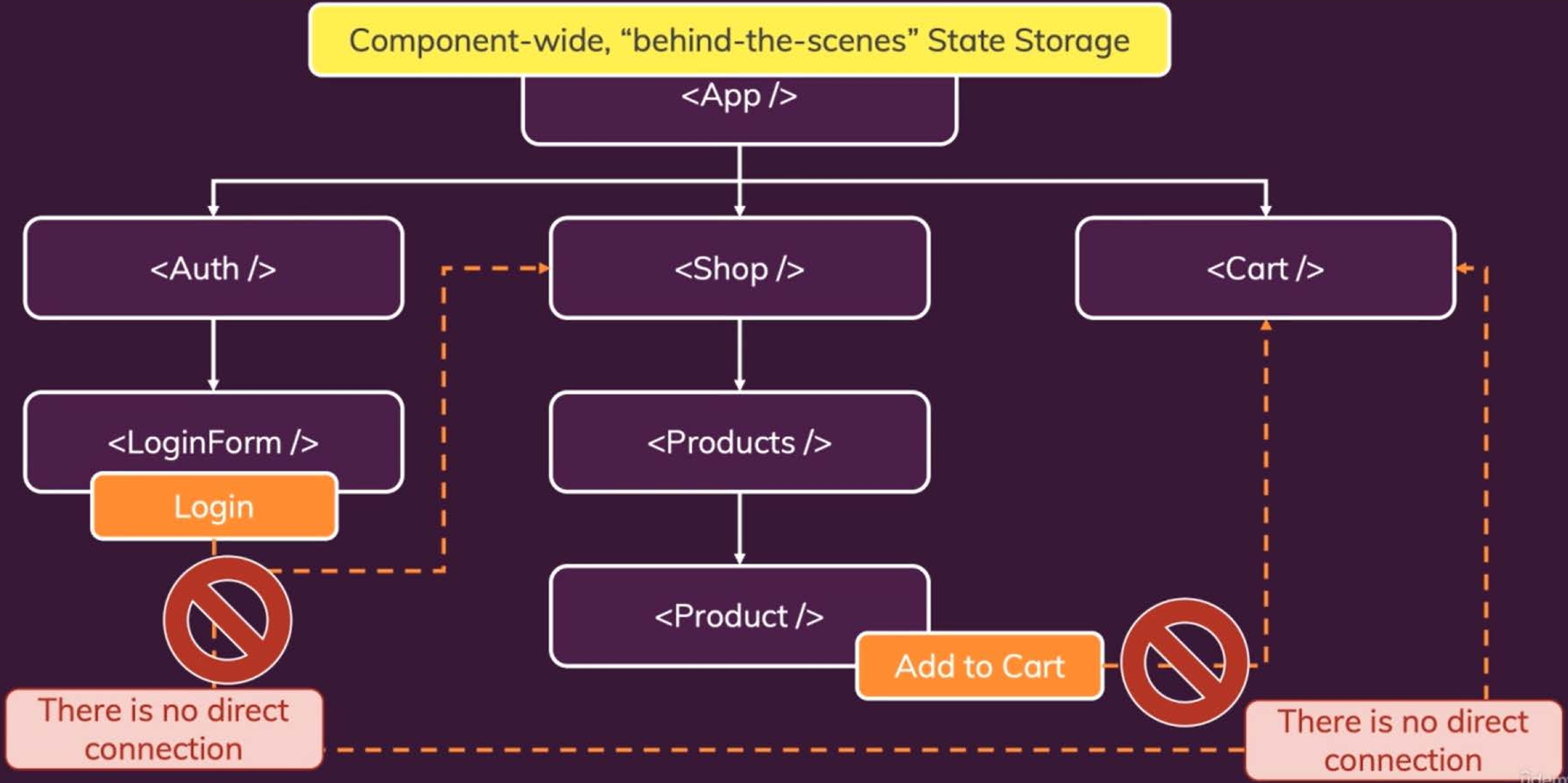
Instead: Use Props & Functions passed via Props
<App />
<Auth />
<Shop />
<Cart />
<LoginForm />
Login
<Products />
<Product />
Add to Cart
There is no direct connection
There is no direct connection

Component Trees & Component Dependencies


Instead: Use Props & Functions passed via Props
<App />
<Auth />
<Shop />
<Cart />
<LoginForm />
<Products />
Login
<Product />
Add to Cart
There is no direct connection
There is no direct connection

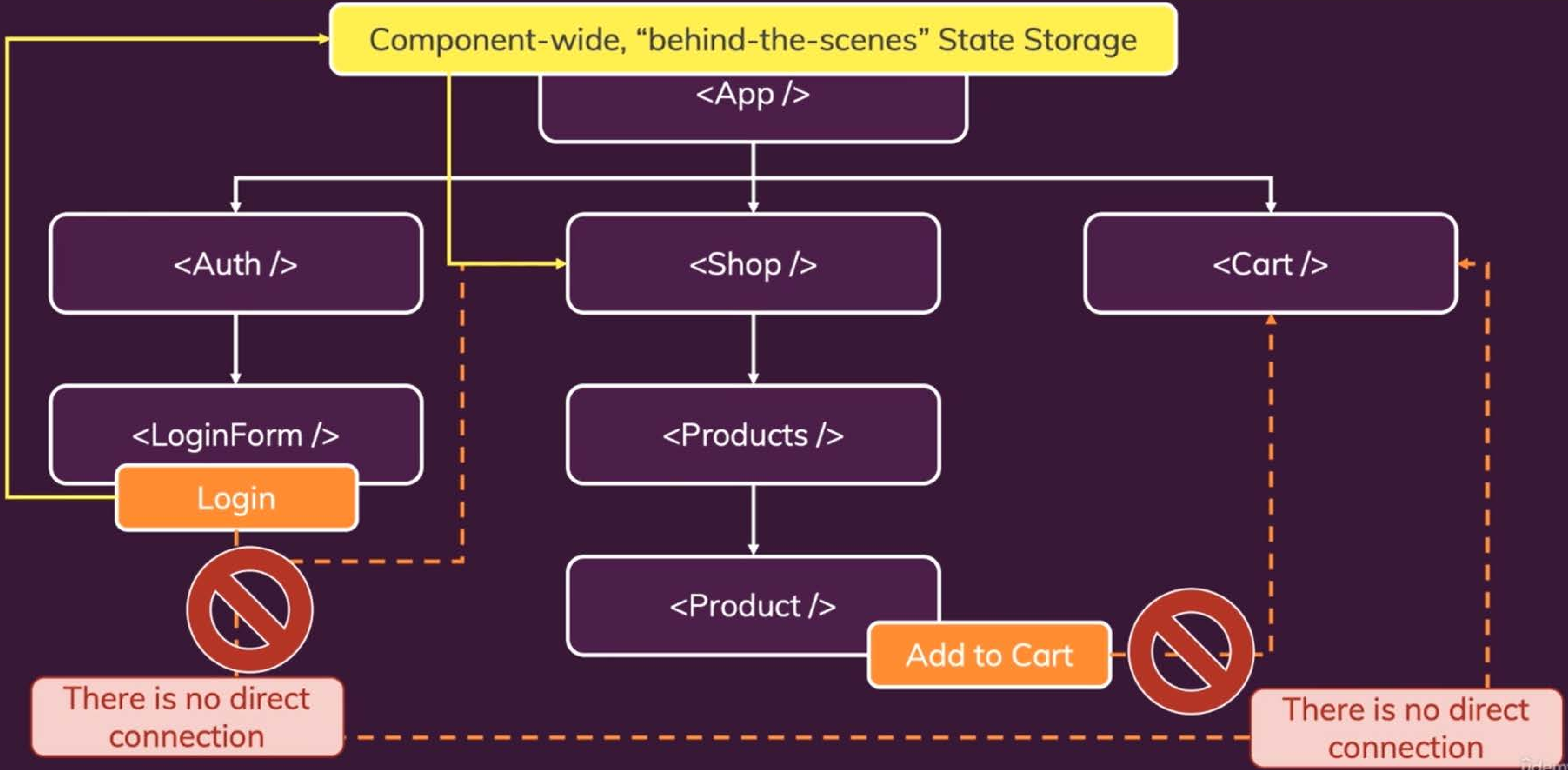
ACADE MIND

Context to the Rescue!

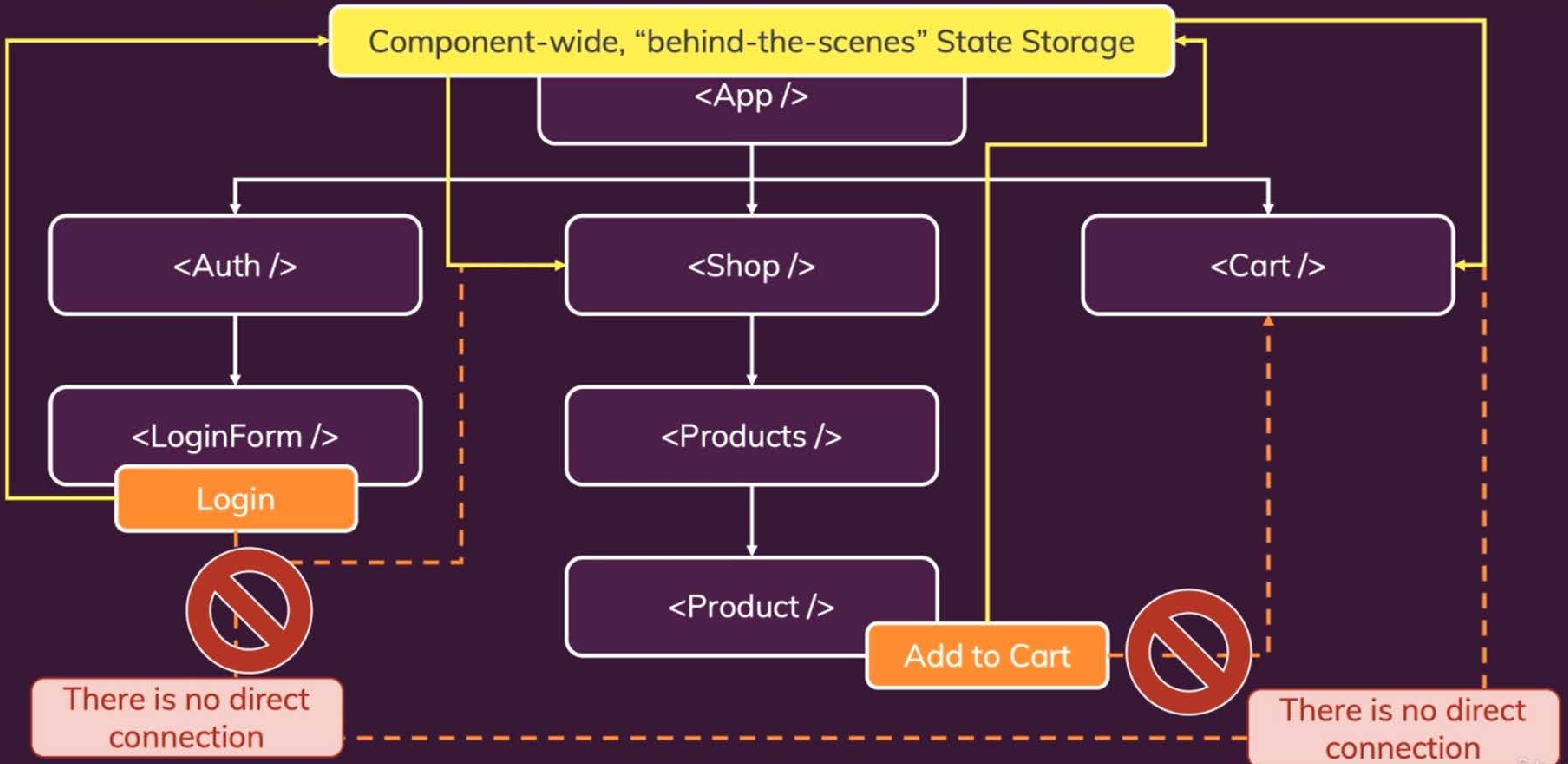


Component-wide, "behind-the-scenes" State Storage
<App />
<Auth />
<Shop />
<Cart />
<LoginForm />
Login
<Products />
<Product />
Add to Cart
There is no direct connection
There is no direct connection

Context to the Rescue!


Component-wide, "behind-the-scenes" State Storage
<App />
<Auth />
<Shop />
<Cart />
<LoginForm />
<Products />
Login
<Product />
Add to Cart
There is no direct connection
There is no direct connection

Context to the Rescue!


Component-wide, "behind-the-scenes" State Storage
<App />
<Auth />
<Shop />
<Cart />
<LoginForm />
<Products />
Login
<Product />
Add to Cart
There is no direct connection
There is no direct connection

Context Limitations

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

React App

Provide more ways to bail out

github.com/facebook/react/issues/14110

Closed Provide more ways to bail out inside Hooks #14110

gaearon opened this issue on 5 Nov 2018 · 135 comments

Load more...

sebmarkbage commented on 18 Dec 2018 Member

My personal summary is that new context is ready to be used for low frequency unlikely updates (like locale/theme). It's also good to use it in the same way as old context was used. I.e. for static values and then propagate updates through subscriptions. It's not ready to be used as a replacement for all Flux-like state propagation.

👍 34

53 hidden items

Load more...

gaearon commented on 30 Jan 2019 Member Author

This discussion will be more productive if we have an actual code example we can discuss.

That said this discussion is also going in circles. I'm not sure it's helpful to keep reiterating the same points.

This is a "nice to have" request but it ties into many other questions around context. We won't likely address it in isolation but we can revisit it in about half a year and see where we are.

Ideally there could be a follow up issue focused on this particular request that also has a realistic demo that shows the problematic use case suffering from the performance problem. That would anchor the discussion. Thanks!

Jessidhia commented on 30 Jan 2019 • edited Collaborator

ACADE MIND

Context Limitations

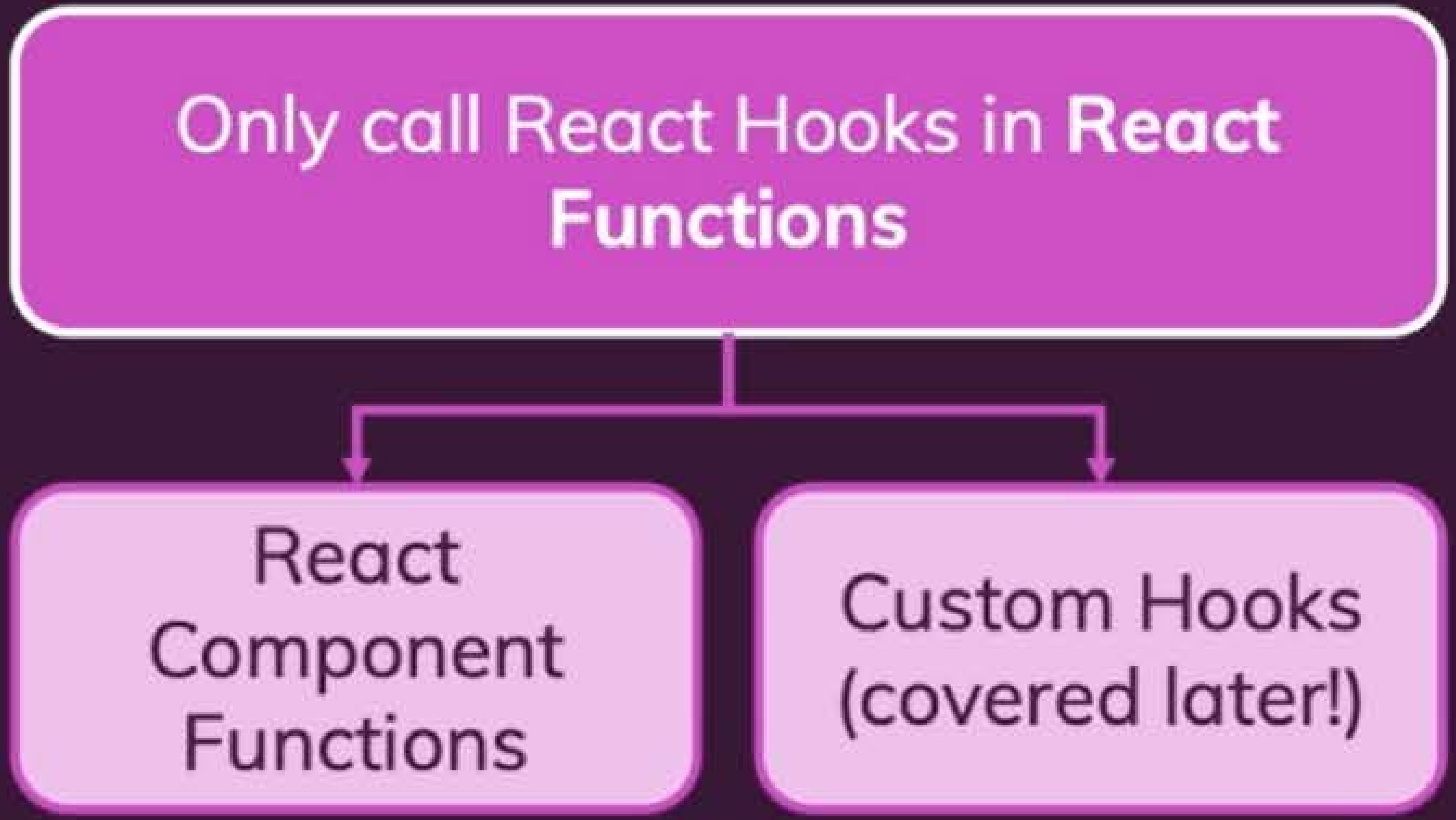
React Context is **NOT optimized** for high frequency changes!

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

Rules of Hooks


Only call React Hooks in **React Functions**
React Component Functions
Custom Hooks (covered later!)

EXPLORER
ontext.js
Button.js
MainHeader.js
Home.js
Login.js
REACT-THE-COMPLETE-G...
Home
Home.js
Home.module...
Login
Login.js 1
Login.module....
MainHeader
MainHeader.js
MainHeader....
Navigation.js
Navigation.m...
UI
Button
Button.js
Button.mod...
Card
store
OUTLINE
TIMELINE
NPM SCRIPTS
src > components > Login > Login.js > emailReducer
PROBLEMS
Peek Problem (⌥F8)
Quick Fix... (⌘.)

```
import React, { useState, useEffect, useReducer, useContext } from 'react';

import Card from '../UI/Card/Card';
import classes from './Login.module.css';
import Button from '../UI/Button/Button';
import AuthContext from '../../store/auth-context';

const emailReducer = (state, action) => {
  useState();

};
```

```
(alias) useState<undefined>(): [undefined, React.Dispatch<(prevSta
te: undefined) => undefined>] (+1 overload)
import useState
```

Returns a stateful value, and a function to update it.

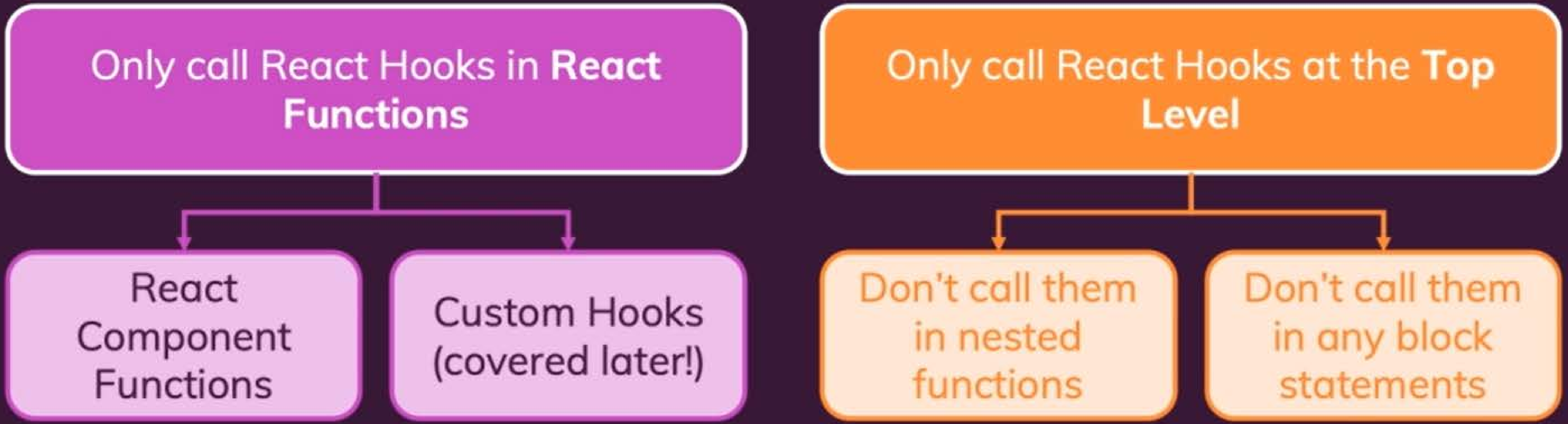
@version — 16.8.0

@see — https://reactjs.org/docs/hooks-reference.html#usestate

```
React Hook "useState" is called in function "emailReducer" which
is neither a React function component or a custom React Hook
function. eslint(react-hooks/rules-of-hooks)
```

```
Note that
To create a production build, use npm run build.
```

Rules of Hooks



- Only call React Hooks in **React Functions**
  - React Component Functions
  - Custom Hooks (covered later!)
- Only call React Hooks at the **Top Level**
  - Don't call them in nested functions
  - Don't call them in any block statements

EXPLORER
ontext.js
Button.js
MainHeader.js
Home.js
Login.js
REACT-THE-COMPLETE-G...
Home
Home.js
Home.module...
Login
Login.js 1
Login.module....
MainHeader
MainHeader.js
MainHeader....
Navigation.js
Navigation.m...
UI
Button
Button.js
Button.mod...
Card
store
OUTLINE
TIMELINE
NPM SCRIPTS
src > components > Login > Login.js > Login > useEffect() callback
PROBLEMS 1
Peek Problem (⌥F8)
Quick Fix... (⌘.)
ûdemy

```
    isValid: null,
  });

  const authCtx = useContext(AuthContext);

  useEffect(() => {
    console.log('EFFECT RUNNING');
    useContext();
```

```
(alias) useContext<any>(context: React.Context<any>): any
import useContext
```

Accepts a context object (the value returned from `React.createContext`) and returns the current context value, as given by the nearest context provider for the given context.

@version — 16.8.0

@see — https://reactjs.org/docs/hooks-reference.html#usecontext

```
React Hook "useContext" cannot be called inside a callback. React
Hooks must be called in a React function component or a custom
React Hook function. eslint(react-hooks/rules-of-hooks)
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.
```

EXPLORER
ontext.js
Button.js
MainHeader.js
Home.js
Login.js
REACT-THE-COMPLETE-G...
Home
Home.js
Home.module...
Login
Login.js 1
Login.module....
MainHeader
MainHeader.js
MainHeader....
Navigation.js
Navigation.m...
UI
Button
Button.js
Button.mod...
Card
store
OUTLINE
TIMELINE
NPM SCRIPTS
src > compon
PROBLEMS 1
OUTPUT
TERMINAL
1: node
Peek Problem (⌥F8)
Quick Fix... (⌘.)

```
(alias) useState<undefined>(): [undefined, React.Dispatch<(prevState: undefined) => undefined>] (+1 overload)
import useState
```

Returns a stateful value, and a function to update it.

@version — 16.8.0

@see — https://reactjs.org/docs/hooks-reference.html#usestate

```
React Hook "useState" is called conditionally. React Hooks must be called in the exact same order in every component render. eslint(react-hooks/rules-of-hooks)
```

```
  })
  co

  })

  co

  if
    useState();
  }

  useEffect(() => {
    console.log('EFFECT RUNNING');
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.
```

Rules of Hooks

- Only call React Hooks in **React Functions**
  - React Component Functions
  - Custom Hooks (covered later!)
- Only call React Hooks at the **Top Level**
  - Don't call them in nested functions
  - Don't call them in any block statements

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

EXPLORER
ontext.js
Button.js
MainHeader.js
Home.js
Login.js
REACT-THE-COMPLETE-G...
src > components > Login > Login.js > Login
Home
Home.js
Home.module...
Login
Login.js M
Login.module....
MainHeader
MainHeader.js
MainHeader....
Navigation.js
Navigation.m...
UI
Button
Button.js
Button.mod...
Card
store
OUTLINE
TIMELINE
NPM SCRIPTS
PROBLEMS
OUTPUT
TERMINAL
1: node

```
    value: '',
    isValid: null,
  });

  const authCtx = useContext(AuthContext);

  useEffect(() => {
    console.log('EFFECT RUNNING');

    return () => {
      console.log('EFFECT CLEANUP');
    };
  }, []);

  const { isValid: emailIsValid } = emailState;
  const { isValid: passwordIsValid } = passwordState;

  useEffect(() => {
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.
```

EXPLORER
ontext.js
Button.js
MainHeader.js
Home.js
Login.js
REACT-THE-COMPLETE-G...
Home.js
Home.module...
Login
Login.js M
Login.module....
MainHeader
MainHeader.js
MainHeader....
Navigation.js
Navigation.m...
UI
Button
Button.js
Button.mod...
Card
store
OUTLINE
TIMELINE
NPM SCRIPTS
src > components > Login > Login.js > Login > useEffect() callback
PROBLEMS
OUTPUT
TERMINAL
1: node

```
  const { isValid: emailIsValid } = emailState;
  const { isValid: passwordIsValid } = passwordState;

  useEffect(() => {
    const identifier = setTimeout(() => {
      console.log('Checking form validity!');
      setFormIsValid(emailIsValid && passwordIsValid);
    }, 500);

    return () => {
      console.log('CLEANUP');
      clearTimeout(identifier);
    };
  }, [emailIsValid, passwordIsValid]);

  const emailChangeHandler = (event) => {
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.
```

EXPLORER
ontext.js
Button.js
MainHeader.js
Home.js
Login.js
REACT-THE-COMPLETE-G...
Home.js
Home.module...
Login
Login.js M
Login.module....
MainHeader
MainHeader.js
MainHeader....
Navigation.js
Navigation.m...
UI
Button
Button.js
Button.mod...
Card
store
OUTLINE
TIMELINE
NPM SCRIPTS
src > components > Login > Login.js > Login
PROBLEMS
OUTPUT
TERMINAL
1: node
Udemy

```
  const { isValid: emailIsValid } = emailState;
  const { isValid: passwordIsValid } = passwordState;

  useEffect(() => {
    const identifier = setTimeout(() => {
      console.log('Checking form validity!');
      setFormIsValid(emailIsValid && passwordIsValid);
    }, 500);

    return () => {
      console.log('CLEANUP');
      clearTimeout(identifier);
    };
  }, [emailIsValid, passwordIsValid]);

  const emailChangeHandler = (event) => {
    dispatchEmail({ type: 'USER_INPUT', val: event.target.value });
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.
```

EXPLORER
ontext.js
Button.js
MainHeader.js
Home.js
Login.js
REACT-THE-COMPLETE-G...
src > components > Login > Login.js > Login
Home.js
Home.module...
Login
Login.js 1, M
Login.module....
MainHeader
MainHeader.js
MainHeader....
Navigation.js
Navigation.m...
UI
Button
Button.js
Button.mod...
Card
store
OUTLINE
TIMELINE
NPM SCRIPTS
PROBLEMS 1
OUTPUT
TERMINAL
1: node
Peek Problem (⌥F8)
Quick Fix... (⌘.)
ûdemy

```
  useEffect(() => {
    const identifier = setTimeout(() => {
      console.log('Checking form validity!');
      setFormIsValid(emailIsValid && passwordIsValid);
    }
    
      r
    
    
    }
  }, [emailIsValid]);

  const emailChangeHandler = (event) => {
    dispatchEmail({ type: 'USER_INPUT', val: event.target.value });

    // setFormIsValid(
    //   event.target.value.includes('@') && passwordState.isValid
```

```
const emailIsValid: any

React Hook useEffect has a missing dependency: 'passwordIsValid'.
Either include it or remove the dependency array. eslint(react-
hooks/exhaustive-deps)
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.
```