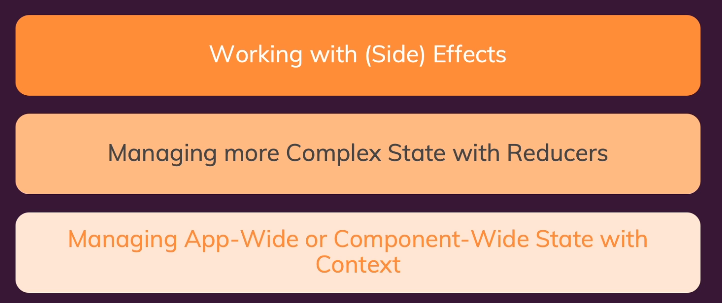
Section 10: Advanced: Handling Side Effects, Using…



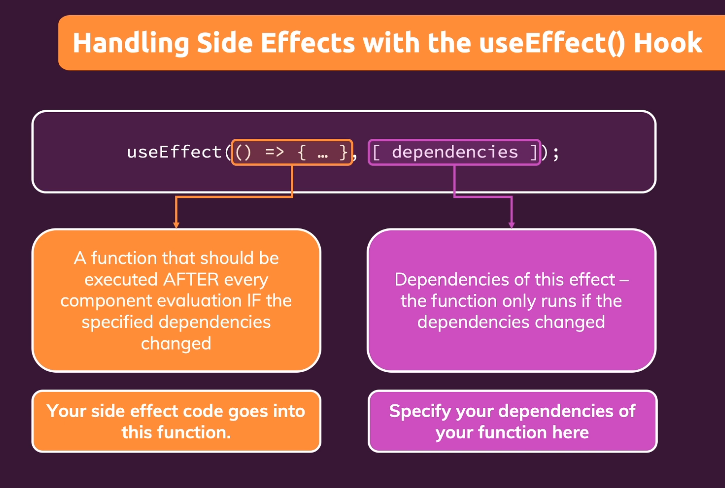
Source code to start project

<https://github.com/academind/react-complete-guide-code/tree/10-side-effects-reducers-context-api/code>

110. What are ”Side Effects” & Introducing userEffect

Use dependencies to execute function (IOC and DI)

(Avoid excute every time when run function html, you can use dependencies to avoid this problem)



111. Using the useEffect() Hook

Step 1 go to App.js and add userEffect + each status of localStorage

import React, { useEffect, useState } from 'react';

import Login from './components/Login/Login';

import Home from './components/Home/Home';

import MainHeader from './components/MainHeader/MainHeader';

function App() {

  const [isLoggedIn, setIsLoggedIn] = useState(false);

  useEffect(() => {

    const storedUserLoggedInInFormation = localStorage.getItem('isLoggedIn');

    if(storedUserLoggedInInFormation === '1'){

      setIsLoggedIn(true);

    }

  }, []);

  const loginHandler = (email, password) => {

    // We should of course check email and password

    // But it's just a dummy/ demo anyways

    localStorage.setItem('isLoggedIn', '1');

    setIsLoggedIn(true);

  };

  const logoutHandler = () => {

    localStorage.removeItem('isLoggedIn');

    setIsLoggedIn(false);

  };

  return (

    <React.Fragment>

      <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />

      <main>

        {!isLoggedIn && <Login onLogin={loginHandler} />}

        {isLoggedIn && <Home onLogout={logoutHandler} />}

      </main>

    </React.Fragment>

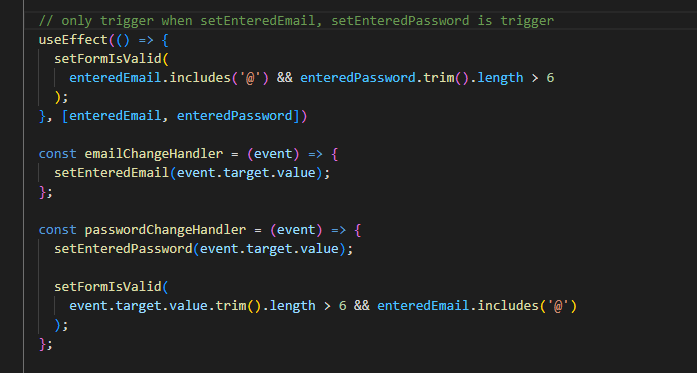
  );

}

export default App;

112. useEffect & Dependencies

Step 1 go to Login.js change like this



113. What to add & Not to Add as Dependencies

there are a **few exceptions** you should be aware of:

* You **DON'T need to add state updating functions** (as we did in the last lecture with setFormIsValid): React guarantees that those functions never change, hence you don't need to add them as dependencies (you could though)
* You also **DON'T need to add "built-in" APIs or functions** like fetch(), localStorage etc (functions and features built-into the browser and hence available globally): These browser APIs / global functions are not related to the React component render cycle and they also never change
* You also **DON'T need to add variables or functions** you might've **defined OUTSIDE of your components** (e.g. if you create a new helper function in a separate file): Such functions or variables also are not created inside of a component function and hence changing them won't affect your components (components won't be re-evaluated if such variables or functions change and vice-versa)

So long story short: You must add all "things" you use in your effect function **if those "things" could change because your component (or some parent component) re-rendered.** That's why variables or state defined in component functions, props or functions defined in component functions have to be added as dependencies!

Here's a made-up dummy example to further clarify the above-mentioned scenarios:

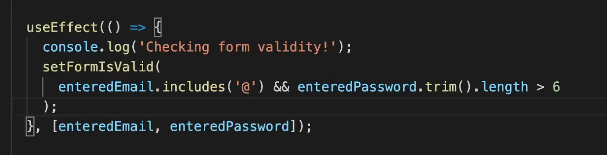
1. import { useEffect, useState } from 'react';
3. let myTimer;
5. const MyComponent = (props) => {
6. const [timerIsActive, setTimerIsActive] = useState(false);
8. const { timerDuration } = props; // using destructuring to pull out specific props values
10. useEffect(() => {
11. if (!timerIsActive) {
12. setTimerIsActive(true);
13. myTimer = setTimeout(() => {
14. setTimerIsActive(false);
15. }, timerDuration);
16. }
17. }, [timerIsActive, timerDuration]);
18. };

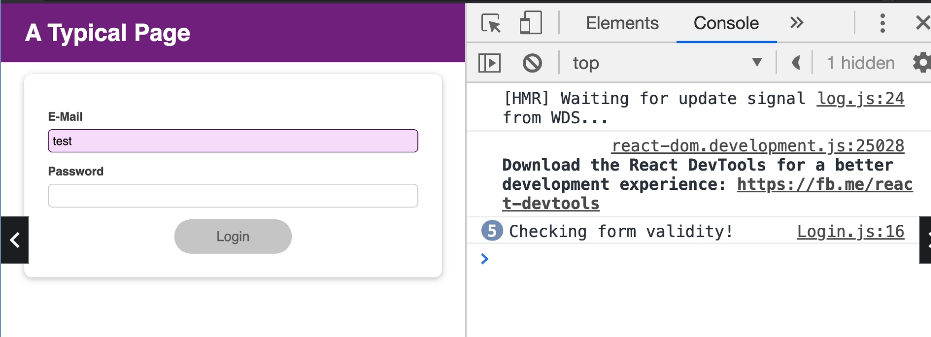
In this example:

* timerIsActive is **added as a dependency** because it's component state that may change when the component changes (e.g. because the state was updated)
* timerDuration is **added as a dependency** because it's a prop value of that component - so it may change if a parent component changes that value (causing this MyComponent component to re-render as well)
* setTimerIsActive is **NOT added as a dependency** because it's that **exception**: State updating functions could be added but don't have to be added since React guarantees that the functions themselves never change
* myTimer is **NOT added as a dependency** because it's **not a component-internal variable** (i.e. not some state or a prop value) - it's defined outside of the component and changing it (no matter where) **wouldn't cause the component to be re-evaluated**
* setTimeout is **NOT added as a dependency** because it's **a built-in API** (built-into the browser) - it's independent from React and your components, it doesn't change

114. Using the useEffect Cleanup Function

Problem when every type input, it immediately trigger





Solve use timmer to reduce check

Step 1 Change UseEffect in Login.js like this

  // only trigger when setEnteredEmail, setEnteredPassword is trigger

  useEffect(() => {

    const identifier = setTimeout(() =>{

      console.log('Checking form validity');

      setFormIsValid(

        enteredEmail.includes('@') && enteredPassword.trim().length > 6

      );

    }, 500);

    return () => {

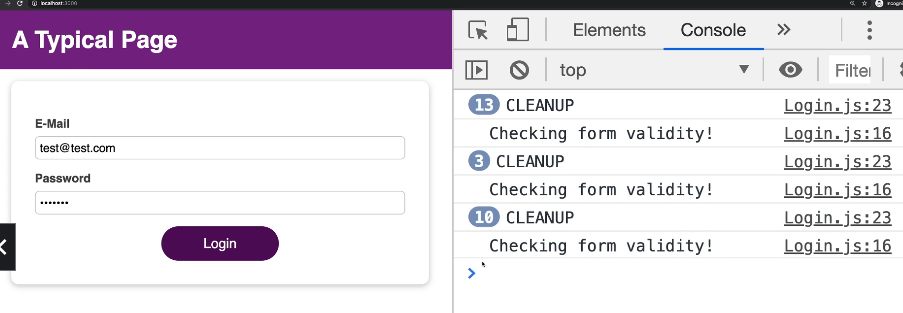
      console.log('CLEANUP');

      clearTimeout(identifier);

    }

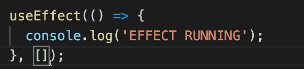
  }, [enteredEmail, enteredPassword])

Step 2 test

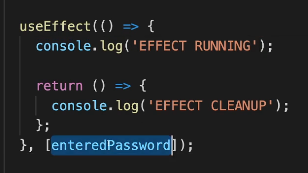


115. useEffect Summary

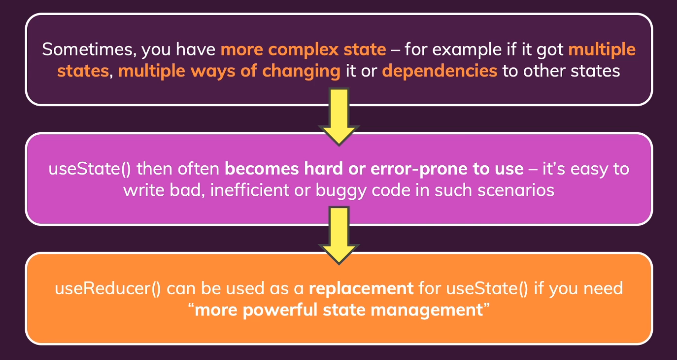
Note 1: Only run one time



Note 2: run when some properties call

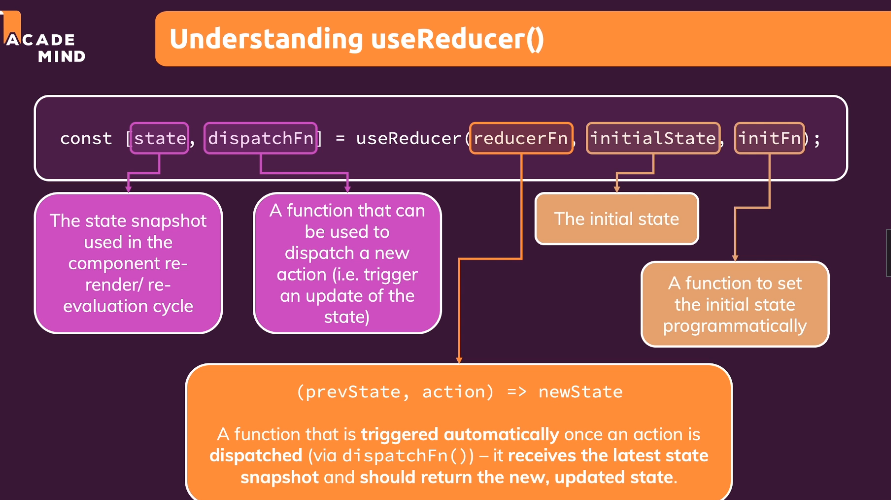


116. Introducing useReducer & Reducers In General



117. Introducing useReducer & Reducers In General

+Để quản lí một số state liên quan với nhau như khi fetching data từ API: status, data, error dễ dàng hơn



Step 1 go to source <https://github.com/academind/react-complete-guide-code/blob/10-side-effects-reducers-context-api/code/07-using-the-usereducer-hook/src/components/Login/Login.js>

And check and add like this, and also change all relate that properties



118. useReducer & useEffect

Step change another properties (password)

Go to Login.js and change like this

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

import Card from '../UI/Card/Card';

import classes from './Login.module.css';

import Button from '../UI/Button/Button';

const emailReducer = (state, action) => {

  if (action.type === 'USER\_INPUT') {

    return { value: action.val, isValid: action.val.includes('@') };

  }

  if (action.type === 'INPUT\_BLUR') {

    return { value: state.value, isValid: state.value.includes('@') };

  }

  return { value: '', isValid: false };

};

const passwordReducer = (state, action) => {

  if (action.type === 'USER\_INPUT') {

    return { value: action.val, isValid: action.val.trim().length > 6 };

  }

  if (action.type === 'INPUT\_BLUR') {

    return { value: state.value, isValid: state.value.trim().length > 6 };

  }

  return { value: '', isValid: false };

};

const Login = (props) => {

  // const [enteredEmail, setEnteredEmail] = useState('');

  // const [emailIsValid, setEmailIsValid] = useState();

  // const [enteredPassword, setEnteredPassword] = useState('');

  // const [passwordIsValid, setPasswordIsValid] = useState();

  const [formIsValid, setFormIsValid] = useState(false);

  const [emailState, dispatchEmail] = useReducer(emailReducer, {

    value: '',

    isValid: null,

  });

  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {

    value: '',

    isValid: null,

  });

  useEffect(() => {

    console.log('EFFECT RUNNING');

    return () => {

      console.log('EFFECT CLEANUP');

    };

  }, []);

  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 });

    // setFormIsValid(

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

    // );

  };

  const passwordChangeHandler = (event) => {

    dispatchPassword({ type: 'USER\_INPUT', val: event.target.value });

    // setFormIsValid(emailState.isValid && event.target.value.trim().length > 6);

  };

  const validateEmailHandler = () => {

    dispatchEmail({ type: 'INPUT\_BLUR' });

  };

  const validatePasswordHandler = () => {

    dispatchPassword({ type: 'INPUT\_BLUR' });

  };

  const submitHandler = (event) => {

    event.preventDefault();

    props.onLogin(emailState.value, passwordState.value);

  };

  return (

    <Card className={classes.login}>

      <form onSubmit={submitHandler}>

        <div

          className={`${classes.control} ${

            emailState.isValid === false ? classes.invalid : ''

          }`}

        >

          <label htmlFor="email">E-Mail</label>

          <input

            type="email"

            id="email"

            value={emailState.value}

            onChange={emailChangeHandler}

            onBlur={validateEmailHandler}

          />

        </div>

        <div

          className={`${classes.control} ${

            passwordState.isValid === false ? classes.invalid : ''

          }`}

        >

          <label htmlFor="password">Password</label>

          <input

            type="password"

            id="password"

            value={passwordState.value}

            onChange={passwordChangeHandler}

            onBlur={validatePasswordHandler}

          />

        </div>

        <div className={classes.actions}>

          <Button type="submit" className={classes.btn} disabled={!formIsValid}>

            Login

          </Button>

        </div>

      </form>

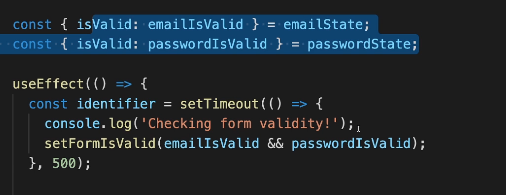
    </Card>

  );

};

export default Login;

Note: use this to avoid repeat in useEffect



119. Adding Nested Properties As Dependencies to useEffect

In the previous lecture, we used object destructuring to add object properties as dependencies to useEffect().

1. const { someProperty } = someObject;
2. useEffect(() => {
3. // code that only uses someProperty ...
4. }, [someProperty]);

This is a **very common pattern and approach**, which is why I typically use it and why I show it here (I will keep on using it throughout the course).

I just want to point out, that they **key thing is NOT that we use destructuring** but that we **pass specific properties instead of the entire object** as a dependency.

We could also write this code and it would **work in the same way**.

1. useEffect(() => {
2. // code that only uses someProperty ...
3. }, [someObject.someProperty]);

This works just fine as well!

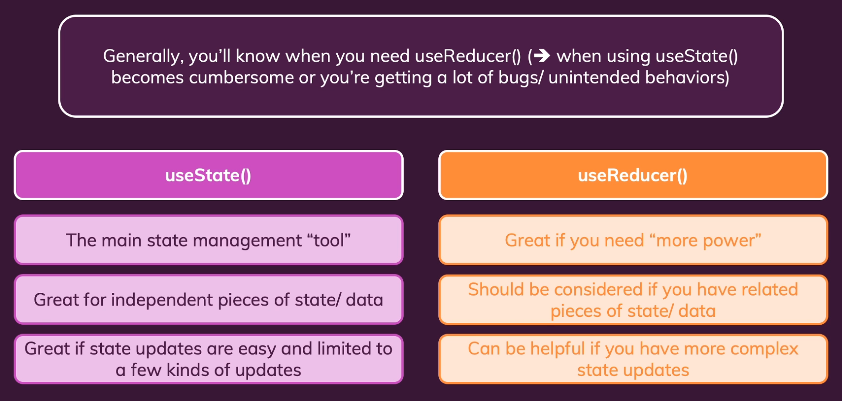
But you should **avoid** this code:

1. useEffect(() => {
2. // code that only uses someProperty ...
3. }, [someObject]);

Why?

Because now the **effect function would re-run whenever ANY property** of someObject changes - not just the one property (someProperty in the above example) our effect might depend on

120. useReducer vs useState for State Management

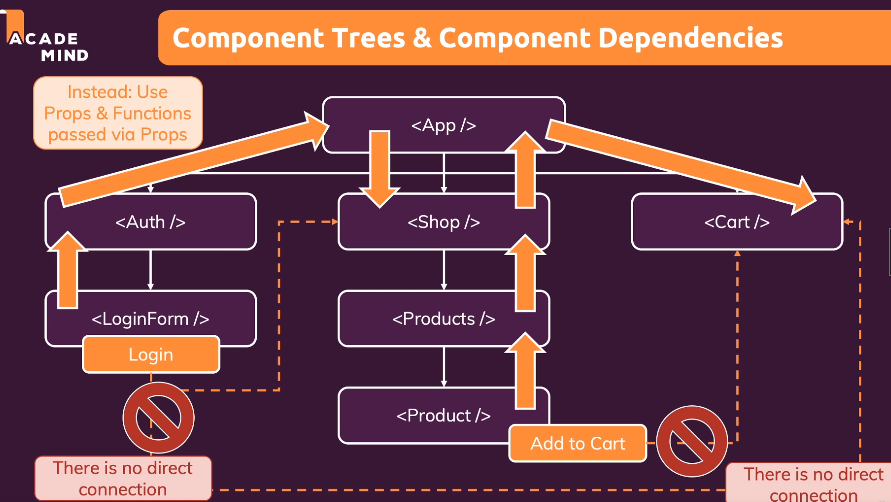


121. Introducing React Context (Context API)

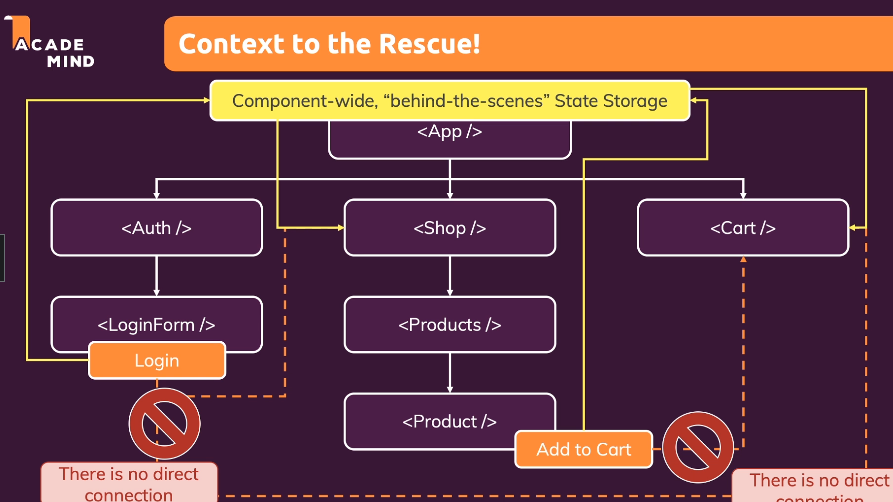
Problem: when you pass props through a a lot of component (Việc sử dụng “prop drilling” thì việc truyền dữ liệu đến các component con lồng nhau sâu sẽ rất cồng kềnh )

Do đó, React Context xuất hiện để khắc phục những nhược điểm của “prop drilling”.

**React Context** sẽ cho phép chúng ta có thể tạo data và truyền nó với một provider đến tất cả component trong ứng dụng React mà không cần dùng “prop drilling”.



Solved by use react context



122. Using the React Context API

Step 1: create new folder component/store

Step 2: create new auth-context.js in folder store

import React from "react";

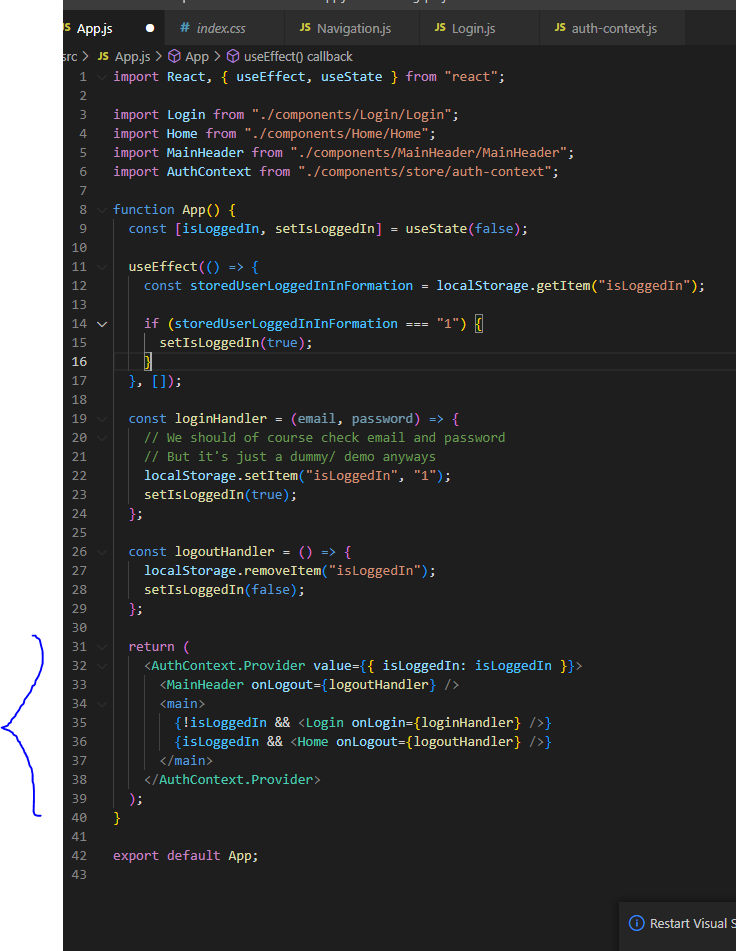
const AuthContext = React.createContext({

    isLoggedIn: false

});

export default AuthContext;

Step 3: in App.js wrapper like this (this is parent componenet will init context)



Step 4: in Navigation.js change like this

import React from "react";

import AuthContext from "../store/auth-context";

import classes from "./Navigation.module.css";

const Navigation = (props) => {

  return (

    <AuthContext.Consumer>

      {(ctx)=>{

        return (

          <nav className={classes.nav}>

          <ul>

            {ctx.isLoggedIn && (

              <li>

                <a href="/">Users</a>

              </li>

            )}

            {ctx.isLoggedIn && (

              <li>

                <a href="/">Admin</a>

              </li>

            )}

            {ctx.isLoggedIn && (

              <li>

                <button onClick={props.onLogout}>Logout</button>

              </li>

            )}

          </ul>

        </nav>

        )

      }}

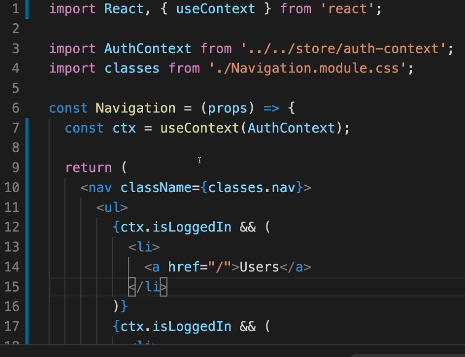
    </AuthContext.Consumer>

  );

};

export default Navigation;

or like this



Step 5: Done

123. Tapping Into Context with the useContext Hook

124. Making Context Dynamic

Use same with logout

Step 1 change App.js

   <AuthContext.Provider value={{ isLoggedIn: isLoggedIn, onLogout: logoutHandler }}>

      <MainHeader onLogout={logoutHandler} />

      <main>

        {!isLoggedIn && <Login onLogin={loginHandler} />}

        {isLoggedIn && <Home onLogout={logoutHandler} />}

      </main>

    </AuthContext.Provider>

Step 2 go to Navigator change like this

 <AuthContext.Consumer>

      {(ctx)=>{

        return (

          <nav className={classes.nav}>

          <ul>

            {ctx.isLoggedIn && (

              <li>

                <a href="/">Users</a>

              </li>

            )}

            {ctx.isLoggedIn && (

              <li>

                <a href="/">Admin</a>

              </li>

            )}

            {ctx.isLoggedIn && (

              <li>

                <button onClick={ctx.onLogout}>Logout</button>

              </li>

            )}

          </ul>

        </nav>

        )

      }}

    </AuthContext.Consumer>

Step 3: now logout will be changed

125. Building & Using a Custom Context Provider Component

(put all useEffect, function in auth-context.js)

Step 1 Change auth-context.js like this

import React, { useState, useEffect } from "react";

const AuthContext = React.createContext({

    isLoggedIn: false,

    onLogout: () => {},

    onLogin: (email, password) => {}

  });

  export const AuthContextProvider = (props) => {

    const [isLoggedIn, setIsLoggedIn] = useState(false);

    useEffect(() => {

      const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');

      if (storedUserLoggedInInformation === '1') {

        setIsLoggedIn(true);

      }

    }, []);

    const logoutHandler = () => {

      localStorage.removeItem('isLoggedIn');

      setIsLoggedIn(false);

    };

    const loginHandler = () => {

      localStorage.setItem('isLoggedIn', '1');

      setIsLoggedIn(true);

    };

    return (

      <AuthContext.Provider

        value={{

          isLoggedIn: isLoggedIn,

          onLogout: logoutHandler,

          onLogin: loginHandler,

        }}

      >

        {props.children}

      </AuthContext.Provider>

    );

  };

  export default AuthContext;

Step 2 then in App.js

import React, { useContext, useEffect, useState } from "react";

import Login from './components/Login/Login';

import Home from './components/Home/Home';

import MainHeader from './components/MainHeader/MainHeader';

import AuthContext from './store/auth-context';

function App() {

  const ctx = useContext(AuthContext);

  return (

    <React.Fragment>

      <MainHeader />

      <main>

        {!ctx.isLoggedIn && <Login />}

        {ctx.isLoggedIn && <Home />}

      </main>

    </React.Fragment>

  );

}

export default App;

Step 3 and index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import {AuthContextProvider } from './store/auth-context';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

    <AuthContextProvider >

        <App />

    </AuthContextProvider >

);

Step 4 in MainHeader.js

import React from 'react';

import Navigation from './Navigation';

import classes from './MainHeader.module.css';

const MainHeader = () => {

  return (

    <header className={classes['main-header']}>

      <h1>A Typical Page</h1>

      <Navigation  />

    </header>

  );

};

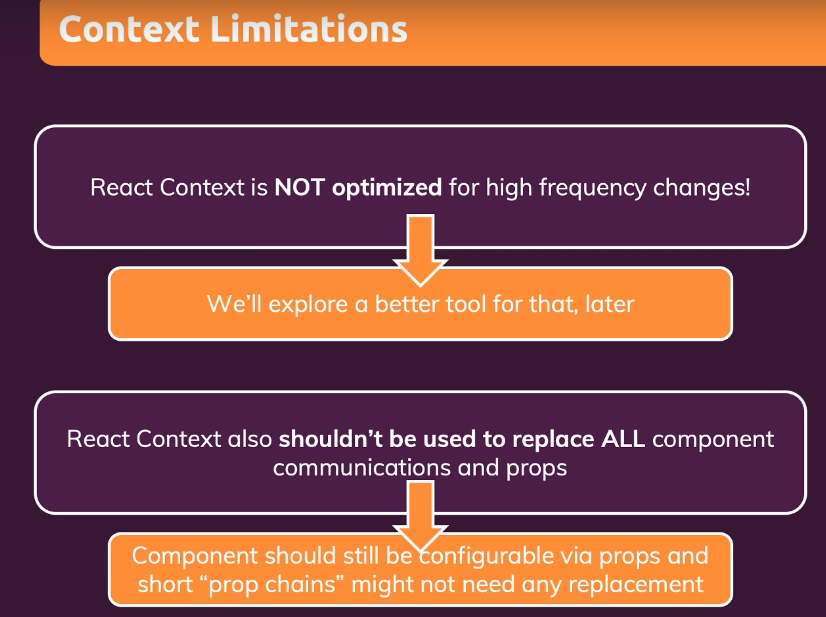
export default MainHeader;

Step 5 Done

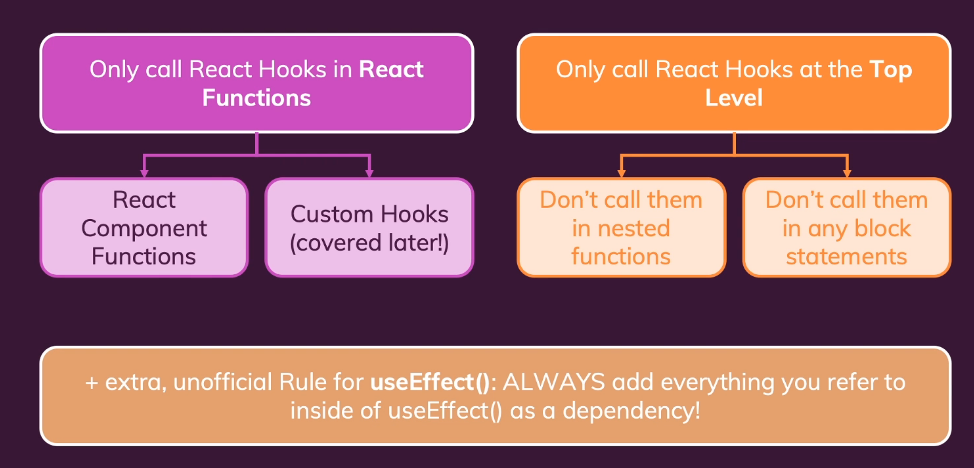
Note: you can check code here

<https://github.com/academind/react-complete-guide-code/tree/10-side-effects-reducers-context-api/code/12-building-and-using-a-custom-context-provider-cmp>

126. React Context Limitations



127. Learning rule of hook



128. Refactoring an Input Component (you can use resources to check)

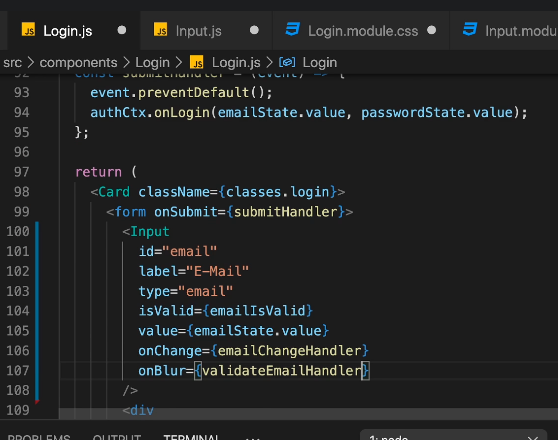
Step 1 add new folder Input and new js Input.js

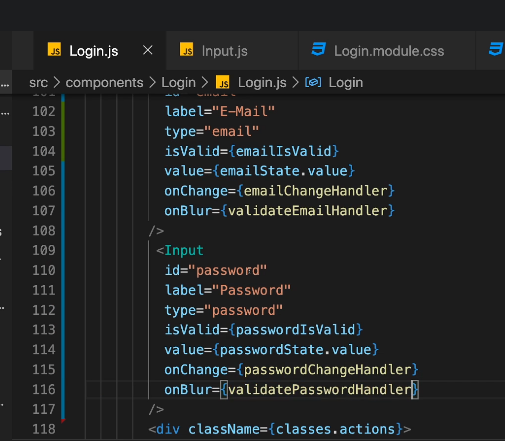
https://github.com/academind/react-complete-guide-code/blob/10-side-effects-reducers-context-api/code/13-finished/src/components/UI/Input/Input.js

Step 2 add new Input.module.css

https://github.com/academind/react-complete-guide-code/blob/10-side-effects-reducers-context-api/code/13-finished/src/components/UI/Input/Input.module.css

Step 3 in Login.js

.



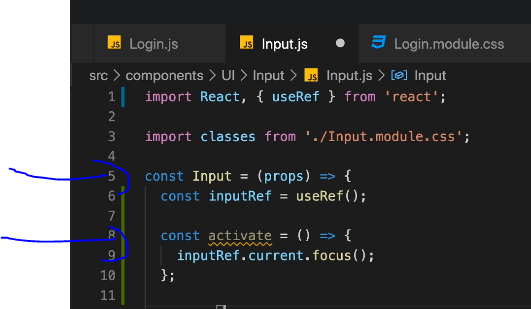
129. Diving into “Forward Refs”

# Refs là gì ?

Ref được viết tắt và có ý nghĩa là reference , là một thuộc tính của 1 thẻ Jsx và tham chiếu tới chính nó . Hay nói cách khác là ref cho phép chúng ta có thể truy cập trực tiếp đến element và sửa đổi nó ngay lập tức mà không cần đến props hay state để component bị re-render lại . Nó khá giống việc chúng ta làm việc với DOM element thông qua việc gọi “document.getElementById()”

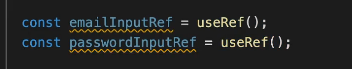
Bài toán : Làm thế nào để có thể lấy được text từ input của component Con và hiển thị được nó ở component Cha và đứng từ component Cha gọi được hàm increaseNumber từ component Con

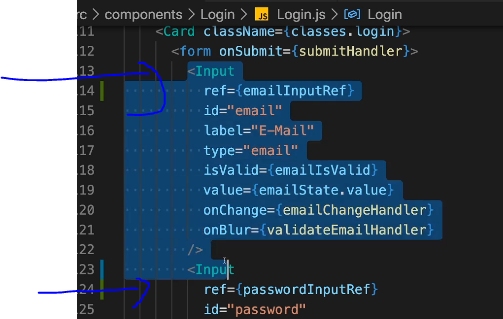
Step 1 in Input.js ( Import useEffect, useRef )



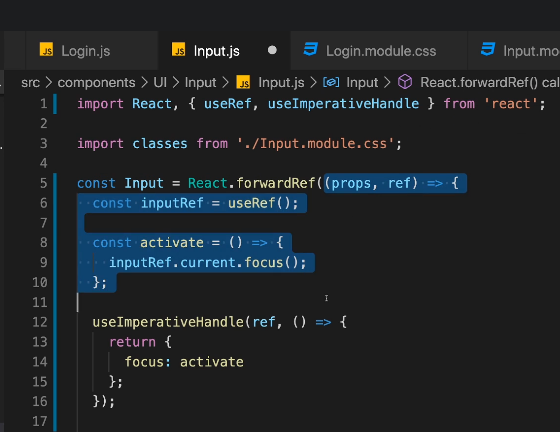
Step 2 in Login.js

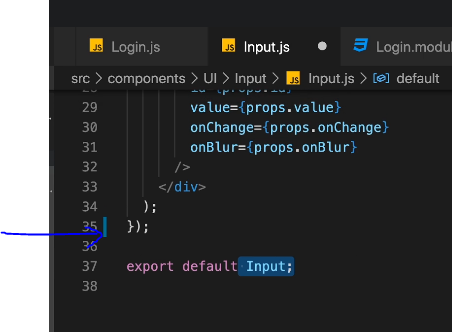
+ Import useEffect, useRef in





Step 3 in Input.js





Step 4 change Login.js like this

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 Input from '../UI/Input/Input';

import AuthContext from '../store/auth-context';

const emailReducer = (state, action) => {

  if (action.type === 'USER\_INPUT') {

    return { value: action.val, isValid: action.val.includes('@') };

  }

  if (action.type === 'INPUT\_BLUR') {

    return { value: state.value, isValid: state.value.includes('@') };

  }

  return { value: '', isValid: false };

};

const passwordReducer = (state, action) => {

  if (action.type === 'USER\_INPUT') {

    return { value: action.val, isValid: action.val.trim().length > 6 };

  }

  if (action.type === 'INPUT\_BLUR') {

    return { value: state.value, isValid: state.value.trim().length > 6 };

  }

  return { value: '', isValid: false };

};

const Login = () => {

  const [formIsValid, setFormIsValid] = useState(false);

  const [emailState, dispatchEmail] = useReducer(emailReducer, {

    value: "",

    isValid: null,

  });

  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {

    value: "",

    isValid: null,

  });

  useEffect(() => {

    console.log("Effect running");

    return () => {

      console.log("Effect cleanup");

    };

  }, []);

  const { isValid: emailIsValid } = emailState;

  const { isValid: passwordIsValid } = passwordState;

  const authCtx = useContext(AuthContext);

  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 });

  };

  const passwordChangeHandler = (event) => {

    dispatchPassword({ type: "USER\_INPUT", val: event.target.value });

  };

  const validateEmailHandler = () => {

    dispatchEmail({ type: "INPUT\_BLUR" });

  };

  const validatePasswordHandler = () => {

    dispatchPassword({ type: "INPUT\_BLUR" });

  };

  const submitHandler = (event) => {

    event.preventDefault();

    if (formIsValid) {

      authCtx.onLogin(emailState.value, passwordState.value);

    } else if (!emailIsValid) {

      emailInputRef.current.focus();

    } else {

      passwordInputRef.current.focus();

    }

  };

  return (

    <Card className={classes.login}>

      <form onSubmit={submitHandler}>

        <div

          className={`${classes.control} ${

            emailState.isValid === false ? classes.invalid : ""

          }`}

        >

          <label htmlFor="email">E-Mail</label>

          <Input

            type="email"

            id="email"

            value={emailState.value}

            onChange={emailChangeHandler}

            onBlur={validateEmailHandler}

          />

        </div>

        <div

          className={`${classes.control} ${

            passwordState.isValid === false ? classes.invalid : ""

          }`}

        >

          <label htmlFor="password">Password</label>

          <Input

            type="password"

            id="password"

            value={passwordState.value}

            onChange={passwordChangeHandler}

            onBlur={validatePasswordHandler}

          />

        </div>

        <div className={classes.actions}>

          <Button type="submit" className={classes.btn} disabled={!formIsValid}>

            Login

          </Button>

        </div>

      </form>

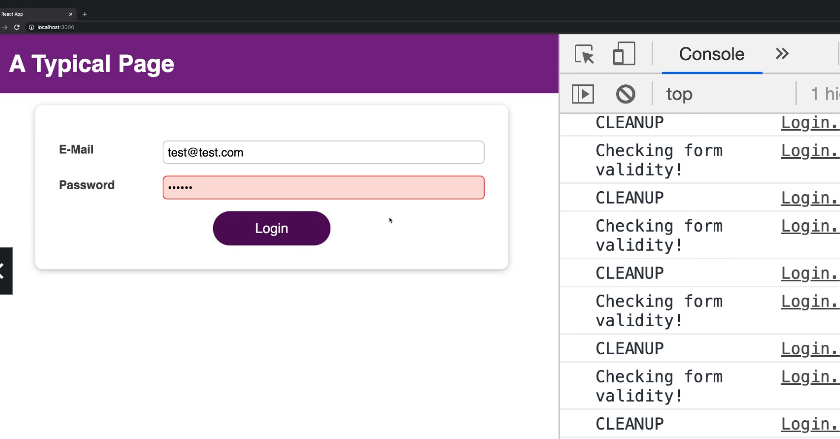
    </Card>

  );

};

export default Login;

Step 5 result



130 Module Resources

You may want to **compare your code to mine** (e.g. to find + fix errors).

For that, you find **multiple code snapshots** for this module here in this Github repository: <https://github.com/academind/react-complete-guide-code/tree/10-side-effects-reducers-context-api>