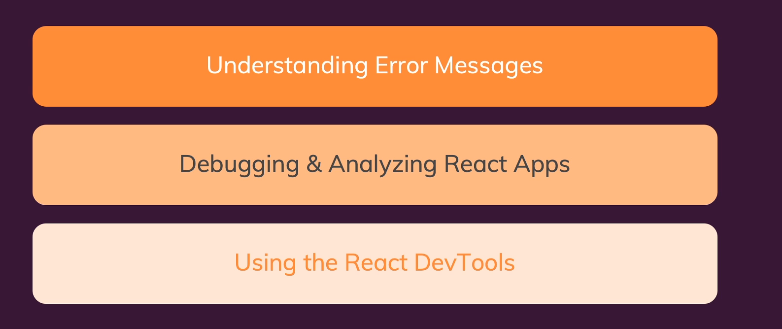
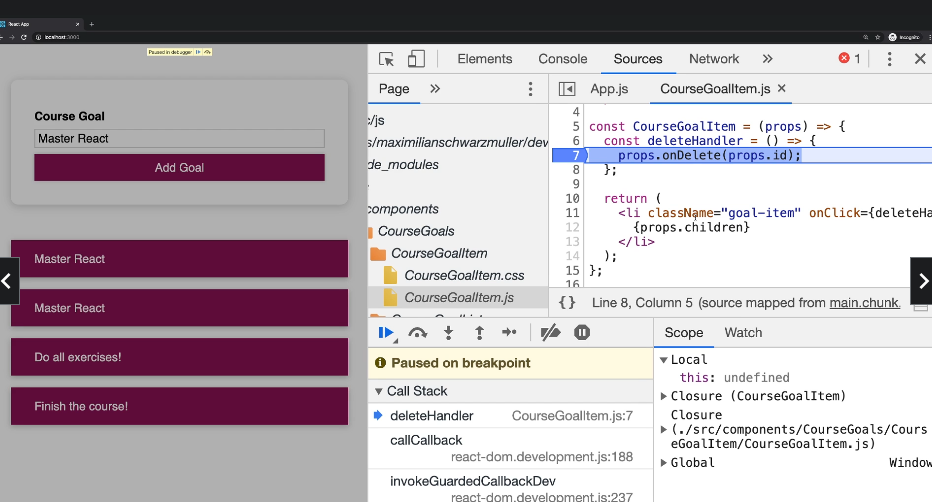
Section 7: Debugging React Apps



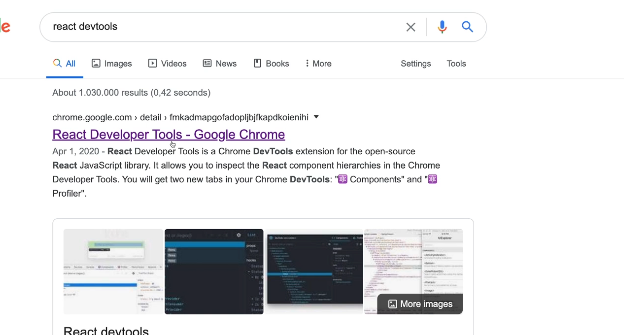
86. Working with Breakpoints

You can use Sources to debug breakpoint by click line like this (you can hover to this element to guest what value)

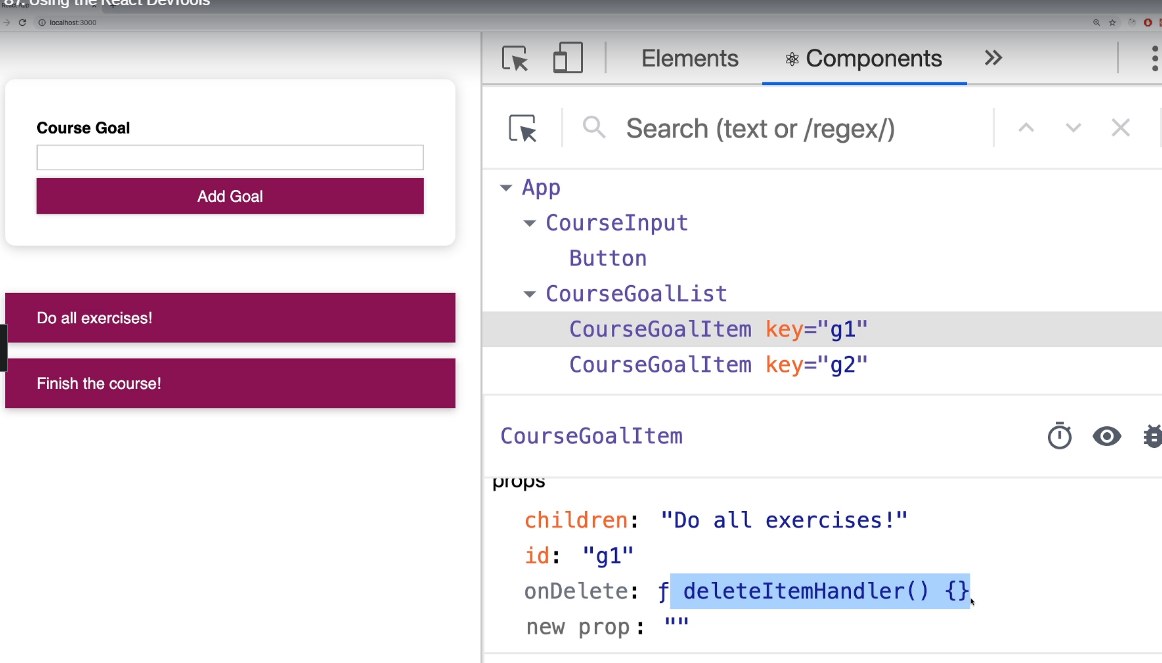


87. Using the React DevTools

Step 1 Search React tool



Step 2 check (you can check element, props, id, source)



88. 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/07-debugging>

89. Module Introduction

Practice you had learned

Homework



90. Adding a ‘User’ Component

Step 1: Download resource first and clone 01

Step 2 create new folder components

Step 3 create new folder components/UI

Step 4 create new folder components/Users

Step 5 create new AddUsers.js in folder Users

import React from "react";

const AddUser = (props) => {

  const addUserHandler = (event) => {

    event.preventDefault();

  };

  return (

    <form onSubmit={addUserHandler}>

      <label htmlFor="username">Username</label>

      <input id="username" type="text" />

      <label htmlFor="age">Age (Year)</label>

      <input id="age" type="number" />

      <button type="submit">Add User</button>

    </form>

  );

};

export default AddUser;

Step 6 in App.js Pass like this

import React from 'react';

import AddUser from './components/Users/AddUser';

function App() {

  return (

    <div>

        <AddUser />

    </div>

  );

}

export default App;

Step 7 Adding index.css like this

\* {

  box-sizing: border-box;

}

html {

  font-family: sans-serif;

  background: #1f1f1f;

}

body {

  margin: 0;

}

Step 8 run result



91. Adding a re-usable “Card’ components

Step 1 create new Card.js in folder UI

import React from "react";

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

const Card = (props) => {

    return <div className={`${classes.card} ${props.className}`}>{props.children}</div>

}

export default Card;

Step 2 create new Card.module.css in folder UI

.card {

    background: white;

    box-shadow:  0 2px 8px rgba(0,0,0,0.26);

    border-radius: 10px;

}

Step 3 add new AddUser.module.css in folder Users

.input {

    margin: 2rem auto;

    padding: 1rem;

    width: 90%;

    max-width: 40rem;

  }

  .input label {

    display: block;

    font-weight: bold;

    margin-bottom: 0.5rem;

  }

  .input input {

    font: inherit;

    display: block;

    width: 100%;

    border: 1px solid #ccc;

    padding: 0.15rem;

    margin-bottom: 0.5rem;

  }

  .input input:focus {

    outline: none;

    border-color: #4f005f;

  }

Step 4 In AddUser.js change like this

import React from "react";

import Card from "../UI/Card";

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

const AddUser = (props) => {

  const addUserHandler = (event) => {

    event.preventDefault();

  };

  return (

    <Card className={classes.input}>

      <form onSubmit={addUserHandler}>

        <label htmlFor="username">Username</label>

        <input id="username" type="text" />

        <label htmlFor="age">Age (Year)</label>

        <input id="age" type="number" />

        <button type="submit">Add User</button>

      </form>

    </Card>

  );

};

export default AddUser;

Step 5 result



92. Adding a re-usable “Button” Component

Step 1: Add new Button.js in folder UI

import React from "react";

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

const Button = (props) => {

  return (

    <button

      className={classes.button}

      type={props.type || "button"}

      onClick={props.onClick}

    >{props.children}</button>

  );

};

export default Button;

Step 2: create new Button.module.css in UI

.button {

    font: inherit;

    border: 1px solid #4f005f;

    background: #4f005f;

    color: white;

    padding: 0.25rem 1rem;

    cursor: pointer;

  }

  .button:hover,

  .button:active {

    background: #741188;

    border-color: #741188;

  }

  .button:focus {

    outline: none;

  }

Step 3: change button to Button import from js (file AddUser.js)

import React from "react";

import Card from "../UI/Card";

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

import Button from "../UI/Button";

const AddUser = (props) => {

  const addUserHandler = (event) => {

    event.preventDefault();

  };

  return (

    <Card className={classes.input}>

      <form onSubmit={addUserHandler}>

        <label htmlFor="username">Username</label>

        <input id="username" type="text" />

        <label htmlFor="age">Age (Year)</label>

        <input id="age" type="number" />

        <Button type="submit">Add User</Button>

      </form>

    </Card>

  );

};

export default AddUser;

Step 4 run result



93. Managing the User Input State

Step 1 add useState in AddUser.js to manage Object

import React from "react";

import Card from "../UI/Card";

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

import Button from "../UI/Button";

const AddUser = (props) => {

  const [enteredUsername, setEnteredUsername] = useState("");

  const [enteredAge, setEnteredAge] = useState("");

  const addUserHandler = (event) => {

    event.preventDefault();

  };

  const usernameChangeHandler = (event) => {

    setEnteredUsername(event.target.value);

  };

  const ageChangeHandler = (event) => {

    setEnteredAge(event.target.value);

  };

  return (

    <Card className={classes.input}>

      <form onSubmit={addUserHandler}>

        <label htmlFor="username">Username</label>

        <input id="username" type="text" onChange={usernameChangeHandler} />

        <label htmlFor="age">Age (Year)</label>

        <input id="age" type="number" onChange={ageChangeHandler} />

        <Button type="submit">Add User</Button>

      </form>

    </Card>

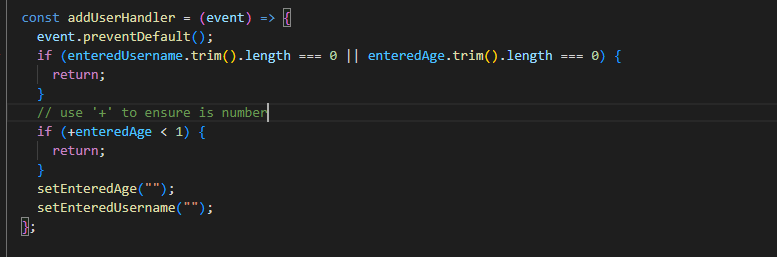
  );

};

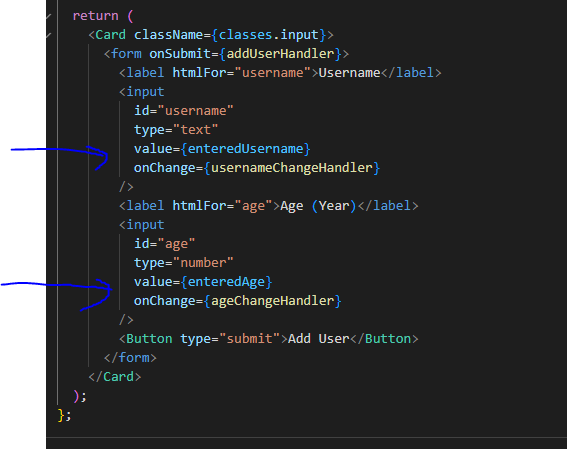
export default AddUser;

94. Adding Validation & Resetting Logic

Step 1 in AddUser change like this



Step 2 also in AddUser



95. Adding a Users List Component

Step 1 add UserList.module.css in folder Users like this

.users {

  margin: 2rem auto;

  width: 90%;

  max-width: 40rem;

}

.users ul {

  list-style: none;

  padding: 1rem;

}

.users li {

  border: 1px solid #ccc;

  margin: 0.5rem 0;

  padding: 0.5rem;

}

Step 2 Add UserList.js

import React from "react";

import Card from "../UI/Card";

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

const UsersList = (props) => {

    console.log(props.users)

    return (

      <Card className={classes.users}>

        <ul>

          {props.users && props.users.map((user) => (

            <li key={user.id}>

              {user.name} ({user.age} years old)

            </li>

          ))}

        </ul>

      </Card>

    );

  };

  export default UsersList;

Step 3 in App.js add like this

import React from 'react';

import AddUser from './components/Users/AddUser';

import UsersList from './components/Users/UsersList';

function App() {

  return (

    <div>

        <AddUser />

        <UsersList users={[]}/>

    </div>

  );

}

export default App;

96. Managing a List Of Users via State

Step 1: in App.js add new function addUserHandler and pass like this

import React, { useState } from "react";

import AddUser from "./components/Users/AddUser";

import UsersList from "./components/Users/UsersList";

function App() {

  const [usersList, setUsersList] = useState([]);

  const addUserHandler = (uName, uAge) => {

    setUsersList((prevUsersList) => {

      return [

        ...prevUsersList,

        { name: uName, age: uAge, id: Math.random().toString() },

      ];

    });

  };

  return (

    <div>

      <AddUser onAddUser={addUserHandler} />

      <UsersList users={usersList} />

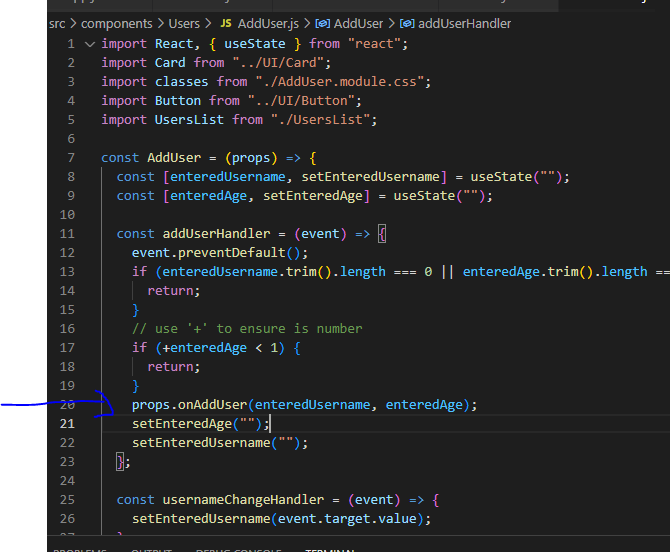
    </div>

  );

}

export default App;

Step 2 in AddUser.js add like this



97. Adding the ‘ErrorModal’ Component

Step 1 add ErrorModal.module.css in folder UI

.backdrop {

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    height: 100vh;

    z-index: 10;

    background: rgba(0, 0, 0, 0.75);

  }

  .modal {

    position: fixed;

    top: 30vh;

    left: 10%;

    width: 80%;

    z-index: 100;

    overflow: hidden;

  }

  .header {

    background: #4f005f;

    padding: 1rem;

  }

  .header h2 {

    margin: 0;

    color: white;

  }

  .content {

    padding: 1rem;

  }

  .actions {

    padding: 1rem;

    display: flex;

    justify-content: flex-end;

  }

  @media (min-width: 768px) {

    .modal {

      left: calc(50% - 20rem);

      width: 40rem;

    }

  }

Step 2 Add ErrorModal.js

import React from "react";

import Card from "./Card";

import Button from "./Button";

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

const ErrorModal = (props) => {

  return (

    <div>

      <div className={classes.backdrop} />

      <Card className={classes.modal}>

        <header className={classes.header}>

          <h2>{props.title}</h2>

        </header>

        <div className={classes.content}>

          <p>{props.message}</p>

        </div>

        <footer className={classes.actions}>

          <Button>Okay</Button>

        </footer>

      </Card>

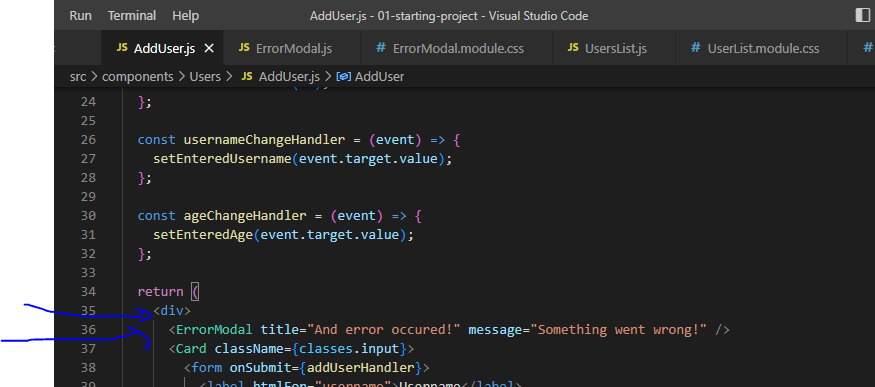
    </div>

  );

};

export default ErrorModal;

Step 3 add new ErrorModal in AddUser.js and around div



98. Managing the Error State

Step 1 Change AddUser.js like this

import React, { useState } from "react";

import Card from "../UI/Card";

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

import Button from "../UI/Button";

import UsersList from "./UsersList";

import ErrorModal from "../UI/ErrorModal";

const AddUser = (props) => {

  const [enteredUsername, setEnteredUsername] = useState("");

  const [enteredAge, setEnteredAge] = useState("");

  const [error, setError] = useState();

  const addUserHandler = (event) => {

    event.preventDefault();

    if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {

      setError({

        title: "Invalid input",

        message: "Please enter a valid name and age",

      });

      return;

    }

    // use '+' to ensure is number

    if (+enteredAge < 1) {

      setError({

        title: "Invalid age",

        message: "Please enter a valid age (> 0)",

      });

      return;

    }

    props.onAddUser(enteredUsername, enteredAge);

    setEnteredAge("");

    setEnteredUsername("");

  };

  const usernameChangeHandler = (event) => {

    setEnteredUsername(event.target.value);

  };

  const ageChangeHandler = (event) => {

    setEnteredAge(event.target.value);

  };

  const errorHandler = () => {

    setError(null);

  }

  return (

    <div>

      {error && <ErrorModal title={error.title} message={error.message} onConfirm={errorHandler}/>}

      <Card className={classes.input}>

        <form onSubmit={addUserHandler}>

          <label htmlFor="username">Username</label>

          <input

            id="username"

            type="text"

            value={enteredUsername}

            onChange={usernameChangeHandler}

          />

          <label htmlFor="age">Age (Year)</label>

          <input

            id="age"

            type="number"

            value={enteredAge}

            onChange={ageChangeHandler}

          />

          <Button type="submit">Add User</Button>

        </form>

        <UsersList />

      </Card>

    </div>

  );

};

export default AddUser;

Step 2 in ErrorModal.js

import React from "react";

import Card from "./Card";

import Button from "./Button";

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

const ErrorModal = (props) => {

  return (

    <div>

      <div className={classes.backdrop} onClick={props.onConfirm}/>

      <Card className={classes.modal}>

        <header className={classes.header}>

          <h2>{props.title}</h2>

        </header>

        <div className={classes.content}>

          <p>{props.message}</p>

        </div>

        <footer className={classes.actions}>

          <Button onClick={props.onConfirm}>Okay</Button>

        </footer>

      </Card>

    </div>

  );

};

export default ErrorModal;

Step 3 done

Graphical user interface, application, Teams

Description automatically generated

99.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/08-practice-project>

Section 9: Diving Deeper Working with Fragments

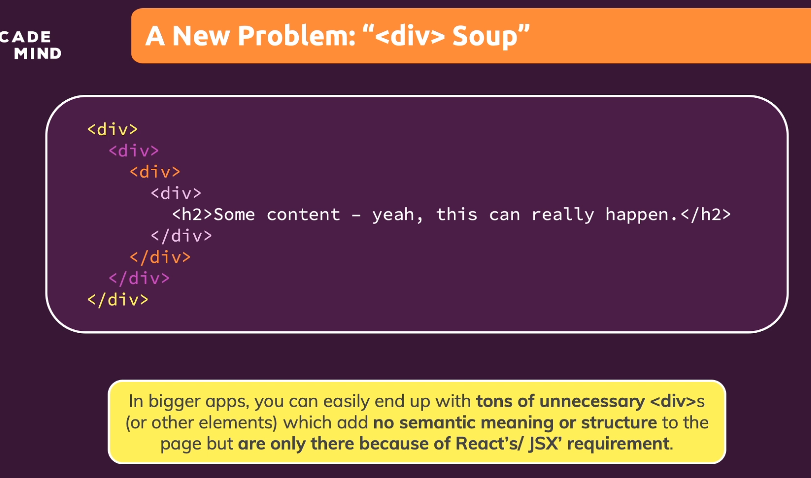


101. JSX Limitations & Workarounds



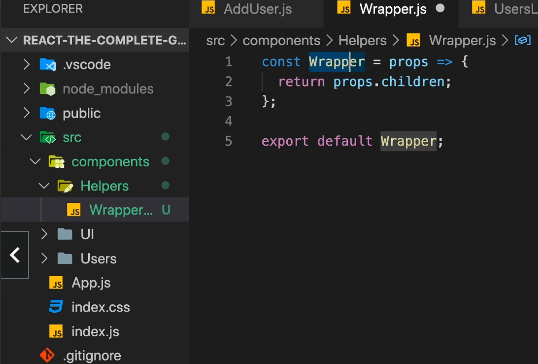
102. Creating a Wrapper Component

Problem

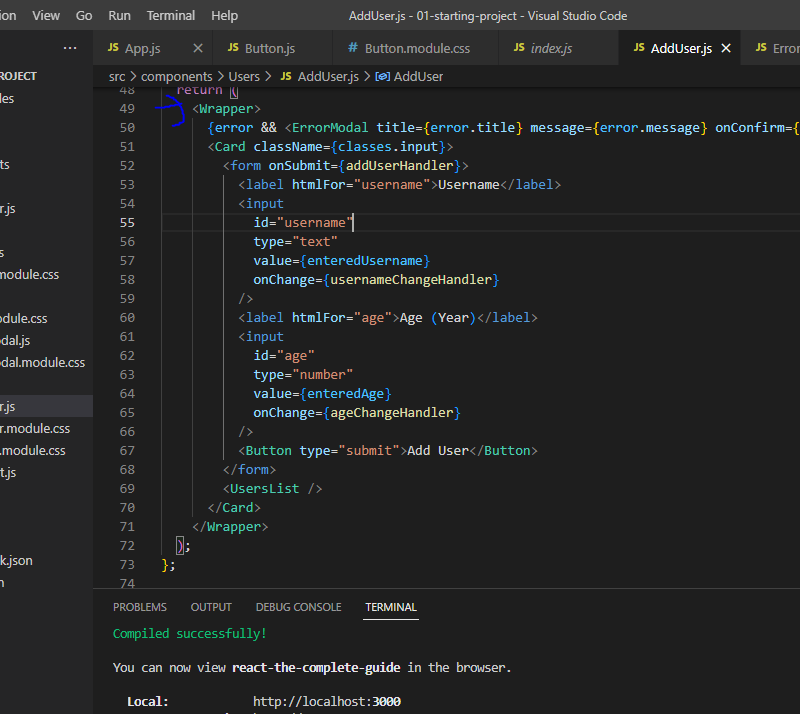


Step 1: create new Folder Helpers

Step 2: create Wrapper.js



Step 3: In AddUser.js change div to Wrapper



Step 4: Solve Problem

Before

Graphical user interface, text, application

Description automatically generated

After

Graphical user interface, text, application

Description automatically generated

103. React Fragments (another solution for 102)

Fragment là một common pattern được giới thiệu kể từ khi React 16 ra đời. Nó cho phép bạn return nhiều element từ một component mà không làm sinh ra những DOM element ko cần thiết.

## Sự ra đời

Dưới đây là một common pattern của một React component trả về một list các children:

class Table extends React.Component {

render() {

return (

<table>

<tr>

<Columns />

</tr>

</table>

);

}

}

<Columns /> component sẽ phải trả về nhiều phần tử <td> để đầu ra HTML hiển thị hợp lệ. Nếu một div cha được sử dụng bên trong render() của <Columns /> component, thì sẽ cho ra kết quả HTML không hợp lệ.

<Table /> component lúc này sẽ có dạng:

<table>

<tr>

<div>

<td>Hello</td>

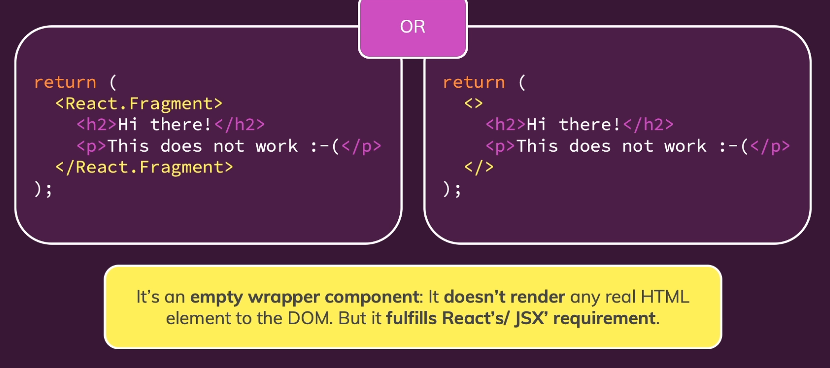
<td>World</td>

</div>

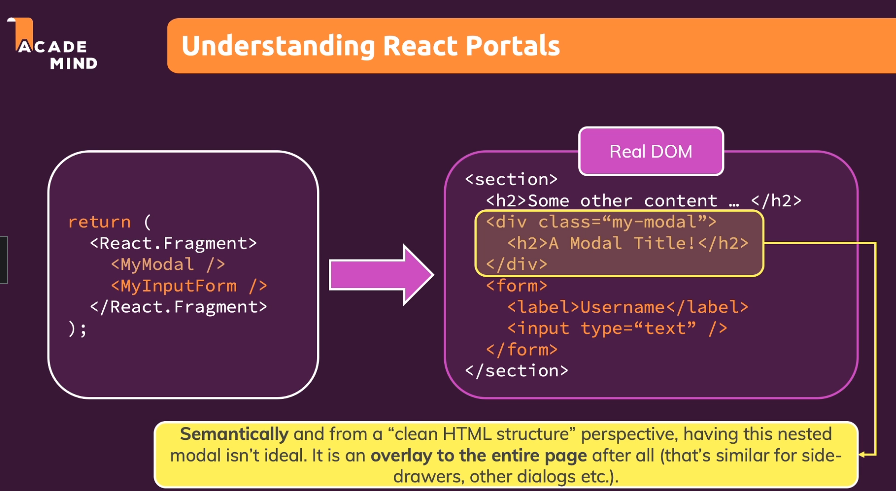
</tr>

</table>

Chính vì thế, fragment được giới thiệu để giải quyết điều này.(có 2 loại)



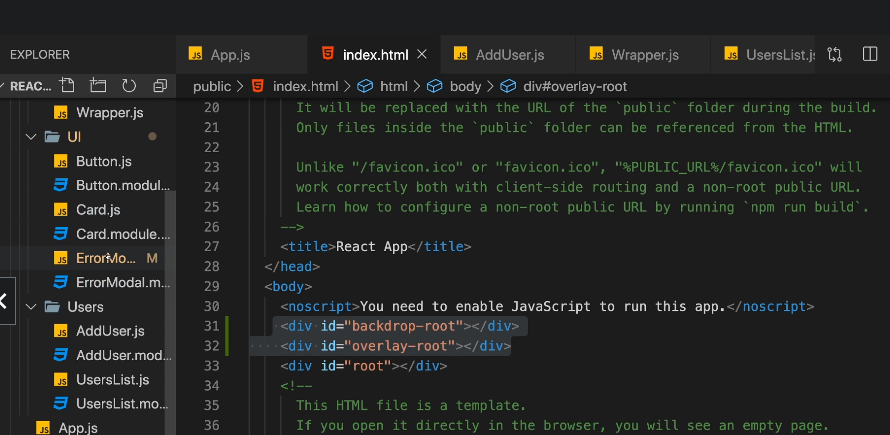
104. Introducing React Portals



105. Working With Portals

Move div to another component or move to first html (to make sure html no complicate and some function depper in div)

Step 1 in index.html, add like this



Step 2 in ErrorModal.js

import React from 'react';

import ReactDOM from 'react-dom';

import Card from './Card';

import Button from './Button';

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

const Backdrop = (props) => {

  return <div className={classes.backdrop} onClick={props.onConfirm} />;

};

const ModalOverlay = (props) => {

  return (

    <Card className={classes.modal}>

      <header className={classes.header}>

        <h2>{props.title}</h2>

      </header>

      <div className={classes.content}>

        <p>{props.message}</p>

      </div>

      <footer className={classes.actions}>

        <Button onClick={props.onConfirm}>Okay</Button>

      </footer>

    </Card>

  );

};

const ErrorModal = (props) => {

  return (

    <React.Fragment>

      {ReactDOM.createPortal(

        <Backdrop onConfirm={props.onConfirm} />,

        document.getElementById('backdrop-root')

      )}

      {ReactDOM.createPortal(

        <ModalOverlay

          title={props.title}

          message={props.message}

          onConfirm={props.onConfirm}

        />,

        document.getElementById('overlay-root')

      )}

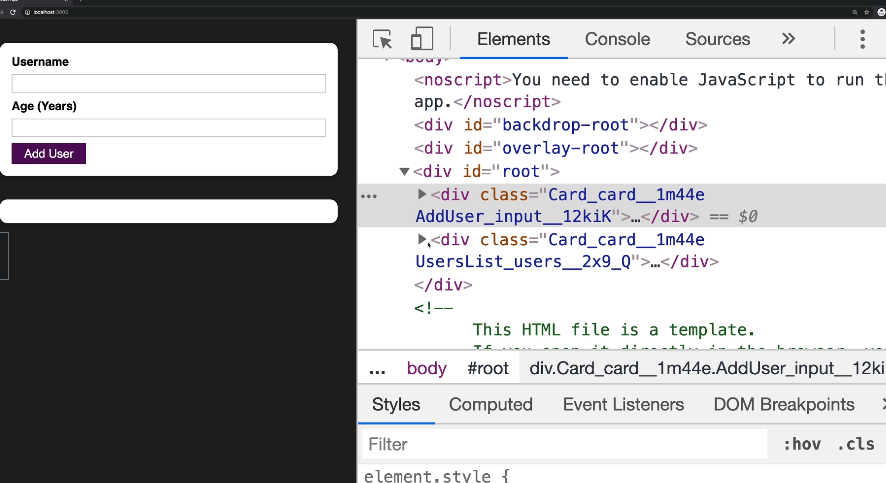
    </React.Fragment>

  );

};

export default ErrorModal;

Result



106. Working with “ref’’s

Allow read current value or set value without use useState

Step 1 in AddUser.js

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

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

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

import ErrorModal from '../UI/ErrorModal';

import Wrapper from '../Helpers/Wrapper';

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

const AddUser = (props) => {

  const nameInputRef = useRef();

  const ageInputRef = useRef();

  const [error, setError] = useState();

  const addUserHandler = (event) => {

    event.preventDefault();

    const enteredName = nameInputRef.current.value;

    const enteredUserAge = ageInputRef.current.value;

    if (enteredName.trim().length === 0 || enteredUserAge.trim().length === 0) {

      setError({

        title: 'Invalid input',

        message: 'Please enter a valid name and age (non-empty values).',

      });

      return;

    }

    if (+enteredUserAge < 1) {

      setError({

        title: 'Invalid age',

        message: 'Please enter a valid age (> 0).',

      });

      return;

    }

    props.onAddUser(enteredName, enteredUserAge);

    nameInputRef.current.value = '';

    ageInputRef.current.value = '';

  };

  const errorHandler = () => {

    setError(null);

  };

  return (

    <Wrapper>

      {error && (

        <ErrorModal

          title={error.title}

          message={error.message}

          onConfirm={errorHandler}

        />

      )}

      <Card className={classes.input}>

        <form onSubmit={addUserHandler}>

          <label htmlFor="username">Username</label>

          <input id="username" type="text" ref={nameInputRef} />

          <label htmlFor="age">Age (Years)</label>

          <input id="age" type="number" ref={ageInputRef} />

          <Button type="submit">Add User</Button>

        </form>

      </Card>

    </Wrapper>

  );

};

export default AddUser;

107. Controlled vs Uncontrolled Components

When you use ref, you will uncontrolled, but when use useState you will ‘controlled’

108. 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/09-fragments-portals-refs>