섹션 9: 심화학습 더 깊이 다이빙: 프래그먼트 작업, Portals & “Refs”

작성일시	@March 2, 2023 12:58 AM
강의날짜	@2023/03/01
편집일시	@March 2, 2023 1:49 AM
분야	React
공부유형	강의
복습	☐
태그	

110. JSX 제한 사항 및 해결 방법

- 루트 JSX 요소는 1개여야 함
- 자바스크립트 Return에서는 하나만 반환할 수 있음
- <div>로 감싸야함

<div> Soup

- 최종 HTML 페이지에 불필요한 div 또는 기타 요소가 많이 있을 수 있음
- 애플리케이션은 늦어질 수 있음

111. 컴포넌트 감싸기(wrapper) 만들기

```
// Wrapper.js
const Wrapper = props => {
  return props.children;
};

export default Wrapper;

// <div> -> <Wrapper>
```

112. 리액트 조각

- Wrapper를 직접 만들지 않아도 됨
- `<> </>` 를 사용할 수 있음

- `<React.Fragment> <React.Fragment />`
- `import fragment from react` 후 `<Fragment> <Fragment/>` 도 가능함

114. 포털 작업하기

```
// Index.html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
// [1] 포털용 div 만들기
    <div id="backdrop-root"></div>
    <div id="overlay-root"></div>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>

// 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;
```

115. “Ref”로 작업하기

- 코드가 짧아짐

```
// AddUser.js
import React, { useState, useRef } from 'react';

import Card from '../UI/Card';
import classes from './AddUser.module.css';
import Button from '../UI/Button';
import ErrorModal from '../UI/ErrorModal';
import Wrapper from '../Helpers/Wrapper';

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 input',
        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;
```