React

<React 설치>
SPA
single page
앵귤러 리액트 뷰

컴포넌트 함수형 클래스형
노드기반 https://nodejs.org/ko/download -> LTS

```
C:\Users\it>node -v
v18.17.1

C:\Users\it>npm -v
9.6.7

C:\Users\it>npx -v
9.6.7

C:\Users\it>
```

CRA creat react A

```
>npx create-react-app first-app

>npm uninstall -g create-react-app
>npm install create-react-app
>npx create-react-app first-app


C:\Users\it>cd C:\react_works

C:\react_works>cd first-app

C:\react_works\first-app>npm start
```
(안되면 재실행 하면 됨)

props 부모로부터 받아오는 값
state 현재 내자신에서 받아오는 값

<React 구성하기>
[App.js]

```
import logo from './logo.svg';
import './App.css';
import Device from "./Device";
```

```
import Counter from "./Counter";
import Food from './Food';

function App() {
  return (
    <div>
      <h1>안녕</h1>
      <h2>안녕</h2>
      <Device />
      <Counter />
      <Food />
    </div>
  );
}

export default App;
```

[Counter.js]

```
import  {Component} from 'react';

class Counter extends Component{
    constructor(props){
        super(props);
        this.state = {
            count : 0

        }
        this.increaseCount = this.increaseCount.bind(this);
        this.decreaseCount = this.decreaseCount.bind(this);

    }
    //증가
    increaseCount(){
        this.setState(({count}) => {
            return {
                count : count +1
            }
        });

    }
    //감소
    decreaseCount(){
        // this.setState(({count}) => {
        //     return {
        //         count : count -1
        //     }
        // });
```

```
        this.setState((st)=> {
            console.log("st : " + st.count)
            return {
                count : st.count -1
            }
        });

    }
    render(){
        return(
            <div>
                <span>카운트 :  {this.state.count} </span>
                <button onClick={this.increaseCount}>카운트 증가</button>
                <button onClick={this.decreaseCount}>카운트 감소</button>

            </div>



        )
    }
}

export default Counter;
```

[Food.js]

```
    const mydata = {
        "myfood" : [
            {
                name : "햄버거",
                price : "오천원",
                buy : "배달"
            },

            {
                name : "도시락",
                price : "육천원",
                buy : "포장"
            },

            {
                name : "커피",
                price : "삼천원",
                buy : "매장"
            }
        ]
```

```
}

const myphone = {
    "phone" : [
        {
            name : "아이폰",
            ram : "6GB",
            touch : "yes",
            face : "yes"
        },

        {
            name : "갤럭시 노트",
            ram : "8GB",
            touch : "yes",
            face : "yes"
        },

        {
            name : "갤럭시 S22",
            ram : "8GB",
            touch : "no",
            face : "yes"
        },

        {
            name : "갤럭시 z",
            ram : "64GB",
            touch : "yes",
            face : "yes"
        }
    ]
}

function Food() {
    return (
        <div>
            <h1> Food 컴포넌트(음식)</h1>
            {
                mydata.myfood.map((food, i) => (
                    <div key={i}>
                        이름 : {food.name}<br />
                        가격 : {food.price}<br />
                        방법 : {food.buy}<br /><br />
                    </div>
                ))
            }
```

```
                <h1>Food 컴포넌트(폰)</h1>
                {
                    myphone.phone.map((p, j) => (
                        <div key={j}>
                            이름 : {p.name}<br />
                            램 : {p.ram}<br />
                            터치 : {p.touch}<br />
                            안면인식 : {p.face}<br /><br />
                        </div>
                    ))
                }
            </div>
        );
    }

    export default Food;
```

<구조 분해 할당>
구조 분해 할당(Destructuring Assignment)은 JavaScript에서 배열이나 객체를 해체하여 그 값을 개별 변수에 할당하는 문법입니다. 이를 통해 코드를 더 간결하게 작성하고 변수를 쉽게 선언하고 초기화할 수 있습니다.

<스프레드 연산자>
스프레드 연산자(...)는 JavaScript에서 객체와 배열을 확장하거나 병합하는 데 사용되는 유용한 기능입니다. 이 연산자를 사용하면 기존 객체나 배열의 요소를 새로운 객체나 배열에 쉽게 복사하거나 추가할 수 있음.

<State란?>
state는 React 컴포넌트의 내부 데이터 저장소입니다. React 컴포넌트는 상호작용성과 동적인 데이터 표시를 위해 state를 사용합니다. 각 컴포넌트는 자체 state를 가질 수 있으며, state가 변경될 때 컴포넌트는 다시 렌더링됩니다.

state의 주요 특징과 사용법은 다음과 같습니다:

내부 데이터 저장소: state는 컴포넌트의 내부에 저장되며 컴포넌트의 상태를 나타냅니다. 이를 통해 컴포넌트는 데이터를 보유하고 화면에 표시할 수 있습니다.

가변성: state는 가변(mutable)하며 변경될 수 있습니다. 컴포넌트가 상태를 변경하면 React는 변경 사항을 감지하고 필요한 경우 컴포넌트를 다시 렌더링합니다.

setState 메서드: state를 변경하기 위해서는 setState 메서드를 사용해야 합니다. 이 메서드를 호출하면 React에게 새로운 상태를 설정하고 컴포넌트를 다시 렌더링하도록 알립니다.

비동기 업데이트: setState 호출은 비동기적으로 처리됩니다. 이것은 상태 업데이트가 즉시 반영되지 않을 수 있음을 의미합니다. React는 여러 setState 호출을 배치 처리하여 최적화합니다.

컴포넌트 라이프사이클: state 변경은 React 컴포넌트 라이프사이클 메서드 중 하나에서 이루어집니다. 주로 componentDidMount, componentDidUpdate 등의 메서드에서 상태를 변경하고 처리합니다.

단일 소유원칙: React에서는 각 데이터를 변경할 때 하나의 컴포넌트가 state를 소유하고 업데이트하는 것이 좋습니다. 다른 컴포넌트에서 state를 직접 수정하면 예측 불가능한 동작이 발생할 수 있습니다.

state를 사용하여 컴포넌트의 상태를 관리하면 데이터의 동적인 표시, 상호작용 및 UI 업데이트를 간단하게 구현할 수 있습니다. React 컴포넌트에서 state를 효과적으로 활용하면 동적 웹 애플리케이션을 개발하는 데 도움이 됩니다.

나머지 사항들은 소스코드 참조

---

componentDidMount, componentDidUpdate, useState, Hook 등은 소스 코드를 참조바랍니다.

<React 패키지 설치>
npm install axios react-router-dom bootstrap react-bootstrap

package.json - npm 설치 버전 확인하는곳

https://yts.mx/api/v2/list_movies.json?sort_by=rating
영화 정보 API

나머지 사항들은 소스코드 참조