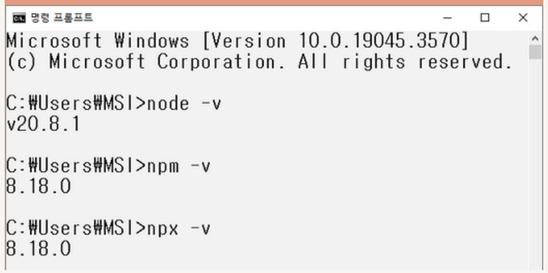
1

2



3 : 설치 명령어 작성

```
PS C:\Users\MSI\ezen_webspring_2023_A> cd src/main
PS C:\Users\MSI\ezen_webspring_2023_A\src\main> npx create-react-app reactapp
```

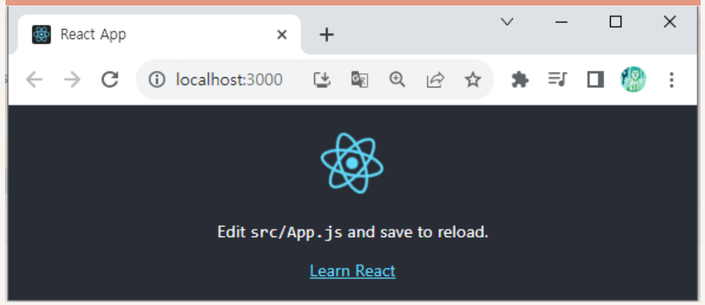
4 : 대략 5분이내 설치 완료

```
Success! Created reactapp at C:\Users\MSI\ezen_webspring_2023_A\src\main\reactapp
Inside that directory, you can run several commands:
```

5 : 실행 명령어 작성

```
PS C:\Users\MSI\ezen_webspring_2023_A> cd src/main/reactapp
PS C:\Users\MSI\ezen_webspring_2023_A\src\main\reactapp> npm start
```

6 : 실행 결과



Node.js 설치

1. https://nodejs.org/ko
2. [23/10/17] node 20.8.1 현재 버전 설치
3. 명령프롬프트[cmd] 에서 3가지 명령어 작성후 버전 확인
    a. node -v v16.x.x 이상이면 됩니다.
    b. npm -v v8.x.x 이상이면 됩니다.
    c. npx -v v8.x.x 이상이면 됩니다.

React.js 설치

1. 인텔리제이 터미널[alt + f12]
2. 명령어 작성 : main폴더로 이동후 react 프로젝트 생성
    a. cd src/main
    b. npx create-react-app 프로젝트명
3. 대략 5분이내 설치 완료
    a. 오류 발생시 : npm uninstall -g create-react-app

React.js 실행

1. 인텔리제이 터미널[alt + f12]
2. 명령어 작성 : react프로젝트폴더로 이동후 실행
    a. cd src/main/프로젝트명 또는 cd 프로젝트명
    b. npm start [pc가 변경되고 git clone 될때 npm update 한번해주세요]
3. 웹페이지가 자동으로 열리면서 http:/localhost:3000 접속됩니다.

1

```jsx
function 컴포넌트1() {
    return <h1>함수반환값은 HTML 형식입니다.</h1>
}
export default 컴포넌트1;

```

2

index.js × 1_컴포넌트.jsx

```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

import 컴포넌트1 from './component/example/day01/1_컴포넌트';

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

//root.render( <React.StrictMode> <App /> </React.StrictMode> );
root.render( <React.StrictMode> <컴포넌트1 /> </React.StrictMode> );

reportWebVitals();

```

tip] JSX확장자를 이용한 함수형 컴포넌트 원형

----------파일명.jsx 파일 생성----------

function 컴포넌트명( ) {

    return HTML문법작성

}

export default 컴포넌트명;

---

why) export default 란??

- 해당 jsx파일을 import 하면 반환해줄 컴포넌트(함수)명을 뜻합니다.

예제1] JSX파일 형식으로 컴포넌트(함수) 만들기

1. 1_컴포넌트.jsx 파일 생성
2. 함수(컴포넌트) 정의
3. return 에 HTML 문법을 작성
4. 함수 밖에 export default 컴포넌트명

예제1] index.js에서 컴포넌트 랜더링(실행) 하기

1. index.js 파일 열기
2. 컴포넌트 호출하기 : 7번줄에 상대경로로 작성하여 컴포넌트 호출
    a. import 컴포넌트명(함수명) from 상대경로/상대경로/jsx파일명
3. 11번줄에 기존 코드를 주석처리
4. 12번줄에 7번줄에서 import한 컴포넌트명을 작성하여 랜더링에 적용

1

index.js × 2_컴포넌트.jsx ×

```jsx
function 컴포넌트2() {
    return (
        <>
            <h1>함수반환값은 HTML 형식입니다.</h1>
            <h1>HTML 작성코드가 2줄이상일때는 (<> </>)소괄호와 기본태그 또는 div 로 묶어주세요</h1>
        </>
    )
}
export default 컴포넌트2;
```

2

index.js ×

```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

import 컴포넌트1 from './component/example/day01/1_컴포넌트';
import 컴포넌트2 from './component/example/day01/2_컴포넌트';

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

//root.render( <React.StrictMode> <App /> </React.StrictMode> );
//root.render( <React.StrictMode> <컴포넌트1 /> </React.StrictMode> );
root.render( <React.StrictMode> <컴포넌트2 /> </React.StrictMode> );

reportWebVitals();
```

tip] HTML문법이 2줄 이상일때

return (<> HTML 작성 <>)

또는

return (HTML 작성)

예제2] JSX 형식에서 HTML 작성하는 방법

1. 2_컴포넌트.jsx 파일 생성
2. 예제1 처럼 HTML 문법이 한줄일때는 제약이 없다.
3. 예제2 처럼 HTML 문법이 여러줄일때는 아래와 같이 작성한다
    a. return (<> HTML 작성 <>)
    b. return (HTML 작성)

예제2] index.js에서 컴포넌트 랜더링(실행) 하기

1. index.js 파일 열기
2. 8번줄에 컴포넌트를 상대경로로 작성
    a. import 컴포넌트명(함수명) 상대경로/상대경로/상대경로/jsx파일명
3. 11번/13번 줄에 기존 코드를 주석처리
4. 14번줄에 8번줄에서 import한 컴포넌트명을 작성하여 랜더링에 적용

1

```jsx
function 컴포넌트3() {
    return (
        <>
            <h1> 다른 컴포넌트 호출해보기 </h1>
            <내가만든태그 />
        </>
    )
}

function 내가만든태그() {
    return <div> --내가만든태그-- 에서 작성된 HTML 입니다. </div>
}

export default 컴포넌트3;
```

2

index.js × 3_컴포넌트.jsx

```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

import 컴포넌트1 from './component/example/day01/1_컴포넌트';
import 컴포넌트2 from './component/example/day01/2_컴포넌트';
import 컴포넌트3 from './component/example/day01/3_컴포넌트';

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

//root.render( <React.StrictMode> <App /> </React.StrictMode> );
//root.render( <React.StrictMode> <컴포넌트1 /> </React.StrictMode> );
//root.render( <React.StrictMode> <컴포넌트2 /> </React.StrictMode> );
root.render( <React.StrictMode> <컴포넌트3 /> </React.StrictMode> );


reportWebVitals();
```

tip] 컴포넌트 내부에서 다른 컴포넌트 호출 하는 방법

return (〈〉

〈호출할컴포넌트명 /〉

〈〉)

예제3] JSX파일에 컴포넌트 내부에서 다른 컴포넌트 호출

1. 3_컴포넌트.jsx 파일 생성
2. 컴포넌트3(함수) 선언 하고 적절하게 HTML 작성한다.
3. 내가만든태그(함수) 선언 하고 적절하게 HTML 작성한다.
4. 5번에서 다른 컴포넌트(함수) 호출하기
    a. 〈컴포넌트명 /〉

예제3] index.js에서 컴포넌트 실행하기

1. index.js 파일 열기
2. 9번줄에 컴포넌트를 상대경로로 작성
    a. import 컴포넌트명(함수명) 상대경로/상대경로/상대경로/jsx파일명
3. 13~15 번줄에 기존 코드를 주석처리
4. 16번줄에 9번줄에서 import한 컴포넌트 작성하여 랜더링에 적용

1

index.js 4_컴포넌트.jsx

```jsx
function 컴포넌트4() {
    return (
        <>
            <h1> 다른 컴포넌트 호출해보기 </h1>
            <내가만든태그에속성 내용="컴포넌트4에서 넣어준 내용" />
        </>
    )
}
function 내가만든태그에속성( props ) {
    return <div> -- 전달된 객체 : { props.내용 }-- 에서 작성된 HTML 입니다. </div>
}
export default 컴포넌트4;
```

2

index.js 4_컴포넌트.jsx

```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

import 컴포넌트1 from './component/example/day01/1_컴포넌트';
import 컴포넌트2 from './component/example/day01/2_컴포넌트';
import 컴포넌트3 from './component/example/day01/3_컴포넌트';
import 컴포넌트4 from './component/example/day01/4_컴포넌트';

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

//root.render( <React.StrictMode> <App /> </React.StrictMode> );
//root.render( <React.StrictMode> <컴포넌트1 /> </React.StrictMode> );
//root.render( <React.StrictMode> <컴포넌트2 /> </React.StrictMode> );
//root.render( <React.StrictMode> <컴포넌트3 /> </React.StrictMode> );
root.render( <React.StrictMode> <컴포넌트4 /> </React.StrictMode> );

reportWebVitals();
```

tip] 다른 컴포넌트에게 데이터(객체) 전달하기

```jsx
function 컴포넌트A{
  return ( <>
      <컴포넌트B  속성명="문자데이터"  속성명={숫자데이터}  />
  </> )
}
function 컴포넌트B(props){
   console.log( props );
}
```

---

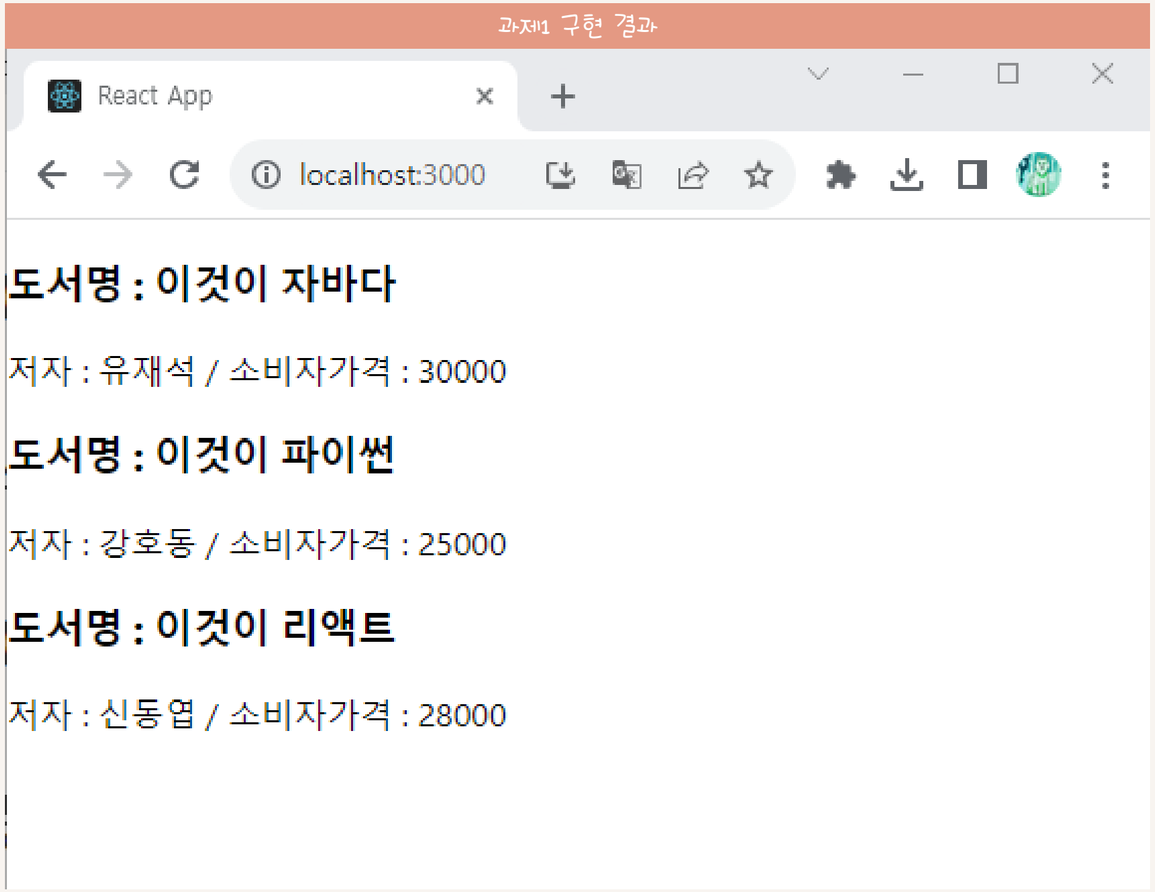
why) props 란?

- 다른 컴포넌트가 전달한 데이터들로 구성된 객체 입니다.
- <컴포넌트B 이름="유재석" 나이={30} /> 전달 할 경우
- 컴포넌트B props 에는 { 이름:유재석 , 나이 : 30 } 이 전달되어 저장 됩니다.
- HTML 문법내 JS문법을 사용시에는 { 객체명.속성명 } 형식으로 사용합니다.

예제4] JSX파일에 컴포넌트 내부에서 다른 컴포넌트 호출

1. 4_컴포넌트.jsx 파일 생성
2. 컴포넌트4(함수) 선언 하고 적절하게 HTML 작성한다.
    a. 컴포넌트4 내부에서 내가만든태그에속성 를 호출합니다.
    b. <내가만든태그에속성 />에 속성에 데이터를 넣어줍니다.
        i. <컴포넌트명 속성명 =데이터 />
3. 내가만든태그에속성 선언하고 props 매개변수 선언 합니다.
    a. 컴포넌트4 로 부터 전달받은 props 객체를 { } 이용하여 출력합니다.

과제1 구현 결과



과제1

1. 과제1_도서목록.jsx 파일 생성합니다.
2. 생성된 jsx파일 내 컴포넌트를 2개 선언합니다.
    a. 도서목록 컴포넌트가 도서 컴포넌트를 3번 호출합니다.
    b. 도서목록 컴포넌트가 도서 컴포넌트를 호출할때
        i. 도서명 , 저자 , 소비자가격 3가지 속성과 데이터를 전달합니다.
    c. index.js에서 도서목록 컴포넌트를 랜더링 하여 실행합니다.
3. 제출
    a. 구현된 화면 캡처후 수업 카카오톡방에 제출