3강

Jsx의 정의와 역할

➢ JSX(JavaScript XML)

```
const element = <h1> Hellow, World1<h1>;
```

- 위 코드는 문자열도 HTML도 아닌 JSX 이다.
- JSX를 작성하면 React 엘리먼트 객체가 생성된다. 즉, 브라우저 DOM이 아닌 JS 객체이다.
- JSX는 필수는 아니지만 작성하면 가독성도 좋고, React가 제공하는 경고/에러 메시지 확인이 용이하다.

➢ JSX(JavaScript XML) 변환 과정

- JSX는 브라우저에서 실행되기 전에 코드가 **번들링(Bundleling = 그룹화 작업)** 되는 과정에서 **바벨(babel = 트랜스파일러의 역할로 호환되는 버전으로 변경)**을 사용하여 자바스크립트 형태로 변환된다.

```
// JSX
const element = (
        <h1 className="greeting">
          Hello, world!
        </h1>
);

// JS로 변환된 JSX
const element = React.createElement(
    'h1',
    {className: 'greeting'},
    'Hello, world!'
);
```

•JSX를 사용하지 않고 React.createElement() 함수를 사용하면 컴포넌트를 렌더링 할 수 있다. 하지만 이 방식은 JSX를 사용하는 방식보다 불편하다. JSX를 사용하면 쉽고 편하게 UI를 렌더링 할 수 있다.

➢ ReactDOM.render(element, container[, callback])

- Render기본 의미는 **무엇인가를 (코드 등) 그려내거나 지금과는 다른 어떤 상태로 만든다는 뜻을** 가지고 있다.
- 개발자가 작성한 JSX를 화면에 렌더링 하기 위해서는 ReactDOM.render( ) 함수를 사용해야한다. 이 함수는 컴포넌트(Component)를 페이지에 렌더링하는 역할을 하며, react-dom 모듈을 불러와 사용할 수 있다.
- element : JSX로 작성한 화면에 출력할 내용
- container : 첫 번째 인자인 JSX를 랜더링 해서 보여줄 DOM 안의 위치

예

```
ReactDom.render(
        <h1>Hello World</h1>,
      document.body
);
```

- 위 함수를 이용하면 최종결과가 HTML, CSS, JS로 만들어지고, 이를 브라우저가 이해할 수 있어 원하는 결과물이 출력된다.

➢ JSX(JavaScript XML) 장점

1.보기 쉽고 익숙하다.

1. JSX는 HTML 코드와 비슷하기 때문에 일반 자바스크립만 사용한 코드보다 더 익숙하며 **가독성이 좋다.**

2.높은 활용도

1. JSX에는 div, span 같은 HTML 태그를 사용할 수 있으며, 개발자가 만든 컴포넌트도 JSX 안에서 작성할 수 있다.

➢ JSX(JavaScript XML) 문법

- **반드시 태그는 닫혀야 한다.**
  - , <p>, , <a> 같이 짝이 있는 태그의 경우 **반드시 닫는 태그가 존재**해야 한다. 그렇지 않을 경우 에러가 발생한다.
  - <img/>, <input/>,
 같은 단독 태그(self-closing tag)의 경우에는 반드시 태그를 닫아줘야 한다. 그렇지 않을 경우 에러가 발생한다.
- **렌더링 될 루트 엘리먼트는 하나만 존재 해야 한다.**

➢ JSX(JavaScript XML) 문법

잘못된 예

```
ReactDOM.render(
      <div> Hello </div>
       <div> Bye </div>,
        document.getElementById('root')
 );
```

```
ReactDOM.render(
    <div>
        <div> Hello </div>
        <div> Bye </div>,
     </div>,
       document.getElementById('root')
 );
```

- 왼쪽의 예제처럼 렌더링 될 리액트 엘리먼트에서 루트 엘리먼트가 두 개 이상일 경우 에러가 발생한다. 때문에 두 개 이상의 루트 엘리먼트가 존재할 경우 오른쪽과 같이 반드시 하나의 엘리먼트로 감싸져야 한다. (JSX는 자식 엘리먼트를 가질 수 있다.)
- Virtual DOM에서 컴포넌트 변화를 감지해 낼 때 효율적으로 비교할 수 있도록 컴포넌트 내부는 하나의 DOM 트리 구조로 이루어져야 한다는 규칙 때문에, **리액트에서는 반드시 컴포넌트에 여러 요소가 있다면 반드시 부모 요소로 감싸야 한다.**

➢ 복수 엘리먼트 리턴

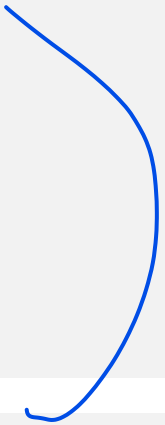
```
배열 문법 사용
ReactDOM.render(
   [ // 리액트 엘리먼트 배열을 반환
      <div key="1">Hello</div>, // 복수의 아이템을 리턴할 경우
      <div key="2">Bye</div> // 각 엘리먼트에 key 속성과 고유값을 지정해야 한다.
   ],
      document.getElementById('root')
);
```

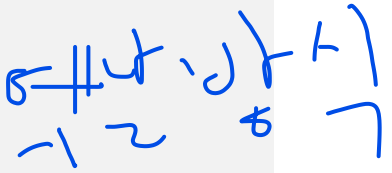
➢ 리액트 v16.2.0 부터 추가된 프래그먼트(Fragment) 패턴 사용

```
ReactDOM.render(
  <React.Fragment> // React.Fragment 컴포넌트로 리액트 엘리먼트를 감싸준다.
    <div>Hello</div>
    <div>Bye</div>
 </React.Fragment>
   document.getElementById('root')
);
```

➢ 리액트 v16.2.0 부터 추가된 프래그먼트(Fragment) 패턴 사용

```
ReactDOM.render(
 <> // 또는 빈 태그로 감싸준다.
   <div>Hello</div>
   <div>Bye</div>
 </>
  document.getElementById('root')
);
```





- React.Fragment 컴포넌트는 실제로 DOM엘리먼트로 생성되지 않는다. 단지 HTML로 트랜스파일될 때 존재하지 않는 것으로 취급하라고 JSX에게 알려준다.
- 각 엘리먼트들이 배열에 담겨져 있는 것이 아니므로 쉼표나 구분자가 필요 없다.
- key속성과 고유 값을 지정할 필요가 없다.

JSX(JavaScript XML) 장점 및 문법

➢ JSX에서는 중괄호 { } 를 사용하여 JS 표현식을 쓸 수 있다.

```
const name = 'Josh Perez';
const element = <h1>Hello, {name} </h1>; // Hello, Josh Perez
ReactDOM.render(
    element,
    document.getElementById('root')
 );
//-------------------------------------------------------------------------
ReactDOM.render(
     <p>Random number : {Math.random() * 100} </p>,
      document.getElementById('root')
 );
// 중괄호는 표현식 먼저 평가 돼 그 결과를 리턴하게 만든다.

//주석 사용하기
// JSX에서 주석은 {  }로 감싸준다. 여러 줄 주석만 사용 가능하다.
ReactDOM.render(
   <div>
      <p>Hello</p>
      {/* 주석은 이렇게 작성합니다. */}
      <p
          className='react' // 시작 태그를 여러줄로 작성 시 주석 작성 가능 >World</p>
      // 하지만 이런 주석이나
      /* 이런 주석은 페이지에 그대로 노출됩니다.*/
   </div>,
   document.getElementById('app')
);
```

➢ 삼항 연산자(조건부 연산자)를 사용한 조건부 렌더링

- JSX 내부의 JS 표현식에서는 **if문을 사용할 수 없다.** 때문에 조건에 따라 다른 내용을 렌더링 하고자 할 경우 JSX 밖에서 if 문을 사용하거나, 중괄호 안에서 **삼항 연산자**를 사용하면 된다.

```
class App extends Component {
render() {
    let name = 'React';
     return (
      <div>
          {
            name === 'React' ? ( <h1>This is React.</h1> ) : ( <h1>This is not React.</h1> )
          }
      </div>
      );
   }
}
```

True

False

➢ React.createElement( )

- React.createElement( )는 버그가 없는 코드를 작성하는 데 도움이 되도록 몇 가지 검사를 수행 후, 기본적으로 다음과 같은 객체를 생성한다.

```
const element = {
    type: 'h1',
    props: {
    className: 'greeting',
    children: 'Hello, world!'
  }

}
```

- 이렇게 생성된 객체를 "React 엘리먼트"라고 하며, 이는 화면에 표시하려는 항목에 대한 설명이라고 할 수 있다.
- React는 이러한 객체를 읽고 DOM을 구성하고 최신으로 유지하는 데 이러한 객체를 사용한다.

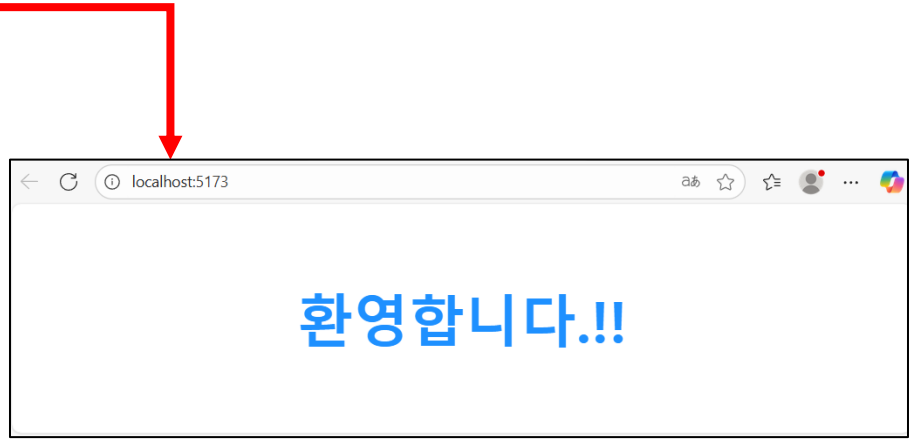
- 프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기
- 아래 코드와 같이 수정한다.

```jsx
function App() {


  return (
    <>
      <div>
        <h1>안녕하세요 React Vite 입니다.</h1>
      </div>
    </>
  )
}

export default App
```

- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- className 사용하기 "welcome"이라는 className을 가진 div 안에 "환영합니다"를 출력하세요

  (단, React는 Class="welcome"가 아니라 className="welcome"임을 주의하자!!)
- **"환영합니다" 글꼴 색을 color="dodgerblue"로 변경하는 CSS를 App.css 파일에 추가 하시오.**

```jsx
function App() {


  return (
    <>
      <div className='welcome'>
        <h1>환영합니다.!!</h1>
      </div>
    </>
  )
}

export default App
```



- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 변수를 선언하고 JSX에서 출력하세요.
- - 변수명: name - 값: "홍길동“ - 출력: "이름: 홍길동

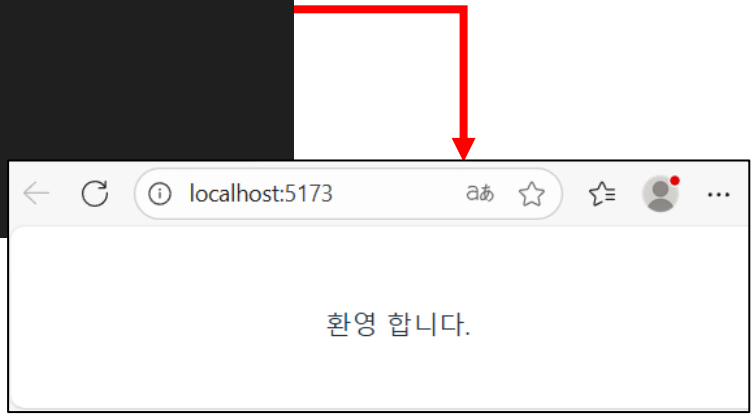
```jsx
function App() {
  const name = "홍길동"

  return (
    <>
      <div>
        <h1>이름 : {name}</h1>
      </div>
    </>
  )
}

export default App
```

- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 이미지 태그 작성하기
- JSX에서 img 태그를 사용하여 이미지를 표시하세요.
- - src: https://picsum.photos/150
- - alt: "샘플 이미지"

```jsx
function App() {


  return (
    <>
      <div>
        <img src="https://picsum.photos/150" alt="쌤플 랜덤 이미지" />
      </div>
    </>
  )
}

export default App
```

- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 조건부 렌더링 isLoggedIn 변수가 true이면 "환영합니다!", false이면 "로그인하세요"를 출력하세요.
- 단. 삼항연산자 이용해 작성하세요.

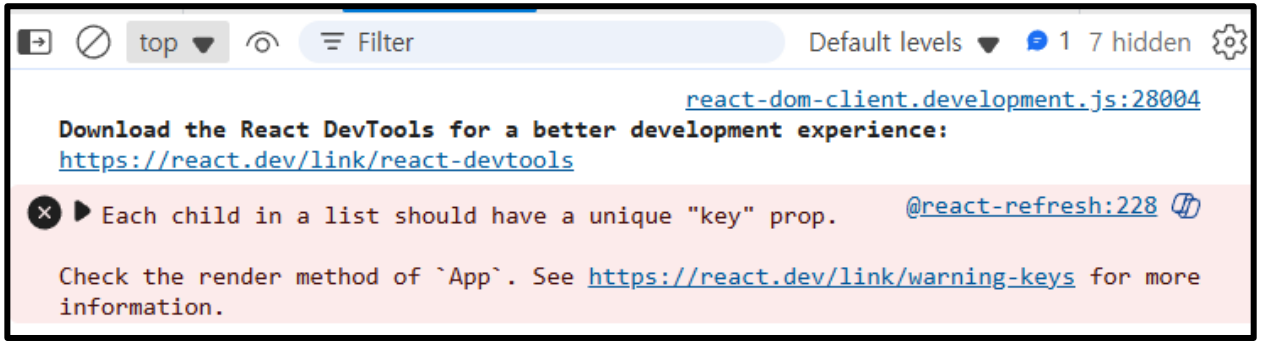
```jsx
function App() {

  const isLoggedIn = true;
  return (
    <>
      <div>
        <p>{isLoggedIn===true?"환영 합니다.":"로그인하세요"}</p>
      </div>
    </>
  )
}

export default App
```



- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 리스트 렌더링 : fruits 배열의 모든 항목을 <ul> 의 <li> 태그로 출력하세요.
- fruits = ["사과", "바나나", "오렌지"]
- map() 이용해 작성

```jsx
function App() {
  const fruits =["사과","바나나","오렌지"]
  return (
    <>
      <div>
        <ul>
          {fruits.map((item) => (
            <li>{item}</li>
          ))}
        </ul>
      </div>
    </>
  )
}

export default App
```

- 화면 출력이 잘 된 듯 보이나 console창에 아래와 같은 오류가 출력되어 있다.
- 이유는 가 3개가 생성이 되어 있는데 map()을 이용하여 새배열을 생성하여 출력하였기 때문에 각각의 의 index가 모두 같기때문에 생기는 오류이다.

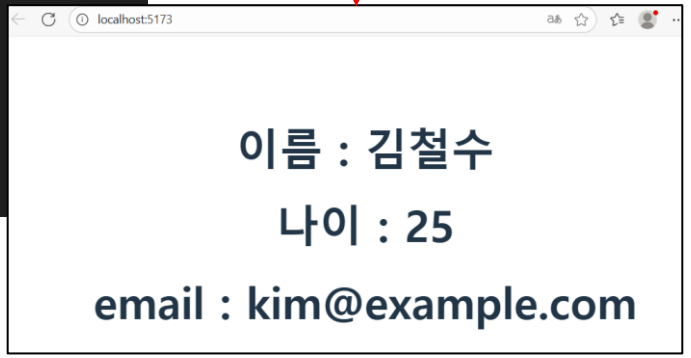



```
return (
  <>
    <div>
      <ul>
        {fruits.map((item, index) => (
          <li key={index}>{item}</li>
        ))}
      </ul>
    </div>
  </>
)
```

태그 안에 key={index}를 지정하여 <li> 태그가 여러 개여도 서로 다른 태그임을 명시해주어야 한다.

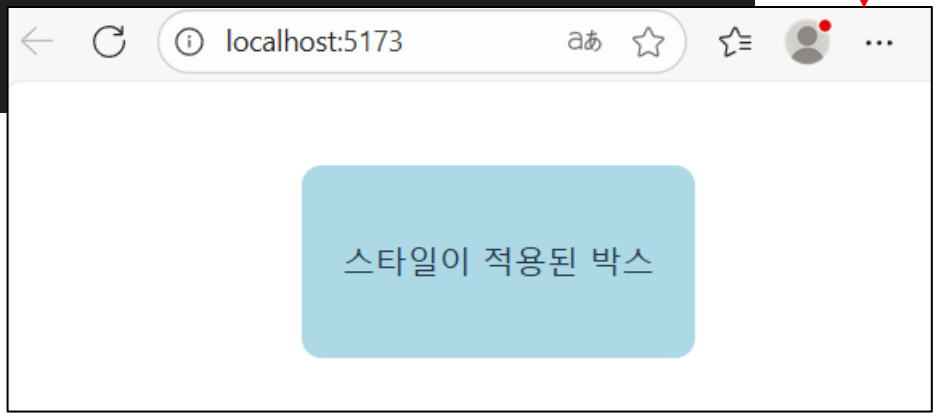
- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 객체 데이터 출력 : user 객체의 정보를 화면에 출력하세요
- user = { name: "김철수", age: 25, email: "kim@example.com" };

```jsx
function App() {

const user = { name: "김철수", age: 25, email: "kim@example.com" };

  return (
    <>
      <div>
        <h1>이름 : {user.name}</h1>
        <h1>나이 : {user.age}</h1>
        <h1>email : {user.email}</h1>
      </div>
    </>
  )
}

export default App
```



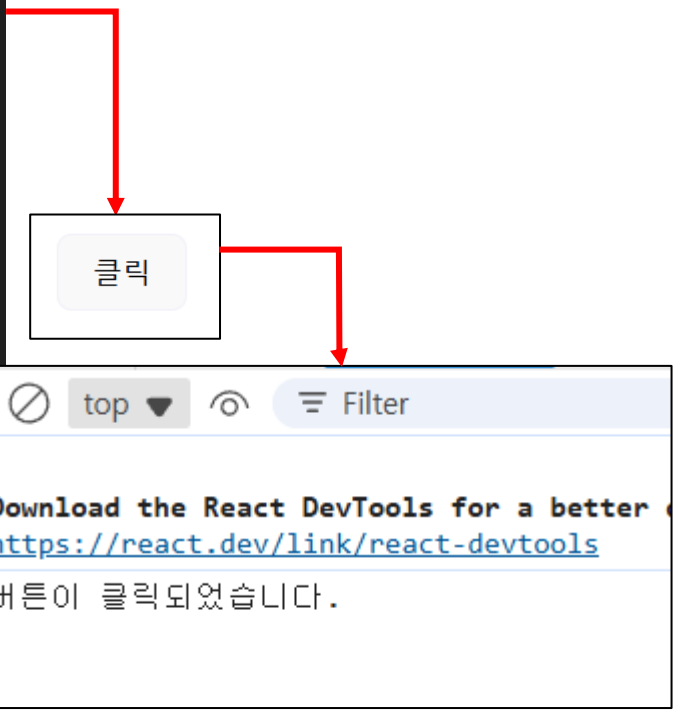
- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 인라인 스타일 적용 : div 요소에 인라인 스타일을 적용
- - 배경색: lightblue, - 패딩: 20px, - 테두리 반경: 10p

```jsx
function App() {

  return (
    <>
      <div style={{backgroundColor: "lightblue", padding: "20px", borderRadius: "10px"}}>
        <p>스타일이 적용된 박스</p>
      </div>
    </>
  )
}

export default App
```



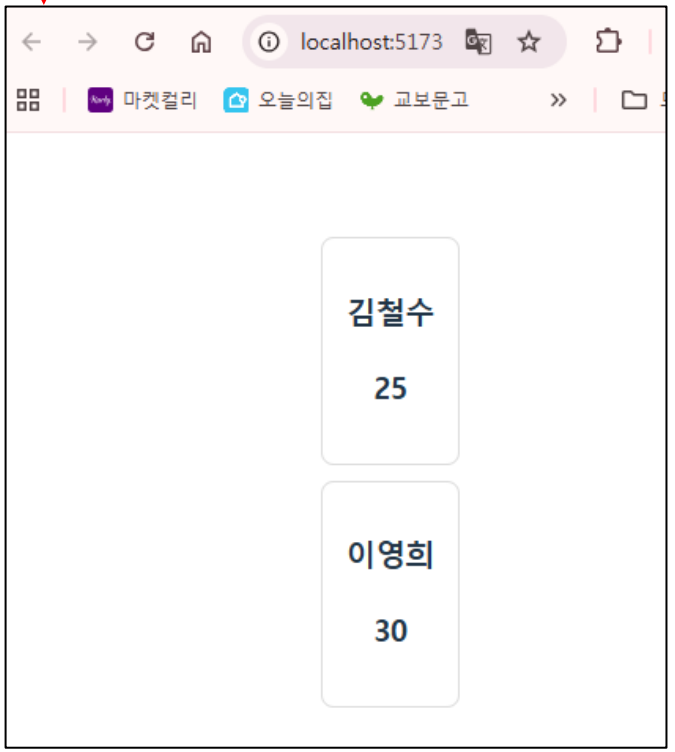
- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 버튼 클릭 이벤트 : 버튼을 클릭하면 콘솔에 "버튼이 클릭되었습니다!"를 출력 하시오.

```jsx
function App() {

  const handelClick = () =>{
    console.log("버튼이 클릭되었습니다.")
  }

  return (
    <>
      <div>
        <button type='button' onClick={handelClick}>클릭</button>
      </div>
    </>
  )
}

export default App
```



- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 컴포넌트 분리 : UserCard 컴포넌트를 만들고 App에서 사용하세요

```jsx
// 컴포넌트 이름은 반드시 대문자로 작성한다.
// React는 소문자로 시작하는 태그를 "HTML 태그"로 인식하기 때문
function UserCard(props) {
  return (
    <>
      <div
        style={{
          border: '1px solid #ddd',
          padding: '15px',
          margin: '10px',
          borderRadius: '8px',
        }}
      >
        <h3>{props.name}</h3>
        <h3>{props.age}</h3>
      </div>
    </>
  );
}
function App() {
  return (
    <>
      <div>
        {/* JSX에서도 대문자로 사용해야 한다. */}
        <UserCard name="김철수" age={25} />
        <UserCard name="이영희" age={30} />
      </div>
    </>
  );
}

export default App;
```



- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 배열 객체 렌더링 : products 배열의 모든 상품을 카드 형태로 출력하세요.
- const products = [

  { id: 1, name: "노트북", price: 1200000 },

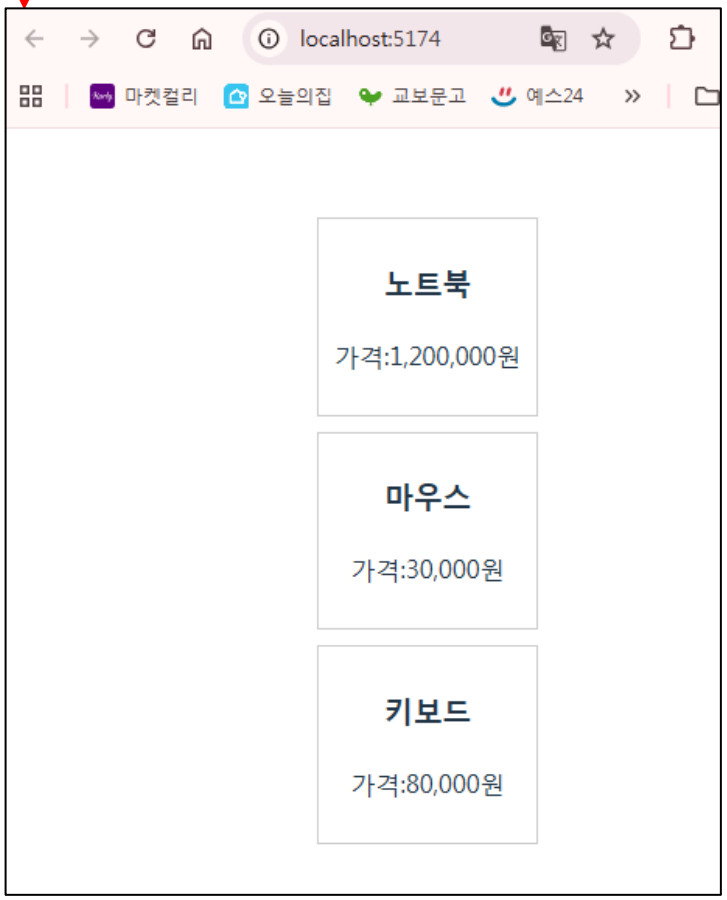
  { id: 2, name: "마우스", price: 30000 },

  { id: 3, name: "키보드", price: 80000 }

  ];
- map() 이용해 작성하세요

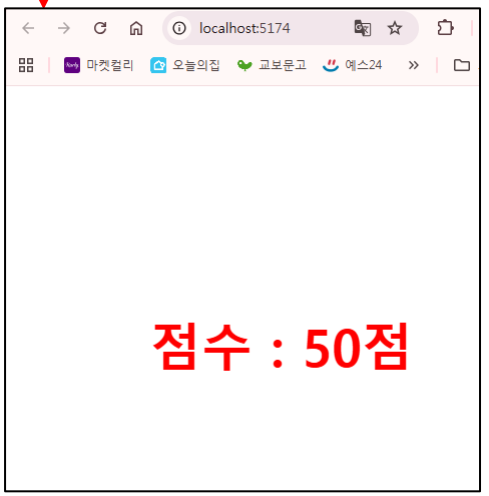
JSX(JavaScript XML) 간단한 예제11

```jsx
function App() {
  const products = [
    { id: 1, name: '노트북', price: 1200000 },
    { id: 2, name: '마우스', price: 30000 },
    { id: 3, name: '키보드', price: 80000 },
  ];
  return (
    <>
      <div>
        {products.map((product, index) => (
          <div
            key={product.id}
            style={{
              border: '1px solid #ccc',
              padding: '10px',
              margin: '10px',
            }}
          >
            <h3>{product.name}</h3>
            <p>가격:{product.price.toLocaleString()}원</p>
          </div>
        ))}
      </div>
    </>
  );
}

export default App;
```



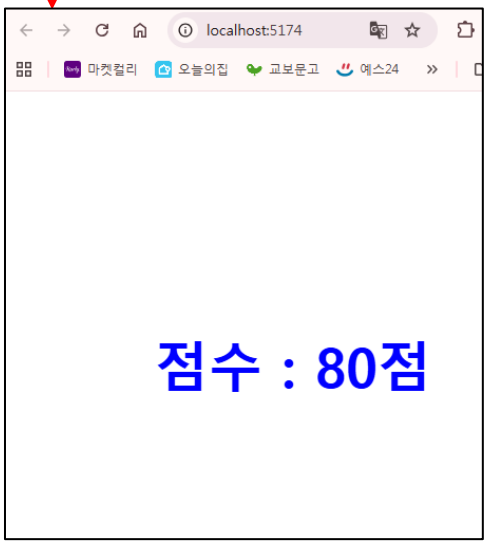
- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 조건부 스타일링 : score 값에 따라 다른 색상을 표시하세요. (삼항연산자 이용한 예)
  - 80점 이상: 파란색
  - 60점 이상: 초록색
  - 60점 미만: 빨간색

```jsx
function App() {
  const score = 50;

  return (
    <>
      <div>
        <h1
          style={{
            color: score >= 80 ? 'blue' : score >= 60 ? 'green' : 'red',
          }}
        >
          점수 : {score}점
        </h1>
      </div>
    </>
  );
}

export default App;
```



- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 조건부 스타일링 : score 값에 따라 다른 색상을 표시하세요. (함수를 이용한 예)
  - 80점 이상: 파란색
  - 60점 이상: 초록색
  - 60점 미만: 빨간색

```jsx
function App() {
  const score = 80;
  const getColor = (score) => {
    if (score >= 80) return 'blue';
    else if (score >= 60) return 'green';
    else return 'red';
  };

  return (
    <>
      <div>
        <h1 style={{ color: getColor(score) }}>점수 : {score}점</h1>
      </div>
    </>
  );
}

export default App;
```



- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- Fragment 사용하기 : 불필요한 div 없이 여러 요소를 반환하세요.
- 아래처럼 Fragment를 사용하지 않고 를 이용하면 HTML에 불필요한 태그가 생성된다.

```jsx
function Header() {
  return (
    // Fragment(프래그먼트)를 사용하여 여러 요소 반환
    // 아래처럼 작성하면 HTML에 <div>가 불필요하게 생성된다.
    <div>
      <h1>메인 제목</h1>
      <h2>부제목</h2>
      <p>설명 문구</p>
    </div>
  );
}

function App() {
  return (
    <>
      <div>
        <Header />
      </div>
    </>
  );
}

export default App;
```

- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- Fragment 사용하기 : 불필요한 div 없이 여러 요소를 반환하세요.
- 아래처럼 Fragment를 사용하면 HTML에 불필요한 태그가 생성되지 않는다.

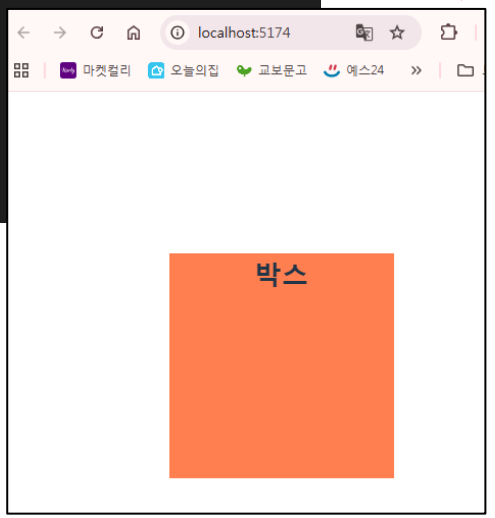
```jsx
function Header() {
  return (
    // Fragment(프래그먼트)를 사용하여 여러 요소 반환
    // Fragment는 <>와 </> 부분
    // 아래처럼 Fragment를 이용하면 불필요한 <div>태그가 생성되지 않는다.
    <>
      <h1>메인 제목</h1>
      <h2>부제목</h2>
      <p>설명 문구</p>
    </>
  );
}

function App() {
  return (
    <>
      <div>
        <Header />
      </div>
    </>
  );
}

export default App;
```

- **프로젝트 디렉토리에서 VS 실행 -> src 폴더 밑에 -> App.jsx 파일 열기**
- 동적 클래스명 적용 : isActive 상태에 따라 다른 className을 적용하세요
  - true일 때: "box active"
  - false일 때: "box“

<App.css 파일>

```
.box {
width: 200px;
height: 200px;
background-color: lightgray;
transition: all 0.3s;
}
.box.active {
background-color: coral;
transform: scale(1.1);
}
```

```jsx
function App() {
  const isActive = true;
  return (
    <>
      <div className={isActive ? 'box active' : 'box'}>
        <h2>박스</h2>
      </div>
    </>
  );
}

export default App;
```



React 기본 구조 흐름

React 컴포넌트(Component)

- 리액트는 화면에서 UI 요소를 구분할 때 '**컴포넌트**'라는 단위를 사용한다. 쉽게 말하면 리액트에서 앱을 이루는 가장 **작은 조각**이라고 할 수 있고, 레고 블록으로 집을 쌓게 된 경우 **하나의 블록이 '컴포넌트**'라고 할 수 있다
- 컴포넌트를 통해 **UI를 재사용 가능한 개별적인 여러 조각으로 나누고** 각 조각을 개별적으로 살펴볼 수 있다. 개념적으로 컴포넌트는 자바스크립트 **함수**와 유사하다. "props"라는 임의의 입력을 받은 후 화면에 어떻게 표시되는지를 기술하는 리액트 엘리먼트를 반환한다. 컴포넌트를 정의하는 가장 간단한 방법은 **자바스크립트 함수를 작성**하는 것이다
- **리액트는 컴포넌트와 함께 동작하고, 리액트 앱은 모두 컴포넌트로 구성된다.**

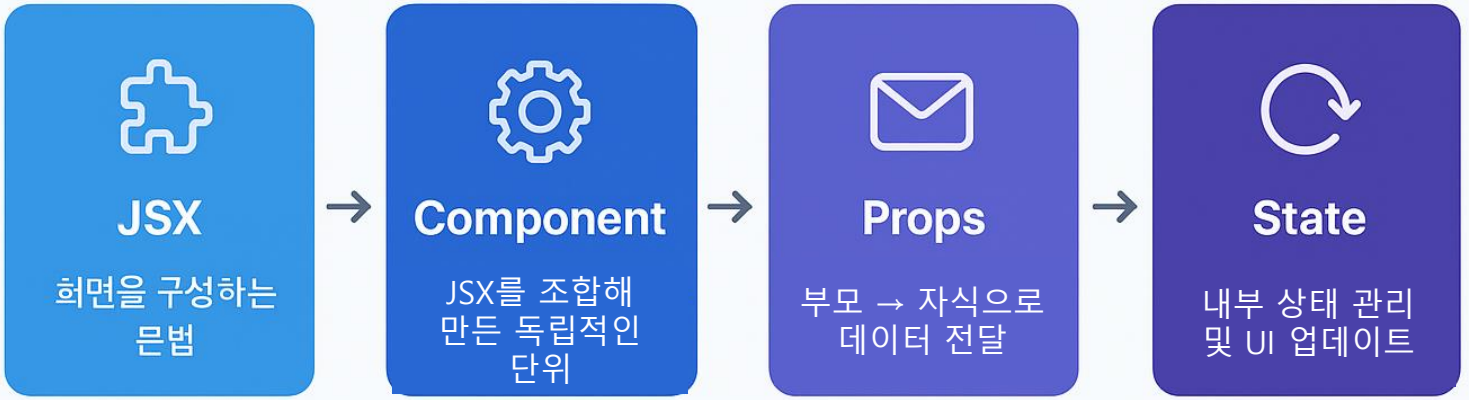
./src/App.jsx

```
import React from 'react';

function App() {
  return <div>Hellow!!!</div>;
}

export default App;
```

App 컴포넌트 정의

App 컴포넌트는 html을 반환하고 있다.

React 기본 구조 흐름 (JSX → Component → Props → State)



JSX로 화면을 만들고 → Component로 나누고,
Props로 데이터 전달 → State로 동적으로 변경한다

React 렌더링 흐름도

5단계 흐름:

1. **Ex01.jsx - 컴포넌트 작성**
   -함수 작성 및 export

2. **App.jsx - Ex01 import 및 사용**
   -Ex01을 불러와서 사용

3. **main.jsx - App을 root에 렌더링**
   -ReactDOM으로 root에 연결

4. **index.html - root div 제공**
   -div id="root" 제공 및 main.jsx 실행

5. **브라우저 - 최종 화면 표시!**
   -실제 브라우저 창 형태로 결과 표시

기초 연습 문제

문제1] 간단한 인사말 컴포넌트

조건

① "안녕하세요, React입니다!"라는 텍스트를 화면에 표시하는 컴포넌트를 만드세요.

② src 폴더안에 -> components 폴더 생성 -> Ex01.jsx 파일을 생성하여 작성하시오.

힌트

✓function 키워드로 컴포넌트 만들기

✓return 안에 JSX 작성

✓div 태그 사용

<출력 결과>

문제2] 자기소개 카드

조건

① 본인의 이름, 나이, 취미를 표시하는 프로필 카드 컴포넌트를 만드세요.

② css는 src 폴더 안의 App.css 안에 기존에 작성한 css 아래 부분에 작성하시오.

③ src 폴더안에 -> components 폴더 생성 -> Ex02.jsx 파일을 생성하여 작성하시오.

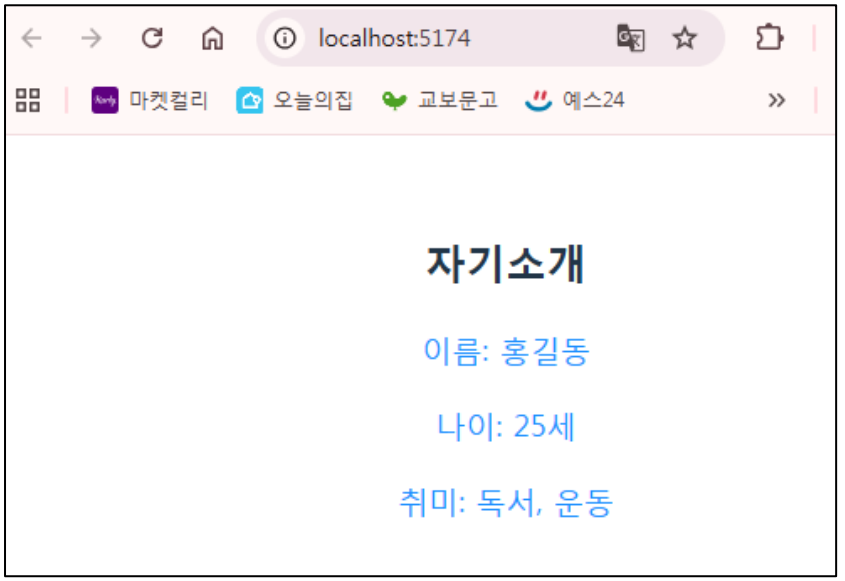
힌트

✓div로 카드 영역 만들기

✓p 태그로 각 정보 표시

✓className으로 스타일 적용

<출력 결과>



문제3] 헤더와 푸터

조건

① 페이지 상단에 헤더(제목), 하단에 푸터(저작권 정보)를 표시하는 레이아웃을 만드세요.

② css는 src 폴더 안의 App.css 안에 기존에 작성한 css 아래 부분에 작성하시오.

③ src 폴더안에 -> components 폴더 생성 -> Ex03.jsx 파일을 생성하여 작성하시오.

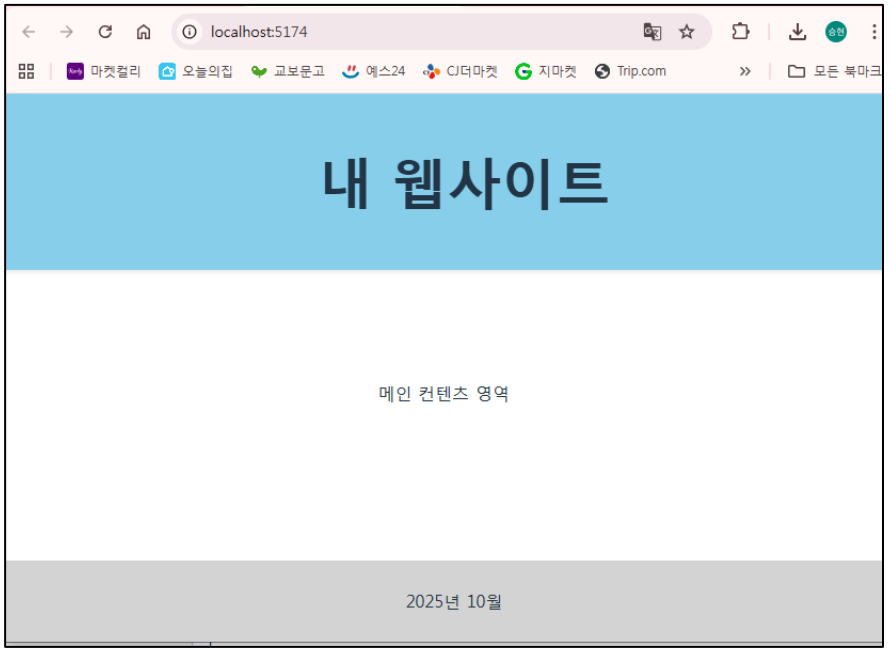
힌트

✓header와 footer 태그 사용

✓부모 div로 감싸기

✓className으로 스타일링

<출력 결과>



문제3] CSS는 아래 코드 적용 하시오.

```css
.header {
  position: fixed; /* 상단 고정 */
  top: 0;
  left: 0;
  width: 100%;
  padding: 20px; /* 여백 추가 */
  background-color: skyblue;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1); /* 그림자 */
}
main {
  margin: 200px auto;
}
.footer {
  position: fixed; /* 하단 고정 */
  left: 0;
  bottom: 0;
  width: 100%;
  padding: 10px;
  background-color: lightgray;
  text-align: center;
}
```

문제4] 과일 목록

조건

① 사과, 바나나, 오렌지, 포도 4가지 과일을 ul/li 태그를 사용하여 목록으로 표시하세요.

② src 폴더안에 -> components 폴더 생성 -> Ex04.jsx 파일을 생성하여 작성하시오.

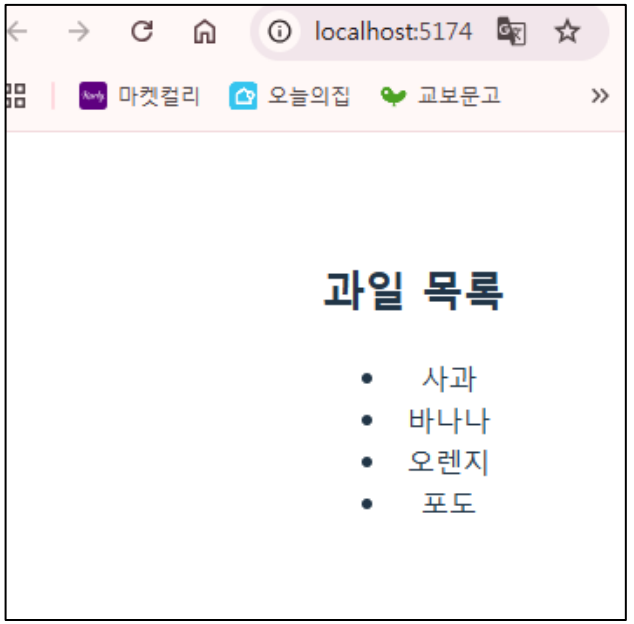
힌트

✓ul 태그 사용

✓각 과일은 li 태그로

✓JSX에서는 태그를 닫아야 함

<출력 결과>



문제5] 간단한 테이블

조건

① 이름, 나이, 직업 정보를 가진 3명의 사람 데이터를 테이블로 표시하세요.

② css는 src 폴더 안의 App.css 안에 기존에 작성한 css 아래 부분에 작성하시오.

③ src 폴더안에 -> components 폴더 생성 -> Ex05.jsx 파일을 생성하여 작성하시오.

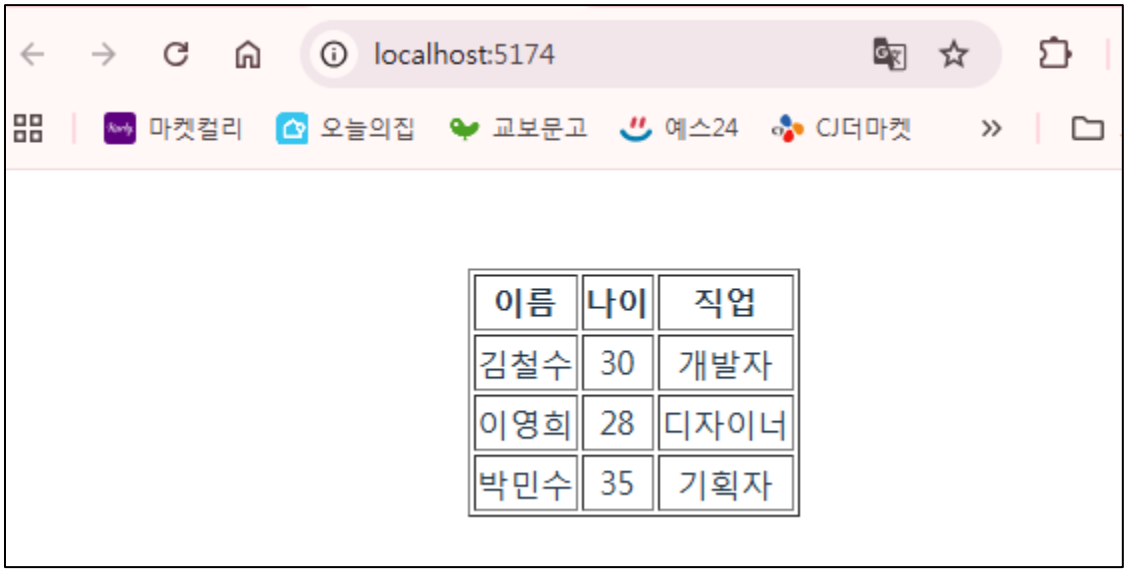
힌트

✓table, thead, tbody 태그 사용

✓tr로 행, th/td로 셀 만들기

✓3명의 데이터 입력

<출력 결과>



이름	나이	직업
김철수	30	개발자
이영희	28	디자이너
박민수	35	기획자

문제6] 이미지 갤러리

조건

① 3개의 이미지를 가로로 나란히 배치하고, 각 이미지 아래에 설명을 추가하세요.

② css는 src 폴더 안의 App.css 안에 기존에 작성한 css 아래 부분에 작성하시오.

③ public 폴더안에 -> images 폴더 생성 -> 제공한 이미지(image01.png, image02.png, image03.png) 파일을 복사 붙여넣기 한다.

④ src 폴더안에 -> components 폴더 생성 -> Ex06.jsx 파일을 생성하여 작성하시오.

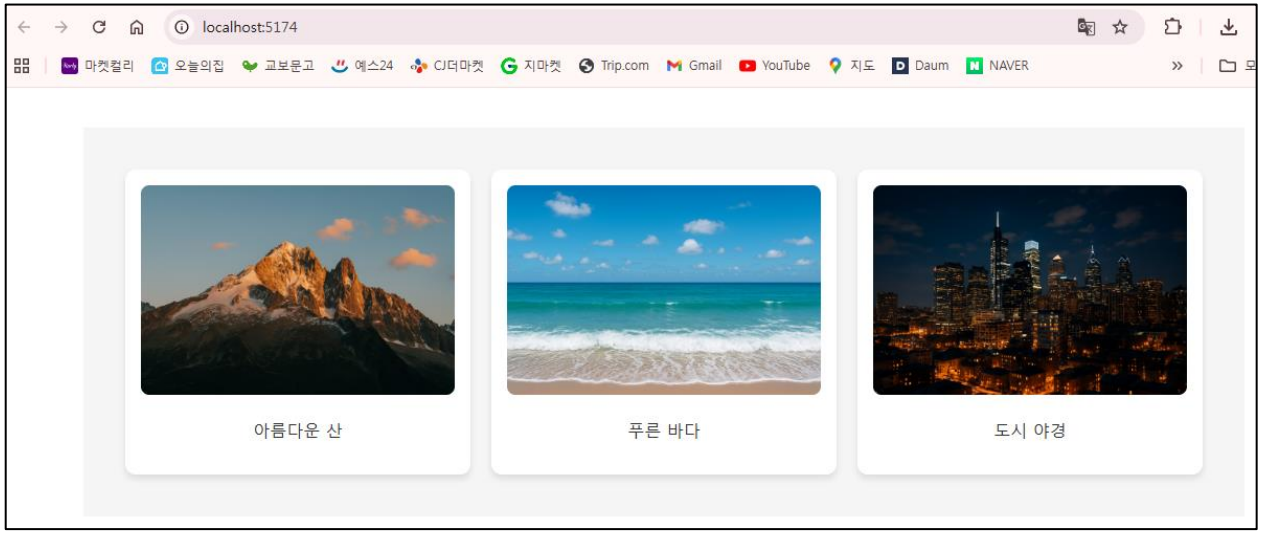
힌트

✓div로 각 이미지 카드 만들기

✓img 태그의 src 속성

✓alt 속성 필수

<출력 결과>



문제7] 색상 박스

조건

① 빨강, 파랑, 초록 배경색을 가진 3개의 div 박스를 만들고, 각각 다른 크기로 표시하세요.

② css는 인라인 스타일로 작성하시오.

③ src 폴더안에 -> components 폴더 생성 -> Ex07.jsx 파일을 생성하여 작성하시오.

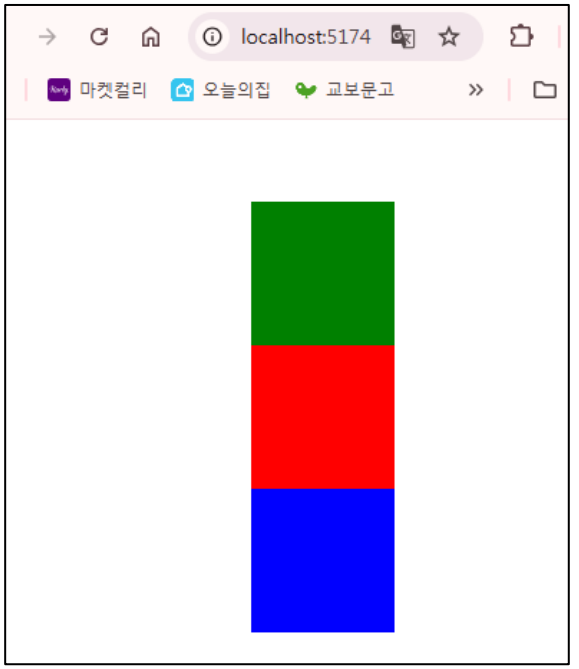
힌트

✓div 태그 3개 사용

✓style 속성으로 인라인 스타일

✓backgroundColor, width, height 설정

<출력 결과>



문제8] 메뉴 리스트

조건

① 레스토랑 메뉴 3개(메뉴명, 가격, 설명)를 카드 형태로 표시하세요.

② css는 src 폴더 안의 App.css 안에 기존에 작성한 css 아래 부분에 작성하시오.

③ src 폴더안에 -> components 폴더 생성 -> Ex08.jsx 파일을 생성하여 작성하시오.

힌트

✓각 메뉴를 div로 카드 만들기

✓h3로 메뉴명, p로 설명과 가격

✓className으로 스타일링

< 출력 결과 >


localho...
오늘의집
교보문고
김치찌개
얼큰한 김치찌개
8,000원
된장찌개
구수한 된장찌개
7,000원
비빔밥
건강한 비빔밥
9,000원

문제9] 학생 정보 카드

조건

① 학생 이름, 학번, 학과를 표시하는 카드를 만드세요. 정보는 각각 다른 색상으로 강조하세요.

② css는 인라인 스타일로 작성하시오.

③ src 폴더안에 -> components 폴더 생성 -> Ex09.jsx 파일을 생성하여 작성하시오.

힌트

✓div로 카드 영역

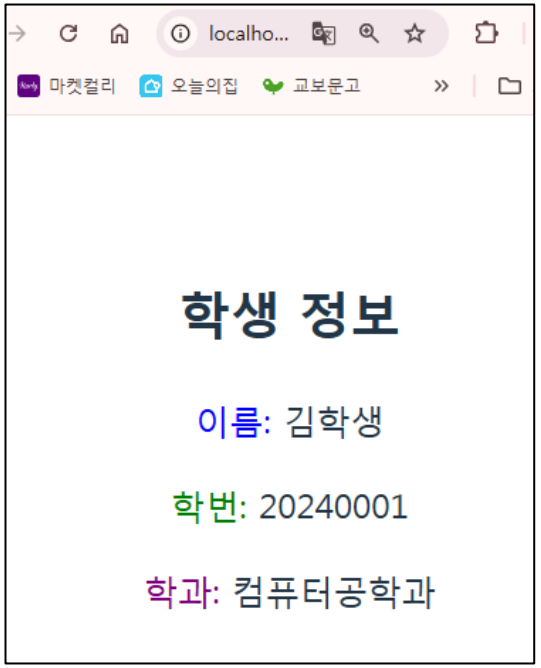
✓p태그로 각 정보 표시

✓span으로 각 정보 감싸기

(각 정보는 이름, 학번, 학과 글자를 의미)

✓style 속성으로 색상 지정

<출력 결과>



문제10] 간단한 내비게이션 바

조건

① 홈, 소개, 연락처 3개의 메뉴를 가로로 배치한 내비게이션 바를 만드세요.

② css는 src 폴더 안의 App.css 안에 기존에 작성한 css 아래 부분에 작성하시오.

③ src 폴더안에 -> components 폴더 생성 -> Ex10.jsx 파일을 생성하여 작성하시오.

힌트

✓nav 태그 사용

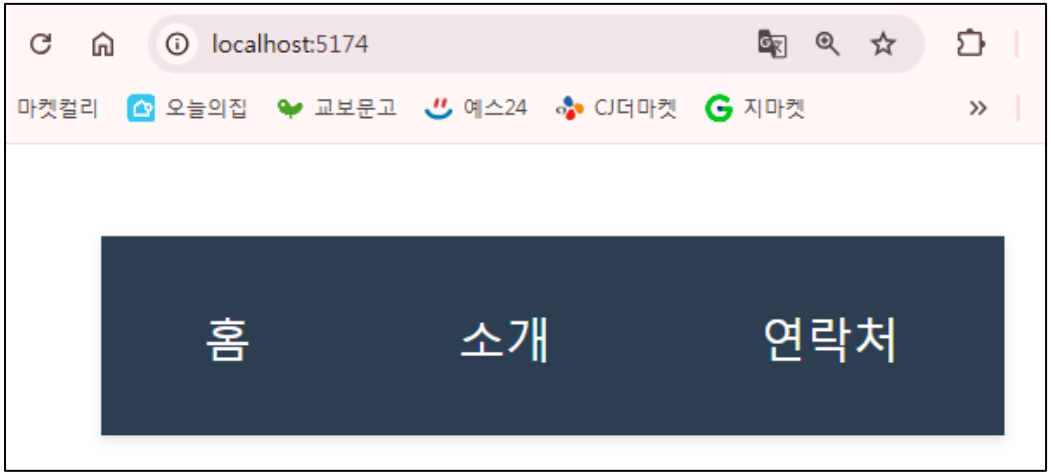
✓ul과 li로 메뉴 구성

✓a 태그로 링크 (href="#")

✓.nav-list { display: flex; }

✓.nav-list li { margin: 0 10px; }

<출력 결과>



문제11] 제품 카드

조건

① 제품 이름, 가격, 이미지를 포함한 제품 카드 컴포넌트를 만드세요.

② 단, img src="https://images.unsplash.com/photo-1590658268037-6bf12165a8df?w=400"이용해 작성하시오.

③ src 폴더안에 -> components 폴더 생성 -> Ex11.jsx 파일을 생성하여 작성하시오.

힌트

✓div로 카드 구조

✓img, h3, p 태그 활용

✓가격은 강조 표시(strong 태그 이용)

<출력 결과>

무선 이어폰

고음질 블루투스 이어폰

89,000원

문제12] 주소 정보

조건

① 회사명, 주소, 전화번호, 이메일을 표시하는 연락처 정보 섹션을 만드세요.

② src 폴더안에 -> components 폴더 생성 -> Ex12.jsx 파일을 생성하여 작성하시오.

힌트

✓section 또는 div 사용

✓각 정보를 p 태그로

✓strong으로 레이블 강조

<출력 결과>

연락처

회사명:테크 컴퍼니

주소:서울시 강남구 테헤란로 123

전화:02-1234-5678

이메일:info@techcompany.com

문제13] 아이콘과 텍스트

조건

① 간단한 아이콘(이모지 사용)과 함께 설명 텍스트를 표시하는 3개의 기능 소개 박스를 만드세요.

② src 폴더안에 -> components 폴더 생성 -> Ex13.jsx 파일을 생성하여 작성하시오.

<windows에서 이모지 입력하는 방법>

단축키: Windows키 + . (마침표) 또는 Windows키 + ; (세미콜론)

이모지 창이 열리면 검색하거나 선택

<Mac에서 이모지 입력하는 방법>

단축키: Command + Control + Space

이모지 선택기가 나타남

③ https://emojipedia.org/ => 영어로 직접 원하는 이모지 검색 가능 예) laugh(웃음) : 😄

문제13] 아이콘과 텍스트

힌트

✓div로 각 박스 만들기

✓이모지는 텍스트로 입력

✓span으로 아이콘 크기 조절

<출력 결과>