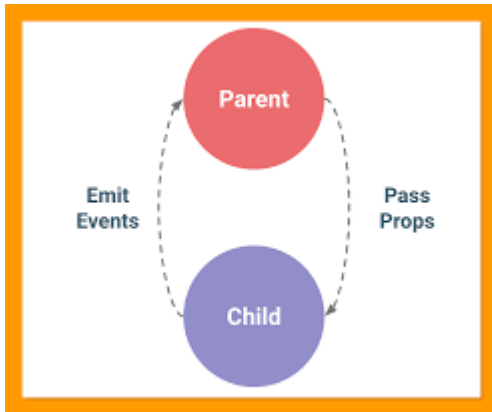


# props(속성)



부모 컴포넌트에서 자식 컴포넌트로 전달하기 위한 데이터  
**properties**의 줄임말

외부에서 전달되는 매개변수, 부모 컴포넌트에서 자식 컴포넌트로 데이터를 전달할 때 사용됩니다.

이것이 React 컴포넌트 간에 정보를 주고받는 주요 메커니즘 중 하나입니다.

부모 컴포넌트는 자식 컴포넌트에게 **props**를 통해 데이터를 전달하고, 자식 컴포넌트는 이를 활용하여 화면에 정보를 표시하거나 특정 동작을 수행할 수 있습니다.

**React에서는 props를 통한 단방향 데이터 전달이 주요한 패턴입니다.**



단, 상태 관리 라이브러리인 Redux, Context API 등을 사용하면 컴포넌트 간의 데이터 흐름을 조금 더 유연하게 관리할 수 있습니다.



자식 컴포넌트에서 부모 컴포넌트로 데이터를 넘기는 법

- 부모 컴포넌트에서 props로 함수를 내려주고, 자식 컴포넌트에서 그 함수를 호출하면서 부모 컴포넌트로 데이터를 전달할 수 있습니다.

---

## 예

```
<Message value="안녕하세요" color="blue" />
```

위 코드는 Message 함수형 컴포넌트를 호출한다. 즉 Message 함수를 호출한다.

이때 이 함수의 파라미터로 다음과 같은 파라미터 객체가 전달된다.

```
{ value:"안녕하세요", color:"blue" }
```

이렇게 컴포넌트에 전달되는 파라미터 객체를 props 이라고 한다.

## ex02 프로젝트 생성

프로젝트를 만들 적당한 디렉토리에서 다음 명령을 실행하자.

```
npx create-react-app ex02
```

Visual Studio Code에서 ex02 프로젝트 디렉토리를 열고

다음 명령을 실행하여 프로젝트를 개발 모드로 실행하자.

```
npm start
```

---

## Message 컴포넌트 구현

src/App.jsx

```
import React from "react";
import Message from "../Message";

function App() {
  return (
    <div>
      <Message value="안녕하세요" color="blue" />
    </div>
  );
}
```

```

    </div>
  );
}
export default App;

```

## src/Message.jsx

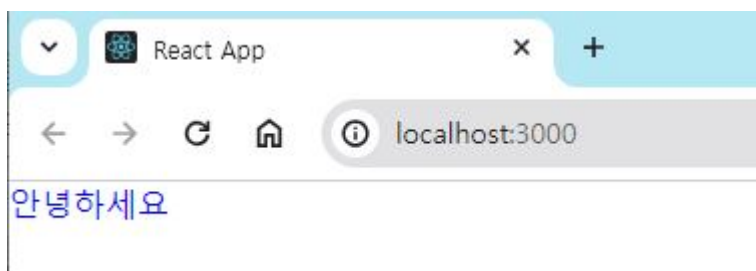
```

import React from 'react'

// App컴포넌트에서 전달받은 속성을 props파라미터 객체를 통해 조회
function Message(props) {
  const msgStyle = {color: props.color};
  return (
    <div style={msgStyle}>
      {props.value}
    </div>
  )
}

export default Message;

```



## number type props

### src/Message.jsx

```
import React from 'react'

function Message(props) {
  const msgStyle = {color: props.color, fontSize: props.size};
  return (
    <div style={msgStyle}>
      {props.value}
    </div>
  )
}

export default Message;
```

## src/App.jsx

```
import React from "react";
import Message from "../Message";

function App() {
  return (
    <div>
      <Message value="안녕하세요" color="blue" size={20} />
      <Message value="안녕하세요" color="blue" size={30} />
      <Message value="안녕하세요" color="blue" size={40} />
    </div>
  );
}

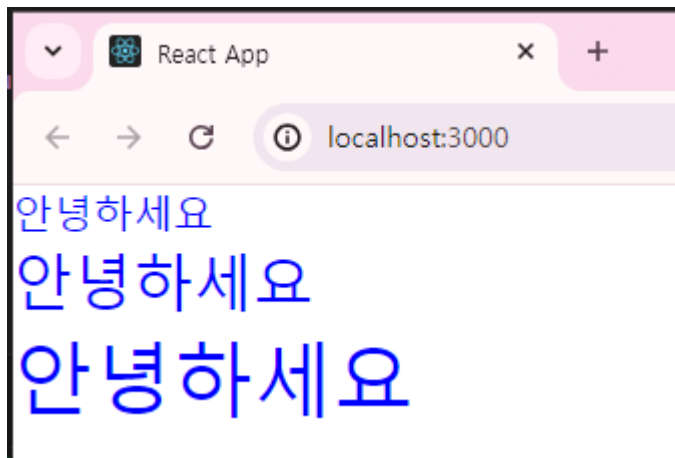
export default App;
```

### size="20"

여기서 "20" 값은 string 타입이다. 따라서 font-size 속성이 적용되지 않는다.

### size={20}

string 타입이 아니라면 위와 같이 { } 중괄호로 묶어야 한다.



## PersonTable 컴포넌트 구현

### src/personTable.css 생성

```
table.PersonTable { border-collapse: collapse; margin: 5px;
}
table.PersonTable td { border: 1px solid gray; padding: 4px; }
```

### src/PersonTable.jsx 생성

```
import React from 'react';
import './personTable.css';

function PersonTable(props) {
  let trlist = props.persons.map(person =>
    <tr><td>{ person.name }</td><td>{ person.age }</td></tr>);
  return (
    <table className="PersonTable">
      <tbody>
        <tr><td>이름</td><td>나이</td></tr>
        { trlist }
      </tbody>
    </table>
  );
}
```

```

        </tbody>
      </table>
    );
  }

  export default PersonTable;

```

## src/App.jsx

```

import React from 'react';

import PersonTable from './PersonTable';

function App() {
  let persons1 = [
    { name: '홍길동', age: 16 },
    { name: '임꺽정', age: 19 },
    { name: '전우치', age: 20 }
  ];
  let persons2 = persons1.slice(0);
  persons2.reverse();
  return (
    <div>
      <PersonTable persons={ persons1 } />
      <hr />
      <PersonTable persons={ persons2 } />
    </div>
  );
}

export default App;

```

## import PersonTable from './PersonTable';

PersonTable.jsx 파일에서 export default된 PeronsTable 함수를 import 한다.

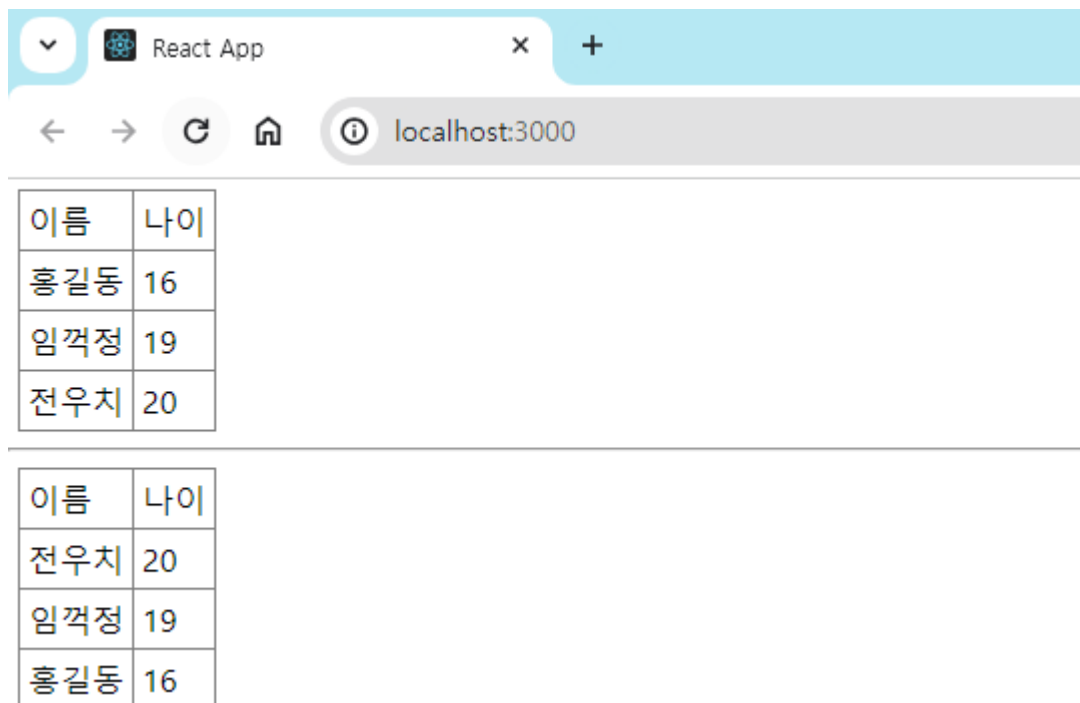
```
let persons2 = persons1.slice(0);
```

```
persons2.reverse();
```

person1 배열을 복제하여 person2 배열을 만들고  
person2 배열의 순서를 뒤집는다.

```
<PersonTable persons={ persons1 } />
```

PersonTable 컴포넌트를 호출하면서 persons1 배열을 props로 전달한다.  
persons1 부분은 자바스크립트 표현식이므로 { } 괄호로 묶어야 한다.



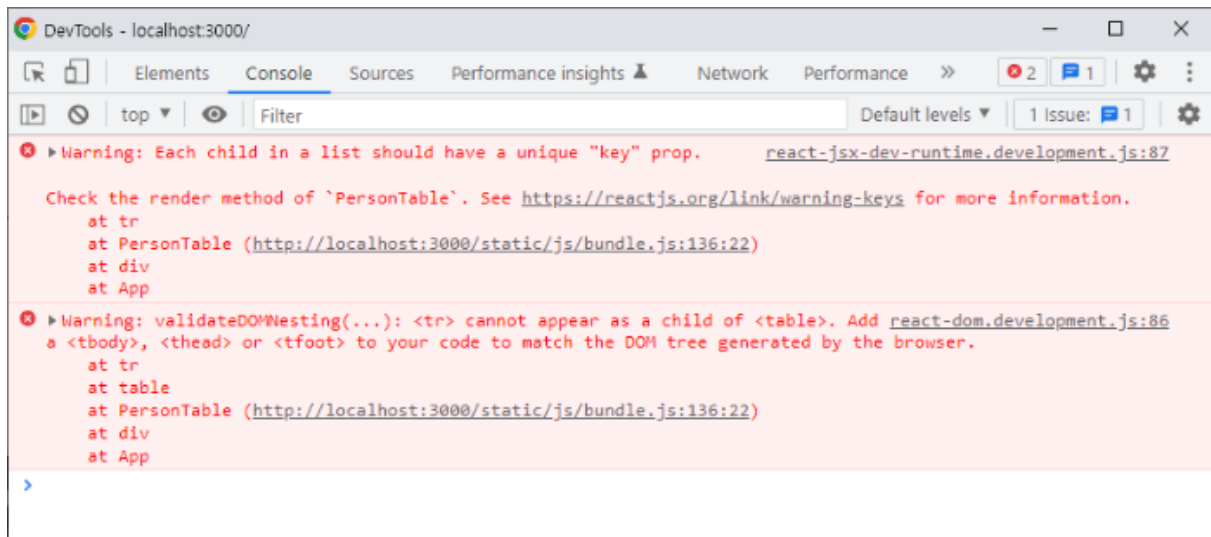
이름	나이
홍길동	16
임꺽정	19
전우치	20

---

이름	나이
전우치	20
임꺽정	19
홍길동	16

## key prop

F12 키를 눌러서 웹브라우저 개발자창을 열고, Console 탭을 보자.



Warning: Each child in a list should have a unique "key" prop

태그 목록의 각각의 항목에는 "key" prop가 있어야 한다는 경고

Warning: validateDOMNesting(...): <tr> cannot appear as a child of <table>.  
Add a <tbody>, <thead> or <tfoot> to your code to match the DOM tree generated by the browser.

<table> 태그 바로 아래에 <tr> 태그가 오면 안되고, <thead> <tbody> 태그가 필요하다는 경고

## "key" prop가 있어야 하는 이유

<tr> 태그나 어떤 태그가 반복된 목록의 경우, 그 목록 태그에 "key" prop가 있어야 한다. 그리고 "key" prop의 값은 DB 테이블의 기본키(primary key) 같은 유일한 값이어야 한다.

리액트는 데이터가 수정 삭제될 때 마다,

전체 코드를 다시 실행해서 화면 Virtual DOM 객체들을 전부 다시 생성하고, 직전에 생성했던 Virtual DOM 객체들과 비교하여 달라진 부분을 찾는다.

이렇게 달라진 부분을 찾을 때, 목록을 구성하는 태그에 "key" prop이 있으면 좀 더 정확하고 빠르게 달라진 부분을 찾을 수 있다.

## 주의



배열의 index를 key 값으로 사용하는 예제 코드를 가끔 볼 수 있는데, 잘못된 구현이다. 기본키(primary key) 값 같은 것을 사용해야 한다.

## src/App.jsx 수정

```
import React from 'react';

import type { Person } from './PersonTable';
import PersonTable from './PersonTable';

function App() {
  let persons1 = [
    { id: 31, name: '홍길동', age: 16 },
    { id: 32, name: '임꺽정', age: 19 },
    { id: 33, name: '전우치', age: 20 }
  ];
  let persons2 = persons1.slice(0);
  persons2.reverse();
  return (
    <div>
      <PersonTable persons={ persons1 } />
      <hr />
      <PersonTable persons={ persons2 } />
    </div>
  );
}

export default App;
```

## src/PersonTable.jsx 수정

```
import React from 'react';
import './PersonTable.css';

function PersonTable(props) {
  let trlist = props.persons.map(person =>
    <tr key={ person.id }><td>{ person.name }</td><td>{ per
```

```

son.age }</td></tr>);
    return (
      <table className="PersonTable">
        <tbody>
          <tr key={0}><td>이름</td><td>나이</td></tr>
          { trlist }
        </tbody>
      </table>
    );
  }

  export default PersonTable;

```

## 태그 사이의 내용을 보여 주는 children

### App.jsx

```

import MyComponent from './MyComponent';

const App = () => {
  return <MyComponent>리액트</MyComponent>
}

export default App;

```

함수의 파라미터가 객체라면 그 값을 바로 비구조화 해서 사용할 수 있다.

### MyComponent.jsx

```

//아래와 같이 파라미터로 default props를 설정
const MyComponent = ({name = '기본 이름', children}) => {
  return (
    <div>
      안녕하세요, 제 이름은 {name} 입니다. <br />

```

```
        children 값은 {children} 입니다.  
    </div>  
  )  
}  
  
export default MyComponent;
```