React

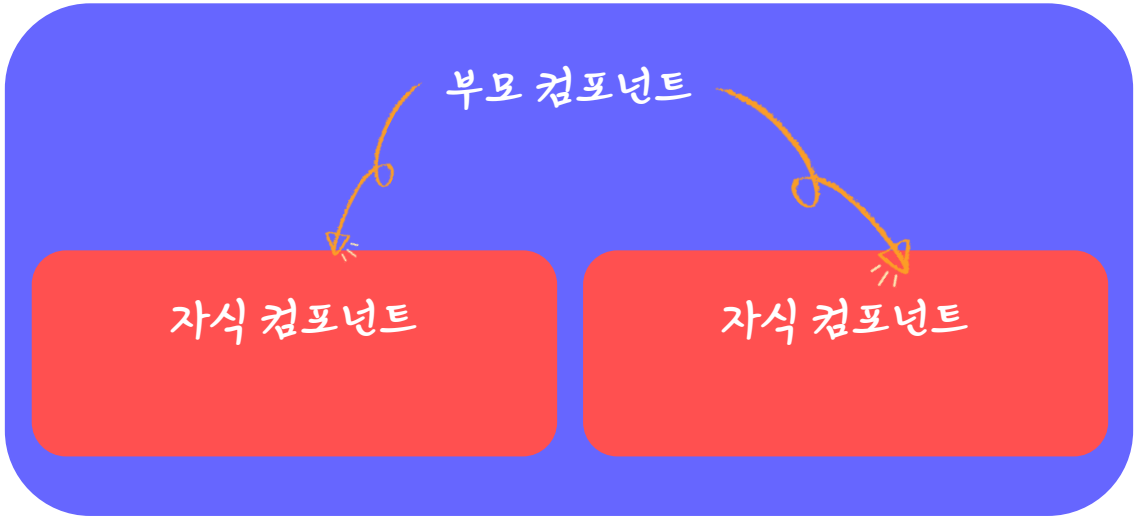
02. State와 props

- state와 useState -

Props

props

- 부모가 되는 컴포넌트에서 자식 컴포넌트에게 보내는 데이터
  - 상위 → 하위로만 데이터 전송, 그 반대로는 보낼 수 없음
- 여러 번 쓰이는 컴포넌트에서 내부 데이터만 다르게 사용하고 싶을 때 사용
  - 다양한 데이터를 props 라는 객체로 받아서 사용



props란?

- properties 를 줄인 표현으로 컴포넌트 속성을 설정할 때 사용하는 요소
- props는 컴포넌트끼리 값을 전달하는 수단
- 상위 컴포넌트에서 하위 컴포넌트로 전달 (단방향 데이터 흐름)

```
{/* 일반 사용법 */}
<ClassComponent></ClassComponent>

{/* props 사용법 */}
<ClassComponent title="제목" content="내용"></ClassComponent>
```

props 사용 방법 (함수형 컴포넌트)

- 부모 컴포넌트에서 전달한 props는 함수형 컴포넌트에서 함수의 파라미터로 전달받으며, JSX 내부에서 { } 기호로 감싸서 사용한다.

```
<FuncComponent name="코딩온"></FuncComponent>
```

* 부모 컴포넌트에서 name props 전달

```
const FuncComponent = (props) => {
    return (
        <>
        <div>안녕? {props.name}</div>
        <div>반가워!</div>
        </>
    );
}
```

* 자식 컴포넌트에서 name props 받음

props 사용 방법 (함수형 컴포넌트)

```
function MainHeader({ text }) {
    return (
        <h1>{text}</h1>
    )
}
export default MainHeader;
```

MainHeader.js

하위 컴포넌트에서 props

- 함수의 인자에 전달받을 props 이름 작성
- 상위 컴포넌트에서 text라는 값을 전달받아서 h1 내부에 넣어주겠다!

```
function App() {
  return (
    <div>
      <MainHeader text="Hello, props world!" />
    </div>
  );
}

export default App;
```

App.js

상위 컴포넌트에서의 props 사용

- 하위 컴포넌트 생성시 만들어준 props의 이름을 태그의 속성처럼 작성!
- Hello, ~ 라는 문자열이 하위 컴포넌트로 전달

defaultProps

- 부모 컴포넌트에서 props가 전달되지 않았을 때 기본값으로 보여줄 props를 설정하는 것

```
<FuncComponent ></FuncComponent>
```

* 부모 컴포넌트

```
const FuncComponent = (props) => {
    return (
        <>
        <div>안녕? {props.name}</div>
        <div>반가워!</div>
        </>
    );
}

FuncComponent.defaultProps = {
    name: '홍길동'
}
```

* 자식 컴포넌트

props.children

- 부모 컴포넌트에서 자식 컴포넌트를 호출할 때 태그 사이에 작성한 문자열

```
<FuncComponent name="코딩온">자식 내용</FuncComponent>
```

* 부모 컴포넌트

```
const FuncComponent = (props) => {
    return (
        <>
        <div>안녕? {props.name}</div>
        <div>반가워!</div>
        <h4>{props.children}</h4>
        </>
    );
}
```

* 자식 컴포넌트

클래스형 컴포넌트 props

```
class ClassComponent extends Component {
    render() {
        return(
            <h1>Class Component 입니다. 이름은 { this.props.name }</h1>
        );
    }
```

props 데이터 사용

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

class ClassProps extends Component {
  render() {
    return (
      <div>
        <h1>이름: {this.props.name}</h1>
        <h2>나이: {this.props.age}</h2>
      </div>
    )
  }
}
export default ClassProps;
```

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

class ClassProps extends Component {
  render() {
    const { name, age } = this.props;
      return (
        <div>
          <h1>이름: {name}</h1>
          <h2>나이: {age}</h2>
        </div>
      )
  }
}
export default ClassProps;
```

State

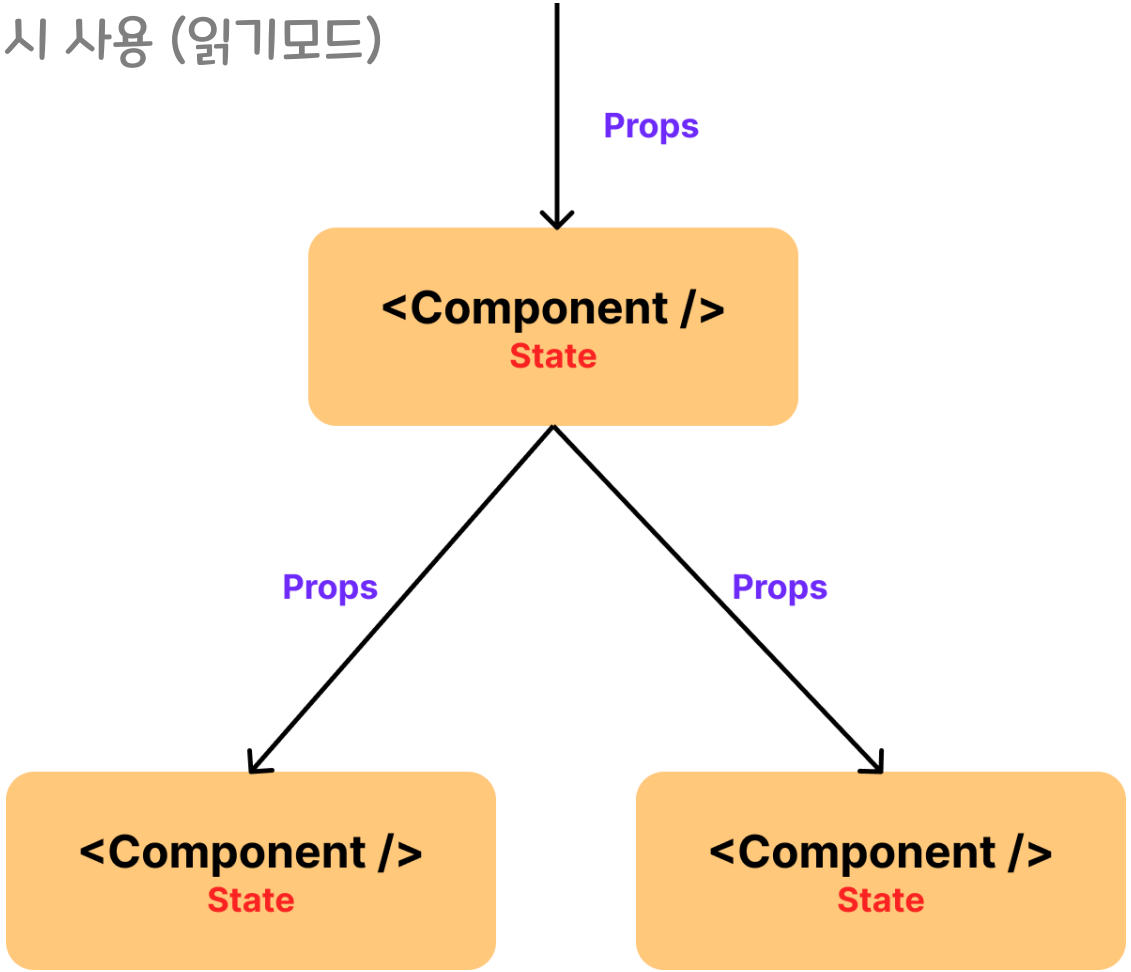
state란?

- React에서 앱의 유동적인 데이터를 다루기 위한 개체
- 계속해서 변하는 특정 상태
- 상태에 따라 다른 동작을 함
- 왜 사용할까?
  - State가 변경될시 자동으로 재랜더링 되기 때문
  - 이 점이 변수와 다른 점

- 말이 조금 어렵지만, '상태를 바꿀 수 있는 변수'

state vs. props

- **props**: 부모 컴포넌트에서 자식 컴포넌트에 데이터 전달 시 사용 (읽기모드)
  - React는 **단방향 데이터 흐름**! 기억하시죠? 😉
- **state**: 특정 컴포넌트가 갖는 상태 (값)
  - 컴포넌트 **내부에서** 선언되고 **내부에서** 값을 변경함!



클래스형 컴포넌트의 state

- 기존 형태

```
import React, { Component } from "react";

class ClassState extends Component {
    constructor(props) {
        super(props);

        this.state = {
            number: 0,
        };
    }
    render() {
        const { number } = this.state;
        return (
            <div> <h3>Number : {number}</h3> </div>
        );
    }
}
export default ClassState;
```

클래스형 컴포넌트의 state

- 현재 버전

```
import React, { Component } from "react";

class ClassState extends Component {
    state = {
        number: 0,
    };
    render() {
        const { number } = this.state;
        return (
            <div> <h3>Number : {number}</h3> </div>
        );
    }
}
export default ClassState;
```

함수형 컴포넌트의 state

- 함수형 컴포넌트는 state 기능이 원래 없었다.
- React 16.8 버전 이후부터 useState라는 함수가 생겼고,
  이를 통해 함수형 컴포넌트에서도 상태관리를 할 수 있음!

함수형 컴포넌트의 useState()

```jsx
import React, { useState } from 'react';

const SayFunction = () => {
    const [ message, setMessage ] = useState("");
    const onClickEnter = () => { setMessage("안녕하세요~"); };
    const onClickLeave = () => { setMessage("안녕히가세요."); };

    return (
        <div>
            <button onClick={onClickEnter}>입장</button>
            <button onClick={onClickLeave}>퇴장</button>
            <h1>{message}</h1>
        </div>
    );
};

export default SayFunction;
```

함수형 컴포넌트의 useState()

```jsx
import React, { useState } from 'react';

const SayFunction = () => {
    const [ message, setMessage ] = useState("");
    const onClickEnter = () => { setMessage("안녕하세요~"); };
    const onClickLeave = () => { setMessage("안녕히가세요."); };

    return (
        <div>
            <button onClick={onClickEnter}>입장</button>
            <button onClick={onClickLeave}>퇴장</button>
            <h1>{message}</h1>
        </div>
    );
};

export default SayFunction;
```

- useState 함수의 인자에는 **상태의 초기값**
  - useState의 초기값은 **숫자**일 수도, **문자**일 수도, **배열**일 수도 있다. 즉, **값의 형태가 자유로움**
- useState 함수는 배열을 반환
  - 첫번째 원소: **현재 상태**
  - 두번째 원소: 상태를 바꿔주는 setter 함수

```jsx
import { useState } from "react";

const [스테이트이름, 스테이트변경함수] = useState(초기값);
```

[현재상태 , setter함수]

: useState는 배열 반환

함수형 컴포넌트의 useState()

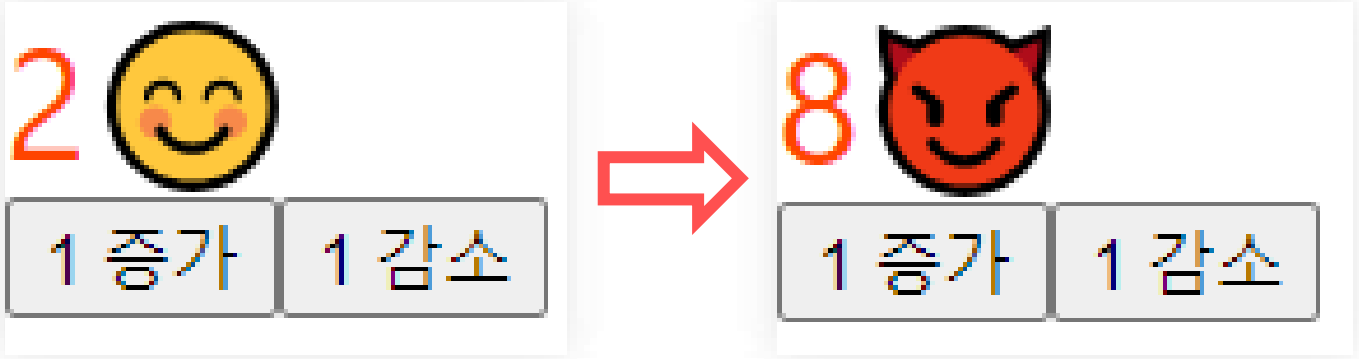
```jsx
import React, { useState } from 'react';
function Counter() {
  const [ number, setNumber ] = useState(0);
  const onClickEnter = () => setNumber(number+1);

  return (
    <div>
      <h1>{number}</h1>
      <button onClick={onClickEnter}>+1</button>
    </div>
  );
}

export default Counter;
```

추가 실습1 (선택)

- 실습1을 빨리 끝냈다면,
- 숫자뒤에 원하는 이모지를 쓰고
  숫자가 8이상이라면, (7 초과라면) 기존의 이모지에서 다른 이모지로 변경하는 코드 작성해보기

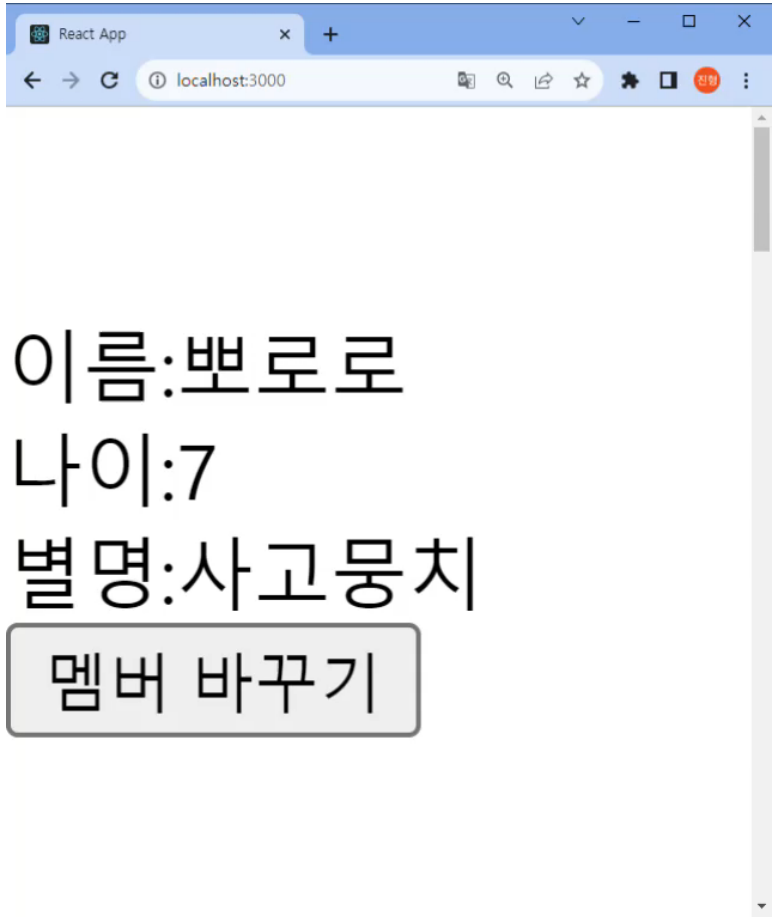


hint!! 3항 연산자 이용

추가 실습2, props와 state 사용 (선택)

- 멤버 바꾸기 버튼 클릭 → props로 전달된 objArr 값이 순서대로 변하는 Change.js 컴포넌트 만들기
- 함수형 컴포넌트로 구현!

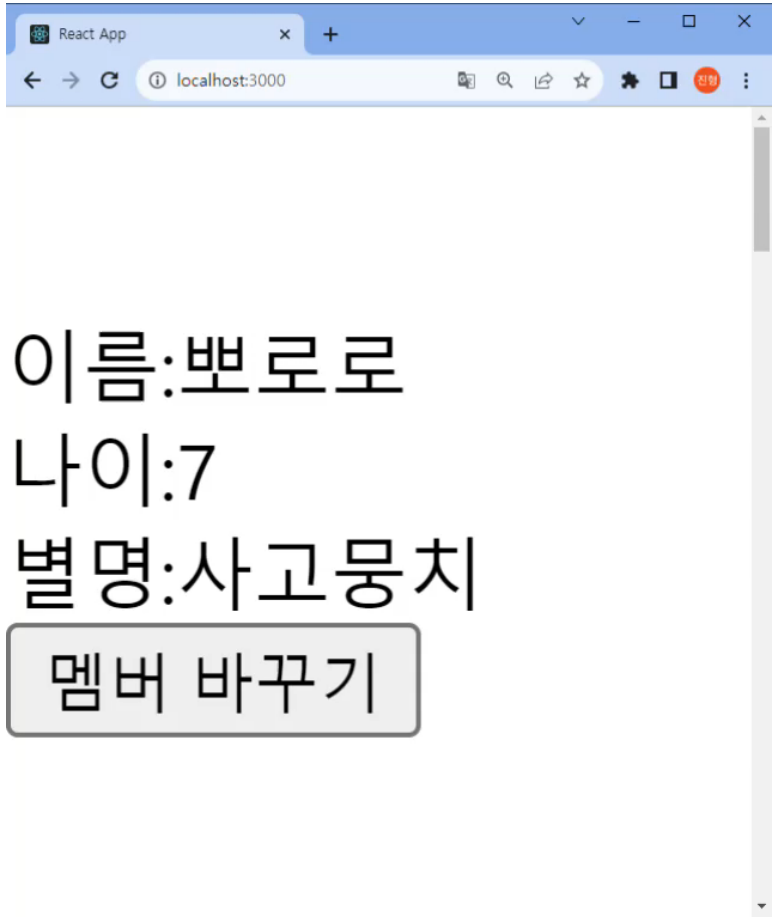
다음장 참고



추가 실습2, props와 state 사용 (선택)

- 멤버 바꾸기 버튼 클릭 → props로 전달된 objArr 값이 순서대로 변하는 Change.js 컴포넌트 만들기
- 함수형 컴포넌트로 구현!

다음장 참고



실습 참고!!

- 배열을 props로 가지는 ChangeObj 컴포넌트를 만들면 됩니다.

```
import ChangeObj from './ChangeObj';

export default function PororoObj() {
  const pororoObjArr = [
    {
      name: '뽀로로',
      age: '7',
      nickName: '사고뭉치',
    },
    {
      name: '루피',
      age: '5',
      nickName: '공주님',
    },
    {
      name: '크롱',
      age: '4',
      nickName: '장난꾸러기',
    },
  ];

  return (
    <div> <ChangeObj objArr={pororoObjArr} /> </div>
  );
}
```

pdf 에서 복사해서 사용하세요 →