

# AI 활용 빅데이터분석 풀스택웹서비스 SW 개발자 양성과정

## React



부산대학교 소프트웨어교육센터  
PUSAN NATIONAL UNIVERSITY SOFTWARE EDUCATION CENTER



# 컴포넌트 / JSX

- 컴포넌트 (Component)

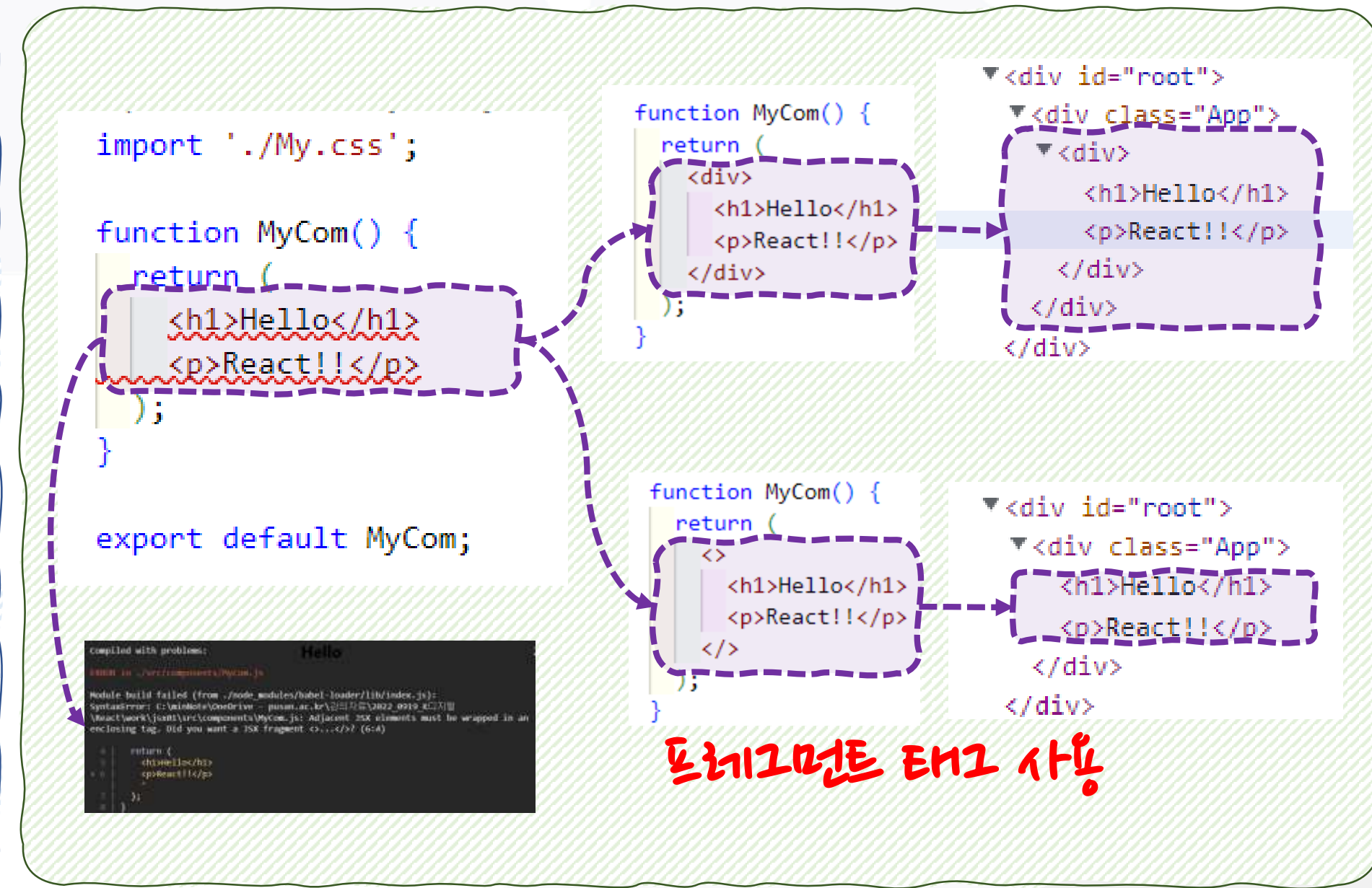
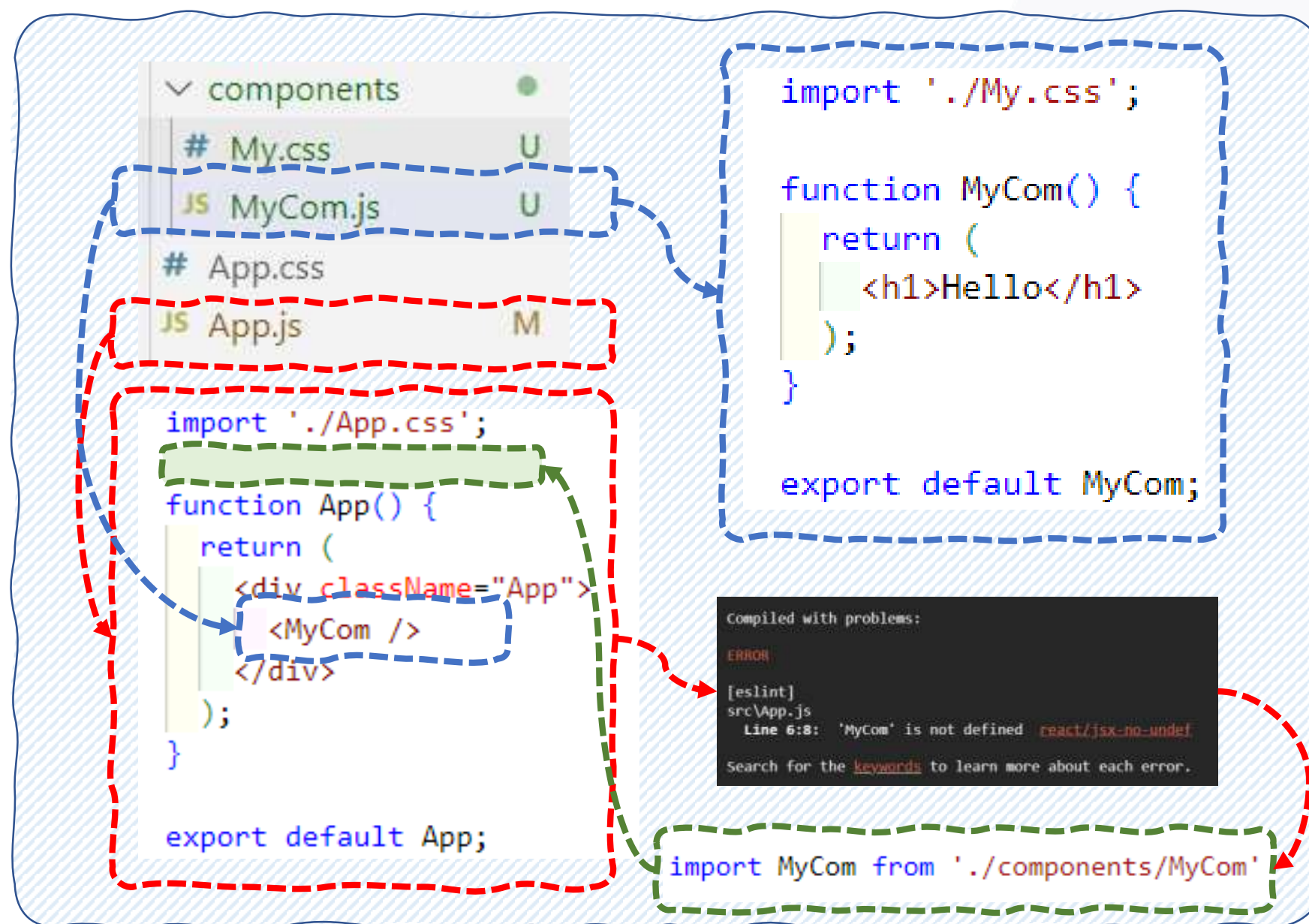
- 리액트로 개발하는 어플리케이션의 최소한의 단위로 재사용이 가능한 조각
- props라는 input값을 받아 View(state) 상태에 따라 화면에 어떻게 출력이 되는지 Output 결정하여 React Element를 반환하는 함수
- 함수형 컴포넌트와 클래스형 컴포넌트

- JSX (JavaScript XML)

- 컴포넌트 생성시 사용되는 Javascript에 XML을 추가한 확장한 문법
  - 브라우저에서 실행 전 바벨을 사용하여 일반 자바스크립트 형태의 코드로 변환
  - 자바스크립트와 HTML을 동시에 작성

# JSX 특징

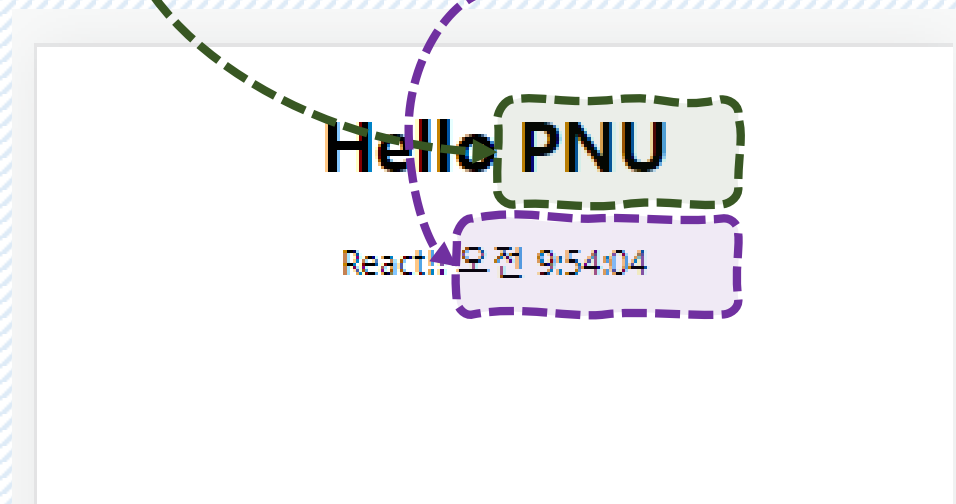
- 컴포넌트는 반드시 부모 요소 하나만 반환
  - 리액트 Virtual DOM에서 변화를 효율적으로 감지
  - `<Fragment></Fragment>`, `<></>`



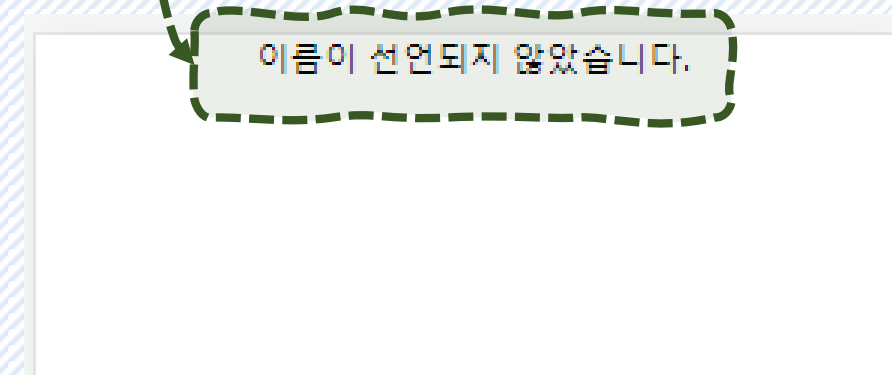
# JSX 특징

- 자바스크립트 표현식
  - {}에 작성
  - undefined를 반환하지 않도록 처리

```
function MyCom() {  
  const name = 'PNU' ;  
  return (  
    <>  
    <h1>Hello {name}</h1>  
    <p>React!! {new Date().toLocaleTimeString()}</p>  
    </>  
  );  
}
```



```
import './My.css';  
  
function MyCom() {  
  const name = undefined;  
  return name || '이름이 선언되지 않았습니다.';  
}  
  
export default MyCom;
```



**false 값으로 ||, &&연산**

– false값으로 간주되는 값  
false, 0, -0, "",  
null, undefined, NaN



# JSX 특징

## • 조건부 렌더링

– JSX 내부에는 조건문 사용할 수 없어 삼항 연산자 사용

```
function MyCom() {  
  const name = 'PNU';  
  
  return (  
    <>  
      if (name === 'PNU') {  
        <h1>{name}님 안녕하세요.</h1>  
      }  
      else {  
        <h1>Hello </h1>  
      }  
    <p>React!! {new Date().toLocaleTimeString()}</p>  
    </>  
  );  
}
```

if (name === 'PNU')

**PNU님 안녕하세요.**

else

**Hello**

React!! 오전 10:08:06

### 변수사용

```
function MyCom() {  
  const name = 'PNU';  
  
  let pname ;  
  if (name === 'PNU') pname = <h1>{name}님 안녕하세요.</h1>  
  else pname = <h1>Hello </h1>  
  
  return (  
    <>  
      {pname}  
      <p>React!! {new Date().toLocaleTimeString()}</p>  
    </>  
  );  
}
```

**PNU님 안녕하세요.**

React!! 오전 10:18:10

### 삼항 연산자 사용

```
function MyCom() {  
  const name = 'PNU';  
  
  return (  
    <>  
      { (name === 'PNU') ? <h1>{name}님 안녕하세요.</h1> : <h1>Hello </h1> }  
      <p>React!! {new Date().toLocaleTimeString()}</p>  
    </>  
  );  
}
```

**PNU님 안녕하세요.**

React!! 오전 10:37:21

# JSX 특징

## • 스타일 시트 적용

### 인라인 형식 : 객체형식으로

```
function MyCom() {  
  const name = 'PNU';  
  
  return (  
    <>  
      <h1>{name}님 안녕하세요.</h1>  
      <p style={  
        {  
          color: '#21005D',  
          backgroundColor: '#EADDFD',  
          fontSize: '2rem'  
        }  
      }>React!!</p>  
    </>  
  );  
}
```

### 변수사용 : 객체

```
function MyCom() {  
  const name = 'PNU';  
  const inStyle = {  
    color: '#21005D',  
    backgroundColor: '#EADDFD',  
    fontSize: '2rem'  
  };  
  
  return (  
    <>  
      <h1>{name}님 안녕하세요.</h1>  
      <p style={inStyle}>React!!</p>  
    </>  
  );  
}
```

```
<div id="root">  
  <div class="App">  
    <h1>...</h1>  
    <p style="color: rgb(33, 0, 93); background-color: rgb(234, 221, 255); font-size: 2rem;">React!!</p>  
  </div>  
</div>
```

### 스타일 시트 파일 : 캐스캐이딩 (cascading) Index.html 파일에 CSS파일을 추가

```
import './My.css';  
  
function MyCom() {  
  const name = 'PNU';  
  
  return (  
    <>  
      <h1>{name}님 안녕하세요.</h1>  
      <p className="myp">React!!</p>  
    </>  
  );  
}  
  
export default MyCom;
```

App.css에만  
클래스스타일 정의된 경우

PNU님 안녕하세요.

React!!

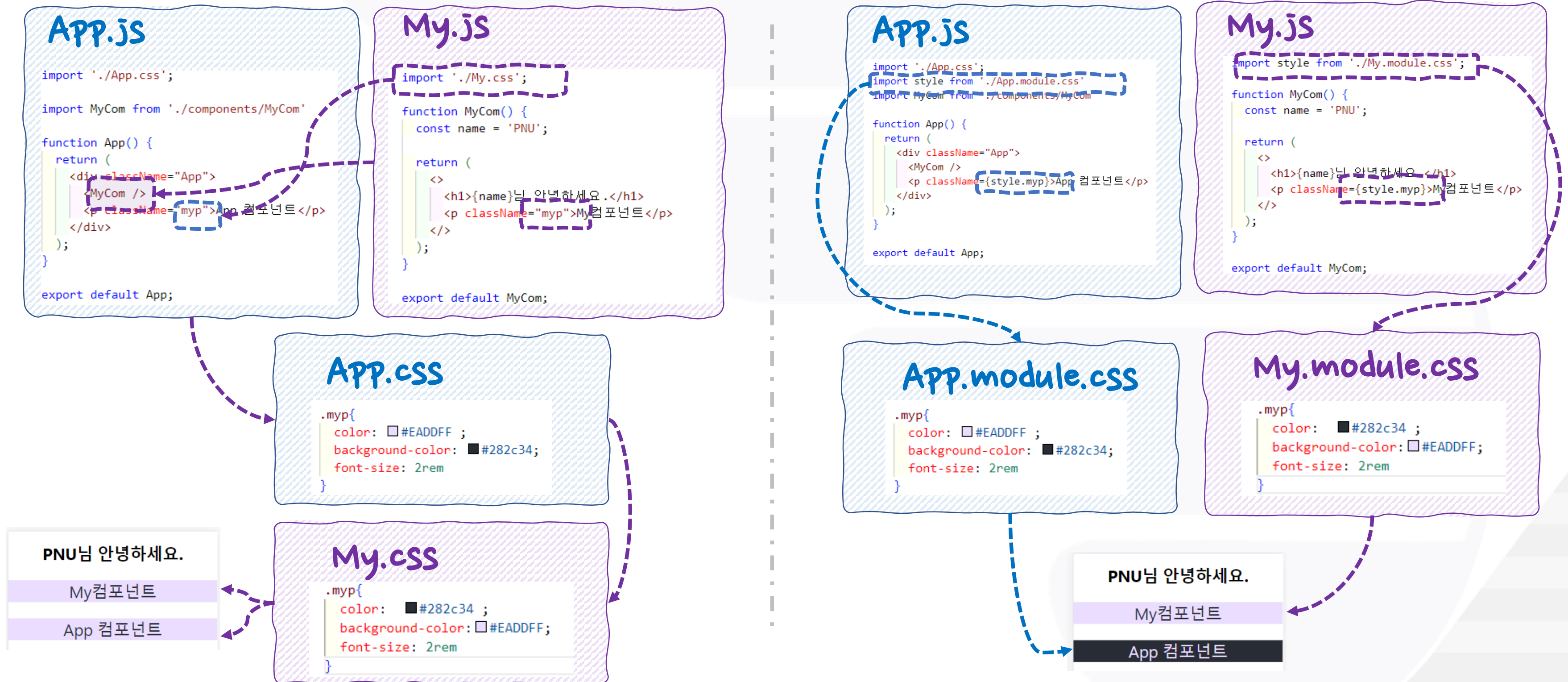
App.css와 My.css에  
정의된 경우 캐스캐이딩

PNU님 안녕하세요.

React!!

# JSX 특징

## • 각 컴포넌트에 서로 다른 스타일 시트 적용





# 오브젝트(Object)

- 자바스크립트 기본 데이터 타입으로 키와 값의 쌍으로 이루어진 집합
  - 값은 함수(메소드)를 가질 수도 있음
- **오브젝트 생성**
  - 리터럴 표기법
  - 생성자 함수 이용한 방법 : 객체를 생성하고 초기화
- **프로토타입(prototype)**
  - 자바스크립트 모든 객체는 하나 이상의 다른 객체로 상속 받으며 상속되는 정보를 제공하는 프로토타입 객체를 가짐
- **오브젝트 값 참조**
  - 오브젝트.키
  - 오브젝트[키] => 키 명은 문자열
  - 오브젝트.메소드()
- **오브젝트 순회**
  - for (let k in 오브젝트)
  - for (let [k, v] of Object.entries(오브젝트))
- **오브젝트 복사**
  - 얇은 복사 : 오브젝트copy = 오브젝트
  - 깊은 복사 : 오브젝트copy = {...오브젝트}

```
//생성자 함수를 이용한 생성
obj = new Object();
console.log(obj);
```

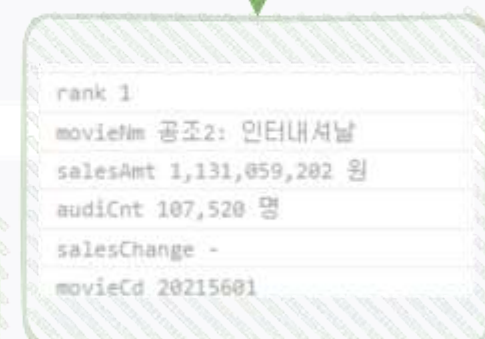
```
//리터럴 오브젝트 생성
obj = {
  rank : "1",
  movieNm : "공조2: 인터내셔날",
  salesAmt : "1,131,059,202 원",
  audiCnt : "107,520 명",
  salesChange : "-",
  movieCd : "20215601"
}
console.log(obj);
```

```
//오브젝트 값 참조
//.연사자 표기법
console.log(obj.movieNm);
//대괄호 표기법
console.log(obj["movieNm"]);
```

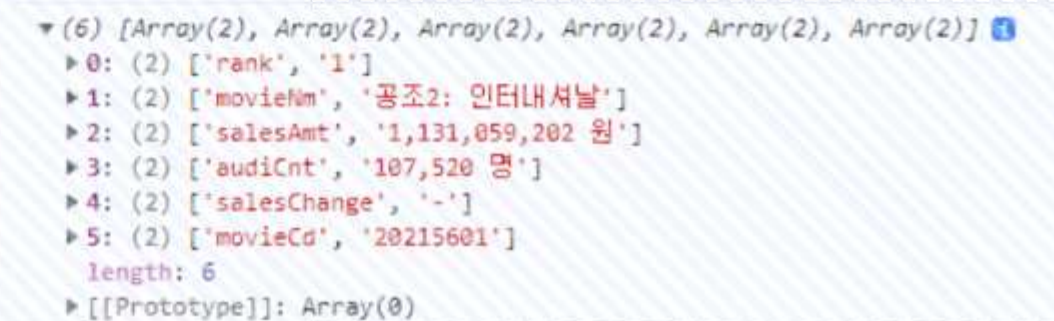
```
//오브젝트 순회
for (let k in obj) { console.log(k, obj[k]) }
for (let [k, v] of Object.entries(obj)) { console.log(k, v) }
```

```
//오브젝트 복사
let obj2 = {...obj};
obj2.rank = "2";
console.log('원본 =>', obj.rank);
console.log('복사본 =>', obj2.rank);
```

```
//오브젝트를 배열로
let objToarr = Object.entries(obj);
console.log(objToarr);
```



rank	1
movieNm	공조2: 인터내셔날
salesAmt	1,131,059,202 원
audiCnt	107,520 명
salesChange	-
movieCd	20215601



```
▼ (6) [Array(2), Array(2), Array(2), Array(2), Array(2), Array(2)] ⓘ
▶ 0: (2) ['rank', '1']
▶ 1: (2) ['movieNm', '공조2: 인터내셔날']
▶ 2: (2) ['salesAmt', '1,131,059,202 원']
▶ 3: (2) ['audiCnt', '107,520 명']
▶ 4: (2) ['salesChange', '-']
▶ 5: (2) ['movieCd', '20215601']
length: 6
▶ [[Prototype]]: Array(0)
```



# 컴포넌트에서 오브젝트 사용

```
function MyCom() {  
  const mv = {  
    rank: "1",  
    movieNm: "공조2: 인터내셔날",  
    salesAmt: "1,131,059,202 원",  
    audiCnt: "107,520 명",  
    salesChange: "-",  
    movieCd: "20215601"  
  }  
  return (  
    <>  
    <h1>박스오피스</h1>  
    <main>  
      <ul>  
        <li>{mv.rank}</li>  
        <li>{mv.movieNm}</li>  
        <li>{mv.salesAmt}</li>  
        <li>{mv.audiCnt}</li>  
        <li>{mv.salesChange}</li>  
        <li>{mv.movieCd}</li>  
      </ul>  
    </main>  
    </>  
  );  
}
```

박스오피스	
1	공조2: 인터내셔날
	1,131,059,202 원
	107,520 명
	-
	20215601

```
function MyCom() {  
  const mv = {  
    rank: "1",  
    movieNm: "공조2: 인터내셔날",  
    salesAmt: "1,131,059,202 원",  
    audiCnt: "107,520 명",  
    salesChange: "-",  
    movieCd: "20215601"  
  }  
  const lis = [] ;  
  for (let k in mv) {  
    lis.push(<li key={k + mv.movieCd}>{mv[k]}</li>)  
  }  
  console.log(lis)  
  return (  
    <>  
    <h1>박스오피스</h1>  
    <main>  
      <ul>  
        {lis}  
      </ul>  
    </main>  
    </>  
  );  
}
```

오브젝트를 순회하면서  
배열을 생성

# 컴포넌트 이벤트 추가

```
function MyCom() {  
  const mv = { ...  
  }  
  const lis = [] ;  
  for (let k in mv) { ...  
  }  
  //console.log(lis)  
  
  let count=0;  
  const upCount = () => {  
    count++ ;  
    console.log(count)  
  }  
  return (  
    <>  
    <h1>박스오피스</h1>  
    <main>  
      <ul> ...  
      </ul>  
      <section className="sbt">  
        <span onClick={upCount}>❤️</span>  
        <span>{count}</span>  
      </section>  
    </main>  
    </>  
  );  
}
```

함수작성

작성된 함수명  
cf) 함수명()과 구분

Console  
1  
2  
3  
4  
5  
6

```
function MyCom() {  
  const mv = { ...  
  }  
  const lis = [];  
  for (let k in mv) { ...  
  }  
  //console.log(lis)  
  
  let count = 0;  
  return (  
    <>  
    <h1>박스오피스</h1>  
    <main>  
      <ul> ...  
      </ul>  
      <section className="sbt">  
        <span onClick={() => {  
          count++;  
          console.log(count)  
        }}>❤️</span>  
        <span>{count}</span>  
      </section>  
    </main>  
    </>  
  );  
}
```

함수작성

```
function MyCom() {  
  const mv = { ...  
  }  
  const lis = [] ;  
  for (let k in mv) { ...  
  }  
  //console.log(lis)  
  
  let count=0;  
  const upCount = (n) => {  
    count = count + n ;  
    console.log(count)  
  }  
  return (  
    <>  
    <h1>박스오피스</h1>  
    <main>  
      <ul> ...  
      </ul>  
      <section className="sbt">  
        <span onClick={  
          () => {  
            upCount(10);  
          }  
        }>❤️</span>  
        <span>{count}</span>  
      </section>  
    </main>  
    </>  
  );  
}
```

인수 전달 시 사용

Console  
10  
20  
30  
40  
50  
60

# 컴포넌트 폼 이벤트 처리

```
<form onSubmit={ (e) => {
  e.preventDefault();
  e.target.reset();
  alert('등록')
}}>
```

글자가

1. **폼 submit시 브라우저 새로 고침 방지**  
 2. **폼 입력값 초기화**

## 글자가 수정될 경우

```
<input type="text" onChange={(e) => {  
  console.log(e.target.value);  
}} />
```

또 새로 리셋

## 포괄내용리셋

```
<button type="reset">취소 </button>
```

```
<button type="submit">Submit</button>
```

## 품의 onSubmit이벤트 발생

## 박스오피스

1  
공조2: 인터내셔널  
1,131,059,202 원  
107,520 명  
-  
20215601

0

댓글

재미있어요

취소 등록

Console Net

top ▼

天

재

재  
□

재

ΣΗΠΙ

재민

THE

ΣΗΠΙΟΙ

재미있

재미있다

재미있게

재미있어

재미있게

재미있어요



# 컴포넌트 props

- 컴포넌트는 사용자 정의 태그임으로 태그의 속성을 가짐
- 이러한 속성들은 컴포넌트 생성시 함수의 매개변수로 받음
- 이 매개변수는 속성들을 가지는 object
- 읽기 전용으로 수정해서는 안됨

# 컴포넌트 probs

## App.js

```
function App() {  
  const items = [  
    { rank: "1", movieNm: "공조2: 인터내셔날", salesAmt: "1,131,059,202 원" },  
    { rank: "2", movieNm: "정직한 후보2", salesAmt: "927,485,030 원" }  
  ];  
  
  return (  
    <main>  
      <MyHeader />  
      <MyCom item={items[0]} />  
      <MyCom item={items[1]} />  
    </main>  
  );  
}
```

## MyHeader.js

```
function MyHeader() {  
  return (<h1>박스오피스</h1>) ;  
}
```

## Mycom.js

```
function MyCom(probs) {  
  const mv = {...probs.item};  
}
```

## Mycom.js

```
function MyCom(probs) {  
  const mv = {...probs.item};  
}
```

```
probs : ▼ {rank: '1', movieNm: '공조2: 인터내셔날', salesAmt: '1,131,059,202 원',  
  audiCnt: '107,520 명',  
  movieCd: '20215601',  
  movieNm: '공조2: 인터내셔날',  
  rank: '1',  
  salesAmt: '1,131,059,202 원',  
  salesChange: '-'}  
▶ [[Prototype]]: Object
```

```
전개연산자 : ▼ {rank: '1', movieNm: '공조2: 인터내셔날', salesAmt: '1,131,059,202 원',  
  audiCnt: '107,520 명',  
  movieCd: '20215601',  
  movieNm: '공조2: 인터내셔날',  
  rank: '1',  
  salesAmt: '1,131,059,202 원',  
  salesChange: '-'}  
▶ [[Prototype]]: Object
```

```
probs : ▼ {rank: '2', movieNm: '정직한 후보2', salesAmt: '927,485,030 원',  
  audiCnt: '89,931 명',  
  movieCd: '20219628',  
  movieNm: '정직한 후보2',  
  rank: '2',  
  salesAmt: '927,485,030 원',  
  salesChange: '-'}  
▶ [[Prototype]]: Object
```

```
전개연산자 : ▼ {rank: '2', movieNm: '정직한 후보2', salesAmt: '927,485,030 원',  
  audiCnt: '89,931 명',  
  movieCd: '20219628',  
  movieNm: '정직한 후보2',  
  rank: '2',  
  salesAmt: '927,485,030 원',  
  salesChange: '-'}  
▶ [[Prototype]]: Object
```

# 리액트 Hook

- 함수 컴포넌트에서 React state와 생명주기 기능(lifecycle features)을 “연동(hook into)”할 수 있게 해주는 함수
  - React 버전 16.8부터 React 요소로 새로 추가
  - 상태 관련 로직을 추상화해 독립적인 테스트와 재사용이 가능해 레이어 변화 없이 재사용
  - 기존의 라이프사이클 메서드 기반이 아닌 로직 기반으로 나눌 수 있어서 컴포넌트를 함수 단위로 잘게 쪼갤 수 있다는 이점
- Hook 사용 규칙
  - 최상위에서만 Hook을 호출
    - 반복문, 조건문, 중첩된 함수 내에서 Hook을 실행하면 안됨
    - 이 규칙을 따르면 컴포넌트가 렌더링될 때마다 항상 동일한 순서로 Hook이 호출되는 것이 보장
  - 리액트 함수 컴포넌트에서만 Hook을 호출
    - JS함수에서는 Hook을 호출해서는 안됨



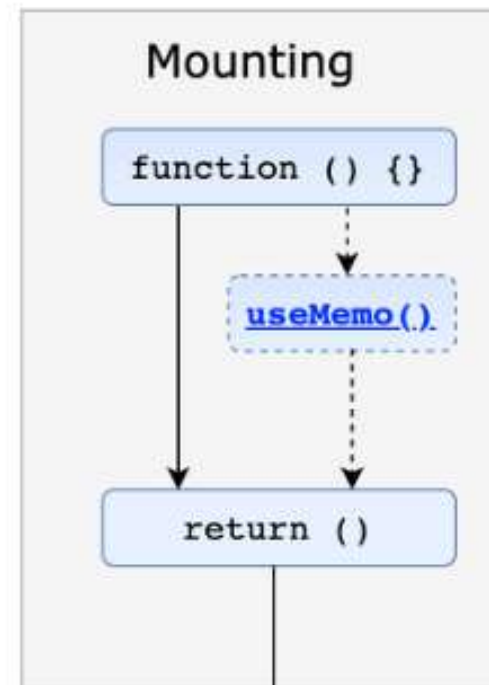
# 리액트 Hook

- 함수 컴포넌트에서 React state와 생명주기 기능(lifecycle features)을 “연동(hook into)”할 수 있게 해주는 함수
  - React 버전 16.8부터 React 요소로 새로 추가
  - 상태 관련 로직을 추상화해 독립적인 테스트와 재사용이 가능해 레이어 변화 없이 재사용
  - 기존의 라이프사이클 메서드 기반이 아닌 로직 기반으로 나눌 수 있어서 컴포넌트를 함수 단위로 잘게 쪼갤 수 있다는 이점

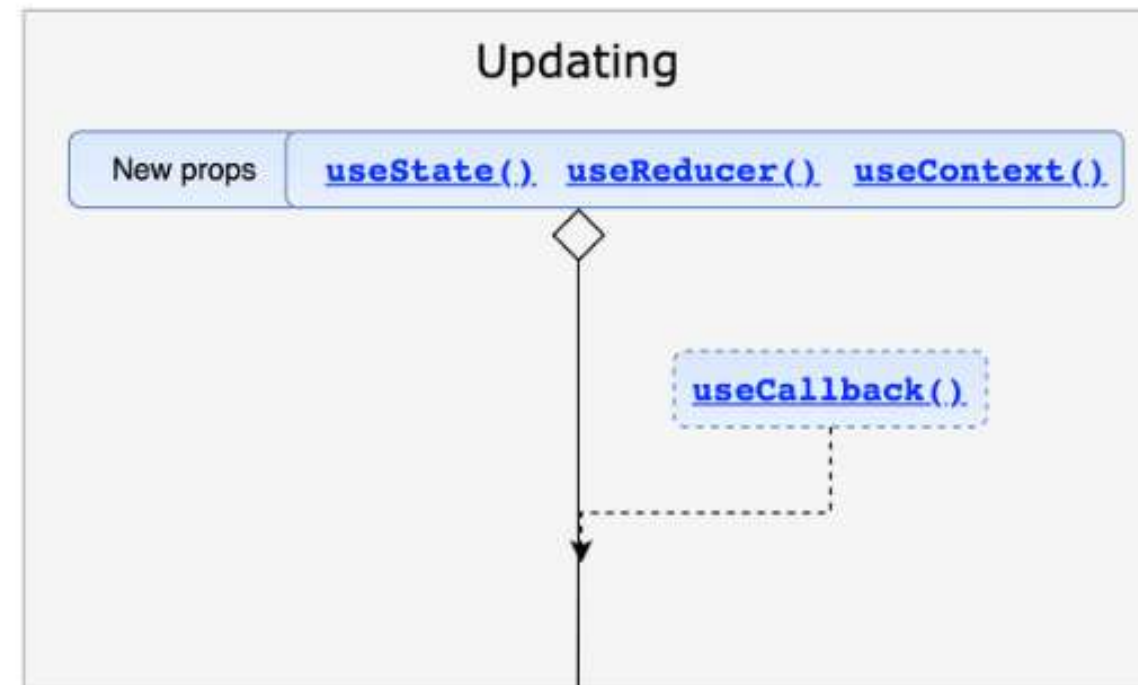
# 리액트 Hook 생명주기

## "Render phase"

Pure and has no side effects. May be paused, aborted or restarted by React.

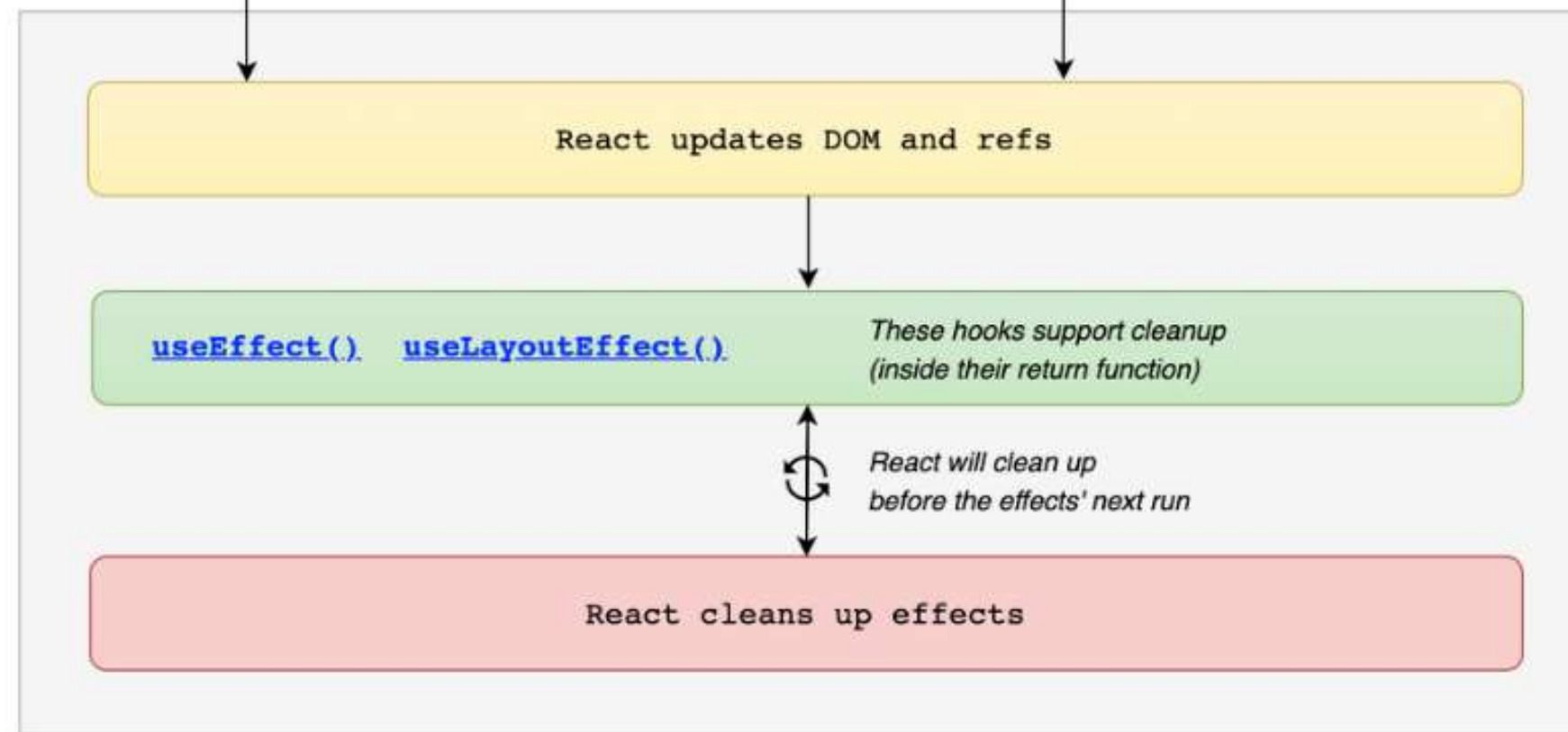


## Updating



## "Commit phase"

Can work with DOM, run side effects, schedule updates.



## "Cleanup phase"

Runs before a component is removed. Prevents memory leaks.

## 생명주기 순서

1. 함수형 컴포넌트가 호출
2. 함수형 컴포넌트 내부 실행
3. return()으로 화면에 렌더링
4. useEffect()가 실행

[출처] <https://wavez.github.io/react-hooks-lifecycle/>

# Hook : useState

- 컴포넌트 state

- 컴포넌트가 가지는 상태값으로 상태값이 변경되면 화면이 재 랜더링

- useState

- 동적으로 변경되는 값을 관리할 때 사용하며 상태 유지값과 그값을 갱신하는 함수를 반환
  - `const [state, setState] = useState(initialState);`

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

1. useState import

```
function MyCom(probs) {  
  const mv = {...probs.item};
```

```
  //오브젝트 구조분해  
  //const { movieNm } = mv ;  
  //console.log(movieNm)
```

2. useState()는 배열 반환

=>구조분해 할당

```
  const lis = [] ;  
  for (let k in mv) { ...  
  }  
  let [count, setCount] = useState(0);
```

```
  const upCount = () => {  
    setCount(count++);  
  }
```

4. setcount()에 의해 state

count 변수 값이 변경

```
  return (  
    <>  
    <ul>  
      {lis}  
    </ul>  
    <section className="sbox">  
      <span onClick={upCount}>❤️</span>  
      <span>{count}</span>  
    </section>  
  </>  
  );  
}
```

3. onclick이벤트 처리

5. State의 변수 값이 변경되면

변경되면 DOM에 바로 반영

구조분해(Destructuring)

할당

배열이나 객체의 속성을  
분해해서 그 값을 변수에  
담을 수 있게 하는 표현식

## 박스오피스

1  
공조2: 인터내셔널  
1,131,059,202 원  
107,520 명  
-  
20215601

❤️ 5

2  
정직한 후보2  
927,485,030 원  
89,931 명  
-  
20219628

❤️ 2

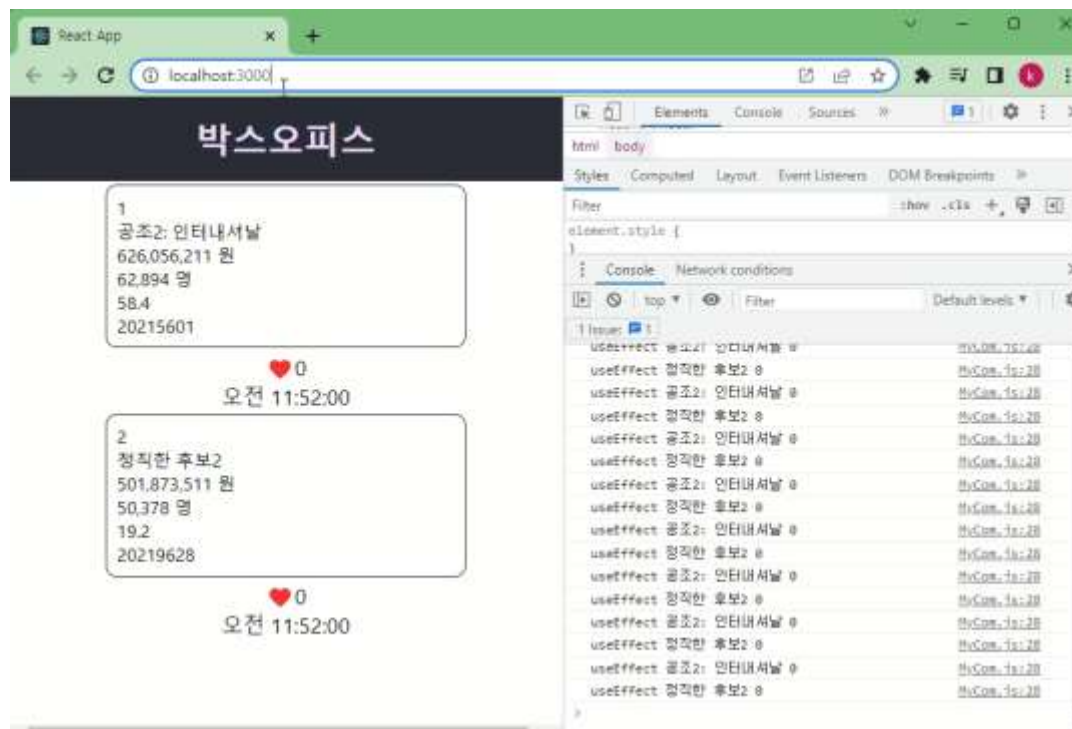


# Hook : useEffect

- 컴포넌트 내에서 렌더링이 수행된 이후 실행되는 메서드
- `useEffect(() => {}, dependency값)`

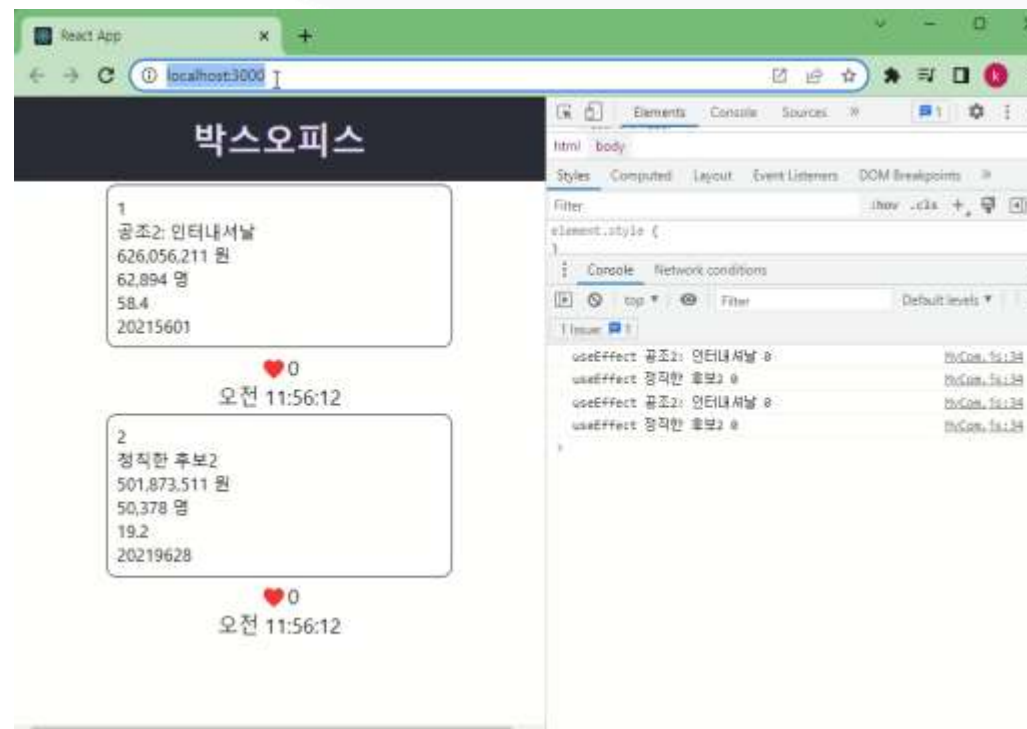
## dependency가 없을 경우

```
//dependency가 없을 경우 : 컴포넌트가 렌더링 될때마다 실행
useEffect(() => {
  console.log(`useEffect ${mv.movieNm} ${count}`)
})
```



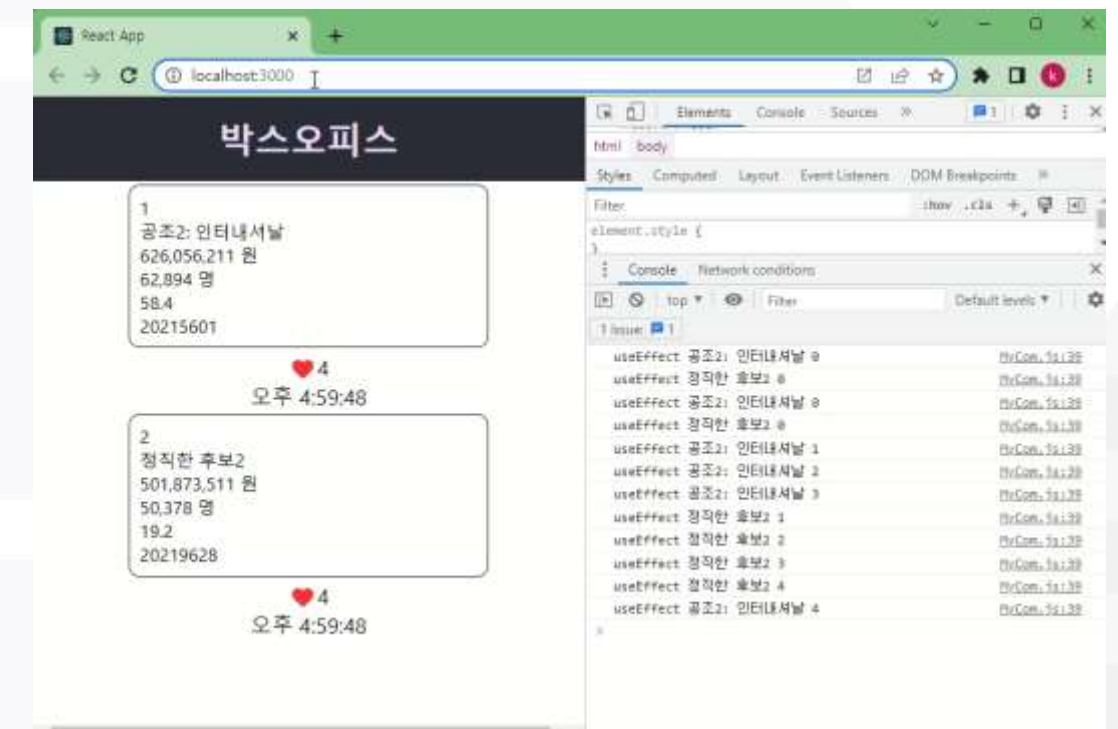
## dependency가 빈배열

```
//dependency가 빈배열 : 컴포넌트가 처음 렌더링되었을때 실행
useEffect(() => {
  //dependency가 배열에 값이 있을 경우 : 값이 변경될때 실행
  console.log(`useEffect ${mv.movieNm} ${count}`)
}, [])
```



## dependency배열에 값이 있을 경우

```
//dependency가 배열에 값이 있을 경우 : 값이 변경될때 실행
useEffect(() => {
  console.log(`useEffect ${mv.movieNm} ${count}`)
}, [count])
```



# Hook : useEffect cleanup

- 컴포넌트가 unmount되어 DOM에서 제거될 경우 실행되는 메소드 `useEffect(() => { ... return ()=> {} }, dependency값)`

```
Mycom

let [timerFlag, setTimerFlag] = useState();
const handleTimer = () => {
  setTimerFlag(!timerFlag);
}

return (
  <>
    <ul className='boxul' onClick={handleTimer}>
      </ul>
      <section className="sbox">...
      </section>
      {timerFlag && <MyComTimer />}
    </>
  );
```

Toggle 기능 구현

false일때 조건부 렌더링

```
MycomTimer

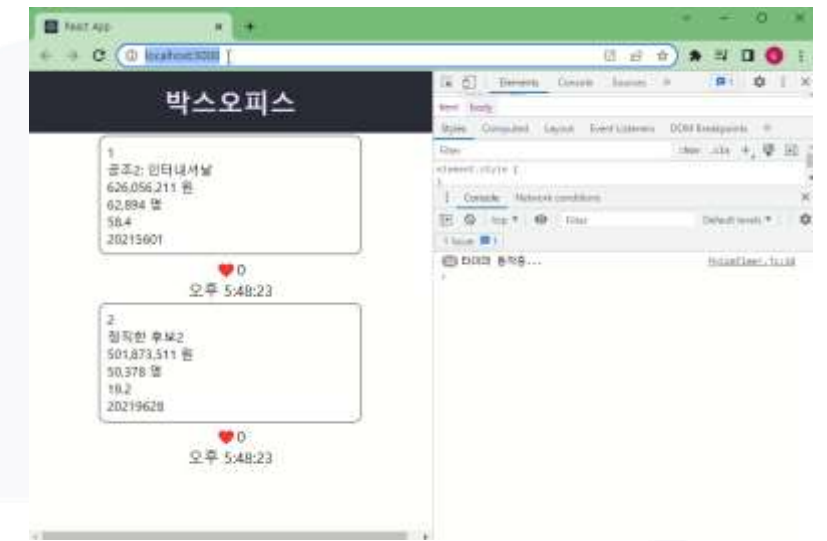
function MyComTimer() {
  const [timeValue, setTimeValue] = useState();

  useEffect(() => {
    const timer = setInterval(() => {
      setTimeValue(new Date().toLocaleTimeString());
      console.log('타이머 동작중...');
    }, 1000);

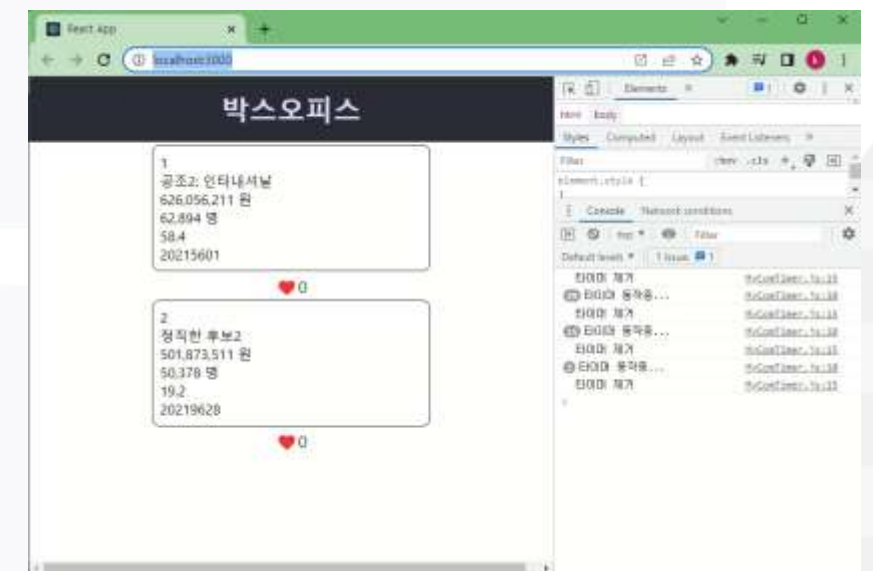
    return () => {
      clearInterval(timer);
    };
  }, []);

  return (
    <section className="sbox">
      <span>{timeValue}</span>
    </section>
  );
}
```

컴포넌트 제거후에도 계속 동작



cleanup 처리 후에는 컴포넌트 제거



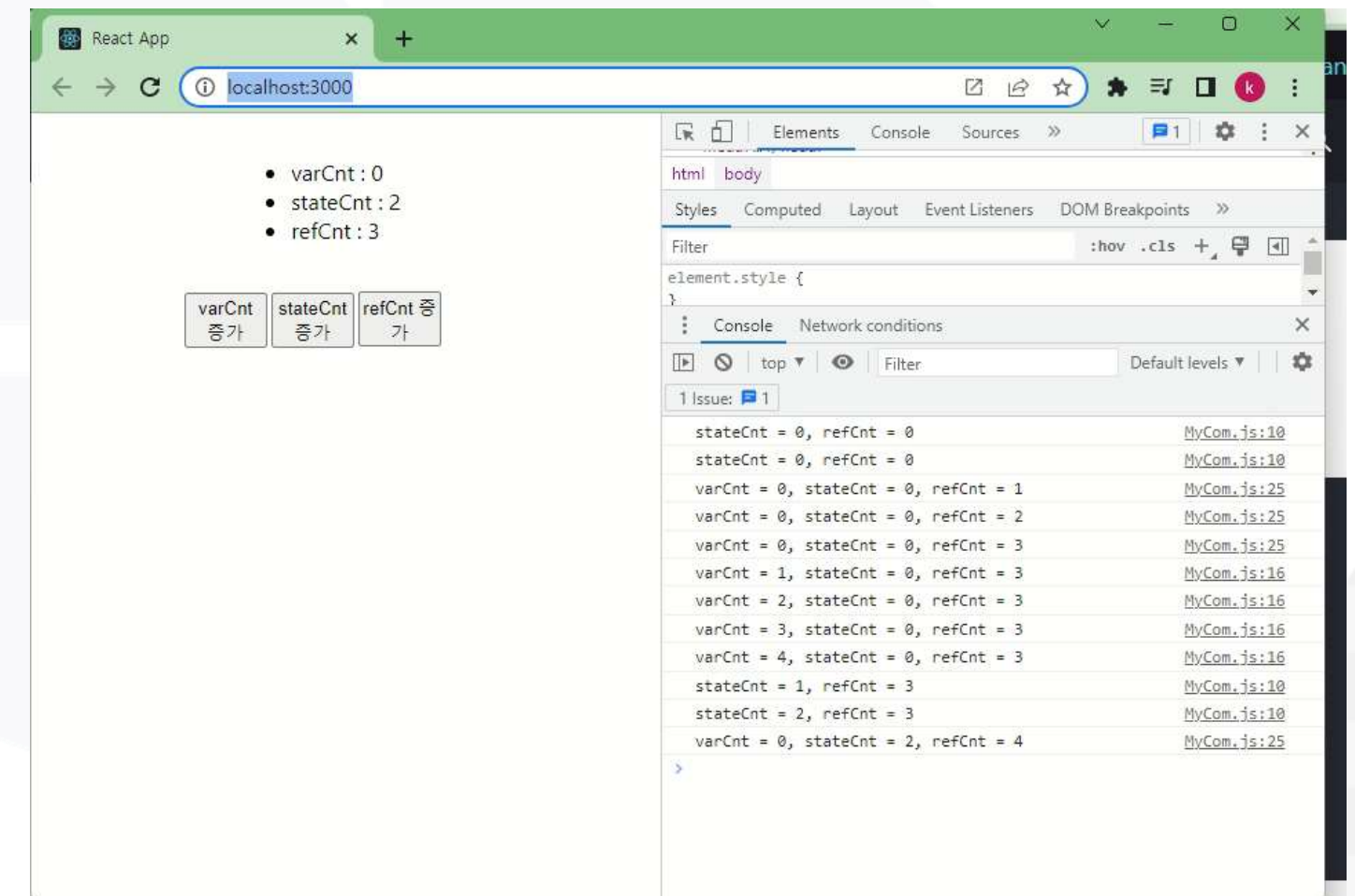
# Hook : useRef()

- .current 프로퍼티로 전달된 인자(initialValue)로 초기화된 변경 가능한 ref 객체를 반환하고 반환된 객체는 컴포넌트의 전 생애주기를 통해 유지

```
function MyCom() {  
  let varCnt = 0;  
  const [stateCnt, setStateCnt] = useState(0);  
  const refCnt = useRef(0);  
  
  useEffect(() => {  
    console.log(`stateCnt = ${stateCnt}, refCnt = ${refCnt.current}`) ;  
  }, [stateCnt]);  
  
  const handleVar = () => {  
    varCnt++;  
    console.log(`varCnt = ${varCnt}, stateCnt = ${stateCnt}, refCnt = ${refCnt.current}`) ;  
  };  
  
  const handleState = () => {  
    setStateCnt(stateCnt + 1);  
  };  
  
  const handleRef = () => {  
    refCnt.current = refCnt.current + 1 ;  
    console.log(`varCnt = ${varCnt}, stateCnt = ${stateCnt}, refCnt = ${refCnt.current}`) ;  
  };  
}
```

state변수값이 변경되면 화면이 재 렌더링이 되고 컴포넌트 변수는 초기화

current속성에 값저장  
ref변수 값은 변경이되어도 재렌더링이 되지 않고 재렌더링이 된 경우에도 이전 값을 그대로 유지





# Hook : useRef DOM 핸들링

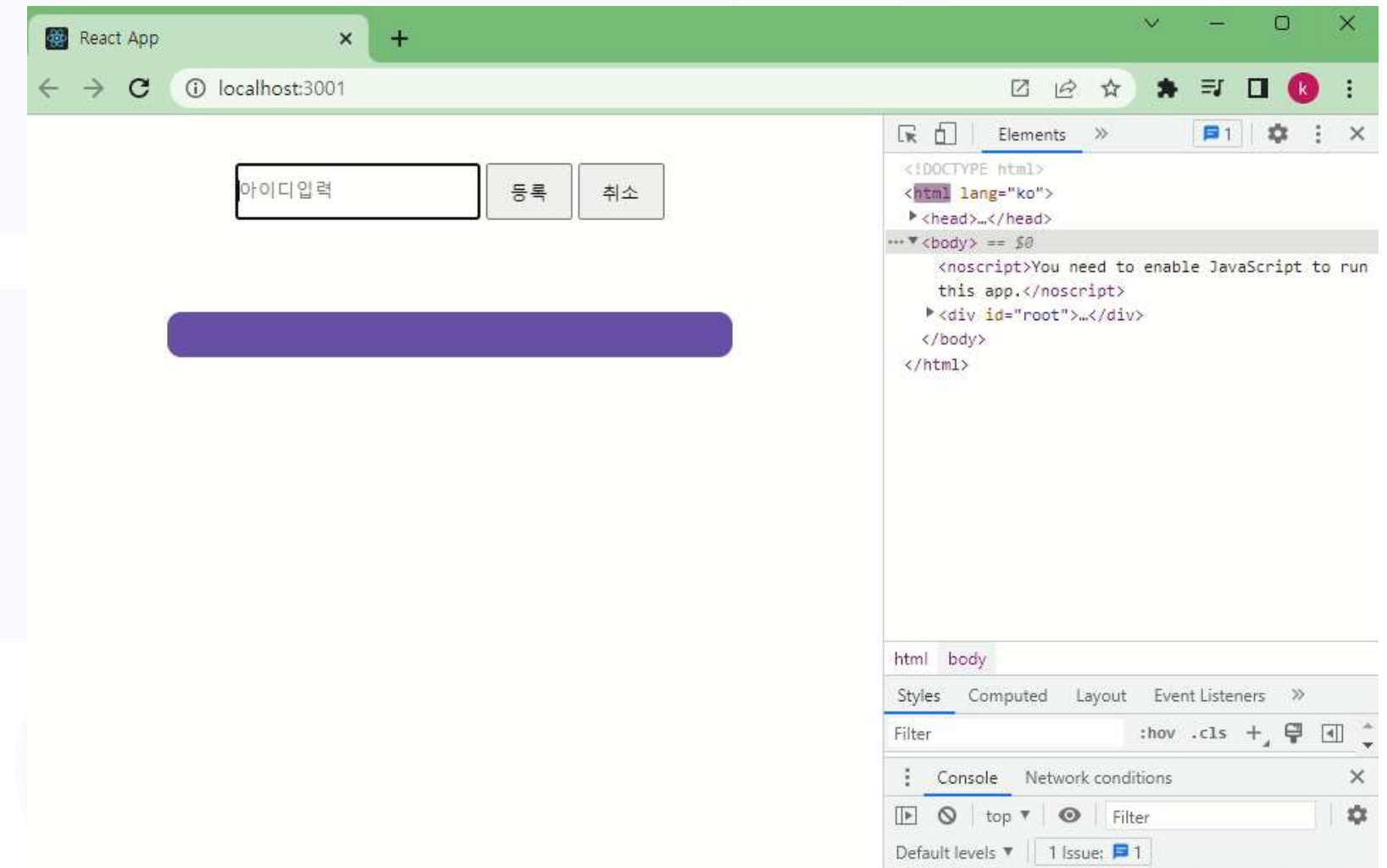
```
function MyCom() {  
  const txtRef = useRef();  
  const [txtState, setTxtState] = useState([]);  
  const [txtList, setTxtList] = useState([]);  
  
  useEffect(() => {  
    txtRef.current.focus();  
  }, []);  
  
  useEffect(() => {  
    setTxtList(txtState.map((item) => {return - }, [txtState]);  
  
  const handleClick = (e) => {  
    e.preventDefault();  
    setTxtState([txtRef.current.value, ...txtState]);  
    txtRef.current.value = '';  
    txtRef.current.focus();  
  }  
  
  const handleReset = () => {  
    txtRef.current.focus();  
    setTxtState([]);  
    setTxtList([]);  
  }  
  
  return (  
    <>  
      <form>  
        <input ref={txtRef} type="text" placeholder='아이디입력' />  
        <button onClick={handleClick}>등록</button>  
        <button type="reset" onClick={handleReset}>취소</button>  
      </form>  
      <section className='msg'>  
        { txtList }  
      </section>  
    </>  
  );  
}

```

컴포넌트가 맨처음 렌더링되면  
input 요소에 포커스

기존 배열에 추가

input 요소에 ref속성을 지정하여 useRef 훅으로 지정한 변수 지정



# 실습과제

← → ↻ ⓘ localhost:3000/?

박스오피스

영화명입력

1 공조2: 인터내셔널  0

2 정직한 후보2  0

3 인생은 아름다워  0

- 영화명 : 공조2: 인터내셔널
- 매출액 : 626,056,211 원
- 관객수 : 62,894 명
- 증감율 : 58.4

# 실습과제

React App - JSON  
localhost:3000

## 박스오피스

	영화명	매출액	관객수	증감율
1	공조2: 인터내셔날	852,947,696	4,919,945	▼ 13.7
2	육사오(6/45)	107,823,318	1,860,281	▼ 5
3	헌트	32,176,484	4,335,212	▲ 3.1
4	인생은 아름다워	32,463,000	13,710	▲ 133
5	정직한 후보2	23,454,000	2,606	▲ 100
6	알라딘	24,049,000	12,797,927	▲ 32.5
7	인생은 뷰티풀: 비타돌체	58,615,500	40,664	▲ 56.4
8	극장판 엄마 까투리: 도시로 간 까투리 가족	14,463,520	143,856	▼ 20.5
9	블랙폰	11,915,897	104,242	▼ 9.2
10	한산: 용의 출현	10,721,100	7,252,472	▲ 14.3

육사오(6/45)는 전일대비 순위 증감분은 0입니다.

```
db
  {} boxoffice.json U

import db from '../db/boxoffice.json'
import {useState, useEffect} from 'react'

function BoxRow() {
  const dailyBoxOfficeList = db.boxOfficeResult.dailyBoxOfficeList ;

  "boxOfficeResult": {
    "boxofficeType": "일별 박스오피스",
    "showRange": "20220920~20220920",
    "dailyBoxOfficeList": [
      {
        "rnum": "1",
        "rank": "1",
        "rankInten": "0",
        "rankOldAndNew": "OLD",
        "movieCd": "20215601",
        "movieNm": "공 조2: 인터내 셔날",
        "openDt": "2022-09-07",
        "salesAmt": "852947696",
        "salesShare": "67.8",
        "salesInten": "-147941303",
        "salesChange": "-14.8",
        "salesAcc": "50716377449",
        "audiCnt": "86892",
        "audiInten": "-13852",
        "audiChange": "-13.7",
        "audiAcc": "4919945",
        "scrnCnt": "2109",
        "showCnt": "9129"
      },
      { ...
    ]
  },
}
```