Chapter 5. 컴포넌트와 Props

교재: 처음만난 리액트 (저자: 이인제, 한빛출판사)

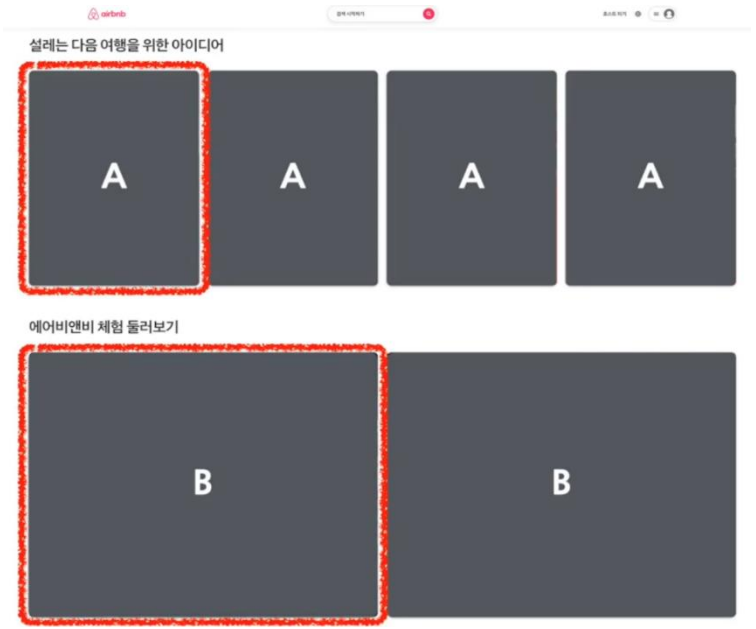
Contents

- CHAPTER 5: 컴포넌트와 Props

SECTION 5.1 컴포넌트에 대해 알아보기

◦ 컴포넌트 기반 구조 (Component-Based)

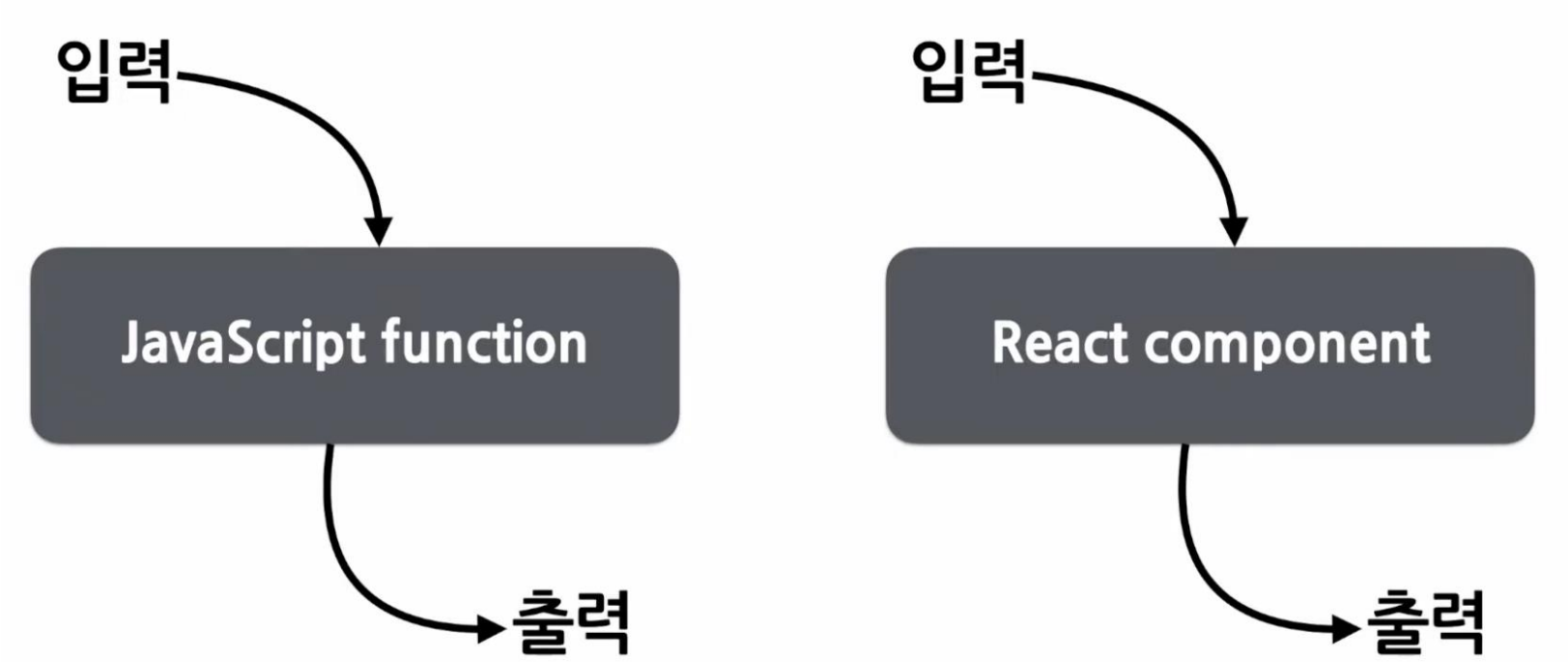
- 컴포넌트 - 독립적인 기능을 수행하는 작은 기능 단위 모듈
- 리액트에서는 모든 페이지가 컴포넌트로 구성됨
- 하나의 컴포넌트는 또 다른 여러 개의 컴포넌트의 조합으로 구성될 수 있음
  - 레고 블록을 조립하는 것처럼 컴포넌트를 조합해서 사용



- 컴포넌트 사용 - 개발 시간을 줄이고 유지보수 비용도 줄일 수 있음

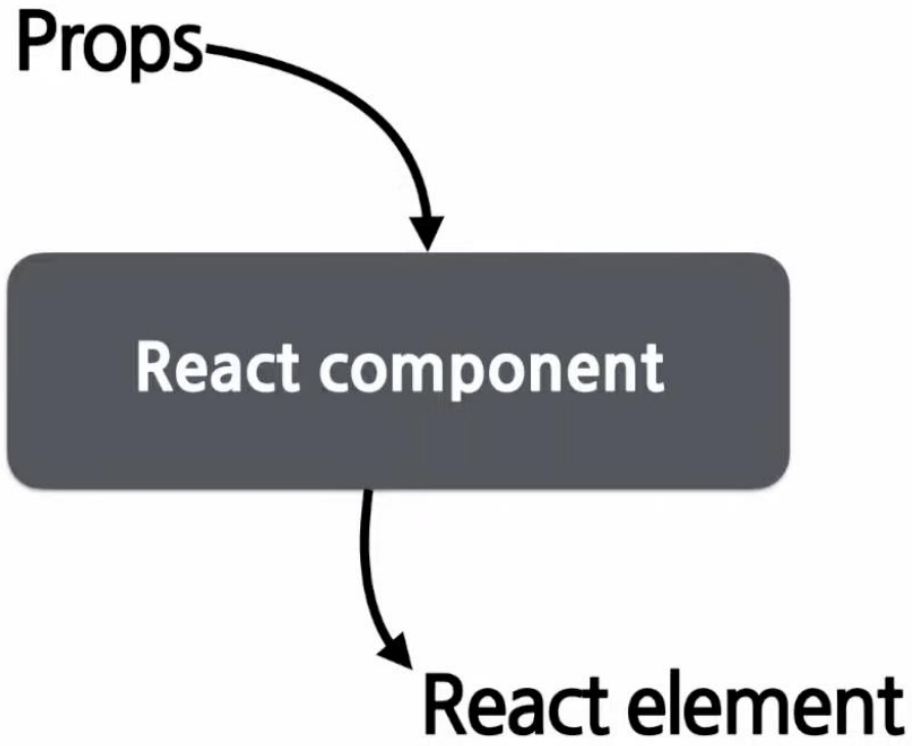
SECTION 5.1 컴포넌트에 대해 알아보기

- 함수와 리액트 컴포넌트



SECTION 5.1 컴포넌트에 대해 알아보기

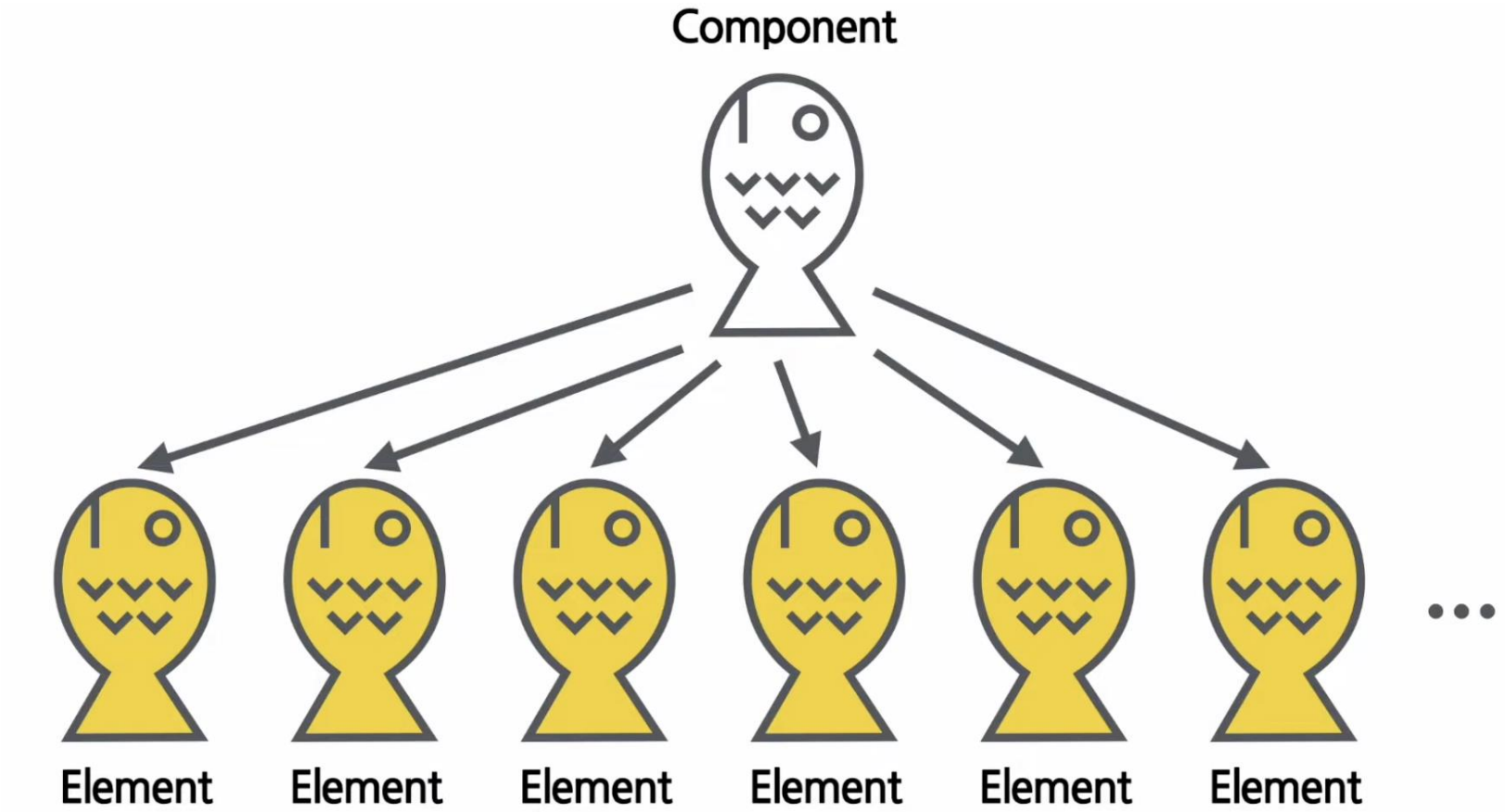
- **리액트 컴포넌트**
  - 어떠한 속성들을 입력으로 받아서 그에 맞는 리액트 엘리먼트를 생성하여 리턴



SECTION 5.1 컴포넌트에 대해 알아보기

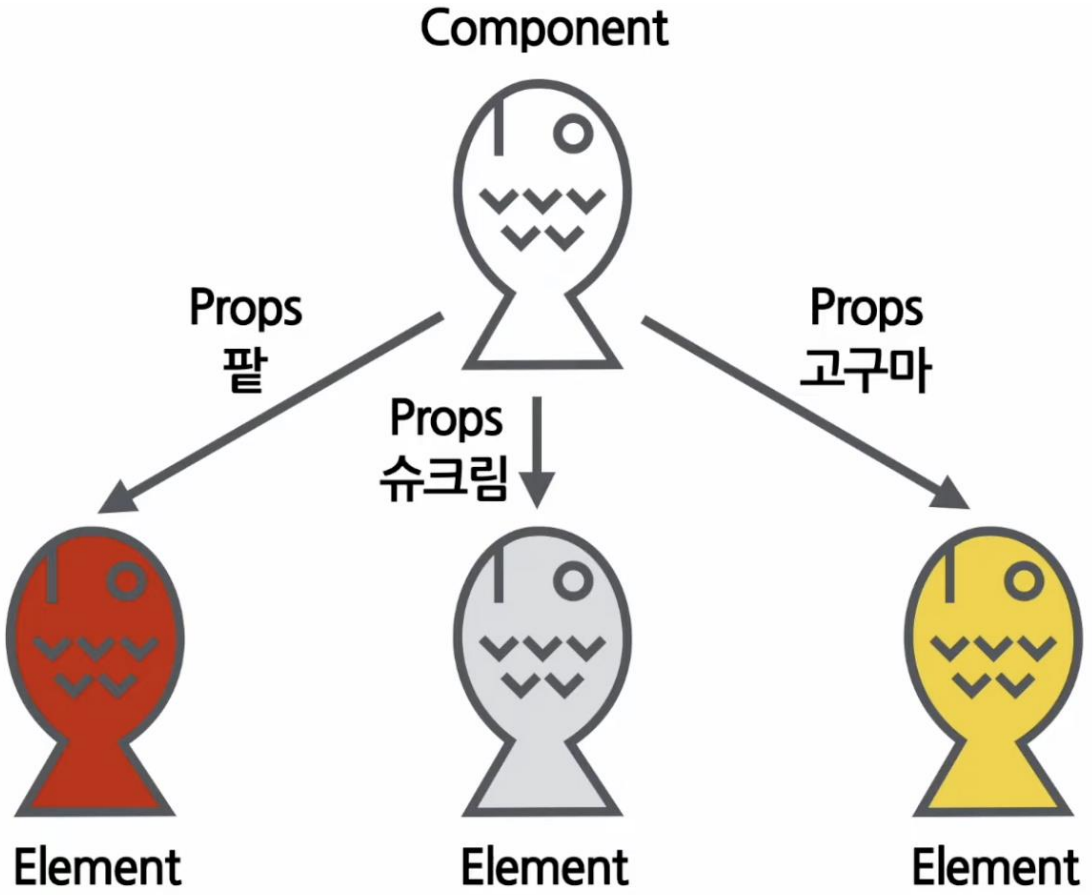
◦ 리액트 컴포넌트

- 객체 지향 개념에서 나오는 클래스와 인스턴스의 개념과 비슷



SECTION 5.2 Props에 대해 알아보기

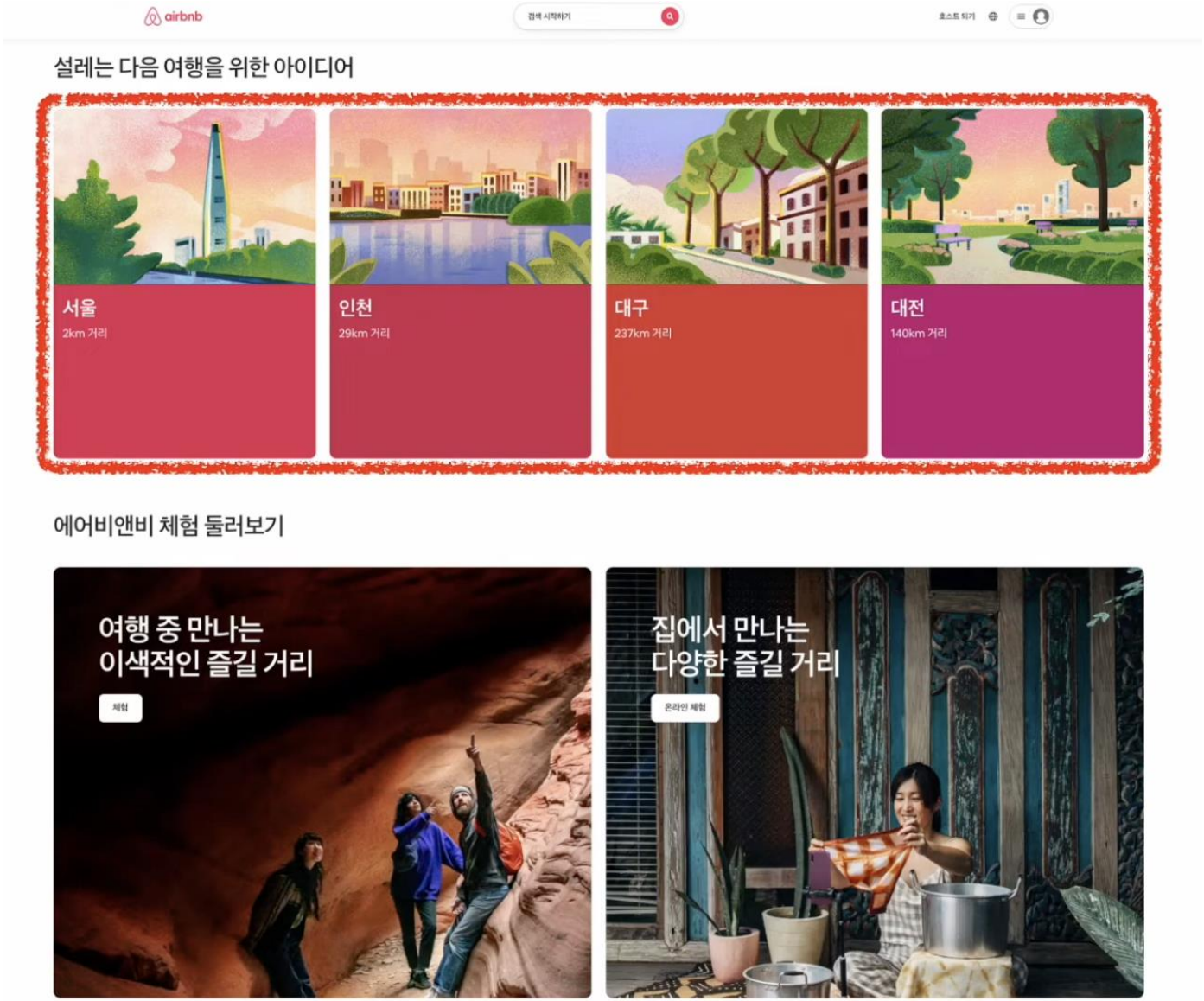
- **Props**
  - Property - 속성
  - **리액트 컴포넌트의 속성**



컴포넌트에서 눈에 보이는 글자나 색깔 등의
속성을 바꾸고 싶을 때 사용하는 컴포넌트의 속 재료

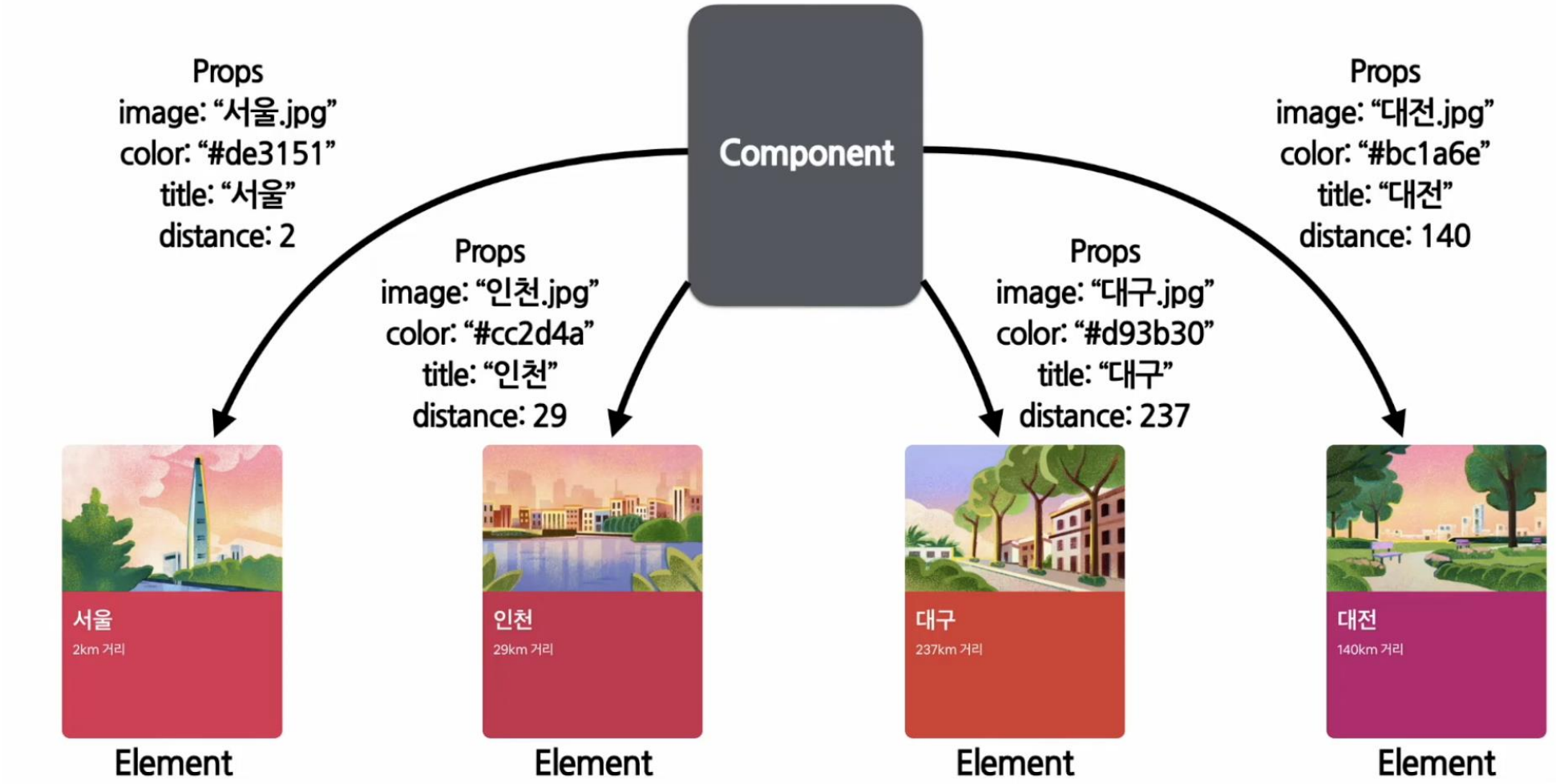
SECTION 5.2 Props에 대해 알아보기

- **Props**



SECTION 5.2 Props에 대해 알아보기

- **Props**



SECTION 5.2 Props에 대해 알아보기

◦ **Props**

- **컴포넌트에 전달할 다양한 정보를 담고 있는 자바스크립트 객체**

- 컴포넌트의 모습과 속성을 결정하는 역할
- 컴포넌트에 어떤 데이터를 전달하여 다른 모습의 엘리먼트를 화면에 렌더링 하고 싶을 때, 해당 데이터를 props에 넣어서 전달

SECTION 5.2 Props에 대해 알아보기

- **Props**의 특징

  - **Read-Only** : 값을 변경할 수 없음

  - props의 값을 변경하고 싶다면?
    - **새로운 값을 컴포넌트에 전달하여 새로 Element를 생성**

SECTION 5.2 Props에 대해 알아보기

◦ 자바스크립트 함수의 속성

```
function sum(a, b) {
    return a + b;
}
```

- **순수 함수(Pure function)**
  - 입력값(input)을 변경하지 않음
  - 같은 입력값에 대해서는 항상 같은 출력값(output)을 리턴

SECTION 5.2 Props에 대해 알아보기

◦ 자바스크립트 함수의 속성

```
function withdraw(accout, amount) {
    account.total -= amount;
}
```

- **비순수 함수(Impure function)**
  - 입력값(input)을 변경
  - 동일한 입력값에 대해서 다른 값을 출력값(output)으로 리턴

SECTION 5.2 Props에 대해 알아보기

◦ 리액트 컴포넌트의 특징

- All React components must act like pure functions with respect to their props.

  - **모든 리액트 컴포넌트는 그들의 props에 관해서는 Pure 함수와 같은 역할을 해야 한다.**

- 모든 리액트 컴포넌트는 props를 직접 바꿀 수 없고, 같은 props에 대해서는 항상 같은 결과를 보여주어야 한다.

SECTION 5.2 Props에 대해 알아보기

◦ 리액트 컴포넌트의 특징

- 리액트 컴포넌트의 props는 바꿀 수 없고,
  같은 props가 들어오면 항상 같은 엘리먼트를 리턴해야 한다.

SECTION 5.2 Props에 대해 알아보기

- **Props 사용법**
  - JSX를 사용하는 경우
    - 키-값 쌍의 형태로 컴포넌트에 props를 전달

```
function App(props) {
    return (
        <Profile
            name="소플"
            introduction="안녕하세요, 소플입니다."
            viewCount={1500}
        />
    );
}
```

** props의 값 - 문자열 이외에 정수, 변수, 컴포넌트는 중괄호로 감싸줌

SECTION 5.2 Props에 대해 알아보기

- **Props 사용법**
  - props의 형태 - 자바스크립트 객체

```
{
    name: "소플",
    introduction: "안녕하세요, 소플입니다.",
    viewCount: 1500
}
```

SECTION 5.2 Props에 대해 알아보기

- **Props 사용법**
  - JSX를 사용하는 경우
    - props의 값으로 컴포넌트를 넣을 수 있음

```
function App(props) {
    return (
        <Layout
            width={2560}
            height={1440}
            header={
                <Header title="소플의 블로그입니다." />
            }
            footer={
                <Footer />
            }
        />
    );
}
```

SECTION 5.2 Props에 대해 알아보기

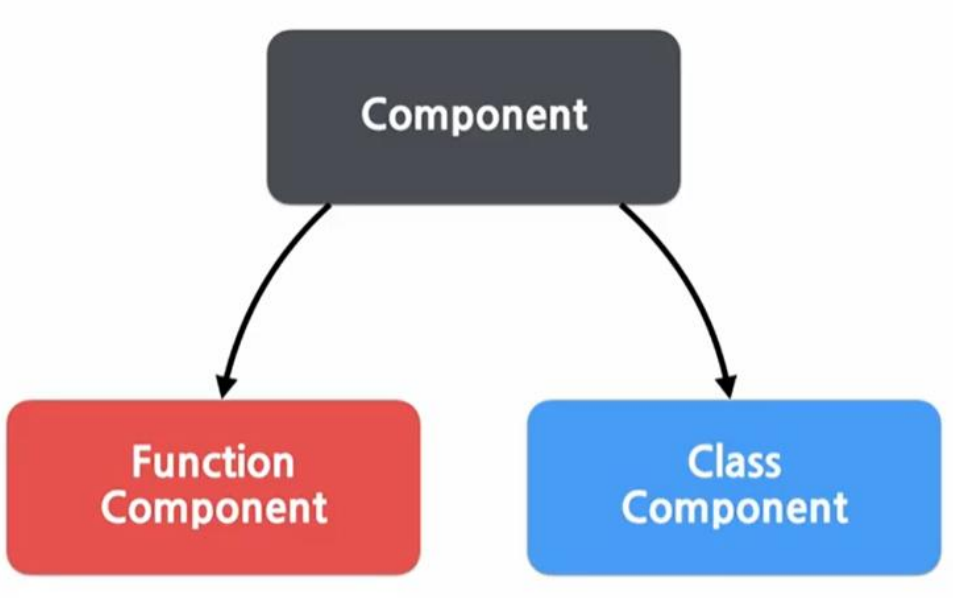
- **Props 사용법**
  - JSX를 사용하지 않는 경우
    - createElement( ) 함수

```
React.createElement(
    type,
    [props],
    [...children]
)
```

```
React.createElement(
    Profile,
    {
        name: "소플",
        introduction: "안녕하세요, 소플입니다.",
        viewCount: 1500
    },
    null
);
```

SECTION 5.3 컴포넌트 만들기

◦ **컴포넌트의 종류**

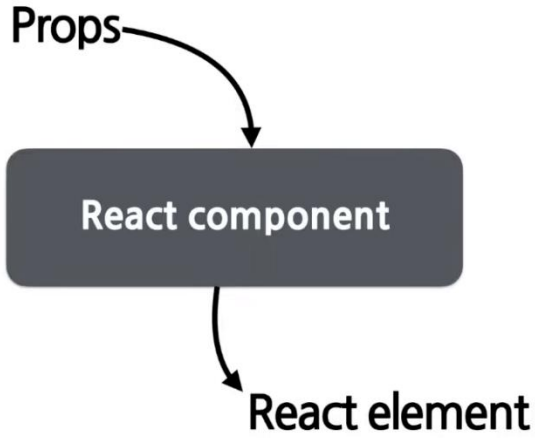


- 초기 버전에서는 클래스 컴포넌트를 주로 사용
- 이후 함수 컴포넌트를 개선해서 주로 사용
- 함수 컴포넌트를 개선하는 과정에서 개발된 것이 훅(hook)

SECTION 5.3 컴포넌트 만들기

- **함수 컴포넌트(Function Component)**
  - "컴포넌트는 Pure 함수와 같은 역할"
    - 리액트 컴포넌트를 일종의 함수로 취급

```
function Welcome(props) {
    return <h1>안녕, {props.name}</h1>;
}
```



간단한 코드가 장점

SECTION 5.3 컴포넌트 만들기

- **클래스 컴포넌트(Class Component)**
  - 자바스크립트 ES6의 클래스를 사용해서 만들어진 컴포넌트

```
class Welcome extends React.Component {
    render() {
        return <h1>안녕, {this.props.name}</h1>;
    }
}
```

  - 클래스 컴포넌트는 React.Component를 상속받아서 만듦

SECTION 5.3 컴포넌트 만들기

◦ 컴포넌트 이름 짓기

- 컴포넌트의 이름은 항상 대문자로 시작해야 함
  - 소문자로 시작하는 컴포넌트는 DOM 태그로 인식. ex) div, span

▪ HTML div 태그로 인식

```
const element = <div />;
```

▪ Welcome이라는 리액트 Component로 인식

```
const element = <Welcome name="인덕" />
```

SECTION 5.3 컴포넌트 만들기

- 컴포넌트 렌더링
  - 사용자가 정의한 Component를 사용한 element

```
const element = <Welcome name="인덕" />
```

  - 사용자 정의 컴포넌트를 렌더링하는 코드

```
function Welcome(props) {
    return <h1>안녕, {props.name}</h1>;
}

const element = <Welcome name="인덕" />

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

Welcome 컴포넌트에
{ name: “인덕”} props를 넣어서 호출

리액트 엘리먼트 생성

React DOM을 통해 실제 DOM 업데이트

SECTION 5.4 컴포넌트 합성

- **컴포넌트 합성**
  - 여러 개의 컴포넌트를 합쳐서 하나의 컴포넌트를 만드는 것

  - 컴포넌트 안에 또 다른 컴포넌트를 사용할 수 있음
  - 복잡한 화면을 여러 개의 컴포넌트로 나눠서 구현할 수 있음

SECTION 5.4 컴포넌트 합성

- 컴포넌트 합성
  - 예제

여러 개의 컴포넌트를 합쳐서 또 다른 컴포넌트를 만드는 것

```
function Welcome(props) {
    return <h1>안녕, {props.name}</h1>;
}

function App(props) {
    return (
        <div>
            <Welcome name="Mike" />
            <Welcome name="Steve" />
            <Welcome name="Jane" />
        </div>
    )
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <App />
);
```

SECTION 5.4 컴포넌트 합성

- 리액트 앱의 기본적인 구조

** 기존 웹페이지에 리액트를 연동하면 Root Node가 여러 개일 수 있음

SECTION 5.5 컴포넌트 추출

- 컴포넌트 추출(Extracting)
  - 큰 컴포넌트에서 일부를 추출해서 새로운 컴포넌트를 만드는 것

- 컴포넌트 추출을 하는 이유?
  - 컴포넌트가 작아질수록 해당 컴포넌트의 기능과 목적이 명확해짐
  - 또한, Props도 단순해짐

재사용성이 높아짐

SECTION 5.5 컴포넌트 추출

- 컴포넌트 추출 과정 1 – Avatar 컴포넌트 추출

```jsx
function Comment(props) {
    return (
        <div className="comment">
            <div className="user-info">
                <img className="avatar"
                    src={props.author.avatarUrl}
                    alt={props.author.name}
                />
                <div className="user-info-name">
                    {props.author.name}
                </div>
            </div>

            <div className="comment-text">
                {props.text}
            </div>

            <div className="comment-date">
                {formatDate(props.date)}
            </div>
        </div>
    );
}
```

```jsx
function Avatar(props) {
    return (
        <img className="avatar"
            src={props.user.avatarUrl}
            alt={props.user.name}
        />
    );
}
```

props.author -> props.user

: 재사용성을 위해 보편적인 단어로 변경

SECTION 5.5 컴포넌트 추출

- 컴포넌트 추출 과정 2 – UserInfo 컴포넌트 추출

```jsx
function Comment(props) {
    return (
        <div className="comment">
            <div className="user-info">
                <Avatar user={props.author} />
                <div className="user-info-name">
                    {props.author.name}
                </div>
            </div>

            <div className="comment-text">
                {props.text}
            </div>

            <div className="comment-date">
                {formatDate(props.date)}
            </div>
        </div>
    );
}
```

```jsx
function UserInfo(props) {
    return (
        <div className="user-info">
            <Avatar user={props.user} />
            <div className="user-info-name">
                {props.user.name}
            </div>
        </div>
    );
}
```

SECTION 5.5 컴포넌트 추출

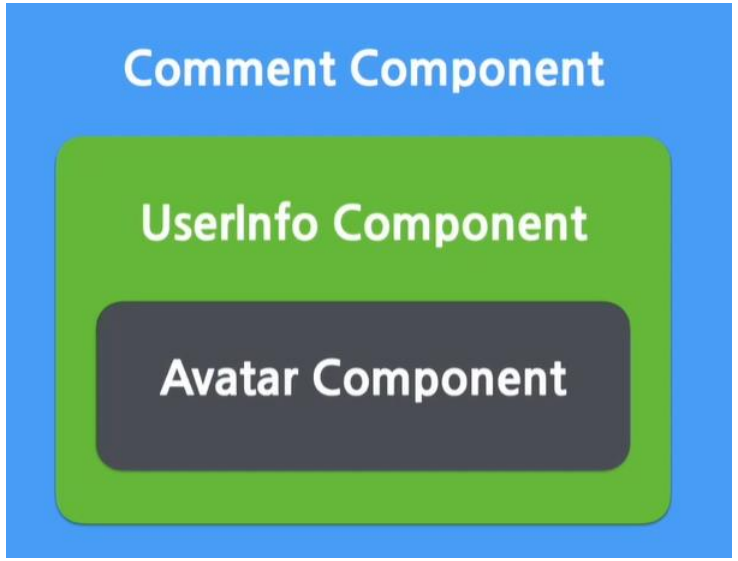
- 컴포넌트 추출 과정 3 – UserInfo 컴포넌트 적용

```
function Comment(props) {
    return (
        <div className="comment">
            <UserInfo user={props.author} />
            <div className="comment-text">
                {props.text}
            </div>
            <div className="comment-date">
                {formatDate(props.date)}
            </div>
        </div>
    );
}
```

```
function Comment(props) {
    return (
        <div className="comment">
            <div className="user-info">
                <img className="avatar"
                    src={props.author.avatarUrl}
                    alt={props.author.name}
                />
                <div className="user-info-name">
                    {props.author.name}
                </div>
            </div>

            <div className="comment-text">
                {props.text}
            </div>

            <div className="comment-date">
                {formatDate(props.date)}
            </div>
        </div>
    );
}
```

SECTION 5.5 컴포넌트 추출

- **추출된 컴포넌트 구조**



- **컴포넌트 추출 방법**
  - 기능 단위로 구분
  - 재사용이 가능한 형태로 추출

- **재사용 가능한 컴포넌트가 많을수록 개발속도가 빨라짐**

PRACTICE 5.6 댓글 컴포넌트 만들기

- VS Code - 05 폴더 생성
- Comment.jsx – 댓글 컴포넌트

```jsx
import React from "react";

function Comment(props) {
    return (
        <div>
            <h1>제가 만든 첫 컴포넌트입니다.</h1>
        </div>
    );
}

export default Comment;
```

PRACTICE 5.6 댓글 컴포넌트 만들기

- CommentList.jsx – 댓글 목록 컴포넌트

```jsx
import React from "react";
import Comment from "./Comment";

function CommentList(props) {
    return (
        <div>
            <Comment />
        </div>
    );
}

export default CommentList;
```

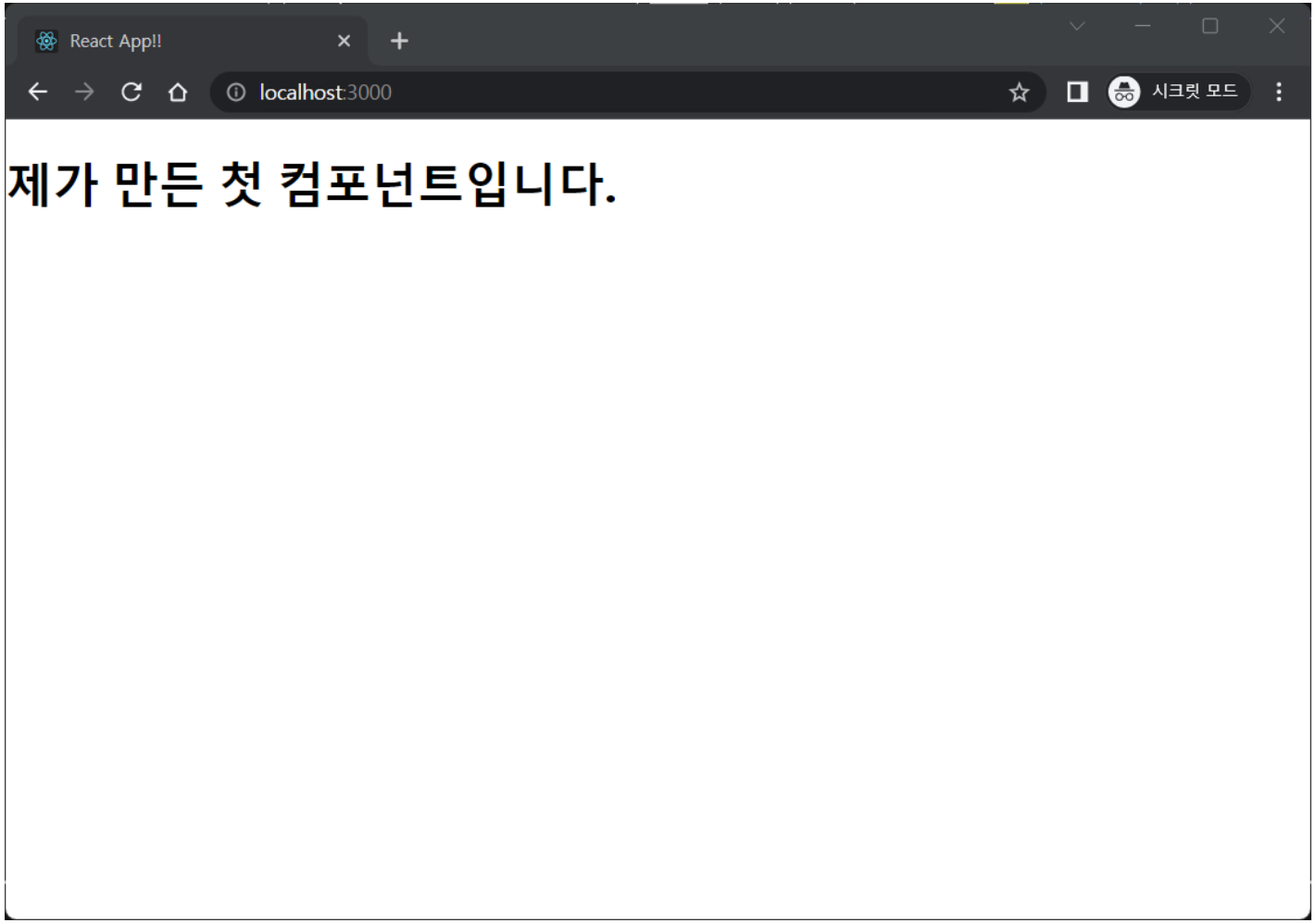
PRACTICE 5.6 댓글 컴포넌트 만들기

- **index.js**

```
import React, { StrictMode } from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
// import App from './App';
import reportWebVitals from './reportWebVitals';

import CommentList from './05/CommentList';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <StrictMode>
        <CommentList />
    </StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

PRACTICE 5.6 댓글 컴포넌트 만들기

- 실행 결과



PRACTICE 5.6 댓글 컴포넌트 만들기

- Comment 컴포넌트에 스타일 입히기
- Comment.jsx

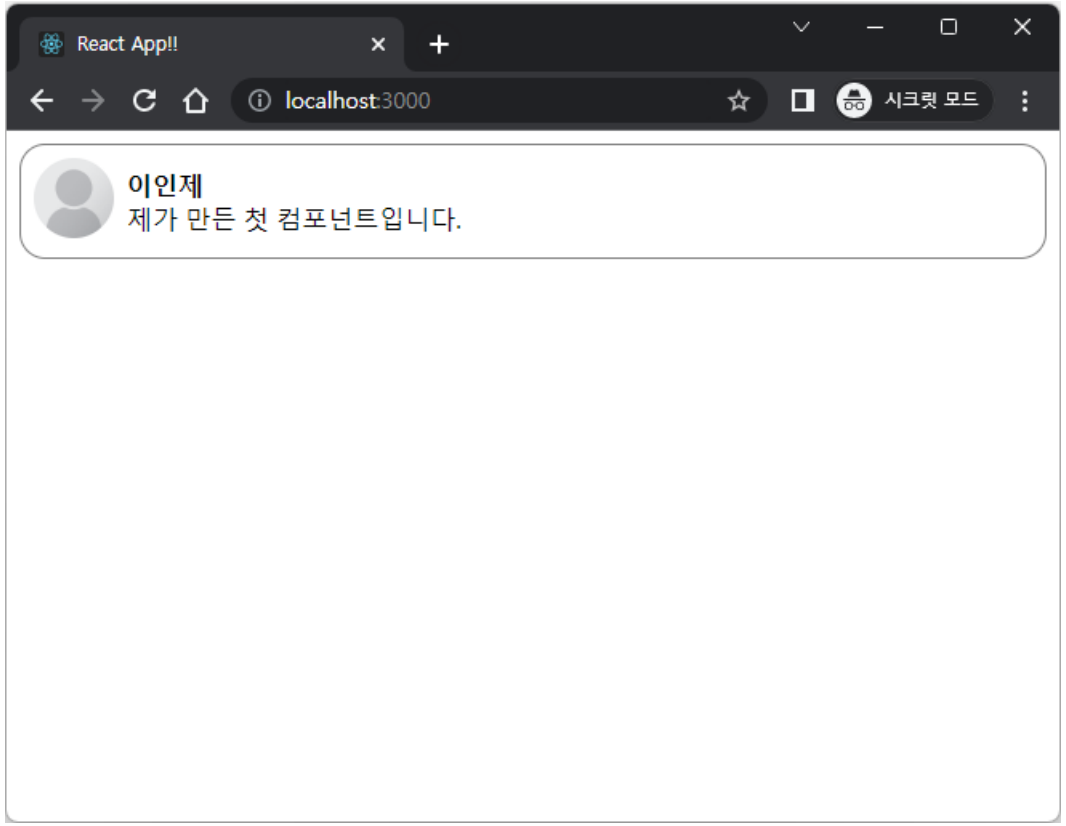
```jsx
function Comment(props) {
    return (
        <div style={styles.wrapper}>
            <div style={styles.imageContainer}>
                <img
                    src="https://upload.wikimedia.org/wikipedia/commons/8/89/Portrait_Placeholder.png"
                    style={styles.image}
                    alt="사용자 이미지"
                />
            </div>
            <div style={styles.contentContainer}>
                <span style={styles.nameText}>이인제</span>
                <span style={styles.commentText}>
                    제가 만든 첫 컴포넌트입니다.
                </span>
            </div>
        </div>
    );
}
```

```jsx
const styles = {
    wrapper: {
        margin: 8,
        padding: 8,
        display: "flex",
        flexDirection: "row",
        border: "1px solid grey",
        borderRadius: 16,
    },
    imageContainer: {},
    image: {
        width: 50,
        height: 50,
        borderRadius: 25,
    },
    contentContainer: {
        marginLeft: 8,
        display: "flex",
        flexDirection: "column",
        justifyContent: "center",
    },
    nameText: {
        color: "black",
        fontSize: 16,
        fontWeight: "bold",
    },
    commentText: {
        color: "black",
        fontSize: 16,
    },
};
```

Image url - https://upload.wikimedia.org/wikipedia/commons/8/89/Portrait_Placeholder.png

PRACTICE 5.6 댓글 컴포넌트 만들기

- 실행 결과



PRACTICE 5.6 댓글 컴포넌트 만들기

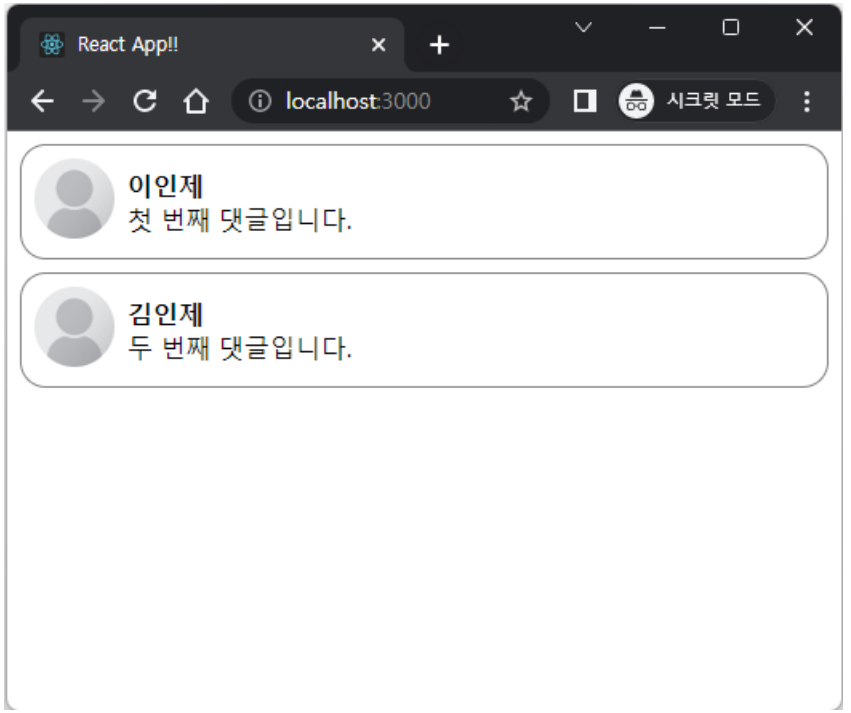
- Comment 컴포넌트에 props 추가
- Comment.jsx

```jsx
function Comment(props) {
    return (
        <div style={styles.wrapper}>
            <div style={styles.imageContainer}>
                <img
                    src="https://upload.wikimedia.org/wikipedia/commons/8/89/Portrait_Placeholder.png"
                    style={styles.image}
                    alt="사용자 이미지"
                />
            </div>
            <div style={styles.contentContainer}>
                <span style={styles.nameText}>{props.name}</span>
                <span style={styles.commentText}>
                    {props.comment}
                </span>
            </div>
        </div>
    );
}
```

PRACTICE 5.6 댓글 컴포넌트 만들기

- CommentList 컴포넌트에 속성 추가
- CommentList.jsx

```jsx
import React from "react";
import Comment from "./Comment";

function CommentList(props) {
    return (
        <div>
            <Comment name="이인제" comment="첫 번째 댓글입니다." />
            <Comment name="김인제" comment="두 번째 댓글입니다." />
        </div>
    );
}

export default CommentList;
```

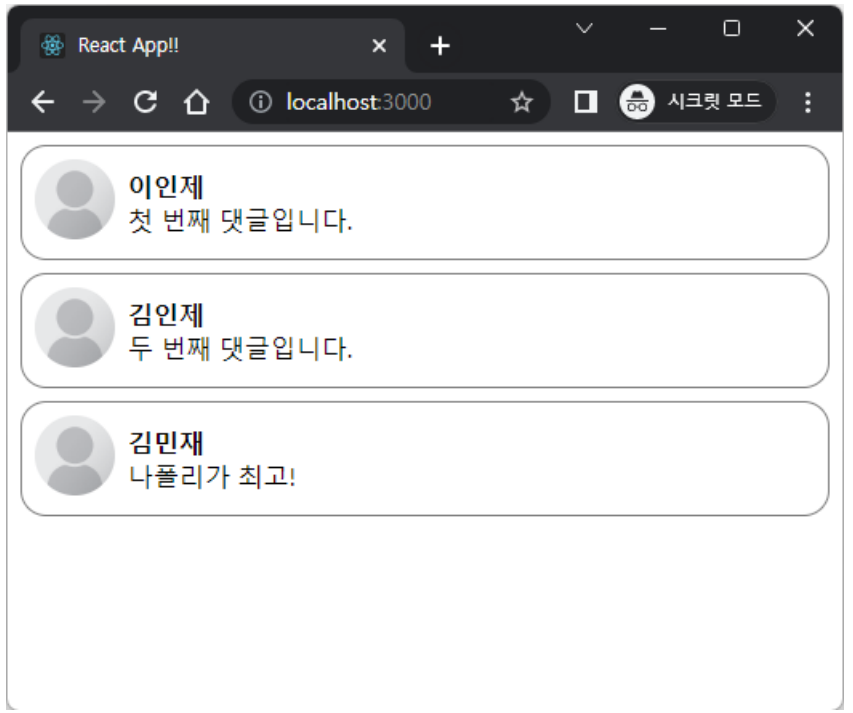


실행결과

PRACTICE 5.6 댓글 컴포넌트 만들기

- 댓글 데이터를 별도의 객체로 분리
- CommentList.jsx

```
const comments = [
    {
        name: "이인제",
        comment: "첫 번째 댓글입니다.",
    },
    {
        name: "김인제",
        comment: "두 번째 댓글입니다.",
    },
    {
        name: "김민재",
        comment: "나폴리가 최고!"
    }
];

function CommentList(props) {
    return (
        <div>
            {comments.map((comment) => {
                return (
                    <Comment name={comment.name} comment={comment.comment} />
                );
            })}
        </div>
    );
}
```



실행결과

[요약]

- 리액트 컴포넌트
  - 컴포넌트 기반 구조
    - 작은 컴포넌트들이 모여서 하나의 컴포넌트를 구성하고 이러한 컴포넌트들이 모여서 전체 페이지를 구성
  - 개념적으로 자바스크립트 함수와 비슷함
    - 속성들을 입력으로 받아서 그에 맞는 리액트 엘리먼트를 생성하여 리턴함
- Props
  - Props의 개념
    - 리액트 컴포넌트의 속성
    - 컴퍼넌트에 전달할 다양한 정보를 담고 있는 자바스크립트 객체
  - Props의 특징
    - 읽기 전용
    - 리액트 컴포넌트의 props는 바꿀 수 없고, 같은 props가 들어오면 항상 같은 엘리먼트를 리턴해야 함

[요약]

- Props 사용법
  - JSX를 사용할 경우 컴포넌트에 키-값 쌍 형태로 넣어주면 됨
  - 문자열 이외에 정수, 변수, 그리고 다른 컴포넌트들이 들어갈 경우에는 중괄호를 사용해서 감싸주어야 함
  - JSX를 사용하지 않는 경우에는 createElement() 함수의 두 번째 파라미터로 자바스크립트 객체를 넣어주면 됨

- 컴포넌트 만들기
  - 컴포넌트의 종류
    - 클래스 컴포넌트와 함수 컴포넌트로 나뉨
  - 함수 컴포넌트
    - 함수 형태로 된 컴포넌트
  - 클래스 컴포넌트
    - ES6의 클래스를 사용하여 만들어진 컴포넌트

[요약]

- 컴포넌트의 이름 짓기
  - 컴포넌트의 이름은 항상 대문자로 시작해야 함
  - 소문자로 시작할 경우 컴포넌트를 DOM 태그로 인식하기 때문
- 컴포넌트 렌더링
  - 컴포넌트로부터 엘리먼트를 생성하여 이를 리액트 DOM에 전달

- 컴포넌트 합성
  - 여러 개의 컴포넌트를합쳐서 하나의 컴포넌트를 만드는 것

- 컴포넌트 추출
  - 큰 컴포넌트에서 일부를 추출해서 새로운 컴포넌트를 만드는 것
  - 기능 단위로 구분하는 것이 좋고, 나중에 곧바로 재사용이 가능한 형태로 추출