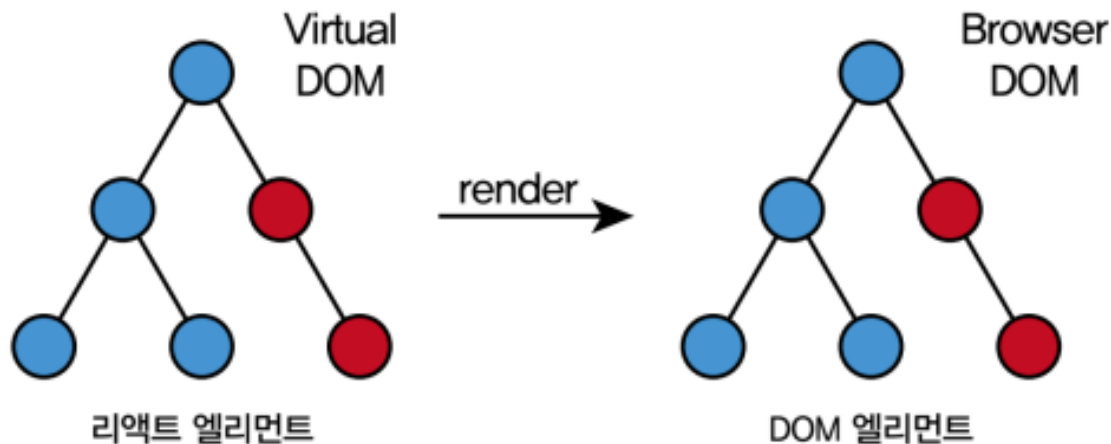


엘리먼트 렌더링

엘리먼트 정의

- Elements are the smallest building blocks of React apps.
 - 출처: 리액트 공식 홈페이지
- Element: DOM 엘리먼트(HTML 엘리먼트)
 - 크롬 브라우저 개발자 도구>'엘리먼트 탭' 에서 확인
- 리액트 엘리먼트=가상돔(Virtual DOM)
 - 리액트 엘리먼트는 DOM 엘리먼트의 가상 표현
 - 화면에서 보이는 것을 기술



엘리먼트 렌더링

```
<div id="root"></div>
```

```
const element = <h1>안녕, 리액트</h1>;
```

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

- 렌더링 과정

1. ReactDOM의 createRoot() 함수로 root DOM 노드 생성
2. element를 render() 함수의 인자로 전달하여 root DOM 노드에 렌더링

*** 리액트 엘리먼트가 렌더링되는 과정은 Virtual DOM에서 실제 DOM으로 이동하는 과정이다. ***

엘리먼트 구조

- 리액트 엘리먼트는 객체로 존재함
 - 엘리먼트를 객체로 만드는 명령

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

- 리액트 엘리먼트의 유형이 HTML 태그가 아닌 컴포넌트(자바스크립트 객체)의 경우에는 type에 리액트 컴포넌트 이름이 들어감

엘리먼트 구조

- 리액트 엘리먼트 구조 예

```
<button class="bg-green">  
  <b>Hello, element!</b>  
</button>
```

```
01  {  
02    type: 'button',  
03    props: {  
04      className: 'bg-green',  
05      children: {  
06        type: 'b',  
07        props: {  
08          children: 'Hello, element!'  
09        }  
10      }  
11    }  
12  }
```

createElement() 함수 동작 과정 확인

```
1 import React from "react";
2
3 function Button(props) {
4   return (
5     <div>
6       <button className={`bg-${props.color}`}>
7         <b>{props.children}</b>
8       </button>
9     </div>
10  );
11 }
12
13 export default Button;
```

Button.jsx

내용을 확인하셨으면 확인 버튼을 눌러주세요

확인

ConfirmDialog.jsx

```
1 import React from "react";
2 import Button from "./Button";
3
4 function ConfirmDialog() {
5   return (
6     <div>
7       <p>내용을 확인하셨으면 확인 버튼을 눌러주세요</p>
8       <Button color="green">확인</Button>
9     </div>
10  );
11 }
12
13 export default ConfirmDialog;
```

createElement() 함수 동작 과정 확인

index.js

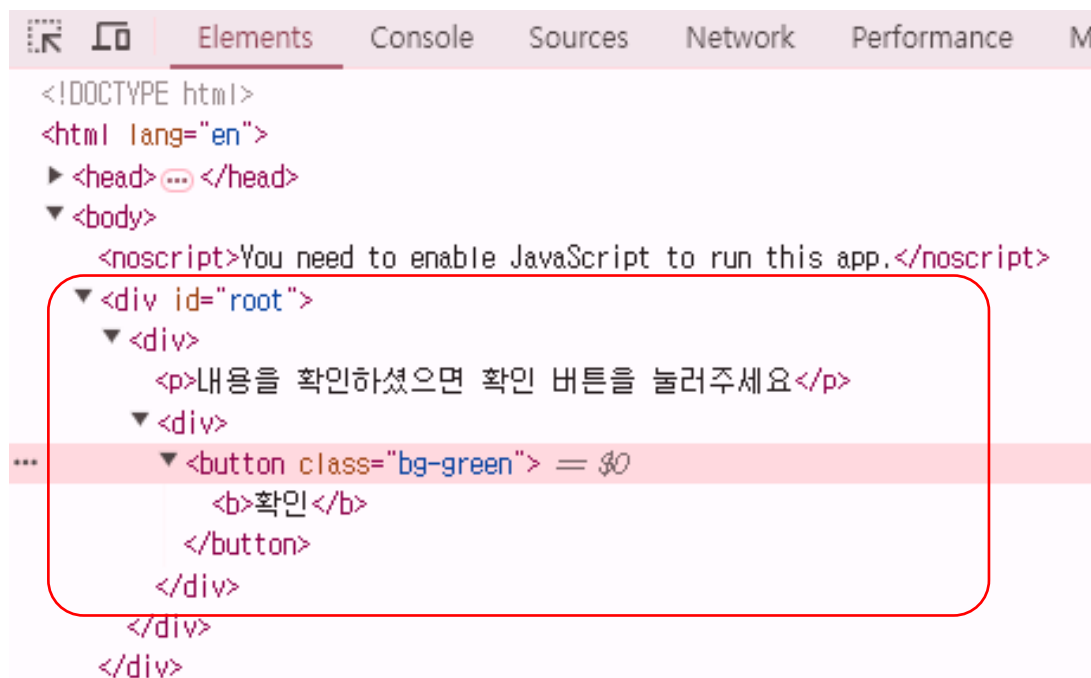
```
import ConfirmDialog from "../components/ConfirmDialog";

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(
  <React.StrictMode>
    <ConfirmDialog />
  </React.StrictMode>
);
```

createElement() 함수 동작 과정 확인

개발자도구>엘리먼트->탭 결과확인



```
<!DOCTYPE html>
<html lang="en">
  <head> ... </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root">
      <div>
        <p>내용을 확인하셨으면 확인 버튼을 눌러주세요</p>
        <div>
          <button class="bg-green"> = $0
            <b>확인</b>
          </button>
        </div>
      </div>
    </div>
  </body>
</html>
```

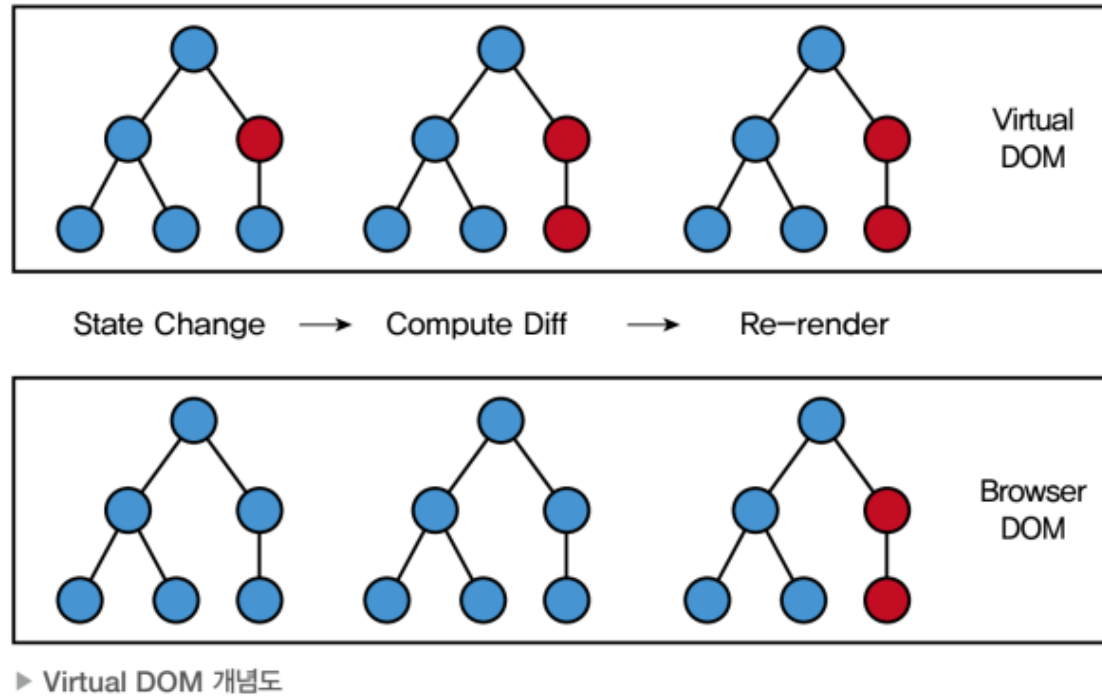
내용을 확인하셨으면 확인 버튼을 눌러주세요

확인

리액트 엘리먼트 특징

- 불변성

- 엘리먼트는 생성된 후에 children이나 attribute을 바꿀 수 없다.
- 화면에 변경된 엘리먼트들을 보여주기 위해서는 새로운 엘리먼트를 만든다.
- Virtual DOM은 변경된 부분을 계산하고 해당 부분만을 다시 렌더링한다.



**** 상태관리와 화면갱신 횟수는 리액트 개발과정에서 성능을 좌우하는 중요한 요소이다. ****

렌더링된 엘리먼트 업데이트하기

안녕, 리액트!

현재 시간: 오후 5:20:40

- 내부적으로 tick() 호출될 때마다 새로운 엘리먼트를 생성해서 화면 렌더링한다.

```
function tick() {  
  const element = (  
    <div>  
      <h1>안녕, 리액트!</h1>  
      <h2>현재 시간: {new Date().toLocaleTimeString()}</h2>  
    </div>  
  );  
  const root = ReactDOM.createRoot(document.getElementById("root"));  
  root.render(element);  
}  
  
setInterval(tick, 1000);
```

엘리먼트 업데이트 실습

- 시계만들기

Date 객체 이용하여 현재 시간 확인
`new Date().toLocaleTimeString()`

안녕, 리액트!

현재 시간: 오후 5:20:40

Clock.jsx

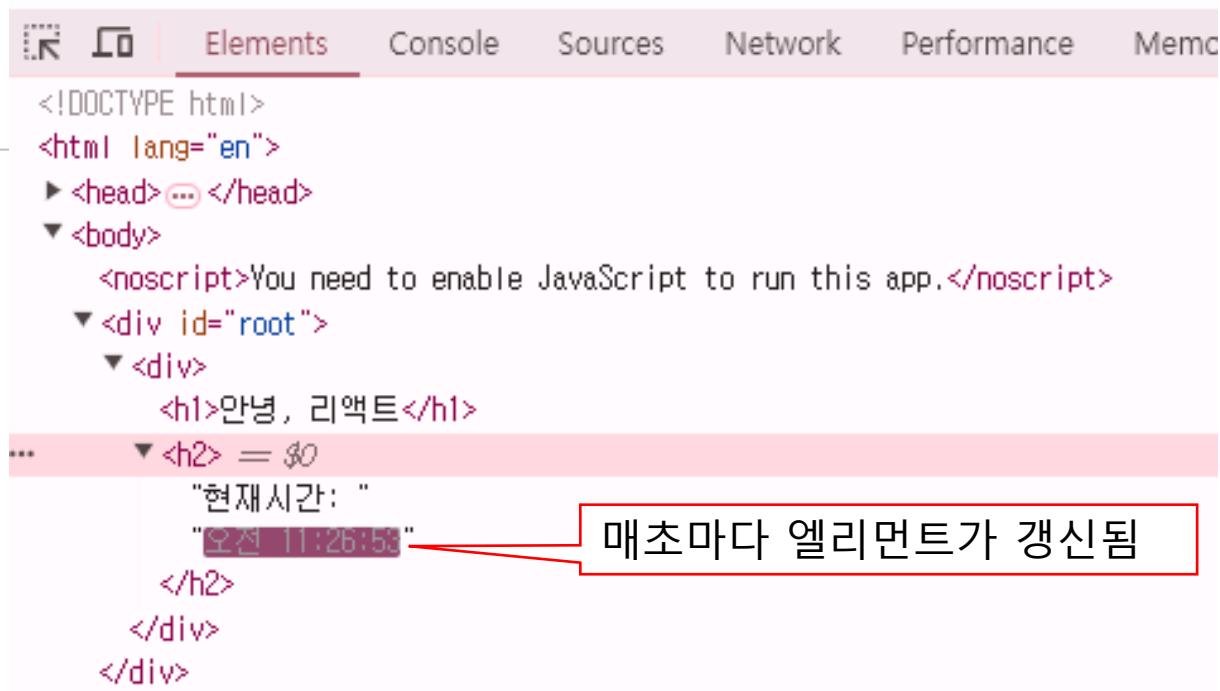
```
function Clock(props) {  
  return (  
    <div>  
      <h1>안녕, 리액트</h1>  
      <h2>현재시간: {new Date().toLocaleTimeString()}</h2>  
    </div>  
  );  
}
```

엘리먼트 업데이트 실습

- 시계만들기

index.js

```
const root = ReactDOM.createRoot(document.getElementById("root"));
setInterval(() => {
  root.render(
    <React.StrictMode>
      <Clock />
    </React.StrictMode>
  );
}, 1000);
```



컴포넌트와 Props

리액트는 컴포넌트 기반 구조이다.

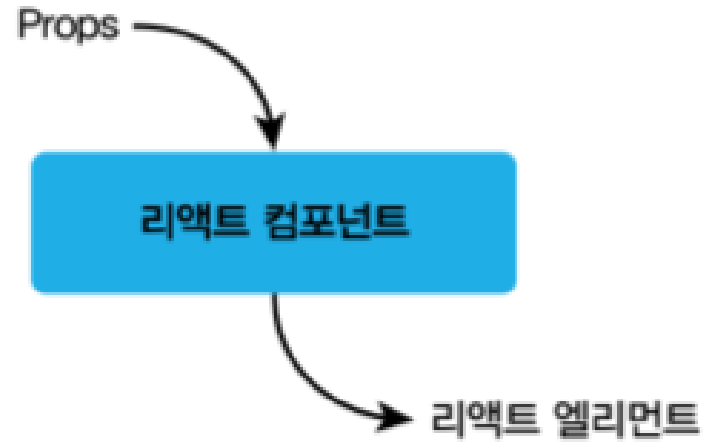
- 함수와 리액트 컴포넌트



- 컴포넌트를 이용하면 개발 시간과 유지보수 비용을 줄일 수 있다.

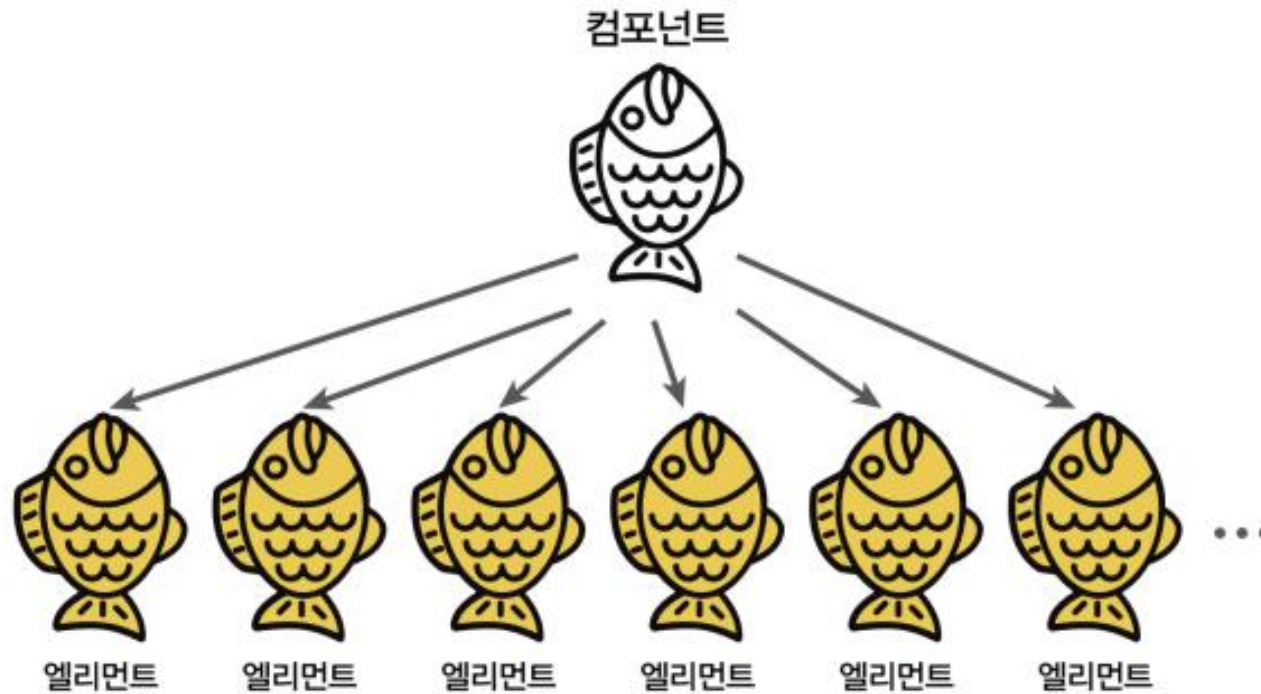
리엑트는 컴포넌트 기반 구조이다.

- 리엑트 컴포넌트 구조



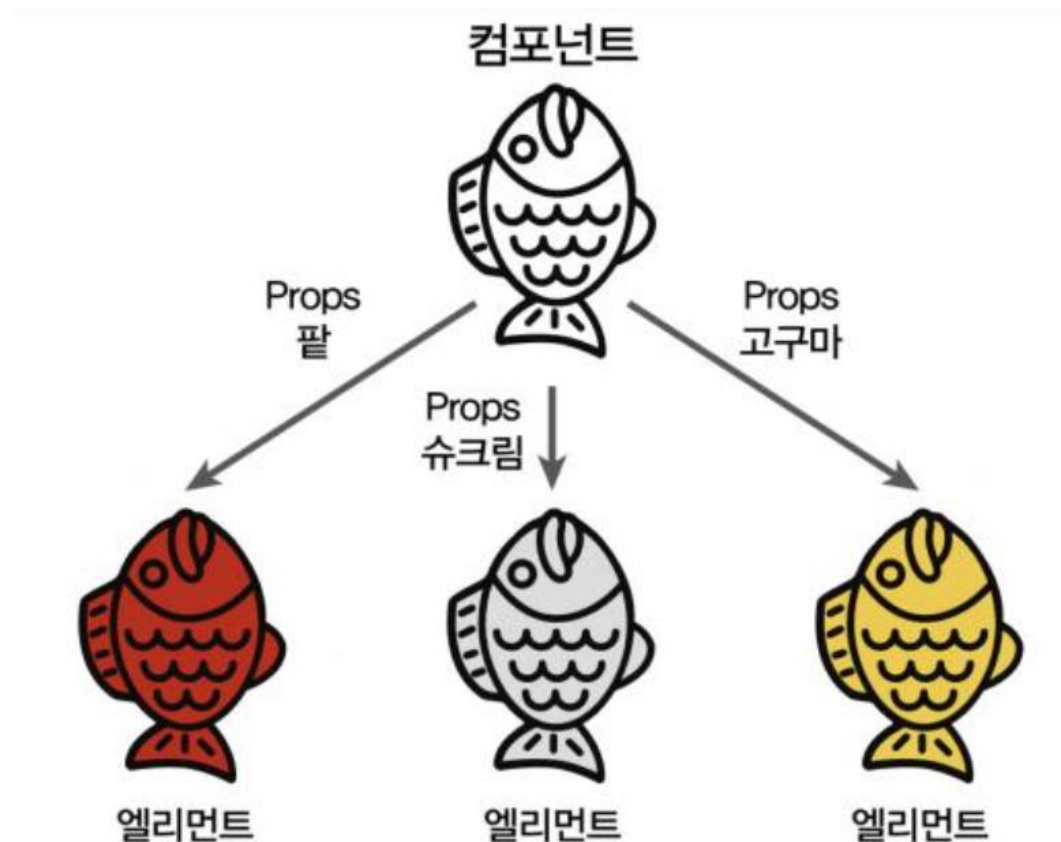
컴포넌트와 엘리먼트

- 리액트 컴포넌트는 객체지향 패라다임의 클래스,
- 리액트 엘리먼트는 인스턴스와 비슷한 개념



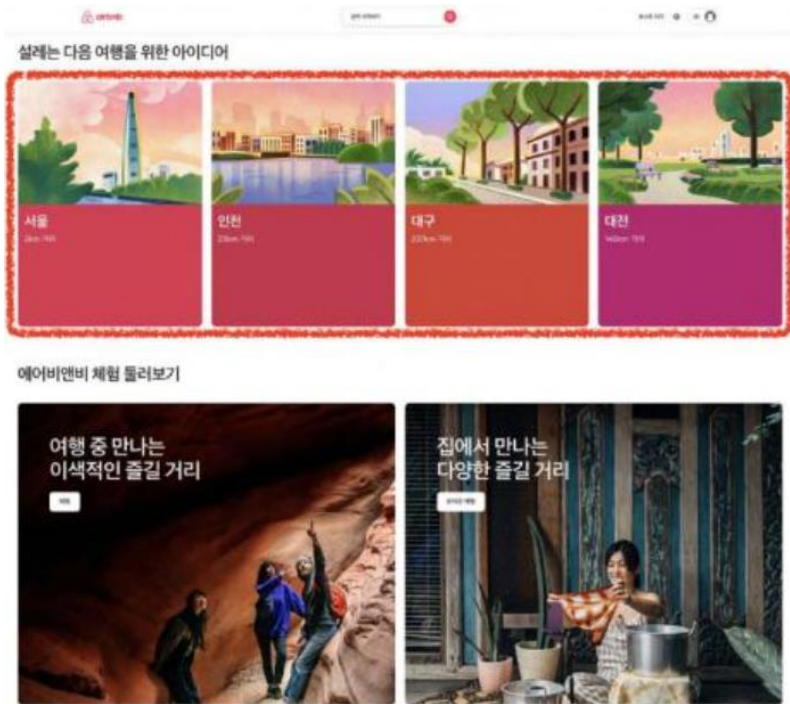
Props 개념

- Property: 컴포넌트 속성
 - props: 엘리먼트에 들어가는 값을 저장하는 장소(객체)
 - 컴포넌트에 전달할 다양한 정보를 담고있는 자바스크립트 객체



Props 개념

- 에어비앤비의 컴포넌트와 props 사용 예



Props 개념

➤ 컴포넌트의 모양과 속성을 결정하는 것은 'props'이다.

컴포넌트에 어떤 데이터를 전달하고 데이터에 따라 다른 모습의 엘리먼트를 화면에 렌더링 하고 싶을 때, 해당 데이터를 props에 넣어 전달한다.

Props 특징

- 읽기 전용이다.
- 리액트 컴포넌트는 props에 대해 pure 함수와 같다.
 - 모든 리액트 컴포넌트는 props를 직접 바꿀 수 없다.
 - 컴포넌트는 같은 props에 대해서는 항상 같은 결과(리액트 엘리먼트)를 리턴한다.

```
01 // pure 함수
02 // input을 변경하지 않으며 같은 input에 대해서 항상 같은 output을 리턴
03 function sum(a, b) {
04     return a + b;
05 }
```

```
01 // impure 함수
02 // input을 변경함
03 function withdraw(account, amount) {
04     account.total -= amount;
05 }
```

Props 사용법

- props 객체에 담기는 내용은 키-값 형식으로 구성됨
예) App 컴포넌트에서 Profile 컴포넌트에 전달되는 속성과 속성값 예
 - 문자열 이외의 값이 props 객체에 저장될 경우 {} 사용한다.

```
01 function App(props) {  
02     return (  
03         <Profile  
04             name="소플"    문자열  
05             introduction="안녕하세요, 소플입니다."  
06             viewCount={1500} 정수  
07         />  
08     );  
09 }
```

props 객체에 저장된 내용

```
01 {  
02     name: "소플",  
03     introduction: "안녕하세요, 소플입니다.",  
04     viewCount: 1500  
05 }
```

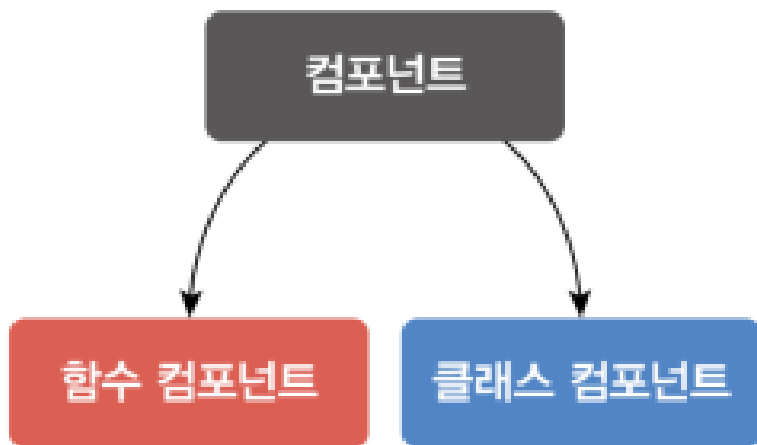
Props 사용법

- props 값으로 컴포넌트가 올 수 있다.

```
01  function App(props) {
02      return (
03          <Layout
04              width={2560}
05              height={1440}
06              header={
07                  <Header title="소플의 블로그입니다." />
08              }
09              footer={
10                  <Footer />
11              }
12          />
13      );
14  }
```

컴포넌트 만들기

➤ 컴포넌트 종류



```
01  function Welcome(props) {  
02      return <h1>안녕, {props.name}</h1>;  
03  }
```

```
01  class Welcome extends React.Component {  
02      render() {  
03          return <h1>안녕, {this.props.name}</h1>;  
04      }  
05  }
```

컴포넌트 만들기

- 컴포넌트 이름은 항상 대문자로 시작한다.
 - DOM 엘리먼트와 구별하기 위함.
 - 리엑트는 소문자는 내장 컴포넌트(<div>,)로 대문자는 리엑트 컴포넌트로 구분한다.

```
01 // HTML div 태그로 인식
02 const element = <div />;
03
04 // Welcome이라는 리엑트 컴포넌트로 인식
05 const element = <Welcome name="인제" />;
```


컴포넌트 렌더링

```
01  function Welcome(props) {  
02      return <h1>안녕, {props.name}</h1>;  
03  }  
04  
05  const element = <Welcome name="인제" />;  
06  const root = ReactDOM.createRoot(document.getElementById('root'));  
07  root.render(element);
```

1. Welcome 컴포넌트 선언
2. Welcome 컴포넌트 호출(props (name="인제"))되고 결과를 element에 저장
3. root Dom Node의 render() 호출되면서 Welcome 컴포넌트가 브라우저에 출력

컴포넌트 활용 예시

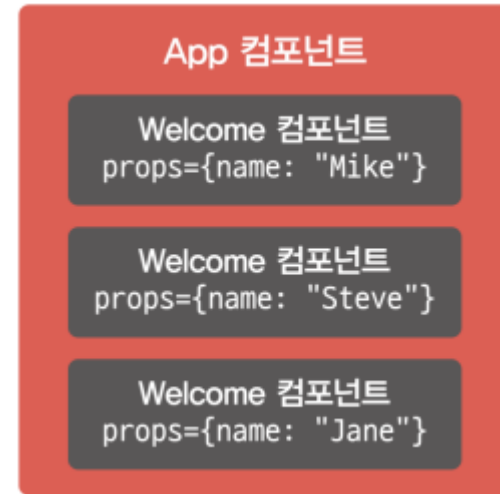
- 리액트에서는 컴포넌트안에 컴포넌트를 사용할 수 있다.
 - 복잡한 화면을 여러 개의 컴포넌트로 나눠서 구현 가능하다

```
01 function Welcome(props) {  
02     return <h1>Hello, {props.name}</h1>;  
03 }  
04  
05 function App(props) {  
06     return (  
07         <div>  
08             <Welcome name="Mike" />  
09             <Welcome name="Steve" />  
10             <Welcome name="Jane" />  
11         </div>  
12     )  
13 }  
14  
15 const root = ReactDOM.createRoot(document.getElementById('root'));  
16 root.render(<App />);
```

Welcome.jsx

App.js

index.js



[컴포넌트 합]성

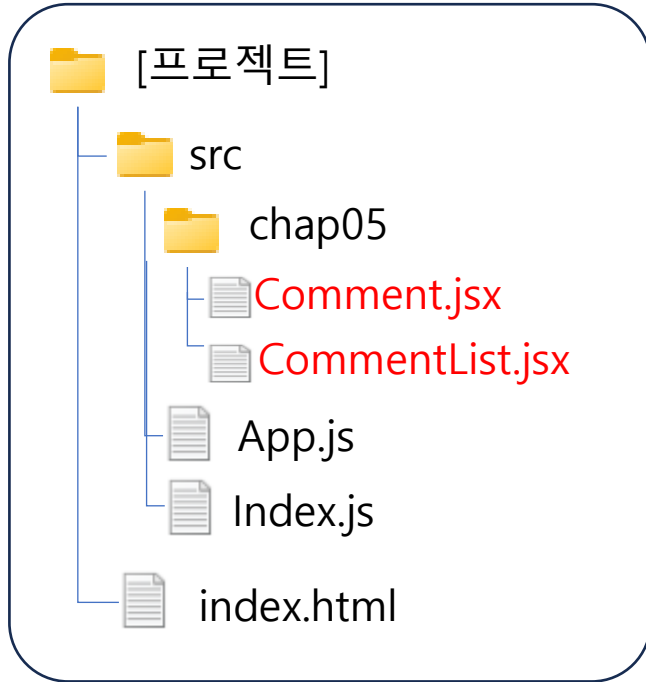
Hello, Mike

Hello, Steve

Hello, Jane

[결과화면]

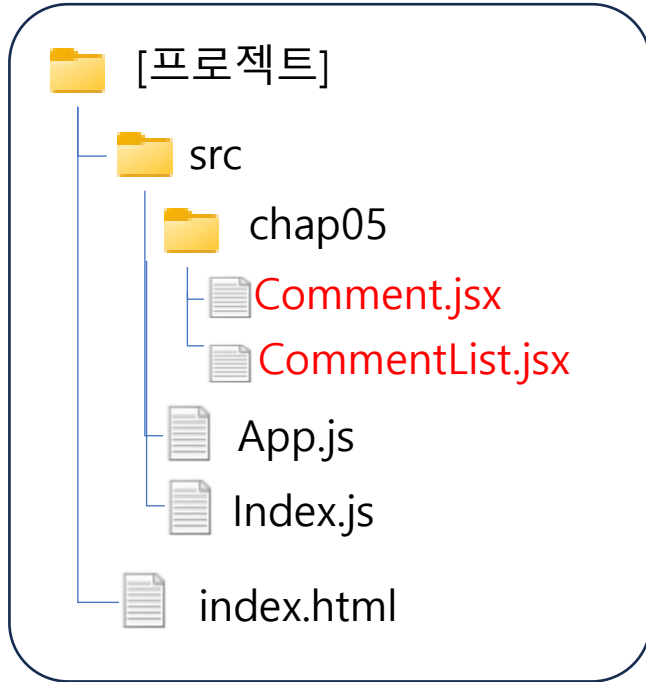
댓글 컴포넌트 만들기 실습



1. Comment 컴포넌트 코드 작성

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

댓글 컴포넌트 만들기 실습



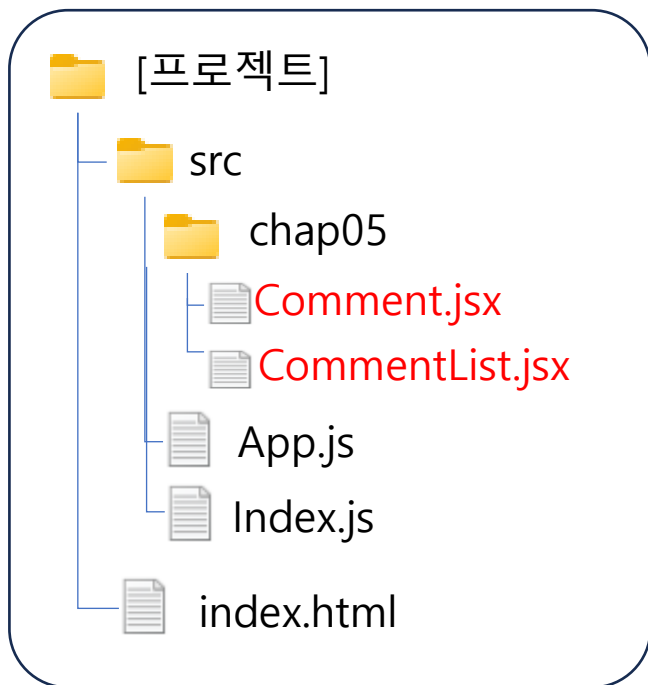
2. CommentList 컴포넌트 작성

```
import Comment from "../Comment";

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

export default CommentList;
```

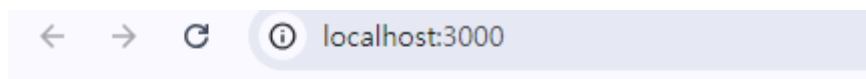
댓글 컴포넌트 만들기 실습



3. Index.js에서 CommentList 컴포넌트 불러오기

```
import CommentList from "../chap05/CommentList";

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



제가 만든 첫 컴포넌트입니다.

댓글 컴포넌트 만들기 실습

4. Comment.jsx 파일에 스타일 코드 추가하기

```
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,  
},  
};
```

댓글 컴포넌트 만들기 실습

4. Comment 컴포넌트에 스타일 입히기

```
<div style={styles.wrapper}>
  <div style={styles.imageContainer}>
    
  </div>
  <div style={styles.contentContainer}>
    <span style={styles.nameText}>홍길동</span>
    <span style={styles.commentText}>제가 만든 첫 컴포넌트입니다.</span>
  </div>
</div>
```



댓글 컴포넌트 만들기 실습

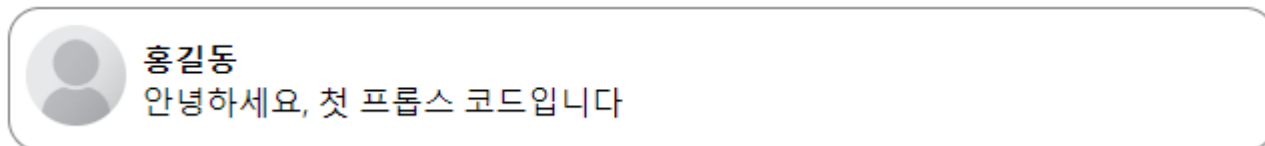
5. Comment 컴포넌트에 작성자 이름과 댓글 내용이 동적으로 변화되도록 props 추가하기

```
<div style={styles.contentContainer}>  
  <span style={styles.nameText}>{props.name}</span>  
  <span style={styles.commentText}>{props.comment}</span>  
</div>
```



6. CommentList 컴포넌트 수정

```
<div>  
  <Comment name={"홍길동"} comment={"안녕하세요, 첫 프롭스 코드입니다"} />  
</div>
```



댓글 컴포넌트 만들기 실습

7. CommentList 컴포넌트에 Comment 컴포넌트 추가(댓글 추가)하기

```
<div>  
  <Comment name={"홍길동"} comment={"안녕하세요, 첫 프롭스 코드입니다"} />  
  <Comment name={"황진이"} comment={"리액트 재미있어요."} />  
</div>
```



홍길동

안녕하세요, 첫 프롭스 코드입니다



황진이

리액트 재미있어요.

8. CommentList 컴포넌트에 Comment 컴포넌트 추가(본인 이름으로 댓글 추가)하기

정리

- **리액트 컴포넌트**

- 컴포넌트 기반 구조

- : 작은 컴포넌트들이 모여서 하나의 컴포넌트를 구성하고 이러한 컴포넌트들이 모여서 전체 페이지를 구성

- 개념적으로는 자바스크립트의 함수와 비슷함

- : 속성들을 입력으로 받아서 그에 맞는 리액트 엘리먼트를 생성하여 리턴함

- **Props**

- Props의 개념

- : 리액트 컴포넌트의 속성

- : 컴포넌트에 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

- Props 사용법

- : JSX를 사용할 경우 컴포넌트에 키-값 쌍 형태로 넣어 주면 됨

- : 문자열 이외에 정수, 변수, 그리고 다른 컴포넌트 등이 들어갈 경우에는 중괄호를 사용해서 감싸주어야 함

- : JSX를 사용하지 않는 경우에는 createElement() 함수의 두 번째 파라미터로 자바스크립트 객체를 넣어 주면 됨

정리

- **컴포넌트 만들기**

- 컴포넌트의 종류
 - : 클래스 컴포넌트와 함수 컴포넌트로 나눔
- 함수 컴포넌트
 - : 함수 형태로 된 컴포넌트
- 클래스 컴포넌트
 - : ES6의 클래스를 사용하여 만들어진 컴포넌트
- 컴포넌트 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 컴포넌트를 합쳐서 하나의 컴포넌트를 만드는 것

- **컴포넌트 추출**

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