

# React 구조

# 주요특징

- 리액트 소개, 시작하기
- 엘리먼트 렌더링
- 컴포넌트와 Props
- State와 생명주기
- 훅
- 이벤트 핸들링
- 조건부 렌더링
- 리스트와 키
- 폼

# React 개요 및 환경 구축

## ■ 자바스크립트 프레임워크 인기 순위

프레임워크	NPM 주간 다운로드
리액트	1,293만 6,453
뷰 / 뷰3	251만 3,494
앵귤러(CLI)	174만 9,234
프리액트	120만 1,728
릿	86만 5,826
스벨트	50만 2,888
알파인JS	19만 2,740
솔리드JS	11만 5,257
HTMZ	2만238
쿼	7,527

표 1. 가장 인기 있는 프론트엔드 자바스크립트 프레임워크(다운로드 횟수 기준)

# React 개요 및 환경 구축

- 계속해서 상승하는 React 인기
  - 페이스북, 넷플릭스, 인스타그램, 에어앤비 등 리액트로 만들어짐
  - 데이터통신은 모두 비동기로 이루어짐
  - 빠르게 (부드러운) 화면 갱신
  - 많은 코드가 하나의 컴포넌트(예, 상단 메뉴바)를 기존 기법으로 구현할 때, 많은 코드 필요.
  - 리액트는 복잡한 코드를 단순히 묶어서 사용할 수 있다.
  - **사용자 인터페이스를 만들기 위한 자바스크립트 라이브러리,**
  - =사용자 정의 태그를 만들기 위한 자바스크립트 라이브러리
  - =사용자 컴포넌트를 만들기 위한 자바스크립트 라이브러리

# React 개요 및 환경 구축

## ■ Node.js 설치

- 크로스플랫폼 오픈소스 자바스크립트 런타임 환경으로 윈도우, 리눅스, macOS 등을 지원
- 주로 확장성 있는 네트워크 애플리케이션과 서버 사이드 개발에 사용되는 소프트웨어 플랫폼
- 내장 HTTP 서버 라이브러리를 포함하고 있어 웹 서버에서 아파치 등의 별도의 소프트웨어 없이 동작하는 것이 가능



# 리액트 첫 프로젝트 생성

1. 프로젝트를 위한 작업공간(workspace) 만들기 (\*\*대문자 있으면 안됨\*\*)

2. VSCODE에서 폴더 열기(open folder)

3. workspace에 폴더 생성(project01)

4. New terminal에서 project01 폴더로 이동



```
PS D:\dev\react\react01> npm start
```

5. 터미널에서 node --version : > react01@0.1.0 start

것. 삭제 후 재설치)

6. 터미널에서 "npx create-react-app" : > react-scripts start

react 프로젝트 설치하겠음'

Starting the development server...

```
PS D:\dev\react\react01> npx create-react-app .
```

8. Happy hacking! -> 단어 출력, react 프로젝트 성공적으로 설치됨

Happy hacking!

- (워크스페이스에 파일들 생성됨)

```
PS D:\dev\react\react01>
```

9. 리액트 실행시키는 명령: "npm start" -> project01에 있는 프로젝트 실행됨.

# 리액트 첫 프로젝트 생성

9. 리액트 실행시키는 명령: "**npm start**" -> project01에 있는 프로젝트 실행됨.

```
PS D:\dev\react\react01> npm start
```

```
> react01@0.1.0 start
```

```
> react-scripts start
```

```
Starting the development server...
```

```
You can now view react01 in the browser.
```

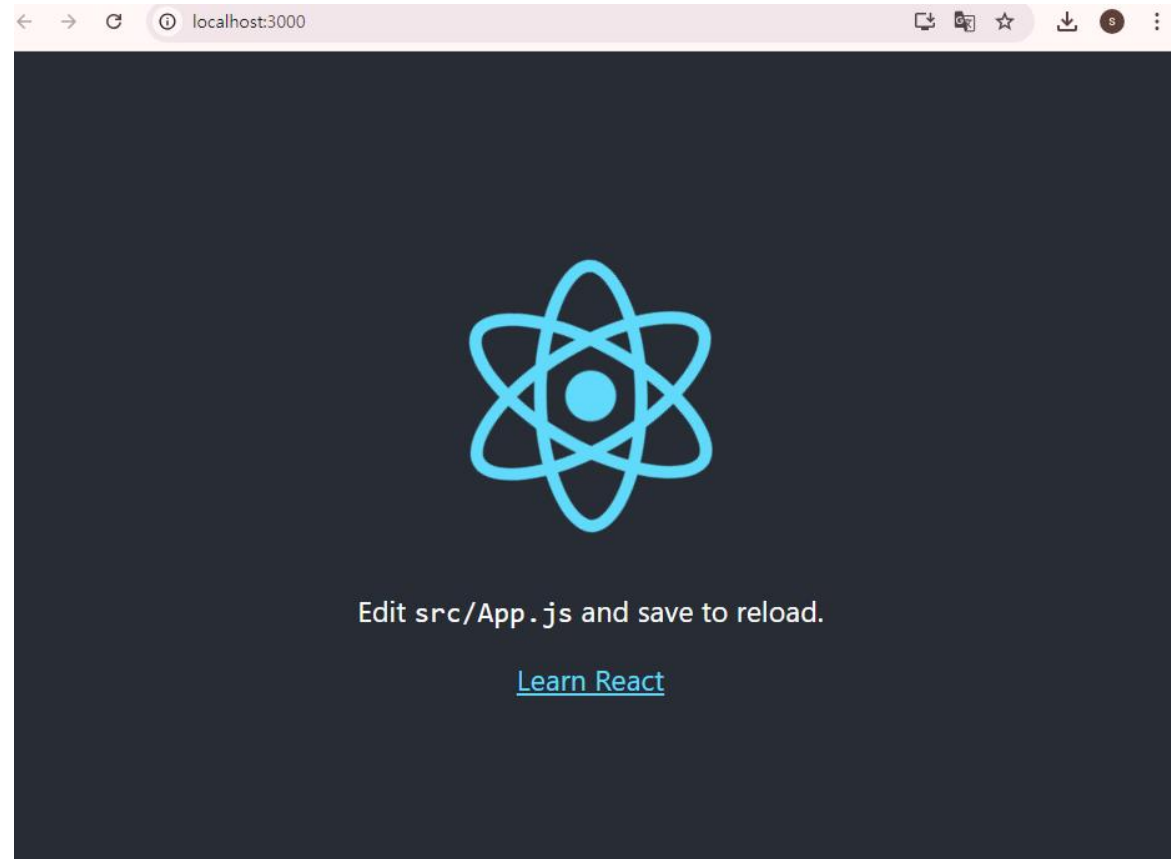
```
Local: http://localhost:3000
```

```
On Your Network: http://172.30.1.6:3000
```

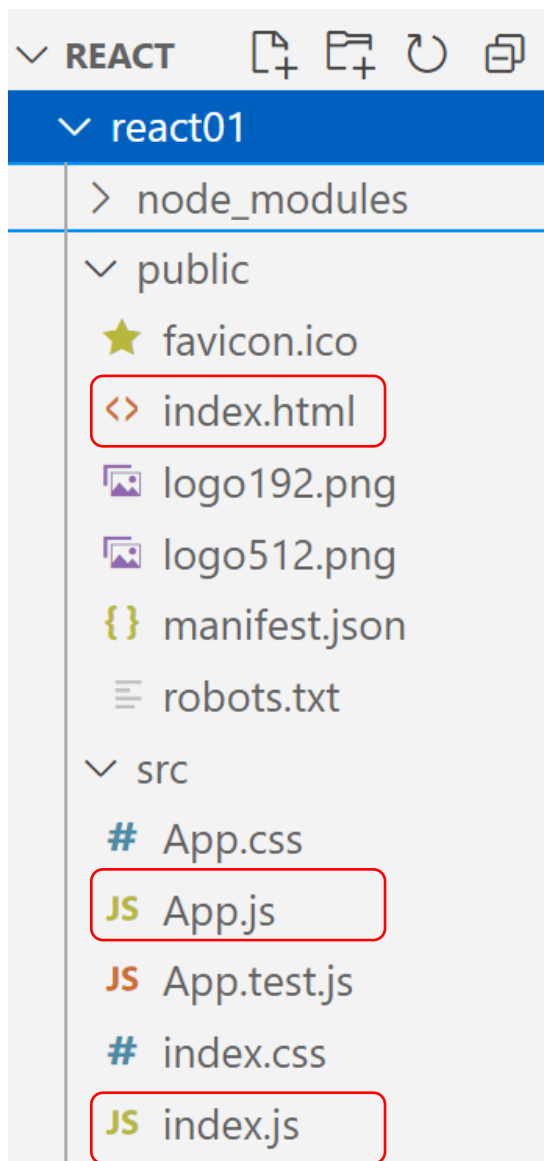
```
Note that the development build is not optimized.  
To create a production build, use npm run build.
```

```
webpack compiled successfully
```

```
█
```



# React 프로젝트 폴더 구조



## ➤ 리액트 기본 프로젝트의 화면 출력 과정 이해

html      Single 웹페이지

index.js    index.html 의 id = 'root' 요소를 가져와서 엘리먼트(App.js)를 렌더링함

App.js    화면 요소(엘리먼트) 구성





# "Hello, React!"를 화면에 띄우기

**Hello, React!!**

**Welcome, React!!**

```
function App() {  
  return (  
    <div>  
      <h1>Hello, React!!</h1>  
      <h1>Welcome, React!!</h1>  
    </div>  
  );  
}
```

# "Hello, React!"를 화면에 띄우기

**Hello, React!!**

**Welcome, React!!**

[naver](#)

```
function App() {  
  return (  
    <div>  
      <h1>Hello, React!!</h1>  
      <h1>Welcome, React!!</h1>  
      코드 추가  
    </div>  
  );  
}
```

JSX

# JSX

- SPA

서버로부터 새로운 페이지를 다시 불러오지 않고, 현재의 페이지를 동적으로 다시 작성함으로써 유저와 소통하는 웹사이트

➔ 새 페이지를 넘어갈 때마다 정보를 가져오지 않아도 된다. --=> 새로 고침이 필요하지 않다.

- JSX

자바스크립트에 XML 을 추가한 확장 문법 하나의 파일에 자바스크립트와 HTML을 동시에 작성할 수 있다. 자바스크립트와 HTML 파일에 분리감이 없기 때문에 가독성 높고, 개발자 입장에서는 작성하기 쉽다.

1. 여러 요소가 있다면 반드시 **부모 요소**로 감싸야 한다.  
아래 예시처럼 하나의 div 요소로 묶어서 리턴해야 한다.

```
function App() {  
  return(  
    <div>  
    </div>  
  )  
}
```

```
return (  
  <h1>hello world!</h1>  
  <h2>hello world!</h2>  
)
```

Error!

```
return (  
  <div>  
    <h1>hello world!</h1>  
    <h2>hello world!</h2>  
  </div>  
)
```

## 2. 표현식 사용 가능

자바스크립트와 html 같이 사용하므로 자바스크립트의 변수, 함수 사용하게 된다. 이때 중괄호로 자바스크립트 사용한다는 표시함 조건문, 반복문은 사용불가.

```
function App() {  
  let name='홍길동'  
  return (  
    <div>  
      <h1>{name}님의 웹페이지 입니다. </h1>  
      <h2>hello world!</h2>  
    </div>  
  )  
}
```

```
function App() {  
  function userName(name){  
    return name;  
  }  
  return (  
    <div>  
      <h1>{username('홍길동')}님의 웹페이지 입니다. </h1>  
      <h2>hello world!</h2>  
    </div>  
  )  
}
```

### 3. 리액트에서는 class 대신 className을 사용.

```
function App() {  
  function userName(name){  
    return name;  
  }  
  return (  
    <div className="App">  
      <h1>{username('홍길동')}님의 웹페이지 입니다. </h1>  
      <h2>hello world!</h2>  
    </div>  
  )  
}
```

```
src > # App.css > .App  
1  ∨ .App {  
2    text-align: center;  
3  }
```

홍길동님의 웹페이지입니다.

Hello, World!

4. 모든 태그는 여는 태그와 닫는 태그가 반드시 존재한다.

```
function App() {  
  let name='홍길동'  
  return (  
    <div>  
      <input>{name}  
    </div>  
  )  
}
```

Error!

```
function App() {  
  let name='홍길동'  
  return (  
    <div>  
      <input>{name}</input>  
    </div>  
  )  
}
```

# JSX 실습

1. 주어진 코드를 참조하여 아래의 결과가 출력되도록 코드 완성하시오.

**Hello, React!!**

**Welcome, React!!**

[naver](#)

```
function App() {  
  const name = "React";  
  const naver = {  
    name: "naver",  
    url: "http://www.naver.com",  
  };  
  return (  
  
  );  
}
```



# JSX 실습

2. 주어진 코드를 활용하여 아래의 결과가 출력되도록 App.js 코드를 수정하십시오.

**Hello, Paring Yang**

```
function App() {  
  function formatName(user) {  
    return user.firstName + user.lastName;  
  }  
  const user = {  
    firstName: "Paring",  
    lastName: "Yang",  
  };  
  return (  
  
  );  
}
```

# JSX 실습

3. 주어진 코드를 이용하여 아래의 결과가 나오도록 App.js 코드 수정하십시오.

**Hello, ParingYang**

**Hello, Stranger!**

```
function App() {  
  function formatName(user) {  
    return user.firstName + user.lastName;  
  }  
  function getGreeting(user) {  
    if (user) return <h1>Hello,  
    {formatName(user)}</h1>;  
    return <h1>Hello, Stranger!</h1>;  
  }  
  const user = {  
    firstName: "Paring",  
    lastName: "Yang",  
  }  
  return (  
      
  );  
}
```

## ■ 삼항연산자 문법 예시

- name 변수에 입력되는 값에 따라 홍길동님과 함께했던 어려운 수업, 코딩쌤과 함께하는 즐거운 수업을 표현

```
function App() {  
  let name = "홍길동"  
  return (  
    <div>  
      {name === "코딩쌤"? (<h1>{name}님과 함께하는 즐거운 리액트 수업!!</h1>) :  
        (<h1>{name}님과 함께했던 어려운 리액트 수업!!</h1>)}  
    </div>  
  )  
}
```

실행됨

## ■ 삼항연산자 문법 예시

- **name** 변수에 입력되는 값에 따라 홍길동님과 함께했던 어려운 수업, 코딩쌤과 함께하는 즐거운 수업을 표현

```
function App() {
  let name = "코딩쌤"
  return (
    <div>
      {name === "코딩쌤"? (<h1>{name}님과 함께하는 즐거운 리액트 수업!!</h1>
                           <h2>너무 재밌어요!!</h2>):
        (<h1>{name}님과 함께했던 어려운 리액트 수업!!</h1>)}
    </div>
  )
}
```

오류



<h1>과 <h2>를 하나의 부모태그로 묶어라.

## ■ 삼항연산자 문법 예시

- **name** 변수에 입력되는 값에 따라 홍길동님과 함께했던 어려운 수업, 코딩쌤과 함께하는 즐거운 수업을 표현

```
function App() {
  let name = "코딩쌤"
  return (
    <div>
      {name === "코딩쌤"? (
        <div>
          <h1>{name}님과 함께하는 즐거운 리액트 수업!!</h1>
          <h2>너무 재밌어요!!</h2> </div>)
        :
        (<h1>{name}님과 함께했던 어려운 리액트 수업!!</h1>)}
      </div>
    )
  }
}
```

실행됨

# JSX- 스타일취급방법

## ■ 스타일 취급방법

- JSX Style 적용할 때에는 **객체 형태로 삽입**한다.
- 예시) H1 태그에 대하여 배경은 black, 글자색은 red로 스타일 적용한다고 할 때,

`<h1 style="background-color: black; color: red;"> //오류,,`

⇒ `<h1 style={{background-color: "yellow", color: "red"}}> // 스타일은 자바스크립트 {} 객체 {}로 들어간다`  
즉, `style = { }`, 객체 속성은 캐멀기법으로 지정하고, 속성값은 `""`으로 감싼다.

⇒ JSX의 인라인 형식의 스타일은 자바스크립트 객체이므로, 스크립트 변수로 지정할 수 있다.

```
let title1 = {  
  background-color: "yellow",  
  color: "red",  
}
```

⇒ `<h1 style={title1}>`



# JSX- 스타일취급방법

## ■ 스타일 취급방법

- JSX Style 적용할 때에는 **객체 형태로 삽입**한다.
- 예시) H2 태그에 대하여 배경은 black, 글자색은 red로 스타일 적용한다고 할 때,

⇒ 외부 스타일 문서로 지정할 수 있다. App.css 문서에 스타일 정의 예

```
src > # App.css >  .App-logo
```

```
1  .App {  
2    |  text-align: center;  
3  }  
4  .Title1 {  
5    |  background-color:  black;  
6    |  color:  yellow;  
7  }
```

App.css 문서에 코드 추가

**코딩샘님과 함께 하는 리엑트는 흥미롭습니다.**

환영합니다.!!!

=> <h1 className="Title1">

# JSX- 스타일 취급 실습

1. 다음과 같은 결과가 출력되도록 주어진 코드에 인라인 스타일을 추가하시오.

안녕하세요

잘 지내시죠?

버튼



안녕하세요

잘 지내시죠?

버튼

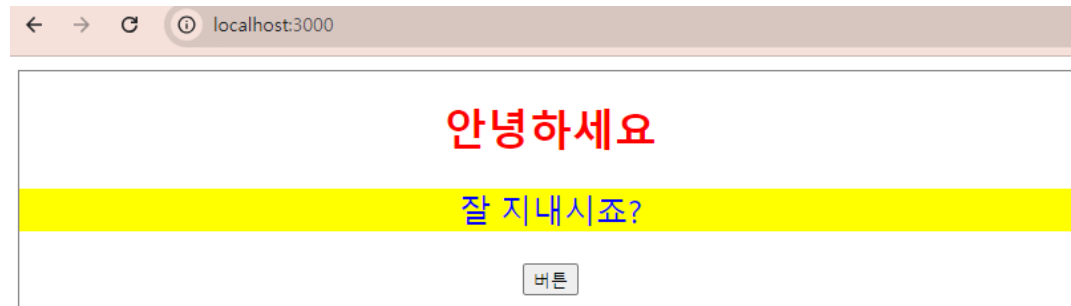
2. 1번의 인라인 스타일 객체를 삭제하고 변수로 지정하여 다시 작성하시오.

“잘 지내시죠?” 문장의 스타일에 배경색을 노란색, 글자크기 24px 로 추가하기

자바스크립트 객체 속성명에는 -(하이픈)을 허용하지 않으므로 css속성명을 모두 캐멀케이스로 작성한다.

3. 2번의 스타일 객체를 주석 달고 외부 App.css 파일에 스타일로 작성하고 코드를 수정하시오.

.App 클래스 스타일 박스 지정: 가로 800px, 세로 180px, 테두리색상 gray, margin 10px

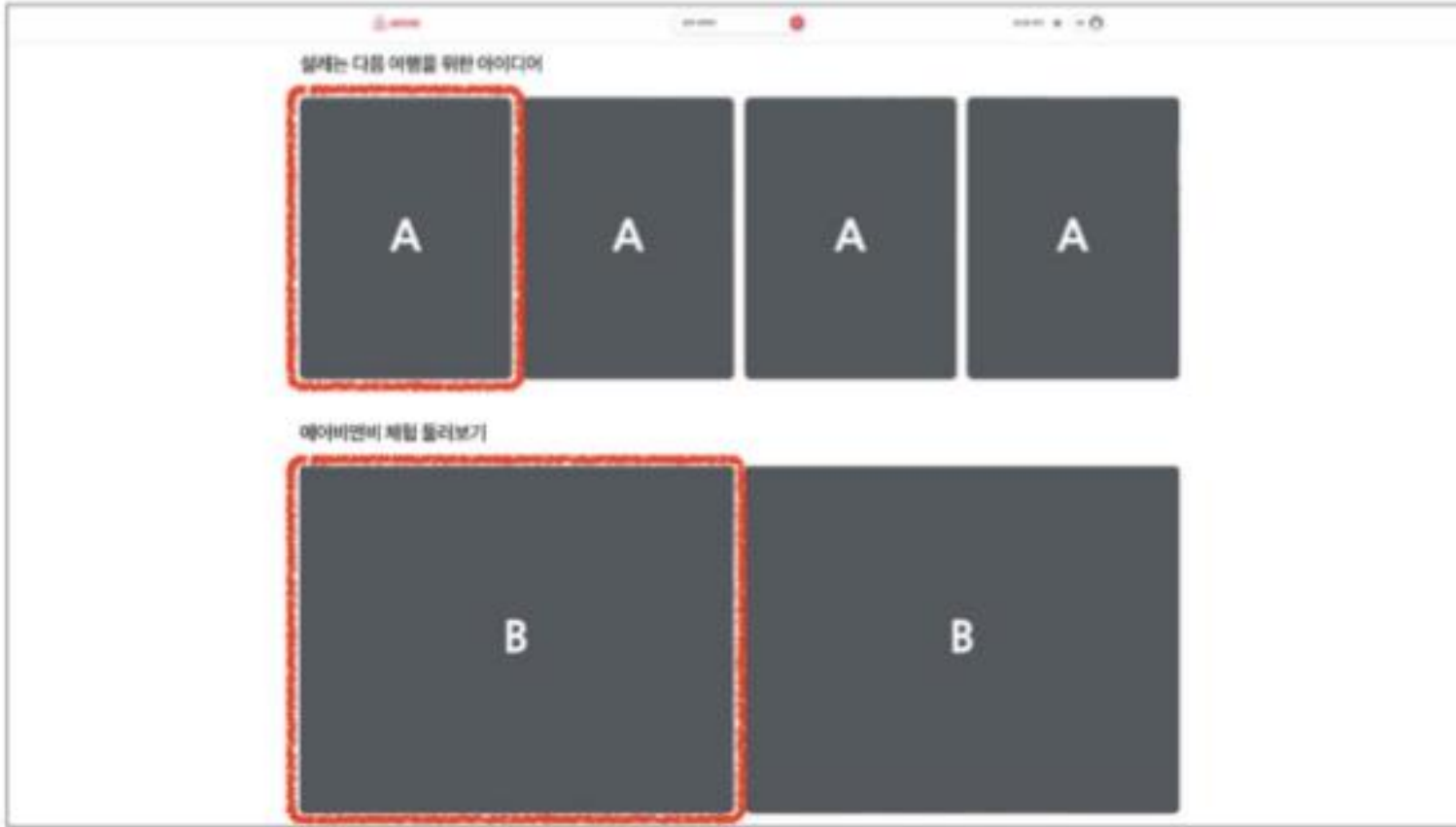




컴포넌트

# 리엑트는 컴포넌트 기반 구조이다.

## ➤ 에어비엔비 웹사이트의 컴포넌트 구조



컴포넌트를 이용하면 개발 시간과 유지 보수 비용을 줄일 수 있다.



# 리액트 컴포넌트

- 화면과 관련된 코드를 모두 App.js에 작성하면 코드양이 기하급수적으로 늘어나게 된다.
- 리액트에서는 기본적으로 화면 요소를 다양한 수준의 컴포넌트로 분할함으로써 재사용성과 유지보수성을 높인다.
- 리액트 컴포넌트-사용자 정의 태그 만드는 것
  - 리액트로 만들어진 앱을 이루는 최소 단위. 화면을 이루는 단위

# 리액트 컴포넌트화 예시

**Hello, React!!!**

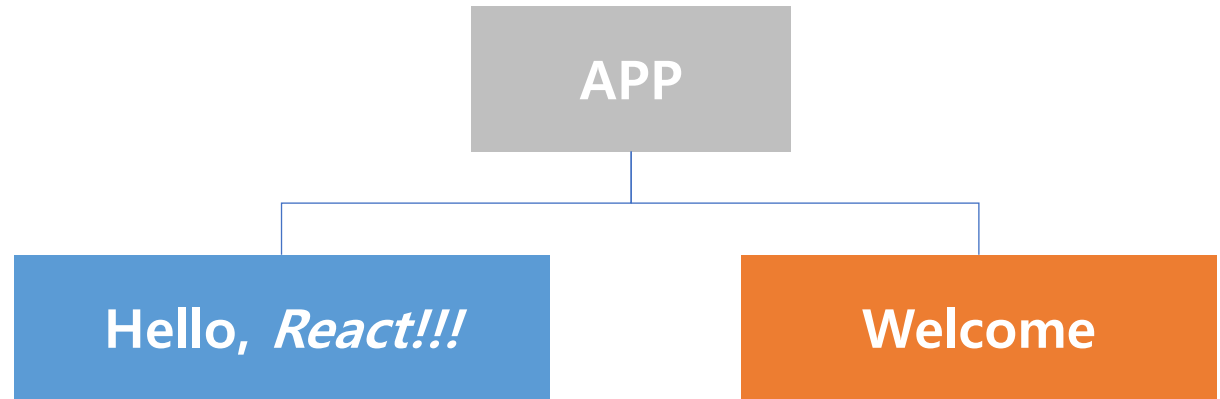
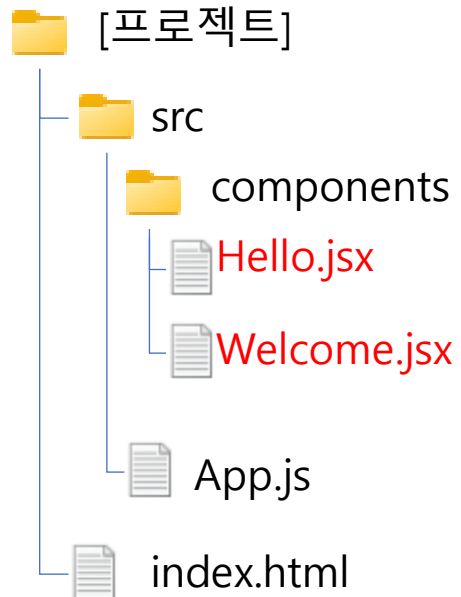
**Welcome**



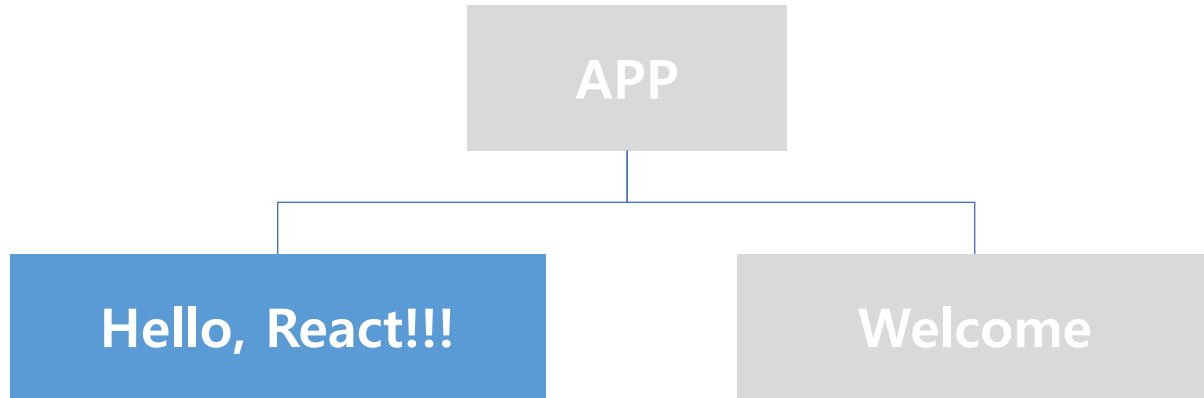
```
<div className="App">  
  <h1>  
    Hello, <i>React!!!</i>  
  </h1>  
  <h1>  
    Welcome  
  </h1>  
</div>
```

# 리액트 컴포넌트화 예시

```
<div className="App">
  <h1>
    Hello, <i>React!!!</i>
  </h1>
  <h1>
    Welcome
  </h1>
</div>
```



# 리액트 컴포넌트화 예시



Hello.jsx

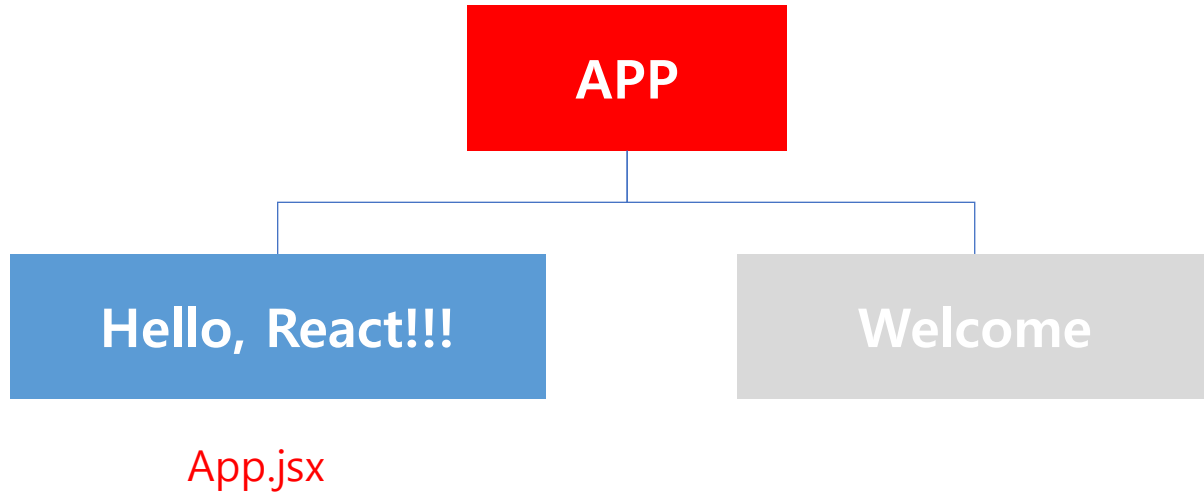
```
import React from "react";  
  
function Hello() {  
  return (  
    <div>  
      <h1>Hello, React!!!</h1>  
    </div>  
  );  
}  
  
export default Hello;
```

① React 라이브러리 불러오기

② 함수 정의, 함수명 반드시 대문자로 시작

③ 외부에서 사용할 수 있도록 보내내기

# 리액트 컴포넌트화 예시

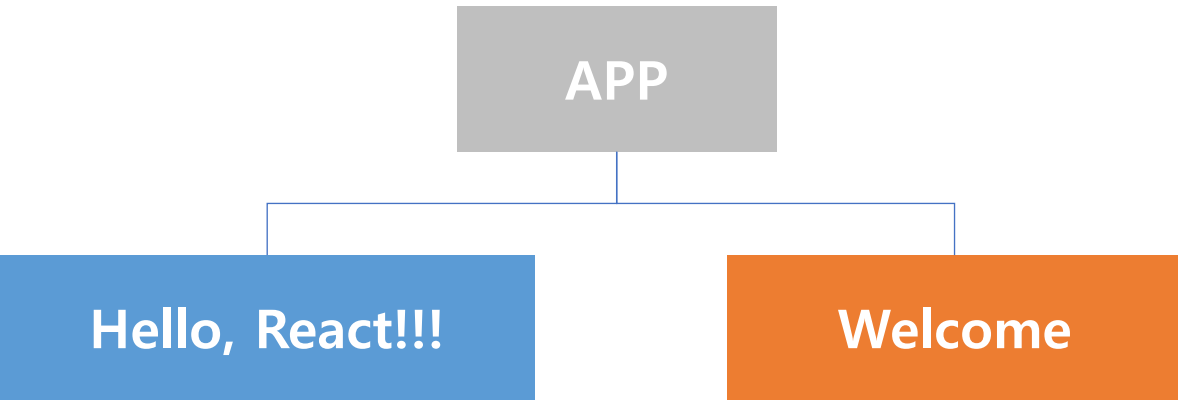


```
import Hello from "../components/Hello";  
  
function App() {  
  return (  
    <div>  
      <Hello />  
    </div>  
  );  
}
```

① Hello 컴포넌트 불러오기

② <Hello /> 요소 작성

# 리액트 컴포넌트화 예시



[Welcome.jsx 코드 완성하기]

```
import React from "react"; -----> ①

function Welcome() {
  ②
}

③
```

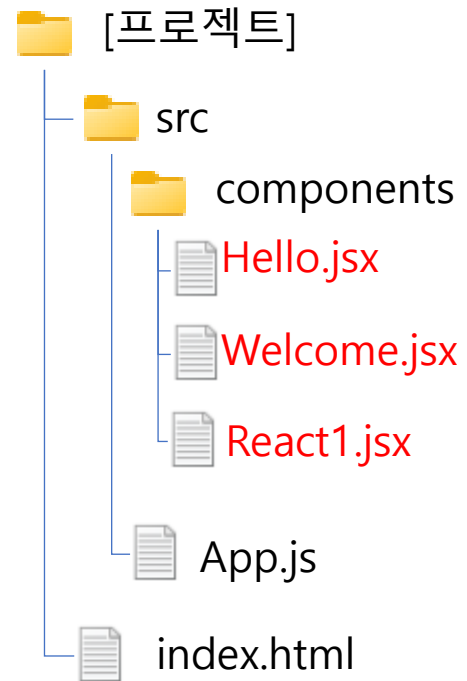
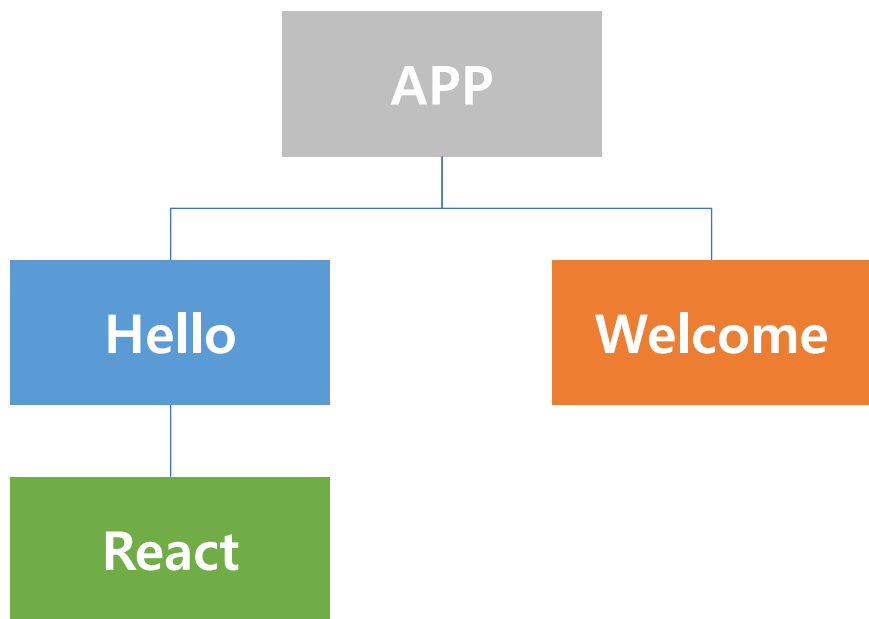
[App.jsx 에 Welcome 요소 추가하기]

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

function App() {
  return (
    <div>
      <Hello />
    </div>
  );
}
```



# 리액트 컴포넌트화 예시



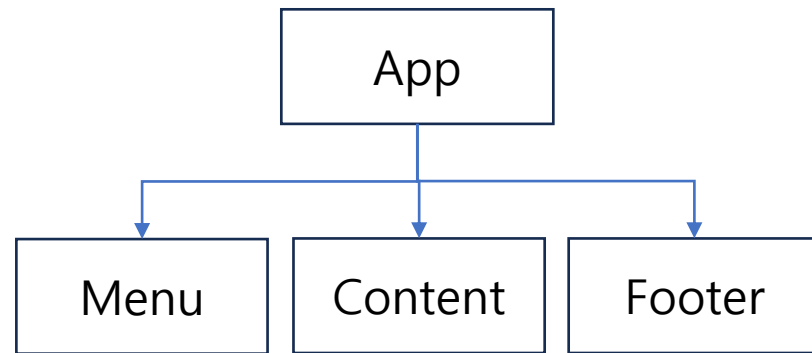
# 리액트 컴포넌트로 SPA 만들기

Home | menu1 | menu2 | menu3 | about me



어느 날, 어린 왕자는 사막 여우를 만났습니다. 여우는 첫 만남에 거리를 유지하라고 조언했지만, 어린 왕자는 사람들 사이에서는 서로를 이해할 수 없다고 말합니다. "넌 무엇을 찾고 있니?" 어린 왕자가 여우에게 물었습니다. "난 사람을 찾고 있어," 여우가 대답했습니다. "그들은 존스롭게 일상 반복을 하지. 내가 만난 사람들은 하는 대로 하지, 너와 나는 다르지." 어린 왕자는 여우가 가진 묘한 외로움을 느끼게 되었습니다. 그는 여우에게 깊이 통찰할 수 있는 우정의 의미를 배우게 되었고, 그를 잊지 못했습니다

Copyright 2024. fox&littlePrince

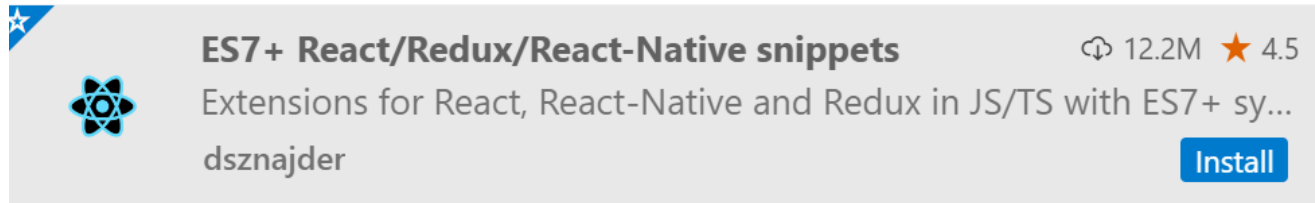


# 컴포넌트 작성 지원 확장앱 설치

- 컴포넌트 파일 작성 편의를 위한 확장앱 설치

Extension 설치: 단축키 등을 가지고 있음

ES7+React/Redux/React-Native snippets



Component 폴더 생성 & 파일 생성

src > component > 파일이름.js 생성 (컴포넌트 이름은 대문자로 시작할 것.)

컴포넌트 이름은 반드시 대문자로 시작.

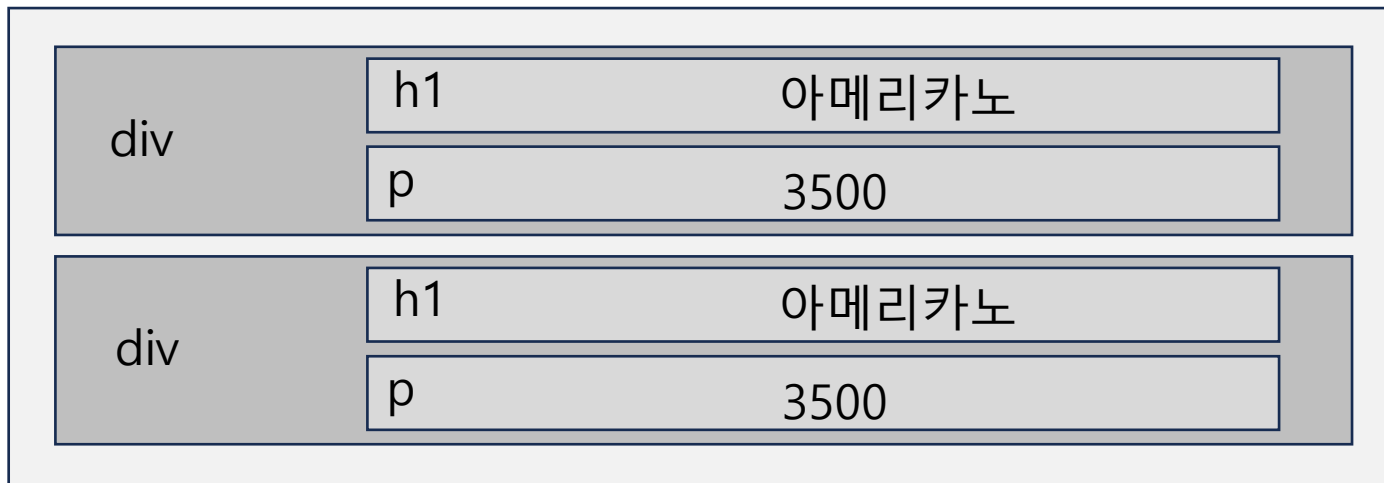
일반 HTML 태그를 구분할 수 있어야 한다.

HTML 태그는 소문자로, 컴포넌트는 대문자로

1. src > component 폴더 만들기 Menu.jsx 파일 생성
2. Rafce엔터 -> 기본 구조 만들어짐
3. Menu 라는 이름의 컴포넌트 구조 만듦.

# 리액트 컴포넌트

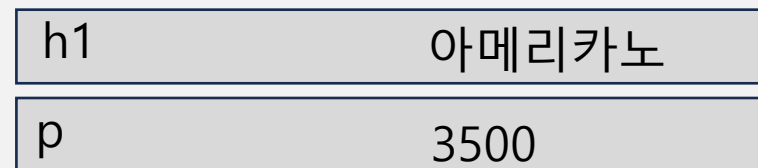
기존의 HTML



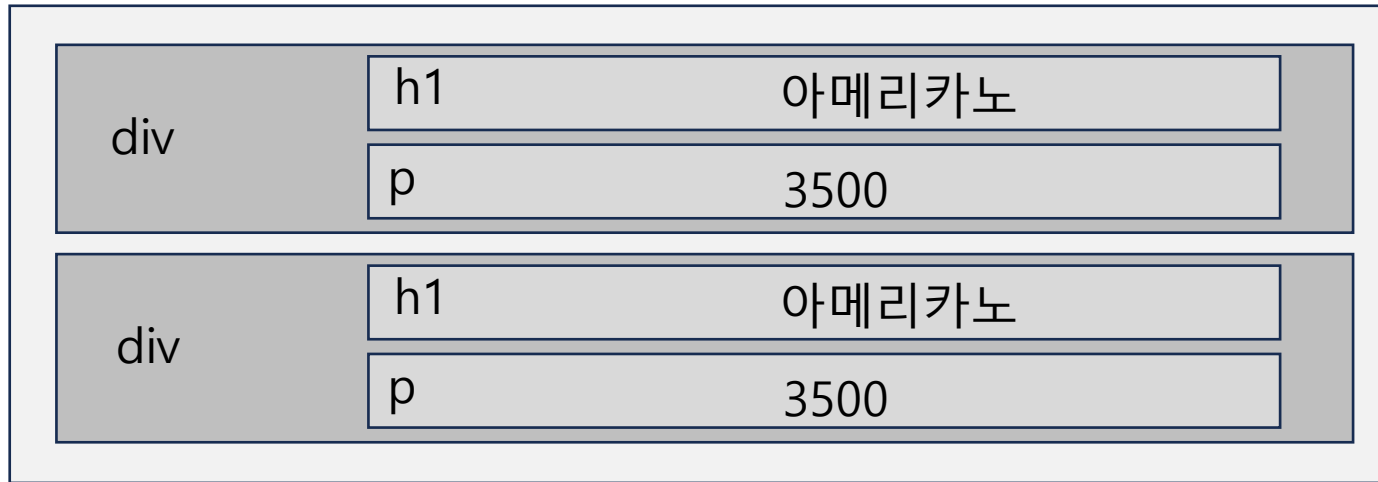
아메리카노 메뉴를 출력한다고 할 때,  
div, h1, p 태그로 표현한다.  
아메리카노 메뉴 반복될 때마다 같은 내용  
반복된다.  
따라서 아메리카노 메뉴 반복될 때마다,  
메뉴란 이름의 사용자 정의 태그를 만들어  
사용



Menu라는 컴포넌트



# 컴포넌트 생성



Menu 컴포넌트화

```
<div style={{ border: "1px solid gray" }}>  
  <h1>아메리카노</h1>  
  <p>3500원</p>  
</div>  
<div style={{ border: "1px solid gray" }}>  
  <h1>아메리카노</h1>  
  <p>3500원</p>  
</div>
```

# Menu 컴포넌트 생성

1. src > component 폴더 만들기 Menu.jsx 파일 생성
2. Rafce엔터 -> 기본 구조 만들어짐
3. Menu 라는 이름의 컴포넌트 구조 만듦.

```
import React from "react";

const Menu = () => {
  const style = {
    border: "1px solid gray",
  };
  return (
    <div style={style}>
      <h1>아메리카노</h1>
      <p>3500원</p>
    </div>
  );
};

export default Menu;
```

4. App.js에 Menu 컴포넌트 추가

```
import Menu from "../component/Menu";

function App() {

  return (
    <div>
      <Menu />
    </div>
  );
}
```

# 컴포넌트 생성시 주의사항

//컴포넌트: 화면을 구성하는 단위

//1. 컴포넌트 만들 때는 반드시 대문자로 시작해야 한다.

//2. 컴포넌트는 함수, 클래스 형식으로 만들 수 있다.

```
const Menu = () => { // 함수형 컴포넌트 만듦
  return(
    <div>
      <h1>아메리카노</h1>
      <p>3500</p>
    </div>
  )
}
```

//3. 다른 곳에서 사용해야 하기 위해서 export 를 통해서 내보내야 한다.

```
export default Menu;
```

/\*\*public 폴더의 index.html에 띄워주기 위해서는 Menu 컴포넌트를 App.js로 가져와서 사용한다. 따라서 export를 반드시 적어준다.

//4. App.js 문서에서 불러오기

```
import Menu from "../component/Menu";
```

과제

let display = "on"

데이터에 있는 값에 따라 메뉴 컴포넌트를 보여줄지 말지 결정하는 코드 작성(삼항연산자 사용)

**아메리카노**

3500원

**준비중입니다.**



- Book.jsx, Library.jsx 코드 분석후 활용하기

이 책의 이름은 처음 만난 파이썬입니다.

이 책은 총 300페이지로 이뤄져 있습니다.

이 책의 이름은 처음 만난 **AWS**입니다.

이 책은 총 400페이지로 이뤄져 있습니다.

이 책의 이름은 처음 만난 리액트입니다.

이 책은 총 500페이지로 이뤄져 있습니다.

# 실습

- Book.jsx, Library.jsx 코드 분석후 활용하기

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

<실행 결과>

**이 책의 이름은 처음 만난 파이썬입니다.**

이 책은 총 300페이지로 이뤄져 있습니다.

**이 책의 이름은 처음 만난 AWS입니다.**

이 책은 총 400페이지로 이뤄져 있습니다.

**이 책의 이름은 처음 만난 리액트입니다.**

이 책은 총 500페이지로 이뤄져 있습니다.

# 리액트 프로그래밍 화면 출력 실습

- Book.jsx, Library.jsx 코드 분석후 활용하기

1. src 폴더 아래에 components 폴더 만들기
2. components 폴더에 Book.jsx 와 Library.jsx 파일 만들기 (컴포넌트 파일명은 대문자로 시작)
3. index.js 문서 코드 수정하기

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

<실행 결과>

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

**이 책의 이름은 처음 만난 파이썬입니다.**

**이 책은 총 300페이지로 이뤄져 있습니다.**

**이 책의 이름은 처음 만난 AWS입니다.**

**이 책은 총 400페이지로 이뤄져 있습니다.**

**이 책의 이름은 처음 만난 리액트입니다.**

**이 책은 총 500페이지로 이뤄져 있습니다.**

# 리액트 프로그래밍 화면 출력 실습

## 4. 코드 작성:

```
import React from "react";                                     Book.jsx

function Book(props) {
  return (
    <div>
      <h1>`이 책의 이름은 ${props.name}입니다.`</h1>
      <h2>`이 책은 총 ${props.numOfPage}페이지로 이뤄져 있습니다.`</h2>
    </div>
  );
}

export default Book;
```

```
import React from "react";                                     Library.jsx
import Book from "../Book";

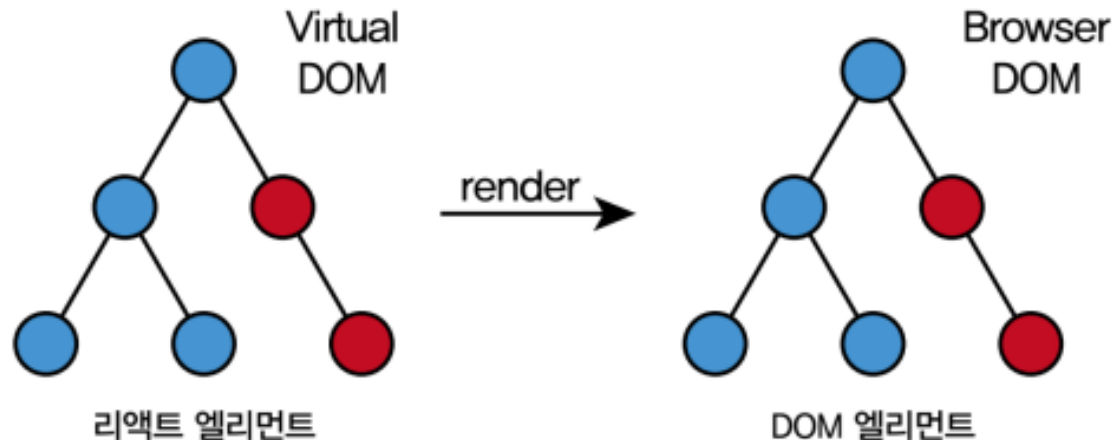
function Library(props) {
  return (
    <div>
      <Book name="처음 만난 파이썬" numOfPage={300} />
      <Book name="처음 만난 AWS" numOfPage={400} />
      <Book name="처음 만난 리액트" numOfPage={500} />
    </div>
  );
}

export default Library;
```

# 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 자바스크립트 객체 형태로 존재한다.
  - 컴포넌트 유형(type), 속성, 내부 자식들에 대한 정보 포함

```
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  }
```

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



# 엘리먼트 구조

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

```
01  {  
02      type: Button,  
03      props: {  
04          color: 'green',  
05          children: 'Hello, element!'  
06      }  
07  }
```

- 리액트 엘리먼트는 객체로 존재함
  - 엘리먼트를 객체로 만드는 명령
  - React.createElement(  
 type,  
 [props],  
 [...children]  
)

# createElement() 함수 동작 과정 확인

```
1 import React from "react";
```

Button.jsx

```
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;
```

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";

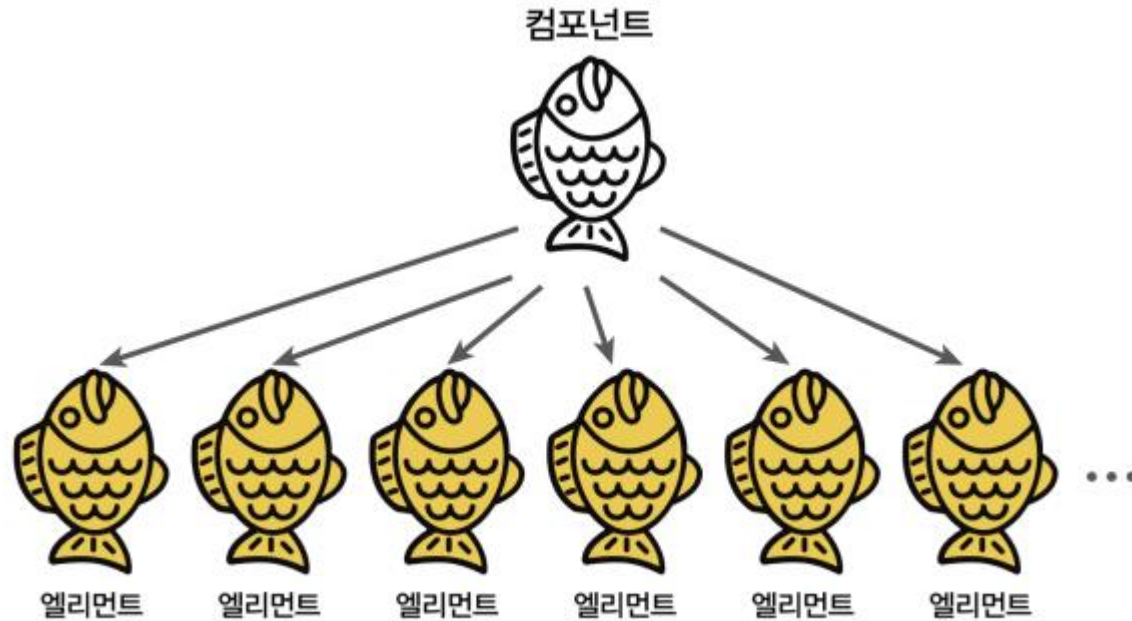
const root = ReactDOM.createRoot(document.getElementById("root"));

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

# 리엑트 엘리먼트 특징

- 불변성

- 엘리먼트는 생성된 후에 `children`이나 `attribute`을 바꿀 수 없다.

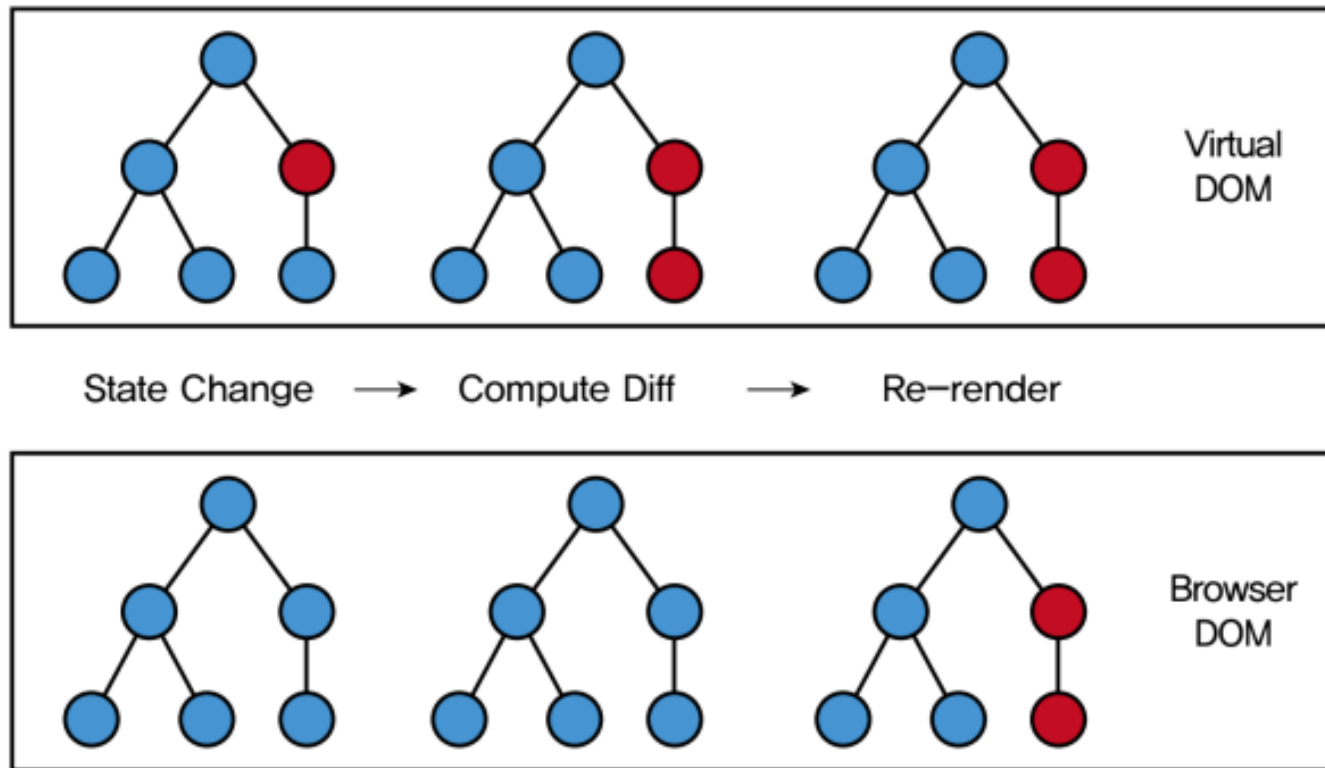


▶ 컴포넌트와 엘리먼트

# 리엑트 엘리먼트 특징

- 불변성

- 엘리먼트는 생성된 후에 `children`이나 `attribute`을 바꿀 수 없다.
- 화면에 변경된 엘리먼트들을 보여주기 위해서는 새로운 엘리먼트를 만든다.
- Virtual DOM은 변경된 부분을 계산하고 해당 부분만을 다시 렌더링한다.



# 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);
```

index.js

# 교재 실습

- 시계만들기