

리엑트를 다루는 기술

1장 리엑트 시작

1.1 왜 리엑트인가?

페이스북 개발 팀이 만든 데이터가 변할 때마다 기존 뷰를 날려버리고 처음부터 새로 렌더링 하는 방식. --> 애플리케이션 구조가 매우 간단해지고 작성해야할 코드양이 많이 줄어듦

1.1.1 리엑트 이해

- 자바스크립트 라이브러리
- DOM을 사용자 화면에 보여줌(렌더링)
- 재사용이 가능한 API, 컴포넌트

1.1.1.1 초기 렌더링

- render함수 : 초기렌더링 - 함수형 컴포넌트로 JSX를 반환
- 컴포넌트가 어떻게 생겼는지 정의
- 컴포넌트내부에 컴포넌트 들어갈수있음

1.1.1.2 조화과정

- 업데이트 : 화면갱신 컴포넌트에 데이터변화가 있을때 새로운 요소로 갈아끼움 <--- render함수

1.2 리엑트의 특징

1.2.1 Virtual DOM

1.2.1.1 DOM이란?

Document Object Model 객체로 문서 구조를 표현하는 방법 => 태그에 의한 구조표현

- 동적 UI에 최적화 되어 있지 않음 -> 자바스크립트를 사용하여 동적으로 만들 -> 규모 큰 앱 성능이슈 발생 -> DOM을 조작할 때마다 엔진이 웹페이지를 새로 그리기 때문에 업데이트가 너무 잦으면 성능이 저하될 수 있다

리엑트는 Virtual DOM을 사용하여 DOM업데이트를 추상화 -> DOM처리 횟수를 최소화, 효율적으로 진행 (=> useEffect)

1.2.1.2 Virtual DOM

실제 DOM(브라우저에 노출되는 Element)에 접근하여 조작하는 대신, 이를 추상화한(DOM을 메모리에 복사한것) 자바스크립트 객체를 구성하여 사용 => 지속적으로 데이터가 변화하는 대규모 어플리케이션 구축에 적합

1.2.2 기타 특징

다양한 라이브러리 보유 -> 적합한 오픈소스 고르기

1.3 작업환경 설정

Node.js 설치 - 프로젝트를 개발하는데 필요한 주요도구들이 Node.js를 사용하며 패키지 매니저 도구인 npm(진보된 도구: yarn)이 설치됨

ex) 지금은 자동 실행됨 -바벨(babel) : ECMAScript를 호환 -웹팩(webpack) : 모듈화된 코드를 한 파일로 합치고 (번들링) 코드를 수정할때마다 웹 브라우저를 리로딩하는 등의 여러기능

1.3.2 yarn

npm을 대체할수 있는 도구로 npm보다 더 빠르며 효율적인 캐시 시스템과 기타 부가기능 제공 / npm사용해도 무방

```
$ npm install -g yarn
```

1.3.3.1 VS CODE 확장 프로그램 설치

```
$ npm install -g eslint
```

ESLint: 자바스크립트 문법 및 코드 스타일을 검사해 주는 도구

1.3.5 create-react-app으로 프로젝트 생성

```
$ yarn create react-app hello-react
```

프로젝트 이름은 대문자 불가능

```
$ cd hello-react
```

터미널에서 프로젝트 폴더로 이동

```
$ yarn start
```

2장 JSX

2.1 코드 이해하기

- 프로젝트 생성과정에서 node_modules 디렉토리에 react모듈이 설치됨
- import 구문을 통해 리액트를 불러와서 사용할수 있다(일반 자바스크립트에서는 불가) => import : 다른 js 불러오기
-> 분할된 js들을 import해서 조립 => 번들러(bundler : webpack)를 사용 : 파일을 묶듯이 연결
- 코드 짤때 편의상 분할해서 개발하고 사용시에는 웹브라우저에서 인식할 수 있도록 병합을 해줌 => yarn start

작성(ES6):Node용 -웹팩-> 병합 -바벨-> 변환(ES5):웹브라우저용

- function을 사용하여 만든 컴포넌트 : 함수형 컴포넌트
 - 프로젝트에서 컴포넌트를 렌더링하면 함수에서 반환하고 있는 내용을 나타냄
 - return으로 반환되는 코드는 JSX
-

2.2 JSX란?

자바스크립트 확장 문법 코드 번들링 되는 과정에서 바벨을 사용하여 일반 자바스크립트 형태로 변환

```
<div>
Hello<b>react</b>
</div>
```

위 아래 같은 코드

```
function App(){
  return React.createElement("div", null, "Helo",
    React.createElement("b", null, "react"));
}
```

- JSX는 공식적인 자바스크립트 문법이 아님 개발자들이 임의로 만든 문법
-

2.3 JSX의 장점

2.3.2 더욱 높은 활용도

```
document.getElementById('root')
```

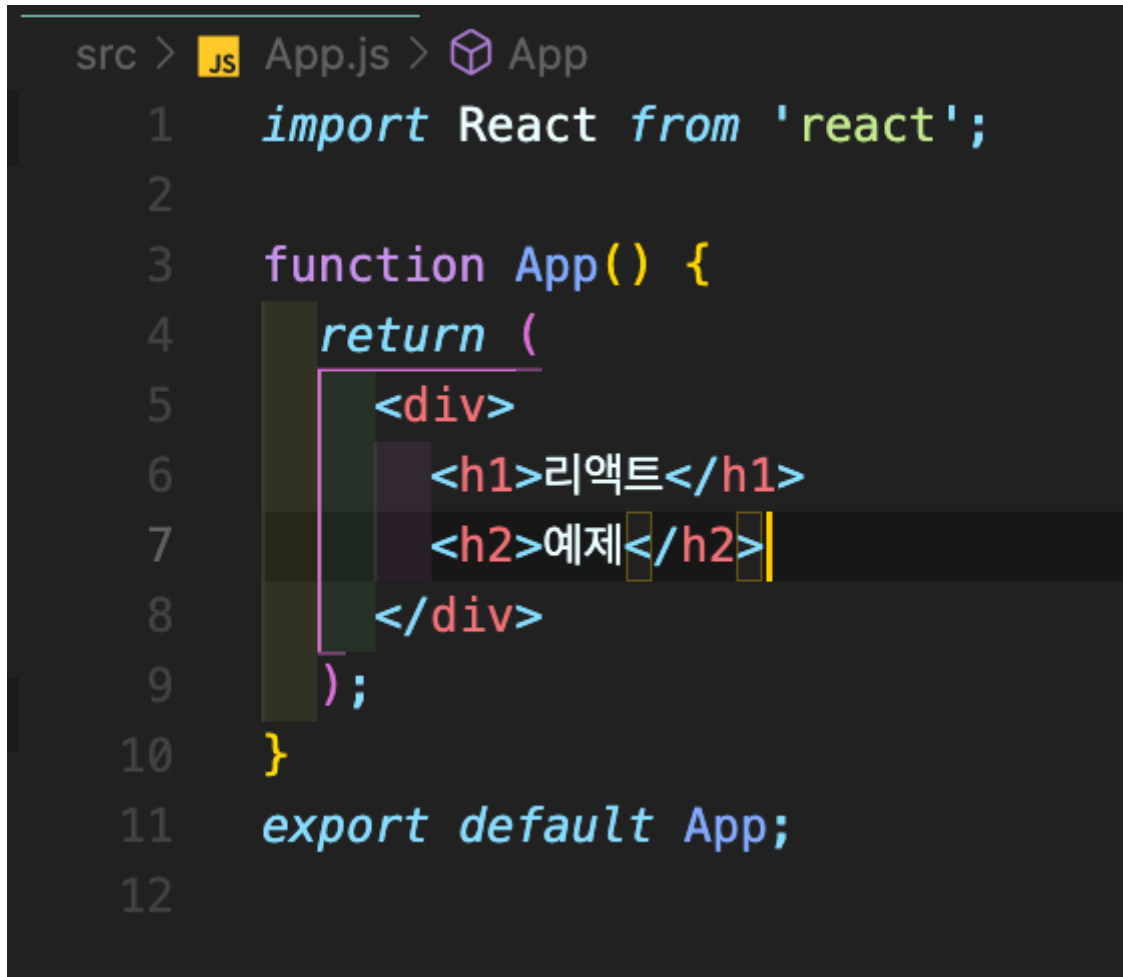
id가 root인 요소안에 렌더링을 하도록 설정 public/index.html파일을 열어보면 있다.

2.4.1 감싸인 요소

컴포넌트에 여러요소가 있다면 반드시 부모 요소 하나로 감싸야함

```
<Fragment></Frangment>
<></>
```

브라우저에서는 그냥 div



```
src > Js App.js > App
1  import React from 'react';
2
3  function App() {
4    return (
5      <div>
6        <h1>리액트</h1>
7        <h2>예제</h2>
8      </div>
9    );
10 }
11 export default App;
12
```

2.4.2 자바스크립트 표현

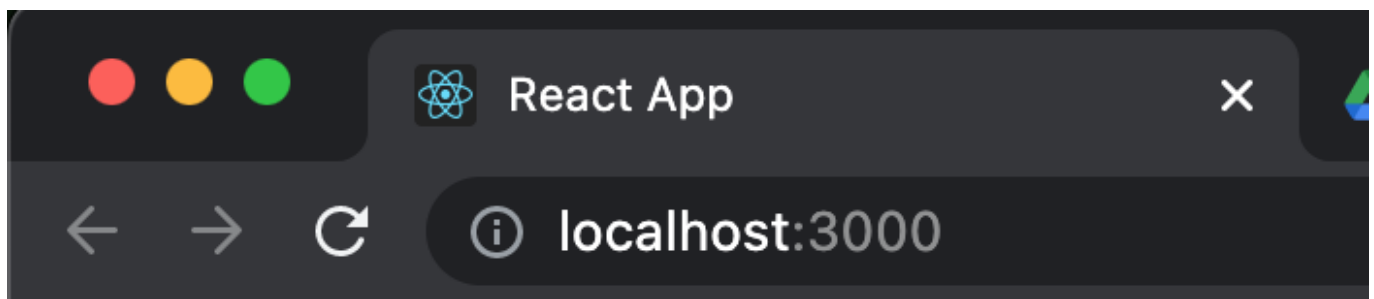
- 자바스크립트 표현식을 작성하려면 jsx내부에서 코드를 {} 로 감싸준다

2.4.3 if문 대신 조건부 연산자

- jsx내부에서는 if문을 사용할수 없어서 조건부 연산자(삼항 연산자)를 써야함

```
{조건 ? 결과1(참인경우 출력할 값) : 결과2(거짓일때 출력할 값)}
```

```
src > JS App.js > App
1  import React from 'react';
2
3  function App() {
4    const name = '리액트';
5    return (
6      <div>
7        {name === '리액트' ? <h1>리액트입니다</h1> : <h2>리액트 아닙니다</h2>}
8      </div>
9    );
10 }
11 export default App;
12
```



리액트입니다

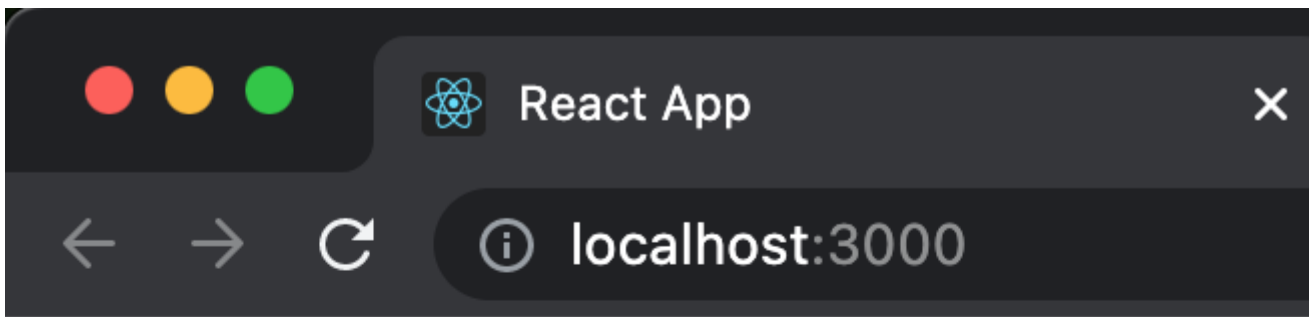
2.4.4 and연산자(&&)를 사용한 조건부 렌더링

{조건이 참인경우만 뒤에 실행 && 실행할코드}

```

JS App.js U X JS index.js U
src > JS App.js > App
1  import React from 'react';
2
3  function App() {
4    const number = 0;
5    return number && <div>&& 조건문 | number가 0일때 </div>;
6  }
7  export default App;
8

```



0

- 조건이 0일때 조건이 거짓이므로 뒤에코드는 렌더링하지 않지만 0은 보여줌
- 보통 여러줄의 jsx일 경우 ()로 감싸준다. 필수는 아님

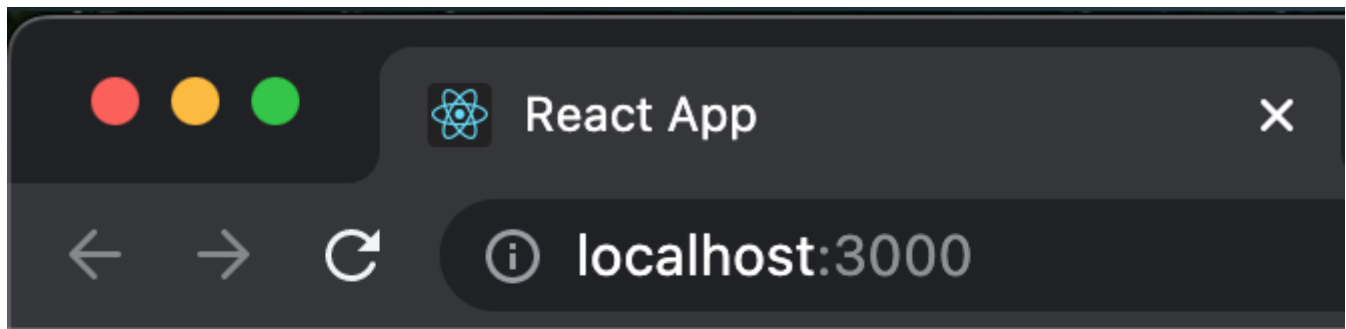
2.4.5 undefined를 렌더링 하지 않기

함수가 undefined를 return하면 에러 -> 반드시 tag를 리턴(jsx)

{앞의 조건이 false, null, undefined일 때 || 뒤에코드 출력}

2.4.6 인라인 스타일링

- DOM요소에 스타일을 적용할 때는 객체형태로 넣어주어야함 - 카멜표기법
 - 객체를 선언하지 않고 style값을 지정하려면 -> {{이중으로 감싸주어야한다}}



React

```
src > JS App.js > App
1  import React from 'react';
2
3  function App() {
4    const name = 'React';
5
6    return (
7      <div
8        style={{
9          backgroundColor: 'black',
10         color: 'aqua',
11         fontSize: '48px',
12         fontWeight: 'bold',
13         padding: 16,
14       }}
15     >
16       {name}
17     </div>
18   );
19 }
```



```
19 }  
20 export default App;  
21
```

2.4.7 class 대신 className

최신버전부터는 class를 className으로 변환시켜주고 경고를 띄웁니다.

2.4.8 꼭 닫아야하는 태그

jsx는 태그를 닫지않으면 오류 -> self-closing tag를 사용해도 됨

2.4.9 주석

```
{/* 보통 주석 쓰는법 */} // 시작태그 안에서는 기본 주석도 가능
```