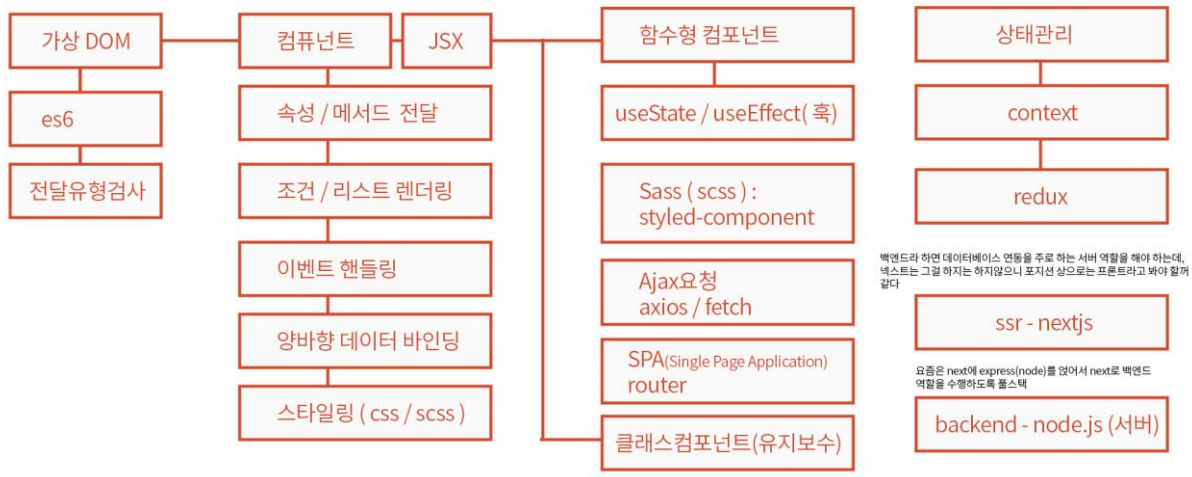
React 시작



컴포넌트

컴포넌트는 UI 를 구성하는 조각(piece)에 해당되며, 독립적으로 분리되어 재사용을 됨을 목적으로 사용됩니다.
React 앱에서 컴포넌트는 개별적인 JavaScript 파일로 분리되어 관리합니다.
(예: Header, HeaderTitle, Wrapper, List, ListItem 컴포넌트)

함수형 컴포넌트

React 컴포넌트는 개념상 JavaScript 함수와 유사합니다. 컴포넌트 외부로부터 속성(props)을 전달 받아 어떻게 UI 를 구성해야 할지 설정하여 React 요소(JSX 를 Babel 이 변환 처리)로 반환합니다. 이러한 문법 구문을 사용하는 컴포넌트를 React 는 '함수형(functional)'으로 분류합니다.

함수형? 클래스형? 각 컴포넌트는 차이가 있나요?

React 세계관에서 함수형과 클래스 컴포넌트는 유사합니다. 다만, 클래스 컴포넌트의 경우 함수형 컴포넌트에 없는 기능을 추가적으로 사용할 수 있다는 점이 다릅니다. 예전에는 클래스형컴포넌트를 사용 현재는 함수형컴포넌트

react 공식페이지(자습서) 참고

클라이언트 사이드 렌더링(CSR) - create-react-app

Create React App : 번거로운 개발 환경 구성(개발 서버, Webpack, Babel 등많음) 없이 React 를 바로 시작하기 위해

npm start (yarn start)	React 프로젝트 개발 서버를 시작합니다.
npm run build (yarn build)	배포를 위해 앱을 정적(Static) 파일로 번들(Bundle) 합니다.

JSX 는 JavaScript 문법 확장(JavaScript eXtension). 으로 구문이 HTML 과 유사합니다. React 애플리케이션 제작 시 꼭 JSX 가 필요한 것은 아니지만, JavaScript 로 UI View 를 구성하는 마크업하는 것은 매우 까다로우므로 특별한 경우가 아니라면 JSX 사용을 권장합니다.

JSX 가 하는 일은 React 요소(Element)를 만드는 겁니다. React 요소는 실제 DOM 요소가 아니라, JavaScript 객체입니다.

프로젝트 생성구조

```
├── README.md
├── node_modules/ # 개발 의존 모듈 집합 디렉토리
├── package.json
├── public/ # 정적 리소스 디렉토리
│   ├── favicon.ico
│   ├── index.html # 애플리케이션 기본 템플릿
│   └── manifest.json
├── src/ # React 애플리케이션 개발 디렉토리
│   ├── App.css
│   ├── App.js # 애플리케이션 파일
│   ├── App.test.js
│   ├── index.css
│   ├── index.js # 엔트리 파일
│   ├── logo.svg
│   └── serviceWorker.js
└── yarn.lock
```

public/index.html

```
    <div id="root"></div>
    <!--
      이 HTML 파일은 템플릿입니다.
      브라우저에서 직접 열면 빈 페이지가 나타납니다.
```

src/index.js

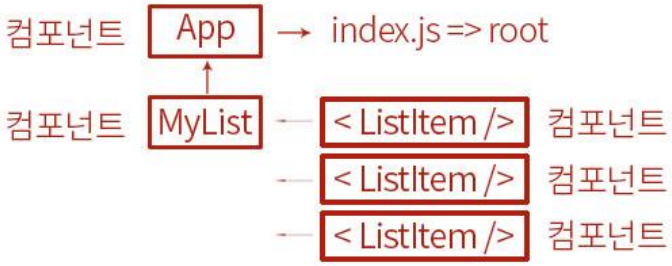
```
import React from 'react' // React 모듈 로드
import ReactDOM from 'react-dom'  // ReactDOM 모듈 로드
import './index.css'     // 메인(인덱스) 스타일 로드
import App from './App'  // 앱 컴포넌트 로드
import * as serviceWorker from './serviceWorker' // 서비스 워커 로드

// ReactDOM 모듈의 렌더 함수를 사용해 #root (src/index.html) 요소
// 내부에 동적으로 App 컴포넌트(React Element)를 렌더링 합니다.
ReactDOM.render(<App />, document.getElementById('root'))
serviceWorker.unregister()
```

App.js

```
import React from 'react'   // React 모듈 로드
import logo from './logo.svg'   // 로고 이미지 로드
import './App.css'  // App 스타일 로드

// 함수형 컴포넌트(Functional Component)
function App() {
  // JSX(JavaScript 문법 확장) 반환
  return (
    <div>

    </div>
  )
}

export default App  // App 컴포넌트 모듈 내보내기
```

컴포넌트 이름은 첫글자 반드시 대문자로 작성한다 App , List 등

< MyList />



1. class component

```
import React , { Component } from 'react';
class 컴포넌트이름 extends Component {
 render() {
    return ( JSX 문법)
  }
}
export default 컴포넌트이름;
```

2. function component

```
import React from 'react'
const 컴포넌트이름 = () => {
    return ( JSX 문법 )
}
export default 컴포넌트이름;
```

 # JSX 규칙 : javascript + XML

JSX 는 리액트 컴포넌트를 작성하면서 return 문에 사용하는 문법 얼핏보면 HTML 같지만, 실제로는 자바스크립트

규칙

1. 태그는 반드시 닫아줘야 한다.
2. 최상단에서는 반드시 div 로 감싸주어야 한다. (Fragment 사용 , <> 상황에따라)
3. JSX 안에서 자바스크립트 값을 사용하고 싶을때는 {}를 사용한다.
   변수값 출력예시 참고 -> { name }
4. 조건부 렌더링을 하고싶으면 &&연산자나 삼항연산자를 사용한다.
5. 인라인 스타일링은 항상 객체형식으로 작성한다.
6. 별도의 스타일파일을 만들었으면 class 대신 className 을 사용한다.
7. 주석은 {/* */}을 사용해 작성한다.

1. 태그는 반드시 닫아줘야 한다.
2. 최상단에서는 반드시 div 로 감싸주어야 한다.

```
  return (
     <div>
       <p>hello react!</p>
        <input type="text" />
     </div>
   )
```

3. JSX 안에서 자바스크립트 값을 사용하고 싶을때는 { }를 사용한다.

```
return (
      <div>
       hello {name}
      </div>
   )
```

4. 조건부 렌더링을 하고싶으면 &&연산자나 삼항연산자를 사용한다.

```
return (
        <div>
         { true ? console.log('참') : console.log('거짓')   }
       </div>
    )
```

5. 인라인 스타일링은 항상 객체형식으로 작성한다.

```
    스타일 작성시 - 빼고 첫글자는 대문자로 작성한다
    font-size : fontSize    , background-color : backgroundColor , line-height : lineHeight , text-indent :
    textIndent

   const style ={   bckgroundColor :'red' }
<div style= { style }>react</div>
   <div style={{ backgroundColor: 'red' }}>react</div>
```

6. 별도의 스타일파일을 만들었으면 class 대신 className 을 사용한다. (권장사항)

```
   <div className="App">   react   </div>
```

7. 주석은 {/* */}을 사용해 작성한다.

```
 <div>
        {/* 주석은 이렇게 */}
        <h1 // 태그안에서 주석작용 >
           react
        </h1>
 </div>
```