React

리액트 공식 홈페이지: https://react.dev/

리액트란?

• 사용자 인터페이스(UI)를 구축하기 위해 메타(페이스북)에서 개발한 Javascript 기반의 라이브러리
• SPA(Single Page Application) 개발에 가장 적합한 라이브러리
• 화면상에 데이터가 변경되는 경우 페이지 전체를 렌더링하는 것이 아니라 필요한 부분만 렌더링 할 수 있다.
• 웹 앱(Web App) 또는 네이티브 앱(Native App)
• 유지보수를 쉽게, DOM 관리
• 성능 최적화 쉽게
• 컴포넌트 단위로 구성하여 코드 재사용성이 높으며, 독립적인 유지보수 가능
• 대부분 공식 라이브러리가 없음 (높은 자유도)
• 자바스크립트 친화적 ES6 기반로 배우기가 쉽다

1. Nodejs

- https://nodejs.org/ko/
- 자바스크립트 런타임 환경
- 리액트 프로젝트를 준비하는 데 필요한 webpack, babel 등의 도구들을 실행하는 데 사용된다.
- npm은 자바스크립트 프로그래밍 언어를 위한 패키지 관리자이다.

2. Yarn

- https://classic.yarnpkg.com/en/docs/install#windows-stable
- 자바스크립트 패키지를 관리하기 위해서 사용된다.
- yarn은 npm동작 방식과 비슷하지만 npm의 단점을 보완해 성능과 속도를 개선한 패키지 관리도구이다.

Node.js 를 설치하면 npm 이 설치되어서 npm 으로 해도 되긴 하지만, yarn을 사용하면 훨씬 빠르다.

npm VS yarn

npm은 여러 패키지를 설치할 경우 패키지가 완전히 설치될 때까지 기다렸다가 다른 패키지로 이동한다. 작업이 패키지별로 순차적으로 실행된다는 것이다. yarn은 이 작업들을 병렬로 설치하기 때문에 성능과 속도가 향상된다는 장점이 있다.

노드 설치가 완료된 후
C:\Users\yeoni>node -v

3. VS Code (에디터)

- https://code.visualstudio.com/
- **VS Code 확장설치 (부가기능설치)**
- 기본 브라우저는 Chrome 설정해야 한다.

톱니바퀴 → Settings → browser 검색 → 아래로 내려오면 Custom Browser : Chrome 선택
톱니바퀴 → Settings → local 검색 → Extensions(왼쪽)
→ 아래로 내려오면 Use local IP as host 체크

확장 페키지

① Live Server
② Korean Language Pack for Visual Studio Code
③ Auto Close Tag
④ Auto Complete Tag
⑤ Auto Import
⑥ Auto Rename Tag
⑦ Reactjs code snippets - 코드 자동 생성

ES6

[실습]

Folder : ES6

File : 01_템플릿 리터럴.html

=> **html 문서를 완성하려면 ! 누르고 하고 Enter**

리액트 컴포넌트

- 컴포넌트는 UI를 구성하는 조각(piece)에 해당하며, 독립적으로 분리되어 재사용이 됨을 목적으로 사용된다.
- React 앱에서 컴포넌트는 개별적인 JavaScript 파일로 분리되어 관리한다.

함수형 컴포넌트

- React 컴포넌트는 개념상 JavaScript 함수와 유사하다.
- 컴포넌트 외부로부터 속성(props)을 전달받아 어떻게 UI를 구성해야 할지 설정하여 React 요소 (JSX를 Babel이 변환 처리)로 반환한다.
- 이러한 문법 구문을 사용하는 컴포넌트를 React는 '함수형(functional)'으로 분류한다.

JSX 규칙 : JavaScript + XML

- JSX는 리액트에서 사용하는 파일로 JavaScript 코드 안에 HTML과 유사한 코드를 작성할 수 있게 해준다.
- JSX는 리액트 컴포넌트를 작성하면서 return 문에 사용하는 문법이다.
- 브라우저에서 직접 실행되지 않으며, Babel과 같은 변환기를 통해 Javascript로 변환 후 실행한다.
- JSX가 하는 일은 React 요소(Element)를 만드는 것이다.
- 얼핏 보면 JSX는 JavaScript 문법 확장(JavaScript eXtension) 으로 구문이 HTML과 유사하다.
  하지만 React 요소는 실제 DOM 요소가 아니라, JavaScript 객체이다.
- 리액트 자체 빌드 도구 덕분에 *.js, *.jsx로 작성해도 아무런 문제 없이 실행은 가능하나 jsx파일로 컴포넌트 만들 것을 권장한다.

규칙

1. 태그는 반드시 닫아줘야 한다.
2. 최상단에서는 반드시 div로 감싸주어야 한다. (Fragment 사용, <></> 상황에 따라)
3. JSX안에서 자바스크립트 값을 사용하고 싶을 때는 { }를 사용한다.
   변수값 출력 예시 참고 -> { name }
4. 조건부 렌더링을 하고 싶으면 &&연산자나 삼항 연산자를 사용한다.
5. 인라인 스타일링은 항상 객체형식으로 작성한다.
   스타일 작성 시 - 빼고 첫 글자는 대문자로 작성한다.
6. 별도의 스타일 파일을 만들었으면 class 대신 className을 사용한다. (권장사항)
7. 주석은 {/* */}을 사용해 작성한다.

※ <></> Fragment

브라우저 상의 HTML 트리 구조에서 흔적을 남기지 않고 그룹화를 해준다.
그룹화를 하는 이유는 실행될 때 JSX에 작성한 내용은 하나의 JavaScript 객체로 변환되는데 하나의 태그로 감싸지지 않으면 변환이 되지 않는다.

[실습] 첫 번째 : React 프로젝트 생성
Project : day01

① Ctrl + backtick(`) - 터미널 열기
② npx create-react-app day01

※ npx create-react-app day01 명령으로 프로젝트 생성 시 에러가 뜨면
[에러]

```
npm ERR! code ENOENT
npm ERR! syscall lstat
npm ERR! path C:\Users\bitcamp\AppData\Roaming\npm
npm ERR! errno -4058
npm ERR! enoent ENOENT: no such file or directory, lstat 'C:\Users\bitcamp\AppData\Roaming\npm'
npm ERR! enoent This is related to npm not being able to find a file.
npm ERR! enoent
```

[**해결 방법** - 첫 번째 또는 두 번째 방법으로 해결해본다]
첫 번째 방법 => **C:\Program Files\nodejs\node_modules** 안의 **npm 폴더**를
C:\Users\bitcamp\AppData\Roaming에 복사한다.

두 번째 방법 => Node를 제거하고 다시 설치해본다.
※ Nodejs 삭제
① 제어판 - 프로그램 삭제
② 아래의 경로를 찾아서 모두 삭제
C:\Program Files (x86)\Nodejs
C:\Program Files\Nodejs
C:\Users\User\AppData\Roaming\npm
C:\Users\User\AppData\Roaming\npm-cache

day01

public

index.html --- index.html

src (우리가 작업하는 위치)

components

Dog.js

Test01.js

Test02.js

App.js - main 이다 --- App.jsx

index.js --- index.jsx

[서버 실행] 서버가 가동되면서 크롬에 결과가 뜬다.

PS D:\React\workspace> cd day01

PS D:\React\workspace\day01> npm start / yarn start

※ Project를 USB에 복사할 때는 Project안의 node_modules 폴더의 용량이 크기 때문에 가져가지 않는다. ---> 삭제한 후 가져간다.

node_modules 폴더가 없으면 **npm start**가 안되므로 다시 **npm i** 수행해서 **node_modules** 폴더를 다시 만들어 사용하면 된다.

※ *.js 파일에서 rsc 입력 후 엔터를 치면 자동으로 틀이 만들어진다.

public/index.html

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

src/index.js

ReactDOM 모듈의 렌더 함수를 사용해 #root (public/index.html) 요소 내부에 동적으로 App 컴포넌트(React Element)를 렌더링 한다.

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

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

App.js

```
import React from 'react' //React 모듈 코드
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
```

[실습] 두 번째 : React 프로젝트 생성
Project : day01_vite

Vite 활용 방식

- 기존 방식은 세팅에 시간이 오래 걸리고 다소 무겁다는 특징이 있다.
- Vite를 이용하면 세팅도 빠르고 개발 단계에서도 수정사항을 빠르게 반영하여 실행 화면을 업데이트 해주기 때문에 매우 유용하다.
- Vite 사용을 위해선 Node.js가 설치되어 있어야 한다.

※ Vite

Frontend 개발을 위해 설계된 빌드도구.
빠른 개발 환경을 제공하며, React, Vue, Svelte 등 다양한 프레임워크를 지원

① Ctrl + backtick(`) - 터미널 열기

② npm create vite@latest day01_vite

Props

Props란?

- properties의 약자로 부모 컴포넌트에서 자식 컴포넌트로 전달되는 데이터(객체, 배열 등)나 함수이다.
- 컴포넌트 간의 데이터를 전달하기 위해 사용되는 리액트 객체이다.

자식 간의 전달은 할 수 없다. 부모와 자식 간에서만 가능하다

보내는 방법

- import 한 컴포넌트 태그 내에 속성이름=값 형태로 작성하여 전달
- 여러 개의 값 전달할 때는 중괄호 안에 전달받는 속성 이름을 콤마(,)로 구분하여 나열
- 숫자를 보내는 경우 {} 안에 작성
- JavaScript Object로 전달하는 데이터는 {{ }} 중괄호 2개를 사용하여 내부에 이름과 값을 작성한다.

받는 방법

- 부모로부터 전달받은 props를 직접 받으려면 { } 안에 속성 이름 작성한다.
- 함수의 매개변수를 props로 작성한 다음 props.속성이름 형태로 값을 사용할 수 있다.

Hook ?

https://ko.reactjs.org/docs/hooks-state.html

Hook은 React 16.8버전에 새로 추가되었습니다.
Hook은 클래스 컴포넌트를 작성하지 않아도 state와 같은 특징들을 사용할 수 있습니다.

Hook의 개요

함수형 컴포넌트는 렌더링 할 때마다 내부의 것들을 기억하지 못한다.
다시 생성, 초기화해야 한다. (변수, 함수 등)

내부의 것들을 유지하기 위해서 hook이 등장했다 - useXXX

useState

- 값이 유동으로 변할 때
[형식]
- const [상태데이터, 상태데이터의 값을 변경해주는 함수] = React.useState(초기값);

※ 웹 스토어에서 **React Developer Tools** (Chrome에 추가)

Hooks - useRef

- 직접 DOM 요소에 접근해야 할 때, useRef 훅을 사용하여 DOM 요소에 직접 접근이 가능하다.
- 리렌더링을 하지 않는다.
- useRef 훅이 반환하는 ref 객체를 이용해서 자식 요소에 접근이 가능하다.
- useRef는 .current 프로퍼티로 전달된 인자(initialValue)로 초기화된 변경 가능한 ref 객체를 반환한다.

Ref를 사용해야 할 때
포커스, 텍스트 선택영역, 혹은 미디어의 재생을 관리할 때
애니메이션을 직접적으로 실행시킬 때
서드 파티 DOM 라이브러리를 React와 같이 사용할 때

- public에 있는 이미지 폴더는 index.html를 기준으로부터 상대경로를 지정해야 한다.
- index.html 안의 이곳으로 렌더링 되기 때문이다.

- src 안에 있는 이미지 파일 처리는 참조변수를 사용한다.
  import 참조변수 from '이미지경로';

- event.preventDefault()
  a 태그는 href를 통해 특정 사이트로 이동하거나,
  submit 태그는 값을 전송하면서 창이 새로고침(reload) 된다.
  이런 태그의 이벤트 기능을 preventDefault를 통하여 동작하지 않도록 막을 수 있다.

useEffect란?

- useEffect는 렌더링, 혹은 변수의 값 혹은 오브젝트가 달라지게 되면, 그것을 인지하고 업데이트를 해주는 함수이다.
- useEffect는 콜백 함수를 부르게 되며, 렌더링 혹은 값, 오브젝트의 변경에 따라 어떠한 함수 혹은 여러 개의 함수들을 동작시킬 수 있다.
- 렌더링 후 useEffect는 무조건 한번은 실행된다.

[형식]
① 컴포넌트가 나타날 때 딱 1번만 함수가 호출
useEffect(() => {
}, [ ]);

② 특정 props가 바뀔 때마다 함수가 호출
useEffect(() => {
}, [props]);

useEffect 라는 Hook을 사용하여 할 수 있는 3가지 동작
- 컴포넌트가 마운트 됐을 때 (처음 나타났을 때)
- 언 마운트 됐을 때 (사라질 때)
- 업데이트될 때 (특정 props가 바뀔 때)

[ ]로 설정하면 컴포넌트가 처음 나타날 때만 useEffect에 등록한 함수가 호출 한다.

useEffect 에서는 함수를 반환 할 수 있는데 이를 cleanup 함수라고 부른다.
cleanup 함수는 useEffect 에 대한 뒷정리를 해준다고 이해하면 되는데, [ ] 안에 내용이 비어 있는 경우에는 컴포넌트가 사라질 때 cleanup 함수가 호출된다.

//e.clientX, e.clientY
//브라우저에서 사용자에게 웹페이지가 보여지는 영역을 기준으로 좌표를 표시

함수형 업데이트
- 비동기적인 방법을 해결하기 위해서 우리는 함수형 업데이트(functional update)를 사용할 수 있다.
  즉 setState에 값을 그대로 전달하는 것이 아니라 함수를 전달하는 것이다.

실행하면 콘솔 창에 UseEffect는 왜 두 번 실행되는 걸까?

[해결 방법]
index.jsp
=> <React.StrictMode> 부분을 주석으로 처리

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

React의 Strict Mode ?
StrictMode는 리액트에서 제공하는 검사 도구이다.
개발 모드일 때 디버그를 통하여, 이 태그로 감싸져있는 App 컴포넌트와 자손까지 검사하는 것이다.
안전하지 않은 생명 주기를 가진 컴포넌트, 권장되지 않은 부분, 배포 후 문제가 될 수 있는 부분들까지 미리 확인하는 것이다.

creat-react-app으로 앱을 만들었기 때문에 기본적으로 생성되어 랜더링을 두 번이나 했었던 것이다.

데이터 읽기, 쓰기

웹 스토리지
HTML5에서 추가된 기술로 로컬스토리지와 세션스토리지로 구분된다.

특징
- 웹 스토리지는 Key와 Value 형태로 이루어졌다.
- 웹 스토리지는 클라이언트에 대한 정보를 저장한다.
- 웹 스토리지는 로컬에만 정보를 저장하고 쿠키는 서버와 로컬에 정보를 저장한다.

종류
로컬스토리지 (localStorage) - 클라이언트에 대한 정보를 영구적으로 저장
세션스토리지 (sessionStorage) - 세션 종료 시(브라우저 닫을 경우) 클라이언트에 대한 정보 삭제

장점
- 서버에 불필요하게 데이터를 저장하지 않는다. (백엔드에 절대로 전송되지 않는다.)
- 저장 가능한 데이터의 용량이 크다. (약 5Mb, 브라우저마다 차이 존재)

단점
- HTML5를 지원하지 않는 브라우저의 경우 사용 불가. (현재는 거의 없다고 봐야 한다.)

테스트용, 실사용시 비추천하는 기능

비동기 통신 - axios

서버에 새로고침 없이 요청할 수 있게 도와준다.
서버로 네트워크 요청을 보내고 응답을 받을 수 있도록 도와준다.

1. jQuery - $.ajax()
2. js - fetch()
   fetch() -> json 형식으로 가져온다.
3. 설치 - axios
   axios.get() -> object 형식으로 가져온다.

- 외부 API 비동기 통신을 위해서 fetch()를 이용한다.
- fetch()에 API 경로를 적어주면 promise가 반환된다.

fetch(url, [options])

fetch(url)
.then(콜백) - 응답 성공
.catch(콜백) - 응답 실패

axios.get(url)
    .then(콜백) - 응답 성공
    .catch(콜백) - 응답 실패

npm install axios / yarn add axioxyarn add axiox

useMemo

- useMemo는 컴포넌트의 성능을 최적화시킬 수 있는 대표적인 react hooks 중 하나이다.
- useMemo에서 Memo는 Memoization을 뜻한다.

memoization?

- 기존에 수행한 연산의 결괏값을 어딘가에 저장해 두고 동일한 입력이 들어오면 재활용하는 프로그래밍 기법을 말한다.

useReducer()

React에서 컴포넌트의 상태 관리를 위해서 useState를 사용해서 상태를 업데이트를 하는데, useReducer를 사용하게 되면 컴포넌트와 상태 업데이트 로직을 분리하여 컴포넌트 외부에서도 상태 관리를 할 수 있다.
즉, 현재 컴포넌트가 아닌 다른 곳에 state를 저장하고 싶을 때 유용하게 사용할 수 있다.

[사용법]

```
const [state, dispatch] = useReducer(reducer, initialState);
```

state : 현재 상태
dispatch : action을 발생시키는 함수
reducer : state와 action를 받아 새로운 state를 반환하는 함수
initialState : 초기값

React-Router

1. 리액트는 SPA (Single Page Application) 방식이다.
일반적으로 클라이언트가 요청을 하면 서버에서 요청한 페이지를 보여준다. 그러면서 로딩되는 모습이 보인다. 하지만 리액트는 모든 문서를 다 읽어드린 다음 클라이언트가 요청을 하면 로딩을 하지 않고 바로 보여주기 때문에 속도가 빠르다.

- 기존 웹 페이지처럼 여러 개의 페이지를 사용하며 새로운 페이지를 로드 하는 기존의 MPA 방식이 아니다.
- 새로운 페이지를 로드하지 않고 하나의 페이지 안에서 필요한 데이터만 가져오는 형태를 가진다.

2. 사용자가 입력한 주소를 감지하는 역할을 하며, 여러 환경에서 동작할 수 있도록 여러 종류의
라우터 컴포넌트를 제공한다.
이중 가장 많이 사용하는 라우터 컴포넌트는 BrowserRouter와 HashRouter이다.

[설치]
npm install react-router-dom
yarn add react-router-dom

 # react-router-dom 변경사항 (2021. 11. 25 기준)
1. Route 컴포넌트를 이제는 Routes 컴포넌트로 필히 감싸주어야 한다.
2. Route 컴포넌트의 매개변수 compent 가 element 로 바뀌었다.
3. useHistory 사라짐 -> useNavigate 함수
4. history.push('/') -> navigate('/')

Route : 어떤 경로로 들어왔을 때 어떤 컴포넌트를 보여 주겠다
Link : Router의 주소를 바꿈, a 태그지만 새로 고침이 안 된다.

: style 를 route path에 사용하면 useParams() 로 불러와 사용할 수 있다.
: 뒤에 나오는 부분이 params의 key 부분이 되어 :name 는 name가 key가 되어 불러오고 있다.

1. Link
- 클릭 시 바로 이동하는 로직 구현 시에 사용 용이
  ex) 상품 리스트에서 상세 페이지 이동 시
- react-router-dom 에서 제공하는 Link 컴포넌트는 DOM 에서 a 태그로 변환이 된다.
- a 태그와 Link 차이
  a : 외부 프로젝트로 이동하는 경우
  Link : 프로젝트 내에서 페이지 전환하는 경우

2. useNavigate
- useNavigate 훅을 실행하면 페이지 이동을 할 수 있게 해주는 함수를 반환한다.
  반환하는 함수를 navigate라는 변수에 저장 후 navigate의 인자로 설정한 path 값을 넘겨주면
  해당 경로로 이동할 수 있다.
- 페이지 전환 시 추가로 처리해야 하는 로직이 있으면 useNavigate 사용

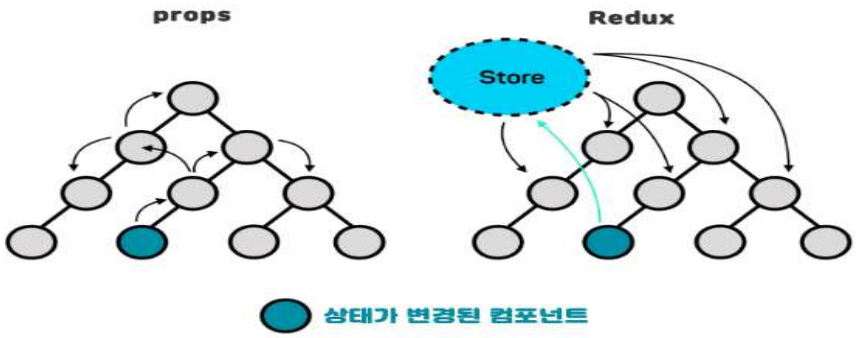
ex) 로그인 버튼 클릭 시
회원가입 되어 있는 사용자 -> Main 페이지로 이동
회원가입이 되어 있지 않은 사용자 -> SignUp 페이지로 이동

Index Routes

- Route 에 들어가는 index 라는 값은 default child routes 라고 생각하면 된다
- 부모에 여러 개의 자식 route 있는 경우 부모 경로에서 + '/' 인 경우 설정

Redux

- 리덕스를 사용하면 컴포넌트들의 상태 관련 로직들을 다른 파일들로 분리시켜서 더욱 효율적으로 관리할 수 있으며 글로벌 상태 관리도 손쉽게 할 수 있다.
- 상태값을 컴포넌트에 종속시키지 않고, 상태 관리를 컴포넌트의 바깥에서 관리 할 수 있게 된다.
- **Redux와 React는 독립적으로 사용될 수 있는 별개의 다른 라이브러리이다.**
- **Redux는 자바스크립트 어플리케이션에서 흔히 쓰이는 상태관리 라이브러리이다.**
- Redux는 Angular, Vue, Ember, jQuery 또는 Vanilla JavaScript와 같은 다른 라이브러리, 프레임워크에서도 사용할 수 있다



store

- 모두 한 곳에서 집중 관리
- 컴포넌트와는 별개로 스토어라는 공간이 있어서 그 스토어 안에 앱에서 필요한 상태를 담는다.
- 컴포넌트에서 상태 정보가 필요할 때 스토어에 접근한다.

action

- Action(액션)은 앱에서 스토어에 운반할 데이터를 말한다. (주문서)
- Action(액션)은 자바스크립트 객체 형식으로 되어있다.

reducer

- Action(액션)을 Store(스토어)에 바로 전달하는 것이 아니다.
- Action(액션)을 Reducer에 전달해야한다.
- Reducer가 주문을 보고 Store의 상태를 업데이트하는 것이다.
- Action을 Reducer에 전달하기 위해서는 dispatch() 메소드를 사용해야 한다.

① Action(액션) 객체가 dispatch() 메소드에 전달된다.
② dispatch(액션)를 통해 Reducer를 호출한다.
③ Reducer는 새로운 Store를 생성한다.

[설치]

yarn add react-redux (npm install react-redux / npm i react-redux)
yarn add redux ← 리덕스가 제대로 설치가 안 되면 또 한 번 한다.

yarn add redux-devtools-extension (npm i redux-devtools-extension)

createStore

앱의 상태 트리 전체를 보관하는 Redux 저장소를 만든다.
앱 내에는 단 하나의 저장소만 있어야 한다.
앱에 하나 이외의 저장소를 만들지 않는다.
-> 대신 여러 개의 리듀서를 하나의 루트 리듀서로 만들기 위해 combineReducers를 사용한다.

반환

앱의 전체 상태를 가지고 있는 객체이다.
이 객체의 상태를 바꾸는 유일한 방법은 액션을 보내는 것뿐이다.
UI를 업데이트하기 위해 상태를 구독 할 수도 있다.

Context

- context를 이용하면 단계마다 일일이 props를 넘겨주지 않고도 컴포넌트 트리 전체에 데이터를 제공할 수 있다.
- context는 React 컴포넌트 트리 안에서 전역적(global)이라고 볼 수 있는 데이터를 공유할 수 있도록 고안된 방법이다. 그러한 데이터로는 현재 로그인한 유저, 테마, 선호하는 언어 등이 있다.

- React 애플리케이션에서 데이터는 props를 통해서 부모에서 자식에게 전달되지만, 애플리케이션 안의 여러 컴포넌트들에게 props를 전달해줘야 하는 경우 context를 이용하면 명시적으로 props를 넘겨주지 않아도 값을 공유할 수 있게 해주는 것이다.

=> 데이터가 필요할 때마다 props를 통해 전달할 필요가 없이 context를 이용해 공유한다.

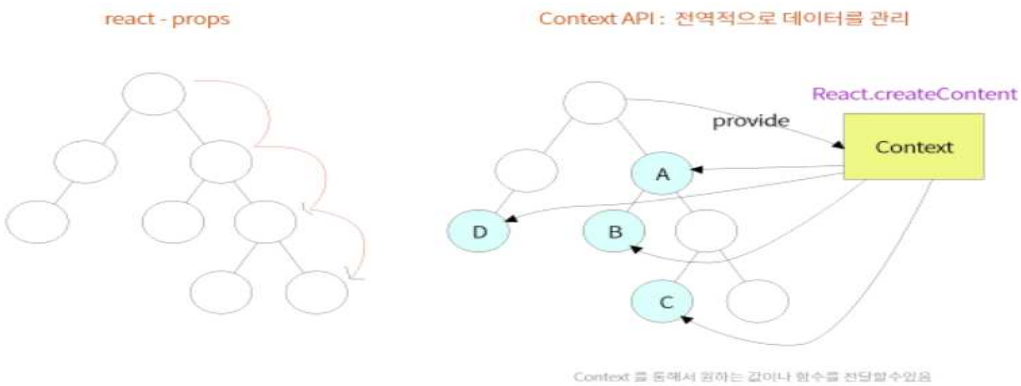
context API를 사용하기 위해서는 Provider, Consumer, createContext가 필요하다.

① **createContext** : context 객체를 생성한다.
createContext 함수 호출 시 Provider와 Consumer 컴포넌트 반환한다.
initialValue는 Provider를 사용하지 않았을 때 적용될 초기값을 의미한다.

② **Provider** : 생성한 context를 하위 컴포넌트에게 전달하는 역할을 한다.

③ **Consumer** : context의 변화를 감시하는 컴포넌트이다.
설정한 상태를 불러올 때 사용한다.

https://ko.reactjs.org/docs/context.html



useContext

- useContext 를 사용하면 기존의 Context 사용 방식보다 더 쉽고 간단하게 Context를 사용이 가능하고, 앞서 다뤘던 useState, useEffect와 조합해서 사용하기 쉽다는 장점이 있다.
- useContext를 사용할 때 주의해야 할 점은 Provider에서 제공한 value가 달라진다면 useContext를 사용하고 있는 모든 컴포넌트가 리렌더링 된다는 점이다. 따라서 useContext를 사용할 때 value 부분을 메모제이션 하는데 신경을 써야한다.

npm install react-router-dom / yarn add react-router-dom

npm install axios / yarn add axios