Create-react-app으로 시작하기-1

Create-react-app 과 next.js 의 가장 큰 차이는 서버사이드 렌더링의 지원 여부이다.

Create-react-app: 서버사이드 렌더링을 지원하지 않는다.

서버사이드 렌더링이 필수인 프로젝트라면 next.js를 선택하는게 좋다.

Cra의 단점은 cli나 eslint 등의 설정을 거의 변경할 수 없다는 점이다.

Next.js 에서는 바벨, 웹팩, eslint 등의 설정을 변경할 수도 있다.

Css를 변경했을 때 바로 적용이 된다.

이렇게 HMR이 동작하는 것은 npm start로 시작을 했을 때,

로컬에 서버를 띄워서 브라우저와 통신을 하기 때문에 가능한 것이다.

참고로 mpm start는 배포할 때 사용하면 안된다. 개발 모드에서만 사용해야 한다.

최적화가 전혀 되어 있지 않다.

배포할 때는 반드시 빌드 명령어를 사용해서 배포해야 한다.

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

여기서 React.StrictMode는 개발환경에서 동작을 한다.  
react에서 잘못 사용한 것을 빠르게 찾아내기 위해서 사용한다.

import logo from './logo.svg';

import './App.css';

function App() {

위에 보면 이미지 파일도 import를 해서 가져오고 있다.  
css나 이미지 파일을 자바스크립트에서 import해서 사용하는 것이 좋다.  
이렇게 사용하면 이미지 경로에 해시 값이 들어가기 때문에 브라우저 캐싱을 효율적으로 활용할 수 있다.

function App() {

  function onClick(){

    import('./data.json').then(({default:data})=>{

      console.log({data});

    })

  }

import를 해서 사용하면 좋은점이 또 있는데, 데이터의 사이즈가 클 경우에, 그리고 이 데이터가 항상 필요한게 아니라 특정 순간에 필요한 경우 그 때 데이터를 받아오는 것이 좋다.

Network 쪽에서 보면, 버튼을 클릭했을 때 동적으로 데이터를 받아오는 것을 알 수 있다.

Create-react-app으로 시작하기-2

  "scripts": {

    "start": "react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },

Npm start로 실행을 하게 되면 기본적으로 http로 시작하게 된다  
https로 시작하고 싶다면 HTTPS=true npm start 를 터미널에 입력하면 된다.  
윈도우용 명령어 🡪 set HTTPS=true && npm start ??? 이걸로 하면 https 가 안된다??

Build 명령어는 배포할 때 사용한다.  
빌드를 하게 되면 정적 파일이 생성된다. Build 폴더 생성됨  
서버에서는 생성된 빌드 폴더 안에 있는 내용들을 정적파일로 서비스 하면 된다.  
별도로 서버에 어플리케이션을 실행하지 않기 때문에 서버사이드 렌더링으로 동작할 수 없다.

Build 폴더 안에 있는 static/css 안에 css 파일이 있는데 기존 src에 있던 App.css, index.css가  
합쳐져 있다고 생각하면 된다.

"use strict";(self.webpackChunkcra\_test=self.webpackChunkcra\_test||[]).push([[468],{468:function(e){e.exports=JSON.parse('{"name":"lee","age":35}')}}]);

이전 시간에 동적 import를 사용해서 data를 받아왔었는데,  
동적 import를 사용하게 되면 해당 파일은 js파일로 만들어지게 된다.

이미지의 경우, 사이즈에 따라서 동작하는 방법이 다르다.  
이미지가 큰 경우, static/media 폴더 안에 들어가게 되고,  
이미지가 작을 경우, static/js 폴더 안에 있는 js 파일에 내장된다.  
이렇게 자바스크립트에 내장하는 한가지 이유는, http 요청 횟수를 줄이는 목적이다.  
그러나 http2.0 이후로는 요청이 많아도 성능 부분에서 그렇게 차이가 크지 않기 때문에 큰 의미는 없다.  
또 한가지 이유는 빠르게 이미지를 보여줄 수 있다.

Npm test를 실행하게 되면 원본 src에 App.test.js가 생성이 되고 test가 실행된다.  
여기서 App.spec.js로 변경해도 테스트 파일로 인식을 한다..  
src 폴더 안데 \_test\_ 파일을 만들고 그 안에다가 js을 넣으면 test나 spec 이외의 다른 이름이 와도  
테스트 파일로 인식한다.  
그러나 폴더에 넣어서 관리하는 것 보다는 .test.js 나 .spec.js로 관리하는게 편하다고 생각한다.  
원본 코드와 붙어있는게 편하다고 생각한다.

Eject 명령어는 react-script를 사용하지 않고 모든 설정 파일을 추출하는 명령이다.  
cra를 기반으로 직접 개발환경을 구축하고 싶을 때 사용한다.  
추출을 하지 ㅇ낳으면 cra의 기능이 추가 됏을 때 단순히 react-scripts 버전만 올리면 되는데  
추출을 하면 설정 파일을 수동으로 수정해야 한다는 단점이 있다.  
따라서 꼭 필요한 경우가 아니라면 관리측면에서 추출하지 않는게 좋다.

Cainiuse

브라우저별로 특정 기능에 대해서 지원 여부를 확인하기 좋다.  
ex) padStart 의 경우에는 IE에서는 지원을 하지 않는다.  
이 때, IE에서도 padStart를 사용하고 싶다면 polyfill 을 추가해야 한다.  
보통 polyfill 을 추가할 때는 core-js 라는 것을 많이 사용한다.  
core-js 깃헙에 들어가서 pad-start 검색하면 찾을 수 있는데, 그것을 index.js에 import 해서 사용하면 된다.  
원래는 core-js를 설치하고 사용해야 하지만, cra가 기본적으로 내장하고 있다.

환경변수는 개발,테스트 또는 배포 환경별로 다른 값을 적용할 때 유용하다.  
전달된 환경변수는 코드에서 process.env.{변수이름} 이런식으로 사용 할 수 있다.

// npm start 로 실행하면 development

// npm test 로 실행하면 test

// npm run build 로 실행하면 production

환경변수가 많아지면, .env 파일이라는 것으로 관리하는게 좋다.

Ex) env.development env.production

Npm start로 실행하면 env.development 변수가 출력되고

Npm run build 한뒤 npx serve -s build 를 실행하면 env.production 변수가 출력된다.