React Starter 02회

목 차

2018. 10. 23

1. React 개발환경 구성

- 리액트를 로컬에서 개발서버처럼 활용해서 세팅하는 방법
- CRA : create-react-app
- webpack 같은 환경구성툴이 필요없다.
- 개발에만 집중할수 있도록 개발환경구성방법은 숨기고 미리구성해놨다.

- Node 와 Yarn 을 설치후 아래 스크립트를 치면 자동구성됨
- npx create-react-app my-app cd my-app
- 로컬서버기동 스크립트는 yarn start
- 로컬서버중지는 컨트롤 C

- NPX
  an npm package runner
  npm 5.2.0 이후 버전을 설치하면 npx 라는 새로운 바이너리가 설치됩니다.
  npx는 npm의 패키지 사용에 도움이되는 도구입니다.

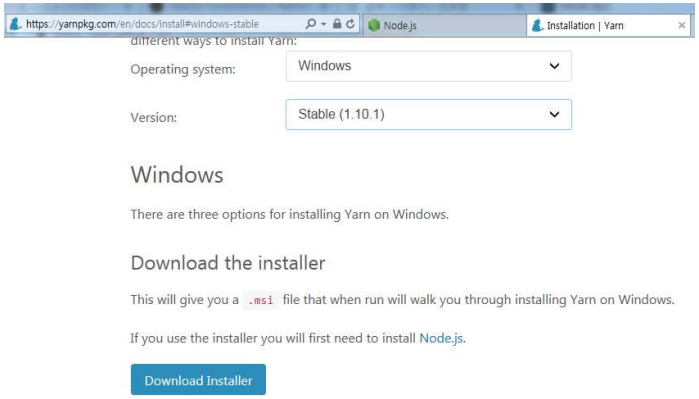
1. React 개발환경 구성

Node.js : 리액트 프로젝트를 준비하기 위해 필요한 webpack, babel 등의 도구들을 실행하는데에 사용됩니다.

Yarn : 자바스크립트 패키지를 관리하기 위해서 사용됩니다.
Node.js 를 설치하면 npm 이 설치되어서 npm 으로 해도 되긴 하지만,
yarn을 사용하면 훨씬 빠릅니다.

Node.js LTS 로 설치
https://nodejs.org/ko/

Yarn
https://yarnpkg.com/en/docs/install#windows-tab



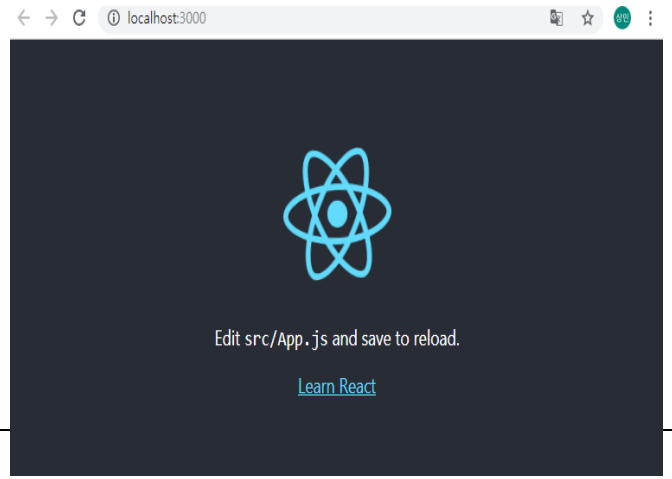
–

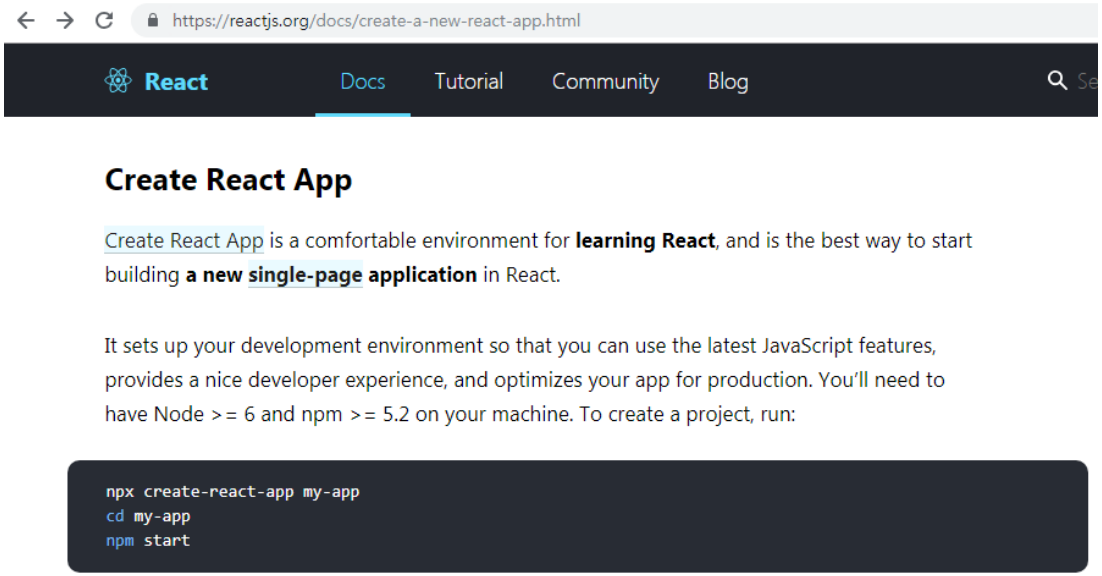
1. React 개발환경 구성

- 리액트 개발환경 만들기 (페이스북 개발팀에서 간단하게 해놨네여)
- npx create-react-app firstPrj

```
C:\study\react\day01>npx create-react-app firstprj
npx: installed 63 in 4.673s

Creating a new React app in C:\study\react\day01\firstprj.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

[ ..................] / fetchMetadata: sill resolveWithNewModule browser-process-hrtime@0.1.3 checking installable status
```

- Cd firstprj
- yarn start
- 개발서버 끌때는 컨트롤C





1. React 개발환경 구성

- CRA 폴더 살펴보기

```
my-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── serviceWorker.js
```

- 아무설정 안해도 위처럼 폴더구조와 SPA형태를 위해 파일들이 자동생성
- 코드가 수정되면 서버재기동 없이 자동 반영됨.

1. React 개발환경 구성

- Index.js 모두 깨끗이 지우고 본인의 헬로우월드 개발서버환경에 만들어 보겠습니다.

- 필요한 라이브러리 두개 임포트 합니다.
import React from 'react'
import ReactDOM from 'react-dom'

- 그리고 const 변수 displayResult 선언하고 JSX로 <h1> hello React 개발서버 </h1>
- 그리고 ReactDOM.render()
- root가 같은 파일 아니라서 직접지정해줘야 하네여. document.getElementById('root')

```
1  import React from 'react'
2  import ReactDOM from 'react-dom'
3
4  //React 임포트한걸로 JSX를 실행
5  const displayResult = <h1>Hello React 개발환경 !!!</h1>
6
7  //디스플레이컨텐츠를 virtualDOM에 렌더링
8  ReactDOM.render(displayResult, document.getElementById('root') )
9
```

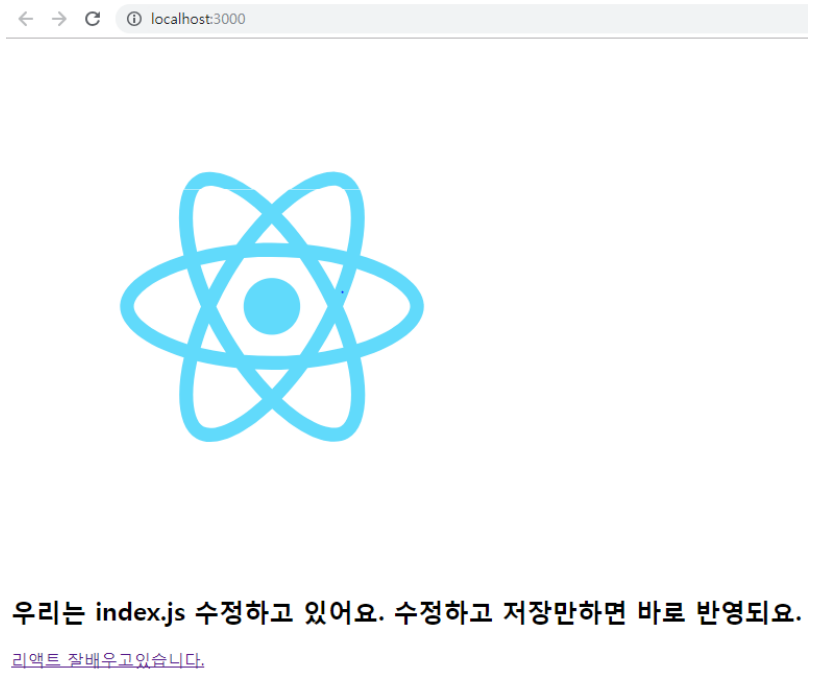
1. React 개발환경 구성

- Index.js 한번 더 연습해봅니다.

- 이미지하나 띄우고 리소스를 임포트해서

```
import logo from './logo.svg'
```

- h2 태그에 우리는 index.js 수정하고 있어요 수정만하면 바로 반영되요
- a 링크에 react.org 연결해봅니다. 리액트 잘배우고있습니다.



-

1. Export Import ES6 모듈시스템

- ES6에서 바뀐것중 획기적인 것중 하나라고 합니다.
- 모듈시스템이라고 하네여.
- 의존성관리가 이전 자바스크립트에서 가장 어려운 문제였다고 하네여.
- 파일첫부분에 필요한 파일을 선언해주는 방식으로 변경 예전에는 require()로 사용
- 선언한 패키지만 사용하는것이죠 import 입니다.

- 모듈로 사용하기 위해서는 펑션(컴포넌트) 앞에 export를 붙혀줍니다.
-
- 간단하게 코드로 보겠습니다.
- 매우간단한 컴포넌트 하나 만들고 이것을 export해주도록 하겠습니다.

- 인자 두개 받아서 더하기를 하는 펑션과
- 상수 하나를 갖는 컴포넌트 입니다. (compo_test.js)
- 컴포넌트의 메소드와 상수앞에
- export 명령어를 추가해주고 저장합니다.

```
function performPlus (a, b) {
```

```
const displaySymbol = "+"
```

- 이젠 index.js에서 compo_test.js를 import해서 사용해봅니다.
- import 중괄호열고 사용할 메소드와 상수를 기록하고 from ‘./해당파일’

1. Export Import ES6 모듈시스템

- <h2>
- <h3>

```
<h2>1 plus 10 ===> { performPlus(1,10) }</h2>
<h3>1 {displaySymbol} 10 ===> { performPlus(1,10) }
```

-

1. Export Import ES6 모듈시스템

- <h2>
- <h3>

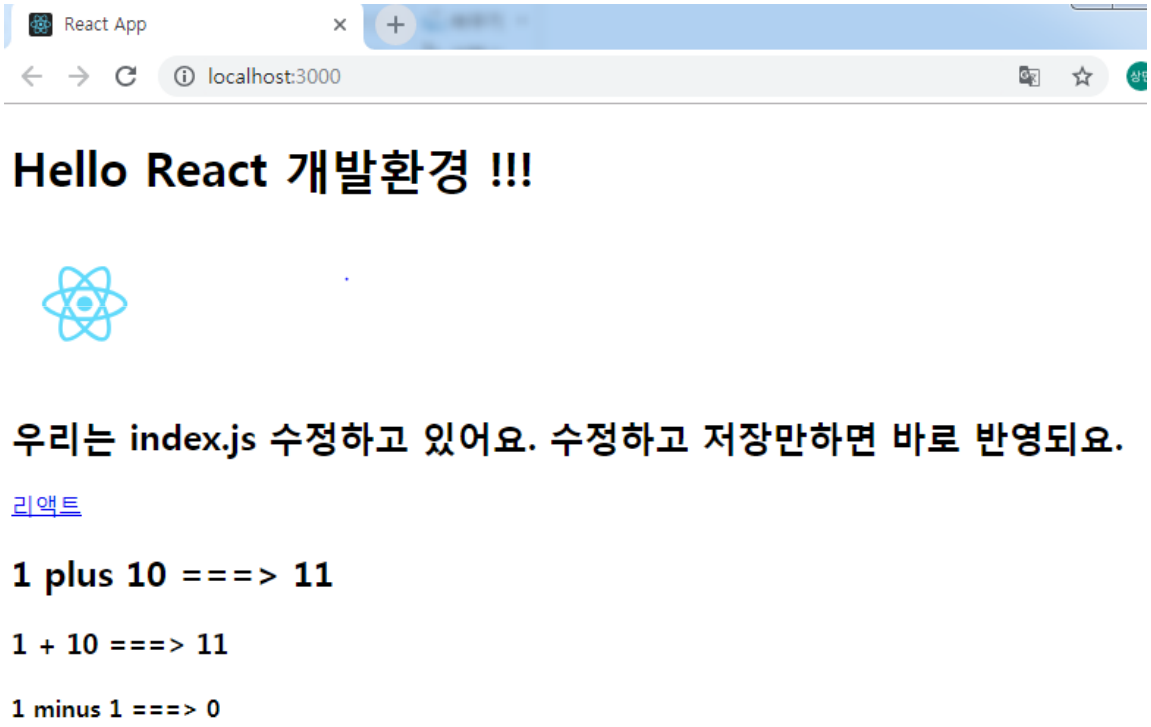
```
<h2>1 plus 10 ===> { performPlus(1,10) }</h2>
<h3>1 {displaySymbol} 10 ===> { performPlus(1,10) }
```

-

1. React 개발환경 구성

- import perform from ‘./compo_test’
- 위처럼 쓰려면
- export default function peformMinus 로 함수를 지정하면 됩니다.

-



2. JSX up

```
const element = <h1>Hello, world!</h1>;
```

This funny tag syntax is neither a string nor HTML.

- Javscript Syntax extension
- JSX 는 리액트 element를 만들어내는 것이다.
- react.org에서도 funny tag라고 하네여.

- 중괄호를 이용하면 어떠한 자바스크립트 표현도 JSX에 포함시킬수 있다.

- JSX표현식을 변수에 할당하거나 함수의 인자로 사용하거나 리턴값으로 사용 가능

- JSX 태그내용이 비면 />로 닫아 주어야 한다. <img
- JSX 자식을 포함할 수 있다.
- JSX 사용자 입력은 안전하다. DOM에 렌더링 전에 escaping 처리한다. XSS방지

- 예제코딩) 객체를 매개변수로 하는 함수를 호출하는 JSX를 만들어봅시다
- 우리가 코딩해볼 함수는
- 유저객체를 매개변수 받아서 성과 이름을 h1 태그로 rendering

2.JSX up – cheat sheet

-이번에는 컴포넌트에 객체를 한번 전달해보죠..

-유저인포 객체는 firstName과 lastName을 키로 갖습니다.

```
const user = {
    firstName: '클레이튼',
    lastName: '커쇼'
};
```

- 함수이름과 매개변수 printFullName(user)
- 리턴은 성과 이름을 한칸띄어서

- 리액트 엘리먼트는 <H1> hello 함수이름(매개변수)

```
<h5>Hello React 개발환경 !!! {printFullName(user)} </h5>
```

2. ES6 배열

- const arrPrac01 = [1,2,3,4]
- 배열을 대괄호로 표시함

- 배열의 길이는 length로 확인 arrPrac01.length

- 배열 뒤쪽에 value 추가는 concat() 사용
- arrPrac01.concat([5,6])
-
- 배열 뒤쪽에 한건 삽입시에는 push(항목), 빨경우에는 pop()
- arrPrac01.push(7)
- arrPrac01.push(8)
- arrprac01.pop()

- 배열을 조작한후 새로운 배열에 나타낼 경우 map(조작할함수)
- const arrPrac03 = arrPrac01.map(function(x) { x+1 })

- 배열중 특정조건에 맞는것만 새로운 배열에 나타낼 경우 filter(조작할함수)
- const arrPrac05 = arrPrac01.filter(function(x) { x % 2 ===0 })

2. ES6 Object

- 데이터와 그 데이터에 관련되는 동작(절차,방법,기능)을 모두 포함하고 있는 개념
- 자바스크립트에서 표현은 { } 를 사용
- const objPrac01 = { key: “value”……. }

- 객체는 프로퍼티와 메소드로 구성됨
- 메소드와 함수의 차이 (일련의동작을 수행한다는 유사점)
- 메소드는 객체에 속해있으므로 객체가 수행의 주체
- 함수는 객체와 상관없는 주체적인 함수객체

- object 키와 밸류를 변수로 받는 예제

```
var person = {
  name : "neko",
  gender : "male"
};

console.log( person.name, person.gender );
// 결과 : neko male

var key1 = 'name', key2 = 'gender';

console.log( person[key1], person[key2] );
// 결과 : neko male
// 위와 동일한 결과를 얻을 수 있습니다.
```

- object 키와 밸류 찍는 예제

```
var object = { ... };

for( var key in object ) {
  console.log( key + '=>' + object[key] );
}
```

- obejct 스프레드로 복사하는 예제

2.JSX – 조금더 Javascript Syntax eXtension

- JSX는기본적으로 단지 React.createElement 함수에 대한 트랜스파일링

- JSX내에서 리액트컴포넌트는 점표시로 참조될 수 있다.(같은스콥내에서)
- const 객체생성시 키 말고 값을 함수로 정의해본후
- 점표시로 참조해보자

```
 const MyCompo = {
    key1 : function getKeyValue1(props) {
              return <div> Imagine a {props.color} here </div>
            }
}
```

렌더링추가

```
<MyCompo.key1 color="blue" />
```

- 사용자 정의 컴포넌트는 대문자로 시작해야함
- 소문자인경우 경고가 뜨거나
- 그냥 문자처럼 출력됨
- Index.js에서 function div 만들어서 호출
- Function hello 만들어서 호출
- Function Div 만들어서 호출해보자

2. webpack

- 리액트로 개발할 경우 다양한 컴포넌트를 만들어서 export하고
- 필요한 컴포넌트를 import하여 최종 컴포넌트를 만들게 된다.
- 이런 개발된 소스를 웹팩이 중간에서 모아서 하나의 파일로 만들어 준다.
- 우선 크게 webpack에게 알려줘야 할 세가지 정보가 있습니다.

어플리케이션의 시작 포인트, 또는 root 역활을 하는 자바스크립트 파일
어떤 방법으로 코드를 변경할것인가
어떤 위치로 변경된 코드를 위치시킬것인가

- webpack에 대한 코드는 webpack.config.js 파일에 작성을 하게 됩니다
- entry 시작지점
- module 은 로딩한 모듈과 제외할 모듈
- output 파일합쳐서 나올 이름과 경로

- yarn eject 해서 볼수 있음.

```
module.export = {
    entry: [
        './app/index.js'
    ],
    module: {
        loaders: [
            { test: /\.coffee$/, exclude: /node_module/, loader: "coffee-loader" }
        ]
    },
    output: {
        filename: "index_bundle.js",
        path: __dirname + '/dist'
    }
};
```

3. 요약 - 리액트를 다루는 기술 책 중심 JSX, 컴포넌트

- 1. CRA로 생성된 프로젝트의 app.js 확인
- import React, { Component } from 'react'
- 예전자바스크립트 방식으로는 const React = require('react')
- const Component = React.Component

- 번들링 : 파일을 사용하기 최적화하도록 묶는것
- 번들링도구가 webpack : 편의성과 확장성이 뛰어나 이걸루 천하통일로 되고있음.
- 파일들을 불러오는 것도 함 : 각종로더가 담당
- css-loader, file-loader, babel-loader(js불러오고 ES6은 ES5로 트랜스파일링)

- 2. class App extends Component {
- 앱 클래스 선언하고 상속받는것 ES6에서 도입됨
- 예전에는 클래스 없고 prototype문법사용

- 3. render()
- 유저에게 html 방식으로 보이도록 처리하는것
- JSX방식 사용하면 좀 편함

3. 요약 - 리액트를 다루는 기술 책 중심 JSX, 컴포넌트

- JSX 장점
- 보기쉽고 좀 익숙하다
- 오류검사가 편리함(태그 잘 닫아야함)
- 활용도가 높다 html태그 비슷하므로..

- JSX 문법
- 최상위 항상 하나로 되야함 fragment로 사용가능
- 자바스크립트에서 중괄호 {}로 변수 표현
- 조건연산자는 삼항연산자 사용 { condition ? '참' : '거짓 ' }
- 조건부 렌더링 if 문만 있다고 생각하면 될듯 { condition && '참이면할것' }
- 인라인스타일링 : 스타일을 class내부에 객체로 선언하고 적용 대시안됨 카멜케이스
- class 예약어이므로 스타일링시 className으로 사용
- 태그 꼭 닫아야 한다. <img /> <input />
- 주석은 {/* 블라블라 */} 그냥// 쓰면 문자로 렌더링되니 주의

- 3. 컴포넌트
- 파일생성하고 export import하여 확인해봄
- 기타 snippet 설치하고 rcc 확인
- props 기본값설정하기, propType 설정하기,

4. 무비카드 컴포넌트 생성하는 연습

- 함수형 컴포넌트 생성하여 렌더링 해봅니다.
- 영화목록을 만든다고 생각하고 컴포넌트 설계해봅니다.

- 영화제목, 주연배우, 영화설명 항목으로 간단하게 해봅니다.
- MovieCard 태그 생성

4. 무비카드 배열과 for문으로 연습

_

4. 무비카드 이미지 추가

–

5. VSC 디버그

- 비주얼 스튜디오 코드에서 디버그
- 익스텐션탭에Debugger for Chrome 설치하고 리로드한후 vsc 종료후 재기동
- index.js에서 브레이크 포인트 찍은후
- 크롬디버거 세팅 디버그뷰에서 open launch.json 클릭하고
- 포트를 현재기동포트인 3000으로 변경후 저장
- yarn start 후 디버그시작 F5 , 넥스트 F10, 디버그종료 쉬프트 F5

- 카운터 만들어서 디버깅 실습
- https://gist.github.com/velopert/5ead29e052d715bae9cf47940c6f97b9

6. Etc - ESLint

- ESLint는 ES + Lint입니다.
- ES는 EcmaScript, 즉 자바스크립트를 의미하는 것
- Lint는 보푸라기라는 뜻인데 프로그래밍 쪽에서는 에러가 있는 코드에 표시를 하는것
- 즉 ESLint는 자바스크립트 문법 중 에러가 있는 곳에 표시를 달아주는 확장팩

- .eslintrc파일 작성하고 여기다 설정내역을 json 형식으로 기록

```
{
"parser": "babel-eslint",
"extends": "airbnb",
"plugins": ["react", "import"],
"rules": {
"semi": 0
}
}
```

7. **Etc –** Reactjs code snippets

- rsc, rcc
- 함수형컴포넌트
- 클래스형컴포넌트