AI 활용 빅데이터분석
풀스택웹서비스 SW 개발자 양성과정

React

부산대학교 소프트웨어교육센터
PUSAN NATIONAL UNIVERSITY SOFTWARE EDUCATION CENTER

리액트 (React)

- 사용자 인터페이스를 만들기 위한 JavaScript 라이브러리
  - 컴포넌트(component) : 사용자가 정의한 태그
  - 선언형
    - 데이터가 변경됨에 따라 적절한 컴포넌트만 효율적으로 갱신하고 렌더링
  - 컴포넌트 기반
    - 스스로 상태를 관리하는 캡슐화된 컴포넌트를 조합해 복잡한 **UI** 생성
  - 한 번 배워서 어디서나 사용
    - **Node** 서버에서 렌더링을 할 수도 있고, **React Native**를 이용하면 모바일 앱 개발

리액트 (React)

- SPA(Single-page application)
  - 하나의 HTML 페이지에 애플리케이션 실행에 필요한 JavaScript와 CSS 같은 모든 자산을 로드하는 애플리케이션
  - 웹 사이트 전체 페이지를 하나의 페이지에 담아 동적으로 화면을 바꿔가며 표현하는 것
    - 페이지 또는 후속 페이지의 상호작용은 서버로부터 새로운 페이지를 불러오지 않으므로 페이지가 다시 로드되지 않음
- React를 사용하여 싱글 페이지 애플리케이션을 만들 수 있지만, 필수 사항은 아님

웹 사이트에 React 추가

- 기존 페이지에 리액트 추가하는 방법

```html
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8" />
  <title>Hello World</title>
  <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

  <!-- Don't use this in production: -->
  <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
</head>

<body>
  <div id="root"></div>

  <script type="text/babel">
    class Hello extends React.Component {
      render() {
        return React.createElement('h1', null, `Hello ${this.props.toWhat}`);
      }
    }

    const root = ReactDOM.createRoot(document.getElementById('root'));
    root.render(React.createElement(Hello, { toWhat: 'World' }, null));
  </script>
</body>
</html>
```

2단계: 스크립트 태그 추가하기(React를 실행 및 만든 컴포넌트)

1단계: HTML 파일에 DOM 컨테이너 설치

3단계: React 컴포넌트 만들기

웹 사이트에 React 추가

- 기존 페이지에 리액트 추가하는 방법(JSX로 추가)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Hello World</title>
    <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

    <!-- Don't use this in production: -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>
    <div id="root"></div>
    <script type="text/babel">

      function MyApp() {
        return <h1>Hello, world!</h1>;
      }

      const container = document.getElementById('root');
      const root = ReactDOM.createRoot(container);
      root.render(<MyApp />);

    </script>
  </body>
</html>
```

JSX(JavaScript XML)

- 자바스크립트에서 XML을 추가한 확장형 문법
  - React “엘리먼트(element)” 를 생성
  - JSX 사용이 필수가 아니지만, 대부분의 사람은 JavaScript 코드 안에서 UI 관련 작업을 할 때 시각적으로 더 도움이 됨
- 표현식 포함
  - JSX 중괄호 안에 유효한 모든 JavaScript 표현식 작성가능
  - const element = ⟨h1⟩Hello, {name}⟨/h1⟩;
- 주입 공격을 방지
  - React DOM은 JSX에 삽입된 모든 값을 렌더링하기 전에 이스케이프 하므로, 애플리케이션에서 명시적으로 작성되지 않은 내용은 주입되지 않음으로 XSS (cross-site-scripting) 공격을 방지
- 객체를 표현
  - Babel은 JSX를 React.createElement() 호출로 컴파일

새로운 React 앱 만들기

- Create React App
  - 새로운 싱글 페이지 애플리케이션으로 React의 간편한 환경을 제공
  - 개발 환경을 설정하고, 최신 JavaScript를 사용하게 해주며, 좋은 개발 경험과 프로덕션 앱 최적화
- 설정
  - Node 설치 (Node 14.0.0 이상 , npm 5.6 이상 버전이 필요)
    - https://nodejs.org/en/

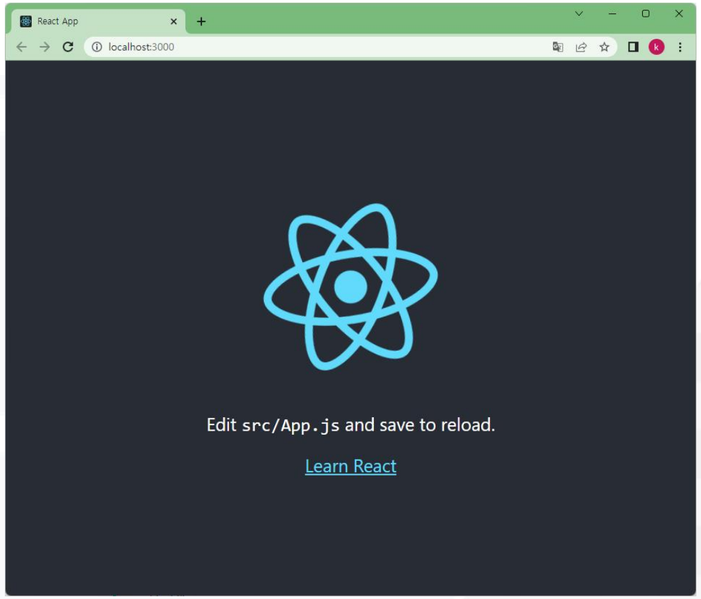
새로운 React 앱 만들기

```
PS C:\minNote\OneDrive - pusan.ac.kr\강의자료\2022_0919_K디지털\React\work> npx create-react-app ./my01

Creating a new React app in C:\minNote\OneDrive - pusan.ac.kr\강의자료\2022_0919_K디지털\React\work\my01.

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


added 1394 packages in 1m

210 packages are looking for funding
  run `npm fund` for details

Initialized a git repository.

Installing template dependencies using npm...

added 56 packages in 9s

210 packages are looking for funding
  run `npm fund` for details
  npm audit fix --force

Run `npm audit` for details.

Created git commit.

Success! Created my01 at C:\minNote\OneDrive - pusan.ac.kr\강의자료\2022_0919_K디지털\React\work\my01
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

    cd my01
    npm start
```

```
Creating a new React app in C:\minNote\OneDrive - pusan.ac.kr\강의자료\2022_0919_K디지털\React\work\my01.

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


added 1394 packages in 1m

210 packages are looking for funding
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
Compiled successfully!

You can now view my01 in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.137.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

프로젝트명은 소문자로 작성

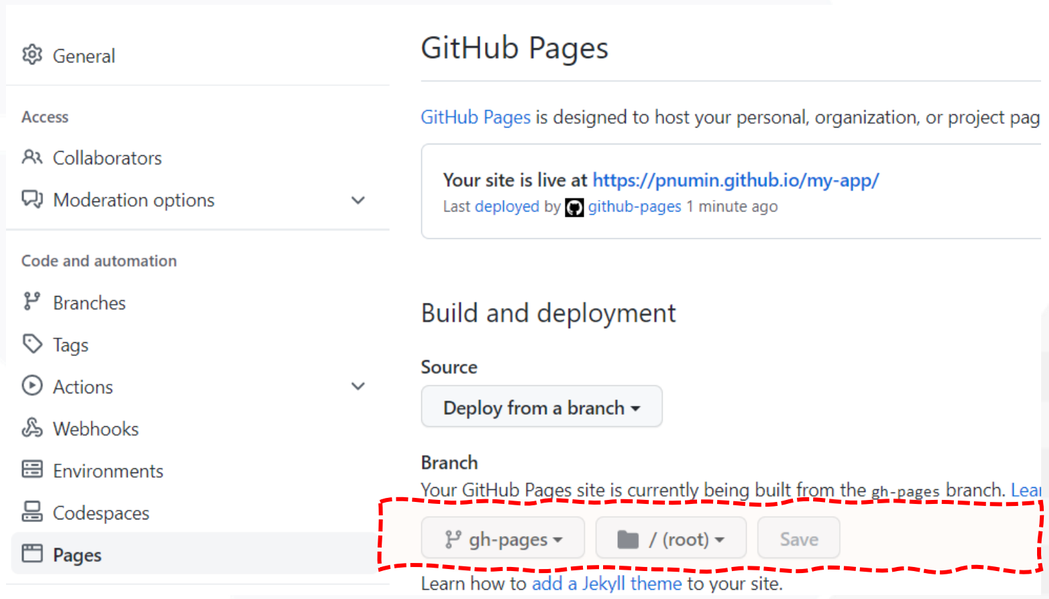
```
npx create-react-app my-app
cd my-app
npm start
```



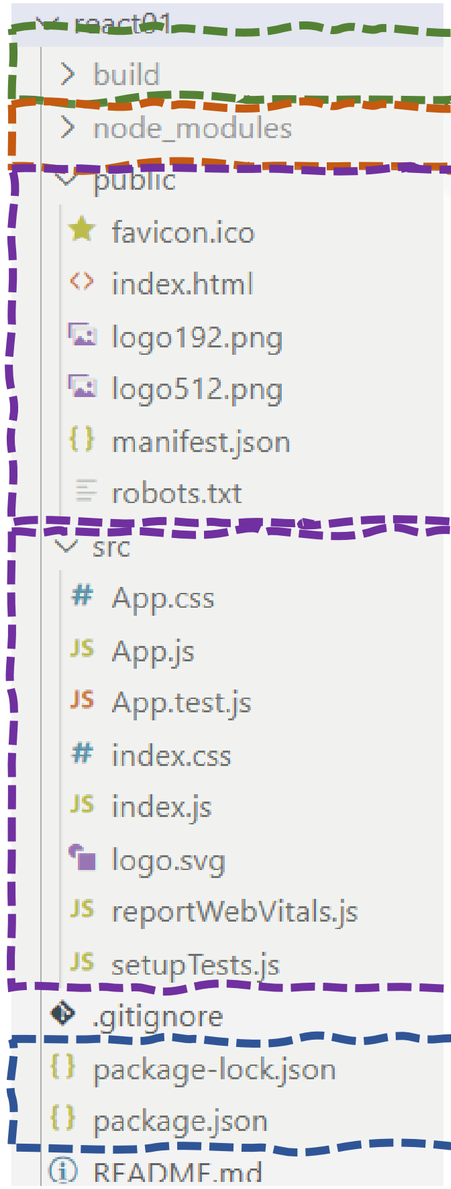
깃허브에 배포하기

1. 터미널을 띄워 새로 생성된 로컬 컴퓨터 디렉토리로 이동
   - cd my01
2. 깃허브에 새로운 레포지토리 생성
3. 원격지 레포지토리와 현재 프로젝트 연결하고 push
   - git remote add origin 새로생성한레포지토리주소
   - git push -u origin master
4. 로컬 컴퓨터 React 프로젝트에 gh-pages 설치
   - npm install gh-pages --save-dev
5. package.json 파일에 홈페이지 주소 설정
   - 스크립트에 추가
     "predeploy": "npm run build",
     "deploy": "gh-pages -d build"
   - "homepage": " 깃허브주소",
6. gh-pages 배포하고 레포지토리에 다시 설정
   - npm run deploy

```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject",
  "predeploy": "npm run build",
  "deploy": "gh-pages -d build"
},
"homepage": "https://pnumin.github.io/my-app",
```



create-react-app 프로젝트 폴더



- 빌드를 한 경우에 생성 : **npm run build** 후 **npx serve -s build**로 실행
- **node_modules: React** 앱을 실행하기 위해 설치한 **Node** 모듈 저장
- **public:** 웹 브라우저에 실제로 보이는 정적 파일**(static files)** 저장
- **src:** 모든 컴포넌트**(components)** 파일 저장**; UI** 조각 저장
- **package-lock.json: npm**이 **node_modules** 또는 **package.json**을 수정할 때 디펜던시 버전 정보(버전명) 저장
- **package.json:** 프로젝트에 대한 메타데이터**(metadata)** 및 디펜던시 버전 정보(범위) 저장

public > index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

React의 결과물이 들어가는 부분으로
src 폴더의 index.js파일을 수정

- 웹 브라우저가 렌더링하는 HTML 문서
  - 이 문서가 웹 브라우저에서 렌더링된 결과물이 바로 SPA에서 말하는 단 1개의 페이지

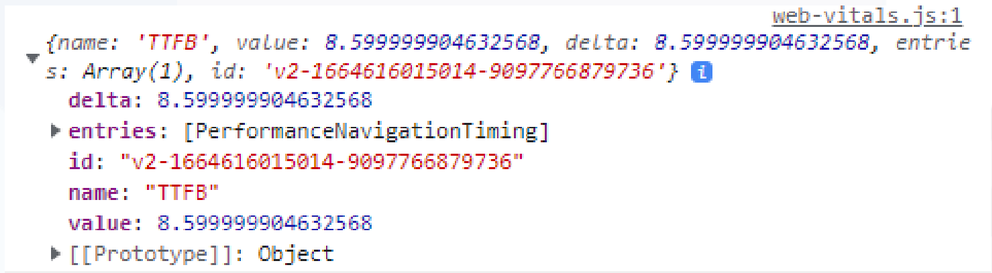
src > index.js

```
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>
);

//measuring performance in your app
reportWebVitals(console.log);
```

- **src 폴더**
  - 만든 모든 컴포넌트, 즉 UI 조각들이 저장되는 곳
- **index.js**
  - React앱의 진입점

reportWebVitals 파일에서 가져온 함수를 실행

- **console.log**를 넣어주면 개발창으로 앱의 퍼포먼스시간들을 분석하여 객체 형태로 보여줌
- **metric(측정도구) 이름**
  name: 'CLS' | 'FCP' | 'FID' | 'LCP' | 'TTFB';
- 측정된 현재값 (값이 작을수록 빠른성능을 가짐)
  value: number;
- 현재 측정값**(current value)**과 최신 측정값**(last-reported value)** 차이
  delta: number;
- 특정 측정도구를 나타내는 유니크한 **ID** 값으로 중복되는 값들을 관리
  id: string;
- 계산된 측정값들의 내용들이 배열로 나열
  entries: (PerformanceEntry | FirstInputPolyfillEntry | NavigationTimingPolyfillEntry)[];



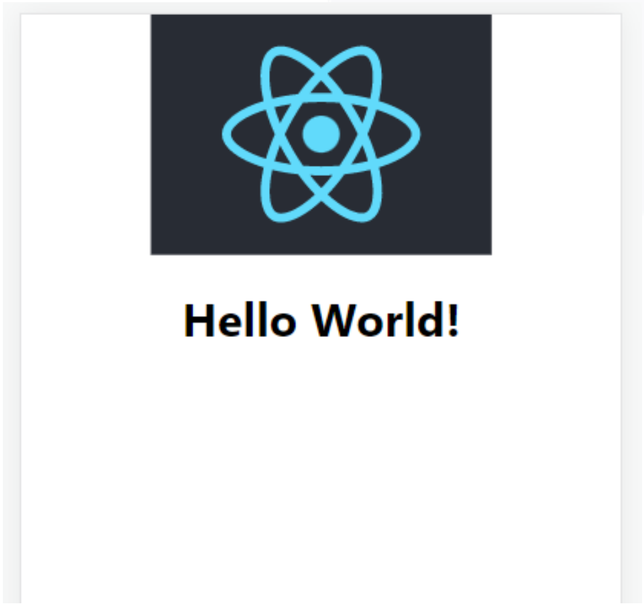
src > App.js

```jsx
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div>
      <div className="AppH1"><img src={logo} className="App-logo2" alt="logo" /></div>
      <h1 className="AppH1">Hello World!</h1>
    </div>
  );
}

export default App;
```

```jsx
<>
  <div className="AppH1"><img src={logo} className="App-logo2" alt="logo" /></div>
  <h1 className="AppH1">Hello React!!</h1>
</>
```

- **직접 만들 컴포넌트들이 모인 루트 컴포넌트**
  - 코드 블록에 사용된 문법은 JSX
    - **JSX**는 웹 브라우저에 보일 **UI** 조각인 **html** 요소를 반환(**return**)
  - JSX 컴포넌트는 **한 개의 부모 요소(parent elements)만 반환**
    - 여러 개의 요소를 반환하고 싶다면 **fragments**라 불리는 <></> 를 사용
  - JSX에서는 class 속성 대신 className을 사용

실행 결과



Elements Console Sources Performance Memory Application Security

```
<!DOCTYPE html>
<html lang="ko">
▼<head>
    <meta charset="utf-8">
    <link rel="icon" href="/favicon.ico">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="theme-color" content="#000000">
    <meta name="description" content="Web site created using create-react-app">
    <link rel="apple-touch-icon" href="/logo192.png">
    <link rel="manifest" href="/manifest.json">
    <title>React App</title>
    <script defer src="/static/js/bundle.js"></script>
  ▶<style>…</style>
  ▶<style>…</style>
  </head>
▼<body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
  ▼<div id="root">
    ▼<div>
      ▶<div class="AppH1">…</div> flex
        <h1 class="AppH1">Hello World!</h1> flex
      </div>
    </div>
  </body>
</html> == $0
```

시계 앱 만들기

./src/components/Tick.jsx

```jsx
import React from 'react';

const Tick = () => {
  return (
    <h2  className="AppH1">현재 시간 : {new Date().toLocaleTimeString()} </h2>
  )
}

export default Tick;
```

./src/App.js

```jsx
import logo from './logo.svg';
import './App.css';
import Tick from './components/Tick';

function App() {
  return (
    <div>
      <div className="AppH1"><img src={logo} className="App-logo2" alt="logo" /></div>
      <h1 className="AppH1">Hello React!</h1>
      <Tick></Tick>
    </div>
  );
}

export default App;
```

./src/index.js

```jsx
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'));
const mytick = () => {
  root.render(
    <React.StrictMode>
      <App />
    </React.StrictMode>
  );
}

setInterval(mytick, 1000);
//measuring performance in your app
reportWebVitals();
```