

Create React App



WITH
YAM009
DERESA KIM





Create React App · Set up a mod

+


← → ↺

create-react-app.dev

☆ b 📷 ⌘ 👤 ⋮

Create React App

Docs Help GitHub 🌙 🔍 Search



Create React App

Set up a modern web app by running one command.

Get Started

Less to Learn

You don't need to learn and configure many build tools. Instant reloads help you focus on development. When it's time to deploy, your bundles are optimized automatically.

Only One Dependency

Your app only needs one build dependency. We test Create React App to make sure that all of its underlying pieces work together seamlessly – no complicated version mismatches.

No Lock-In

Under the hood, we use Webpack, Babel, ESLint, and other amazing projects to power your app. If you ever want an advanced configuration, you can "eject" from Create React App and edit their config files directly.





React 프로젝트 생성 (with CRA)

```
yamoo9@yamoo9ui-MacBookPro: ~/Documents/EUID/React Framework/hands-on/HTML,CSS,JS-to-React
→ HTML,CSS,JS-to-React git:(master) x npx create-react-app ediya-ui
```

```
yamoo9@yamoo9ui-MacBookPro: ~/Documents/EUID/React Framework/hands-on/ediya-ui
→ ediya-ui npm i react-scripts@latest
```

```
+ react-scripts@3.3.1
removed 6 packages, updated 3 packages and audited 918360 packages in 30.857s
found 0 vulnerabilities
```

NPX 명령 사용 권장

create-react-app을 전역(global)에 설치해 사용하는 것보다 최신 버전을 바로 사용할 수 있도록 npx 명령을 사용하세요. (npm v5.2+ 이상)

유지 보수 용이

빌드 도구 업데이트는 일반적으로 어렵고 시간이 많이 걸리지만, CRA는 명령어 한 줄로 업그레이드가 가능합니다.





React 프로젝트 템플릿

CRA 내장 기본 템플릿

- ◆ `cra-template` (기본 값)
- ◆ `cra-template-typescript` (`--template` 사용)

CRA 외부 템플릿

- ◆ `cra-template-*` (`--template` 사용)

CRA 커스텀 템플릿

- ◆ `cra-template-ko` (Sass 추가, 한글 번역)

매번 기본 템플릿을 사용해 프로젝트를 생성하는 대신, 입맛에 맞는 필요한 구성을 설계하여 재사용 할 수 있습니다.



The screenshot shows the npmjs.com page for the package `cra-template-ko`. The page header includes the package name, version 1.0.1, and publication status. Below the header, there are tabs for Readme, Explore (BETA), Dependencies (0), Dependents (0), Versions (2), and Settings. The main content area displays the package name `cra-template-ko` and a description in Korean: "이 프로젝트는 Create React App의 템플릿에 따라 제작된 한국어 cra-template-ko 입니다." To the right of the description is an "Install" button with a terminal snippet: `> npm i cra-template-ko`. Below the description is a terminal window showing the command: `~ npx create-react-app <프로젝트_이름> --template cra-template-ko`. On the right side of the page, there is a sidebar with metadata: License (MIT), Total Files (22), Pull Requests (0), and a list of links for Homepage and Repository, both pointing to `github.com/yamoo9/cra-template-ko`.





React 앱 브라우저 호환성

모든 최신 브라우저

- ◆ Chrome
- ◆ Edge
- ◆ Firefox
- ◆ Safari
- ◆ Opera

Polyfill이 필요한 브라우저

- ◆ IE 9 - 11

```
yamoo9@yamoo9ui-MacBookPro: ~  
→ ~ npm i react-app-polyfill
```

IE 브라우저 지원

- ◆ Promise 또는 Async / Await
- ◆ Fetch API
- ◆ Object.assign ({...object} 사용 시, 필요)
- ◆ Symbol (for - of 사용 시, 필요)
- ◆ Array.from ([...array] 사용 시, 필요)

```
src > JS index.js  
1 import 'react-app-polyfill/ie9'; 56.5K (gzipped: 17.5K)  
2
```

```
src > JS index.js  
1 import 'react-app-polyfill/ie11'; 36.3K (gzipped: 12.5K)  
2 ⚠ 반드시 src/index.js 첫 줄에 코드를 작성해야 합니다.
```





React 앱 브라우저 호환성 — browserlist 설정

Browserlist

package.json 파일에 설정된 browserlist 설정에 맞춰
React 앱이 정상 작동되도록 JavaScript 코드를 출력합니다.

- ◆ 개발 (development)
- ◆ 배포 (production)

Queries

browserlist에 설정 가능한 쿼리 리스트

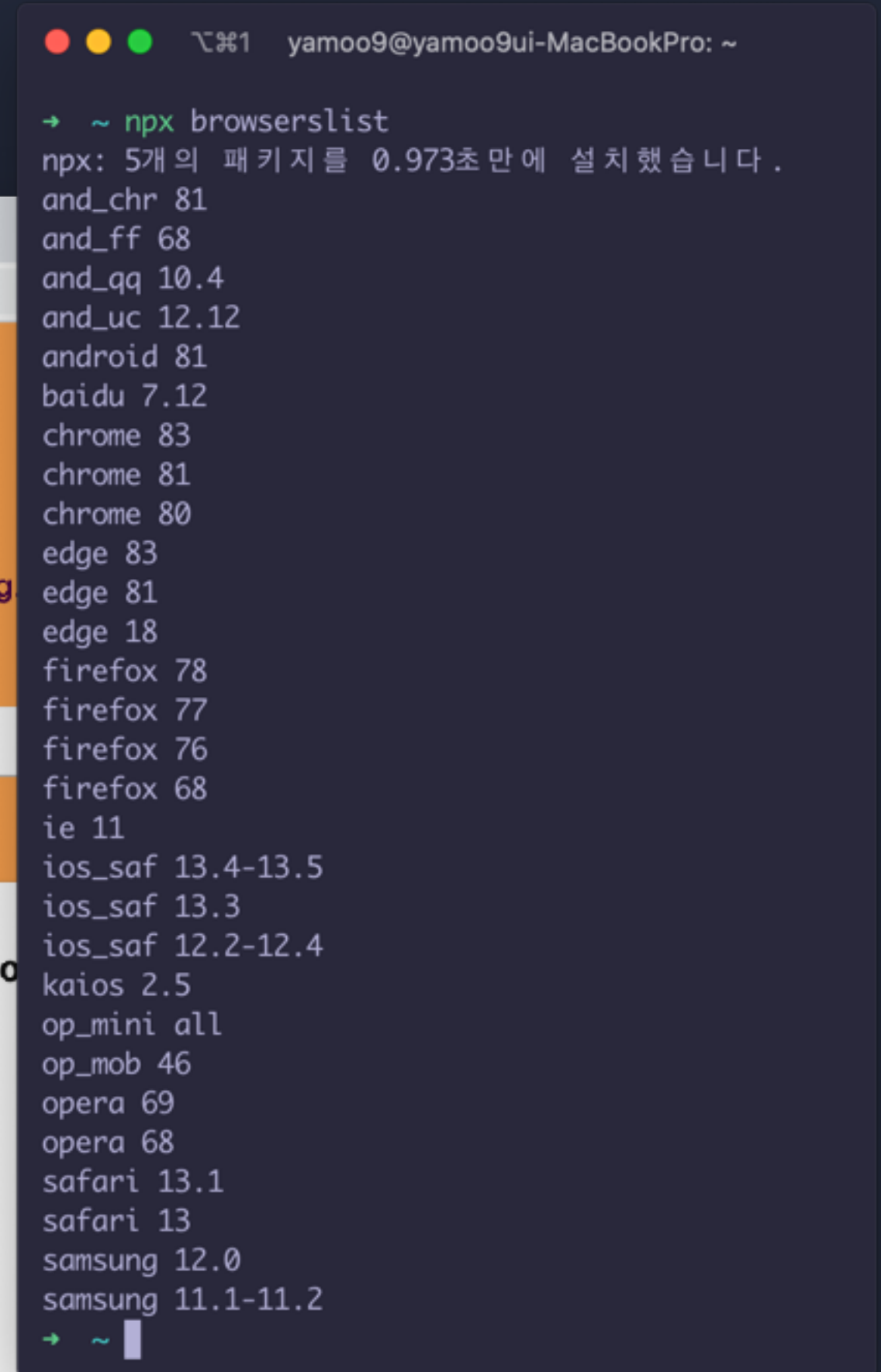
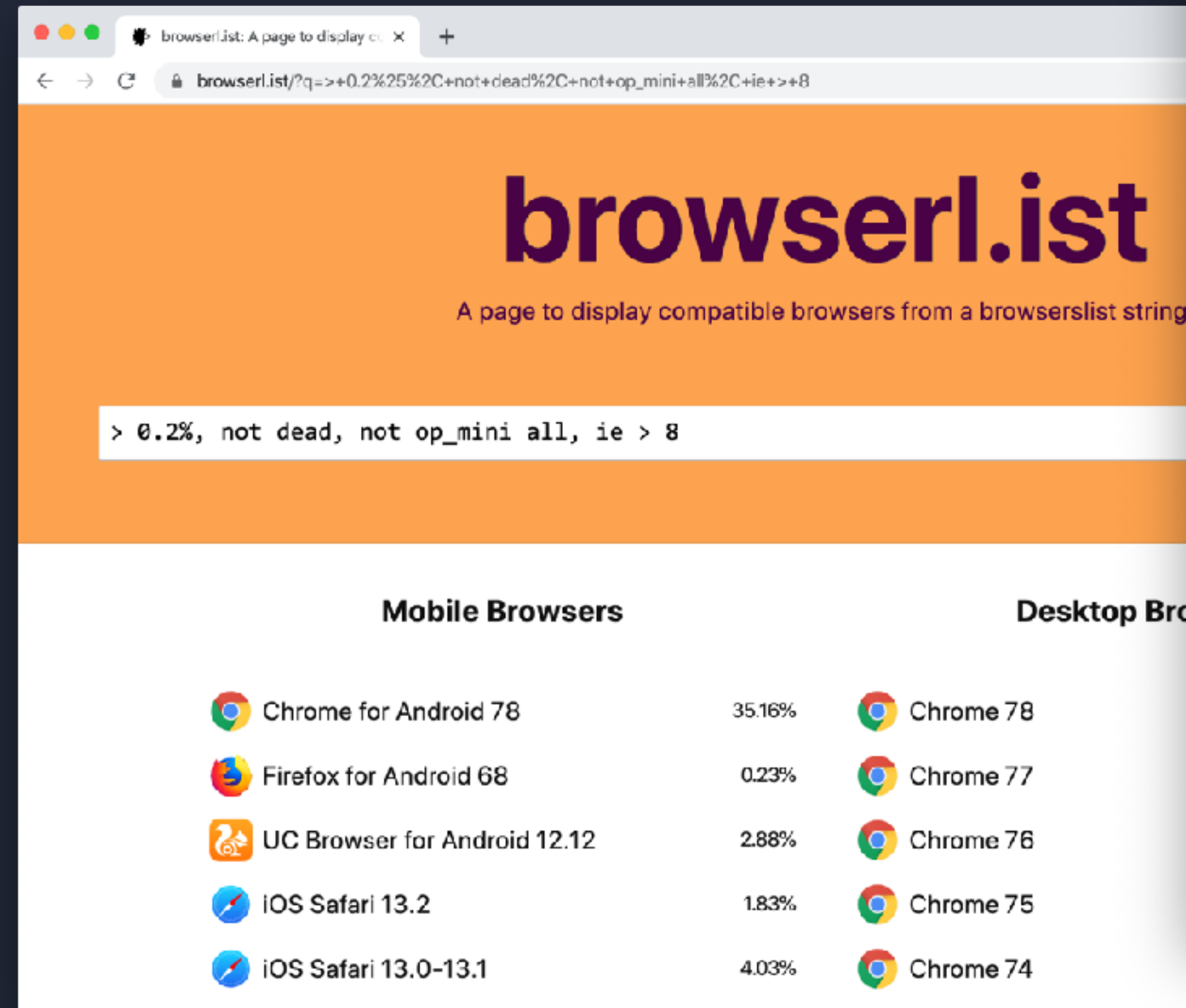
Cache

browserlist 설정 변경 후, 정상적으로 반영되지 않으면?
`node_modules/.cache` 를 삭제해야 합니다.

```
npm cache clean
npm cache verify
```



WITH
YAMOO9
DERESA KIM



0.22%





에디터 + 디버깅 도구 설정

Launch JSON

Visual Studio Code 에디터 안에서 디버깅 하려면?
디버거 도구 설치 및 약간의 설정이 필요합니다.

- ◆ [Debugger for Chrome](#)
- ◆ [.vscode/launch.json](#)

1. 프로젝트를 시작 (npm start)
2. 디버그 아이콘 클릭 (F5)
3. 코드에 중단점 설정하고 디버깅

.vscode > {...} launch.json > Launch Targets > {} Chrome

```
1  {
2    "version": "0.2.0",
3    "configurations": [
4      {
5        "name": "Chrome",
6        "type": "chrome",
7        "request": "launch",
8        "url": "http://localhost:3000",
9        "webRoot": "${workspaceFolder}/src",
10       "sourceMapPathOverrides": {
11         "webpack:///src/*": "${webRoot}/*"
12       }
13     ]
14   ]
15 }
```





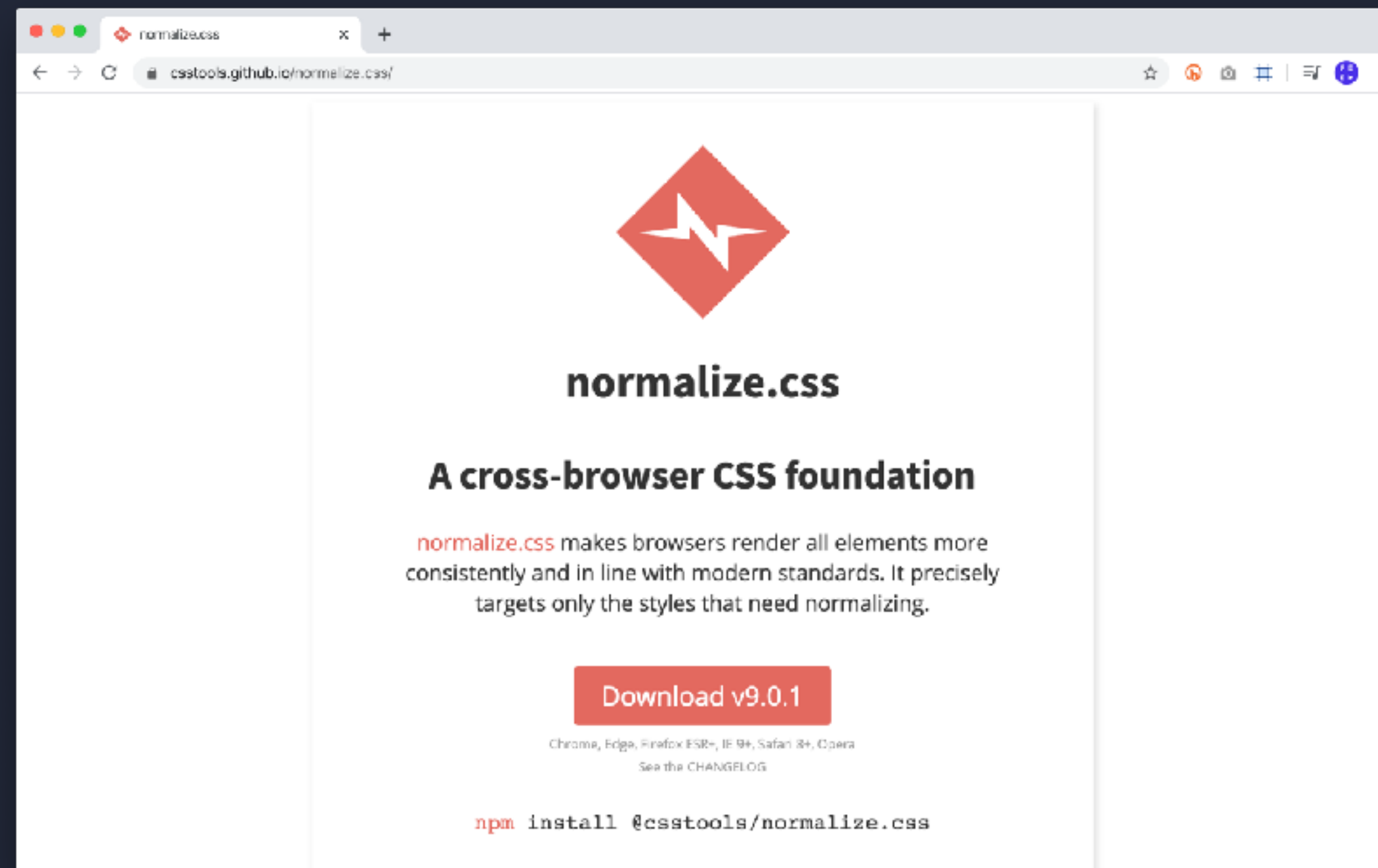
CSS 초기화

PostCSS - Normalize

CSS 엔트리 파일에 1회 호출 하면, normalize.css 스타일을 사용해 브라우저 간 기본 스타일 차이점을 일반화 합니다.

- ◆ [postcss-normalize](#)
- ◆ [CSS Reset](#)

```
// PostCSS Normalize (normalize.css) 사용 설정  
@import-normalize;
```





CSS Grid 레이아웃 설정

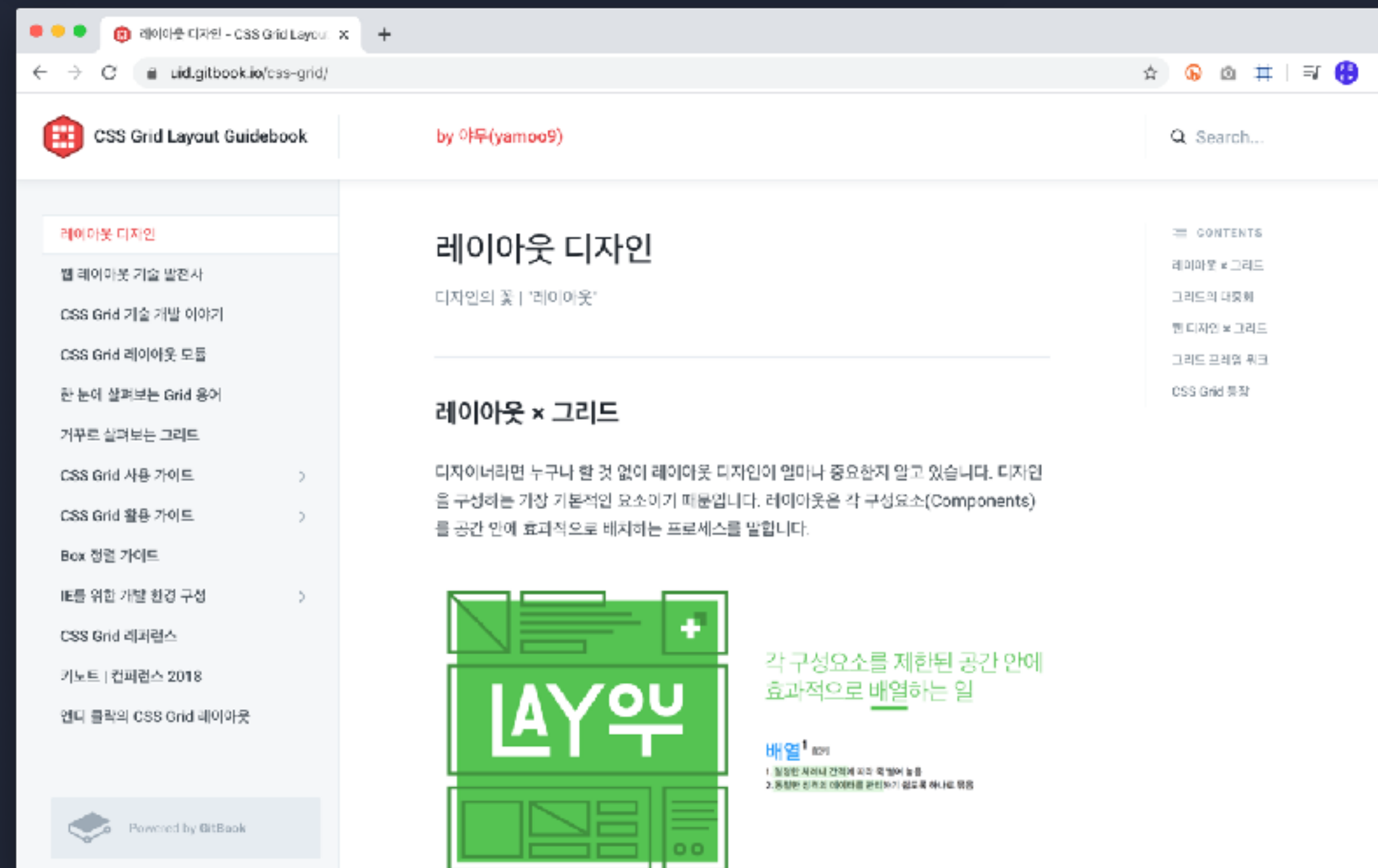
PostCSS - Autoprefixer

CSS Grid 레이아웃을 미지원 브라우저(예: IE)에서 사용하려면 브라우저 제조사 접두사(vender-prefix)를 붙여주는 도구를 사용 할 수 있습니다.

이 설정을 사용하려면 CSS 엔트리 파일 상단에 오른쪽 주석(Comment) 코드를 추가 합니다.

- ◆ CSS Grid 레이아웃
- ◆ Grid Autoplacement support in IE

```
// CSS Grid 모듈 사용 설정
/* autoprefixer grid: autoplace */
```





SASS_PATH 변수

.env

Sass 프리 프로세서를 프로젝트에 사용 할 경우,
절대 경로를 설정하면 유용합니다. (기본 설정: ~)

◆ [env-vscode](#)

◆ .env

📄 .env

```
1 # Sass 경로 설정
2 # windows의 경우: SASS_PATH=./node_modules;./src
3 SASS_PATH=node_modules:src
4
```



WITH
YAM009
DERESA KIM





SVG 컴포넌트

SVG ReactComponent

SVG 파일을 React 컴포넌트로 불러올 수 있습니다. 이 방법은 SVG 코드를 가진 컴포넌트 파일을 별도로 만들지 않아도 되어 유용합니다. (title을 포함한 props를 전달할 수 있습니다.)

- ◆ SVG 란?
- ◆ Accessible SVG (접근성)

```
import { ReactComponent as Logo } from 'assets/logo.svg';

const App = (props) => (
  <div className="App">
    <header className="App__header">
      <a href="/" className="App__HomeLink">
        <Logo title="이름(E.UID) 블렌디드 러닝" style={{ width: '200px' }} />
      </a>
    </header>
  </div>
);
```





public 디렉토리 에셋

정적 에셋 저장소

public 디렉토리에 있는 에셋(자산)은 React Scripts의 모듈 시스템으로 관리하지 않습니다. 변경 사항이 거의 없는 이미지, CSS, 폰트 파일을 관리할 경우 유용합니다.

- ◆ [index.html] → %PUBLIC_URL%
- ◆ [*.(js|x)] → process.env.PUBLIC_URL

```
<img  
  src={process.env.PUBLIC_URL + '/images/logo.svg'}  
  alt="이름(E.UID) 블렌디드 러닝"  
>
```



WITH
YAM009
DERESA KIM





글로벌 변수

간혹 ESLint에서 글로벌 변수를 사용하려 할 경우,
오류 또는 경고를 표시하곤 합니다. 이러한 문제는
글로벌 변수를 명시적으로 설정하면 해결됩니다.

또는 아래 주석을 추가해 해당 라인을 무시할 수 있습니다.
`// eslint-disable-line`

```
.. // 전역 변수  
.. const $ = window.$;  
  
.. // global 추출  
.. const { setTimeout, clearTimeout } = window;
```





코드 분리 관리

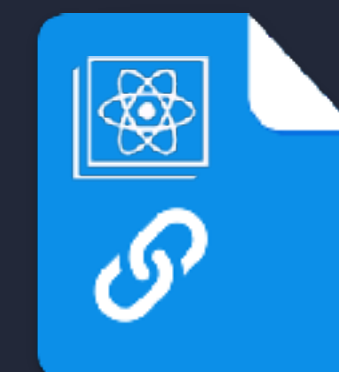
동적 import() + Promise

ES6의 import 구문과 달리, 코드 내부에 import()를 사용해 필요할 경우에만, 모듈을 불러와 사용할 수 있습니다.

- ◆ dynamic import (표준) 제안 (Stage 4)
- ◆ 동적 import()를 사용해 필요할 때 모듈을 불러오는 방법



```
const App = (props) => {  
  
  const handleClick = (e) => {  
    // 동적 import()를 사용하여 필요한 모듈을 불러와 사용할 수 있습니다.  
    import('utils/index.js')  
      // 비동기 호출을 통해 모듈을 불러온 후, 기본 모듈의 별칭을 설정하여 사용할 수 있습니다.  
      .then(({ default: utils }) => {  
        // 불러온 유틸 모듈에 종속된 코드를 사용해 코드 로직을 작성할 수 있습니다.  
        utils.each(document.querySelectorAll('.App *'), (domNode, index) =>  
          domNode.setAttribute('data-index', `node-index-${index}`)  
        );  
      })  
    // 비동기 호출 과정에서 오류가 발생하면 Console 패널에 오류 메시지가 출력됩니다.  
    .catch((e) => console.error(e.message));  
    e.preventDefault();  
  };  
  
  return (  
    <div className="App">  
      <header className="App__header">  
        <a href="/" className="App__HomeLink" onClick={handleClick}>  
          <Logo title="이름(E.UID) 블렌디드 러닝" style={{ width: '200px' }} />  
        </a>  
      </header>  
    </div>  
  );  
  
};
```





코드 분리 관리

동적 import() + Async/Await

Promise 객체 대신, 비동기 함수를 사용할 수도 있습니다.
비동기 함수 사용을 선호한다면? async, await 키워드를
try - catch 문과 함께 사용합니다.

◆ 비동기 함수(Async Function)

```
const App = (props) => {  
  // Async 함수를 사용할 수도 있습니다.  
  const handleClick = async (e) => {  
    e.preventDefault();  
  
    // Async 함수 디버깅을 위해 try - catch 문을 사용합니다.  
    try {  
      // 동적 import()를 사용하여 필요한 모듈을 불러온 후,  
      // 객체 구조 분해 할당을 사용해 필요한 코드를 불러와 사용합니다.  
      const { each } = await import('utils/index.js');  
      // 유틸 모듈 코드를 사용해 DOM 객체를 조작합니다.  
      each(document.querySelectorAll('.App *'), (domNode, index) =>  
        domNode.setAttribute('data-index', `node-index-${index}`)  
      );  
    } catch (e) {  
      // 오류 발생 시, 오류 메시지를 출력합니다.  
      console.error(e.message);  
    }  
  };  
  
  return (  
    <div className="App">  
      <header className="App__header">  
        <a href="/" className="App__HomeLink" onClick={handleClick}>  
          <Logo title="이름(E.UID) 블렌디드 러닝" style={{ width: '200px' }} />  
        </a>  
      </header>  
    </div>  
  );  
};
```





환경 변수

.env

프로젝트 루트에 위치한 .env 파일에 환경 변수를 설정한 후, 전역(Global) 변수처럼 사용할 수 있습니다.

- ◆ `[*.js(x)]` → `process.env.NODE_ENV`
- ◆ `[*.js(x)]` → `process.env.REACT_APP_*`
- ◆ `[index.html]` → `%REACT_APP_*`

```
❏ .env
1  # Sass 경로 설정
2  # windows의 경우: SASS_PATH=./node_modules;./src
3  SASS_PATH=node_modules:src
4
5  # .env에 설정한 환경 변수는 빌드 과정에서 임베드(embed) 됩니다.
6  # ⚠️ 그러므로 외부에 유출되는 코드를 포함해서는 안됩니다!
7  REACT_APP_PRIVATE_KEY=react_private_key_23jkw@klfjsalkd
8
```

```
console.log(process.env.REACT_APP_PRIVATE_KEY);
```

```
<p>REACT_APP_PRIVATE_KEY</p>
```





PWA

Progressive Web App

배포 빌드 시, PWA 옵션(오프라인/캐시 우선 동작)을 사용 하려면 Service Worker 모듈의 register()를 설정합니다. 등록을 하면 기존 웹 페이지보다 안정적이고 빠른 모바일 환경을 제공할 수 있습니다.

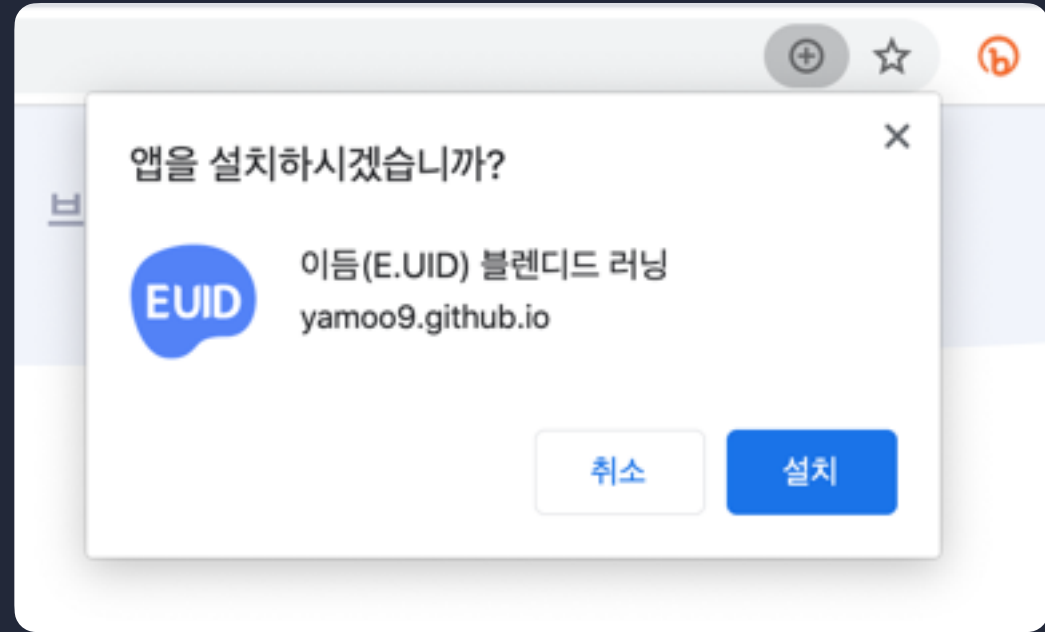
- ◆ PWA 란?
 - ◆ 페이지 속도 향상 (정적 에셋 캐시)
 - ◆ 오프라인 상황에서도 작동 가능 (웹이지만, 앱과 유사)
 - ◆ 스마트 폰에 추가 가능 (앱 처럼 설치 불필요)
 - ◆ PWA 메타데이터 설정 (앱 아이콘, 브랜딩, 컬러 설정 가능)
-
- ◆ 속도, 디버깅 문제로 배포 할 때만 등록!
 - ◆ HTTPS에서만 동작!

```
import * as serviceWorker from 'config/serviceWorker';

ReactDOM.render(<App />, document.getElementById('app'));

// 앱을 오프라인에서 작동시키고 보다 빠르게 로드 하고자 한다면?
// 아래 코드의 unregister()를 register()로 변경합니다.
// [⚠ 주의! 이 방법은 몇 가지 문제를 발생할 수 있습니다.]
// 서비스 워커에 대해 자세히 알아보기: https://bit.ly/CRA-PWA
if (process.env.NODE_ENV === 'production') {
  serviceWorker.register();
}
```





WITH
YAMOO9
DERESA KIM





프리 렌더링

react-snap

SPA를 정적 HTML로 프리 렌더링 합니다. Headless Chrome을 사용해 모든 페이지를 변환/생성 합니다.

실제 DOM 요소 노드의 자식 노드 소유 여부를 확인하여 hydrate() 또는 render()로 조건 분기 합니다.

Install:

```
yarn add --dev react-snap
```

Change package.json:

```
"scripts": {  
  "postbuild": "react-snap"  
}
```

Change src/index.js (for React 16+):

```
import { hydrate, render } from "react-dom";  
  
const rootElement = document.getElementById("root");  
if (rootElement.hasChildNodes()) {  
  hydrate(<App />, rootElement);  
} else {  
  render(<App />, rootElement);  
}
```

