0️⃣

Introduction

0. 들어가기 전에

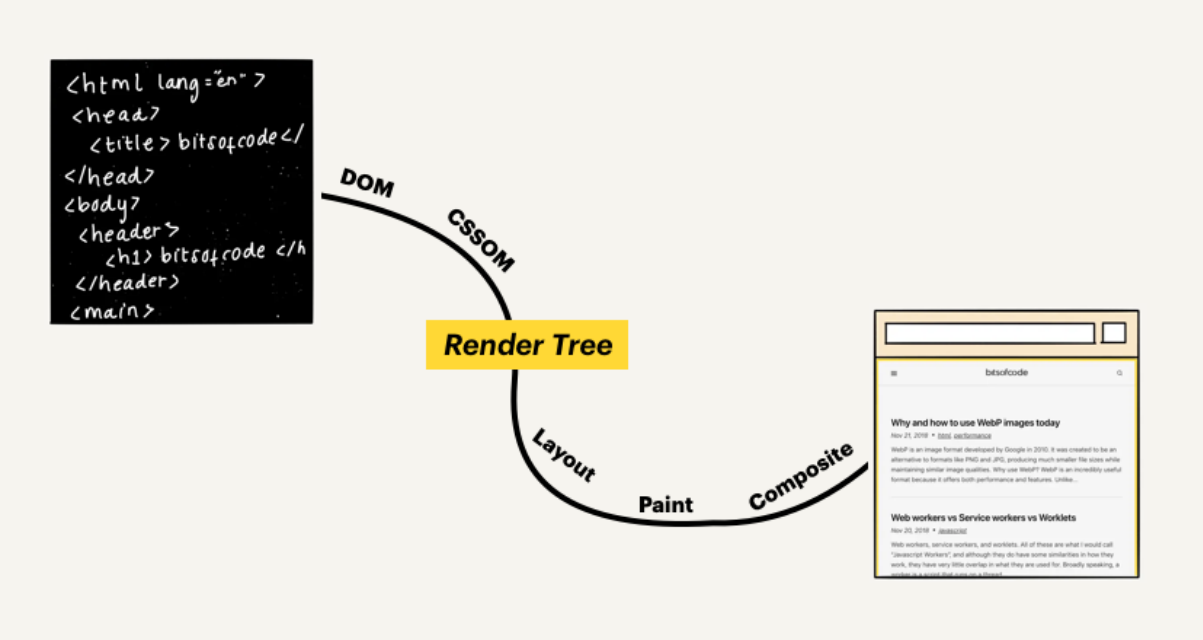
리액트는 자바스크립트의 프론트엔드 라이브러리로, 객체지향에 기반을 두고 있습니다. 따라서 클래스, 객체, 상속 등의 객체 지향에 대한 이해가 부족하다면 자바스크립트에 대한 기초를 닦고 나서 리액트를 공부하는 것을 추천합니다.

또한 리액트는 Node.js를 기반으로 하고 있기 때문에 Node.js에 대한 기초적인 이해가 반드시 필요합니다.

1. React?

리액트의 탄생 배경

리액트의 핵심 철학을 이해하는 것은 "리액트 다운" 코드를 만드는 데 중요합니다. 그러기 위해서 리액트의 탄생 배경에 대해 살펴보겠습니다.

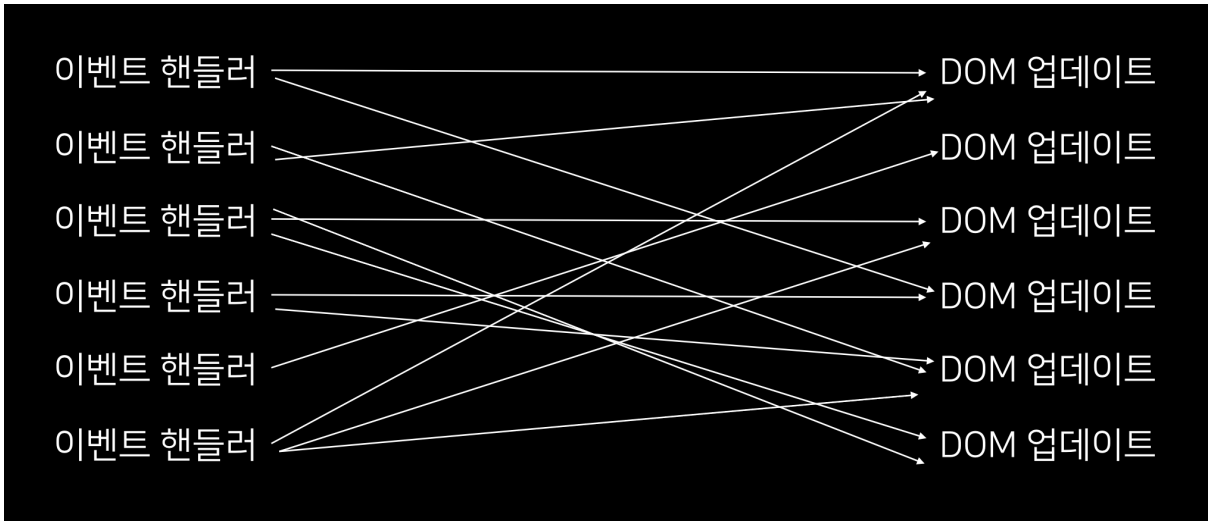


서버에서 응답으로 받은 HTML코드를 브라우저는 DOM(Document Object Model)이라는 트리 구조를 만듭니다. 비슷한 방법으로, CSS는 CSSOM 트리로 만들어집니다. 브라우저는 DOM과 CSSOM을 이용해 렌더링 트리를 만들고 이 렌더링 트리로부터 실제 브라우저의 화면이 그려지게(rendering) 됩니다.

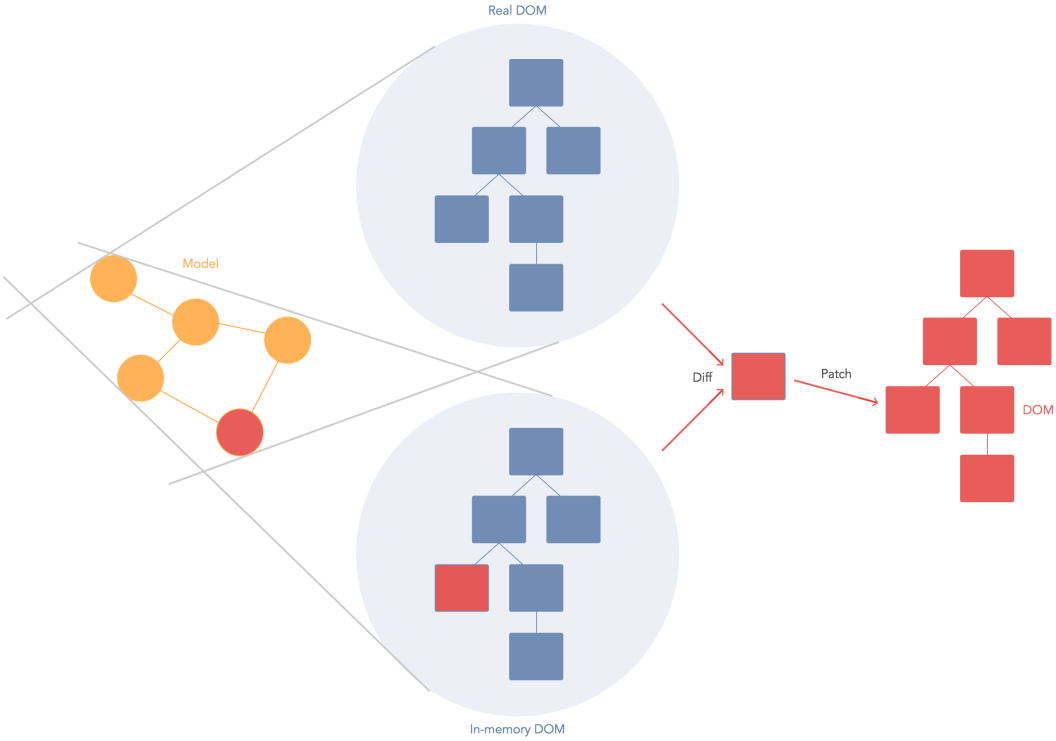
문제는 HTML/CSS는 서버에서 한 번 응답을 받으면 바뀌지 않기 때문에 웹에 동적인 요소를 넣기 위해서는 자바스크립트를 반드시 사용해야 합니다. 자바스크립트는 DOM을 직접 수정할 수 있기 때문에 브라우저 화면을 직접적으로 조작할 수 있게 됩니다.

> https://codepen.io/indosaram/details/KKqMzrm

그러나 웹 앱의 크기가 커지면서 위 예제와 같이 사용자와의 인터랙션을 통해 DOM을 수정하는 코드들이 엄청 늘어나게 되었습니다.



그러다 보니 어떤 코드에서 어떤 DOM을 조작하는지를 통합적으로 관리할 필요성이 생기게 됩니다. 여기서 리액트는 가상 DOM(Virtual DOM)이란 것을 도입해 이 문제를 해결합니다.



Virtual DOM이란 실제 DOM이 아니라 컴퓨터의 메모리상에 존재하는 자바스크립트 객체로, 리액트는 어떤 이벤트가 발생하게 되면 가상 DOM을 업데이트하고, 그 다음 실제 DOM과의 차이점을 비교합니다. 그리고 변경된 부분의 DOM만 업데이트해주는 방식을 통해 빠른 DOM 업데이트가 가능합니다.

Component

```
class NewComponent {
  state = {};
  render() {} // React element => Virtual DOM (in memory)
}
```

리액트는 기존의 HTML의 각 요소들을 컴포넌트(Component)로 나누는 패턴을 사용합니다. 컴포넌트를 사용하게 되면,

1. 가독성
2. 재사용성
3. 유지보수

라는 세 가지 측면에서 큰 장점을 얻게 됩니다.

JSX와 기본 규칙들

JSX는 컴포넌트를 편리하게 사용할 수 있도록 하는 리액트만의 문법입니다. 실제로는 리액트는 Babel이라는 도구를 통해 JSX를 JS로 컴파일해 사용합니다. JSX의 기본적인 문법은 아래와 같습니다. HTML 태그처럼 생겼지만 자바스크립트입니다.

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

표현식

```
const name = 'Josh Perez';
const element1 = <h1>Hello, {name}</h1>;
const element2 = (
  <h1>
    Hello, {formatName(name)}!
  </h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

변수 `name` 에서 선언된 값을 `element` 안의 `{name}` 에 넣을 수 있습니다. 변수명 뿐만 아니라, 자바스크립트 표현식은 모두 중괄호 안에 넣을 수 있습니다.

JSX 엘리먼트

JSX의 모든 태그는 닫혀야 합니다. 만일 텍스트를 포함하지 않는 태그라면 self-closing 태그를 사용할 수 있습니다. 또한, JSX는 자식 엘리먼트를 포함할 수 있습니다.

```
const element = (
  <div>
    <h1>Hello!</h1>
    <h2>Good to see you here.</h2>
    <Body /> {/*<Body></Body>*/}
  </div>
);
```

React fragment

JSX는 두 개 이상의 태그를 사용하려면 반드시 다른 태그로 이를 감싸줘야 합니다. 하지만 적당하게 감쌀 만한 태그가 없다면 React fragment를 사용할 수 있습니다. fragment는 실제 DOM에 나타나지 않기 때문에 불필요하게 `<div>` 태그 등으로 태그를 감쌀 필요가 없습니다.

```
import React from 'react';
import Hello from './Hello';

function App() {
  return (
    <>
      <Hello />
      <div>Goodbye</div>
    </>
  );
}

export default App;
```

어트리뷰트

JSX는 HTML보다 JS와 더 비슷합니다. 따라서 어트리뷰트 이름을 캐멀 케이스(Camel case)로 작성합니다. 예를 들어 `class` 어트리뷰트는 `className` 으로 표기합니다.

```
import React from 'react';
import Hello from './Hello';

function App() {
  return (
    <>
      <Hello />
      <div className="good-bye">Goodbye</div>
    </>
  );
}

export default App;
```

주석

JS와 JSX의 주석은 각각 다음과 같이 표기합니다.

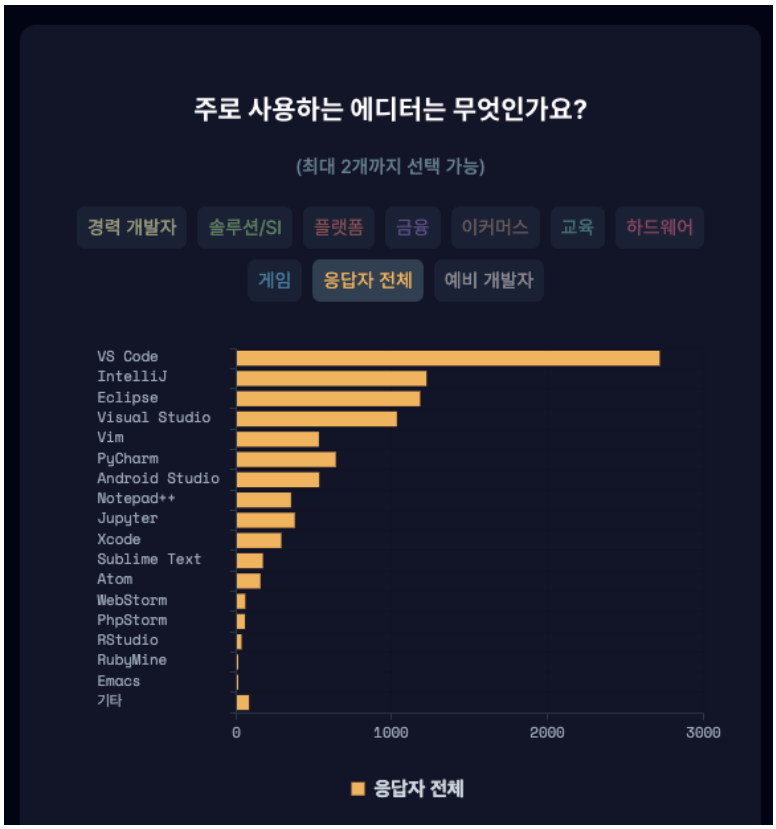
```
import React from 'react';
import Hello from './Hello';
import './App.css';


function App() {
  // comment in javascript

  return (
    <>
      {/* comment in jsx */}
      /* This is normal text */
      <Hello />
      <div className="good-bye">Goodbye</div>
    </>
  );
}

export default App;
```

2. Getting started with React

이번 장에서는 리액트 개발 환경을 설정해 보겠습니다.

Visual Studio Code

Visual Studio Code는 마이크로소프트에서 만든 오픈 소스 코드 편집기입니다. 미려한 디자인과 강력한 성능을 모두 갖추고 있어서 많이 사용되고 있습니다. 2021년 프로그래머스 설문조사에서도 선호도 1위를 차지했습니다.

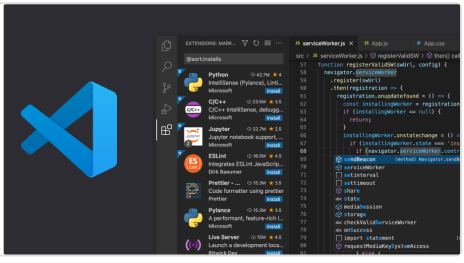


설치는 아래 링크에서 자신의 운영체제에 맞는 버전을 다운로드해 설치하면 됩니다.

Download Visual Studio Code - Mac, Linux, Windows

Visual Studio Code is free and available on your favorite platform - Linux, macOS, and Windows. Download Visual Studio Code to experience a redefined code editor, optimized for

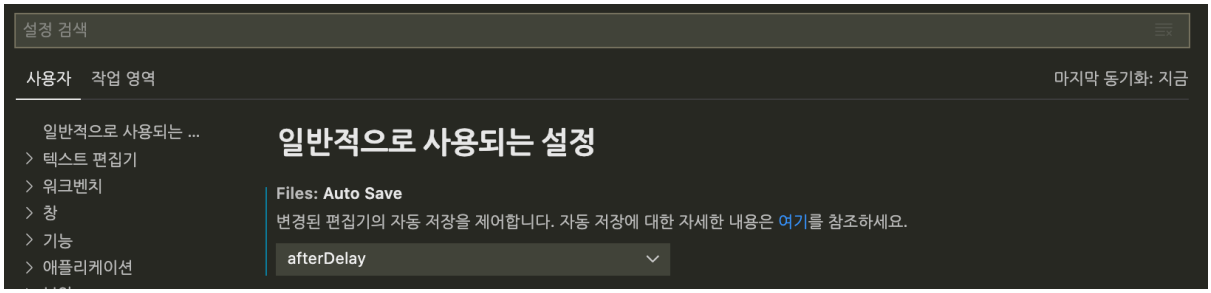
https://code.visualstudio.com/download



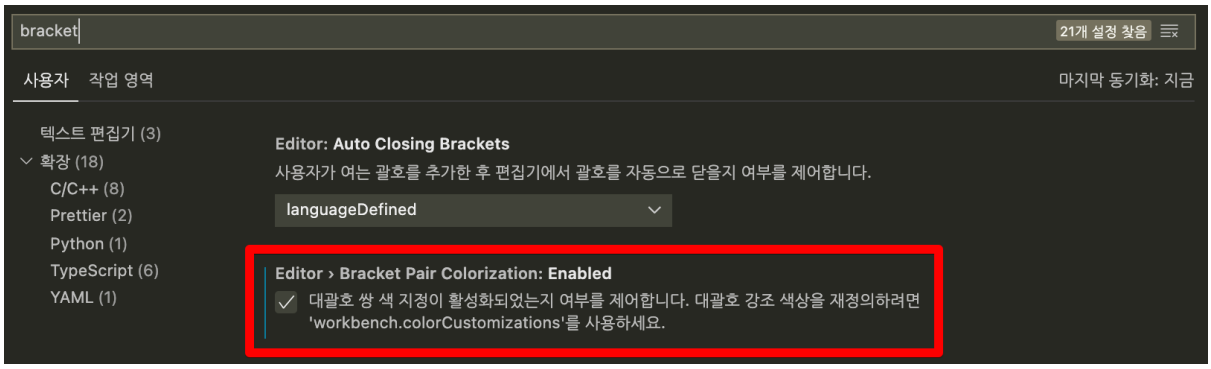
기본 설정

VSCode 자체 설정 2가지를 변경합니다.

Auto Save: 코드가 변경된 다음 자동으로 저장해주는 설정입니다.

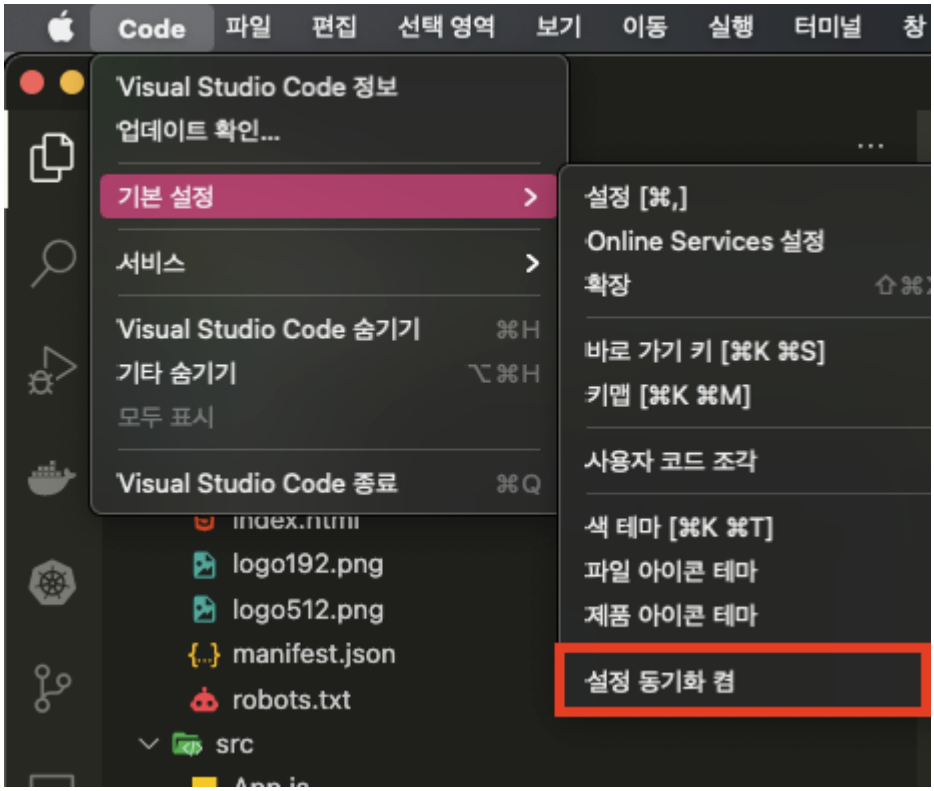


Bracket Pair Colorization: 코드의 괄호끼리 색상을 맞추어주는 설정입니다.



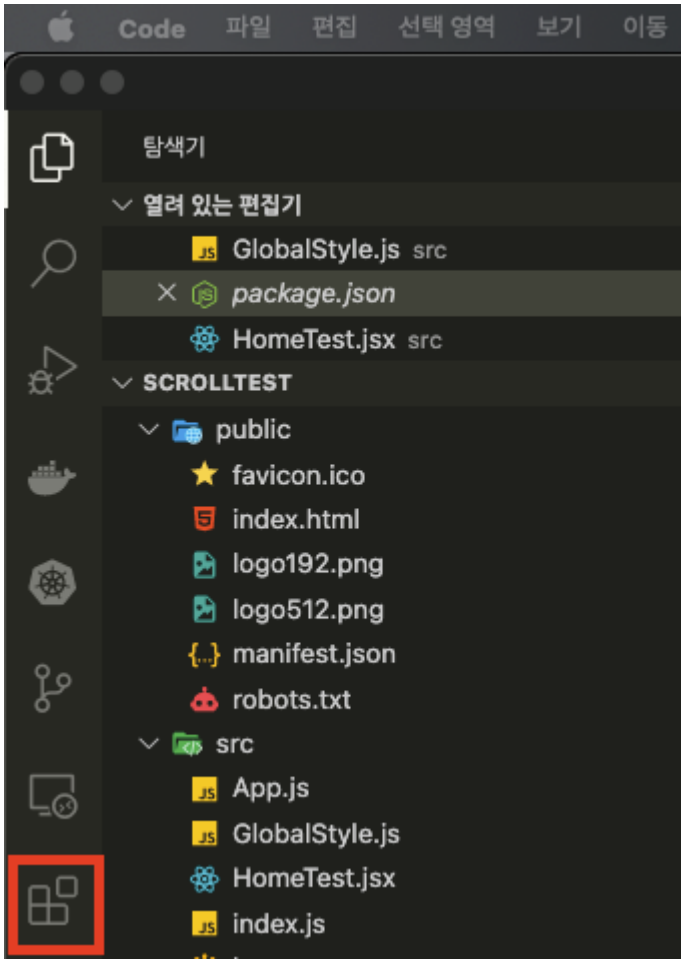
설정 동기화 켜기

VSCode는 깃허브 또는 마이크로소프트 계정으로 로그인하면 계정에 종속되는 환경 설정을 만들 수 있습니다. 이 설정은 클라우드에 저장되기 때문에 설정 동기화를 켜 놓으면, 새로운 컴퓨터에서 VSCode를 설치하더라도 예전에 사용하던 테마, 확장 프로그램, 기타 설정 등이 모두 동기화되기 때문에 아주 편리합니다.



확장 프로그램 설치하기

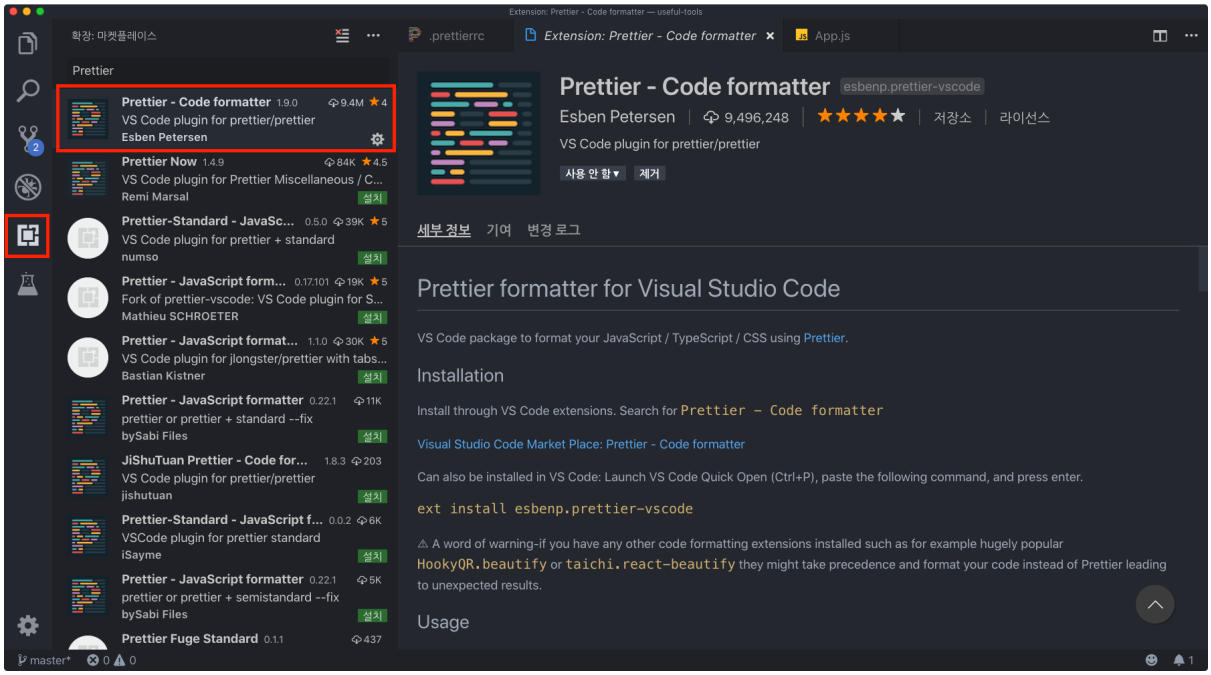
VSCode를 사용하는 가장 큰 이유 중 하나가 바로 확장 프로그램입니다. 프로그래밍을 편리하게 해주는 기능들이 정말 많은데, 이건 개발자들의 취향을 크게 타기도 해서 여기서는 리액트 개발에 필수적인 몇 가지만 추천해 드리겠습니다.



설치 방법은 VSCode 왼쪽 아이콘 중에서 블럭 모양으로 된 아이콘을 클릭한 다음, 검색창에 설치하고자 하는 프로그램의 이름을 입력하면 됩니다.

Prettier

Prettier 는 자동으로 코드의 스타일을 관리해주는 도구입니다.



Prettier는 설정 파일로 `.prettierrc` 파일을 사용합니다. 프로젝트 루트에 `.prettierrc` 파일을 아래와 같이 만듭니다. 아래 내용은 Prettier 공식 홈페이지에 나와 있는 기본 설정입니다.

```
{
  "trailingComma": "es5",
  "tabWidth": 4,
  "semi": false,
  "singleQuote": true
}
```

설정 파일은 json형식으로 되어 있고, 각 키 값은 다음과 같은 의미를 가지고 있습니다.

- `trailingComma` : `"none"`, `"es5"`, `"all"` 으로 설정을 할 수 있는데, 객체 또는 배열이 여러줄로 구성되어 있으면 다음과 같이 맨 마지막 줄에 쉼표를 붙여줍니다.

  ```
  const object = {
    a: 1,
    b: 2,
  };
  ```

  `"none"` 이면 쉼표를 붙이지 않고, `"es5"` 이면 객체, 배열을 사용하게 될 때 쉼표를 붙이고, `"all"` 이면 함수를 사용 할 때 인자를 전달 할 때도 쉼표를 붙입니다.

- `tabWidth`: 들여쓰기(indent)의 크기를 공백의 갯수로 정의합니다.
- `semi` : 세미콜론 (;) 을 쓸지 말지 정합니다.

- `singleQuote`: 문자열을 `'` 로 나타냅니다. 만일 `"` 이 더 좋다면 값을 `false` 로 바꿉니다.

> 💡 코드 스타일은 사람마다, 프로젝트마다 다 다르기 때문에 여기서는 기본값을 사용하겠습니다.

이제 `Alt/option + Shift + F` 를 사용해 코드를 포맷할 수 있습니다.

ESLint

ESLint 는 자바스크립트의 문법을 확인해주는 도구입니다. CRA 로 만든 프로젝트에는 이미 적용이 되어있어서 만약에 우리가 자바스크립트 실수를 하게 되면 터미널에 오류 또는 경고가 나타나게 되죠. 예를 들어서 아까 수정한 코드처럼 `a` 라는 값을 선언 후 사용하지 않으면, 터미널에서 다음과 같은 결과물이 나타나게 됩니다.

```
Compiled with warnings.

./src/App.js
  Line 5:  'a' is assigned a value but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

이번에는 ESLint 의 VS Code 익스텐션을 설치해보세요.

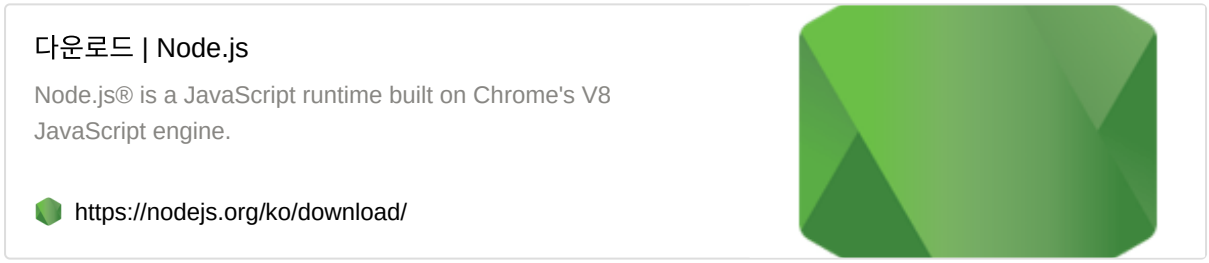
이를 설치하고 나면 터미널에서만 보이던 경고가 에디터상에서도 보이게 됩니다.

하단의 경고 정보는 에디터 가장 아래에 있는 경고 아이콘을 누르면 볼 수 있습니다.

Create React App

Install Node.js

홈페이지에서 안정적인 LTS 버전의 노드를 설치합니다. 2021년 8월 기준, Node 14가 최신 LTS 버전입니다.



노드를 설치하면 Node Package Manager, `npm`이 함께 설치됩니다.

`npm` 과 `npx` 의 차이점

`npm` : Package Manager. 특정 패키지를 설치, 제거, 업데이트하는 도구.

`npx` : Package Runner. `npm@5.2.0` 이상에서 사용할 수 있는 `npm`의 CLI 도구. npm 레지스트리에 올라가 있는 최신 버전의 패키지들을 사용해 해당 명령어를 실행시키고, 의존 패키지들을 삭제해줌.

```
node --version
npm --version
npx --version
```

Create React App

터미널을 열고, 리액트 프로젝트를 만들 폴더에서 `npx` 를 이용해 새로운 리액트 프로젝트를 생성합니다.

```
npx create-react-app react-app
cd react-app
```

`create-react-app` 은 Dev server, WebPack, Babel 과 같은 도구들을 포함하고 있기 때문에 쉽고 빠르게 새로운 리액트 프로젝트를 만들 수 있습니다.

Webpack, Babel 은 무슨 용도인가요?

- Webpack : 여러개로 나누어진 소스코드를 하나로 묶어주는 자바스크립트 번들 도구
- Babel : JSX와 같은 최신 자바스크립트 문법을 이전 버전에서도 호환되도록 컴파일해 주는 도구.

폴더 구조

- `node_modules` : 서드파티 노드 모듈
- `public` : 배포 시 누구나 접근할 수 있는 파일들
- `src` : 기본 컴포넌트 및 다른 소스 코드

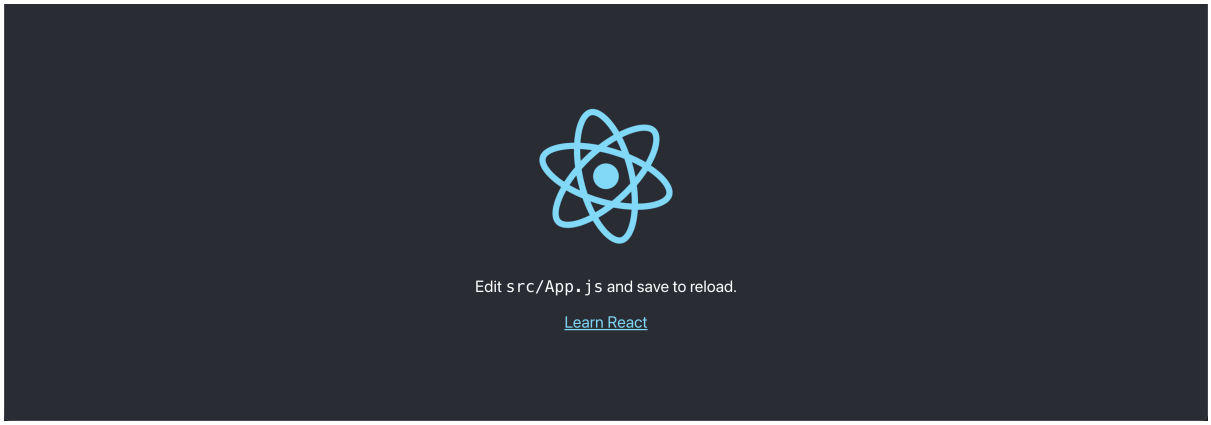
Sample app 실행하기

이제 터미널에서 아래 명령어를 통해 샘플 앱을 실행해 보겠습니다.

```
npm start
```

그러면 아래와 같은 메시지가 콘솔에 출력되고, 브라우저에서 샘플 앱이 실행됩니다.

```
Compiled successfully!

You can now view react-app in the browser.

  http://localhost:3000

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



메세지에서도 알 수 있듯이 localhost의 3000번 포트에서 개발 서버가 실행되고, 개발 서버에서 `index.html` 을 서빙하고 있습니다. `index.html` 파일을 자세히 들여다보면, 리액트 관련 코드가 없는 것을 알 수 있는데요.

```
<body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div> // React DOM will be inserted here
  </body>
```

리액트는 `<div id="root"></div>` 의 아래에 DOM을 삽입하게 됩니다.

이제 필요 없는 파일을 삭제하고 본격적으로 리액트 앱을 개발해 보겠습니다.

- App.css
- App.test.js
- index.css
- logo.svg
- reportWebVitals.js
- setupTests.js