

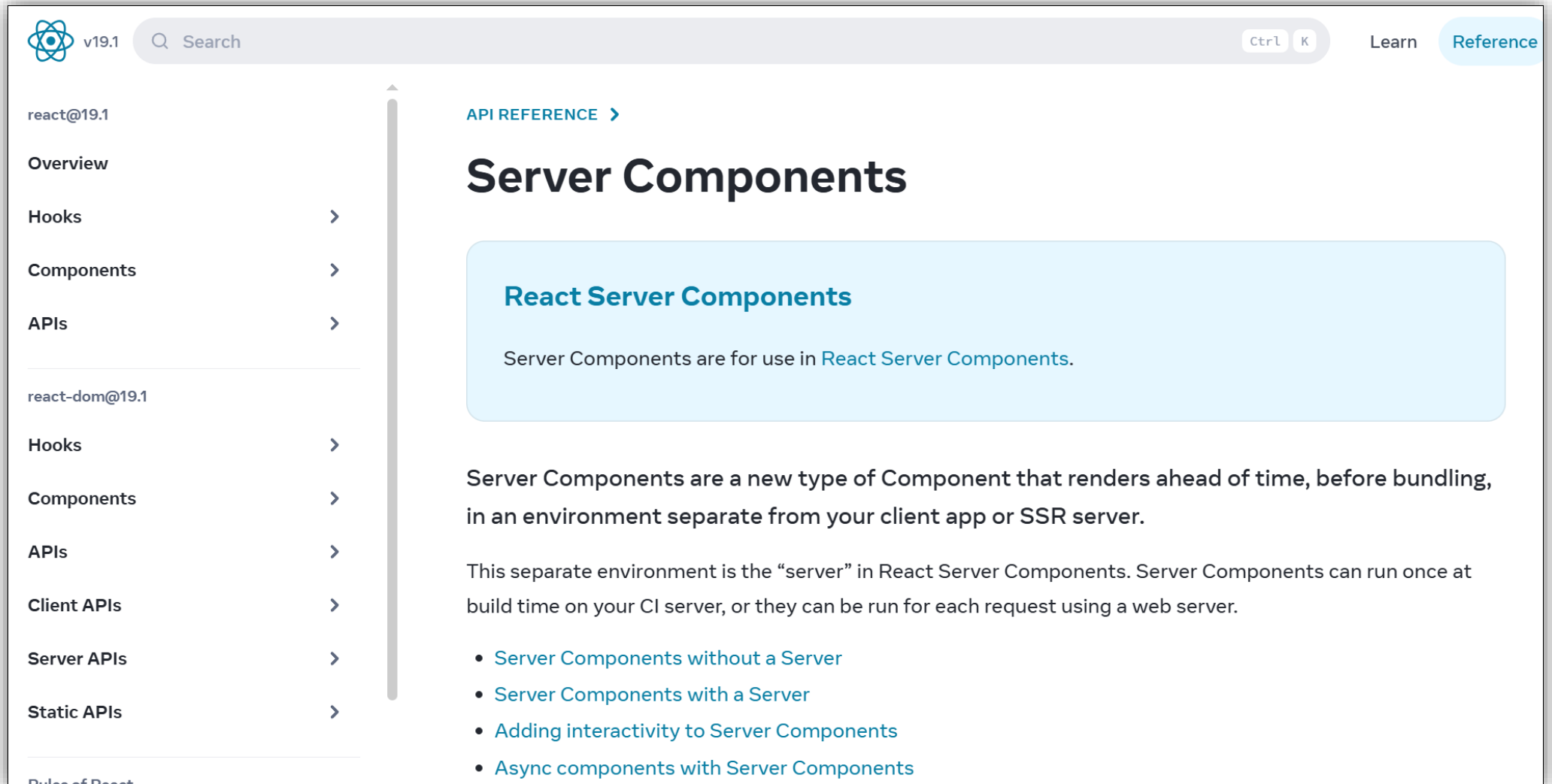
React Server Components

React Server
Component



소프트웨어융합대학원
진혜진

React Server Components



The screenshot shows the React documentation website for version 19.1. The left sidebar contains a navigation menu with sections for 'react@19.1' and 'react-dom@19.1'. Under 'react@19.1', the 'Components' section is selected. The main content area is titled 'Server Components' and features a light blue callout box with the heading 'React Server Components' and a description. Below this, there is a paragraph explaining that Server Components are a new type of Component that renders ahead of time. A list of links is provided at the bottom of the main content area.

react@19.1

Overview

Hooks >

Components >

APIs >

react-dom@19.1

Hooks >

Components >

APIs >

Client APIs >

Server APIs >

Static APIs >

Rules of React

API REFERENCE >

Server Components

React Server Components

Server Components are for use in [React Server Components](#).

Server Components are a new type of Component that renders ahead of time, before bundling, in an environment separate from your client app or SSR server.

This separate environment is the “server” in React Server Components. Server Components can run once at build time on your CI server, or they can be run for each request using a web server.

- [Server Components without a Server](#)
- [Server Components with a Server](#)
- [Adding interactivity to Server Components](#)
- [Async components with Server Components](#)

React Server Components

■ React Server Components

- React 컴포넌트를 서버에서 렌더링하고, 해당 컴포넌트가 반환하는 결과(HTML + 필요한 최소한의 스크립트/데이터)를 클라이언트로 스트리밍
 - 클라이언트 번들 크기 감소
 - 서버 전용 로직 사용 가능
 - 분할 렌더링

React Server Components

■ 기존 렌더링 방식과의 차이점

항목	Client-Side Rendering(CSR)	Server-Side Rendering(SSR)	React Server Components(RSC)
렌더링 위치	브라우저(클라이언트)	서버(Next.js, Express 등)	서버(React 18 RSC) + 클라이언트(필요한 컴포넌트만)
데이터 fetching	클라이언트 중심	서버에서 1차 렌더 후 클라이언트에서 fetch	서버 컴포넌트가 DB나 API에 직접 접근 가능필요 부분만 클라이언트로 전송
번들 크기	클라이언트 번들 크게 증가	SSR 초기 데이터 전송 후 클라이언트에서도 큰 번들	서버 컴포넌트로 처리된 부분은 JS 번들에서 제외클라이언트 컴포넌트만 번들에 포함
상호 작용	완전히 클라이언트	SSR -> 클라이언트 상호 작용	서버 컴포넌트와 클라이언트 컴포넌트가 적절히 분리, 상태와 UI가 혼합되어 동작

■ 클라이언트 컴포넌트와의 차이점

- 렌더링 환경 : 서버 컴포넌트는 오직 서버 측에서만 렌더링되고, 클라이언트에는 HTML 결과만 전달된다.
- 상태 및 상호작용 : 서버 컴포넌트는 상태(state)나 이벤트 핸들러를 포함할 수 없다.
- 데이터 페칭 방식 : 서버 컴포넌트에서는 데이터베이스 조회나 외부 API 호출 등을 그 자리에서 직접 수행할 수 있다.
- 번들링 및 성능: 서버 컴포넌트의 가장 큰 특징 중 하나는 클라이언트 번들 크기에 영향이 없다.
- 조합과 계층 구조: React는 서버 컴포넌트와 클라이언트 컴포넌트를 하나의 트리에서 혼합하여 사용할 수 있도록 해준다.

React Server Components

■ 장점

- **성능 향상:** RSC를 활용하면 브라우저로 전송되는 JS 용량을 크게 줄여 초기 페이지 로딩 속도를 높일 수 있다.
- **SEO 개선:** 서버 컴포넌트가 생성한 콘텐츠는 초기 HTML에 포함되어 전달되므로, 클라이언트에서 JS 실행을 기다리지 않아도 검색 엔진 크롤러가 즉시 콘텐츠를 볼 수 있다.
- **간편한 데이터 패칭 및 상태 관리 감소:** 컴포넌트 단위로 서버에서 데이터를 가져올 수 있기 때문에, useEffect나 전역 상태 관리 없이 필요한 데이터 로직을 해당 컴포넌트에 직접 작성할 수 있다.
- **보안성 및 비밀유지:** 서버 컴포넌트는 API 키, DB 비밀번호 등의 비밀 정보를 서버 영역에서만 처리하므로 보안에 유리하다.
- **코드 분할 및 재사용:** RSC 덕분에 서버용과 클라이언트용 로직을 적절히 분리하여 각 환경에 최적화된 코드 구조를 만들 수 있다.

React Server Components

■ 단점


- 상호작용 한계: RSC 자체만으로는 사용자 상호작용을 처리할 수 없으므로, 결국 클라이언트 컴포넌트와 함께 사용해야 한다.
- 서버 의존성 및 아키텍처 고려: RSC를 활용하려면 결국 서버 런타임(Node.js 등) 환경이 필요하며, 완전히 정적인 사이트(예: 정적 호스팅만으로 동작하는 SPA)에서는 쓸 수 없다.
- 학습 곡선: RSC는 React의 새로운 패러다임이기 때문에, 기존 CSR/SSR 위주로 개발하던 개발자에게는 개념적 학습 부담이 있다.
- 타사 라이브러리 제약: React 생태계의 모든 라이브러리가 아직 RSC를 완전히 지원하는 것은 아니다.
- 디버깅 및 복잡도: 서버와 클라이언트에서 동작하는 컴포넌트가 섞여 있다 보니, 디버깅이나 상태 추적이 복잡할 수 있다.

React Server Components

- **Next.js와의 통합 사례**
 - 서버 컴포넌트 기본 지원
 - 클라이언트 컴포넌트 사용
 - 레이아웃과 스트리밍
 - 데이터 패칭과 캐싱
 - Next.js 13 (Beta) → Next.js 14 (Stable)

React Server Components

- <https://nodejs.org/ko>


 [학습](#) [소개](#) [다운로드](#) [블로그](#) [문서](#) [기여하기](#) [인증](#)

🔍 검색어를 입력하세요... Ctrl + K

Next-10 Survey We need your feedback to help us shape Node.js →

어디서든 JavaScript를 실행하세요

Node.js®는 무료, 오픈소스, 다중 플랫폼 JavaScript 런타임 환경으로 개발자 여러분이 서버, 웹 애플리케이션, 명령어 작성 도구와 스크립트를 만들도록 해줍니다.

Node.js 다운로드 (LTS) 

Node.js 다운로드 v22.16.0¹ LTS. Node.js는 패키지 관리자를 통해서도 다운로드 할 수 있습니다.

Node.js가 제공하는 [학습 자료](#)를 통해 더 많은 정보를 알아보세요.

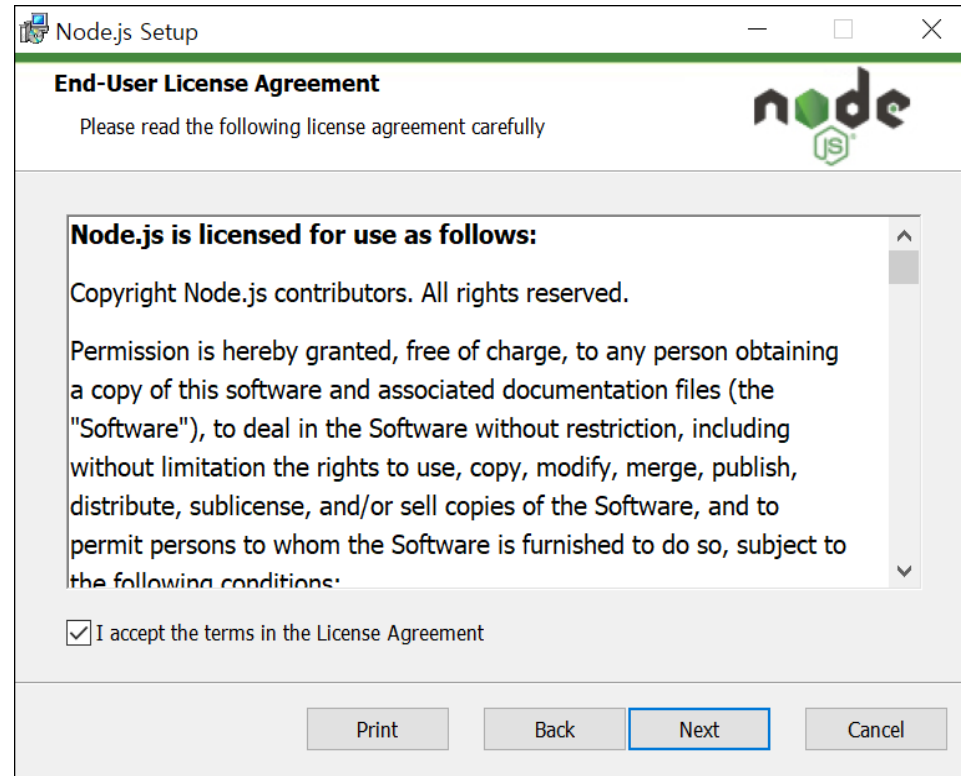
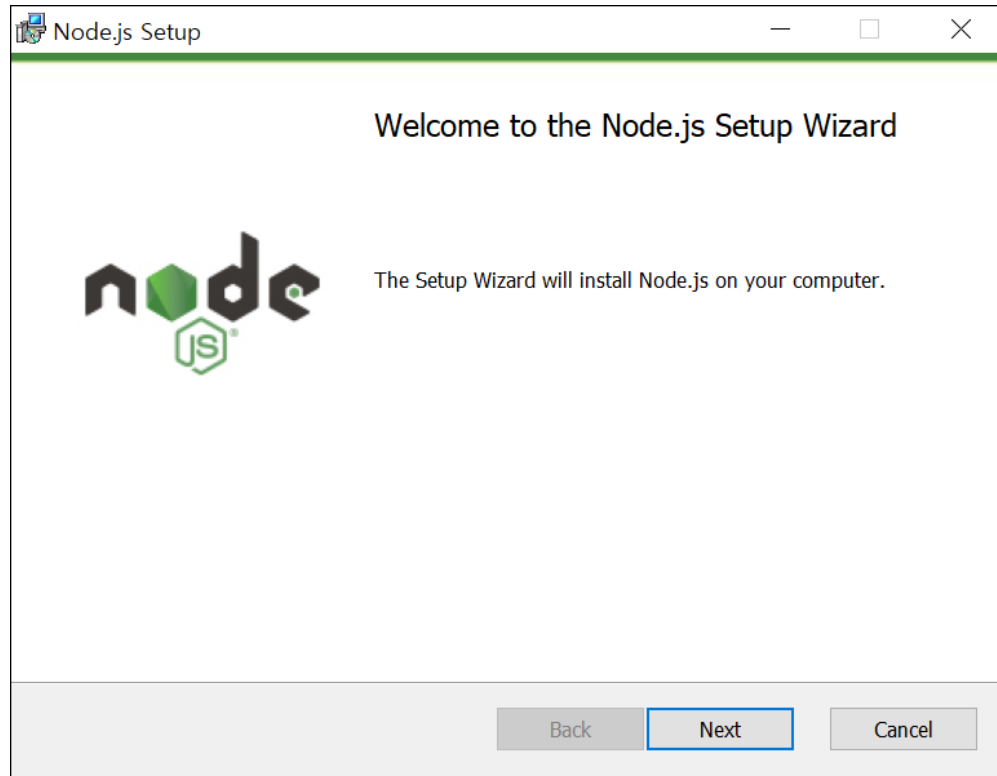
Create an HTTP Server Write Tests Read and Hash a File Streams Pipeline Work with Threads

```
1 // server.mjs
2 import { createServer } from 'node:http';
3
4 const server = createServer((req, res) => {
5   res.writeHead(200, { 'Content-Type': 'text/plain' });
6   res.end('Hello World!\n');
7 });
8
9 // starts a simple http server locally on port 3000
10 server.listen(3000, '127.0.0.1', () => {
11   console.log('Listening on 127.0.0.1:3000');
12 });
13
14 // run with `node server.mjs`
15
```

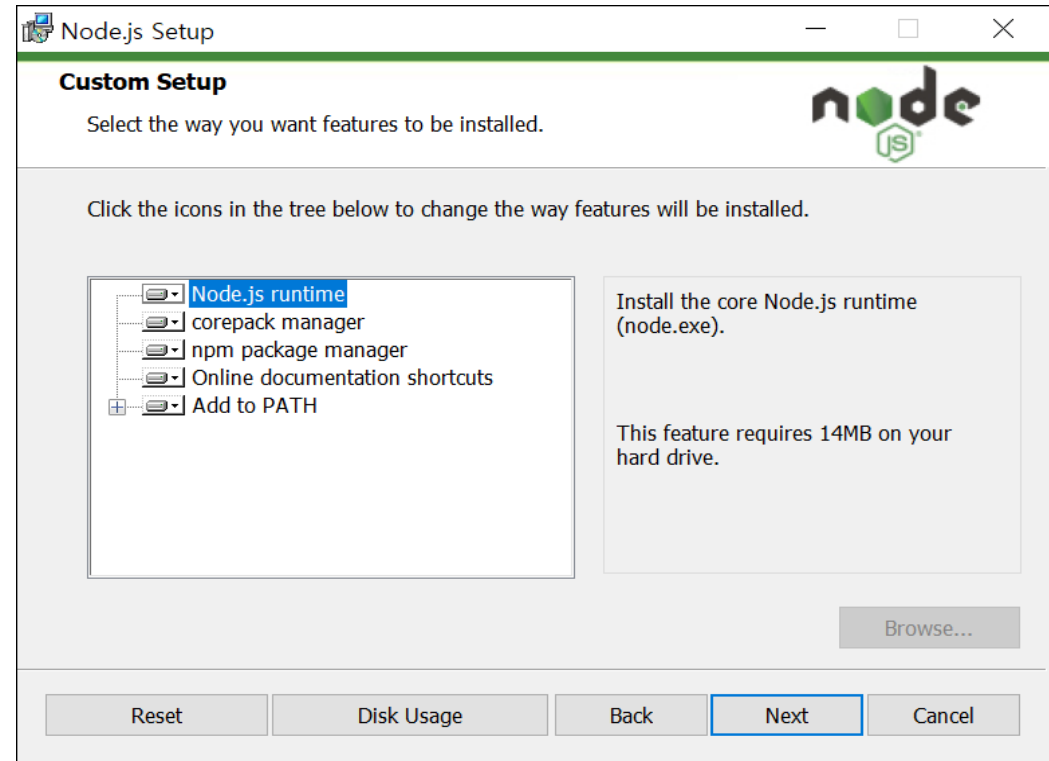
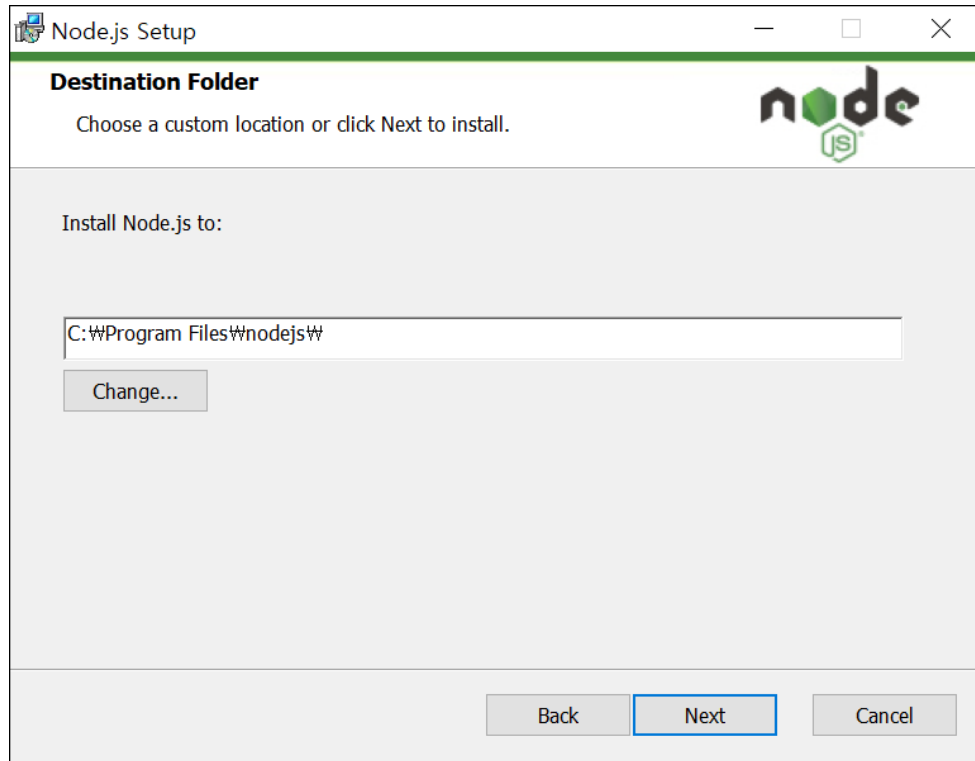
JavaScript

클립보드에 복사

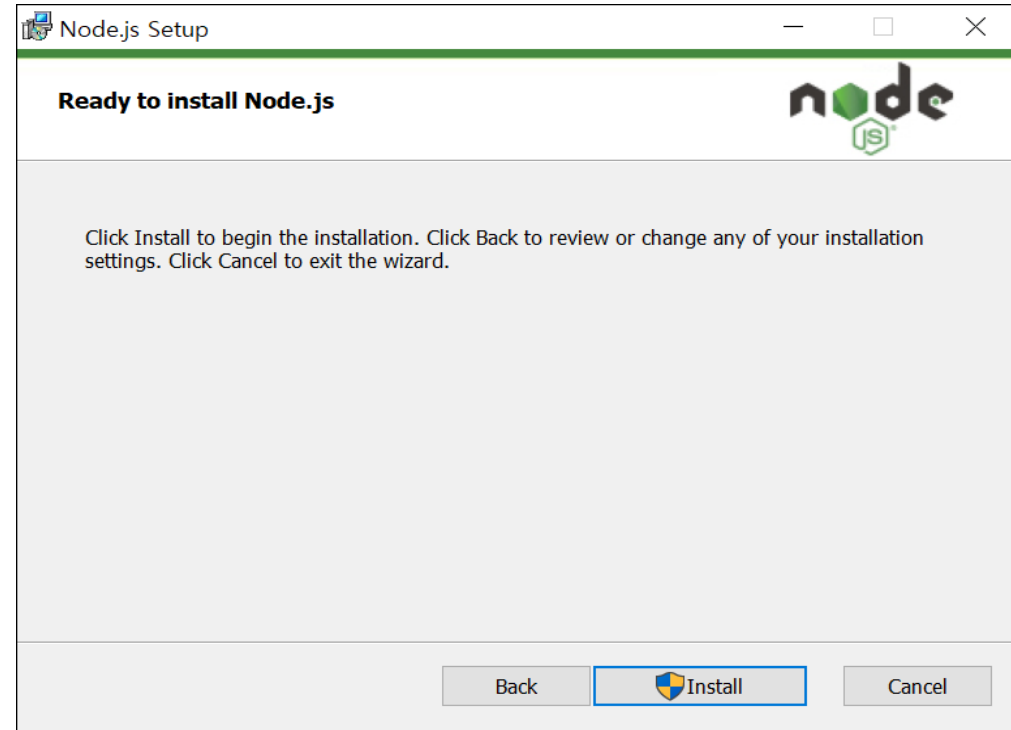
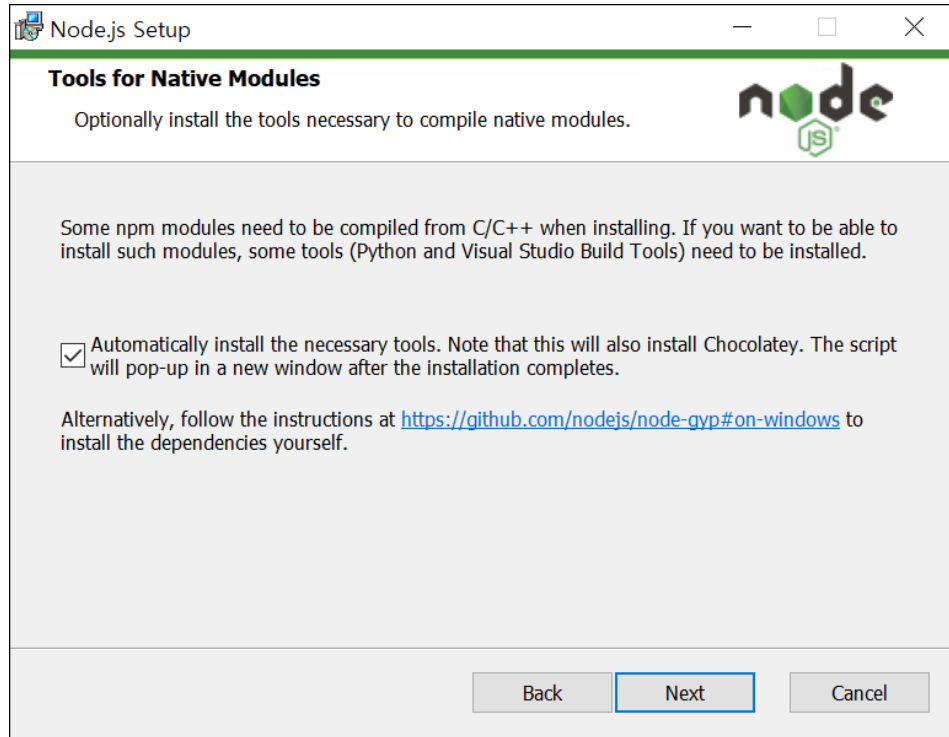
React Server Components



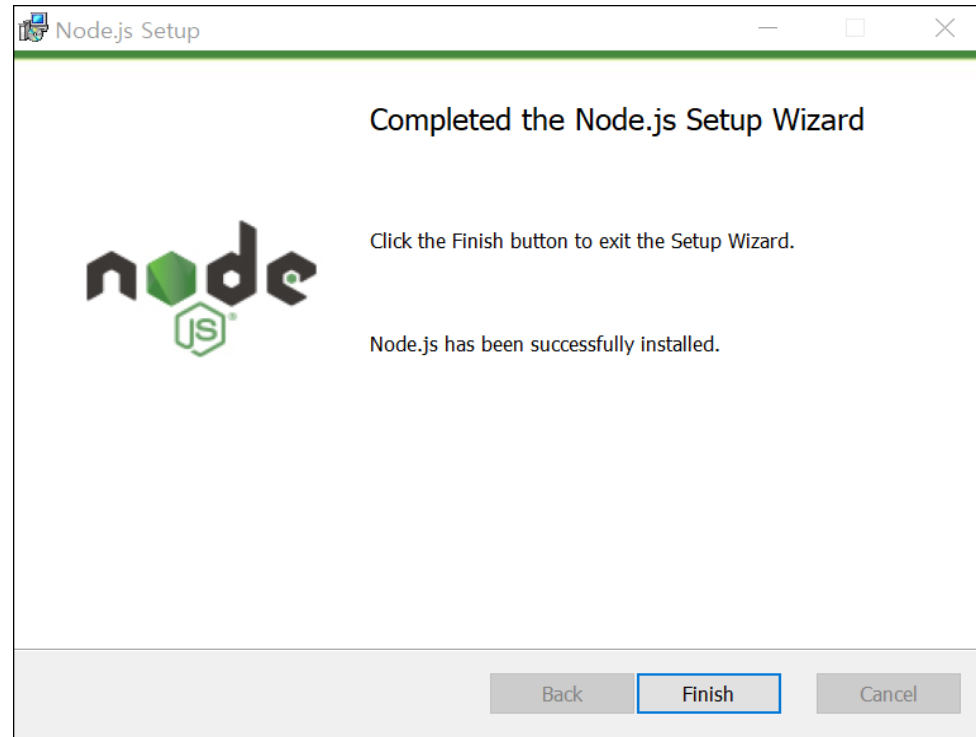
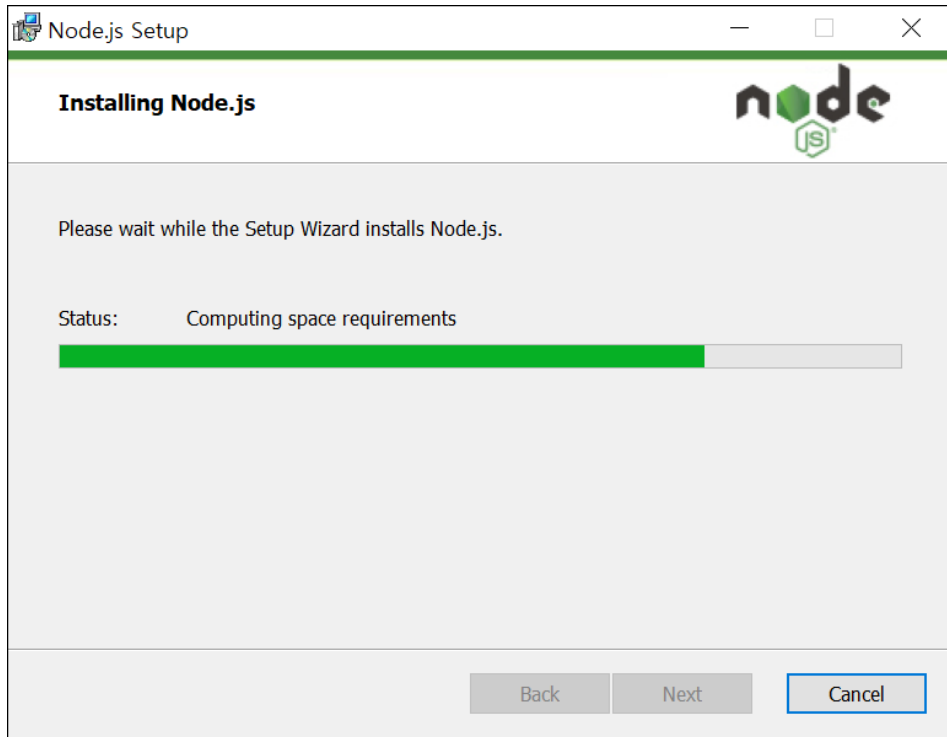
React Server Components



React Server Components

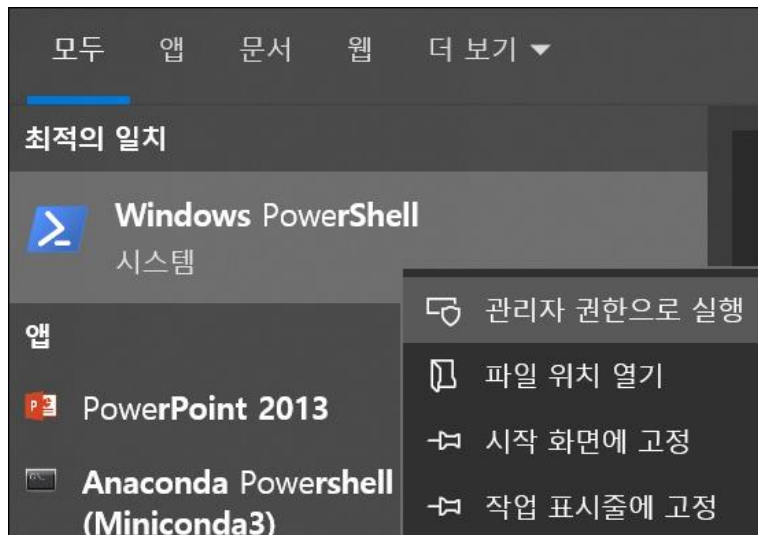


React Server Components



React Server Components

■ Windows에서 React Server Components 실행



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\whjjin> Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

실행 규칙 변경
실행 정책은 신뢰하지 않는 스크립트로부터 사용자를 보호합니다. 실행 정책을 변경하면 about_Execution_Policies 도움말
(https://go.microsoft.com/fwlink/?LinkID=135170)에 설명된 보안 위험에 노출될 수 있습니다. 실행 정책을
변경하시겠습니까?
[Y] 예(Y) [A] 모두 예(A) [N] 아니요(N) [L] 모두 아니요(L) [S] 일시 중단(S) [?] 도움말 (기본값은 "N"): Y
PS C:\Users\whjjin> npx create-next-app@latest my-js-rsc-app
  Would you like to use TypeScript? ... No / Yes
  Would you like to use ESLint? ... No / Yes
  Would you like to use Tailwind CSS? ... No / Yes
  Would you like your code inside a 'src/' directory? ... No / Yes
  Would you like to use App Router? (recommended) ... No / Yes
  Would you like to use Turbopack for 'next dev'? ... No / Yes
  Would you like to customize the import alias ('@/*' by default)? ... No / Yes
  What import alias would you like configured? ... @/*
Creating a new Next.js app in C:\Users\whjjin\my-js-rsc-app.

Using npm.

Initializing project with template: app-tw

Installing dependencies:
- react
- react-dom
- next

Installing devDependencies:
- postcss
- tailwindcss
- eslint
- eslint-config-next
- @eslint/eslintrc

added 373 packages, and audited 374 packages in 23s

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

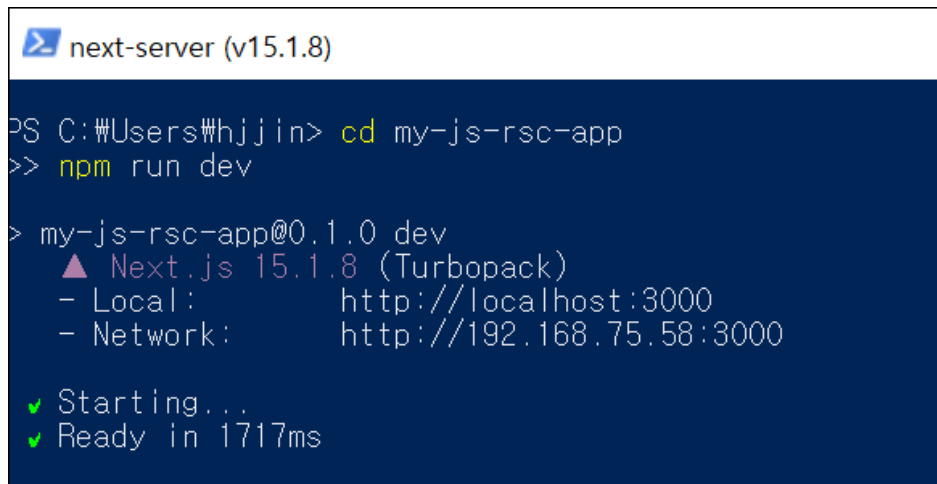
1 critical severity vulnerability

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
success! Created my-js-rsc-app at C:\Users\whjjin\my-js-rsc-app
PS C:\Users\whjjin>
```

React Server Components

■ Windows에서 React Server Components 실행



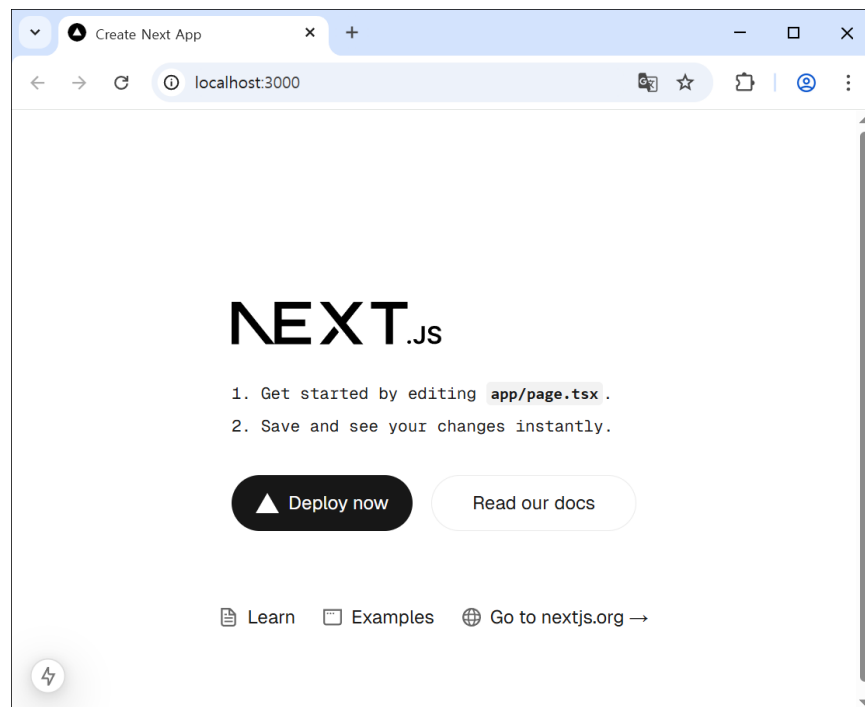
```
next-server (v15.1.8)  
  
PS C:\Users\hjjin> cd my-js-rsc-app  
>> npm run dev  
  
> my-js-rsc-app@0.1.0 dev  
  ▲ Next.js 15.1.8 (Turbopack)  
  - Local:      http://localhost:3000  
  - Network:    http://192.168.75.58:3000  
  
  ✓ Starting...  
  ✓ Ready in 1717ms
```

- 브라우저에서 ➡ <http://localhost:3000> 열면 시작 화면이 나온다.

React Server Components

■ 개발 서버 실행

- 프로젝트 디렉토리로 이동하여 개발 서버를 시작한다.
- `cd my-js-rsc-app`
- `npm run dev`
- 브라우저에서 `http://localhost:3000`을 열어 애플리케이션을 확인할 수 있다.



React Server Components

- **실행 정책 임시 변경**

- `Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass`
- 실행 정책을 변경하겠냐고 묻는 창이 나오면
- [Y] 에 입력하고 Enter

- **JavaScript 기반 Next.js 프로젝트 생성**

- `npx create-next-app@latest my-js-rsc-app`

React Server Components

■ 설치 과정 질문 안내

- Would you like to use TypeScript? ✕ No
- Would you like to use ESLint? ✓ Yes(권장)
- Would you like to use Tailwind CSS? 선택사항
- Would you like to use src/ directory? ✕ No (단순화 위해)
- Would you like to use App Router? ✓ Yes ← RSC 필수
- Would you like to customize import alias? ✕ No

React Server Components

- **설치가 완료되면 다음 메시지가 나온다.**
 - Success! Created my-js-rsc-app at C:\...
- **프로젝트 폴더로 이동 및 서버 실행**
 - `cd my-js-rsc-app`
 - `npm run dev`
- **브라우저 열고 <http://localhost:3000> 접속**
- **Next.js 기본 화면이 뜨면 성공**