



도구

온라인에서 사용해보기

Vue SFC를 사용해보려면 컴퓨터에 아무것도 설치할 필요가 없습니다. 브라우저에서 바로 사용할 수 있는 온라인 플레이그라운드가 있습니다:



Vue SFC Playground

항상 최신 커밋에서 배포됨

컴포넌트 컴파일 결과를 확인하는 데 최적화됨

Vue + Vite on StackBlitz

실제 Vite 개발 서버가 브라우저에서 실행되는 IDE와 유사한 환경

로컬 환경과 가장 유사함

버그를 신고할 때 재현 예시를 제공하기 위해서도 이러한 온라인 플레이그라운드를 사용하는 것이 좋습니다.

프로젝트 스캐폴딩

Vite

Vite는 Vue SFC를 1급으로 지원하는 가볍고 빠른 빌드 도구입니다. Vue의 저자인 Evan You가 만들었습니다!

Vite + Vue로 시작하려면 다음 명령어를 실행하세요:

npm

pnpm

yarn

bun

\$ npm create vue@latest

sh



Vite에 대해 더 알아보려면 [Vite 문서](#)를 참고하세요.

Vite 프로젝트에서 Vue 관련 동작(예: Vue 컴파일러에 옵션 전달 등)을 설정하려면 [@vitejs/plugin-vue](#) 문서를 참고하세요.

위에서 언급한 온라인 플레이그라운드 모두 Vite 프로젝트로 파일 다운로드를 지원합니다.

Vue CLI

Vue CLI는 Vue를 위한 공식 webpack 기반 도구 체인입니다. 현재는 유지보수 모드에 있으며, webpack 전용 기능이 꼭 필요하지 않다면 새로운 프로젝트는 Vite로 시작하는 것을 권장합니다. 대부분의 경우 Vite가 더 나은 개발 경험을 제공합니다.

Vue CLI에서 Vite로 마이그레이션하는 방법:

[VueSchool.io의 Vue CLI -> Vite 마이그레이션 가이드](#)
자동 마이그레이션을 도와주는 도구/플러그인

브라우저 내 템플릿 컴파일에 대한 참고 사항

빌드 단계를 거치지 않고 Vue를 사용할 때는 컴포넌트 템플릿을 페이지의 HTML에 직접 작성하거나 인라인 자바스크립트 문자열로 작성합니다. 이런 경우, Vue는 브라우저에서 즉시 템플릿을 컴파일하기 위해 템플릿 컴파일러를 함께 제공해야 합니다. 반면, 빌드 단계에서 미리 템플릿을 컴파일하면 컴파일러가 필요하지 않습니다. 클라이언트 번들 크기를 줄이기 위해 Vue는 다양한 "빌드"를 제공하여 각 용도에 맞게 최적화합니다.

`vue.runtime.*` 로 시작하는 빌드 파일은 **런타임 전용 빌드**입니다: 컴파일러가 포함되어 있지 않습니다. 이 빌드를 사용할 때는 모든 템플릿이 빌드 단계에서 미리 컴파일되어야 합니다.

`.runtime` 이 포함되지 않은 빌드 파일은 **풀 빌드**입니다: 컴파일러가 포함되어 있어 브라우저에서 직접 템플릿을 컴파일할 수 있습니다. 하지만 페이로드가 약 14kb 증가합니다.

기본 도구 설정은 SFC의 모든 템플릿이 미리 컴파일되므로 런타임 전용 빌드를 사용합니다. 어떤 이유로든 빌드 단계를 거치더라도 브라우저 내 템플릿 컴파일이 필요하다면, 빌드 도구에서 `vue` 를 `vue/dist/vue.esm-bundler.js` 로 별칭(alias) 설정하면 됩니다.

빌드 단계 없이 더 가벼운 대안을 찾고 있다면 [petite-vue](#)를 참고하세요.

IDE 지원



표현식 및 컴포넌트 props에 대한 구문 강조, TypeScript 지원, 인텔리센스를 제공합니다.

❗ TIP

Vue - Official은 Vue 2용 공식 VS Code 확장이었던 **Vetur**를 대체합니다. Vetur가 설치되어 있다면 Vue 3 프로젝트에서는 반드시 비활성화하세요.

WebStorm도 Vue SFC에 대한 훌륭한 내장 지원을 제공합니다.

Language Service Protocol (LSP)을 지원하는 다른 IDE들도 LSP를 통해 Volar의 핵심 기능을 활용할 수 있습니다:

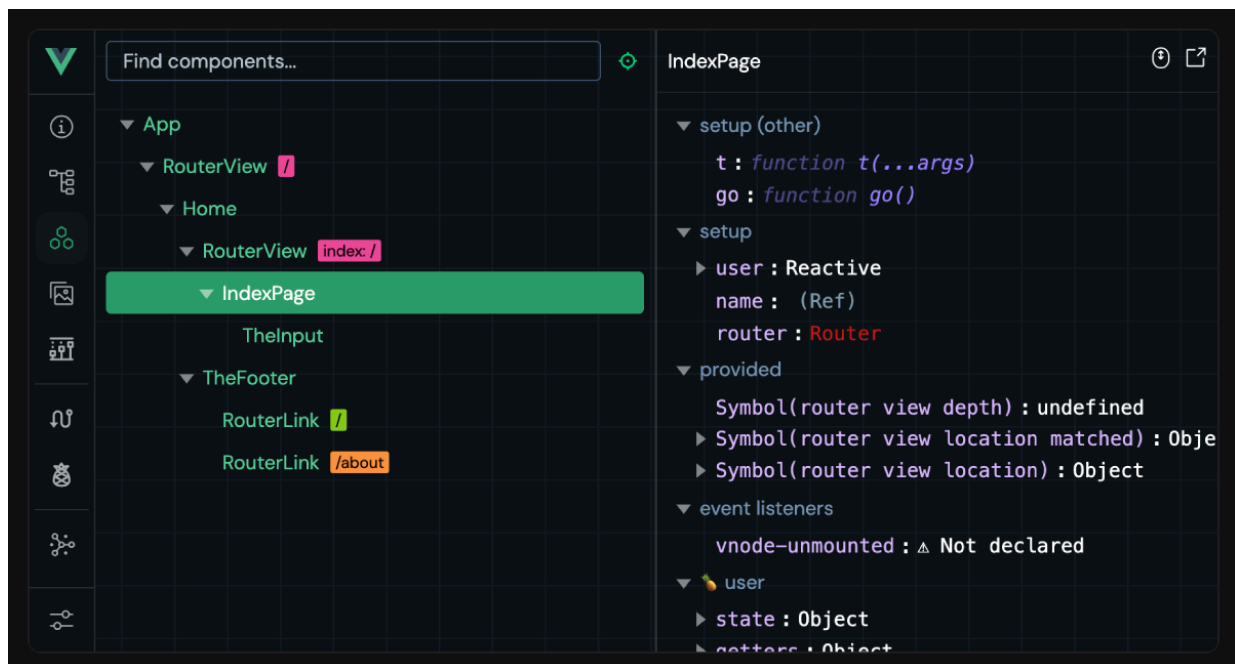
LSP-Volar를 통한 Sublime Text 지원.

coc-volar를 통한 vim / Neovim 지원.

lsp-mode를 통한 emacs 지원

브라우저 개발자 도구

Vue 브라우저 개발자 도구 확장 프로그램을 사용하면 Vue 앱의 컴포넌트 트리를 탐색하고, 개별 컴포넌트의 상태를 검사하며, 상태 관리 이벤트를 추적하고, 성능을 프로파일링할 수 있습니다.



문서

Chrome 확장 프로그램



TypeScript

주요 문서: [Vue와 TypeScript 사용하기](#).

Vue - Official 확장은 `<script lang="ts">` 블록을 사용하는 SFC에 대해 타입 검사, 템플릿 표현식 및 컴포넌트 간 props 검증을 제공합니다.

`vue-tsc` 를 사용하면 커맨드라인에서 동일한 타입 검사를 수행하거나 SFC용 `d.ts` 파일을 생성할 수 있습니다.

테스트

주요 문서: [테스트 가이드](#).

Cypress는 E2E 테스트에 권장됩니다. **Cypress Component Test Runner**를 통해 Vue SFC의 컴포넌트 테스트에도 사용할 수 있습니다.

Vitest는 Vue / Vite 팀원이 만든 테스트 러너로, 속도에 중점을 두고 있습니다. Vite 기반 애플리케이션에서 단위/컴포넌트 테스트에 즉각적인 피드백 루프를 제공합니다.

Jest는 **vite-jest**를 통해 Vite와 함께 사용할 수 있습니다. 하지만 기존 Jest 기반 테스트 스위트를 Vite 기반 환경으로 마이그레이션해야 하는 경우에만 권장하며, Vitest가 훨씬 효율적으로 유사한 기능을 제공합니다.

리닝

Vue 팀은 SFC 전용 리닝 규칙을 지원하는 **eslint-plugin-vue**와 **ESLint** 플러그인을 유지 관리하고 있습니다.

이전에 Vue CLI를 사용했던 사용자는 webpack 로더를 통해 린터가 설정되는 것에 익숙할 수 있습니다. 하지만 Vite 기반 빌드 환경에서는 다음을 권장합니다:



드를 따르세요.

2. **VS Code용 ESLint**와 같은 ESLint IDE 확장 프로그램을 설정하여 개발 중 에디터에서 바로 린터 피드백을 받을 수 있습니다. 이렇게 하면 개발 서버를 시작할 때 불필요한 린팅 비용도 줄일 수 있습니다.
3. 프로덕션 빌드 명령의 일부로 ESLint를 실행하여, 배포 전에 전체 린터 피드백을 받을 수 있습니다.
4. (선택 사항) **lint-staged**와 같은 도구를 설정하여 git 커밋 시 수정된 파일을 자동으로 린트할 수 있습니다.

포매팅

Vue - Official VS Code 확장은 Vue SFC에 대한 포매팅을 기본적으로 제공합니다.

또는 **Prettier**도 Vue SFC 포매팅을 기본 지원합니다.

SFC 커스텀 블록 통합

커스텀 블록은 동일한 Vue 파일에 서로 다른 요청 쿼리로 import되는 형태로 컴파일됩니다. 이러한 import 요청을 처리하는 것은 하위 빌드 도구의 몫입니다.

Vite를 사용하는 경우, 매칭되는 커스텀 블록을 실행 가능한 자바스크립트로 변환하는 커스텀 Vite 플러그인을 사용해야 합니다. [예시](#)

Vue CLI 또는 일반 webpack을 사용하는 경우, 매칭되는 블록을 변환하도록 webpack 로더를 설정해야 합니다. [예시](#)

하위 레벨 패키지

@vue/compiler-sfc

[문서](#)



니다. 메인 `vue` 패키지의 의존성으로 포함되어 있으며, `vue/compiler-sfc` 로 프록시되어 별도로 설치할 필요가 없습니다.

이 패키지는 Vue SFC를 처리하기 위한 하위 레벨 유틸리티를 제공하며, 커스텀 도구에서 Vue SFC를 지원해야 하는 도구 제작자를 위한 것입니다.

① TIP

항상 `vue/compiler-sfc` 디پ 임포트를 통해 이 패키지를 사용하는 것이 좋습니다. 이렇게 하면 Vue 런타임과 버전이 동기화됩니다.

@vitejs/plugin-vue

[문서](#)

Vite에서 Vue SFC 지원을 제공하는 공식 플러그인입니다.

vue-loader

[문서](#)

webpack에서 Vue SFC 지원을 제공하는 공식 로더입니다. Vue CLI를 사용하는 경우 **Vue CLI** 에서 `vue-loader` 옵션 수정 문서도 참고하세요.

기타 온라인 플레이그라운드

[VueUse Playground](#)

[Vue + Vite on Repl.it](#)

[Vue on CodeSandbox](#)

[Vue on Codepen](#)

[Vue on WebComponents.dev](#)

[✍️ GitHub에서 이 페이지 편집](#)

