



빠른 시작

온라인에서 Vue 체험하기

Vue를 빠르게 체험해보고 싶다면 [▶ Playground](#)에서 바로 사용해볼 수 있습니다.

빌드 과정 없이 순수 HTML 환경을 선호한다면, 이 [JSFiddle](#)을 시작점으로 사용할 수 있습니다.

Node.js와 빌드 도구 개념에 익숙하다면, [StackBlitz](#)에서 브라우저 내에서 완전한 빌드 환경을 바로 체험할 수 있습니다.

권장 설정에 대한 안내가 필요하다면, 첫 Vue 앱을 실행, 수정, 배포하는 방법을 보여주는 대화형 [Scrimba](#) 튜토리얼을 시청하세요.

Vue 애플리케이션 생성하기

① 사전 준비 사항

커맨드 라인 사용에 익숙할 것

Node.js 18.3 이상 버전 설치

이 섹션에서는 로컬 컴퓨터에서 Vue 싱글 페이지 애플리케이션을 스캐폴딩하는 방법을 소개합니다. 생성된 프로젝트는 **Vite**를 기반으로 한 빌드 환경을 사용하며, Vue 싱글 파일 컴포넌트 (SFC)를 사용할 수 있습니다.

최신 버전의 **Node.js**가 설치되어 있는지 확인하고, 현재 작업 디렉터리가 프로젝트를 생성하려는 위치인지 확인하세요. 커맨드 라인에서 다음 명령어를 실행하세요(\$ 기호는 입력하지 않습니다):



```
$ npm create vue@latest
```

sh

이 명령어는 공식 Vue 프로젝트 스캐폴딩 도구인 **create-vue**를 설치하고 실행합니다. TypeScript 및 테스트 지원과 같은 여러 선택적 기능에 대한 프롬프트가 표시됩니다:

```
✓ 프로젝트 이름: ... <your-project-name>
✓ TypeScript 추가? ... 아니오 / 예
✓ JSX 지원 추가? ... 아니오 / 예
✓ 싱글 페이지 애플리케이션 개발을 위한 Vue Router 추가? ... 아니오 / 예
✓ 상태 관리를 위한 Pinia 추가? ... 아니오 / 예
✓ 단위 테스트를 위한 Vitest 추가? ... 아니오 / 예
✓ 엔드 투 엔드 테스트 솔루션 추가? ... 아니오 / Cypress / Nightwatch / Playwright
✓ 코드 품질을 위한 ESLint 추가? ... 아니오 / 예
✓ 코드 포매팅을 위한 Prettier 추가? ... 아니오 / 예
✓ 디버깅을 위한 Vue DevTools 7 확장 프로그램 추가? (실험적) ... 아니오 / 예

./<your-project-name>에 프로젝트 스캐폴딩 중...
완료.
```

옵션이 확실하지 않다면, 일단 엔터를 눌러 **No** 를 선택하세요. 프로젝트가 생성되면, 의존성 설치 및 개발 서버 실행을 위한 안내에 따라 진행하세요:

npm pnpm yarn bun

```
$ cd <your-project-name>
$ npm install
$ npm run dev
```

sh

이제 첫 번째 Vue 프로젝트가 실행되고 있을 것입니다! 생성된 프로젝트의 예제 컴포넌트는 옵션 **API**가 아닌 **컴포지션 API**와 `<script setup>` 을 사용하여 작성되어 있습니다. 추가 팁은 다음과 같습니다:

권장 IDE 설정은 **Visual Studio Code + Vue** - 공식 확장 프로그램입니다. 다른 에디터를 사용한다면 **IDE 지원** 섹션을 참고하세요.

백엔드 프레임워크와의 통합 등 더 많은 도구 관련 정보는 **도구 가이드**에서 다룹니다.

빌드 도구 Vite에 대해 더 알고 싶다면 **Vite** 문서를 참고하세요.

TypeScript를 사용하기로 했다면 **TypeScript** 사용 가이드를 참고하세요.

앱을 프로덕션에 배포할 준비가 되면 다음 명령어를 실행하세요:

npm pnpm yarn bun



이 명령어는 프로젝트의 `./dist` 디렉터리에 프로덕션용 빌드를 생성합니다. 앱을 프로덕션에 배포하는 방법에 대해서는 [프로덕션 배포 가이드](#)를 참고하세요.

[다음 단계 >](#)

CDN에서 Vue 사용하기

스크립트 태그를 통해 CDN에서 직접 Vue를 사용할 수 있습니다:

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

html

여기서는 `unpkg`를 사용했지만, `jsdelivr`나 `cdnjs` 등 npm 패키지를 제공하는 다른 CDN도 사용할 수 있습니다. 물론 이 파일을 직접 다운로드하여 직접 서비스할 수도 있습니다.

CDN에서 Vue를 사용할 때는 "빌드 단계"가 필요하지 않습니다. 이로 인해 설정이 훨씬 간단해지며, 정적 HTML을 보강하거나 백엔드 프레임워크와 통합할 때 적합합니다. 하지만 싱글 파일 컴포넌트(SFC) 문법은 사용할 수 없습니다.

글로벌 빌드 사용하기

위 링크는 Vue의 `_글로벌 빌드_`를 로드하며, 모든 최상위 API가 전역 `Vue` 객체의 속성으로 노출됩니다. 다음은 글로벌 빌드를 사용하는 전체 예제입니다:

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

html

```
<div id="app">{{ message }}</div>
```

```
<script>
```

```
  const { createApp, ref } = Vue
```

```
  createApp({
```

```
    setup() {
```

```
      const message = ref('Hello vue!')
```

```
      return {
```

```
        message
```

```
      }
```

```
    }
```

```
  }).mount('#app')
```

```
</script>
```



① TIP

가이드 전반에 걸쳐 컴포지션 API 예제는 `<script setup>` 문법을 사용할 예정이며, 이는 빌드 도구가 필요합니다. 빌드 단계 없이 컴포지션 API를 사용하려면 `setup()` 옵션 사용법을 참고하세요.

ES 모듈 빌드 사용하기

이후 문서에서는 주로 **ES 모듈** 문법을 사용할 것입니다. 대부분의 최신 브라우저는 ES 모듈을 기본적으로 지원하므로, 다음과 같이 CDN에서 네이티브 ES 모듈로 Vue를 사용할 수 있습니다:

```
<div id="app">{{ message }}</div>                                     html

<script type="module">
  import { createApp, ref } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

  createApp({
    setup() {
      const message = ref('Hello Vue!')
      return {
        message
      }
    }
  }).mount('#app')
</script>
```

여기서는 `<script type="module">` 을 사용하고, 가져오는 CDN URL이 Vue의 **ES 모듈 빌드**를 가리키고 있다는 점에 주의하세요.

[CodePen 데모 >](#)

Import maps 활성화하기

위 예제에서는 전체 CDN URL에서 import하고 있지만, 이후 문서에서는 다음과 같은 코드를 자주 보게 될 것입니다:

```
import { createApp } from 'vue'                                     js
```

Import Maps를 사용하여 브라우저에 `vue` import 위치를 알려줄 수 있습니다:



```
{
  "imports": {
    "vue": "https://unpkg.com/vue@3/dist/vue.esm-browser.js"
  }
}
</script>

<div id="app">{{ message }}</div>

<script type="module">
  import { createApp, ref } from 'vue'

  createApp({
    setup() {
      const message = ref('Hello Vue!')
      return {
        message
      }
    }
  }).mount('#app')
</script>
```

CodePen 데모 >

다른 의존성도 import map에 추가할 수 있지만, 반드시 해당 라이브러리의 ES 모듈 버전을 가리키도록 해야 합니다.

① Import Maps 브라우저 지원

Import Maps는 비교적 새로운 브라우저 기능입니다. **지원 범위** 내의 브라우저를 사용해야 합니다. 특히 Safari는 16.4 이상에서만 지원됩니다.

⚠️ 프로덕션 사용 시 주의사항

지금까지의 예제는 Vue의 개발용 빌드를 사용하고 있습니다. CDN에서 Vue를 프로덕션에 사용할 계획이라면 **프로덕션 배포 가이드**를 반드시 참고하세요.

빌드 시스템 없이 Vue를 사용하는 것도 가능하지만, `jquery/jquery` (과거)나 `alpinejs/alpine` (현재)와 같은 상황에서는 `vuejs/petite-vue` 를 사용하는 것이 더 적합할 수 있습니다.

모듈 분리하기

가이드가 더 깊어질수록, 코드를 관리하기 쉽게 여러 자바스크립트 파일로 분리해야 할 수도 있습니다. 예를 들어:



```
<div id="app"></div>
```

html

```
<script type="module">
  import { createApp } from 'vue'
  import MyComponent from './my-component.js'

  createApp(MyComponent).mount('#app')
</script>
```

my-component.js

```
import { ref } from 'vue'
export default {
  setup() {
    const count = ref(0)
    return { count }
  },
  template: `<div>Count is: {{ count }}</div>`
}
```

js

위의 index.html 을 브라우저에서 직접 열면, ES 모듈은 file:// 프로토콜에서는 동작하지 않기 때문에 오류가 발생합니다. 브라우저가 로컬 파일을 열 때 사용하는 프로토콜이 바로 file:// 입니다.

보안상의 이유로, ES 모듈은 http:// 프로토콜에서만 동작합니다. 즉, 브라우저가 웹에서 페이지를 열 때 사용하는 프로토콜입니다. 로컬 컴퓨터에서 ES 모듈을 사용하려면, 반드시 index.html 을 http:// 프로토콜로 제공해야 하며, 이를 위해 로컬 HTTP 서버가 필요합니다.

로컬 HTTP 서버를 시작하려면, 먼저 **Node.js**가 설치되어 있는지 확인한 후, HTML 파일이 있는 디렉터리에서 커맨드 라인으로 `npx serve` 를 실행하세요. 정적 파일을 올바른 MIME 타입으로 제공할 수 있는 다른 HTTP 서버를 사용해도 됩니다.

가져온 컴포넌트의 템플릿이 자바스크립트 문자열로 인라인되어 있다는 점을 눈치챈을 수도 있습니다. VS Code를 사용한다면 **es6-string-html** 확장 프로그램을 설치하고, 문자열 앞에 `/*html*/` 주석을 붙이면 문법 하이라이팅을 받을 수 있습니다.

다음 단계

소개를 건너뛰었다면, 나머지 문서를 읽기 전에 꼭 읽어보시길 강력히 권장합니다.



가이드 계속하기

가이드는 프레임워크의 모든 측면을 자세히 안내합니다.

튜토리얼 체험하기

직접 실습하며 배우는 것을 선호하는 분들을 위한 코스입니다.

예제 살펴보기

핵심 기능과 일반적인 UI 작업 예제를 탐색해보세요.



[GitHub에서 이 페이지 편집](#)