빠른 시작

Try Vue Online

Vue를 빠르게 체험해 보려면 **Playground**에서 직접 사용해 보세요.

빌드 단계가 없는 일반 HTML 설정을 선호하는 경우, 이 **JSFiddle**을 시작점으로 사용할 수 있습니다.

Node.js와 빌드 도구의 개념에 이미 익숙하다면, **StackBlitz**에서 브라우저 내에서 바로 전체 빌드 설정을 시도해 볼 수도 있습니다.

Vue 애플리케이션 만들기

전제 조건

명령줄에 대한 친숙함
Node.js 버전 **18.3** 이상 설치

이 섹션에서는 로컬 컴퓨터에서 Vue 싱글 페이지 애플리케이션을 스캐폴드하는 방법을 소개합니다. 생성된 프로젝트는 **Vite**를 기반으로 빌드 설정을 사용하고 Vue 싱글 파일 컴포넌트(SFC)를 사용할 수 있도록 합니다.

최신 버전의 **Node.js**가 설치되어 있고 현재 작업 디렉터리가 프로젝트를 만들려는 디렉터리인지 확인하세요. 명령줄에서 다음 명령을 실행합니다($ 기호 제외):

npm pnpm yarn bun

sh

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

이 명령은 공식 Vue 프로젝트 스캐폴딩 도구인 **create-vue**를 설치 및 실행합니다. TypeScript 및 테스트 지원과 같은 몇 가지 선택적 기능에 대한 프롬프트가 표시됩니다:

```
✓ Project name: … <your-project-name>
✓ Add TypeScript? … No / Yes
✓ Add JSX Support? … No / Yes
✓ Add Vue Router for Single Page Application development? … No / Yes
✓ Add Pinia for state management? … No / Yes
✓ Add Vitest for Unit testing? … No / Yes
✓ Add an End-to-End Testing Solution? … No / Cypress / Nightwatch / Playwright
✓ Add ESLint for code quality? … No / Yes
✓ Add Prettier for code formatting? … No / Yes
✓ Add Vue DevTools 7 extension for debugging? (experimental) … No / Yes

Scaffolding project in ./<your-project-name>...
Done.
```

옵션에 대해 확신이 서지 않는다면 일단 엔터키를 눌러 No 를 선택하면 됩니다. 프로젝트가 생성되면 지침에 따라 종속 요소를 설치하고 개발 서버를 시작합니다:

npm pnpm yarn bun

sh

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

이제 첫 번째 Vue 프로젝트가 실행 중일 것입니다! 생성된 프로젝트의 예제 컴포넌트는 **Options API** 대신 `<script setup>`과 **Composition API**를 사용하여 작성되었음을 유의하세요. 다음은 몇 가지 추가 팁입니다:

- 권장 IDE 설정은 **Visual Studio Code** + **Volar extension**입니다. 다른 편집기를 사용하는 경우 **IDE** 지원 섹션을 확인하세요.
- 백엔드 프레임워크와의 통합을 포함한 자세한 툴링 내용은 툴링 가이드에서 확인할 수 있습니다.
- 기본 빌드 도구인 Vite에 대해 자세히 알아보려면 **Vite** 문서를 확인하세요.
- 타입스크립트를 사용하기로 선택한 경우 타입스크립트 사용 가이드를 확인하세요.

앱을 프로덕션으로 출시할 준비가 되면 다음을 실행하세요:

npm pnpm yarn bun

```sh
$ npm run build
```

이렇게 하면 프로젝트의 `./dist` 디렉터리에 프로덕션에 사용할 수 있는 앱 빌드가 생성됩니다. 프로덕션 배포 가이드](/guide/best-practices/production-deployment)를 확인하여 앱을 프로덕션에 배포하는 방법에 대해 자세히 알아보세요.

Next Steps >

CDN에서 Vue 사용

스크립트 태그를 통해 CDN에서 직접 Vue를 사용할 수 있습니다:

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

여기서는 **unpkg**를 사용하고 있지만, **jsdelivr** 또는 **cdnjs**와 같이 npm 패키지를 제공하는 모든 CDN을 사용할 수도 있습니다. 물론 이 파일을 다운로드하여 직접 제공할 수도 있습니다.

CDN에서 Vue를 사용하는 경우 "빌드 단계"가 필요하지 않습니다. 따라서 설정이 훨씬 간단해지며 정적 HTML을 향상시키거나 백엔드 프레임워크와 통합하는 데 적합합니다. 하지만 싱글 파일 컴포넌트(SFC) 구문은 사용할 수 없습니다.

글로벌 빌드 사용

위의 링크는 Vue의 _전역 빌드(global build)_를 불러옵니다. 이 빌드에서는 모든 최상위 API가 전역 `Vue` 객체의 속성으로 노출됩니다. 전역 빌드를 사용한 전체 예제는 다음과 같습니다:

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

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

<script>
 const { createApp, ref } = Vue

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

Codepen demo

TIP

가이드 전체에서 **Composition API**에 대한 많은 예제는 빌드 도구가 필요한 `<script setup>` 구문을 사용할 것입니다. 빌드 단계 없이 **Composition API**를 사용하려는 경우 `setup()` 옵션의 사용법을 참조하십시오.

ES 모듈 빌드 사용

이 문서의 나머지 부분에서는 주로 **ES** 모듈 구문을 사용할 것입니다. 현재 대부분의 최신 브라우저는 ES 모듈을 기본적으로 지원하므로 이와 같은 기본 ES 모듈을 통해 CDN에서 Vue를 사용할 수 있습니다:

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

<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>
```

`<script type="module">` 을 사용하고 있으며, 가져온 CDN URL이 대신 Vue의 **ES 모듈 빌드**를 가리키고 있음을 알 수 있습니다.

Codepen demo

임포트맵 활성화

위의 예에서는 전체 CDN URL에서 임포트하고 있지만 나머지 문서에서는 다음과 같은 코드를 볼 수 있습니다:

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

임포트 맵를 사용하여 브라우저에 `vue` 를 어디에서 가져와야 할지 알려줄 수 있습니다:

```html
<script type="importmap">
 {
  "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 demo
임포트 맵에 다른 종속성에 대한 항목을 추가할 수도 있지만 사용하려는 라이브러리의 ES 모듈 버전을 가리키는지 확인해야 합니다.

:::팁 임포트 맵 브라우저 지원 임포트 맵은 비교적 최신의 브라우저 기능입니다. 지원 범위 내에서 브라우저를 사용하는 것을 확인하십시오. 특히, Safari 16.4 이상에서만 지원됩니다. :::

> 운영 환경에서 사용
>
> 지금까지의 예제는 **Vue**의 개발 빌드를 사용한 것으로, 프로덕션 환경에서 **CDN**의 **Vue**를 사용하려면 프로덕션 배포 가이드를 확인하시기 바랍니다.
>
> 빌드 시스템 없이 **Vue**를 사용할 수는 있지만, 고려할 수 있는 대안적 접근 방법은 과거에 **jquery/jquery** 가 사용되었거나 현재 **alpinejs/alpine** 이 사용될 수 있는 컨텍스트에 더 적합할 수 있는 **vuejs/petite-vue** 를 사용하는 것입니다.

모듈 분할

가이드를 자세히 살펴볼수록 코드를 관리하기 쉽도록 별도의 JavaScript 파일로 분할해야 할 수도 있습니다. 예를 들어

html

```
<!-- index.html -->
<div id="app"></div>

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

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

js

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

위의 index.html 파일을 직접 브라우저에서 열면 ES 모듈이 file:// 프로토콜 위에서 작동하지 않아 오류가 발생합니다. 로컬 파일을 열 때 브라우저가 사용하는 프로토콜이기 때문입니다.

보안상의 이유로 ES 모듈은 웹 페이지를 열 때 브라우저가 사용하는 http:// 프로토콜에서만 작동할 수 있습니다. 로컬 컴퓨터에서 ES 모듈이 작동하려면 로컬 HTTP 서버를 사용하여 index.html 을 http:// 프로토콜로 서비스해야 합니다.

로컬 HTTP 서버를 시작하려면 먼저 **Node.js**가 설치되어 있는지 확인한 다음 명령줄에서 HTML 파일이 있는 동일한 디렉토리에서 npx serve 를 실행하십시오. 또는 올바른 MIME 유형으로 정적 파일을 서비스할 수 있는 다른 HTTP 서버를 사용할 수도 있습니다.

가져온 컴포넌트의 템플릿이 JavaScript 문자열로 인라인으로 포함되어 있는 것을 알아챘을 것입니다. VS Code를 사용하는 경우 **es6-string-html** 확장을 설치하고 문자열을 /*html*/ 주석으로 접두사를 붙여 문법 강조 효과를 얻을 수 있습니다.