Vue를 사용하는 다양한 방법

우리는 "모든 사람에게 맞는" 웹이란 없다고 믿습니다. 이것이 Vue가 유연하고 점진적으로 채택될 수 있도록 설계된 이유입니다. 사용 사례에 따라 Vue는 스택 복잡성, 개발자 경험 및 최종 성능 간의 최적 균형을 유지하기 위해 다양한 방식으로 사용될 수 있습니다.

독립 실행형 스크립트

Vue는 독립 실행형 스크립트 파일로 사용할 수 있습니다. 빌드 과정이 필요하지 않습니다! 백엔드 프레임워크가 이미 대부분의 HTML을 렌더링하고 있거나, 프론트엔드 로직이 빌드 과정을 정당화할 만큼 복잡하지 않은 경우, Vue를 스택에 통합하는 가장 쉬운 방법입니다. 이러한 경우, Vue가 jQuery를 대신한다고 생각해도 됩니다.

또한 Vue는 기존 HTML을 점진적으로 향상시키기 위해 특별히 최적화된 **petite-vue**라는 대체 배포판을 제공합니다. 더 작은 기능 세트이지만, 매우 가볍고 빌드 과정이 필요 없는 상황에서 더 효율적인 구현을 위해 사용합니다.

임베디드 웹 컴포넌트

Vue를 사용하여 렌더링 방법에 관계없이 모든 HTML 페이지에 포함할 수 있는 표준 웹 컴포넌트를 빌드 할 수 있습니다. 이 옵션을 사용하면 완전히 소비자에 구애받지 않는 방식으로 Vue를 활용할 수 있습니다. 즉, 웹 컴포넌트는 레거시 앱, 정적 HTML 또는 다른 프레임워크로 빌드된 앱에 포함될 수 있습니다.

SPA (싱글 페이지 웹)

일부 애플리케이션은 풍부한 상호 작용, 깊은 세션 깊이, 프런트엔드에서 사소하지 않은 상태 저장 로직이 필요합니다. 이러한 애플리케이션을 구축하는 가장 좋은 방법은 Vue가 전체 페이지를 제어할 뿐만 아니라 페이지를 다시 로드하지 않고도 데이터 업데이트 및 탐색을 처리하는 아키텍처를 사용하는 것입니다. 이러한 유형의 애플리케이션을 일반적으로 SPA(싱글 페이지 애플리케이션)라고 합니다.

Vue는 다음을 포함하여 최신 SPA를 구축하기 위한 핵심 라이브러리 및 종합 도구 지원을 제공합니다:

- 클라이언트 측 라우터
- 초고속 빌드 툴 체인
- IDE 지원
- 브라우저 개발 도구
- TypeScript 통합
- 테스트 유틸리티

SPA는 일반적으로 API 엔드포인트를 노출하는 백엔드가 필요하지만, Vue를 **Inertia.js**와 같은 솔루션과 페어링하여, 서버 중심 개발 모델을 유지하면서 SPA 이점을 얻을 수도 있습니다.

풀스택 / SSR

순수한 클라이언트 측 SPA는 앱이 SEO 및 컨텐츠 생성 시간에 민감한 경우 문제가 됩니다. 이는 브라우저가 대부분 비어 있는 HTML 페이지를 수신하고, 무엇을 렌더링하기 전에 JavaScript가 로드될 때까지 기다려야 하기 때문입니다.

Vue는 Vue 앱을 서버의 HTML 문자열로 "렌더링"하는 first-class API를 제공합니다. 이를 통해 서버는 이미 렌더링된 HTML을 보낼 수 있으므로, 최종 사용자는 JavaScript가 다운로드되는 동안 컨텐츠를 즉시 볼 수 있습니다. 그러면 Vue는 클라이언트 측에서 앱을 "하이드레이트"(hydrate: 변환)하여 대화형으로 만듭니다. 이를 서버 측 렌더링**(SSR)**이라고 하며, **Largest Contentful Paint(LCP)**와 같은 Core Web Vital 메트릭을 크게 향상시킵니다.

이 패러다임 위에 구축된 Vue 기반의 상위 레벨 프레임워크로 **Nuxt**가 있습니다. 이를 통해 Vue 및 JavaScript를 사용하여 풀스택 앱을 개발할 수 있습니다.

잼스택 / SSG

필요한 데이터가 정적 데이터인 경우 서버 측 렌더링을 미리 수행할 수 있습니다. 즉, 전체 애플리케이션을 HTML로 미리 렌더링하여 정적 파일로 제공할 수 있습니다. 이렇게 하면 사이트 성능이 향상되고 각 요청에 대해 페이지를 동적으로 렌더링할 필요가 없으므로 배포가 훨씬 간단해집니다. Vue는 여전히 이러한 애플리케이션을 하이드레이트하여 클라이언트에서 풍부한 상호 작용을 제공할 수 있습니다. 이 기술을 일반적으로 정적 사이트 생성(SSG)이라고 하며, 잼스택이라고도 합니다.

SSG에는 단일 페이지와 다중 페이지의 두 가지 버전이 있습니다. 두 가지 방식 모두 사이트를 정적 HTML로 미리 렌더링하지만 차이점이 있습니다:

> 초기 페이지 로드 후 단일 페이지 SSG는 페이지를 SPA로 "하이드레이트"합니다. 이렇게 하면 초기 JS 페이로드와 하이드레이션 비용이 더 많이 들지만, 전체 페이지를 다시 로드하는 대신 페이지 콘텐츠를 부분적으로만 업데이트하면 되므로 후속 탐색 속도가 빨라집니다.
>
> 다중 페이지 SSG는 모든 탐색에서 새 페이지를 로드합니다. 장점은 페이지에 상호 작용이 필요하지 않은 경우 최소한의 JS를 제공하거나 아예 JS를 제공하지 않을 수 있다는 것입니다! Astro](**https://astro.build/**)와 같은 일부 다중 페이지 SSG 프레임워크는 Vue 컴포넌트를 사용하여 정적 HTML 안에 대화형 "섬"을 만들 수 있는 "부분 수화"도 지원합니다.

사소하지 않은 상호 작용, 긴 세션 길이 또는 탐색 전반에 걸쳐 지속되는 요소/상태를 기대하는 경우 단일 페이지 SSG가 더 적합합니다. 그렇지 않은 경우 다중 페이지 SSG가 더 나은 선택입니다.

Vue 팀은 또한 지금 읽고 있는 이 웹사이트를 구동하는 정적 사이트 생성기인 **VitePress**를 관리합니다! VitePress는 두 가지 버전의 SSG를 모두 지원합니다. **Nuxt**도 SSG를 지원합니다. 동일한 Nuxt 앱에서 서로 다른 경로에 대해 SSR과 SSG를 혼합 할 수도 있습니다.

웹을 넘어서

Vue는 주로 웹 앱을 구축하기 위해 설계되었지만, 브라우저에만 국한되지는 않습니다. 다음을 수행할 수 있습니다:

Electron을 사용하여 데스크톱 앱 구축
Ionic Vue로 모바일 앱 구축
동일한 코드베이스로 데스크톱 및 모바일 앱 구축은 **Quasar** 또는 **Tauri**를 사용
TresJS로 3D WebGL 경험 구축
Vue의 **Custom Renderer API**를 사용하여 터미널용과 같은 사용자 정의 렌더러 구축