싱글 파일 컴포넌트

소개

Vue 싱글 파일 컴포넌트(일명 `*.vue` 파일, **SFC**로 약칭)는 Vue 컴포넌트의 템플릿, 로직, **그리고** 스타일링을 하나의 파일에 캡슐화할 수 있게 해주는 특별한 파일 형식입니다. 다음은 SFC의 예시입니다:

vue

```vue
<script setup>
import { ref } from 'vue'
const greeting = ref('Hello World!')
</script>

<template>
  <p class="greeting">{{ greeting }}</p>
</template>

<style>
.greeting {
  color: red;
  font-weight: bold;
}
</style>
```

보시다시피, Vue SFC는 고전적인 HTML, CSS, JavaScript 삼위일체의 자연스러운 확장입니다. `<template>`, `<script>`, `<style>` 블록은 컴포넌트의 뷰, 로직, 스타일링을 동일한 파일에 캡슐화하고 함께 배치합니다. 전체 문법은 SFC 문법 명세에 정의되어 있습니다.

SFC를 사용하는 이유

SFC는 빌드 단계가 필요하지만, 그에 따른 많은 이점이 있습니다:

본질적으로 결합된 관심사의 함께 배치
런타임 컴파일 비용 없는 사전 컴파일된 템플릿
컴포넌트 범위의 CSS
Composition API 사용 시 더 편리한 문법
템플릿과 스크립트를 교차 분석하여 더 많은 컴파일 타임 최적화
템플릿 표현식에 대한 자동 완성 및 타입 체크가 가능한 IDE 지원
기본적으로 핫 모듈 교체(HMR) 지원

SFC는 Vue 프레임워크의 대표적인 기능이며, 다음과 같은 상황에서 Vue를 사용할 때 권장되는 접근 방식입니다:

싱글 페이지 애플리케이션(SPA)
정적 사이트 생성(SSG)
더 나은 개발 경험(DX)을 위해 빌드 단계가 정당화될 수 있는 비트리비얼 프론트엔드

그렇다고 해도, SFC가 과하다고 느껴질 수 있는 상황이 있다는 점을 저희도 인지하고 있습니다. 그래서 Vue는 여전히 빌드 단계 없이 순수 JavaScript로도 사용할 수 있습니다. 주로 정적인 HTML에 가벼운 상호작용만 추가하려는 경우, 점진적 향상을 위해 최적화된 6kB의 Vue 서브셋인 petite-vue도 참고하실 수 있습니다.

동작 방식

Vue SFC는 프레임워크 전용 파일 형식이며, @vue/compiler-sfc에 의해 표준 JavaScript와 CSS로 사전 컴파일되어야 합니다. 컴파일된 SFC는 표준 JavaScript(ES) 모듈이므로, 적절한 빌드 설정이 있다면 SFC를 모듈처럼 import할 수 있습니다:

```js
import MyComponent from './MyComponent.vue'

export default {
  components: {
    MyComponent
  }
}
```

SFC 내부의 `<style>` 태그는 개발 중에는 핫 업데이트를 지원하기 위해 일반 `<style>` 태그로 주입됩니다. 프로덕션에서는 추출되어 하나의 CSS 파일로 병합될 수 있습니다.

SFC Playground에서 SFC를 실험해보고 컴파일 과정을 탐색할 수 있습니다.

실제 프로젝트에서는 보통 SFC 컴파일러를 Vite나 Vue CLI (이는 webpack 기반)과 같은 빌드 도구와 통합하여 사용하며, Vue는 SFC를 최대한 빠르게 시작할 수 있도록 공식 스캐폴딩 도구

관심사의 분리는 어떻게 되나요?

전통적인 웹 개발 배경을 가진 일부 사용자는 SFC가 서로 다른 관심사를 한 곳에 섞는 것에 대해 걱정할 수 있습니다. HTML/CSS/JS가 분리되어야 한다고 배웠기 때문입니다!

이 질문에 답하기 위해서는 **관심사의 분리는 파일 형식의 분리와 동일하지 않다**는 점에 동의하는 것이 중요합니다. 엔지니어링 원칙의 궁극적인 목표는 코드베이스의 유지보수성을 높이는 것입니다. 관심사의 분리를 파일 형식의 분리로 맹목적으로 적용하는 것은 점점 더 복잡해지는 프론트엔드 애플리케이션에서는 그 목표를 달성하는 데 도움이 되지 않습니다.

현대 UI 개발에서는 코드베이스를 서로 얽혀 있는 세 개의 거대한 레이어로 나누는 것보다, 느슨하게 결합된 컴포넌트로 나누고 이를 조합하는 것이 훨씬 더 합리적이라는 것을 알게 되었습니다. 컴포넌트 내부에서는 템플릿, 로직, 스타일이 본질적으로 결합되어 있으며, 이들을 함께 배치하는 것이 오히려 컴포넌트를 더 응집력 있고 유지보수하기 쉽게 만듭니다.

싱글 파일 컴포넌트의 아이디어가 마음에 들지 않더라도, Src Imports를 사용하여 JavaScript와 CSS를 별도의 파일로 분리함으로써 핫 리로딩 및 사전 컴파일 기능을 여전히 활용할 수 있습니다.