싱글 파일 컴포넌트

소개

Vue 싱글 파일 컴포넌트(Single-File Components: **SFC**, 일명 *.vue 파일)는 컴포넌트의 템플릿, 로직 및 스타일을 하나의 파일로 묶어낸 특수한 파일 형식입니다. 다음은 SFC 파일의 예입니다:

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 HTML, CSS 및 JavaScript 이 3개를 하나로 자연스럽게 합친 것입니다. <template>, <script> 및 <style> 블록은 하나의 파일에서 컴포넌트의 뷰, 로직 및 스타일을 캡슐화하고 배치합니다. 전체적인 문법은 **SFC** 문법 사양에 정의되어 있습니다.

왜 SFC를 사용해야 하나요

SFC 사용을 위해서는 빌드 방식을 따라야 하지만 다음과 같은 많은 이점이 있습니다:

친숙한 HTML, CSS 및 JavaScript 문법을 사용하여 모듈화된 컴포넌트 작성
본질적으로 사용 목적에 따라 구성됨
런타임 컴파일 비용이 없는 사전 컴파일된 템플릿
컴포넌트 범위 **CSS**
컴포지션 **API**로 작업할 때 더욱 인체공학적인 문법
템플릿과 스크립트를 교차 분석하여 컴파일 시간을 더욱 더최적화
템플릿 표현식을 지원하는 **IDE**의 자동 완성 및 유형 검사
즉시 사용 가능한 핫 모듈 교체(Hot-Module Replacement: HMR) 지원

SFC는 Vue를 프레임워크로 정의하며, 다음과 같이 Vue를 사용하는 데 권장되는 접근 방식입니다.

싱글 페이지 앱(Single Page Application: SPA)
정적 사이트 생성(Static-Site Generator: SSG)
더 나은 개발 경험(DX)을 위해 프론트엔드 개발 방식에 합리적으로 빌드 방식 도입.

SFC가 과도하다고 느껴질 수 있는 시나리오가 있다는 것을 알고 있습니다. 이것이 Vue가 빌드 방식이 아닌 일반 JavaScript를 통해 계속 사용될 수도 있도록 유지되는 이유입니다. 가벼운 상호 작용으로 정적인 HTML을 향상시키려는 대부분의 경우, 점진적 향상에 최적화된 6kb 크기의 Vue 하위 집합인 **petite-vue**를 확인할 수도 있습니다.

작동방식

Vue SFC는 프레임워크별 파일 형식이며, **@vue/compiler-sfc**를 통해 표준 JavaScript와 CSS로 미리 컴파일 되어있어야 합니다. 컴파일된 SFC는 표준 JavaScript(ES) 모듈입니다. 즉, 적절한 빌드 설정으로 SFC를 모듈처럼 가져올 수 있습니다:

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

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

SFC 내부의 `<style>` 태그는 일반적으로 핫 업데이트를 지원하기 위해 개발 중에는 기본 `<style>` 태그로 삽입됩니다. 프로덕션을 위해 단일 CSS 파일로 추출 및 병합할 수 있습니다.

Vue SFC 온라인 연습장에서 SFC로 플레이하고 컴파일 방법을 탐색할 수 있습니다.

실제 프로젝트에서는 일반적으로 SFC 컴파일러를 **Vite** 또는 **Vue CLI**와 같은 빌드 도구(**webpack**을 기반으로)와 통합합니다. Vue는 가능한 한 빨리 SFC를 시작할 수 있도록 공식 스캐폴딩 도구를 제공합니다. 자세한 내용은 **SFC** 도구 섹션에서 확인하세요.

관심사항의 분리에 대하여

일부 개발자는 전통적인 웹 개발 시, 본래 사용 목적의 성격에 따라 파일 타입이 분리되었던 HTML/CSS/JS를 SFC가 다시 한 곳에 혼합한다는 우려를 가질 수 있습니다!

이 우려에 대한 대답은 "**관심사항의 분리가 파일 유형의 분리와 동일한 것이 아니다는 관점으로 바라보는 것이 중요하다**"입니다. 엔지니어링 원칙의 궁극적인 목표는 코드베이스의 유지 관리 가능성을 개선하는 것입니다. 프론트엔드 앱의 사용 목적이 점점 더 복잡해짐에 따라, 파일 유형으로만 분리하게 될 경우, 위 목표(원칙)을 달성(유지)하는 데 도움이 되지 않습니다.

현대적인 UI 개발에서 우리는 코드베이스를 서로 얽혀 있는 세 개의 거대한 계층으로 나누는 대신 컴포넌트로 나누고 유연하게 결합하여 구성하는 것이 훨씬 더 합리적이라는 것을 발견했습니다. 컴포넌트 내부의 템플릿, 로직 및 스타일은 본질적으로 "동일한 사용 목적"으로 결합되어 있으며, 실제로 컴포넌트가 더 응집력 있고 유지 관리가 용이해집니다.

싱글 파일 컴포넌트 아이디어가 마음에 들지 않는다면, JavaScript와 CSS를 별도의 파일로 분리하여 **Src Imports** 방식으로 사용하여도 무관하며, 핫 리로딩 및 사전 컴파일 기능의 이점은 계속 활용할(누릴) 수 있습니다.