플러그인

소개

플러그인은 일반적으로 Vue에 앱 수준의 기능을 추가하는 독립적인 코드입니다. 플러그인을 설치하는 방법은 다음과 같습니다:

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

const app = createApp({})

app.use(myPlugin, {
  /* 선택적 옵션 */
})
```

플러그인은 `install()` 메서드를 노출하는 객체이거나, 설치 함수 자체로 동작하는 함수로 정의됩니다. 설치 함수는 앱 인스턴스와 함께, `app.use()` 에 전달된 추가 옵션(있을 경우)을 받습니다:

```js
const myPlugin = {
  install(app, options) {
    // 앱을 구성합니다
  }
}
```

플러그인에 엄격하게 정의된 범위는 없지만, 플러그인이 유용한 일반적인 시나리오는 다음과 같습니다:

1. 하나 이상의 전역 컴포넌트 또는 커스텀 디렉티브를 `app.component()` 및 `app.directive()` 로 등록합니다.

2. `app.provide()` 를 호출하여 앱 전체에서 주입 가능한 리소스를 만듭니다.

서드를 추가합니다.

4. 위의 조합이 필요한 라이브러리(예: vue-router).

플러그인 작성하기

자신만의 Vue.js 플러그인을 만드는 방법을 더 잘 이해하기 위해, `i18n` (국제화의 약자) 문자열을 표시하는 매우 단순화된 버전의 플러그인을 만들어보겠습니다.

먼저 플러그인 객체를 설정해봅시다. 아래와 같이 별도의 파일에 생성하고 내보내는 것이 로직을 분리하고 유지하는 데 권장됩니다.

plugins/i18n.js

```js
export default {
  install: (app, options) => {
    // 플러그인 코드는 여기에 작성합니다
  }
}
```

번역 함수를 만들고자 합니다. 이 함수는 점으로 구분된 `key` 문자열을 받아, 사용자로부터 제공받은 옵션에서 번역된 문자열을 찾아 반환합니다. 템플릿에서의 사용 예시는 다음과 같습니다:

```template
<h1>{{ $translate('greetings.hello') }}</h1>
```

이 함수가 모든 템플릿에서 전역적으로 사용 가능해야 하므로, 플러그인에서 `app.config.globalProperties` 에 추가하여 전역적으로 사용할 수 있게 만듭니다:

plugins/i18n.js

```js
export default {
  install: (app, options) => {
    // 전역적으로 사용 가능한 $translate() 메서드를 주입합니다
    app.config.globalProperties.$translate = (key) => {
      // `key`를 경로로 사용하여
      // `options`에서 중첩된 속성을 가져옵니다
      return key.split('.').reduce((o, i) => {
        if (o) return o[i]
      }, options)
    }
  }
}
```

서 해당 번역 값을 찾아 반환합니다.

번역 키가 포함된 객체는 `app.use()` 에 추가 매개변수로 플러그인 설치 시 전달해야 합니다:

```js
import i18nPlugin from './plugins/i18n'

app.use(i18nPlugin, {
  greetings: {
    hello: 'Bonjour!'
  }
})
```

이제, 초기 표현식 `$translate('greetings.hello')` 는 런타임에 `Bonjour!` 로 대체됩니다.

참고: 전역 속성 확장하기 TS

> ⓘ TIP
>
> 전역 속성은 가능한 한 적게 사용하세요. 여러 플러그인에서 주입된 전역 속성이 많아지면 앱 전체에서 혼란스러워질 수 있습니다.

플러그인에서 Provide / Inject 사용하기

플러그인을 사용하면 `provide` 를 통해 플러그인 사용자에게 함수나 속성에 접근할 수 있도록 할 수도 있습니다. 예를 들어, 애플리케이션이 번역 객체를 사용할 수 있도록 `options` 매개변수에 접근할 수 있게 할 수 있습니다.

plugins/i18n.js

```js
export default {
  install: (app, options) => {
    app.provide('i18n', options)
  }
}
```

이제 플러그인 사용자는 `i18n` 키를 사용하여 컴포넌트에서 플러그인 옵션을 주입할 수 있습니다:

```vue
<script setup>
import { inject } from 'vue'

const i18n = inject('i18n')
```

```
</script>
```

NPM용 번들링

플러그인을 빌드하여 다른 사람들이 사용할 수 있도록 배포하고 싶다면, Vite의 라이브러리 모드 섹션을 참고하세요.

GitHub에서 이 페이지 편집

Previous
커스텀 디렉티브

Next
Transition