플러그인

소개

플러그인은 일반적으로 Vue에 앱 레벨 기능을 추가하는 자체 코드입니다. 플러그인을 설치하는 방법은 다음과 같습니다:

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

const app = createApp({})

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

플러그인은 `install()` 메서드를 노출하는 객체 또는 단순히 install 함수 자체로 작동하는 함수로 정의됩니다. install 함수는 `app.use()` 에 전달된 추가 옵션과 함께 앱 인스턴스를 받습니다(있는 경우):

```js
const myPlugin = {
  install(app, options) {
    // 앱 환경설정
  }
}
```

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

- **app.component()** 및 **app.directive()** 를 사용하여 하나 이상의 전역 컴포넌트 또는 커스텀 디렉티브를 등록합니다.
- **app.provide()** 를 호출하여 앱 전체에 리소스를 주입 가능하게 만듭니다.
- 일부 전역 인스턴스 속성 또는 메서드를 **app.config.globalProperties** 에 첨부하여 추가합니다.
- 위 목록의 몇 가지를 조합해 무언가를 수행해야 하는 라이브러리(예: **vue-router**).

플러그인 작성하기

고유한 Vue.js 플러그인을 만드는 방법을 더 잘 이해하기 위해 `i18n` (**Internationalization**의 약어) 문자열을 표시하는 매우 간단한 버전의 플러그인을 만들 것입니다.

플러그인 객체를 설정하는 것으로 시작하겠습니다. 로직이 포함되고 분리된 상태로 유지하려면 아래와 같이 별도의 파일로 생성하여 내보내는 것이 좋습니다.

```js
// plugins/i18n.js
export default {
  install: (app, options) => {
    // 플러그인 코드는 여기로
  }
}
```

우리는 앱 전체에서 사용할 수 있는 키를 번역하는 기능을 만들고자 하므로 `app.config.globalProperties` 를 사용하여 이를 노출할 것입니다. 이 함수는 점으로 구분된 `key` 문자열을 수신하며, 커스텀 옵션에서 번역된 문자열을 찾는 데 사용할 것입니다.

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

이 함수는 모든 템플릿에서 전역적으로 사용할 수 있어야 하므로 플러그인의 `app.config.globalProperties` 에 이 함수를 첨부하여 그렇게 만들겠습니다:

```js
// plugins/i18n.js
export default {
  install: (app, options) => {
    // 전역적으로 사용 가능한 $translate() 메서드 주입
    app.config.globalProperties.$translate = (key) => {
      // `key`를 경로로 사용하여
      // `options`에서 중첩 속성을 검색합니다.
      return key.split('.').reduce((o, i) => {
        if (o) return o[i]
      }, options)
    }
  }
}
```

`$translate` 함수는 `greetings.hello` 와 같은 문자열을 받아 사용자가 제공한 구성을 살펴본 후 번역된 값을 반환합니다.

번역된 키가 포함된 객체는 설치 중에 `app.use()` 에 대한 추가 매개 변수를 통해 플러그인에 전달되어야 합니다:

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

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

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

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

참고: 전역 속성 타입 보완하기

> **TIP**
> 웬만하면 전역 속성은 사용하지 마십시오. 앱 전체에서 다른 플러그인에 의해 주입된 전역 속성이 너무 많이 사용되면 혼란스러워질 수 있기 때문입니다.

플러그인에서 Provide / Inject 활용하기

플러그인을 사용하면 `inject` 를 사용하여 플러그인 사용자에게 함수나 속성을 제공할 수도 있습니다. 예를 들어 앱이 `options` 매개변수에 접근하여 번역 객체를 사용할 수 있도록 허용할 수 있습니다.

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

플러그인 사용자는 이제 `i18n` 키를 사용하여 컴포넌트에 플러그인 옵션을 삽입할 수 있습니다:

vue

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

const i18n = inject('i18n')

console.log(i18n.greetings.hello)
</script>
```