Vue 애플리케이션 생성하기

애플리케이션 인스턴스

모든 Vue 애플리케이션은 `createApp` 함수로 새로운 **애플리케이션 인스턴스**를 생성하는 것에서 시작합니다:

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

const app = createApp({
  /* 루트 컴포넌트 옵션 */
})
```

루트 컴포넌트

`createApp` 에 전달하는 객체는 사실 컴포넌트입니다. 모든 앱은 자식 컴포넌트를 포함할 수 있는 "루트 컴포넌트"가 필요합니다.

싱글 파일 컴포넌트를 사용하는 경우, 일반적으로 루트 컴포넌트를 다른 파일에서 import합니다:

```js
import { createApp } from 'vue'
// 싱글 파일 컴포넌트에서 루트 컴포넌트 App을 import합니다.
import App from './App.vue'

const app = createApp(App)
```

이 가이드의 많은 예제에서는 단일 컴포넌트만 필요하지만, 대부분의 실제 애플리케이션은 중첩되고 재사용 가능한 컴포넌트 트리로 구성됩니다. 예를 들어, Todo 애플리케이션의 컴포넌트 트리는 다음과 같을 수 있습니다:

```
├─ TodoList
│  └─ TodoItem
│     ├─ TodoDeleteButton
│     └─ TodoEditButton
└─ TodoFooter
   ├─ TodoClearButton
   └─ TodoStatistics
```

가이드의 뒷부분에서는 여러 컴포넌트를 정의하고 조합하는 방법에 대해 다룰 것입니다. 그 전에, 단일 컴포넌트 내부에서 어떤 일이 일어나는지에 집중하겠습니다.

앱 마운트하기

애플리케이션 인스턴스는 `.mount()` 메서드가 호출되기 전까지 아무것도 렌더링하지 않습니다. 이 메서드는 "컨테이너" 인자를 필요로 하며, 실제 DOM 요소이거나 선택자 문자열일 수 있습니다:

```html
<div id="app"></div>
```

```js
app.mount('#app')
```

앱의 루트 컴포넌트의 내용은 컨테이너 요소 내부에 렌더링됩니다. 컨테이너 요소 자체는 앱의 일부로 간주되지 않습니다.

`.mount()` 메서드는 모든 앱 설정 및 에셋 등록이 완료된 후에 항상 호출해야 합니다. 또한, 에셋 등록 메서드와 달리 반환값이 애플리케이션 인스턴스가 아니라 루트 컴포넌트 인스턴스임을 유의하세요.

DOM 내 루트 컴포넌트 템플릿

루트 컴포넌트의 템플릿은 보통 컴포넌트 자체의 일부이지만, 마운트 컨테이너 내부에 직접 작성하여 별도로 템플릿을 제공할 수도 있습니다:

```html
<div id="app">
  <button @click="count++">{{ count }}</button>
</div>
```

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

```
  data() {
    return {
      count: 0
    }
  }
})

app.mount('#app')
```

루트 컴포넌트에 이미 `template` 옵션이 없다면, Vue는 자동으로 컨테이너의 `innerHTML` 을 템플릿으로 사용합니다.

DOM 내 템플릿은 빌드 단계 없이 Vue를 사용하는 애플리케이션에서 자주 사용됩니다. 또한 서버 사이드 프레임워크와 함께 사용할 수도 있으며, 이 경우 루트 템플릿이 서버에서 동적으로 생성될 수 있습니다.

앱 설정

애플리케이션 인스턴스는 몇 가지 앱 레벨 옵션을 설정할 수 있는 `.config` 객체를 제공합니다. 예를 들어, 모든 하위 컴포넌트에서 발생하는 오류를 포착하는 앱 레벨 오류 핸들러를 정의할 수 있습니다:

```js
app.config.errorHandler = (err) => {
  /* 오류 처리 */
}
```

애플리케이션 인스턴스는 앱 범위의 에셋을 등록할 수 있는 몇 가지 메서드도 제공합니다. 예를 들어, 컴포넌트를 등록할 수 있습니다:

```js
app.component('TodoDeleteButton', TodoDeleteButton)
```

이렇게 하면 `TodoDeleteButton` 을 앱 어디에서나 사용할 수 있습니다. 컴포넌트 및 기타 유형의 에셋 등록에 대해서는 가이드의 뒷부분에서 다룰 예정입니다. 또한 API 레퍼런스에서 애플리케이션 인스턴스 API 전체 목록을 확인할 수 있습니다.

앱을 마운트하기 전에 모든 앱 설정을 적용했는지 꼭 확인하세요!

여러 애플리케이션 인스턴스

동일한 페이지에서 하나의 애플리케이션 인스턴스만 사용할 필요는 없습니다. `createApp` API를 사용하면 여러 Vue 애플리케이션이 동일한 페이지에서 각각의 설정 및 전역 에셋 범위를 가지고 공존할 수 있습니다:

```js
const app1 = createApp({
  /* ... */
})
app1.mount('#container-1')

const app2 = createApp({
  /* ... */
})
app2.mount('#container-2')
```

Vue를 사용하여 서버 렌더링된 HTML을 향상시키고, 큰 페이지의 특정 부분만 Vue로 제어해야 하는 경우, 전체 페이지에 단일 Vue 애플리케이션 인스턴스를 마운트하는 것을 피하세요. 대신, 여러 개의 작은 애플리케이션 인스턴스를 생성하여 각각이 담당하는 요소에 마운트하세요.