

비동기 컴포넌트

기본 사용법

대형 애플리케이션에서는 앱을 더 작은 청크로 나누고, 필요할 때만 서버에서 컴포넌트를 로드해야 할 수 있습니다. 이를 가능하게 하기 위해 Vue는 `defineAsyncComponent` 함수를 제공합니다:

```
import { defineAsyncComponent } from 'vue'

const AsyncComp = defineAsyncComponent(() => {
  return new Promise((resolve, reject) => {
    // ...서버에서 컴포넌트 로드
    resolve(/* 로드된 컴포넌트 */)
  })
})
// ... `AsyncComp`을 일반 컴포넌트처럼 사용
```

js

보시다시피, `defineAsyncComponent` 는 Promise를 반환하는 로더 함수를 인자로 받습니다. 서버에서 컴포넌트 정의를 가져왔을 때 Promise의 `resolve` 콜백을 호출해야 합니다. 로드 실패했음을 나타내려면 `reject(reason)` 을 호출할 수도 있습니다.

ES 모듈 동적 `import`도 Promise를 반환하므로, 대부분의 경우 `defineAsyncComponent` 와 함께 사용합니다. Vite와 webpack 같은 번들러도 이 문법을 지원하며(번들 분할 지점으로 사용), 이를 통해 Vue SFC를 import할 수 있습니다:

```
import { defineAsyncComponent } from 'vue'

const AsyncComp = defineAsyncComponent(() =>
  import('./components/MyComponent.vue')
)
```

js

결과로 생성된 `AsyncComp` 는 실제로 페이지에 렌더링될 때만 로더 함수를 호출하는 래퍼 컴포넌트입니다. 또한, 모든 `props`와 `slot`을 내부 컴포넌트로 전달하므로, 비동기 래퍼를 사용해 원래 컴포넌트를 무리 없이 대체하면서 지연 로딩을 구현할 수 있습니다.



습니다:

```
app.component('MyComponent', defineAsyncComponent(() =>
  import('./components/MyComponent.vue')
))
```

js

부모 컴포넌트 내부에서 직접 정의할 수도 있습니다:

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

const AdminPage = defineAsyncComponent(() =>
  import('./components/AdminPageComponent.vue')
)
</script>

<template>
  <AdminPage />
</template>
```

vue

로딩 및 에러 상태

비동기 작업에는 불가피하게 로딩 및 에러 상태가 수반됩니다. `defineAsyncComponent()` 는 고급 옵션을 통해 이러한 상태를 처리할 수 있습니다:

```
const AsyncComp = defineAsyncComponent({
  // 로더 함수
  loader: () => import('./Foo.vue'),

  // 비동기 컴포넌트가 로딩 중일 때 사용할 컴포넌트
  loadingComponent: LoadingComponent,
  // 로딩 컴포넌트를 표시하기 전의 지연 시간. 기본값: 200ms.
  delay: 200,

  // 로드 실패했을 때 사용할 컴포넌트
  errorComponent: ErrorComponent,
  // 타임아웃이 지정되고 초과되면 에러 컴포넌트가 표시됩니다.
  // 기본값: Infinity.
  timeout: 3000
})
```

js

로딩 컴포넌트가 제공되면, 내부 컴포넌트가 로드되는 동안 먼저 표시됩니다. 로딩 컴포넌트가 표시되기 전 기본 200ms의 지연이 있는데, 이는 빠른 네트워크에서 즉시 로딩 상태가 너무 빨리



에러 컴포넌트가 제공되면, 로더 함수가 반환한 Promise가 거부(reject)될 때 표시됩니다. 요청이 너무 오래 걸릴 때 에러 컴포넌트를 표시하도록 타임아웃을 지정할 수도 있습니다.

자연 하이드레이션 3.5+

이 섹션은 서버 사이드 렌더링을 사용하는 경우에만 적용됩니다.

Vue 3.5+에서는 비동기 컴포넌트가 하이드레이션 전략을 제공하여 언제 하이드레이션할지 제어할 수 있습니다.

Vue는 여러 내장 하이드레이션 전략을 제공합니다. 이 내장 전략들은 사용하지 않을 경우 트리 셰이킹이 가능하도록 개별적으로 import해야 합니다.

설계는 유연성을 위해 의도적으로 저수준입니다. 향후 코어 또는 상위 레벨 솔루션(예: Nuxt)에서 컴파일러 문법 설탕이 추가될 수 있습니다.

Idle 시 하이드레이션

`requestIdleCallback` 을 통해 하이드레이션합니다:

```
import { defineAsyncComponent, hydrateOnIdle } from 'vue'

const AsyncComp = defineAsyncComponent({
  loader: () => import('./Comp.vue'),
  hydrate: hydrateOnIdle(/* 선택적으로 최대 타임아웃 전달 가능 */)
})
```

js

보일 때 하이드레이션

`IntersectionObserver` 를 통해 요소가 보일 때 하이드레이션합니다.

```
import { defineAsyncComponent, hydrateOnVisible } from 'vue'

const AsyncComp = defineAsyncComponent({
  loader: () => import('./Comp.vue'),
  hydrate: hydrateOnVisible()
})
```

js

옵저버를 위한 옵션 객체 값을 선택적으로 전달할 수 있습니다:



미디어 쿼리로 하이드레이션

지정한 미디어 쿼리가 일치할 때 하이드레이션합니다.

```
import { defineAsyncComponent, hydrateOnMediaQuery } from 'vue'

const AsyncComp = defineAsyncComponent({
  loader: () => import('./Comp.vue'),
  hydrate: hydrateOnMediaQuery('(max-width:500px)')
})
```

js

상호작용 시 하이드레이션

지정한 이벤트가 컴포넌트 요소에서 발생할 때 하이드레이션합니다. 하이드레이션이 완료되면 트리거된 이벤트도 재생됩니다.

```
import { defineAsyncComponent, hydrateOnInteraction } from 'vue'

const AsyncComp = defineAsyncComponent({
  loader: () => import('./Comp.vue'),
  hydrate: hydrateOnInteraction('click')
})
```

js

여러 이벤트 타입의 리스트도 전달할 수 있습니다:

```
hydrateOnInteraction(['wheel', 'mouseover'])
```

js

커스텀 전략

```
import { defineAsyncComponent, type HydrationStrategy } from 'vue'

const myStrategy: HydrationStrategy = (hydrate, forEachElement) => {
  // forEachElement는 컴포넌트의 비하이드레이션 DOM의
  // 모든 루트 요소를 순회하는 헬퍼입니다.
  // 루트가 단일 요소가 아닌 프래그먼트일 수 있기 때문입니다.
  forEachElement(e1 => {
    // ...
  })
  // 준비가 되면 `hydrate`를 호출
  hydrate()
}
```

ts



```
// 컴포넌트인 <Comp> 컴포넌트 관련
}
}

const AsyncComp = defineAsyncComponent({
  loader: () => import('./Comp.vue'),
  hydrate: myStrategy
})
```

Suspense와 함께 사용하기

비동기 컴포넌트는 내장 컴포넌트인 `<Suspense>` 와 함께 사용할 수 있습니다. `<Suspense>` 와 비동기 컴포넌트 간의 상호작용은 `<Suspense>` 전용 챕터에서 문서화되어 있습니다.

[🔗](#) [GitHub](#)에서 이 페이지 편집

[< Previous](#)

Provide / inject

[Next >](#)

컴포저블