lazy

`lazy`를 사용하면 컴포넌트가 처음 렌더링될 때까지 해당 컴포넌트의 코드를 로딩하는 것을 지연할 수 있습니다.

```
const SomeComponent = lazy(load)
```

레퍼런스

`lazy(load)`

`lazy`를 컴포넌트 외부에서 호출하여 지연 로딩된 React 컴포넌트를 선언하세요.

```
import { lazy } from 'react';

const MarkdownPreview = lazy(() => import('./MarkdownPreview.js'));
```

아래 예시를 참고하세요.

매개변수

- `load` : Promise 혹은 또 다른 *thenable* (`then` 메서드가 있는 Promise 유사 객체)을 반환하는 함수입니다. React는 반환된 컴포넌트를 처음 렌더링하려고 시도할 때까지 `load` 를 호출하지 않습니다. React가 처음으로 `load` 를 호출하면, 그것이 해결될 때까지 기다리고, 그 후 해결된 값의 `.default` 를 React 컴포넌트로 렌더링합니다. 반환된 Promise와 Promise의 해결된 값은 캐시되므로, React는 `load` 를 한 번만 호출합니다. 만약 Promise가 거부되면, React는 거부 이유를 가장 가까운 Error Boundary가 처리할 수 있도록 `throw` 합니다.

반환값

`lazy` 는 트리에 렌더링할 수 있는 React 컴포넌트를 반환합니다. 컴포넌트의 코드가 여전히 로딩되는 동안 렌더링을 시도하면 일시 중지됩니다. 로딩 중에 Loading Indicator를 표시하려면 `<Suspense>` 를 사용하세요.

`load` 함수

매개변수

`load` 는 매개변수를 받지 않습니다.

반환값

Promise 또는 다른 *thenable* (`then` 메서드가 있는 Promise 유사 객체)을 반환해야 합니다. 결국 `.default` 프로퍼티가 함수, `memo` , `forwardRef` 컴포넌트와 같은 유효한 React 컴포넌트 유형인 객체로 해석되어야 합니다.

사용법

Suspense와 지연 로딩 컴포넌트

일반적으로 정적 `import` 선언으로 컴포넌트를 가져옵니다.

```
import MarkdownPreview from './MarkdownPreview.js';
```

해당 컴포넌트 코드가 처음 렌더링 될 때까지 로드하는 것을 연기하려면 `import` 를 다음과 같이 대체합니다.

```
import { lazy } from 'react';

const MarkdownPreview = lazy(() => import('./MarkdownPreview.js'));
```

위의 코드는 동적 `import()` 에 의존하므로 번들러 또는 프레임워크의 지원이 필요할 수 있습니다. 이 패턴을 사용하려면 임포트하려는 `lazy` 컴포넌트가 `default` 내보내기로 내보내져 있어야 합니다.

이제 요청에 따라 컴포넌트의 코드가 로딩되므로, 로딩하는 동안 표시할 항목도 지정해야 합니다. `lazy` 컴포넌트 또는 해당 부모 컴포넌트 중 하나를 `<Suspense>` 경계Boundary로 감싸서 이 작업을 수행할 수 있습니다.

```
<Suspense fallback={<Loading />}>
  <h2>Preview</h2>
  <MarkdownPreview />
</Suspense>
```

이 예시에서 `MarkdownPreview` 코드는 렌더링을 시도할 때까지 로딩되지 않습니다. `MarkdownPreview` 가 아직 로딩되지 않는 경우에는 그 자리에 `Loading` 코드가 대신 표시됩니다. 체크박스를 선택해 보세요.

App.js Loading.js MarkdownPreview.js 새로고침 Clear 포크

```
import { useState, Suspense, lazy } from 'react';
import Loading from './Loading.js';

const MarkdownPreview = lazy(() => delayForDemo(import('./MarkdownPreview.js')));

export default function MarkdownEditor() {
  const [showPreview, setShowPreview] = useState(false);
  const [markdown, setMarkdown] = useState('Hello, **world**!');
  return (
    <>
```

```
          <textarea value={markdown} onChange={e => setMarkdown(e.target.value)} />
          <label>
```

자세히 보기

이 데모는 인위적인 지연으로 로딩됩니다. 다음에 체크박스를 선택 해제하고 다시 선택하면 `Preview`가 캐시 되어 로딩 상태가 되지 않습니다. 로딩 상태를 다시 보려면 샌드박스에서 “Reset”을 클릭하세요.

Suspense를 사용하여 로딩 상태를 관리하는 방법에 대해 자세히 알아보세요.

문제 해결

`lazy` 컴포넌트의 상태가 의도치 않게 재설정됩니다.

`lazy` 컴포넌트를 다른 컴포넌트 내부에서 선언하지 마세요.

```
import { lazy } from 'react';

function Editor() {
  // 🔴 잘못된 방법: 이렇게 하면 다시 렌더링할 때 모든 상태가 재설정됩니다.
  const MarkdownPreview = lazy(() => import('./MarkdownPreview.js'));
```

```
  // ...
}
```

대신 항상 모듈의 최상위 수준에서 선언하세요.

```
import { lazy } from 'react';

// ✅ 올바른 방법: `lazy` 컴포넌트를 컴포넌트 외부에 선언합니다.
const MarkdownPreview = lazy(() => import('./MarkdownPreview.js'));

function Editor() {
  // ...
}
```


이전
createContext

다음
memo

Meta Open Source

Copyright © Meta Platforms, Inc

uwu?