

Lazy loading in Next.js

nextjs lazyloading explain with example

Sure! Here's a simple explanation of **lazy loading** in **Next.js**, along with an example.

What is Lazy Loading?

Lazy loading means loading parts of your application **only when they are needed** instead of loading everything upfront. This improves performance, reduces the initial bundle size, and speeds up page loading.

In Next.js, you can lazy load React components dynamically using the `next/dynamic` function.

How does Next.js lazy loading work?

- Instead of importing a component directly, you use `dynamic` to import it asynchronously.
 - The component is loaded only when it is rendered for the first time.
 - You can also show a loading spinner or fallback while the component is loading.
-

Example: Lazy Loading a Component in Next.js

Step 1: Create a component to lazy load

Create a simple component, e.g., `HeavyComponent.js`:

```
jsx

// components/HeavyComponent.js
export default function HeavyComponent() {
  return (
    <div>
      <h2>This is a heavy component!</h2>
      <p>Loaded lazily on demand.</p>
    </div>
  )
}
```

```
);  
}
```

Step 2: Import it dynamically using `next/dynamic`

In your page or parent component, import it dynamically:

```
jsx  
  
import dynamic from 'next/dynamic';  
import { useState } from 'react';  
  
// Dynamically import HeavyComponent with a loading fallback  
const HeavyComponent = dynamic(() => import('../components/HeavyComponent'), {  
  loading: () => <p>Loading...</p>,  
  ssr: false, // optional: disables server-side rendering for this component  
});  
  
export default function Home() {  
  const [show, setShow] = useState(false);  
  
  return (  
    <div>  
      <h1>Next.js Lazy Loading Example</h1>  
      <button onClick={() => setShow(true)}>Load Heavy Component</button>  
  
      {/* Load component only when `show` is true */}  
      {show && <HeavyComponent />}  
    </div>  
  );  
}
```

Explanation:

- `dynamic(() => import('../components/HeavyComponent'))` tells Next.js to load this component only when it's rendered.
- The `loading` option shows a fallback while the component is being fetched.
- The `ssr: false` option disables server-side rendering for this component (useful if it relies on browser APIs or you want to load it only on the client).
- The `HeavyComponent` loads only when the user clicks the button.

What happens under the hood?

- The initial JS bundle sent to the client excludes `HeavyComponent` .
 - When the user clicks the button, Next.js fetches the `HeavyComponent` JS chunk separately.
 - The loading fallback (`Loading...`) is displayed until the chunk arrives.
 - Then the component renders.
-

If you want, I can help you with a runnable minimal repo or codesandbox! Would you like that?

if have app routing, then how can i handle

Great question! When you use **Next.js App Router** (the new `/app` directory introduced in Next.js 13+), lazy loading components dynamically works a bit differently but still uses `next/dynamic` .

Lazy Loading with Next.js App Router (`/app`)

In the App Router, you usually work with **React Server Components (RSC)** by default. However, **dynamic imports** for lazy loading client components still work the same way.

Important:

- Only **Client Components** can be lazy loaded dynamically.
 - So, you need to mark components with `"use client"` at the top to make them Client Components.
 - Then use `dynamic` to lazy load those client components.
-

Example with App Router

Step 1: Create a client component that you want to lazy load

```
jsx
```

```
// app/components/HeavyComponent.jsx
"use client";

export default function HeavyComponent() {
  return (
    <div>
      <h2>This is a heavy component!</h2>
      <p>Loaded lazily on demand.</p>
    </div>
  );
}
```

Step 2: Use `dynamic` in a Client Component/page to lazy load

Since you can only use hooks (like `useState`) inside Client Components, your page or parent must also be a Client Component or you need to create a Client wrapper component.

```
jsx

// app/page.jsx or app/page.tsx
"use client";

import dynamic from 'next/dynamic';
import { useState } from 'react';

// Dynamically import with loading fallback
const HeavyComponent = dynamic(() => import('./components/HeavyComponent'), {
  loading: () => <p>Loading...</p>,
  ssr: false,
});

export default function Page() {
  const [show, setShow] = useState(false);

  return (
    <main>
      <h1>Next.js App Router Lazy Loading Example</h1>
      <button onClick={() => setShow(true)}>Load Heavy Component</button>

      {show && <HeavyComponent />}
    </main>
  );
}
```

```
};  
}
```

Notes:

- The page (or parent) here uses `"use client"` because it needs to use `useState` to control showing the lazy-loaded component.
 - `HeavyComponent` is also a Client Component because it uses `"use client"`.
 - You can lazy load any client component this way.
 - The `ssr: false` option prevents server-side rendering for that component chunk (optional, but useful).
-

Summary for App Router Lazy Loading

Step	Detail
1. Mark lazy component with <code>"use client"</code>	Make sure the component to lazy load is a Client Component.
2. Use <code>dynamic</code> from <code>next/dynamic</code> to import	Use dynamic import with fallback UI.
3. Use a Client Component as parent/page	Because hooks (like <code>useState</code>) are needed for conditional rendering.

If you want, I can help with a full minimal example in App Router structure! Would you like that?

2/2