Custom Errors

404 Page

A 404 page may be accessed very often. Server-rendering an error page for every visit increases the load of the Next.js server. This can result in increased costs and slow experiences.

To avoid the above pitfalls, Next.js provides a static 404 page by default without having to add any additional files.

Customizing The 404 Page

To create a custom 404 page you can create a `pages/404.js` file. This file is statically generated at build time.

pages/404.js

```js
export default function Custom404() {
  return <h1>404 - Page Not Found</h1>;
}
```

Note: You can use `getStaticProps` inside this page if you need to fetch data at build time.

500 Page

Server-rendering an error page for every visit adds complexity to responding to errors. To help users get responses to errors as fast as possible, Next.js provides a static 500 page by default without having to add any additional files.

Customizing The 500 Page

To customize the 500 page you can create a `pages/500.js` file. This file is statically generated at build time.

pages/500.js

```js
export default function Custom500() {
  return <h1>500 - Server-side error occurred</h1>;
}
```

Note: You can use `getStaticProps` inside this page if you need to fetch data at build time.

More Advanced Error Page Customizing

500 errors are handled both client-side and server-side by the `Error` component. If you wish to override it, define the file `pages/_error.js` and add the following code:

```js
function Error({ statusCode }) {
  return (
    <p>
      {statusCode
        ? `An error ${statusCode} occurred on server`
        : 'An error occurred on client'}
    </p>
  );
}

Error.getInitialProps = ({ res, err }) => {
  const statusCode = res ? res.statusCode : err ? err.statusCode : 404;
  return { statusCode };
};

export default Error;
```

`pages/_error.js` is only used in production. In development you'll get an error with the call stack to know where the error originated from.

Reusing the built-in error page

If you want to render the built-in error page you can by importing the `Error` component:

```js
import Error from 'next/error';

```

```
export async function getServerSideProps() {
  const res = await fetch('https://api.github.com/repos/vercel/next.js');
  const errorCode = res.ok ? false : res.status;
  const json = await res.json();

  return {
    props: { errorCode, stars: json.stargazers_count },
  };
}

export default function Page({ errorCode, stars }) {
  if (errorCode) {
    return <Error statusCode={errorCode} />;
  }

  return <div>Next stars: {stars}</div>;
}
```

The `Error` component also takes `title` as a property if you want to pass in a text message along with a `statusCode`.

If you have a custom `Error` component be sure to import that one instead. `next/error` exports the default component used by Next.js.

Caveats

- `Error` does not currently support Next.js Data Fetching methods like `getStaticProps` or `getServerSideProps`.
- `_error`, like `_app`, is a reserved pathname. `_error` is used to define the customized layouts and behaviors of the error pages. `/_error` will render 404 when accessed directly via routing or rendering in a custom server.