

Error Handling

This documentation explains how you can handle development, server-side, and client-side errors.

Handling Errors in Development

When there is a runtime error during the development phase of your Next.js application, you will encounter an **overlay**. It is a modal that covers the webpage. It is **only** visible when the development server runs using `next dev` via `pnpm dev`, `npm run dev`, or `yarn dev` and will not be shown in production. Fixing the error will automatically dismiss the overlay.

Here is an example of an overlay:

Failed to compile

```
./pages/index.js:5:0
Module not found: Can't resolve '../components/header'
 3 | import styles from '../styles/Home.module.css';
 4 |
> 5 | import Header from '../components/header';
    |                                     ^
 6 |
 7 | export default function Home() {
 8 |   return (
```

<https://nextjs.org/docs/messages/module-not-found>

This error occurred during the build process and can only be dismissed by fixing the error.

Handling Server Errors

Next.js provides a static 500 page by default to handle server-side errors that occur in your application. You can also [customize this page](#) by creating a `pages/500.js` file.

Having a 500 page in your application does not show specific errors to the app user.

You can also use [404 page](#) to handle specific runtime error like `file not found`.

Handling Client Errors

React [Error Boundaries](#) is a graceful way to handle a JavaScript error on the client so that the other parts of the application continue working. In addition to preventing the page from crashing, it allows you to provide a custom fallback component and even log error information.

To use Error Boundaries for your Next.js application, you must create a class component `ErrorBoundary` and wrap the `Component` prop in the `pages/_app.js` file. This component will be responsible to:

- Render a fallback UI after an error is thrown
- Provide a way to reset the Application's state
- Log error information

You can create an `ErrorBoundary` class component by extending `React.Component`. For example:

```
1  class ErrorBoundary extends React.Component {
2    constructor(props) {
3      super(props);
4
5      // Define a state variable to track whether is an error or not
6      this.state = { hasError: false };
7    }
8    static getDerivedStateFromError(error) {
9      // Update state so the next render will show the fallback UI
10
11      return { hasError: true };
12    }
13    componentDidCatch(error, errorInfo) {
14      // You can use your own error logging service here
15      console.log({ error, errorInfo });
16    }
17    render() {
18      // Check if the error is thrown
19      if (this.state.hasError) {
20        // You can render any custom fallback UI
21        return (
22          <div>
23            <h2>Oops, there is an error!</h2>
```

```

24     <button
25         type="button"
26         onClick={() => this.setState({ hasError: false })}
27     >
28         Try again?
29     </button>
30 </div>
31 );
32 }
33
34 // Return children components in case of no error
35
36 return this.props.children;
37 }
38 }
39
40 export default ErrorBoundary;

```

The `ErrorBoundary` component keeps track of an `hasError` state. The value of this state variable is a boolean. When the value of `hasError` is `true`, then the `ErrorBoundary` component will render a fallback UI. Otherwise, it will render the children components.

After creating an `ErrorBoundary` component, import it in the `pages/_app.js` file to wrap the `Component` prop in your Next.js application.

```

1 // Import the ErrorBoundary component
2 import ErrorBoundary from '../components/ErrorBoundary';
3
4 function MyApp({ Component, pageProps }) {
5     return (
6         // Wrap the Component prop with ErrorBoundary component
7         <ErrorBoundary>
8             <Component {...pageProps} />
9         </ErrorBoundary>
10    );
11 }
12
13 export default MyApp;

```

You can learn more about [Error Boundaries](#) in React's documentation.

Reporting Errors

To monitor client errors, use a service like [Sentry](#), [Bugsnag](#) or [Datadog](#).