Code Splitting

Instead of downloading the entire app before users can use it, code splitting allows you to split your code into small chunks which you can then load on demand.

This project setup supports code splitting via dynamic `import()`. Its proposal is in stage 4. The `import()` function-like form takes the module name as an argument and returns a `Promise` which always resolves to the namespace object of the module.

Here is an example:

`moduleA.js`

```
const moduleA = 'Hello';

export { moduleA };
```

`App.js`

```
import React, { Component } from 'react';

class App extends Component {
  handleClick = () => {
    import('./moduleA')
      .then(({ moduleA }) => {
        // Use moduleA
      })
      .catch(err => {
        // Handle failure
      });
  };

  render() {
    return (
      <div>
        <button onClick={this.handleClick}>Load</button>
      </div>
    );
  }
}

export default App;
```

This will make `moduleA.js` and all its unique dependencies as a separate chunk that only loads after the user clicks the 'Load' button. For more information on the chunks that are created, see the production build section.

You can also use it with `async` / `await` syntax if you prefer it.

With React Router

If you are using React Router check out this tutorial

Also check out the Code Splitting section in React documentation.