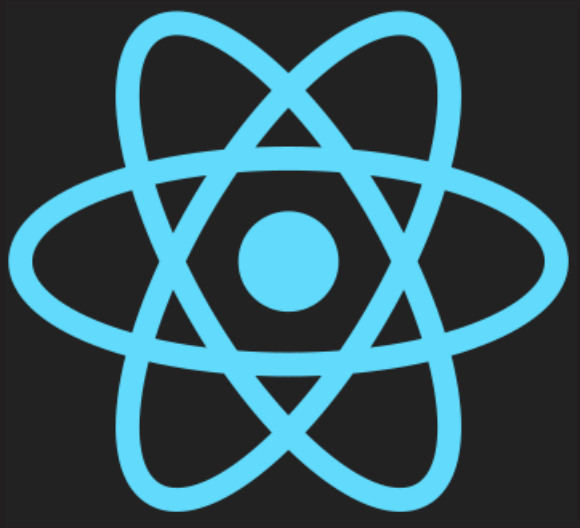
REACT

Code Splitting & Suspense

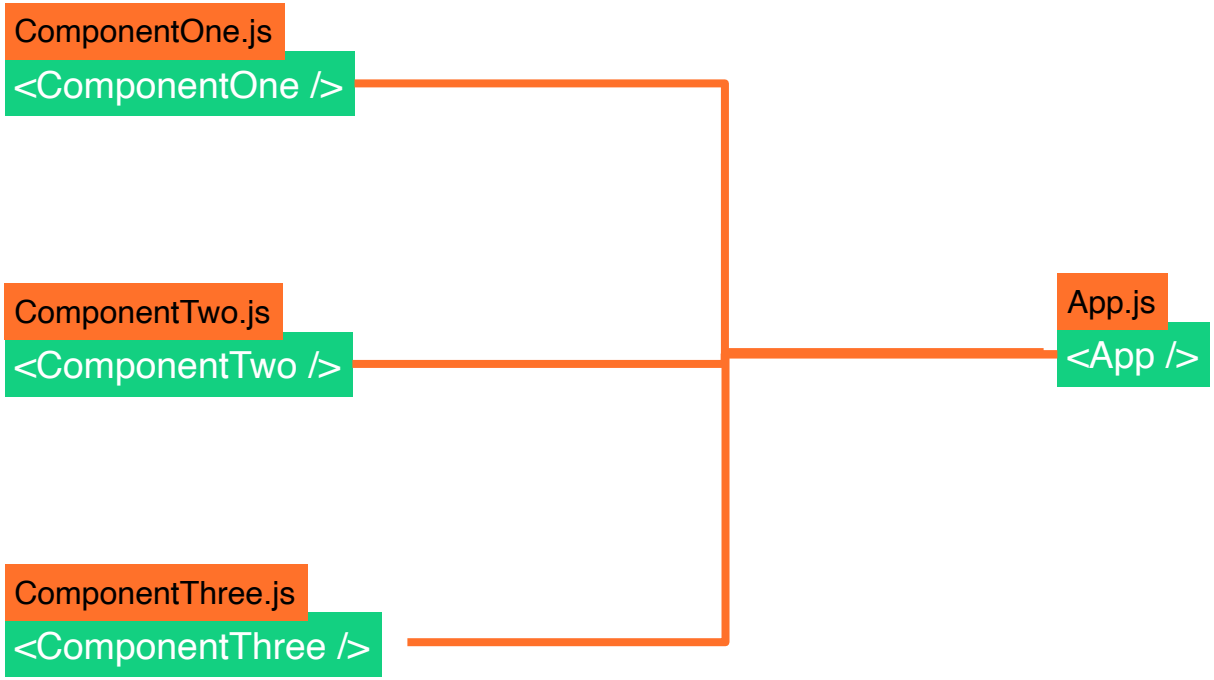


LEARNING OBJECTIVES

- Code Splitting – Producing on-demand code bundles
- React.lazy()
- Suspense component
- Loadable Components

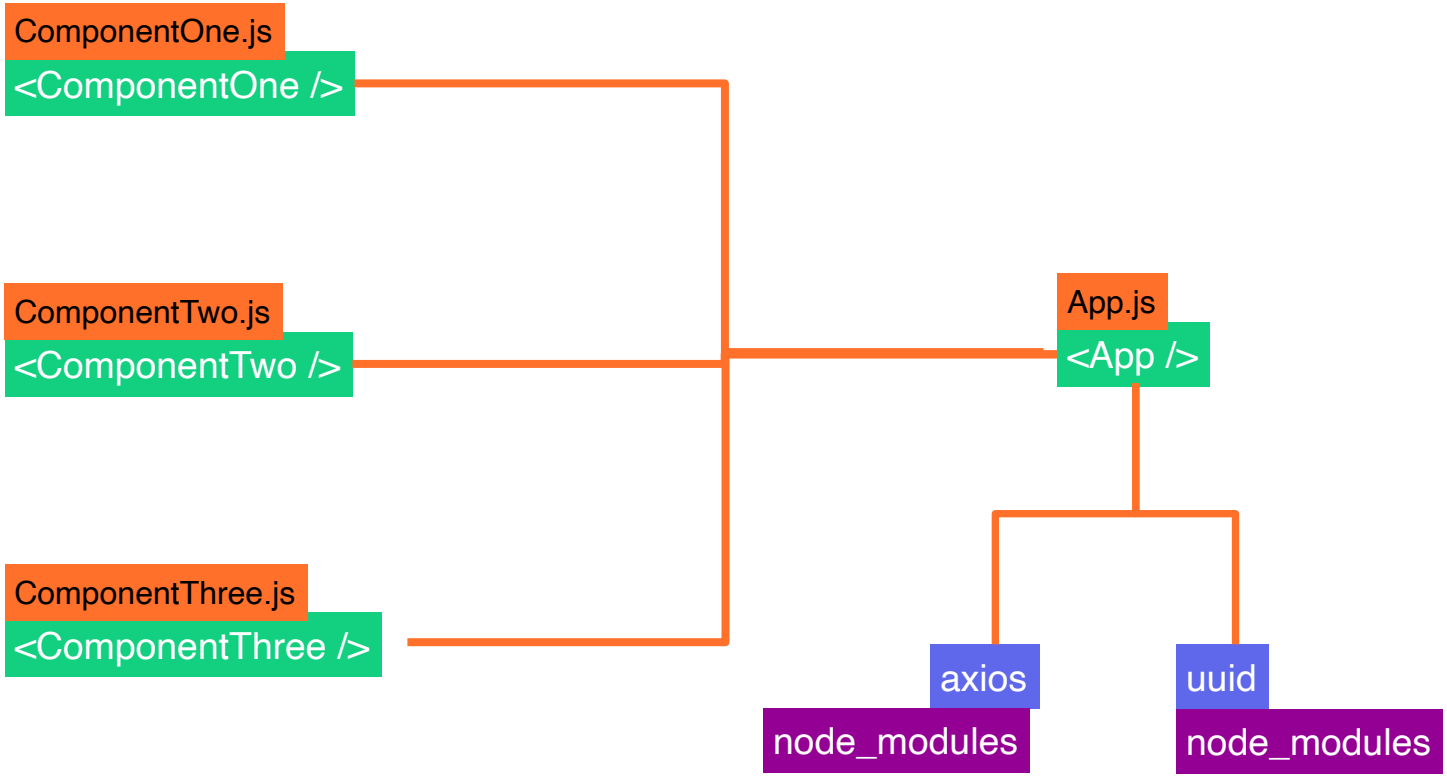
CODE SPLITTING

CODE SPLITTING



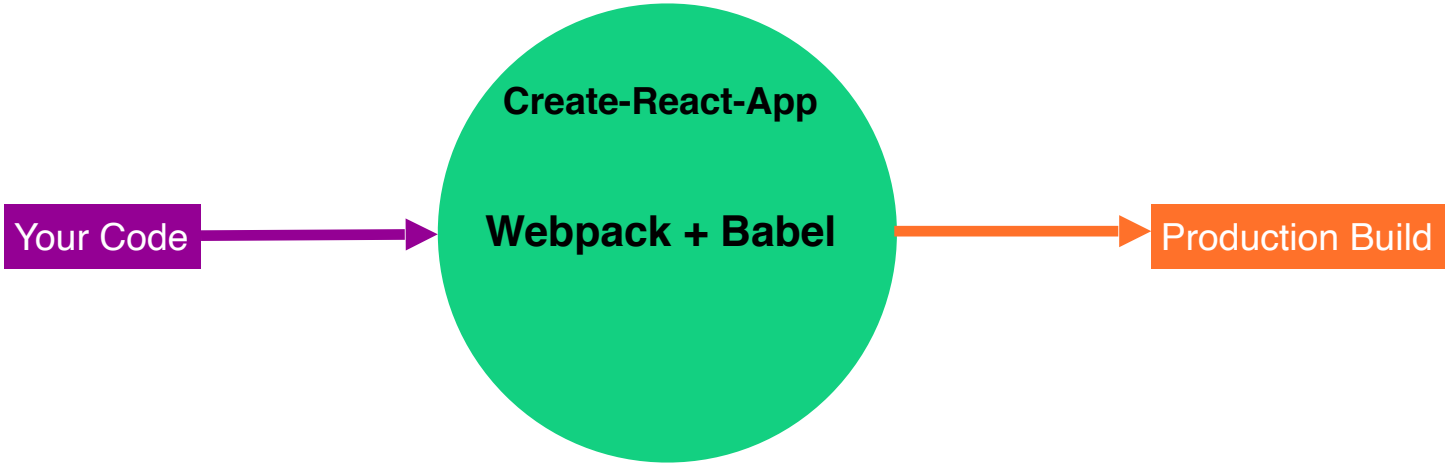
React apps are made of components spread across files

CODE SPLITTING



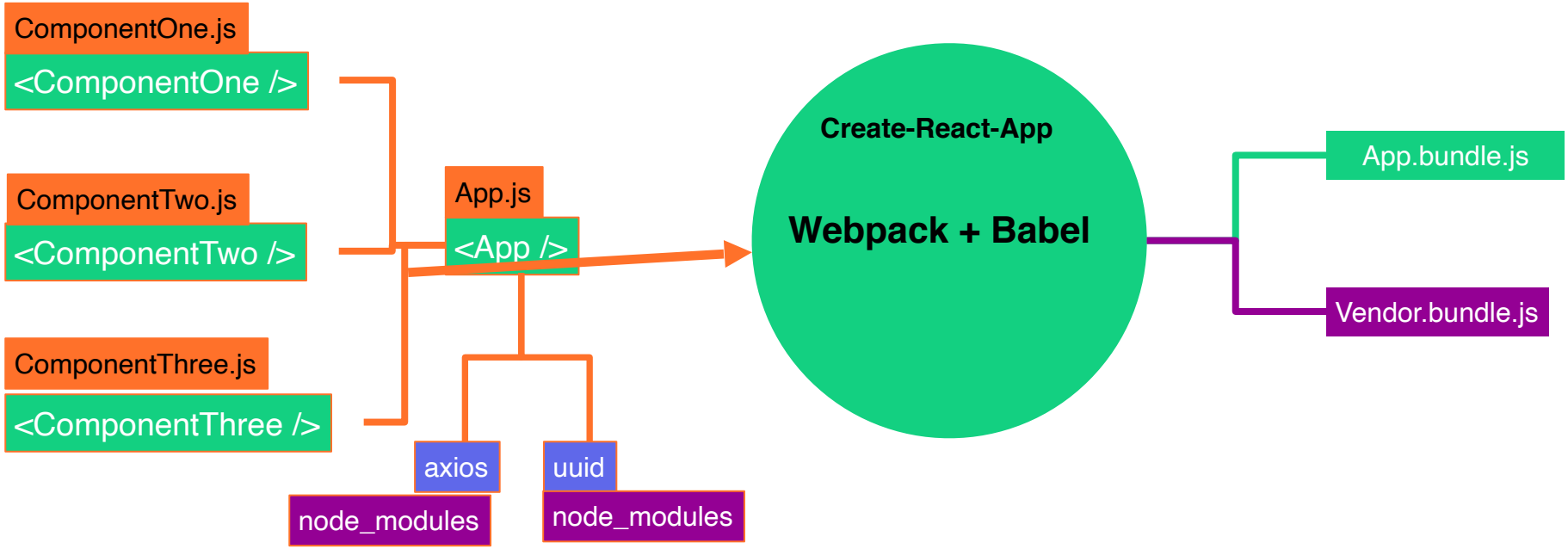
React apps are made of components spread across files

CODE SPLITTING



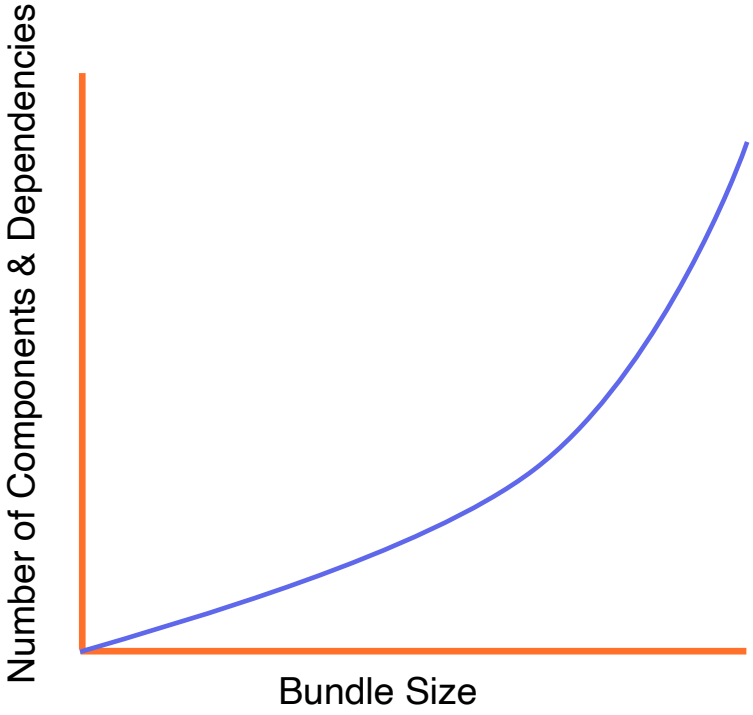
Module bundlers such as webpack at work

CODE SPLITTING



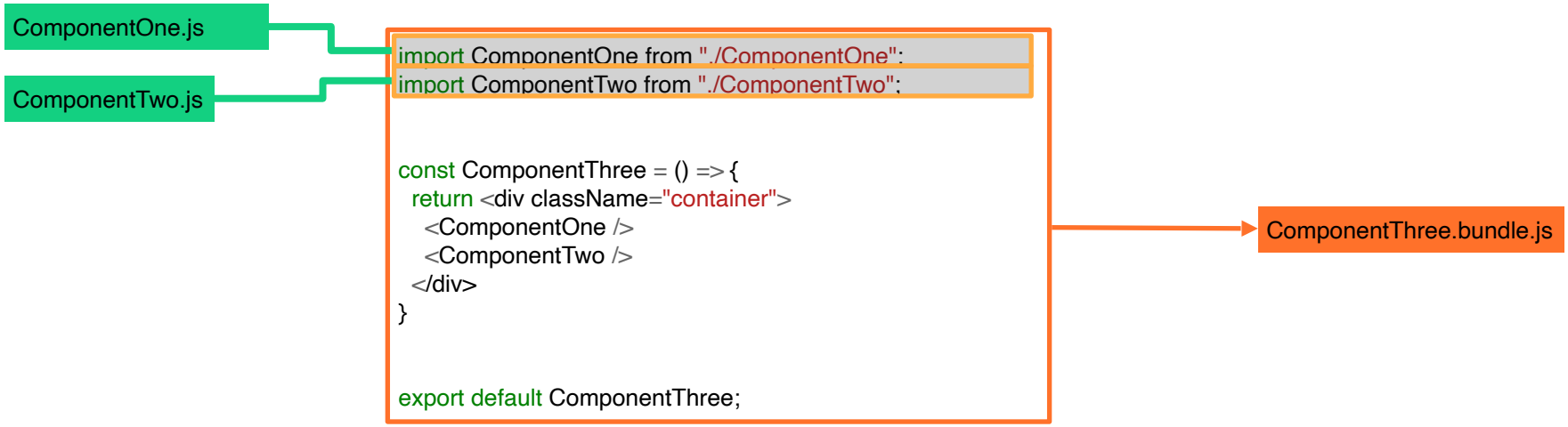
Producing an optimized production build

CODE SPLITTING



As your app grows & evolves, so will the bundle size

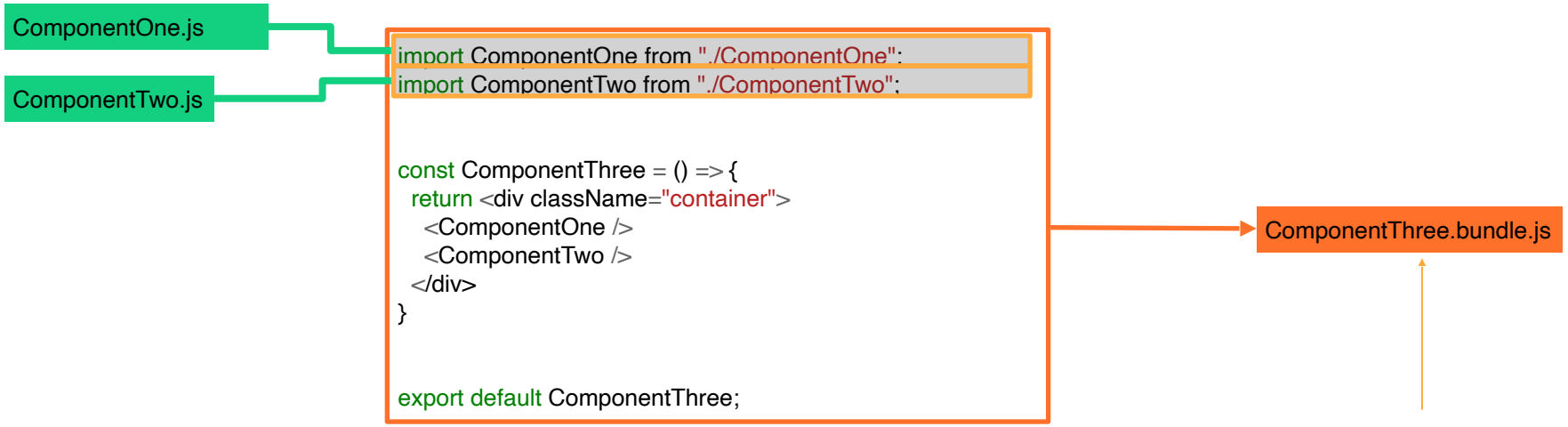
CODE SPLITTING



The ‘import’ statement

CODE SPLITTING

The ‘import’ statement



- If this bundle is big, then it delay the execution of your app
- With a lot of components, this problem only aggravates!

CODE SPLITTING

SPLIT CODE INTO MULTIPLE BUNDLES

REACT LAZY

REACT LAZY

```
import ComponentOne from "./ComponentOne";
```

Static Import statements load up modules before execution begins

REACT LAZY

```
import("./ComponentOne")
  .then(module => {
    // Use the module
  })
  .catch(error => {
    // Manage errors
  });
```

ECMAScript Draft Proposal (As of Late 2019)

Dynamic Import Expressions

REACT LAZY

```
import("./ComponentOne")
  .then(module => {
    // Use the module
  })
  .catch(error => {
    // Manage errors
  });
```

ECMAScript Draft Proposal (As of Late 2019)

Toolchains such as Webpack + Babel will produce separate chunks for dynamically imported components

Dynamic Import Expressions

REACT LAZY

```
import("./ComponentOne")
  .then(module => {
    // Use the module
  })
  .catch(error => {
    // Manage errors
  });
```

ECMAScript Draft Proposal (As of Late 2019)

Toolchains such as Webpack + Babel will produce separate chunks for dynamically imported components

Helps keep your bundle size small so your app loads up faster

Dynamic Import Expressions

REACT LAZY

Lazy Loading Components

```
const MyComponent = React.lazy(() => import("./MyComponent"));
```

SUSPENSE COMPONENT

SUSPENSE COMPONENT

```
const MyComponent = React.lazy(() => import("./MyComponent"));
```

```
<Suspense fallback={<h1>Loading...</h1>}>
  <MyComponent />
</Suspense>
```

Lazy Loading Components & Displaying a fallback UI

SUSPENSE COMPONENT

```
const MyComponent = React.lazy(() => import("./MyComponent"));
```

Note :: We're only talking about lazy loading component code and NOT data fetching from APIs within components

```
<Suspense fallback={<h1>Loading...</h1>}>
  <MyComponent />
</Suspense>
```

Lazy Loading Components & Displaying a fallback UI

SUSPENSE COMPONENT

- Code splitting & Lazy loading components
- Components are loaded only when they need to render & are then cached

Lazy Loading Hundreds of Components

REACT LAZY & SUSPENSE

```
const Photos = lazy(() => import("./components/Photos"));
const Posts = lazy(() => import("./components/Posts"));


const App = () => {
  const [showContent, setShowContent] = useState(false);


  return (
    <div className="App">
      <div className="header">
        <button onClick={() => setShowContent(true)}>Show Content</button>
        <button onClick={() => setShowContent(false)}>Hide Content</button>
      </div>
      <div className="content">
        {showContent ? (
          <>
            <Suspense fallback={<Spinner />}>
              <Photos />
              <Posts />
            </Suspense>
          </>
        ) : null}
      </div>
    </div>
  );
};
```

- Helps bring down the size of initially loaded code

REACT LAZY & SUSPENSE

```
const Photos = lazy(() => import("./components/Photos"));
const Posts = lazy(() => import("./components/Posts"));


const App = () => {
  const [showContent, setShowContent] = useState(false);


  return (
    <div className="App">
      <div className="header">
        <button onClick={() => setShowContent(true)}>Show Content</button>
        <button onClick={() => setShowContent(false)}>Hide Content</button>
      </div>
      <div className="content">
        {showContent ? (
          <>
            <Suspense fallback={<Spinner />}>
              <Photos />
              <Posts />
            </Suspense>
          </>
        ) : null}
      </div>
    </div>
  );
};
```

- Helps bring down the size of initially loaded code
- Components are loaded on-demand

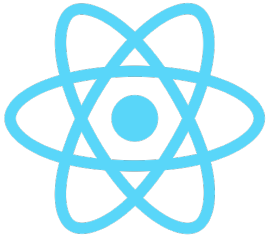
REACT LAZY & SUSPENSE

```
const Photos = lazy(() => import("./components/Photos"));
const Posts = lazy(() => import("./components/Posts"));


const App = () => {
  const [showContent, setShowContent] = useState(false);


  return (
    <div className="App">
      <div className="header">
        <button onClick={() => setShowContent(true)}>Show Content</button>
        <button onClick={() => setShowContent(false)}>Hide Content</button>
      </div>
      <div className="content">
        {showContent ? (
          <>
            <Suspense fallback={<Spinner />}>
              <Photos />
              <Posts />
            </Suspense>
          </>
        ) : null}
      </div>
    </div>
  );
};
```

- Helps bring down the size of initially loaded code
- Components are loaded on-demand
- Suspense component helps render a fallback UI while the component is loaded in asynchronously

LEARNING OBJECTIVES

Taking React to the Next Level



React’s Concurrent Mode → A Game Changer!

Coming Soon in the Stable Version

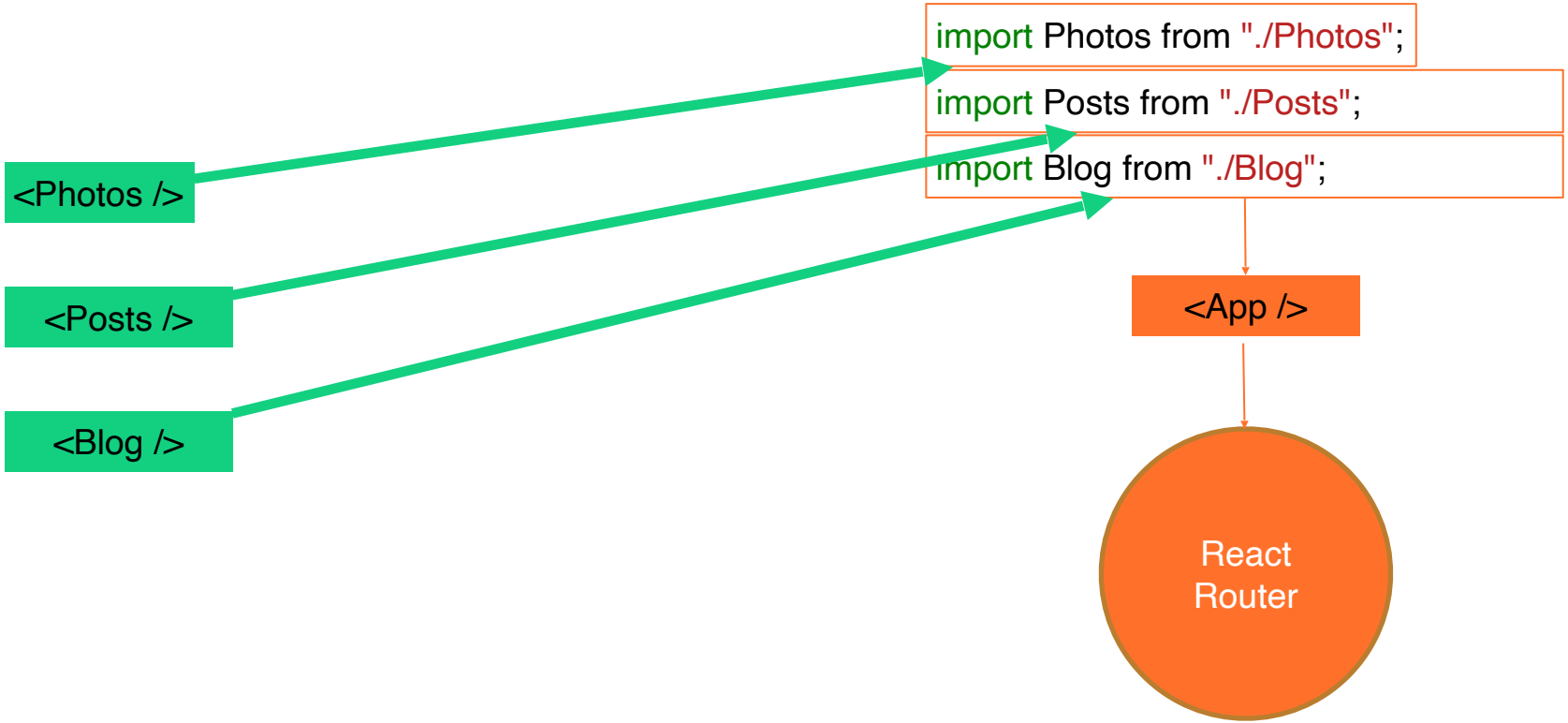
CODE SPLITTING AND LAZY LOADING

CODE SPLITTING AND LAZY LOADING

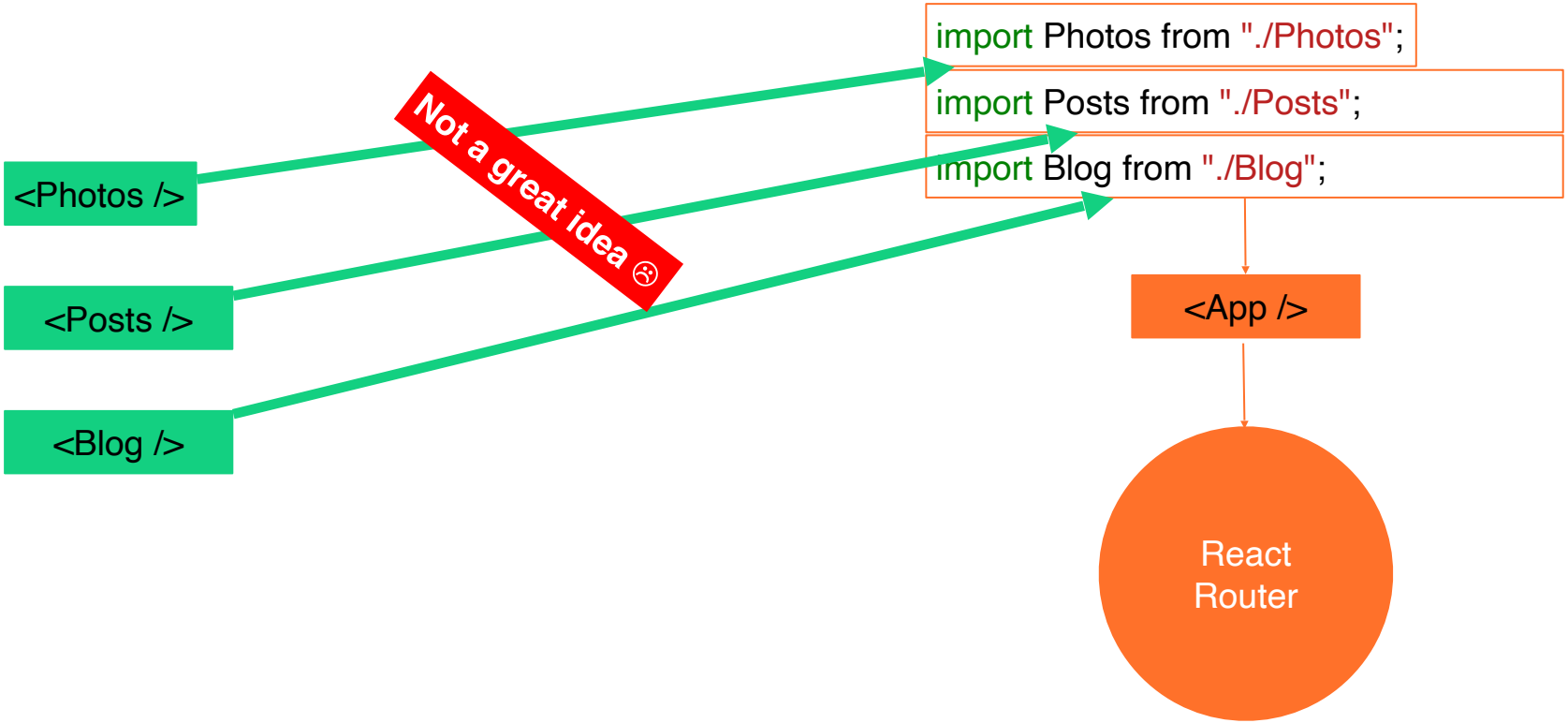
```
const Photos = lazy(() => import("./components/Photos"));
const Posts = lazy(() => import("./components/Posts"));


const App = () => {
  const [showContent, setShowContent] = useState(false);


  return (
    <div className="App">
      <div className="header">
        <button onClick={() => setShowContent(true)}>Show Content</button>
        <button onClick={() => setShowContent(false)}>Hide Content</button>
      </div>
      <div className="content">
        {showContent ? (
          <>
            <Suspense fallback={<Spinner />}>
              <Photos />
              <Posts />
            </Suspense>
          </>
        ) : null}
      </div>
    </div>
  );
};
```

- Helps bring down the size of initially loaded code
- Components are loaded on-demand
- Suspense component helps render a fallback UI while the component is loaded in asynchronously

CODE SPLITTING AND LAZY LOADING



CODE SPLITTING AND LAZY LOADING



Code Splitting & Lazy Loading Saves the Day!

CODE SPLITTING AND LAZY LOADING

BEFORE CODE SPLITTING

Main.chunk.js	5 Kb

AFTER CODE SPLITTING

Main.chunk.js	500 bytes
Post.chunk.js	1.2 Kb
Photos.chunk.js	1.4 Kb
Blog.chunk.js	1.25 Kb

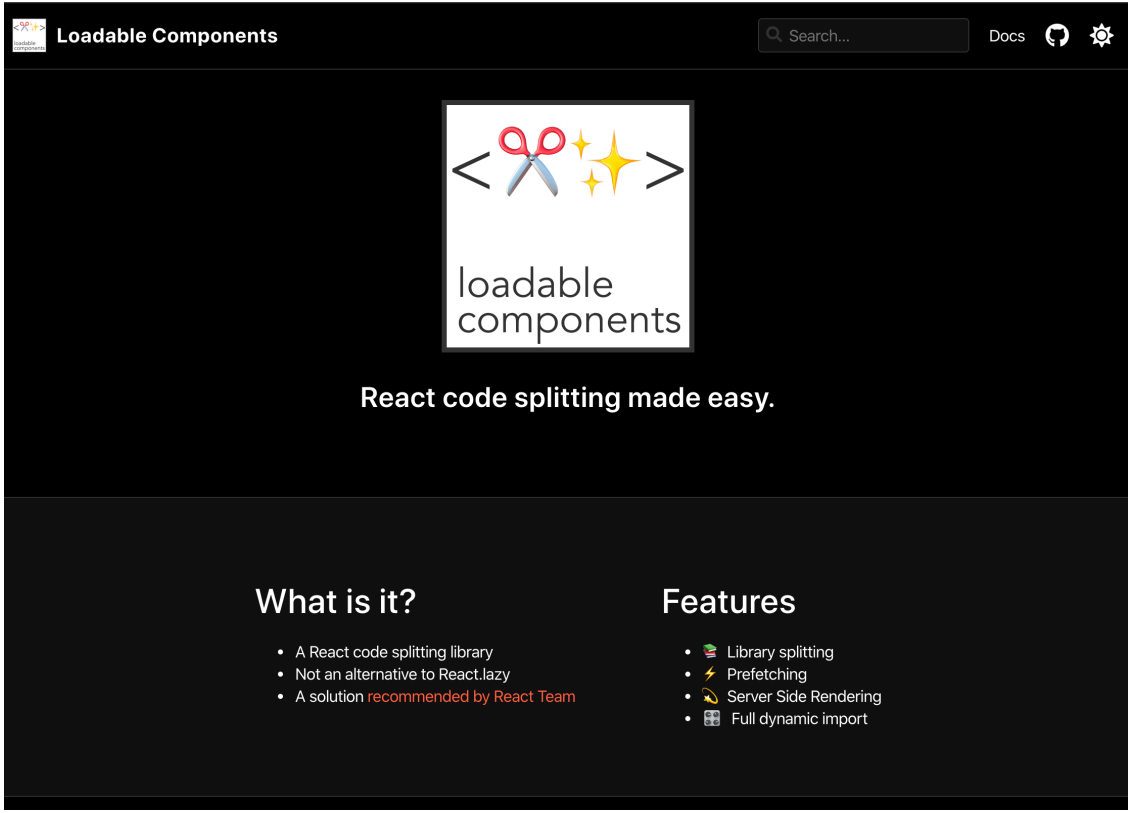
For illustrative purposes only

LOADABLE COMPONENTS FOR REACT

LOADABLE COMPONENTS FOR REACT

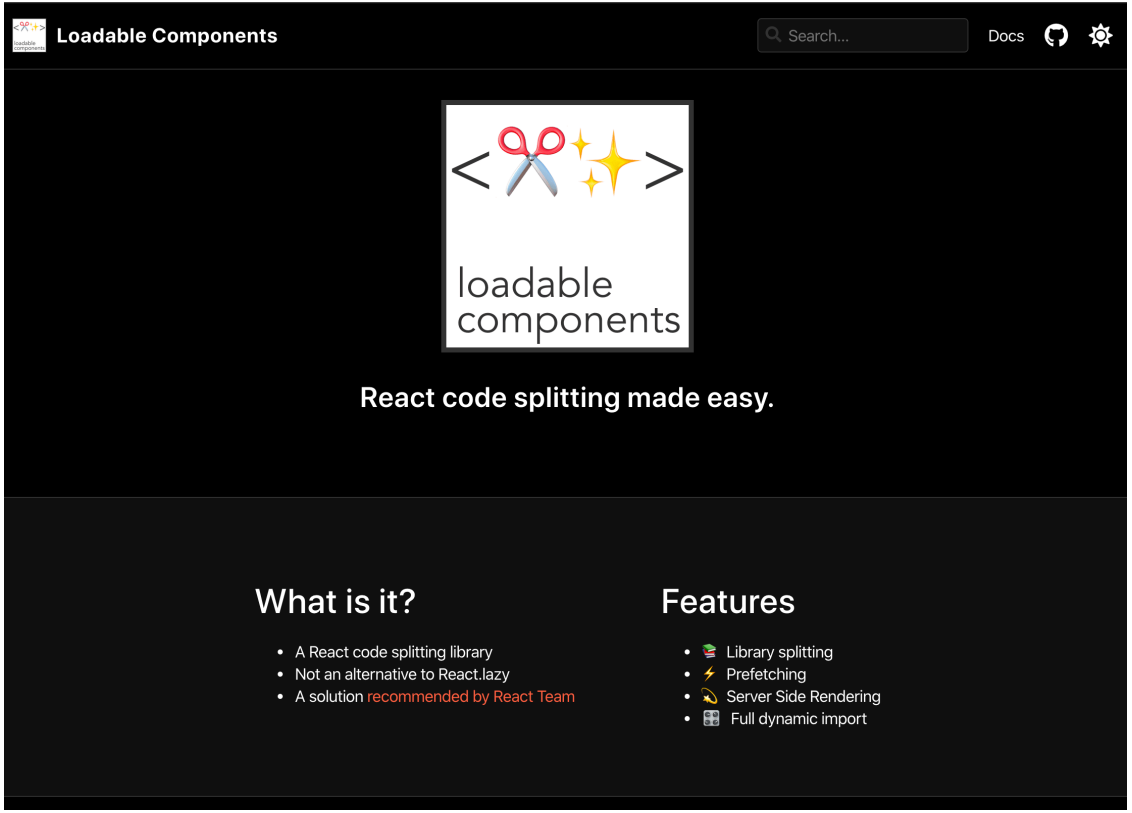
HANDS ON

LOADABLE COMPONENTS FOR REACT



https://www.smooth-code.com/open-source/loadable-components/

LOADABLE COMPONENTS FOR REACT



- Lazy loading without Suspense
- Support for Server Side Rendering (SSR)
- Splitting libraries & external modules
- And more…

https://www.smooth-code.com/open-source/loadable-components/

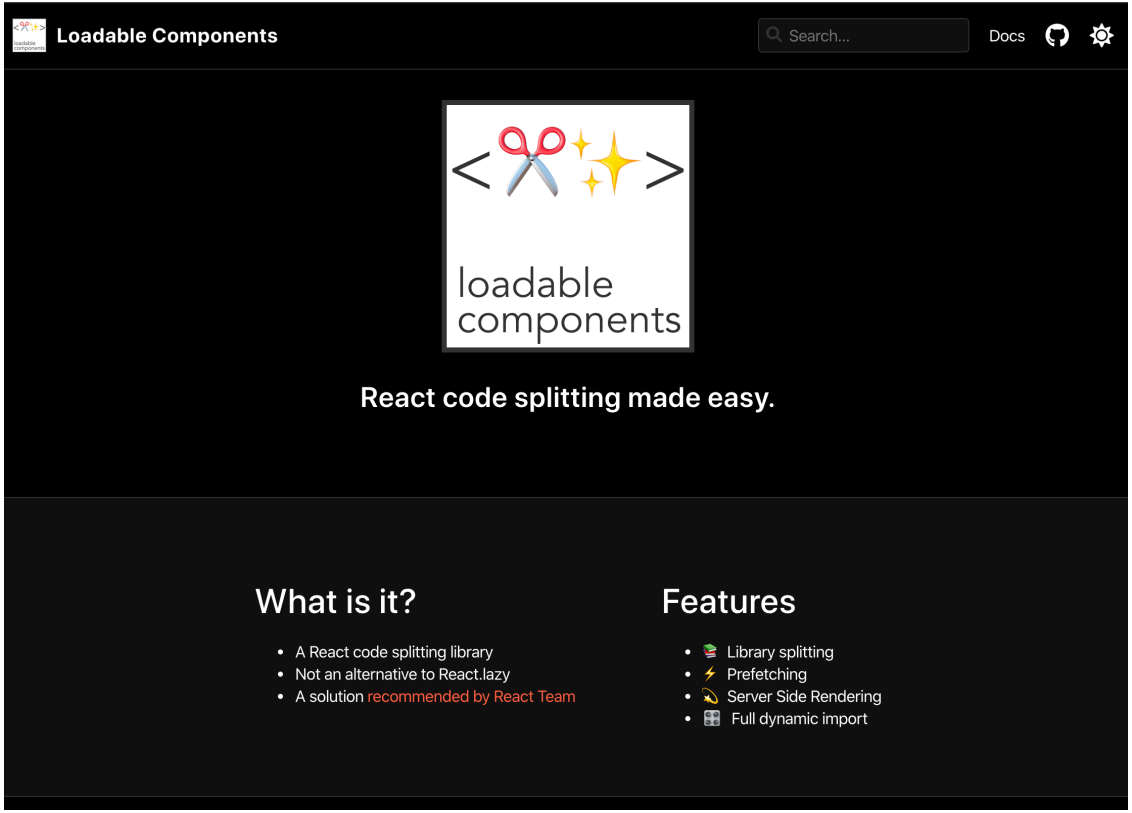
REACT BUILT-IN LAZY& SUSPENSE

```
const Photos = lazy(() => import("./components/Photos"));
const Posts = lazy(() => import("./components/Posts"));
```

&

```
<Switch>
  <Route exact path="/">
    <h1>This is Home!</h1>
  </Route>
  <Route path="/photos">
    <Suspense fallback={<Spinner />}>
      <Photos />
    </Suspense>
  </Route>
  <Route path="/posts">
    <Suspense fallback={<Spinner />}>
      <Posts />
    </Suspense>
  </Route>
</Switch>;
```

LOADABLE COMPONENTS FOR REACT



https://www.smooth-code.com/open-source/loadable-components/

CODE SPLITTING AND LAZY LOADING

Code Splitting & Lazy Loading (with or without Suspense) is a critically important optimization and must be implemented, especially in larger and evolving React apps!

thank you!