Code Splitting Routes

In this lesson, we'll learn how to split routes to send a large chunk of code in different paths.

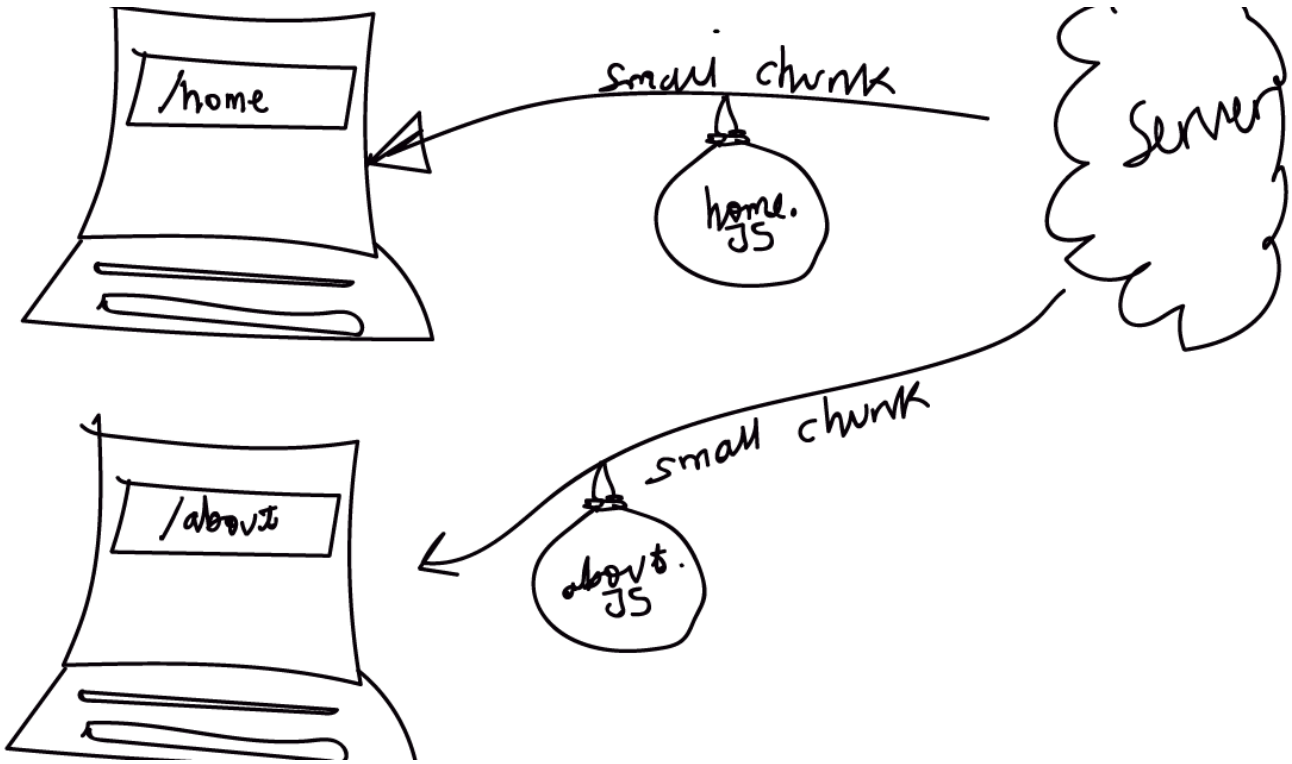
WE'LL COVER THE FOLLOWING

- Using Lazy Loading

Code splitting advocates that instead of sending this large chunk of code to the user at once, you may dynamically send chunks to the user whenever they actually need them.

We have looked at component-based code splitting in the earlier examples, but another common approach is route-based code splitting.

In this method, the code is split into chunks based on their routes in the application.



Using Lazy Loading

We could also take our knowledge of lazy loading one step further by adding route-based code splitting.

Consider a typical React app that uses `react-router` for route matching.

```
const App = () => (
  <Router>
      <Switch>
        <Route exact path="/" component={Home}/>
        <Route path="/about" component={About}/>
      </Switch>
  </Router>
);
```

We could lazy load the `Home` and `About` components so that they are only fetched when the user hits the associated routes.

Here's how we can do that with `React.lazy` and `Suspense`.

```
// Lazy load the route components
const Home = React.lazy(() => import('./Home'))
const About = React.lazy(() => import('./About'))
// Provide a fallback with Suspense
const App = () => (
  <Router>
    <Suspense fallback={<div>Loading...</div>}>
      <Switch>
        <Route exact path="/" component={Home}/>
        <Route path="/about" component={About}/>
      </Switch>
    </Suspense>
  </Router>
);
```

Easy, huh?

We've discussed how `React.lazy` and `Suspense` works, but under the hood, the actual code-splitting and generating separate bundles for different modules is done by a bundler, e.g., Webpack.

If you use `create-react-app`, `Gatsby`, or `Next.js` you already have this set up for you.

Setting this up yourself is also easy, you just need to tweak your `Webpack` config a little bit.

The official `Webpack` documentation has an entire guide on this. The guide may be worth checking if you're handling the bundling configurations in your

application yourself.

In the next lesson, we'll add lazy loading to our bank app project.