React JS + Redux

July 7, 8, 14, 2018

Routers:

- React Router is a package that allows you to configure routes that show only the components you specify on the page depending on the route
- For example, if you have a long list of movies and only want to show the user's favorites when they click on a 'favorites' button, you can do that with React Router

Installation:

- Create-react-app routerapp
- Delete src folder
- Create src folder
- ‘App.js’ file has two components (Title and List)
- npm i react-router-dom --save

React Router components:

- React Router includes several components but the three I've used the most are <BrowserRouter>, <Route>, and <Link>
- The first one, <BrowserRouter>, is usually given an alias of 'Router' and this is the parent component that is used to store all of your <Route> components
- The <Route> components are what tell your app which other components to display based on the route
- And <Link> components are how you create links to those different routes

Sample code (index.js):

```
import React from 'react';  7.8K (gzipped: 3.3K)
import ReactDOM from 'react-dom';  95.4K (gzipped: 30.8K)
import { BrowserRouter as Router, Route } from 'react-router-dom';
import { Title, List } from './components/App';
import './index.css';
ReactDOM.render(
  <Router>
    <div>
      <Route exact path="/" component={Title} />
      <Route path="/list" component={List} />
    </div>
  </Router>,
  document.getElementById('root')
)
```

Rendering a router(index.js):

- Here I have the <Router> component with a child div since that component can only have one child
- Then in that div we place all of the routes needed for the example
- Each <Route> component needs a path which is the URL and then a component that you want rendered when navigating to that path
- Since we have specified the ‘exact’ parameter, only the <Title> component will be rendered on the root route “/” and only the <List> component will be rendered on the “/list” route.
- If we wanted <Title> to be displayed on both routes then we can remove the ‘exact’ parameter
- Then when navigating to “/list” it would match both “/” and “/list”.

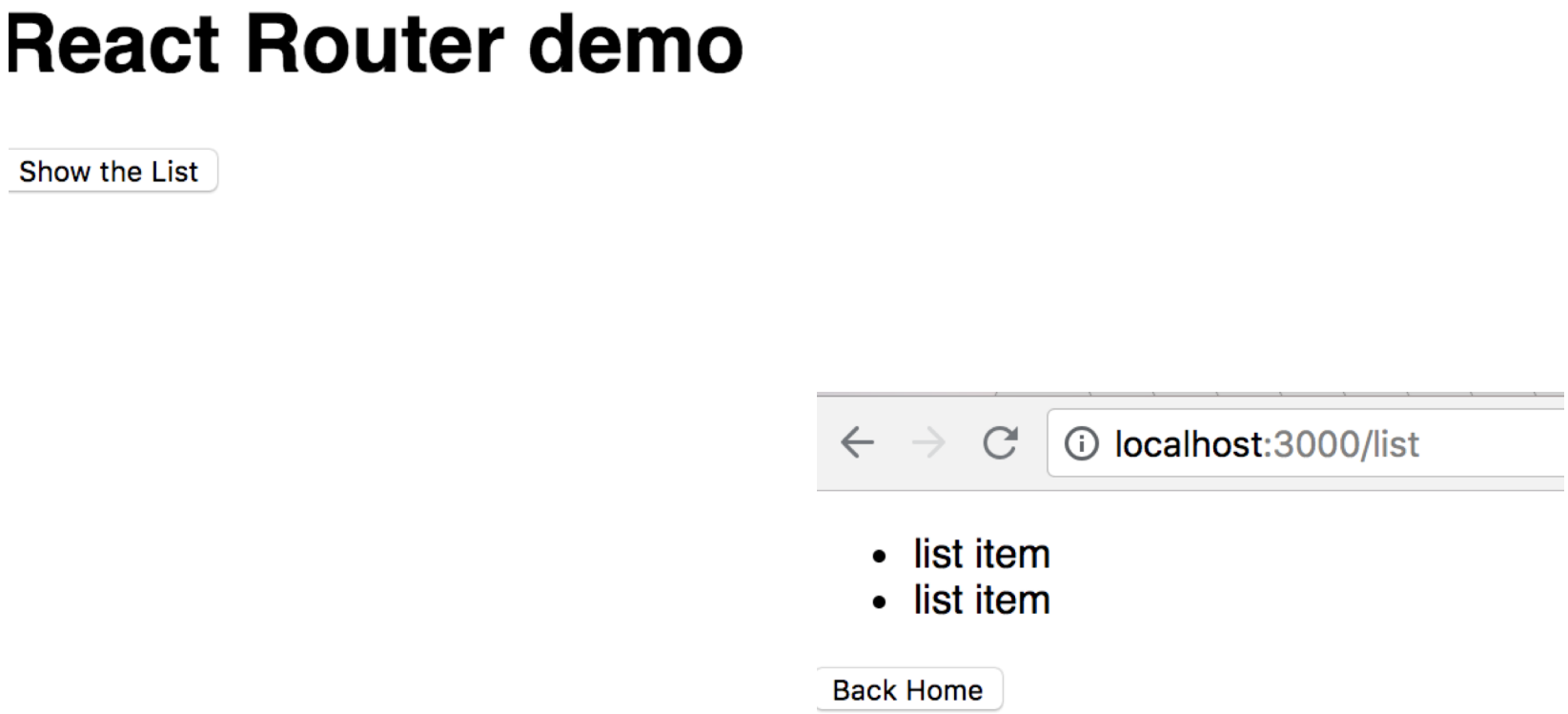
Sample code (App.js):

```
import React, { Component } from 'react';   7.8K (gzipped: 3.3K)
import { Link } from 'react-router-dom';   40.2K (gzipped: 10K)

export const Title = () => {
  return (
    <div className="title">
      <h1>React Router demo</h1>
      <Link to="/list"><button>Show the List</button></Link>
    </div>
  )
}

export const List = () => {
  return (
    <div className="nav">
      <ul>
        <li>list item</li>
        <li>list item</li>
      </ul>
      <Link to="/"><button>Back Home</button></Link>
    </div>
  )
}
```

(App.js):

- Since we are not dealing with state or props or ‘this’ in this example I just made them stateless functional components. ES6 classes would work the same way

Results:

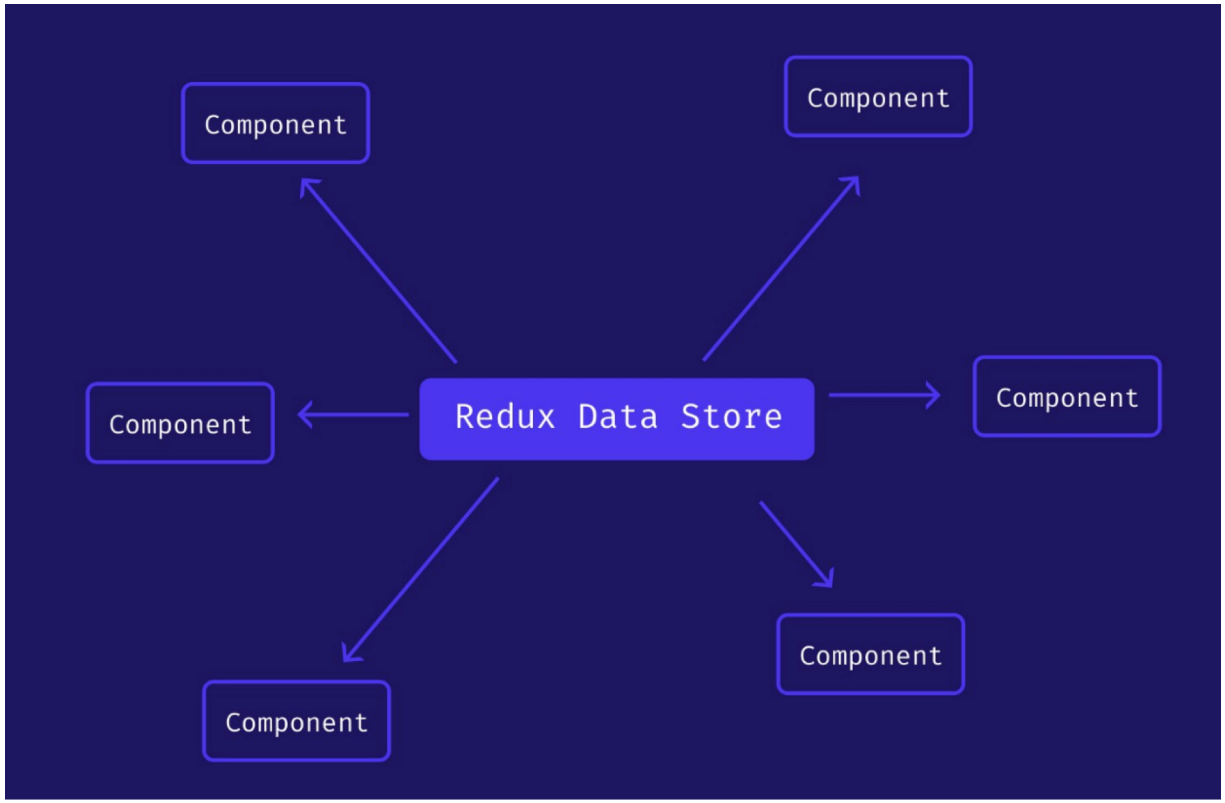

React Router demo
Show the List
localhost:3000/list
list item
list item
Back Home

Limitations

- A routing library is a key component of any complex, single-page application
- React Router isn't the only viable solution in the React/Redux ecosystem
- In fact, there are tons of routing solutions
- Redux-first routing is a paradigm makes Redux the *star* of the routing model, and the common thread among many Redux routing solutions
- As apps would grow in complexity, you would want to reuse components
- components were tied up in the context of their parent's state data
- changing the structure of any parent-level data meant refactoring many child components to match, which turned out to be a massive headache
- With Redux Instead of keeping app data inside a tree of components, it's maintained in a central data store outside of React entirely.
- Any React component can access or update the store, but none of the components need to know about or pass around the entirety of the store

Limitations

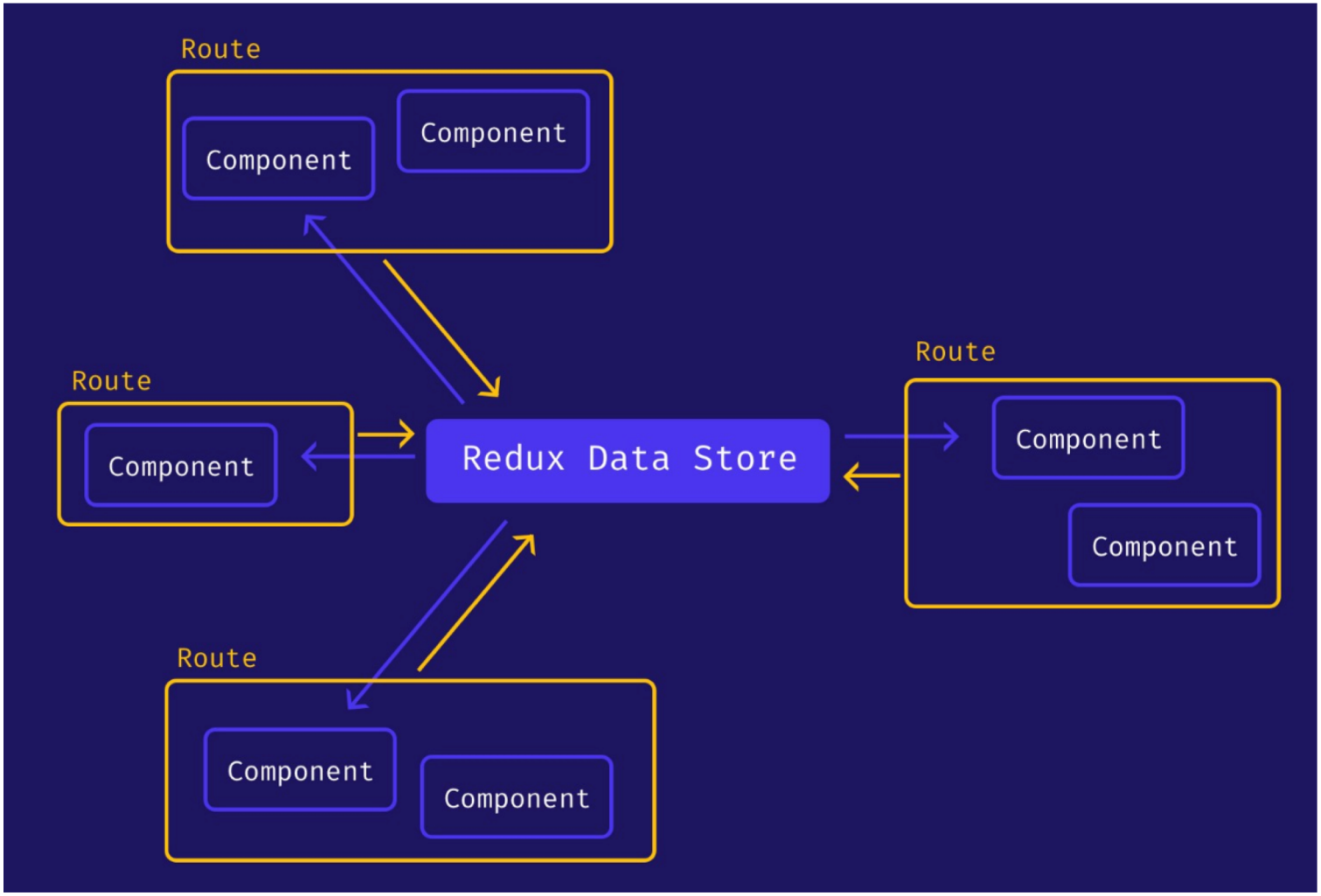
- It unlocked the ability to write React apps with genuinely reusable components at scale



Limitations

- A router in a React app holds routing and URL information, which is a type of application data
- Interestingly, routing data is not stored in the Redux store when using React Router.
- Instead, it is stored in routing components spread throughout a React app's components
- If you want your Redux store to be the source of truth for your application data, you're stuck syncing your React Router's route to your Redux Store on every click

Limitations



Switching to a redux first router

- All app data, including routing data, is in Redux
- Redux First Router dispatches an action on every route change, making it easy to fetch data with a thunk or saga on page load.
- Debugging is easier when you have a record of all route actions alongside all actions
- Components do not need to know where they are nested, nor do they need to point to hard coded routes.

End of Chapter 10