Managing Project Dependencies

Adding Initial Dependencies

New CRA projects come configured with two library dependencies: React and ReactDOM. We're going to add a few more libraries as we start out. Here's what we're going to add:

- **Redux**: the standard state management library for React apps (because without it, this would be a *really* short and pointless course)
- **React-Redux**: the official bindings to allow React components to interact with a Redux store.
- **Redux-Thunk**: the standard Redux addon for enabling asynchronous "side effects" like AJAX calls and complex synchronous logic outside of components.
- **Reselect**: the standard Redux addon to help create "selector functions" that encapsulate state lookups and improve app performance.
- **Lodash**: every useful function you can think of, and a bunch more you didn't even know existed

We'll add more libraries as we go along, but this list would be considered a good basic starting set for a typical React+Redux application.

To help ensure that you get the same versions I'm using right now, we're going to use Yarn to add each library with a specific version. We can also do all this in one big command, so Yarn doesn't have to repeat some of its work:

```
yarn add redux@3.7.2 react-redux@5.0.6 redux-thunk@2.2.0 reselect@3.0.1 lo
dash@4.17.4
```

We need to commit our `package.json` and `yarn.lock` again, but before we do that, we're going to take a bit of a detour.

Managing Dependency Packages for Offline Installation

Managing Dependency Packages for Offline Installation

Package managers are powerful tools. With just a config file and a command, we can download all the dependencies our application needs. Unfortunately, this also introduces many weaknesses and concerns. What happens if the latest version of a package breaks something for me? How can I know that I'm getting the same package contents every time? What if something happens to a package server, or I need to be able to install these packages offline?

The infamous `left-pad` incident and various Github outages have shown that these are very real questions. While the use of hashes and locked URLs can ensure that a package manager like NPM is actually seeing the same file each time, that doesn't help if the network is down.

There's been a few approaches suggested for handling NPM dependencies without needing a network connection. Using NPM or Yarn's local per-machine cache works, but only if you've downloaded the necessary packages on that machine before. A number of people have suggested that you check in your entire `node_modules` folder, but that's a *very* bad idea for a variety of reasons. That's potentially tens or hundreds of thousands of files taking up hundreds of megs on disk, AND that can include platform-specific binaries built post-install (like `node-sass`'s inclusion of `libsass`). If you check in your `node_modules` on a Mac, there's a good chance that it won't work on Windows or Linux (and vice versa).

The most ideal solution is to actually check in the downloaded archives for each package. Since platform-specific artifacts like `libsass` are built *after* installation, you can safely clone a repo on any machine, install the packages from that per-repo cache, and have things built properly for that machine.

Yarn includes an "offline mirror" feature built in. As far as I know, NPM has not had this built in as a specific capability, but there's a third-party tool called Shrinkpack that can use an NPM "shrinkwrap" file as the basis for caching package tarballs in the repo. Based on a recent Twitter conversation, it seems that this is also a future planned feature for a future NPM5 release, and that it's possible to sorta-kinda mimic that with a `.npmrc` file and the `--prefer-offline` flag for NPM right now.

Configuring an Offline Mirror for Packages

Since we're adding our first additional dependencies, now is a great time to set up an "offline mirror" for our project.

First, we need to create a `.yarnrc` file in the repo:

> **Commit 0de287e: Add Yarn config file for an offline cache**

.yarnrc

```
yarn-offline-mirror "./offline-mirror"
yarn-offline-mirror-pruning true
```

We'll set two values. `yarn-offline-mirror` is the folder where we want Yarn to save the package tarballs. By default, if you update package versions, Yarn will only add new files to that folder, but not remove outdated ones. If we set `yarn-offline-mirror-prune` to `true`, it will also remove old package tarballs to match what's currently installed.

Next, we need to actually force Yarn to reinstall everything. The simplest way is to rename the `node_modules` folder to something like `node_modules.bak`, then run `yarn` again. Now, if you look inside the `offline-mirror` older, you should see around 1000 archives, like `react-15.6.1.tgz`. We can `git add` the entire folder, and check them all in:

> **Commit 5d2d933: Commit initial dependency packages**

And with that, anyone else who is using Yarn along with this project *should* be able to use exactly the same packages that I have right now, even if you try to install them while you're offline.

Finally, a bit of housekeeping: I added a LICENSE file to the project, and replaced the default CRA README with a description of this project:

> **Commit 7d327d5: Add LICENSE**

Commit 3191743: Rewrite README to describe this project