React JS

July 27, 2018

Development Software

React Redux

```
npm install react-redux --save
```

Redux

```
npm install redux --save
```

Microsoft's VS Live Share (preview)

https://marketplace.visualstudio.com/items?itemName=MS-vsliveshare.vsliveshare

Babel Dependency and Plugins:

```
npm install -g babel
npm install babel-core --save
npm install babel-loader --save
npm install babel-preset-react --save
npm install babel-preset-env --save
```

React Router V4

- Axios is a library that makes working with AJAX requests simple.
- It is promise-based, so we can use the traditional .then() .catch() model of native JavaScript promises.
- The documentation can be found here: https://github.com/axios/axios.
- To use the library we will be first importing it:

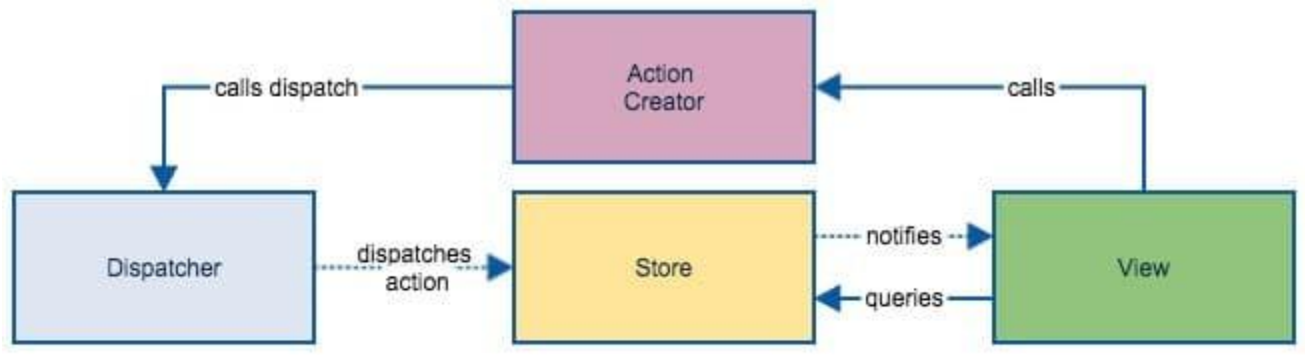
```
import { Router, Route, Switch } from "react-router";
import createBrowserHistory from "history/createBrowserHistory";
const history = createBrowserHistory();
<Router history={history}>
    <Switch>
        <Route exact path="/" component={Home} />
    </Switch>
</Router>
```

- Since these routes are just components themselves, they are generally included in a top-level component that is injected into the real DOM by ReactDOM.
- path defines a path to match against for mounting a specific component. You can also use parameters like so:

```
<Route exact path="/wines/:id" component={EditWine} />
```

- These parameters can be accessed by `this.props.match.params` in the appropriate class.
- Note that parameters are only accessible in the component classes that are mounted via the router itself; not any child components.

Flux Overview



http://www.webdevelopmenthelp.net/2016/12/reactjs-interview-questions.html

Flux Notes

- Store is a place that holds the app's state
- Actions are JavaScript objects that use type property to inform about the data that should be sent to the store.
- The reducer is a function that takes two parameters (state and action) to calculate and return an updated state.
- combineReducers helper function can add any new reducers
- Only the root component should be aware of a redux using connect
  - export default connect(select)(App)
-

Redux - Predictable state container for JavaScript apps

- Everything that changes in the application (including the data and the UI state) is contained in a single object called the state or the state tree
- Understand where state is stored and how state changes. Allows state to be managed in one large store instead of state being managed at a component level
- Redux can be used with
    - React, React Native, Angular, Angular 2, Vue and other UI Layers
- Use Redux when:
    - you have reasonable amounts of data changing over time,
    - you need a single source of truth, and
    - keeping everything in a top-level React component's state are no longer sufficient
- Redux architecture revolves around a strict unidirectional data flow

What kind of data should be put into Redux?

- Do other parts of the application care about this data?
- Do you need to be able to create further derived data based on this original data?
- Is the same data being used to drive multiple components?
- Is there value to you in being able to restore this state to a given point in time (ie, time travel debugging)?
- Do you want to cache the data (ie, use what's in state if it's already there instead of re-requesting it)?
- Should the data be serializable? Yes!
- Each object should be stored once, keyed by ID, and other objects that reference it should only store the ID rather than a copy of the entire object

HTTP Requests with Axios

- Axios is a library that makes working with AJAX requests simple.
- It is promise-based, so we can use the traditional .then() .catch() model of native JavaScript promises.
- The documentation can be found here: https://github.com/axios/axios.
- To use the library we will be first importing it:

```
import axios from "axios";

axios.request({
    url: "endpoint url here",
    method: "GET"
})
.then((response) => {
    console.log(response.data);
})
.catch((err) => {
    console.log(err);
});
```

Babel 5

Browsers do not understand JSX code natively, we need to convert JSX to JavaScript first which can then be used in our browsers. We have plugins which handle including Babel 5's in-browser ES6 and JSX transformer called browser.js. Babel will understand and recognize JSX code in **<script type="text/babel"></script>** tags and transform/convert it to normal JavaScript code

.

Babel Dependency and Plugins:

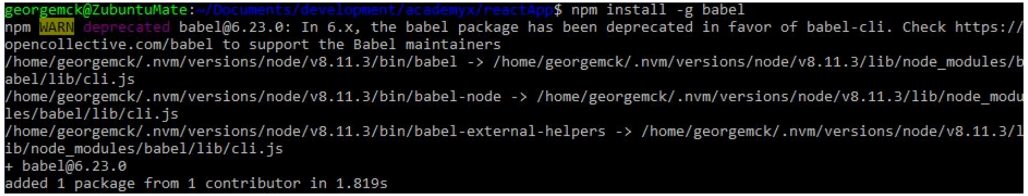
npm install -g babel

npm install babel-core --save

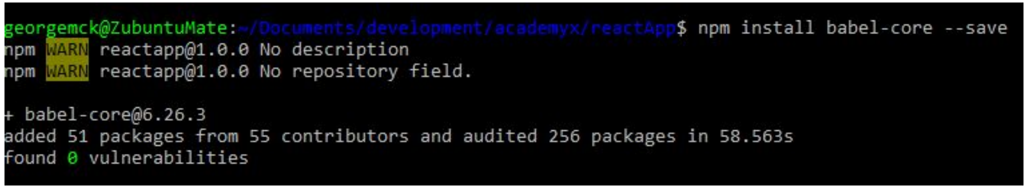
npm install babel-loader --save

npm install babel-preset-react --save

npm install babel-preset-env --save



```
georgemck@ZubuntuMate:~/Documents/development/academyx/reactApp$ npm install -g babel
npm WARN deprecated babel@6.23.0: In 6.x, the babel package has been deprecated in favor of babel-cli. Check https://
opencollective.com/babel to support the Babel maintainers
/home/georgemck/.nvm/versions/node/v8.11.3/bin/babel -> /home/georgemck/.nvm/versions/node/v8.11.3/lib/node_modules/b
abel/lib/cli.js
/home/georgemck/.nvm/versions/node/v8.11.3/bin/babel-node -> /home/georgemck/.nvm/versions/node/v8.11.3/lib/node_modu
les/babel/lib/cli.js
/home/georgemck/.nvm/versions/node/v8.11.3/bin/babel-external-helpers -> /home/georgemck/.nvm/versions/node/v8.11.3/l
ib/node_modules/babel/lib/cli.js
+ babel@6.23.0
added 1 package from 1 contributor in 1.819s
```

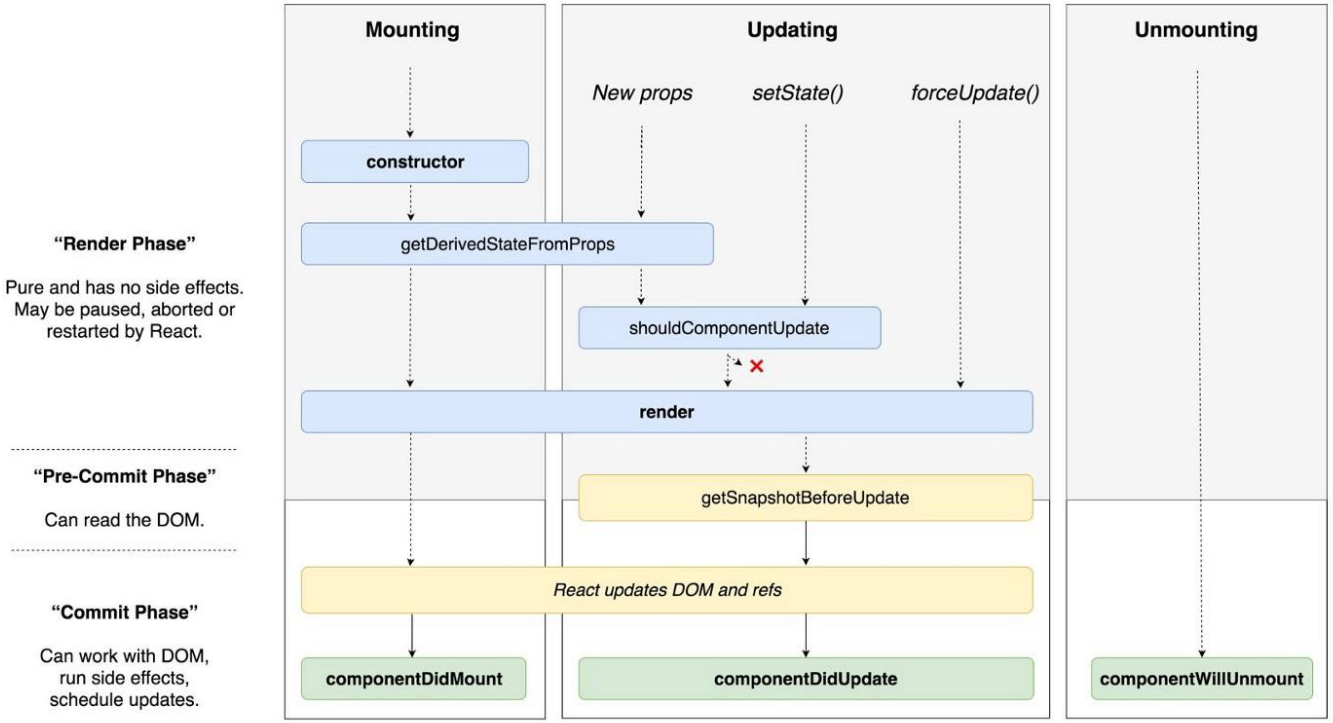


```
georgemck@ZubuntuMate:~/Documents/development/academyx/reactApp$ npm install babel-core --save
npm WARN reactapp@1.0.0 No description
npm WARN reactapp@1.0.0 No repository field.

+ babel-core@6.26.3
added 51 packages from 55 contributors and audited 256 packages in 58.563s
found 0 vulnerabilities
```

http://babeljs.io/docs/en/babel-cli/

Component Lifecycle



https://medium.com/@saharshgoyal/react-16-new-life-cycle-methods-inside-out-fdd269c43ccd

Class Goals

Basic Components

- Props
- Component hierarchy
- Building the components
- Rendering components

DOM Abstraction

- How React views the Document Object Model
- Event listeners
- JSX vs. HTML
- Rendering dynamic HTML
- Using plain JavaScript
- Element and custom factories
- Form components
- Keys and refs

Architecture with Components

- Prop validation
- Built-in and custom validators
- Component composition strategies
- Lifecycle
- Components that retain state
- Nesting components and passing state
- Immutability

Advanced User Interaction

- Animations
- Drag and drop

Routing

- The React router
- The index router
- Parameters
- Active links
- Properties

Routing (cont'd)

- The user interface and the URL
- Changing routes programmatically

Flux

- Flux methodology
- Stores
- Actions
- Dispatchers
- Using Flux asynchronously

Performance Tuning

- Reconciliation
- Batching and sub-tree rendering
- ReactPerf and the performance test application

Testing Components

- Popular Testing Frameworks: Jest and Jasmine
- What to test
- Rendering components for testing
- Traversing components
- Shallow rendering
- Simulating Events

Isomorphic Applications in Node

- Node.js and Express web services
- Rendering React components on the server
- Mounting on the client
- Internal Routes
- Dynamic Data Fetching
- Rendering Routes

Deployment

- Deploying Babel in production
- Build tools and webpack