React Router

`<Route>` Functions like an *if-statement*, conditionally renders the given component based on the URL path

`<Link>` Functions like an *a tag* link, "tricks" user that they are going to a new page, but doesn't truly cause a page refresh

`<Switch>` Allows only 1 route to be matched

`<BrowserHistory>` Uses the Browser history (back and forward buttons) to simulate going from one page to another

React Router Example

Always need to include in `index.js`:

```
import { BrowserRouter }
  from "react-router-dom";
ReactDOM.render(

    <BrowserRouter>
      <App />
    </BrowserRouter>,

  document.getElementById("root"));
```

Put top-level routes in `App.js`:

```
import { Link, Switch, Route }
  from "react-router-dom";

<Switch>
  <Route exact path="/"
    component={Index} />
  <Route path="/about/"
    component={About} />
  <Route path="/post/:id/"
    component={BlogPost} />
</Switch>
```

Link examples:

```
<Link to="/about/">About</Link>
<Link to={"/post/" + postId + "/"}>
  Read More...</Link>
```

React Redux

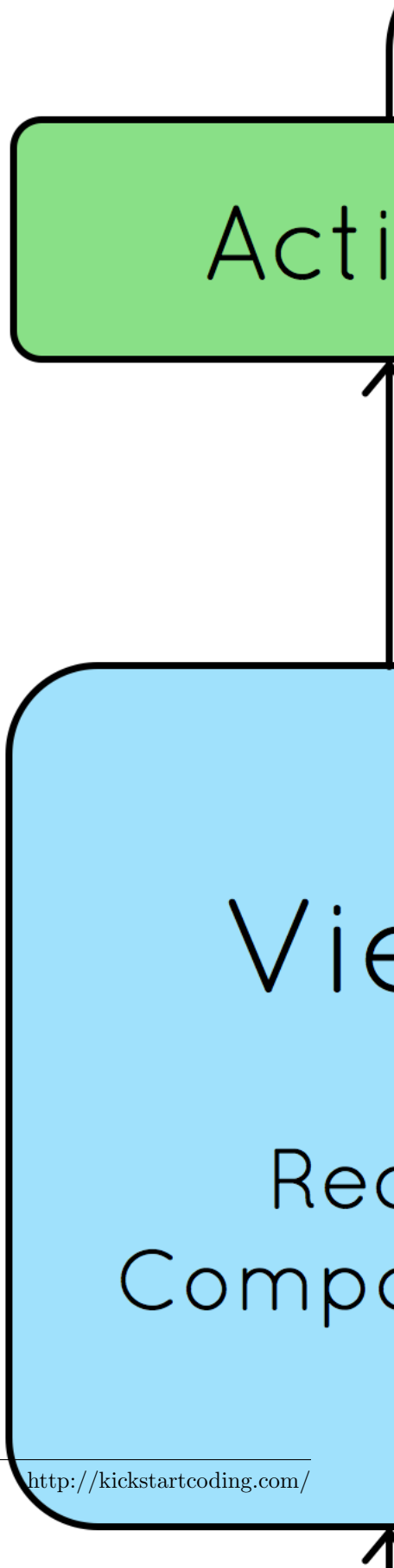
Redux Paradigm *a la* MVC, but for state-management, and for React, useful for large projects where application state becomes too complicated to manage on a single component

Store The Redux state, often used most for data fetched from back-end

Action Represents an "event" that occurs, e.g. an action is *dispatched* when data is `fetch`ed from the back-end, and another when the response comes back

Reducer Triggered when an action is dispatched, a reducer modifies (a duplicate of) the state based on the action that happened

Store state slice Redux is partitioned into slices that handle different aspects of large applications (analogy: *Django apps*)



React Redux Code

MongoDB

noSQL database A database that doesn't use SQL and traditional table / row / column organization

MongoDB NoSQL database that stores JSON *documents*, with no built-in schema-enforcement

document *row* in SQL, single item of data in NoSQL, represented by BSON (a JSON variant)

collection *table* in SQL, group of documents with a name

ObjectID Long random string serving as unique ID for each document

MongoDB CRUD

```
# READ
db.userprofiles.find(
  {name: "janeqhacker"})

# CREATE
db.userprofiles.insertOne({
  name: "janeqhacker",
  email: "janeq@hack.er",
  mood: "happy",
  wordCount: 20,
  posts: [],
})

# UPDATE
db.userprofiles.update(
  { name: "janeqhacker" },
  {
    $push: { posts: "Good idea" },
    $inc: { wordCount: 2 },
    $set: { mood: "thoughtful" }
  }
);

# DELETE
db.userprofiles.deleteOne(
  {name: "janeqhacker"});
```

Express.js

Most popular back-end web framework for Node.js JavaScript