# React-Redux Search Application

Let’s begin by creating our React app, titled redux-articlelist with [create-react-app](https://github.com/facebook/create-react-app).

cd desktop  
npx create-react-app redux-articlelist

Next up, let’s go into the directory and install some of the packages that we will be using to enable the Redux library.

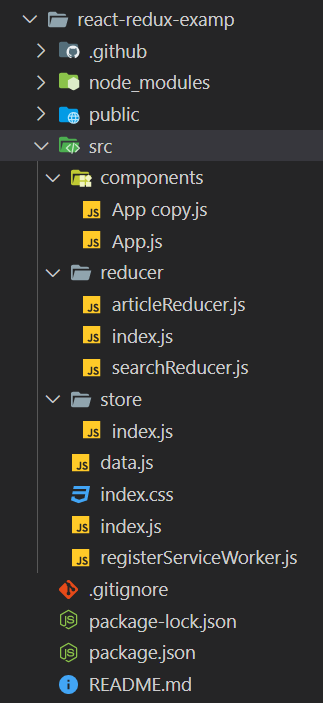
These will include Redux, [React-Redux](https://react-redux.js.org/) to connect React to Redux.

cd redux-articlelist  
npm i redux react-redux

Now let’s open our text editor and set up/clean up our app by removing the logo and app.test.js, and create our file structure for our Redux store.

This will include our actions and reducers folders, as well as an index.js to hold our data.

So, our text editor should look something like this at the final:



Now that we have our initial file set up for our store, let’s go ahead and create our first action and reducer.

**1.**

In our reducers folder create articlelistReducer.js. and searchReducer.js.

**articlelistReducer.js**, should take “data” from the data.js file as an initial state.

After that add this function at the bottom of the file;

function articleReducer(state = INITIAL\_STATE, action) {

  switch (action.type) { default: return state;} }

export default articleReducer;

**2.**

**searchReducer.js**, should take empty string equals to “searchTerm” as an initial state.

After that add this function at the bottom of the file;

function searchReducer(state = INITIAL\_STATE, action) {

  switch (action.type) {

    case 'SEARCH\_SET':

      return { ...state, searchTerm: action.searchTerm };

    default: return state; } }

export default searchReducer;

Note that our reducers takes two parameters, state and action.

**3.**

Now that we have the logic written out for our reducer to search our list, let’s set up our store in our index.js file create in our store folder.

index.js(in our store folder); use createStore and combineReducers from redux and combine your reducers (from reducer folder), after that create a store. Do not forget to export your store.

**4.**

Now let’s hook up our store to React and wrap our application (App) around the Provider in our *source* folder index.js.

index.js(src folder):

Do not forget to import Provider from redux and store from store.

**5.**

Let’s finally go into our App.js and clean it up and add this inside it.

App.js:

function App() {

return (<div className="App">REDUX ARTİCLELİST SEARCH APP</div>);};

export default App;

**6.**

When we return to our application in our browser, we will add-on our “Redux DevTools” extension (go to browser extensions and add extention).

But first add this code inside your store (index.js in store folder) as a second parameter of createStore().

window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_ && window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_()

**7.**

And we should be able to access our Redux DevTools by clicking on the lit up icon (for Chrome) on the top-right side.

This will open our GUI and if we click on state, we can now see our article list state being initialized!

**8.**

OK, awesome. Let’s go back to our text editor and set up our App.js so that it’s connected to our store. We will do that by importing our actions, connect, with state and dispatch, and set up props. Should look like this:

const mapStateToProps = state => ({

  articles: state.articlesState.articles,

  searchTerm: state.searchState.searchTerm,

});

const mapDispatchToProps = dispatch => ({

  onSearch: searchTerm => dispatch({ type: 'SEARCH\_SET', searchTerm }),

});

export default connect(mapStateToProps, mapDispatchToProps)(App);

**9.**

Now we need to list our articles.

In your App component call an <Articles /> component and send this component articles as a property.

Under your App component, create a function named “Articles” that maps articles in an <ul> element. Also give link for all articles by using articles url.

**10.**

In your App component call an <Search /> component and send this component onSearch method as a property.

Under your App component, create a function named “Search” that returns a div. Inside div there should be an <input> that has an onChange handler that has a callback function sends input value (event.target.value) to the onSearch method.

**11.**

Inside Article component filter your articles before your map(). Use filter() method.

Inside your filter method’s callback function, use includes to filter articles according to searchTerm and article title.

Phew! OK, now we should have our entire Redux cycle finished hooked up and running.

Congragulations…