Redux with React

1 counterReducer.js file

Have to update state immutably.

```
import * as actionTypes from '../actions';

const initialState = {
    counter: 0
}

const reducer = (state = initialState, action) =>
 {
    switch (action.type) {
        case actionTypes.INCREMENT:
            return {
                ...state,
                counter: state.counter + 1
            };
        default:
            return state;
    };
};

export default reducer;
```

2 resultReducer.js file

One separate reducer, can't use other separate reducer's state directly

```
import * as actionTypes from '../actions';

const initialState = {
    results: []
}

const reducer = (state = initialState, action) =>
 {
    switch (action.type) {
        case actionTypes.STORE_RESULT:
            return {
                ...state,
                results: state.results.concat({id
: new Date(), value: action.result})
            };
        case actionTypes.DELETE_RESULT:
            const updatedArray = state.results.fi
lter((result) => result.id !== action.resultElId)
;
            return {
                ...state,
                results: updatedArray
            }
        default:
            return state;
    };
};

export default reducer;
```

Connect with Redux in index.js

3 Import necessary files & modules

```
// Modules
import { createStore } from 'redux';
import { combineReducers } from 'redux';
import { Provider } from 'react-redux';

// Own Files
import reducer from './store/reducer';
import counterReducer from './store/reducers/coun
ter';
import resultReducer from './store/reducers/resul
t';
```

4 Connect Multiple Reduces and create store

```
const rootReducer = combineReducers({
    ctr: counterReducer,
    res: resultReducer
});
const store = createStore(rootReducer);

ReactDOM.render(<Provider store = {store}><App />
</Provider>, document.getElementById('root'));
registerServiceWorker();
```

5 Use Redux into Components

```
import React, { Component } from 'react';
import { connect } from 'react-redux';
import * as actionTypes from '../../store/actions
';

class Counter extends Component {
  state = {
      counter: 0
  }

  render () {
      return (
          <div>
              <CounterOutput value={this.props.ct
r} />
              <CounterControl label="Increment" c
licked={this.props.onIncrementCounter} />
              <hr />
              <button onClick = {() => this.props
.onStoreResult(this.props.ctr)}>Store Result</but
ton>
              <ul>
                  {this.props.storedResults.map(s
trResult => {
                      return <li key = {strResult
.id} onClick = {() => this.props.onDeleteResult(s
trResult.id)}>{strResult.value}</li>
                  })}
              </ul>
          </div>
      );
  }
}
const mapStateToProps = state => {
    return {
        ctr: state.ctr.counter,
        storedResults: state.res.results
    };
};

const mapDispatchToProps = dispatch => {
  return {
      onIncrementCounter: () => dispatch({type: a
ctionTypes.INCREMENT}),
      onStoreResult: (result) => dispatch({type:
actionTypes.STORE_RESULT, result: result}),
      onDeleteResult: (id) => dispatch({type: act
ionTypes.DELETE_RESULT, resultElId: id}),
    };
};

export default connect(mapStateToProps, mapDispat
chToProps)(Counter);
```