

React Patterns

Hand in Hand With the Basic Framework

Oliver Sturm • @olivers • oliver@oliversturm.com



Oliver Sturm

- Training Director at DevExpress
- Consultant, trainer, author, software architect and developer for over 25 years
- Contact: oliver@oliversturm.com

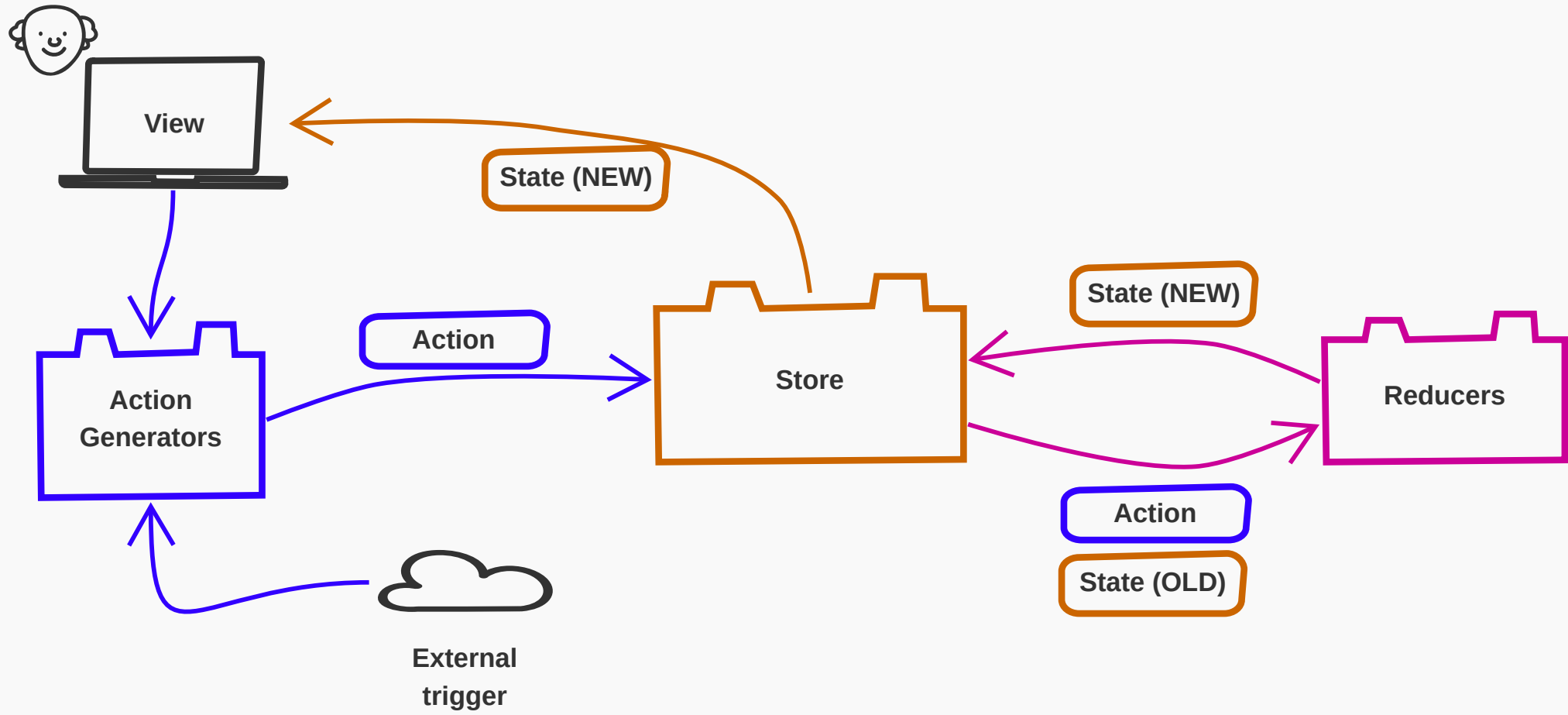
Agenda

- Common Libraries Not Covered By This Talk
- Redux
- Routing
- Handling Side Effects

Common Libraries Not Covered By This Talk

- UI: [Material-UI](#), [reactstrap](#) and many other free and commercial offerings
- Dealing with styles: [styled-components](#) or [CSS Modules](#)
- Framework: [Relay](#), [Apollo Client](#) for GraphQL, [Gatsby](#) for complete apps
- Utility: [Lodash](#), [Ramda](#)
- Outdated: [recompose](#)

Redux



I Like Redux

- The pattern has turned out to be *extremely powerful*
- Promotes *good structure*
 - ... which results in *maintainability*
- People complain about boilerplate
 - They're doing it wrong
- People complain about the learning curve
 - Oh well... it's really not that bad. Agreed: being a good programmer has a learning curve.
- Ideas are being *adopted elsewhere*, e.g. React Hooks ([useReducer](#)), [reactn](#) (global state management)

DEMO

Routing

- You know - how to map URLs to parts of your app
 - `/user/:id`
- The Big Ugly: React Router
 - Very powerful and flexible
 - Component based
- Alternative: *Routing as an aspect of State*
 - My favorite: Rudy, successor to redux-first-router

DEMO

Handling Side Effects

- React *render functions* are expected to be *functionally pure*
- React has *built-in mechanisms* to deal with *side effects*
 - Class-based lifecycle methods (`componentDidMount` etc)
 - Hooks (`useEffect`, `useLayoutEffect`)
 - Logic of side-effects is bound to individual components
- For application-wide logic, *I recommend Sagas!*

DEMO

Sagas Can Handle Any Complexity

```
function* composeSendHandler() {  
  const jobId = makeJobId();  
  yield put(startJob(jobId, 'Sending email'));  
  
  const composeData = yield select(getComposeData);  
  const account = yield select(getAccount);  
  
  // ...  
  
  const sendStatus = yield call(sendMail, message);  
  if (sendStatus) {  
    yield put(sendError(sendStatus));  
  }  
  yield put(completeJob(jobId));  
}
```

Sources

- This presentation:
 - <https://oliversturm.github.io/react-patterns>
 - PDF download: <https://oliversturm.github.io/react-patterns/slides.pdf>
- Demo code:
 - <https://github.com/oliversturm/react-patterns>

Thank You

Please feel free to contact me about the content anytime.

Oliver Sturm • @olivers • oliver@oliversturm.com

