Using useContext and useReducer to Make a Redux-like Global App State

Peter Kellner
DEVELOPER, CONSULTANT AND AUTHOR

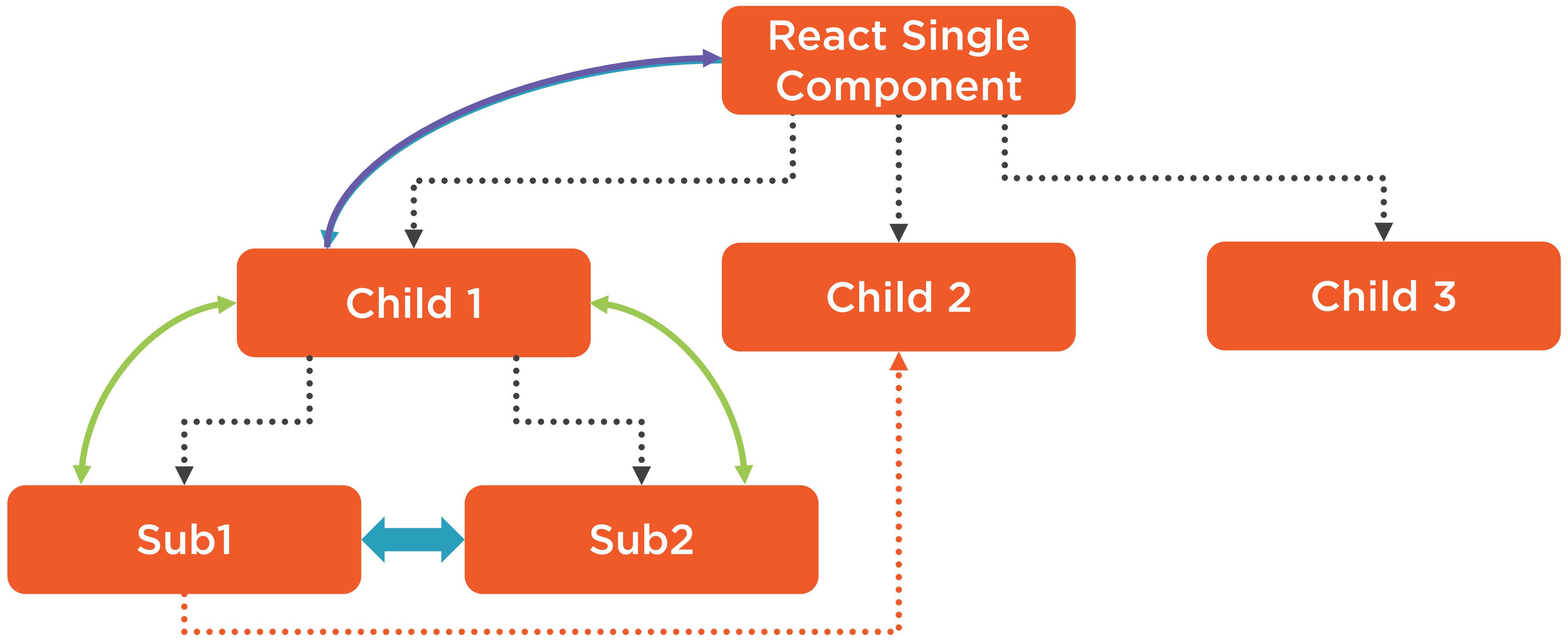
@pkellner   linkedin.com/in/peterkellner99   ReactAtScale.com

Bring together previous learnings

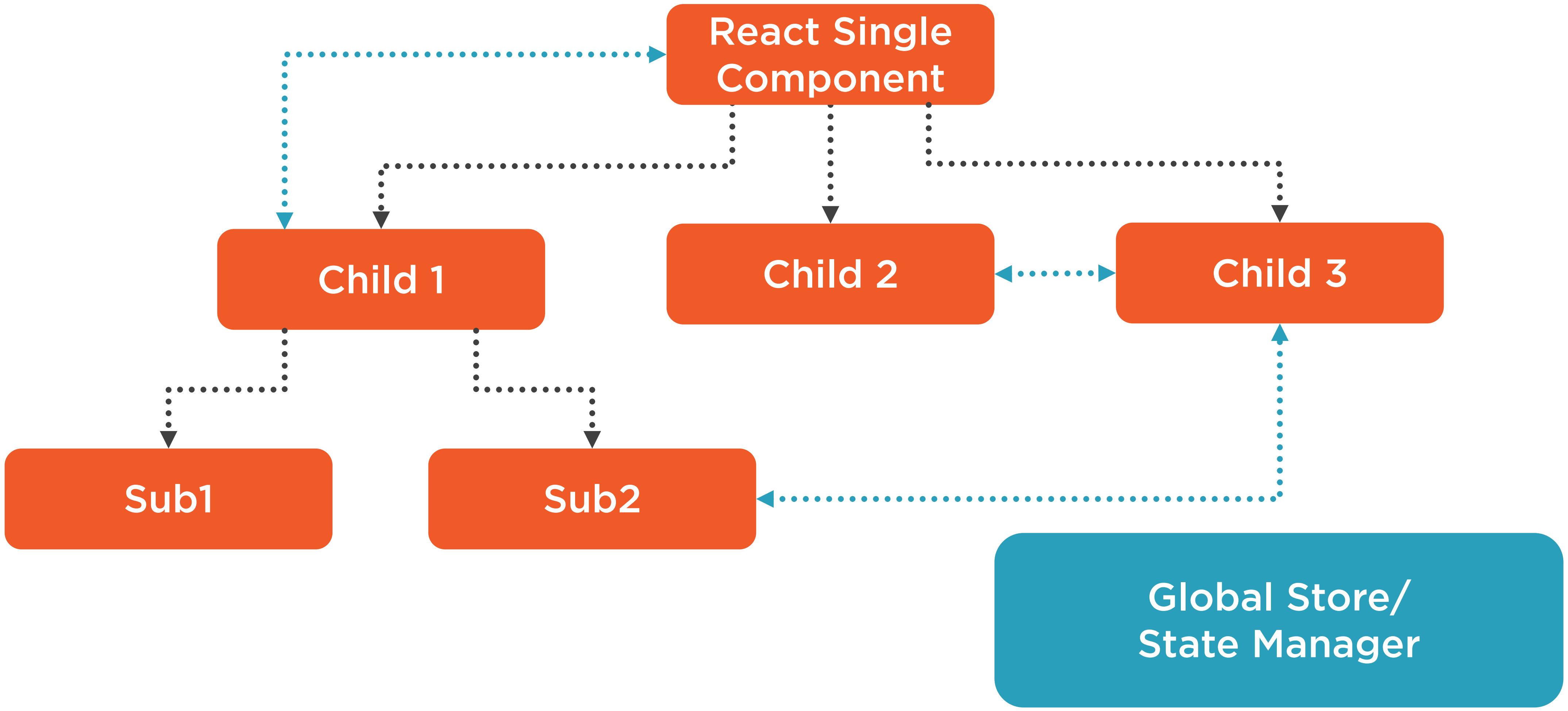
New design pattern for efficient state management across multiple components

Faster pace then previous modules

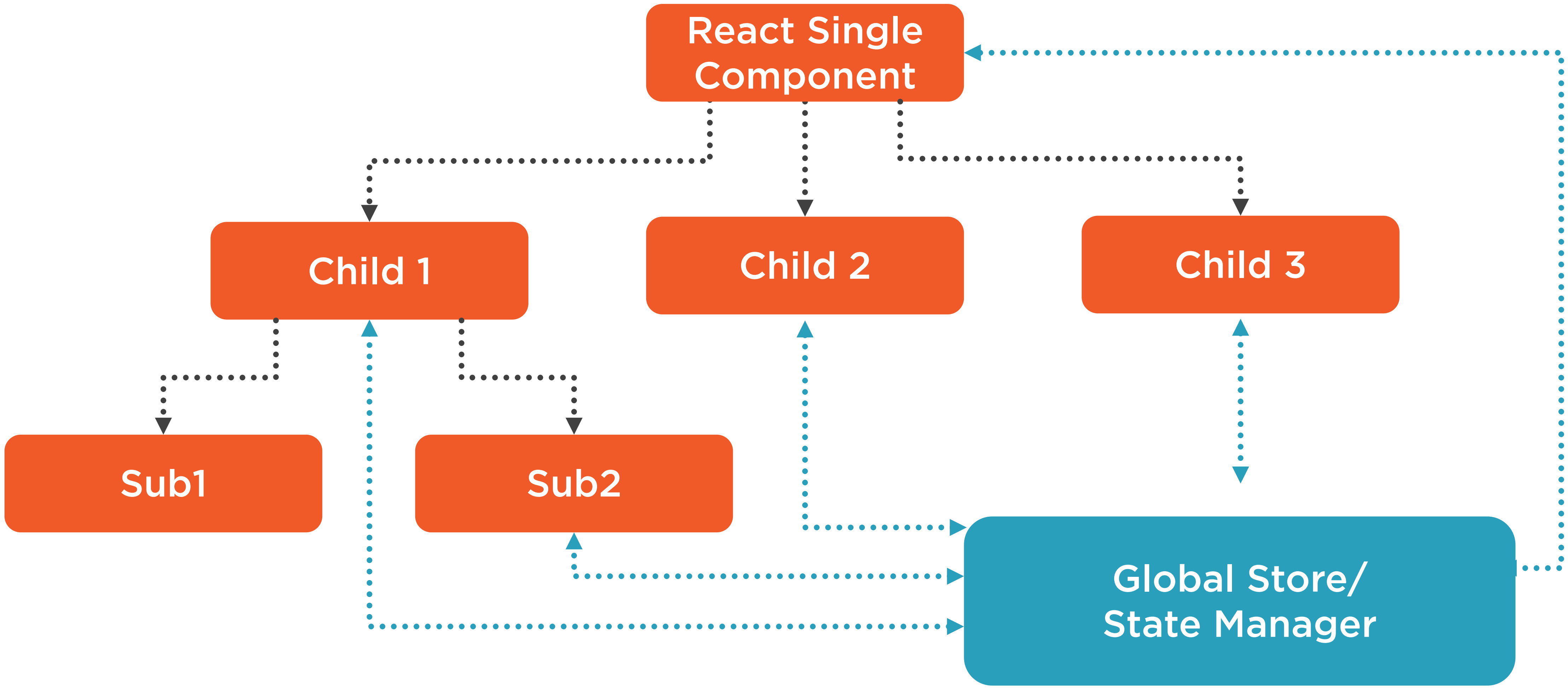
React Component Tree



React Component Tree



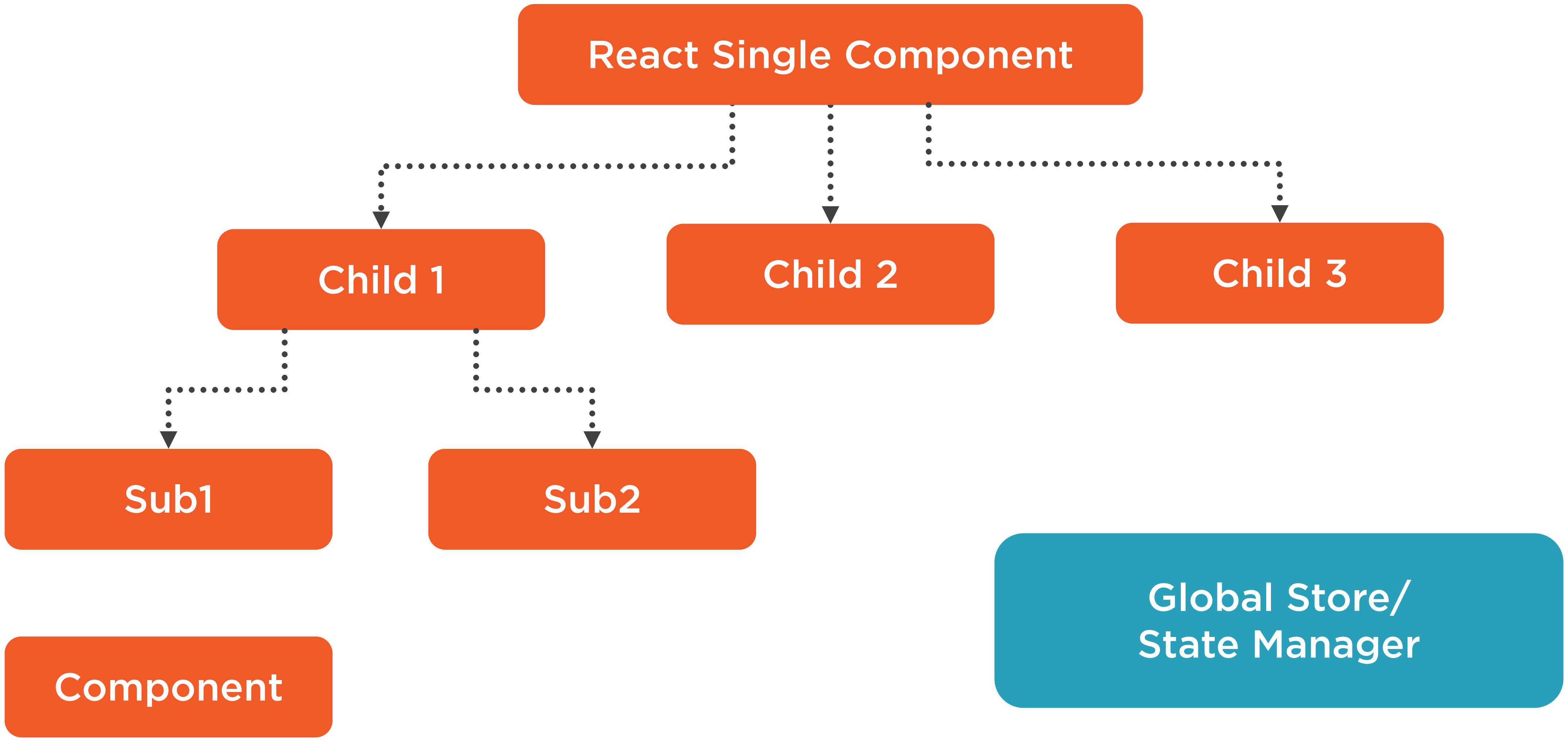
React Component Tree



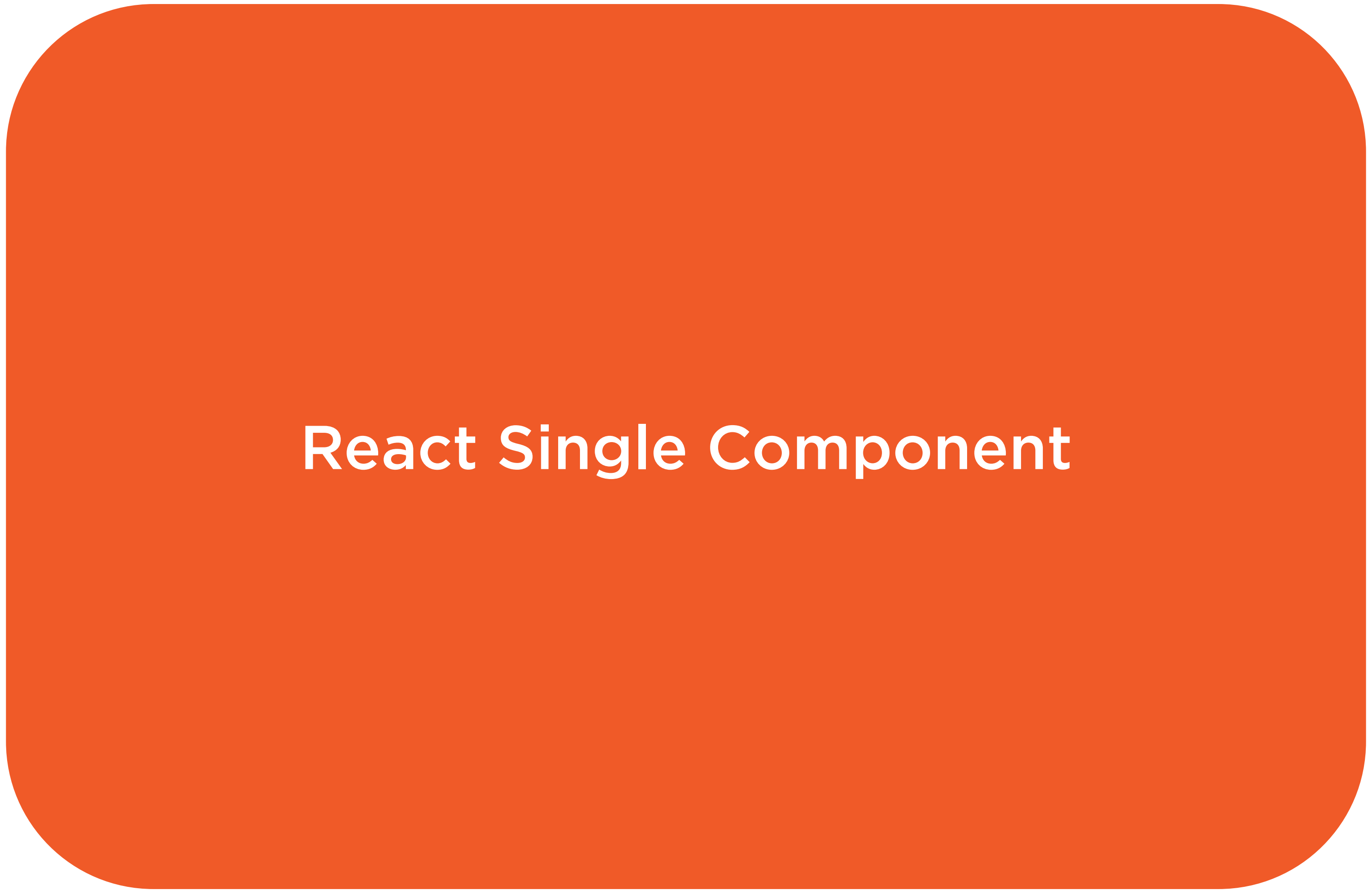
Redux

Redux can be thought of as a central store that holds an entire applications state. Each of the app components can access the stored state without requiring passing data or functions from one component to another.

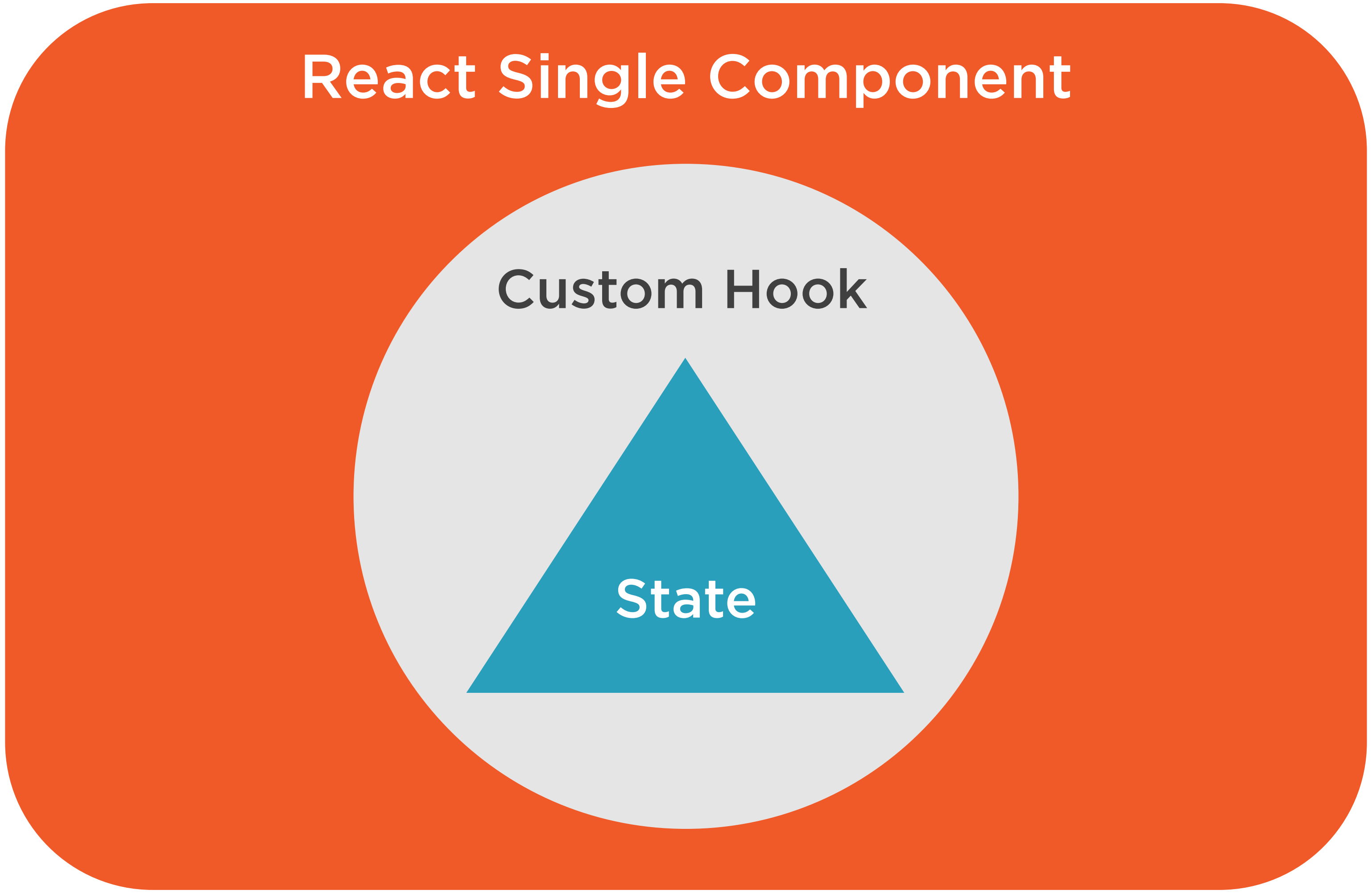
React Component Tree



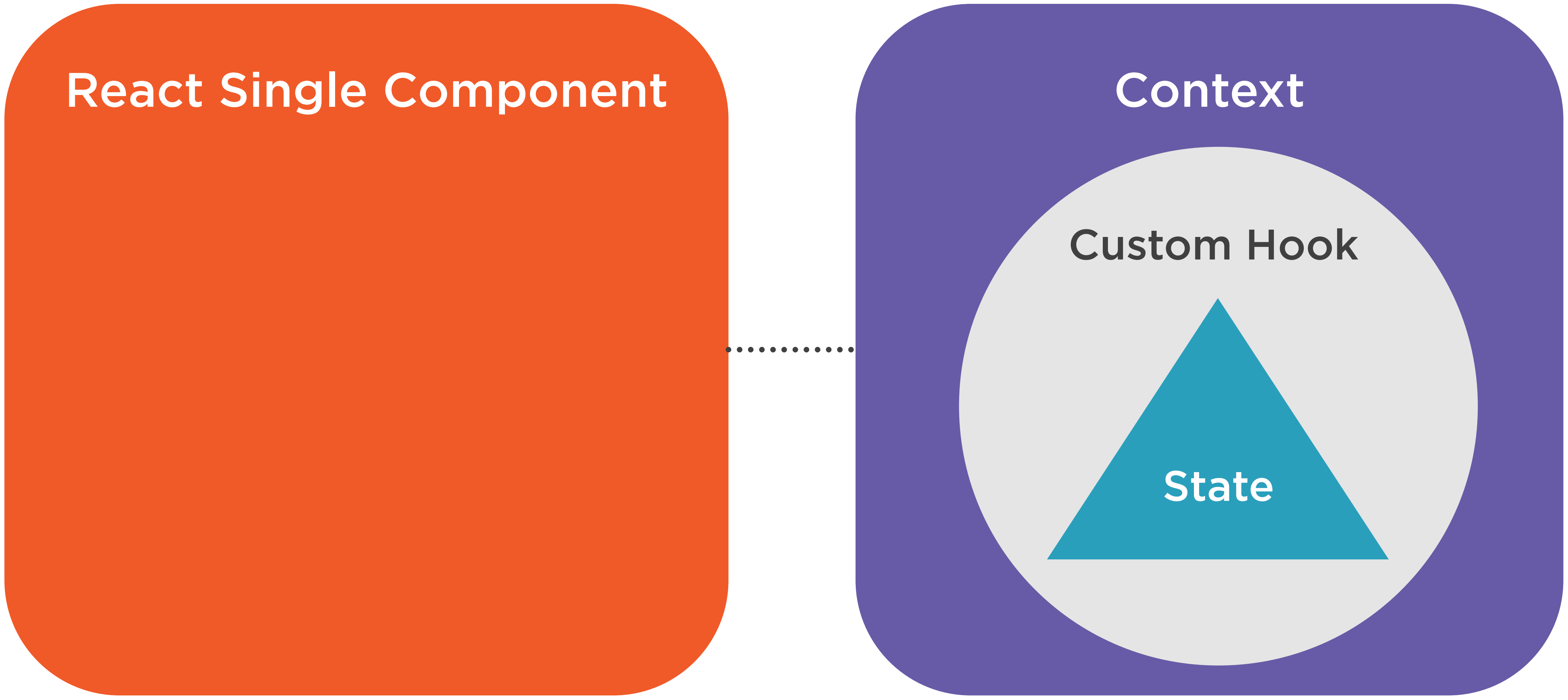
React Component Tree



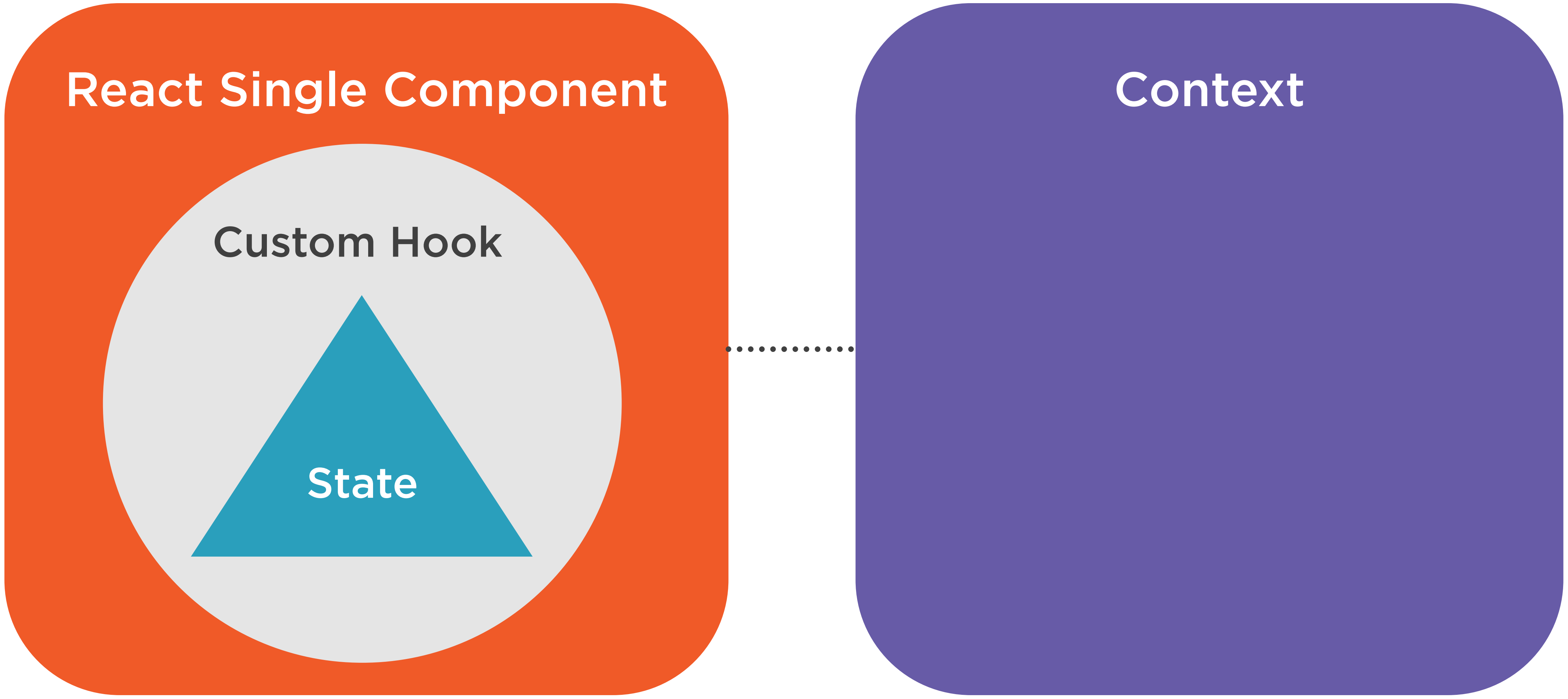
React Component Tree



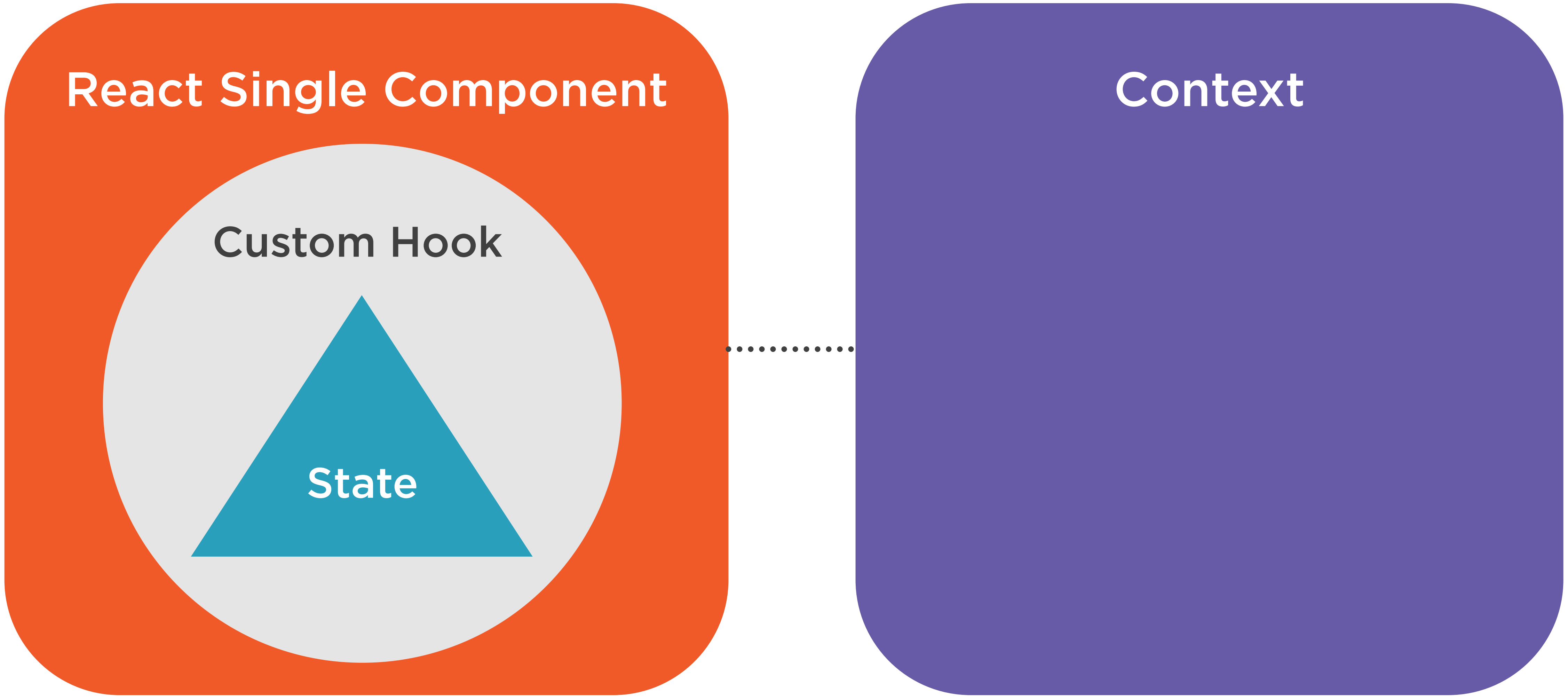
React Component Tree



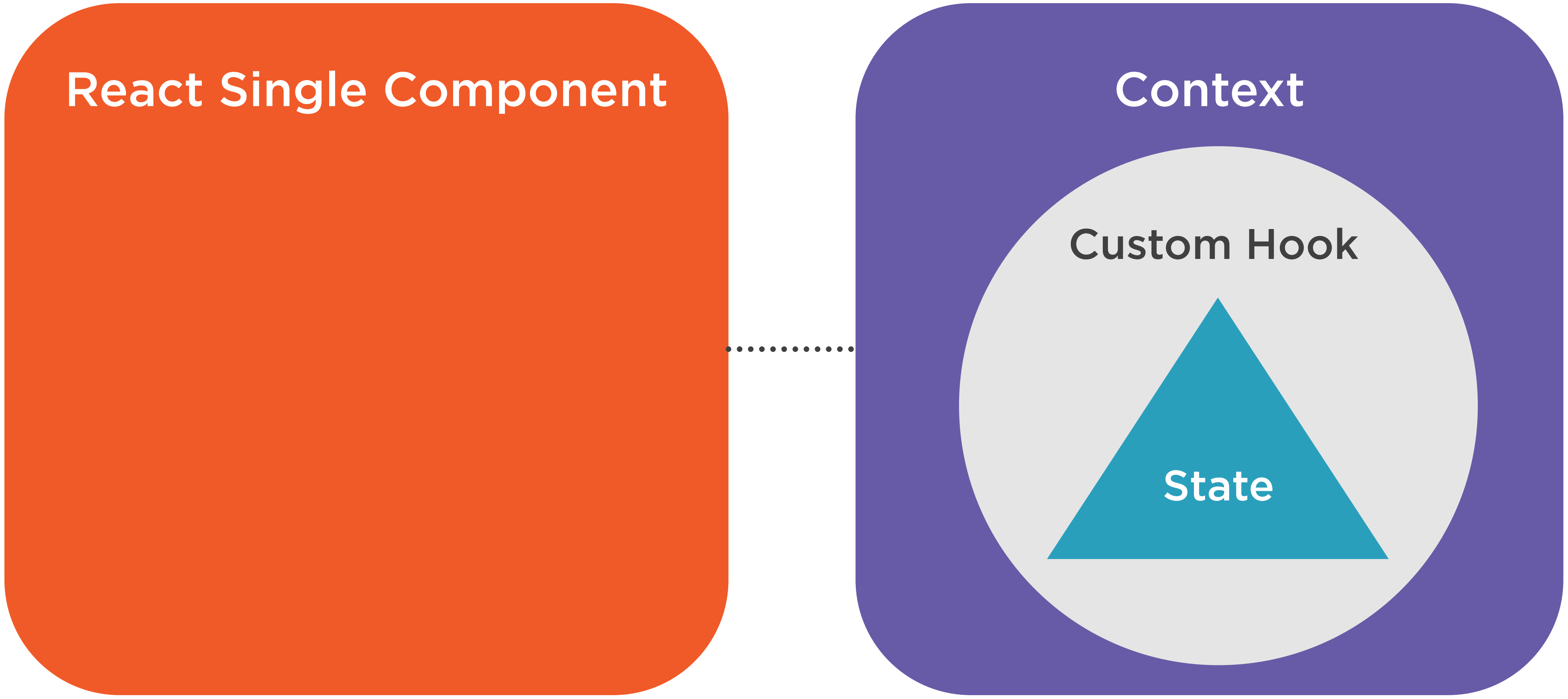
Application with Global Context



Application with Global Context



Application with Global Context



Extend

Coming up, use React Context in non-trivial ways

Adding Error Handling

Goal to learn more about sharing data across component hierarchy

Add two new state variables, hasErrored and error

Add try/catch to REST axios call and set appropriate state variables

Takeaways and Course Wrap

Bringing together multiple concepts

- More robust
- More scalable
- Easier to reason about

Segregate and de-couple state management

- Write apps faster
- Easier to maintain and test
- Improve programmer experience

Functional components with React Hooks are the future for React apps

Stay safe