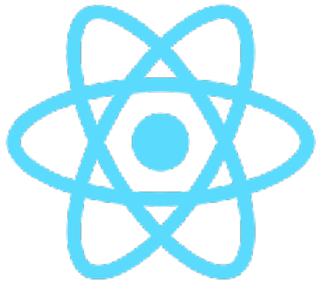
【图解】Redux 工作原理

Yetty
2017.11.28

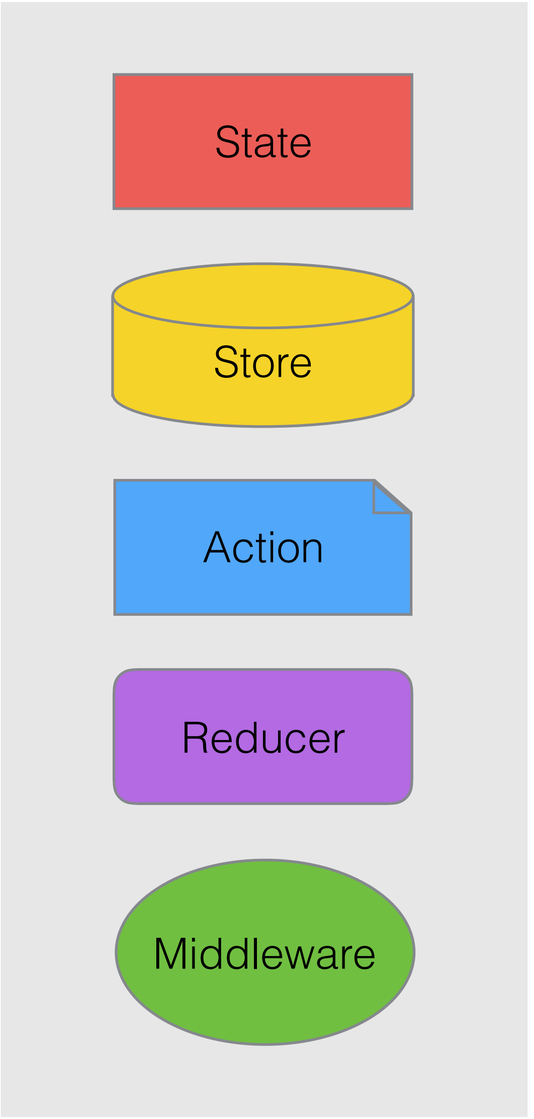
JavaScript 状态容器

提供可预测化的状态管理



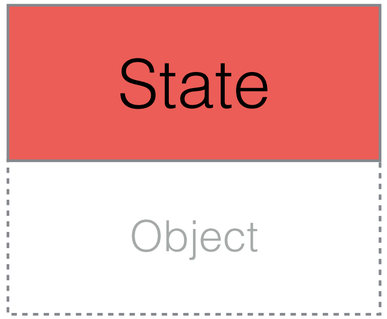
React-redux

Nerv-redux

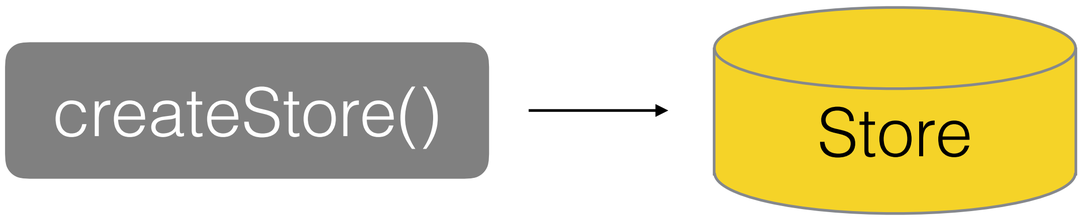


State

【State】应用的状态数据

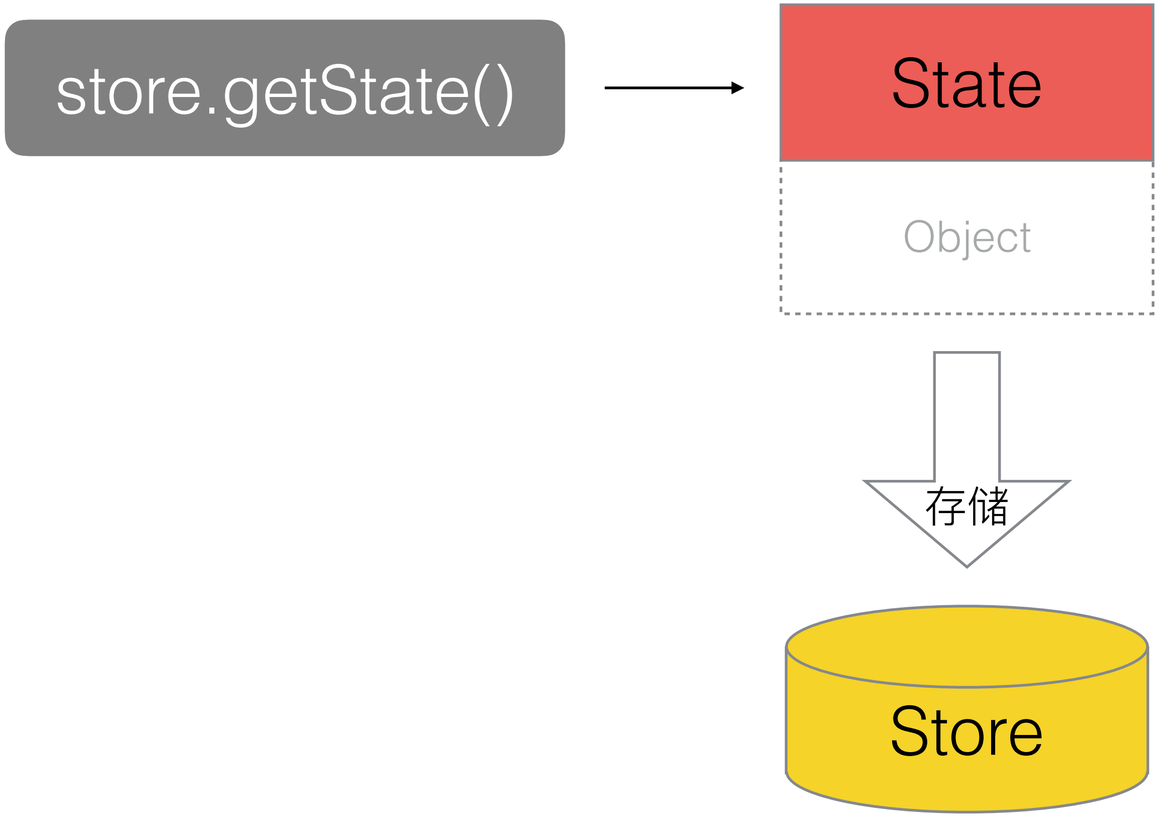


Store



【Store】应用唯一保存数据的容器

Store



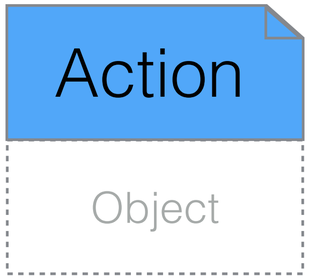
【Store】应用唯一保存数据的容器

Action

必须，表示 Action 的名称

携带的信息

```
const action = {
  type: 'ADD_TODO',
  payload: 'Learn Redux'
};
```



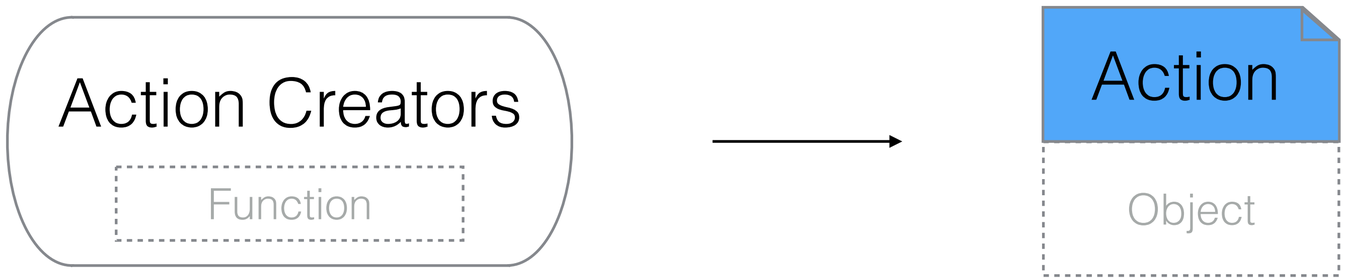
【Action】数据从应用传递到 Store 的载体

Action


```
const ADD_TODO = '添加 TODO';

function addTodo(text) {
  return {
    type: ADD_TODO,
    text
  }
}

const action = addTodo('Learn Redux');
```



【Action Creators】一个用来生成 Action 的函数

Reducer

当前 State

```
const reducer = function (state, action) {
  // ...
  return new_state;
};
```

Reducer

Function

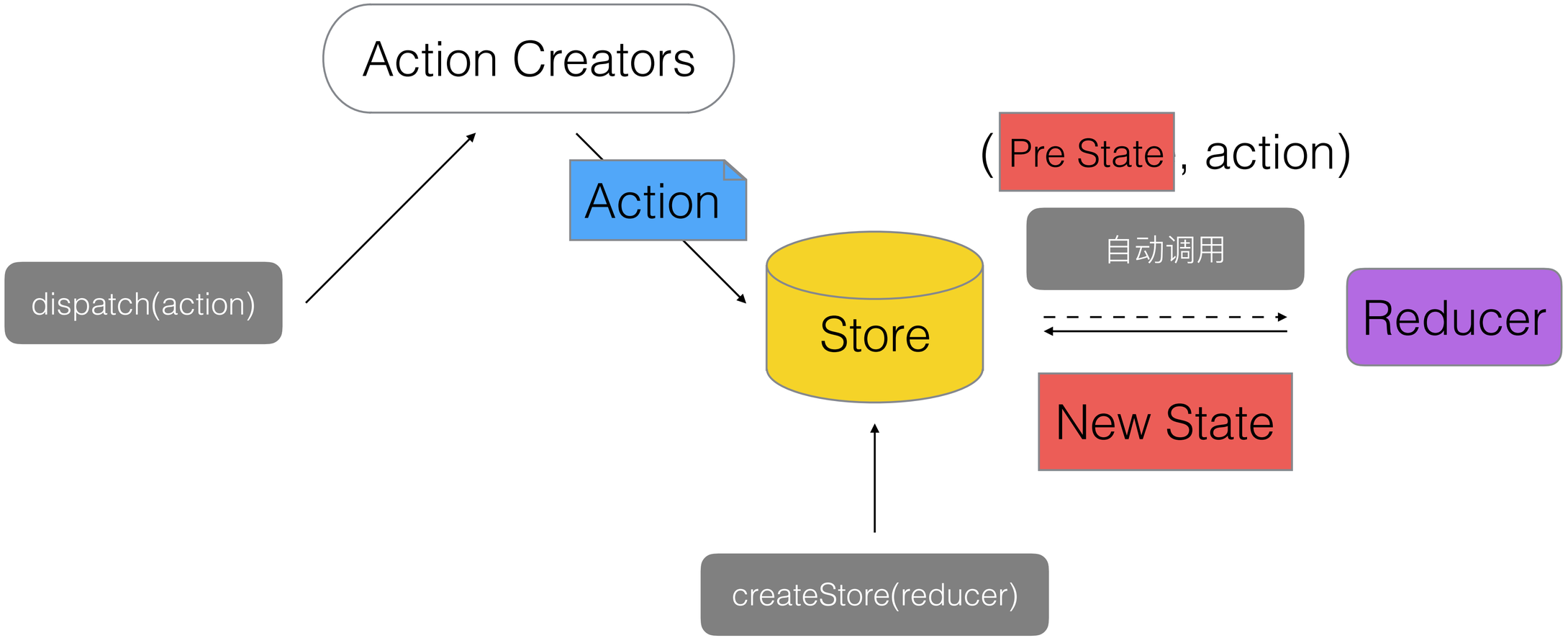
【Reducer】指明应用如何改变 State

```
const state = reducer(1, {
  type: 'ADD',
  payload: 2
});
```

createStore(reducer)

以后每当 store.dispatch 发送过来一个新的 Action，就会自动调用 Reducer，得到新的 State

Redux 的工作流程


Action Creators
Action
dispatch(action)
Store
(Pre State , action)
自动调用
Reducer
New State
createStore(reducer)

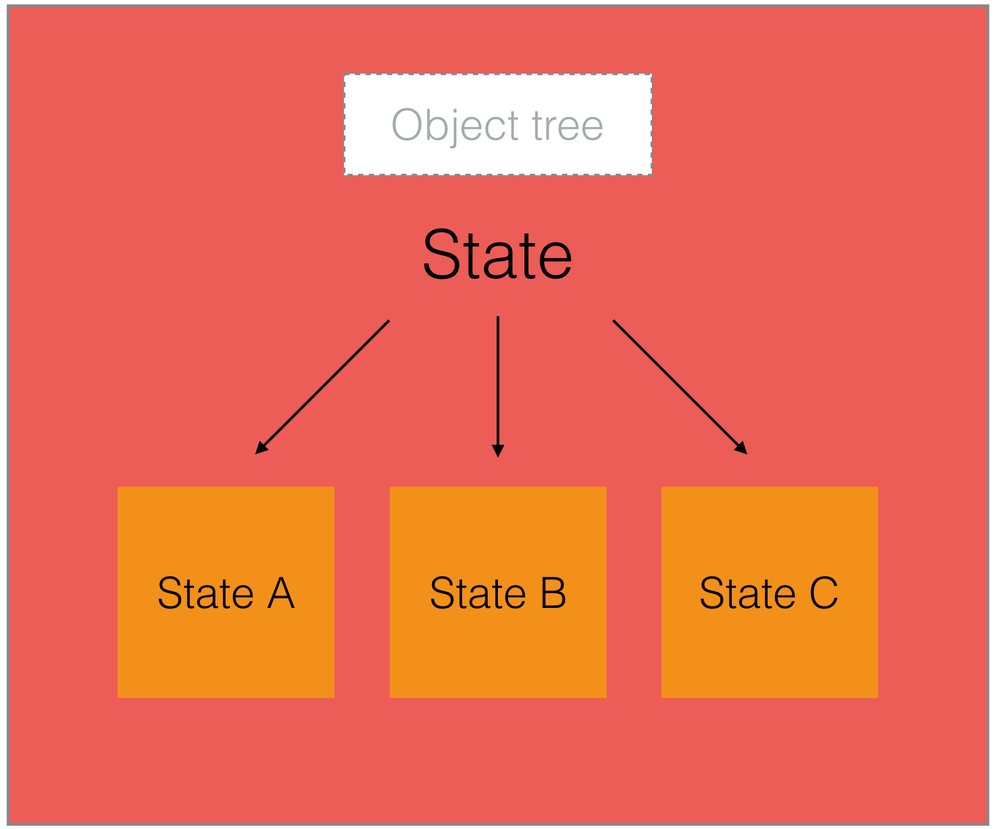
Reducer 拆分

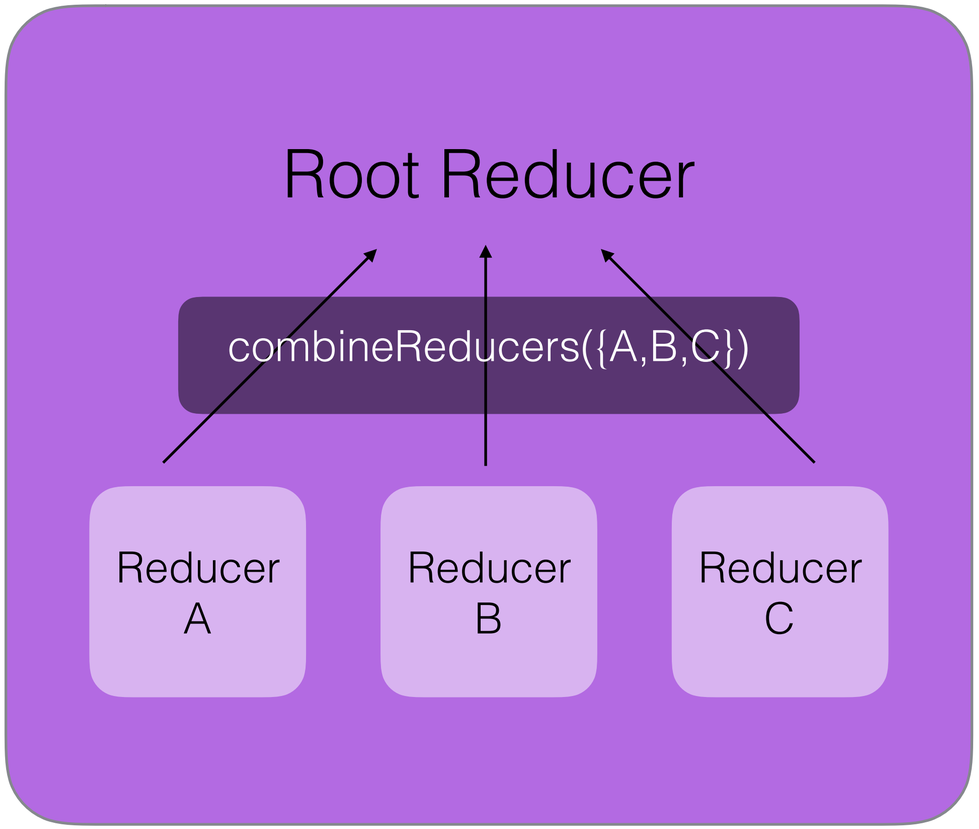
大型应用：

State 庞大 => Reducer 庞大

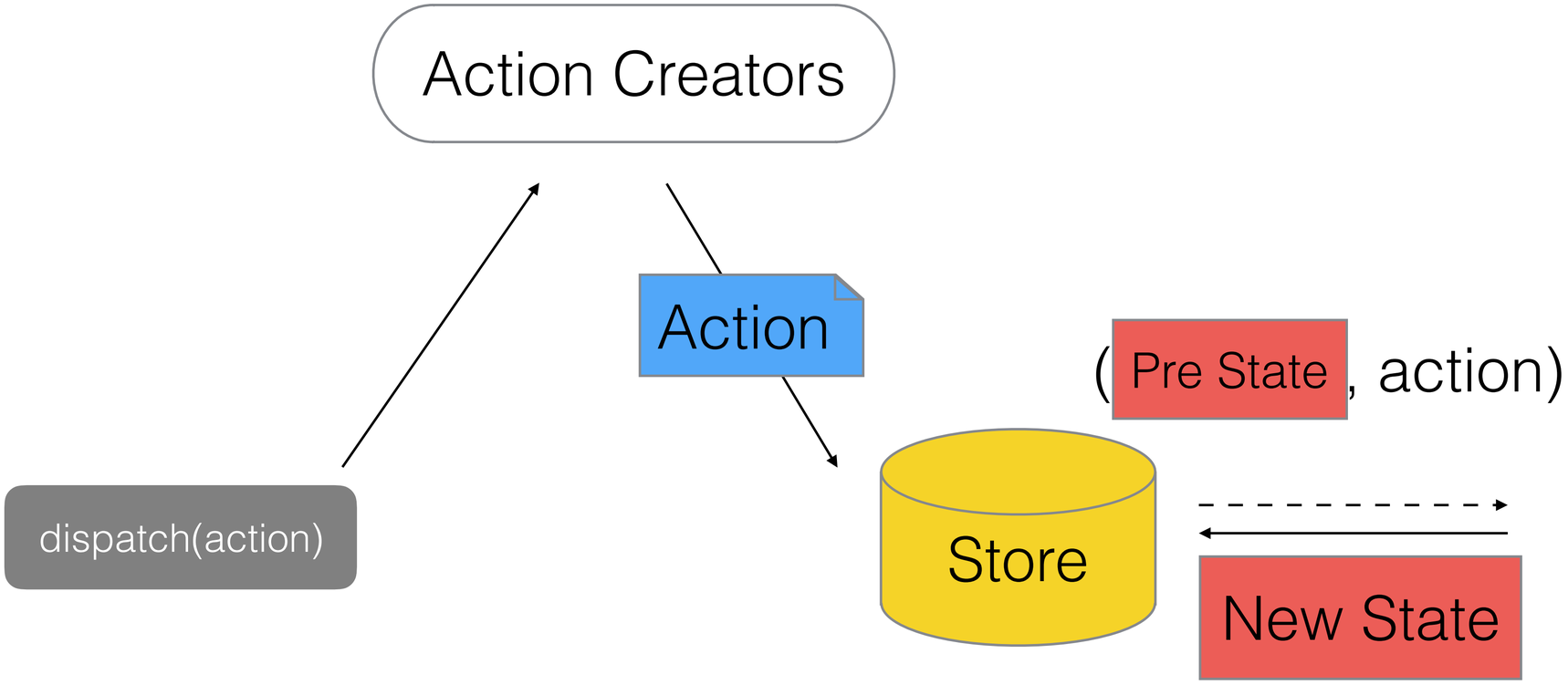
```
const chatReducer = (state = defaultState, action = {}) => {
  const { type, payload } = action;
  switch (type) {
    case ADD_CHAT:
      return Object.assign({}, state, {
        chatLog: state.chatLog.concat(payload)
      });
    case CHANGE_STATUS:
      return Object.assign({}, state, {
        statusMessage: payload
      });
    case CHANGE_USERNAME:
      return Object.assign({}, state, {
        userName: payload
      });
    default: return state;
  }
};
```

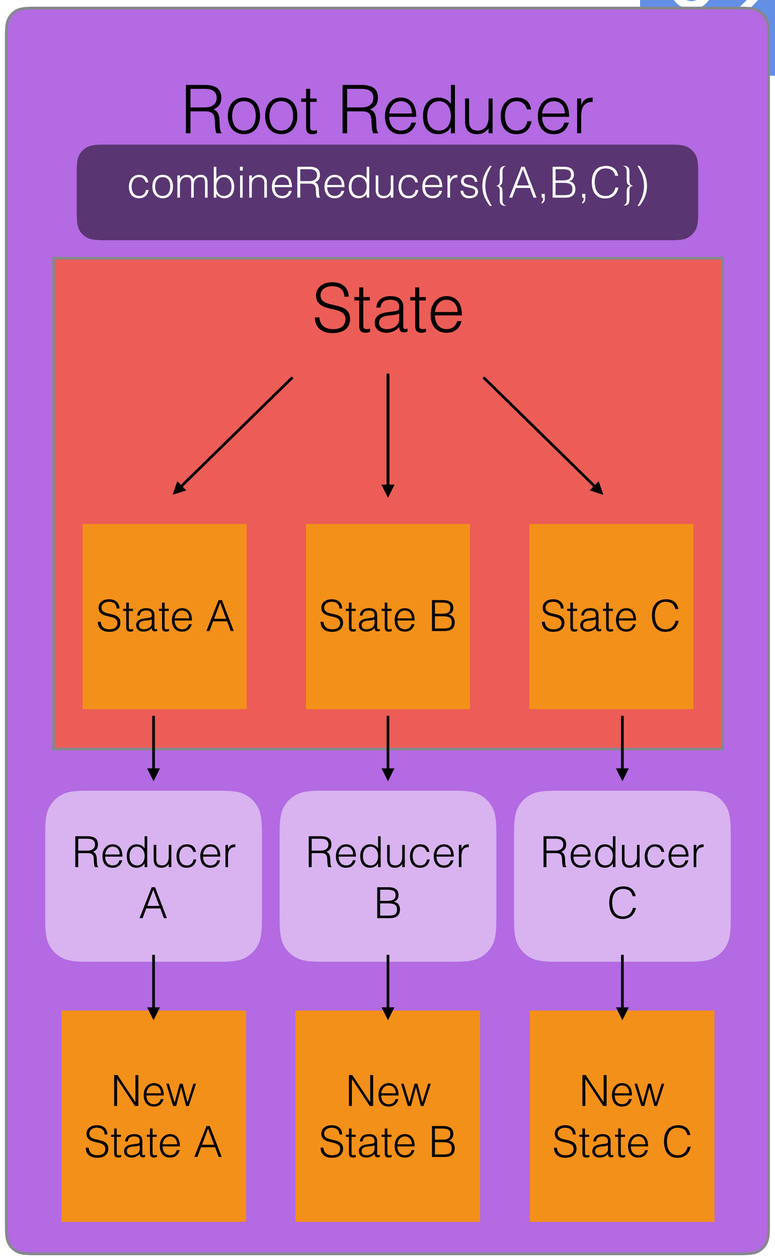
Reducer 拆分


Object tree
State
State A
State B
State C


Root Reducer
combineReducers({A,B,C})
Reducer A
Reducer B
Reducer C

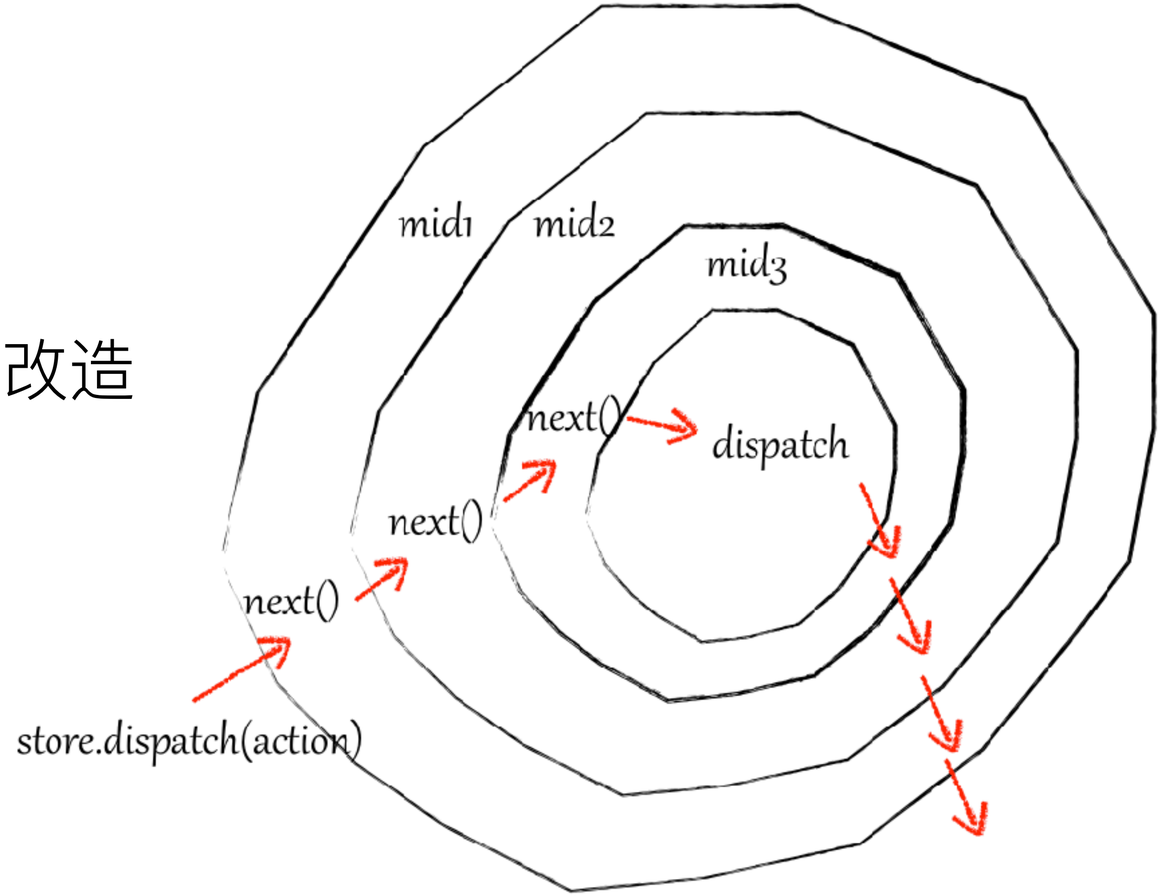
Redux 的工作流程—Reducer 拆分



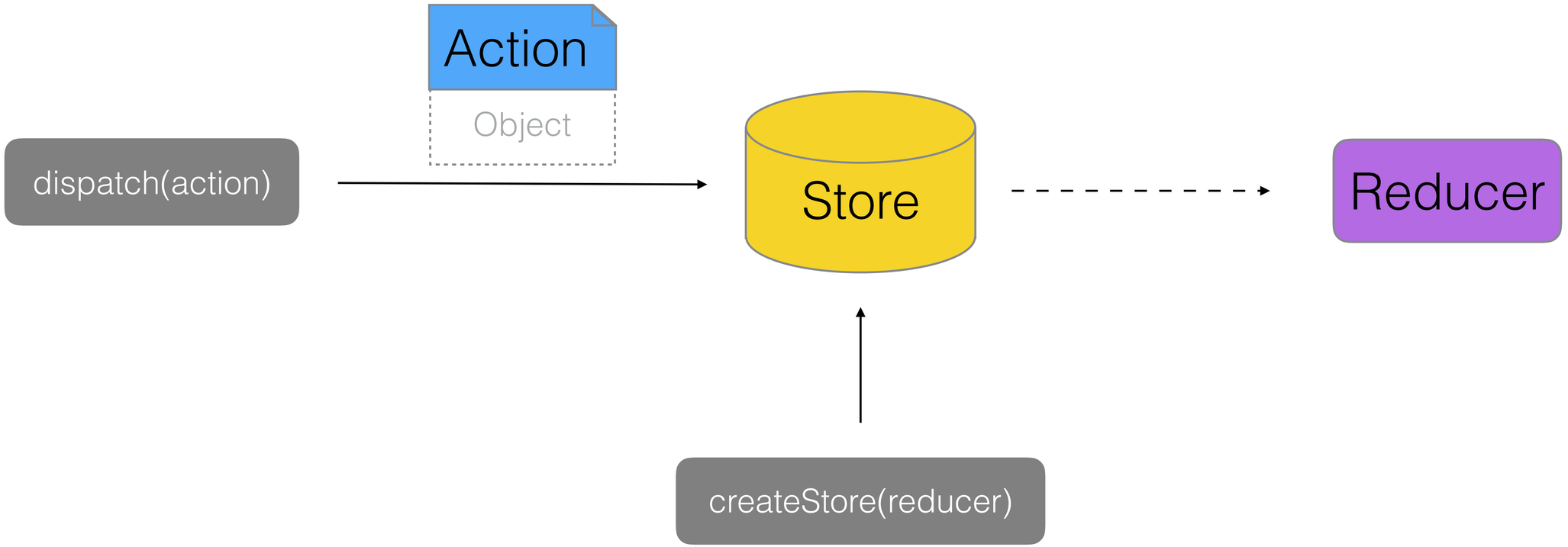


Middleware

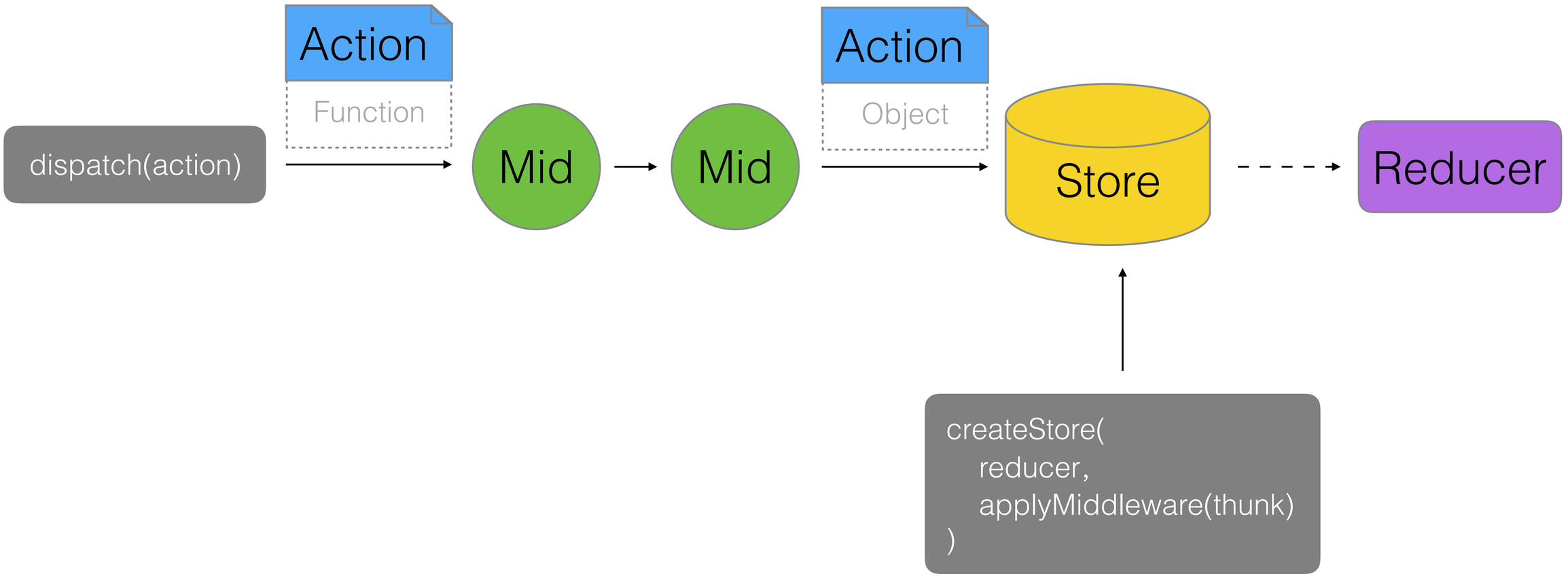
【Middleware】对 store.dispatch 方法进行改造



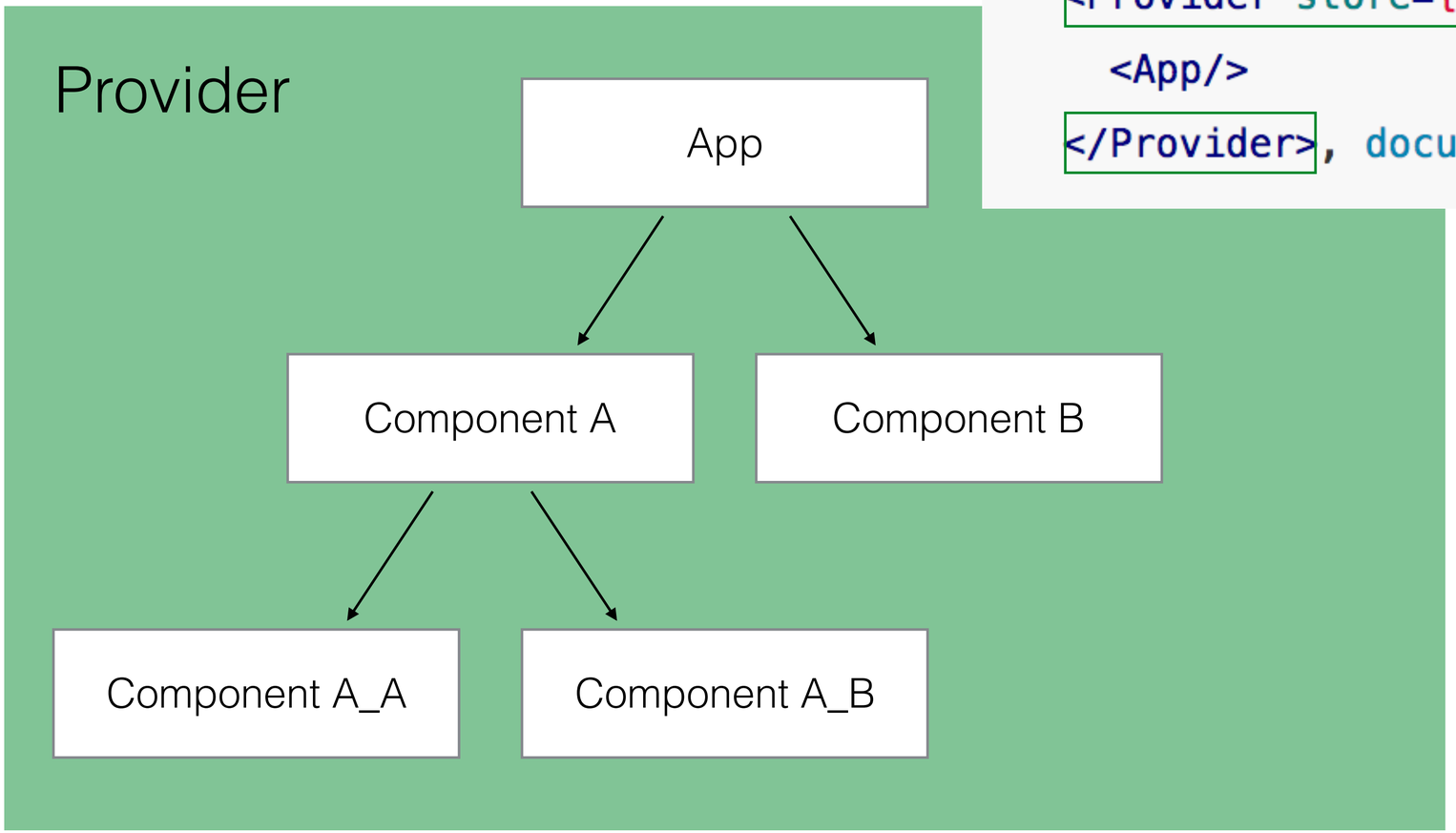
Middleware


Action
Object
dispatch(action)
Store
Reducer
createStore(reducer)

Middleware


dispatch(action)
Action
Function
Mid
Mid
Action
Object
Store
Reducer
createStore(
reducer,
applyMiddleware(thunk)
)

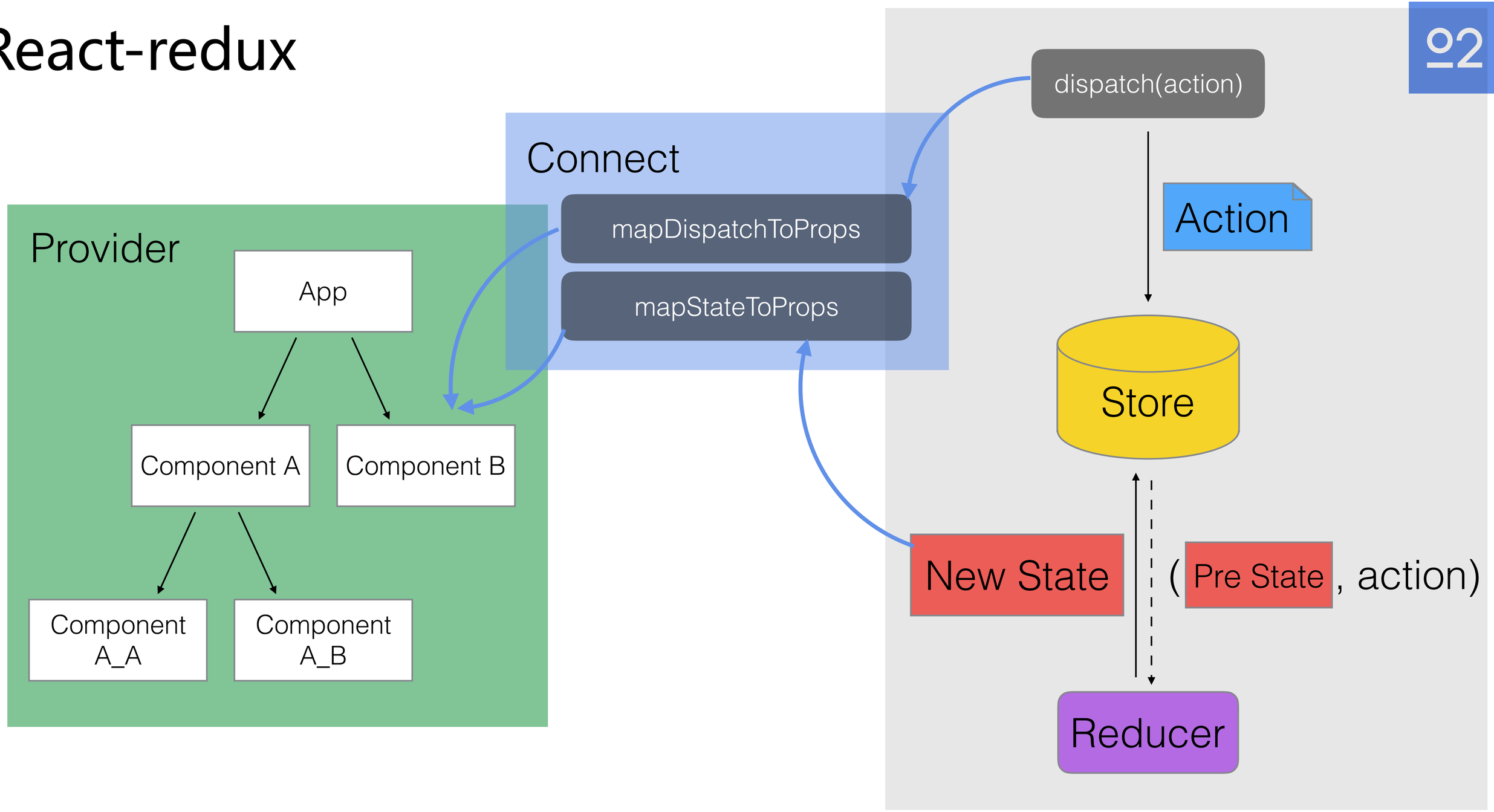
React-redux




```
render(
  <Provider store={store}>
    <App/>
  </Provider>, document.getElementById('app-node'))
```

【Provider】把 Store 提供给它的子组件

React-redux


Connect
mapDispatchToProps
mapStateToProps
Provider
App
Component A
Component B
Component A_A
Component A_B
dispatch(action)
Action
Store
New State
(Pre State , action)
Reducer