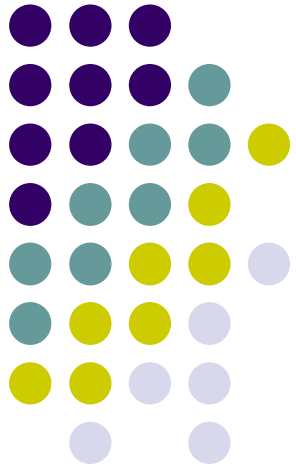
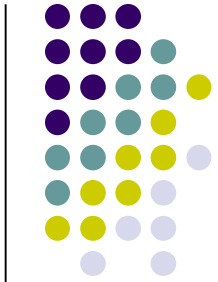
React Redux

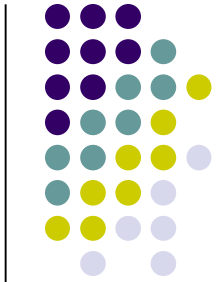


Redux



"Redux is a predictable state container for JavaScript apps“

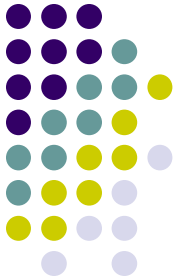
- It is for JavaScript apps
- It is a state container
- It is predictable



Redux is for JavaScript applications

- Redux is not tied to React
- Can be used with React, Angular, Vue or even vanilla JavaScript
- Redux is a library for JavaScript applications.

Redux is a state container



Redux stores the state of your application

Consider a React app - state of a component

State of an app is the state represented by all the individual components of that app

Redux will store and manage the application state

LoginFormComponent

```
state = {
  username: '',
  password: '',
  submitting: false
}
```

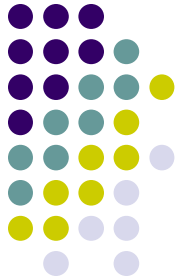
UserListComponent

```
state = {
  users: [ ]
}
```

Application

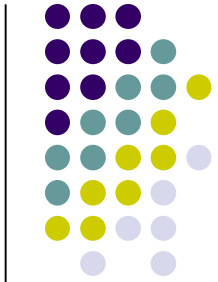
```
state = {
  isUserLoggedIn: true,
  username: 'Vishwas',
  profileUrl: '',
  onlineUsers: [ ],
  isModalOpened: false
}
```

Redux is predictable



- Predictable in what way?
- Redux is a state container
- The state of the application can change
- Ex: todo list app-item (pending)→item (completed)
- In redux, all state transitions are explicit and it is possible to keep track of them
- The changes to your application's state become predictable

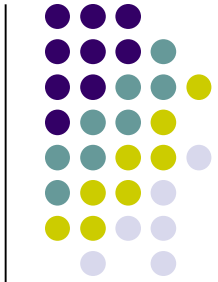
Redux



"Redux is a predictable state container for JavaScript apps"

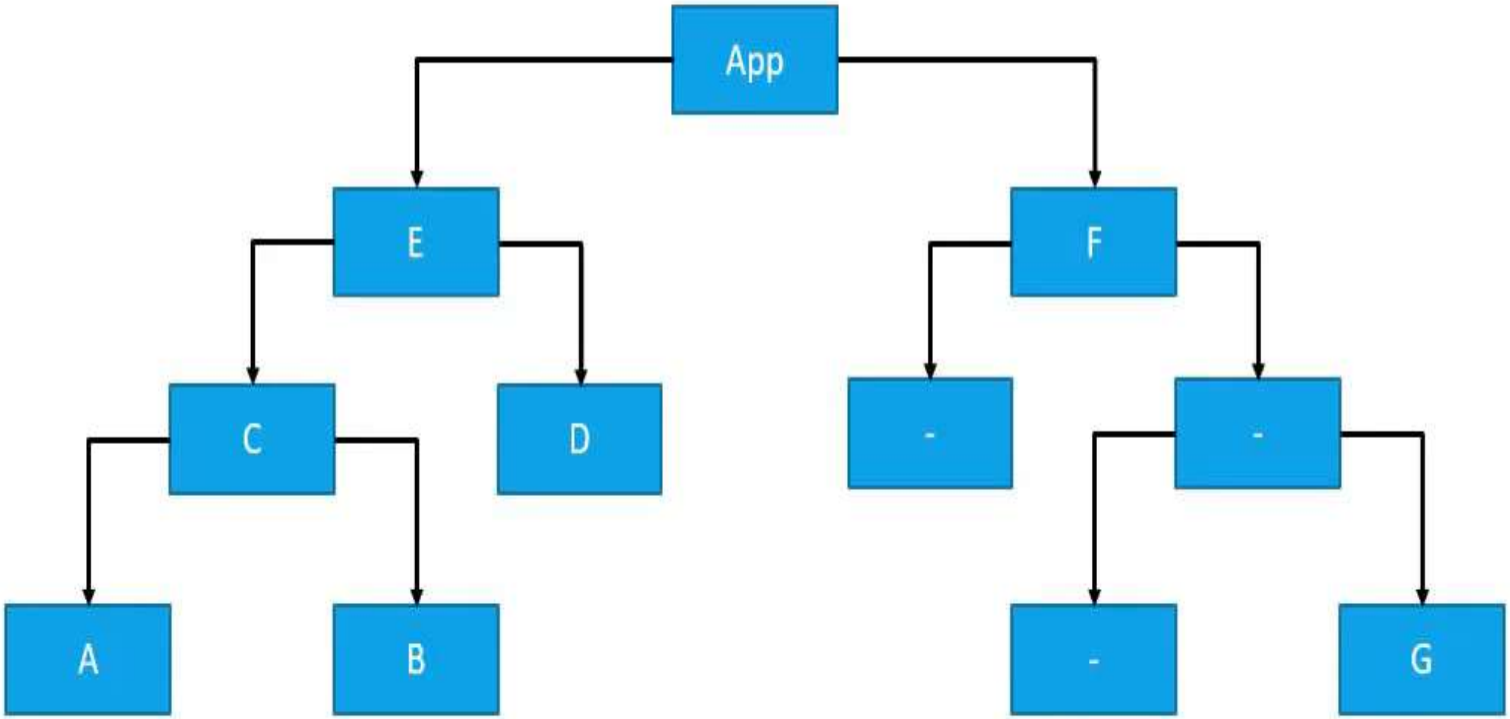
Manage the state of your application in a predictable way, redux can help you.

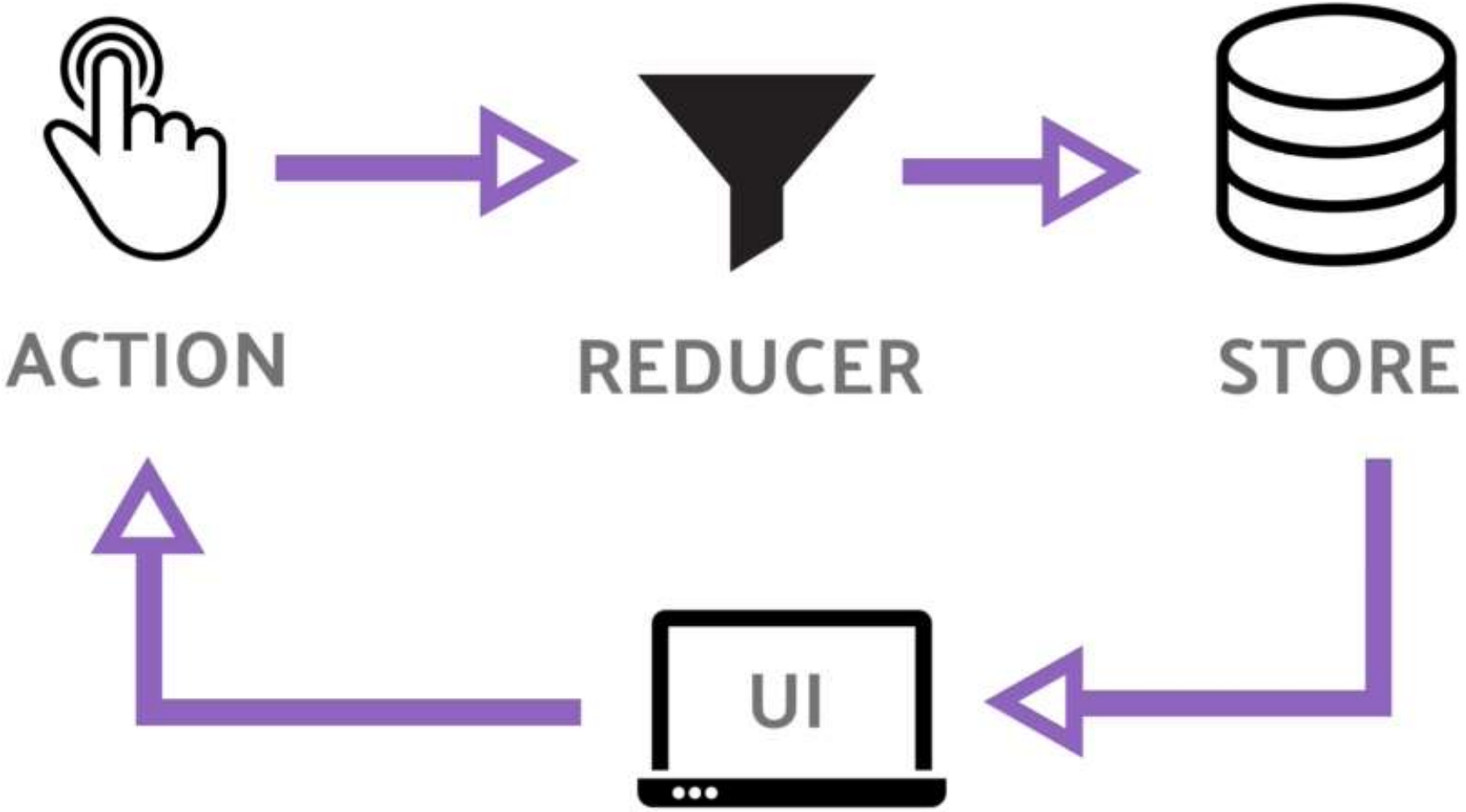
React+Redux?

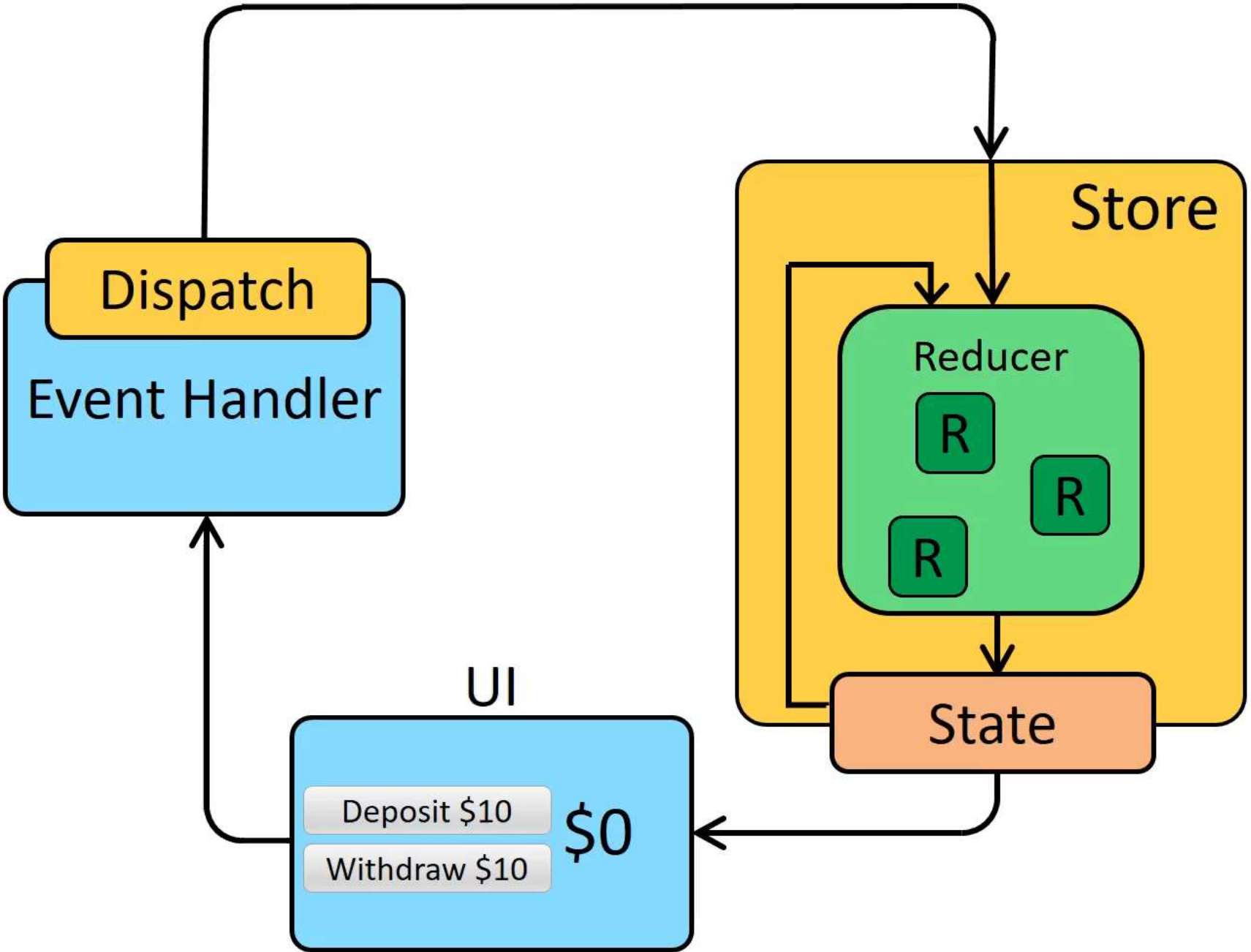


- Why would we want to use redux in a react application?
- Components in React have their own state
- Why do we need another tool to help manage that state?

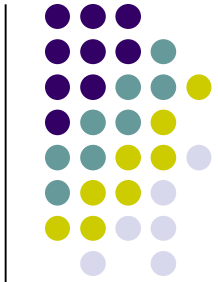
State in a React App




ACTION
REDUCER
STORE
UI

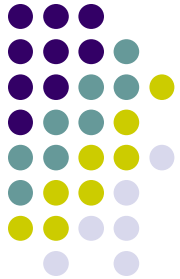

Store
Dispatch
Event Handler
Reducer
R
R
R
State
UI
Deposit $10
Withdraw $10
$0

Why..?



- Redux helps you manage "global" state - state that is needed across many parts of your application.
- The patterns and tools provided by Redux make it easier to understand when, where, why, and how the state in your application is being updated, and how your application logic will behave when those changes occur.
- Redux guides you towards writing code that is predictable and testable, which helps give you confidence that your application will work as expected.

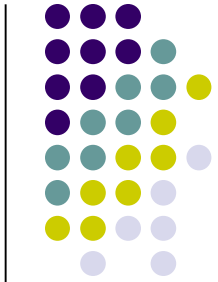
When..?



Redux is more useful when:

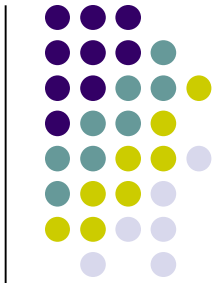
- You have large amounts of application state that are needed in many places in the app
- The app state is updated frequently over time
- The logic to update that state may be complex
- The app has a medium or large-sized codebase, and might be worked on by many people

Blocks of Redux



Actions:

- Actions are the only source of information for the store.
- Actions have a type field that tells what kind of action to perform and all other fields contain information or data.
- And there is one other term called Action Creators, these are the function that creates actions.
- So actions are the information (Objects) and action creator are functions that return these actions.



Reducers:

As we already know, actions only tell what to do, but they don’t tell how to do, so reducers are the pure functions that take the current state and action and return the new state and tell the store how to do.

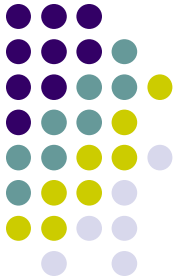
Store:

The store is the object which holds the state of the application

Dispatcher:

To change something in the state, you need to dispatch an action.

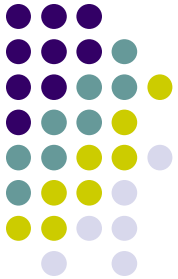
mapStateToProps



- If mapStateToProps argument is specified, the new component will subscribe to Redux store updates.
- This means that any time the store is updated, mapStateToProps will be called.
- The results of mapStateToProps must be a plain object, which will be merged into the component's props.

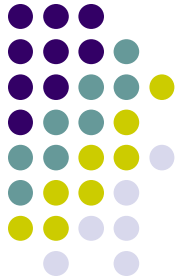
function mapStateToProps(state, ownProps?)

mapDispatchToProps



- Providing a mapDispatchToProps allows you to specify which actions your component might need to dispatch.
- It lets you provide action dispatching functions as props.
- Dispatch is a function of the Redux store. You call store.dispatch to dispatch an action. This is the only way to trigger a state change.

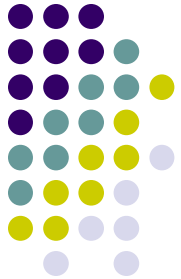
Connect



- React Redux provides a connect function for you to read values from the Redux store (and re-read the values when the store updates).
- The connect function takes two arguments, both optional:
  - mapStateToProps
  - mapDispatchToProps

connect(mapStateToProps,mapDispatchToProps)(Component)

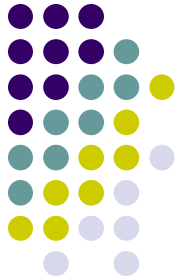
useSelector



- Redux provides a useSelector hook for easy access to state within a function component.
- This hook replaces the mapStateToProps function which was used previously with connect().
- The hooks API is a lot simpler and can be combined nicely with selector functions.

const value = useSelector(state => state.reducerName.value)

useDispatch



- Redux provides a useDispatch hook to make it easy to dispatch actions within a function component.
- This hook replaces the mapDispatchToProps function which was used previously with connect().

```
const dispatch = useDispatch();
```

- React Redux
- React + Redux
- Redux Architecture
- Redux Work Flow
- Why Redux.
- When to Use.
- Blocks of Redux
  - Actions
  - Reducer
  - Store
- Dispatcher
- Connect
- mapStateToProps
- mapDispatchToProps
- UseSelector
- UseDispatch


QUERIES