ReactJS

Redux: an introduction

by
Saikat Bhattacharya

Recap

- Flux: An architecture developed by Facebook
- One-way data binding: makes life simpler
- Difference between traditional MVC and Flux
- Action, Dispatcher, Store and View

Today's plan

- Why do we need redux?
- Three principles of Redux
- Actions, stores and reducers
- Connection React to Redux

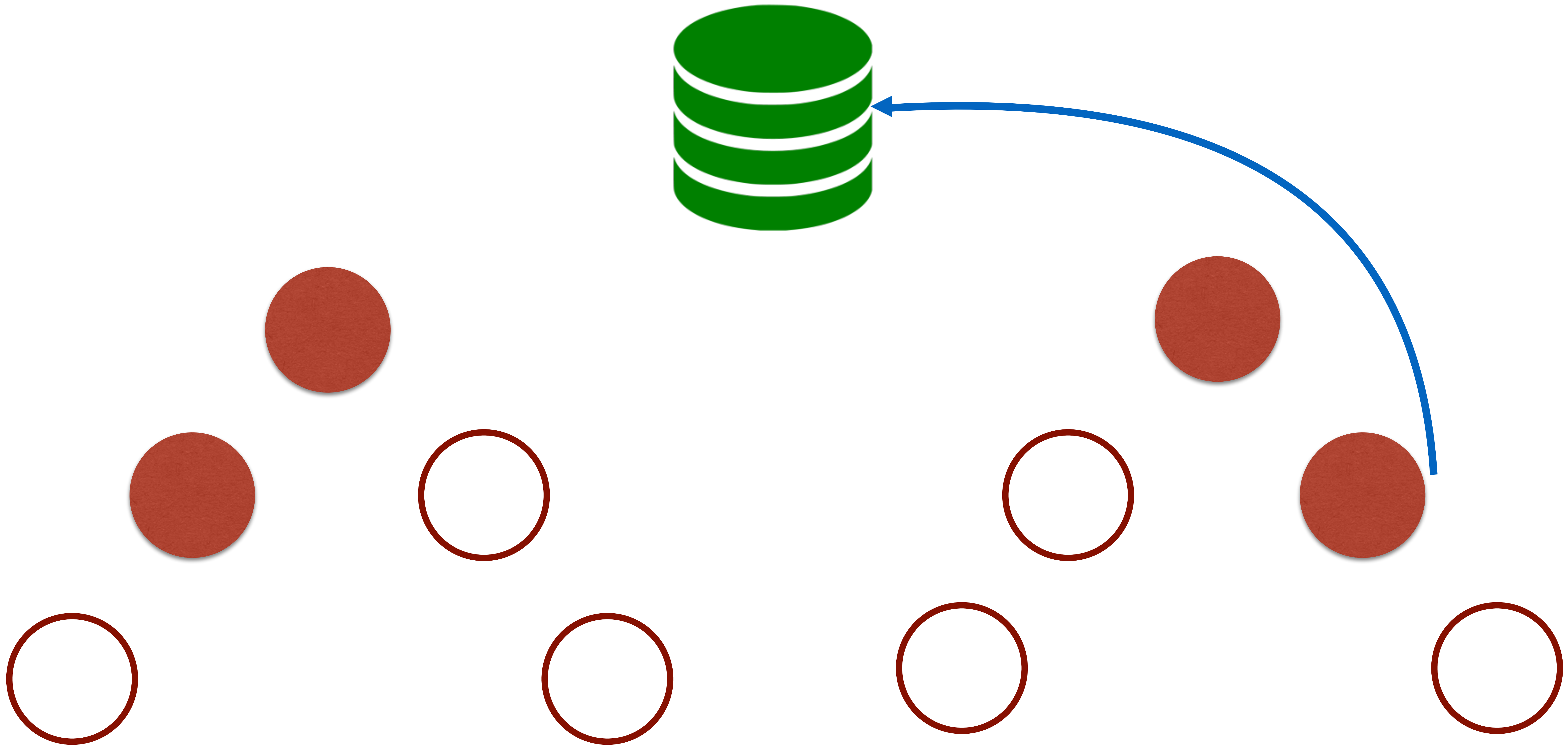
Why do I need Redux?

Redux

- Complex data flows
- Inter component communication
- Multiple actions
- Same data in multiple places

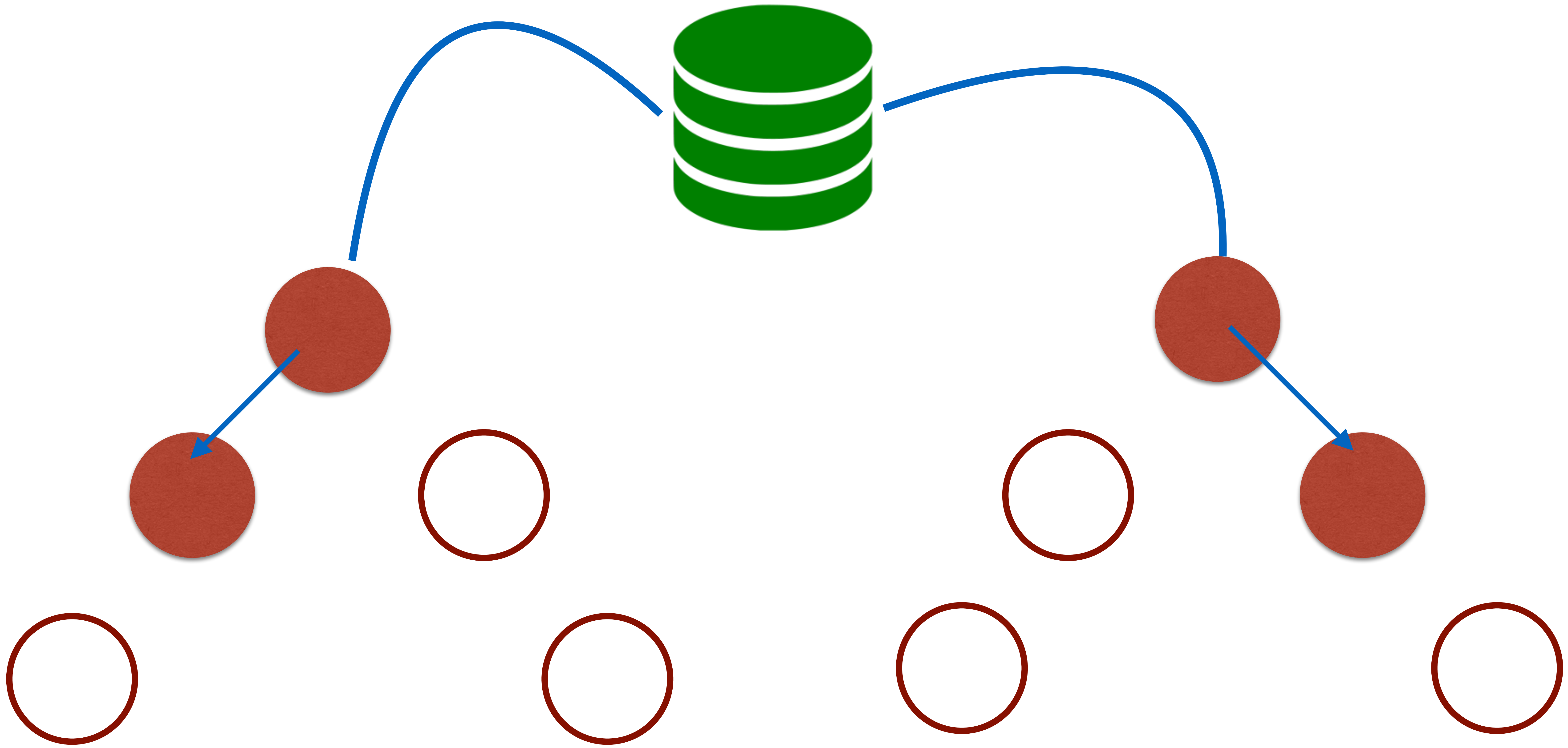
Why do I need Redux?

Redux



Why do I need Redux?

Redux



Three Principles

- Single source of truth
- Only action can trigger the change!
- Reducers can update state

Actions, stores and reducers

Actions

```
{type: ‘USER_CREATE’, username:username}
```

Actions, stores and reducers

Store

```
let store = createStore(reducer);
```

Actions, stores and reducers

Store

- `store.dispatch(action)`
- `store.subscribe(listener)`
- `store.getState()`

No API for changing data in store!!

Actions, stores and reducers

Reducer

Immutability

To change state, return a new object!

Actions, stores and reducers

Reducer

Immutability

```
state = {
     name: saikat,
     role: teacher
}
```

$\longrightarrow$

```
state.role = ‘learner’;
return state;
```

Actions, stores and reducers

Reducer

Immutability

```
state = {
    name: saikat,
    role: teacher
}
```

$\longrightarrow$

```
return state = {
    name: saikat,
    role: learner
}
```

```
Object.assign(target,…sources)
```

Actions, stores and reducers

Reducer

Immutability

```
state = {
    name: saikat,
    role: teacher
}
```

$$\longrightarrow$$

```
return Object.assign({},
    state,
    {role:’learner’}
    );
```

Actions, stores and reducers

Reducer

Why immutability?

- Clarity

  Who and when changed the sate?

- Performance

  No more costly operation to find if state parameter changed.

  It is just a memory comparison:

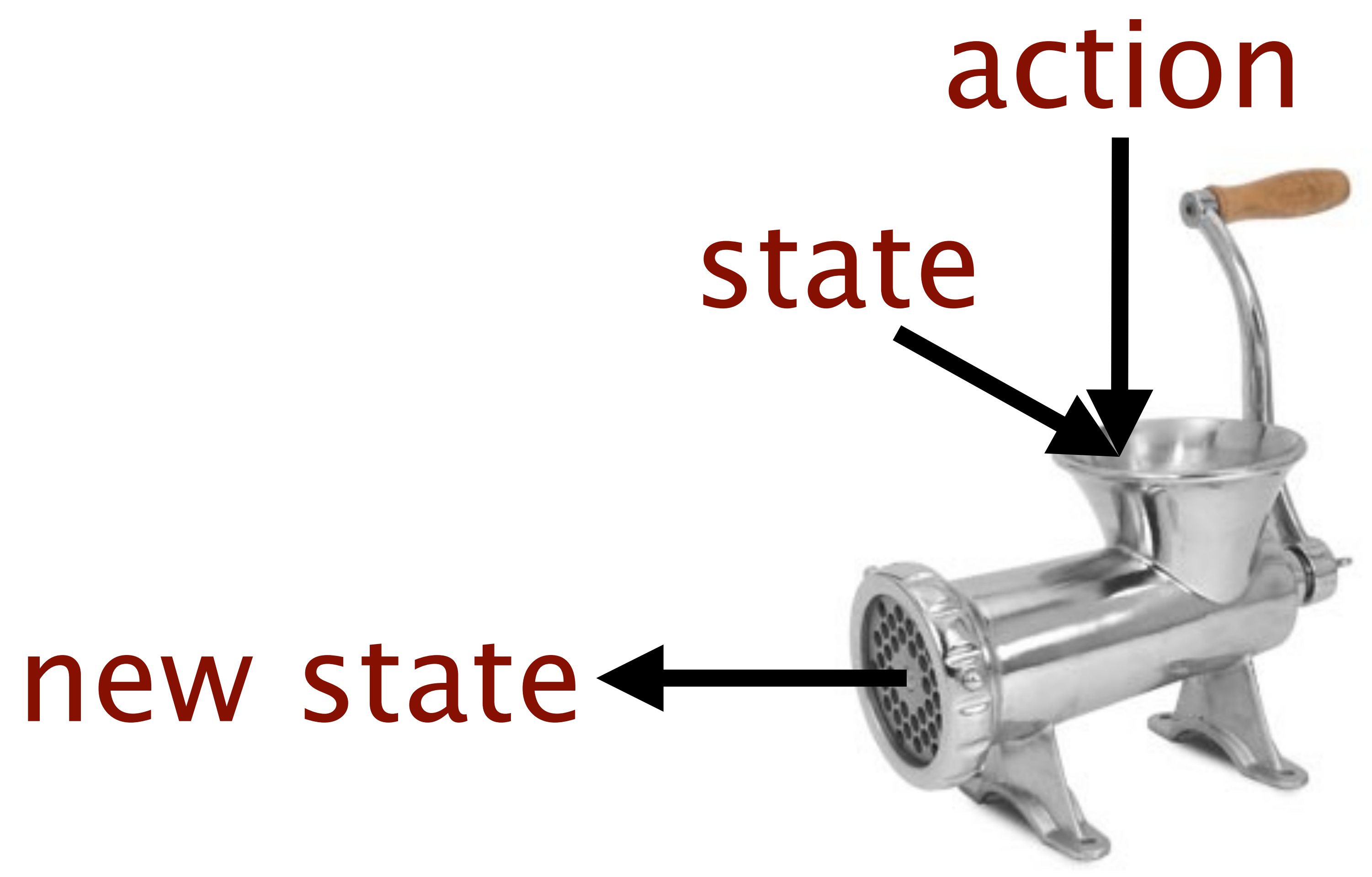
  ```
  if(prevState !== presentState)....
  ```

- An awesome sauce

  Redux dev tools – amazing debugging experience

Actions, stores and reducers

Reducer



Actions, stores and reducers

Reducer

Reducer is nothing but a pure function

```
function (state, action){
    //return new state object
}
```

Actions, stores and reducers

Reducer

```
function (state, action){
   switch(action.type){
      case 'USER_CREATE':
      return Object.assign({},
      state,
      {username:action.username}
      )
   }
}
```

Actions, stores and reducers

Reducer

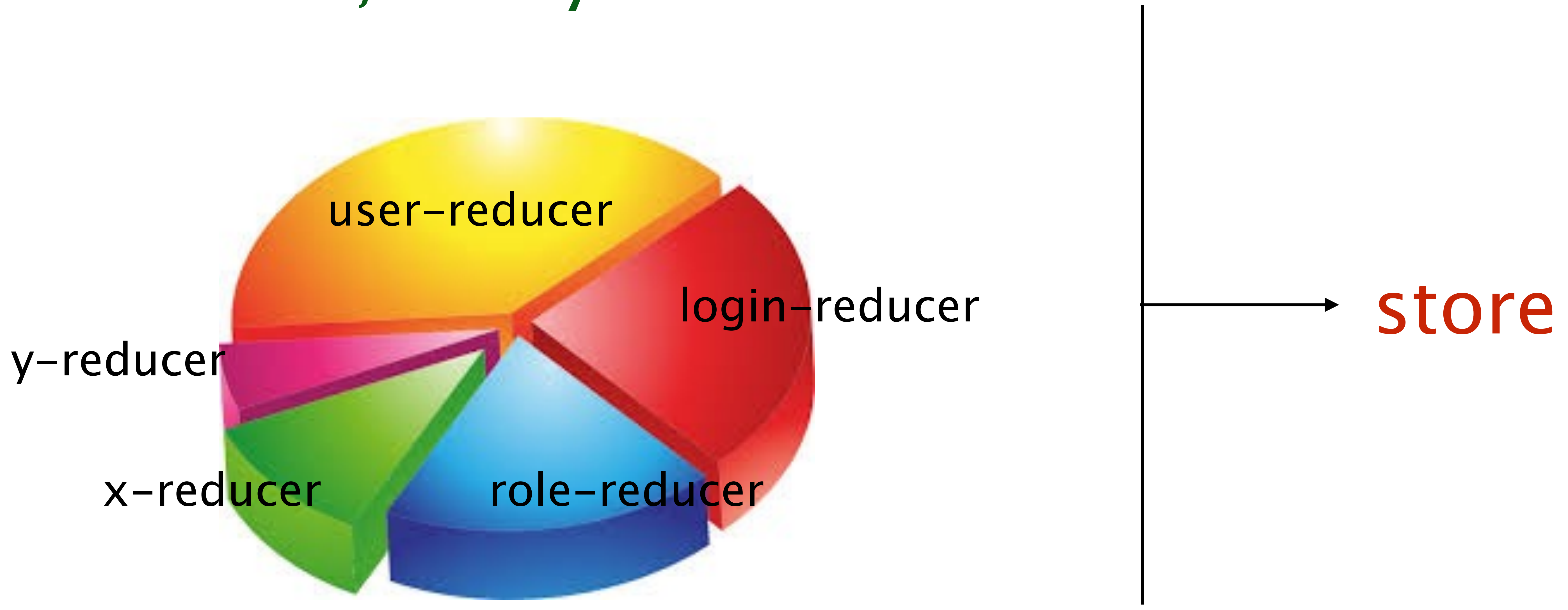
You should not do in a reducer –

- Mutate state
- Any API or DB calls – that causes side effects
- Call any non-pure function

Actions, stores and reducers

Reducer

1 store, many reducers



Summary

- We need redux for handling communication among unrelated components
- Three principles of redux
- Single store
- Action can only create a new state
- Immutable!!
- Reducers – a pure function!