State Management

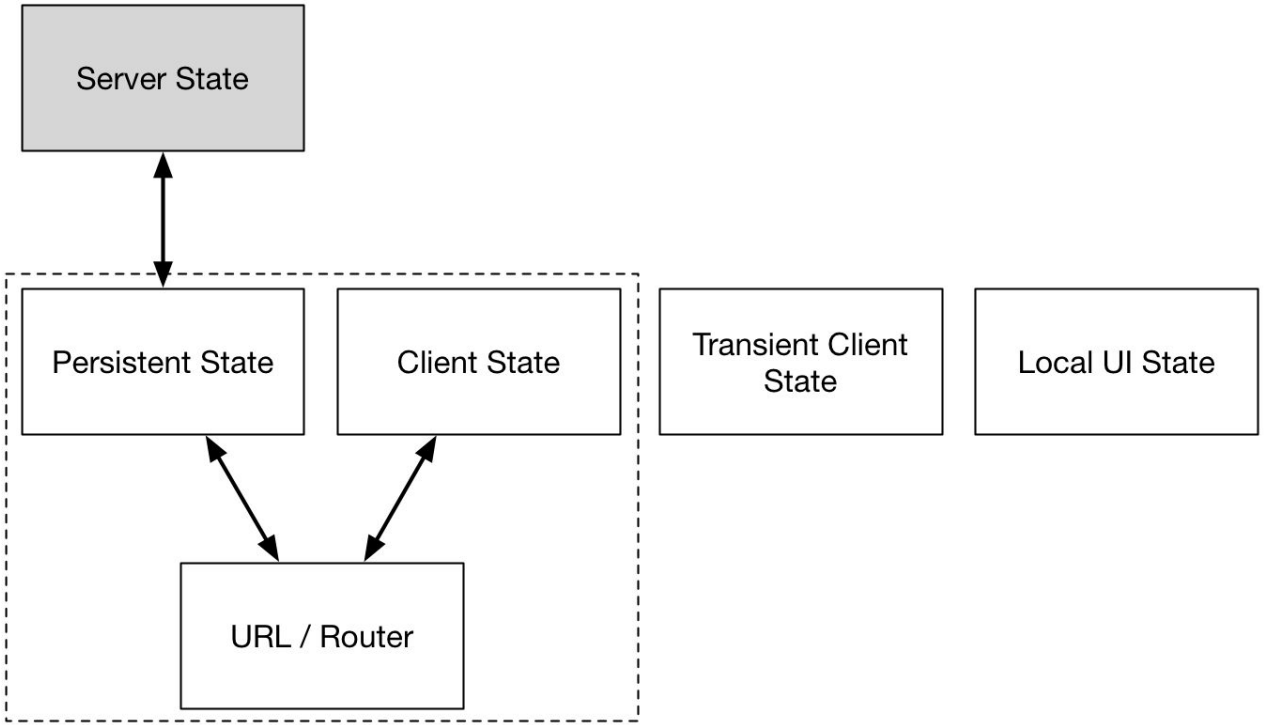
In Angular

- Apurva Zawar

Why State Management?

- ❏ Too many HTTP request calls
- ❏ Re-rendering even for a minor change
- ❏ Synchronization between Client and Server

Types of state



State Synchronization

The persistent state and the server state store the same information.

So do the client state and the URL. Because of this we have to synchronize them.

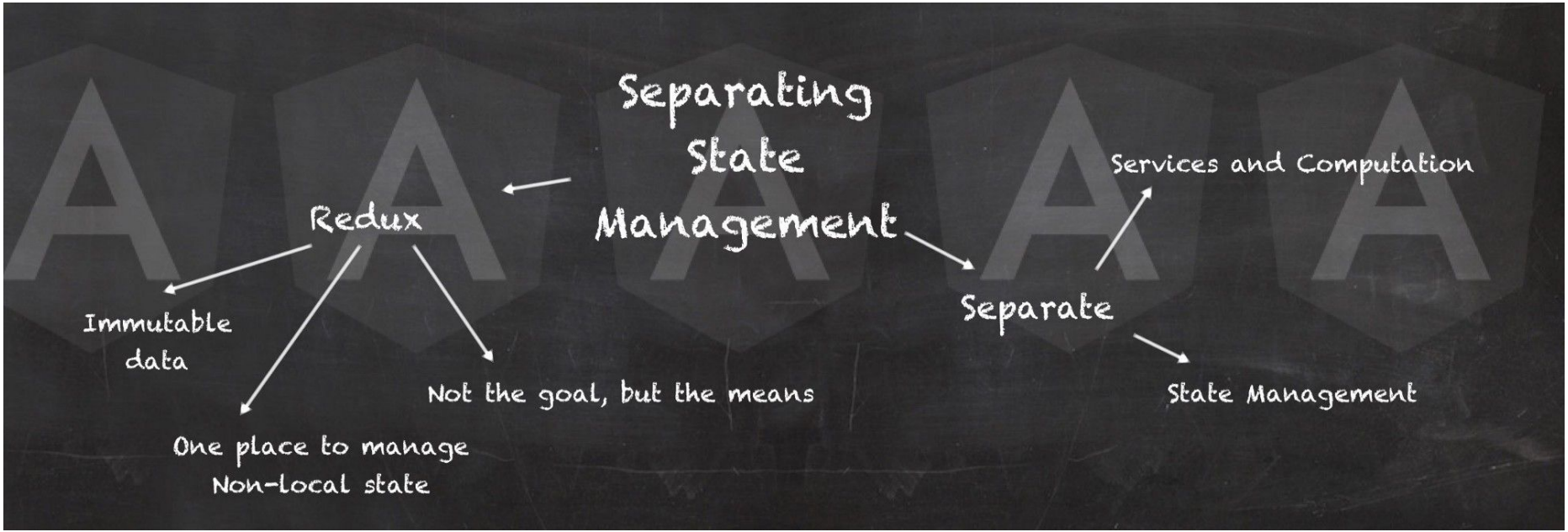
And the choice of the synchronization strategy is one of the most important decisions we make when designing the state management of our applications.

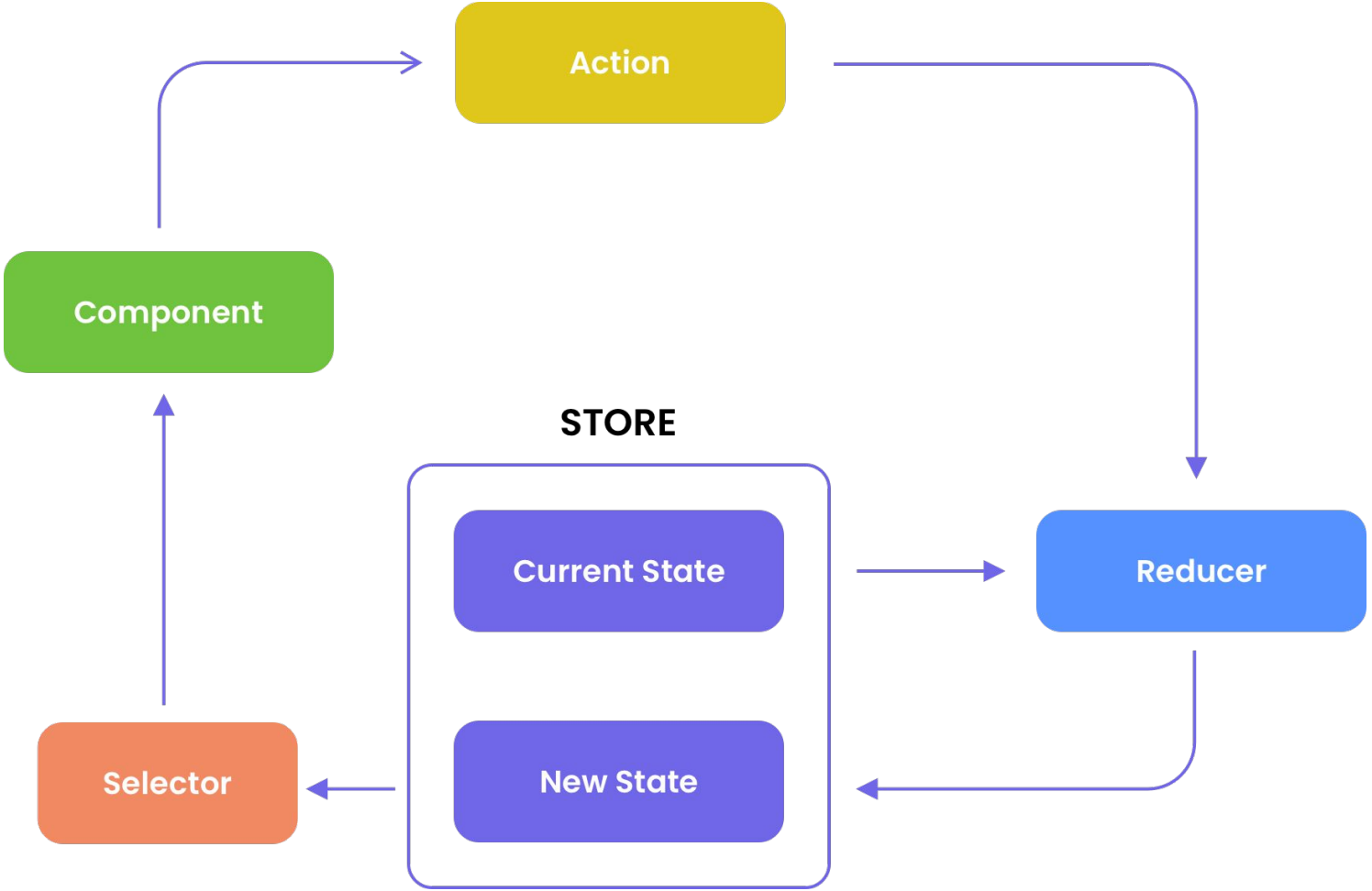
- ❏ Syncing Server state and Persistent state
- ❏ Syncing Client state and URL state

State Management in Angular

- ❏ Component’s interaction via bindings
- ❏ Observable Data Services — Angular services and RxJS
- ❏ Redux Pattern with RxJS
- ❏ Using NgRX or NGXS - Angular libraries built on Redux

A generic way - Redux




Action
Component
STORE
Current State
Reducer
Selector
New State

Action

```
const ADD_TODO = 'ADD_TODO'

{

  type: ADD_TODO,

  text: 'Build my first Redux app'

}
```

Action Creators

```
function addTodoWithDispatch(text) {
  const action = {
    type: ADD_TODO,
    text
  }
  dispatch(action)
}
```

Reducer

```
function todoApp(state = initialState, action) {
  switch (action.type) {
    case ADD_TODO:
      return Object.assign({}, state, {
        todos:
          {
            text: action.text,
            completed: false
          }
      })
    default:
      return state
  }
}
```

Store

```
const store = createStore(todoApp)
```

Thank you