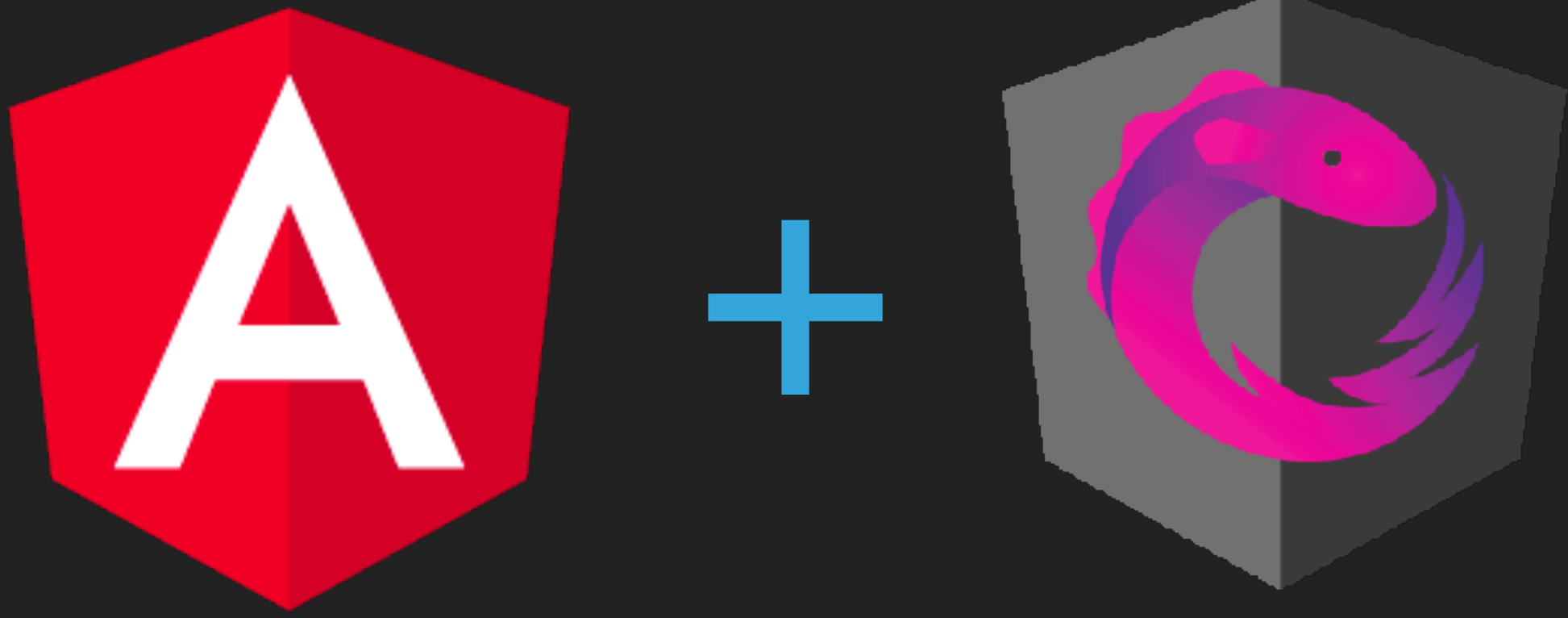


IMPLEMENTING THE REDUX PATTERN TO MAKE SCALABLE APPS

ANGULAR + NGRX

REDUX IS NOT PART OF REACT

Highly adopted in the react community

Authored by a dev who works at facebook

Is framework agnostic

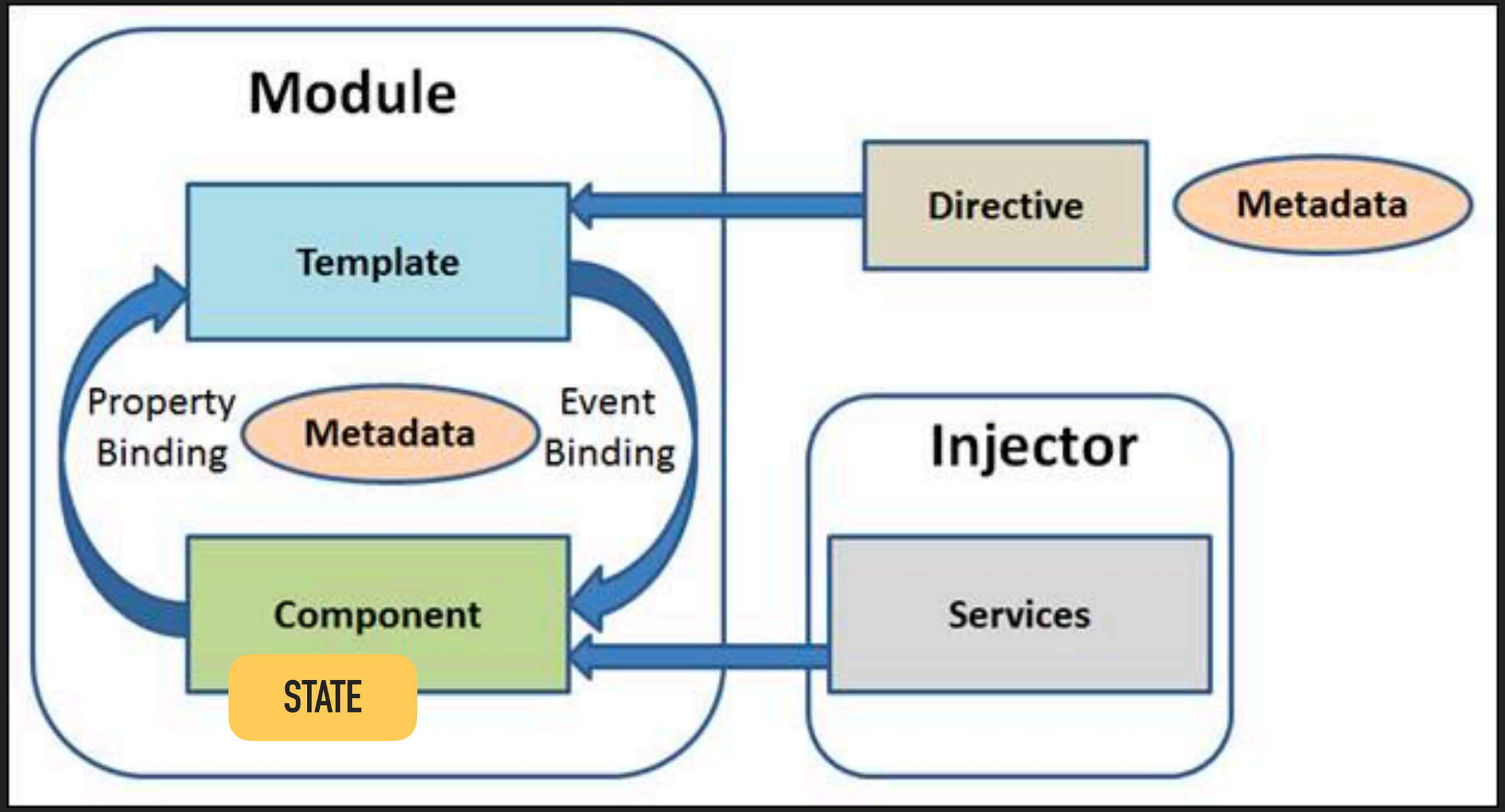
USE THE RIGHT TOOL FOR THE JOB

REDUX MIGHT BE THE RIGHT TOOL IF…

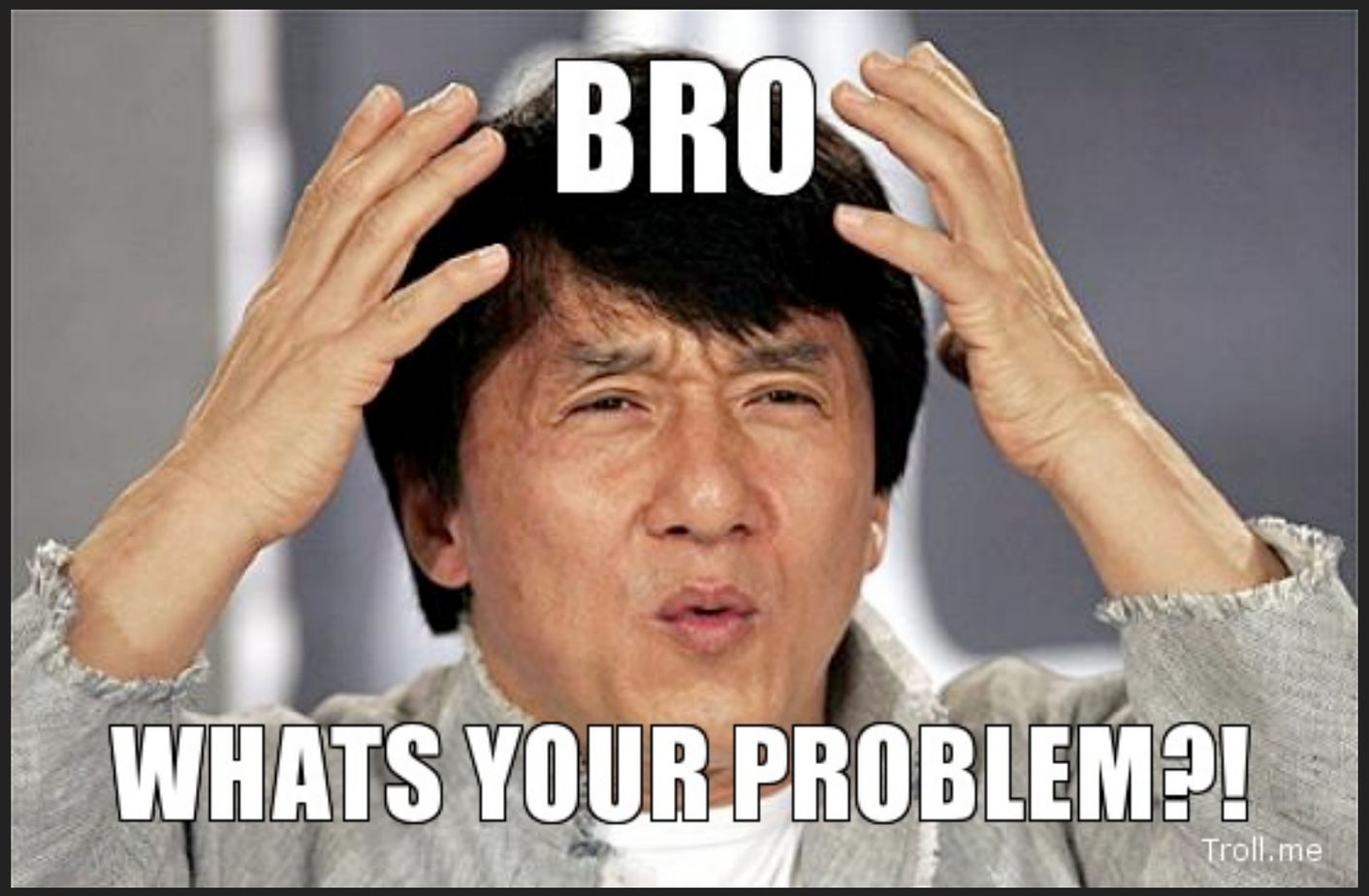
- User workflows are complex
- Your app has a large variety of user workflows (consider both regular users and administrators)
- Users can collaborate
- You're using web sockets or Server Sent Events
- You're loading data from multiple endpoints to build a single view

Otherwise…

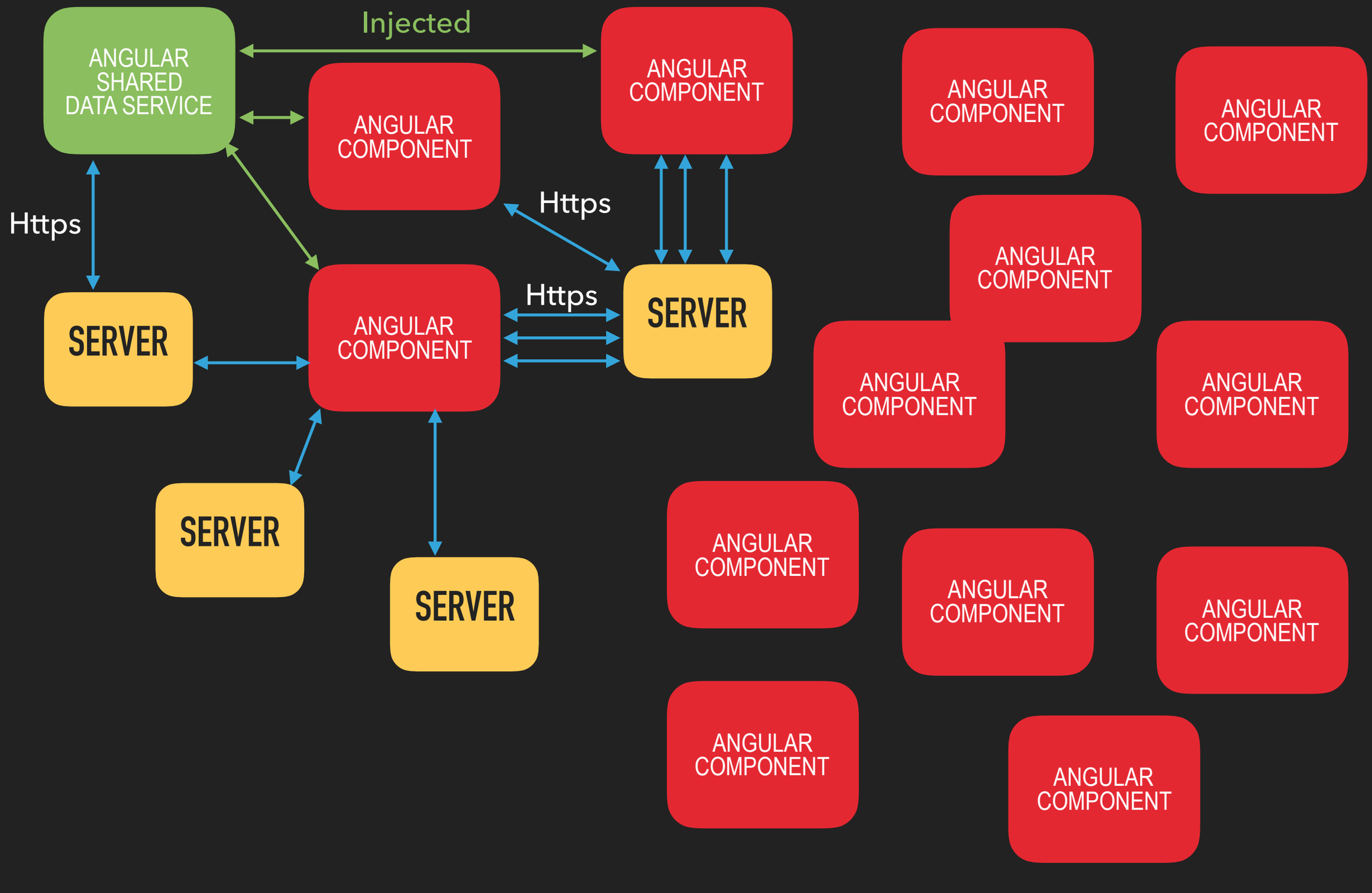
BASIC ANGULAR ARCHITECTURE



THE PROBLEM



THE PROBLEM


ANGULAR SHARED DATA SERVICE
Injected
ANGULAR COMPONENT
ANGULAR COMPONENT
ANGULAR COMPONENT
ANGULAR COMPONENT
ANGULAR COMPONENT
Https
Https
Https
SERVER
SERVER
ANGULAR COMPONENT
SERVER
SERVER
ANGULAR COMPONENT
ANGULAR COMPONENT
ANGULAR COMPONENT
ANGULAR COMPONENT
ANGULAR COMPONENT
ANGULAR COMPONENT
ANGULAR COMPONENT

THE SOLUTION:

DATA FLOW

@ngrx/STORE

Action Stream

REDUCER

(state, action) =>
state

STATE

{ ...appState }

@ngrx/
Effects

Https

SERVER

ACTION

{ type: addToDo
payload: ski }

Dispatch
Action

Dispatch Action

ANGULAR
COMPONENT

Components
Subscribes to
State

BENEFITS OF THE REDUX PATTERN

THESE ARCHITECTURAL DECISIONS:

- Single Source of Truth: The Store - centralized client side state
- One Way Data Flow
- State is only updated through pure functions (reducers)

PROVIDE THESE BENEFITS:

- Eliminates race conditions that mess with view rendering
- Deterministic View Renders
- Deterministic State Reproduction
- State updates are transactional
- Testing is easier
- More performant - Angular change detection OnPush setting

Show Demo App

https://ngrx.github.io/platform/example-app/#/login

DISPATCHING AN ACTION

```
<bc-toolbar (openMenu)="openSidenav()">
  Book Collection
</bc-toolbar>
```

```
openSidenav() {
  this.store.dispatch(new layout.OpenSidenav());
}
```

THE ACTION

```ts
import { Action } from '@ngrx/store';

export const OPEN_SIDENAV = '[Layout] Open Sidenav';
export const CLOSE_SIDENAV = '[Layout] Close Sidenav';

export class OpenSidenav implements Action {
  readonly type = OPEN_SIDENAV;
}

export class CloseSidenav implements Action {
  readonly type = CLOSE_SIDENAV;
}

export type Actions = OpenSidenav | CloseSidenav;
```

```
{
  type: '[Layout] Open Sidenav',
}
```

THE SOLUTION:

DATA FLOW

@ngrx/STORE

Action Stream

REDUCER

(state, action) =>
state

STATE

{ ...appState }

@ngrx/
Effects

Https

SERVER

ACTION

{ type: addToDo
payload: ski }

Dispatch
Action

Dispatch Action

ANGULAR
COMPONENT

Components
Subscribes to
State

THE REDUCER

```
import * as layout from '../actions/layout';

export function reducer(state = initialState, action: layout.Actions): State {
  switch (action.type) {
    case layout.CLOSE_SIDENAV:
      return {
        showSidenav: false,
      };

    case layout.OPEN_SIDENAV:
      return {
        showSidenav: true,
      };

    default:
      return state;
  }
}
```

THE SOLUTION:

DATA FLOW

@ngrx/STORE

Action Stream

REDUCER

(state, action) =>
state

STATE

{ ...appState }

@ngrx/
Effects

Https

SERVER

ACTION

{ type: addToDo
payload: ski }

Dispatch
Action

Dispatch Action

ANGULAR
COMPONENT

Components
Subscribes to
State

STATE

```
import * as layout from '../actions/layout';
```

```
export interface State {
  showSidenav: boolean;
}

const initialState: State = {
  showSidenav: false,
};
```

THE SOLUTION:

DATA FLOW

@ngrx/STORE

Action Stream

REDUCER

(state, action) =>
state

STATE

{ ...appState }

@ngrx/
Effects

Https

SERVER

ACTION

{ type: addToDo
payload: ski }

Dispatch
Action

Dispatch Action

ANGULAR
COMPONENT

Components
Subscribes to
State

SELECTOR & COMPONENT

```ts
export const getShowSidenav = (state: State) => state.showSidenav;
```

```ts
export class AppComponent {
  showSidenav$: Observable<boolean>;
  loggedIn$: Observable<boolean>;

  constructor(private store: Store<fromRoot.State>) {
    this.showSidenav$ = this.store.select(fromRoot.getShowSidenav);
    this.loggedIn$ = this.store.select(fromAuth.getLoggedIn);
  }
```

TEMPLATE

```
<bc-sidenav [open]="showSidenav$ | async">
  <bc-nav-item (navigate)="closeSidenav()" *ngIf="loggedIn$ | async" routerLink="/"
    My Collection
  </bc-nav-item>
  <bc-nav-item (navigate)="closeSidenav()" *ngIf="loggedIn$ | async" routerLink="/b
    Browse Books
  </bc-nav-item>
  <bc-nav-item (navigate)="closeSidenav()" *ngIf="!(loggedIn$ | async)">
    Sign In
  </bc-nav-item>
  <bc-nav-item (navigate)="logout()" *ngIf="loggedIn$ | async">
    Sign Out
  </bc-nav-item>
</bc-sidenav>
```

THE SOLUTION:

DATA FLOW

@ngrx/STORE

Action Stream

REDUCER

(state, action) =>
state

STATE

{ ...appState }

@ngrx/
Effects

Https

SERVER

ACTION

{ type: addToDo
payload: ski }

Dispatch
Action

Dispatch Action

ANGULAR
COMPONENT

Components
Subscribes to
State

EFFECT

```
@Injectable()
export class BookEffects {
  @Effect()
  search$: Observable<Action> = this.actions$
    .ofType<book.Search>(book.SEARCH)
    .debounceTime(this.debounce, this.scheduler || async)
    .map(action => action.payload)
    .switchMap(query => {
      if (query === '') {
        return empty();
      }

      const nextSearch$ = this.actions$.ofType(book.SEARCH).skip(1);

      return this.googleBooks
        .searchBooks(query)
        .takeUntil(nextSearch$)
        .map((books: Book[]) => new book.SearchComplete(books))
        .catch(() => of(new book.SearchComplete([])));
    });
```

THE PROBLEM

Asynchronous Code

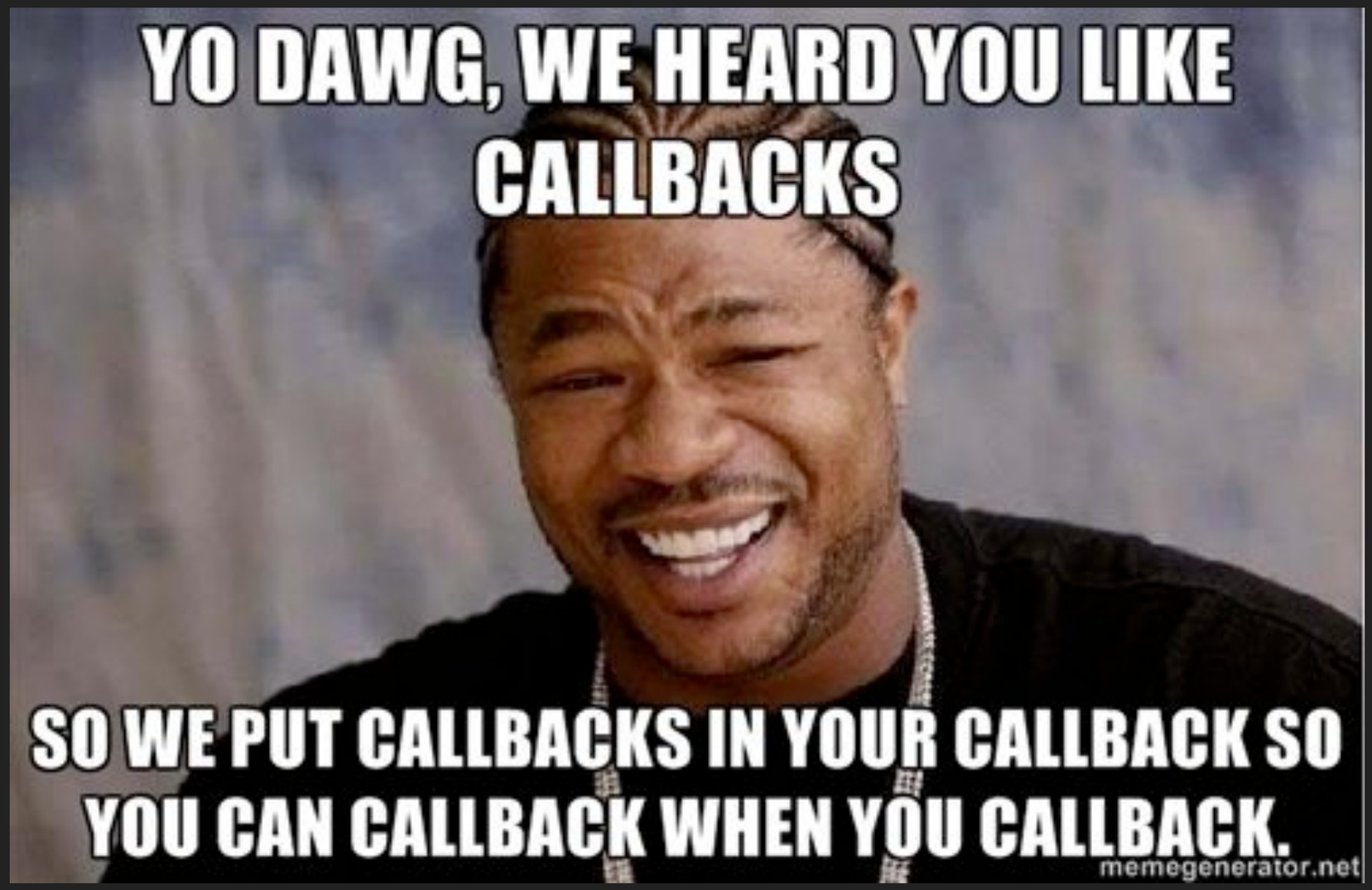
Solution? Callback Functions

```
a({
    parameter : someParameter,
    callback : function(status) {
        if (status == states.SUCCESS) {
          b(function(status) {
              if (status == states.SUCCESS) {
                 c(function(status){
                     if (status == states.SUCCESS) {
                         // Not an exaggeration. I have seen
                         // code that looks like this regularly.
                     }
                 });
              }
          });
        } elseif (status == states.PENDING {
          ...
        }
    }
});
```

THE PROBLEM

Asynchronous Code

Solution? Callback Functions



EFFECT

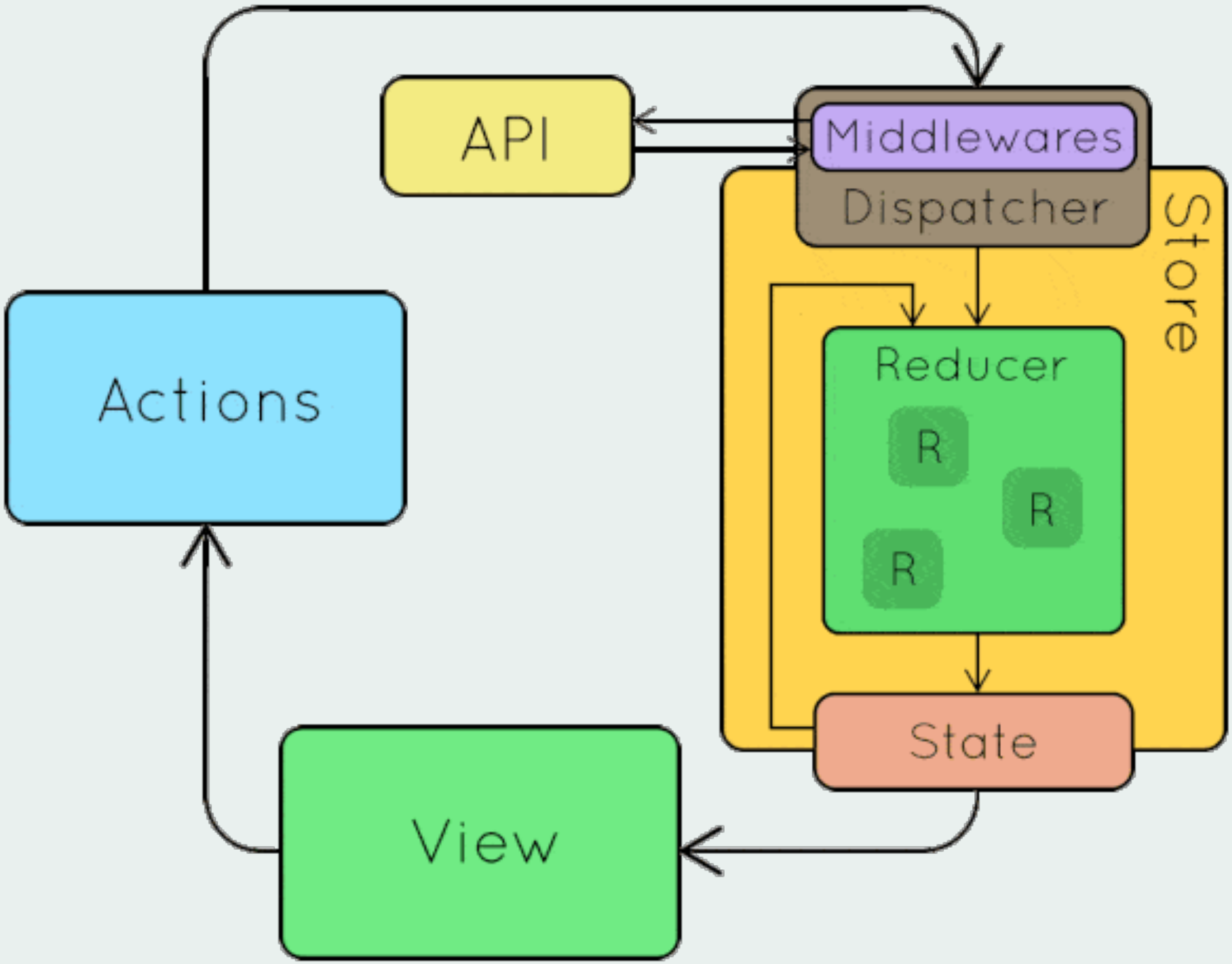
```
@Injectable()
export class DisclosuresEffects {

  @Effect() disclosuresContinue$ = this.actions$
    .ofType(LAST_PAGE_BEFORE_SOMETHING_AWESOME_CONTINUE)
    .map(toPayload)
    .exhaustMap(payload => concat(
        this.updateSecurityCert(toApiSecurityCertification(payload)),
        this.updateAccounts(),
        this.updateBillPay(),
        this.updateAccountConfigs(),
        this.orderTeddyBear(),
        this.updateDebitCardApps(),
        this.applyCoupon(),
        this.updateUserRelationships(),
        this.createWelcomeEmail(payload.emailDisclosures),
        this.sendWelcomeEmail(),
        [new CompleteAction()]
    )
    .catch(error => [new DisclosuresContinueErrorAction(error)])
  );
```

RXJS: OBSERVABLE STREAMS

http://rxmarbles.com

THE REDUX PATTERN: ONE WAY DATA FLOW



BENEFITS OF REDUX

THESE ARCHITECTURAL DECISIONS:

- Single Source of Truth: The Store
- One Way Data Flow
- State is only updated through pure functions (reducers)

PROVIDE THESE BENEFITS:

- Eliminates race conditions that mess with view rendering
- Deterministic View Renders
- Deterministic State Reproduction
- State updates are transactional
- Testing is easier
- More performant - Angular change detection OnPush setting

MORE PERFORMANT CHANGE DETECTION: "ONPUSH"

```
@Component({
  // ...
  changeDetection: ChangeDetectionStrategy.OnPush
})
```

```
let squirl = {
    tail: 'bushy',
    color: 'red-brown'
    age: 2,
    energyLevel: 'high',
    favoriteFood: 'pizza'
};
```

!! Mutating the state, same reference

```
squirl.age = 12;
squirl.energyLevel = 'low';
```

Replacing state, new reference

```
squirl = {
    tail: 'bushy',
    color: 'red-brown'
    age: 12,
    energyLevel: 'low',
    favoriteFood: 'pizza'
};
```

DETERMINISTIC STATE/VIEWS WITH ACTIONS, TRANSACTIONAL LOG

- Show redux dev tools, in demo app

NX: ANGULAR + NGRX CLI

- Show page
- https://blog.nrwl.io/nrwl-nx-an-open-source-toolkit-for-enterprise-angular-applications-38698e94d65

Additional resources:

https://github.com/apasternack/Presentations/tree/master/Ngrx%20-%20Angular

THE END

STAY IN TOUCH

| Adam Pasternack

| Twitter: @AJPasternack

| adam.pasternack@gmail.com

HTTPS://WWW.SLIDESHARE.NET/JAYPHELPS/RXJS-REDUX-REACT-AMAZING. GOOD DESCRIPTION OF ACTIONS REDUCERS