Angular + Ngrx

how the Redux pattern helps you build scalable, maintainable apps

2018 May 4
Columbus OH

Adam Pasternack (@ajpasternack)
<adam.pasternack@gmail.com>

STIR TREK

2018 May 4
Columbus OH

Angular + Ngrx

how the Redux pattern helps you build scalable, maintainable apps

2018 May 4
Columbus OH

Adam Pasternack (@ajpasternack)
<adam.pasternack@gmail.com>

Adam Pasternack

@ajpasternack
<adam.pasternack@gmail.com>

2018 May 4
Columbus OH

Battelle
The Business of Innovation

cardinal

L3 Technologies

Wendy's

Huntington

Microsoft Professional
Program for DevOps

Georgia Institute of Technology

Adam Pasternack

@ajpasternack
<adam.pasternack@gmail.com>

2018 May 4
Columbus OH

90 Degree Labs

Adam Pasternack

@ajpasternack
<adam.pasternack@gmail.com>

STIR TREK

2018 May 4
Columbus OH

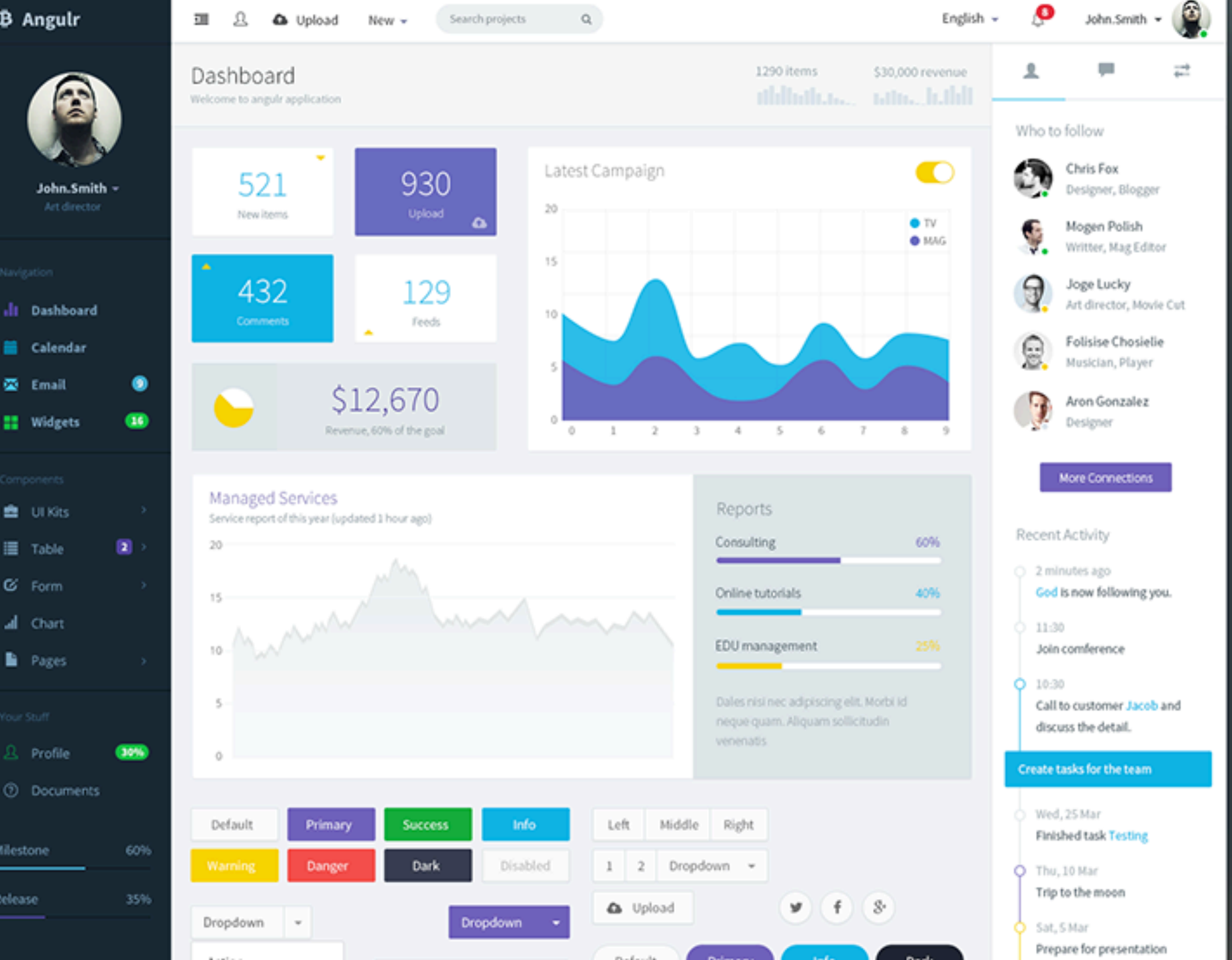
CONDG

Central Ohio .Net Developer Group
4th Thursdays
ICC - Westerville
I-270 and Cleveland Ave

Angular Columbus

(Resurrecting)
With Craig McKeachie
Setting future schedule of talks
and hands on workshops
based on survey feedback

More info on www.meetup.com


Angulr
Upload
New
Search projects
English
John.Smith
Dashboard
Welcome to angulr application
1290 items
$30,000 revenue
John.Smith
Art director
Navigation
Dashboard
Calendar
Email
Widgets
Components
UI Kits
Table
Form
Chart
Pages
Your Stuff
Profile
Documents
Milestone
Release
521
New items
930
Upload
432
Comments
129
Feeds
$12,670
Revenue, 60% of the goal
Latest Campaign
TV
MAG
Managed Services
Service report of this year (updated 1 hour ago)
Reports
Consulting
Online tutorials
EDU management
Default
Primary
Success
Info
Warning
Danger
Dark
Disabled
Left
Middle
Right
Dropdown
Upload
Who to follow
Chris Fox
Designer, Blogger
Mogen Polish
Writter, Mag Editor
Joge Lucky
Art director, Movie Cut
Folisise Chosielie
Musician, Player
Aron Gonzalez
Designer
More Connections
Recent Activity
2 minutes ago
God is now following you.
11:30
Join comference
10:30
Call to customer Jacob and discuss the detail.
Create tasks for the team
Wed, 25 Mar
Finished task Testing
Thu, 10 Mar
Trip to the moon
Sat, 5 Mar
Prepare for presentation


Which One
of You
Should Get
FIRED?

Benefits of using Ngrx

- Eliminate race condition bugs that mess with view rendering
- State updates are transactional
- Deterministic State Reproduction
- Deterministic View Renders
- Testing is simplified
- More performant - Angular change detection OnPush setting

Agenda

1. Benefits of modern Angular
2. Rxjs - the ReactiveX library for Javascript
3. Redux/Ngrx pattern
4. Examine each piece of the Ngrx pattern

improving
It's what we do.

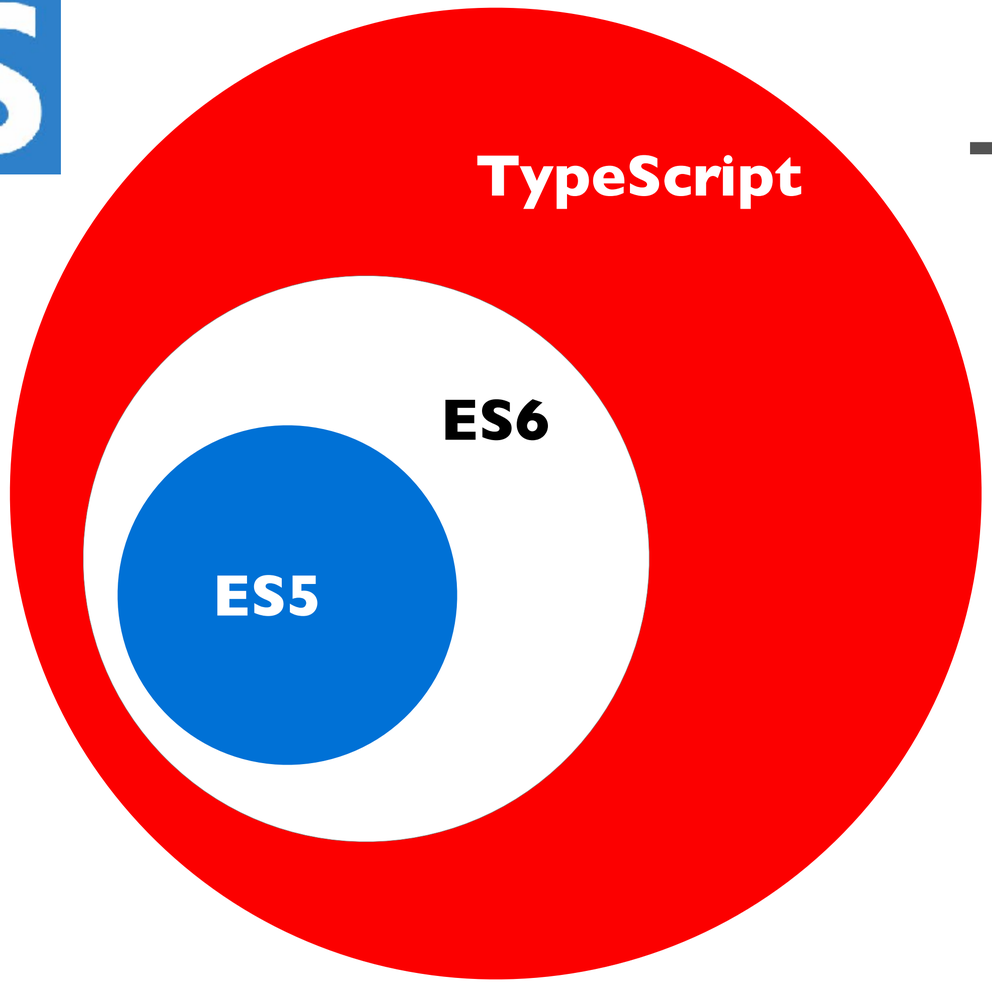
BENEFITS OF MODERN

ANGUAR

(NOT TO BE CONFUSED WITH ITS EVIL OLDER SIBLING ANGULARJS)

improving
It's what we do.

Things that play nice with Angular

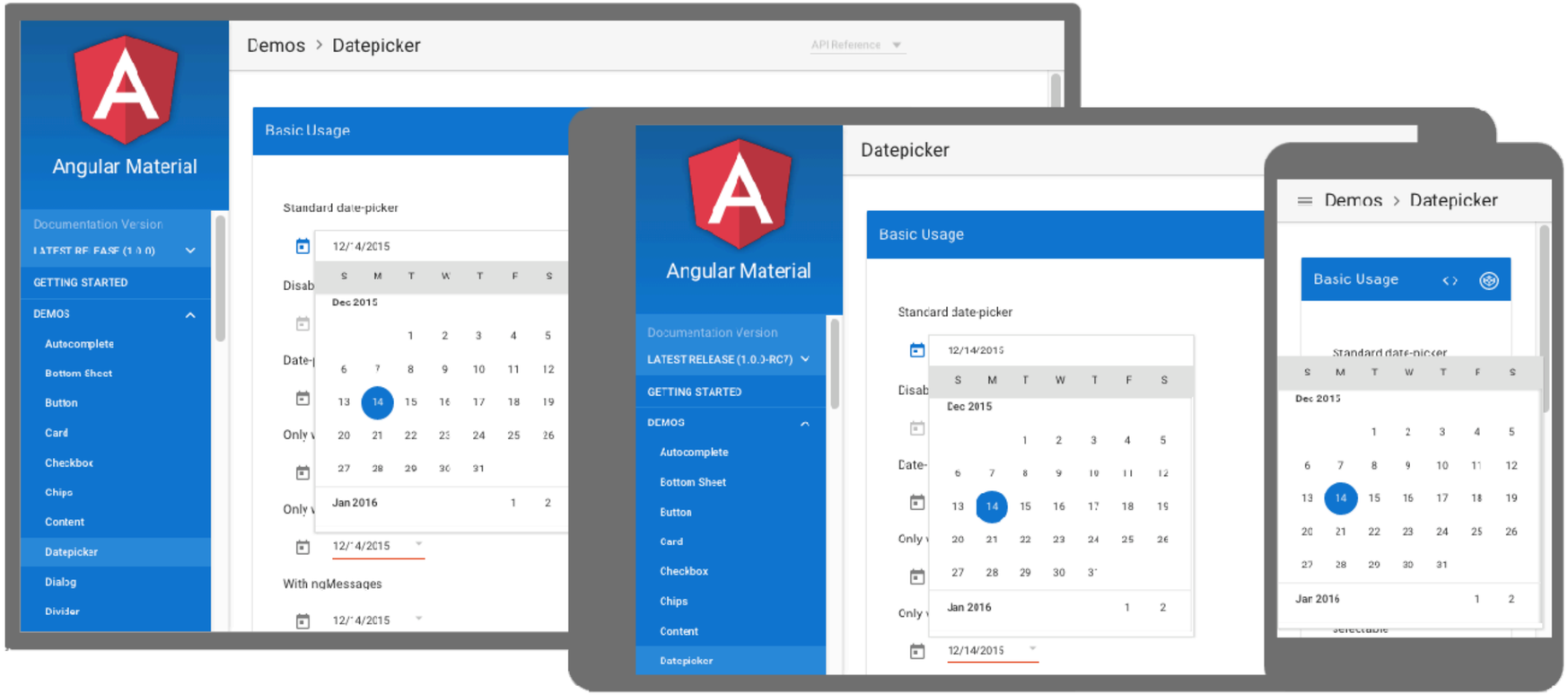


WHAT IS TYPESCRIPT?

improving
It's what we do.

Things that play nice with Angular

Material Design UI Library for Angular



material.angular.io

improving
It's what we do.

Things that play nice with Angular

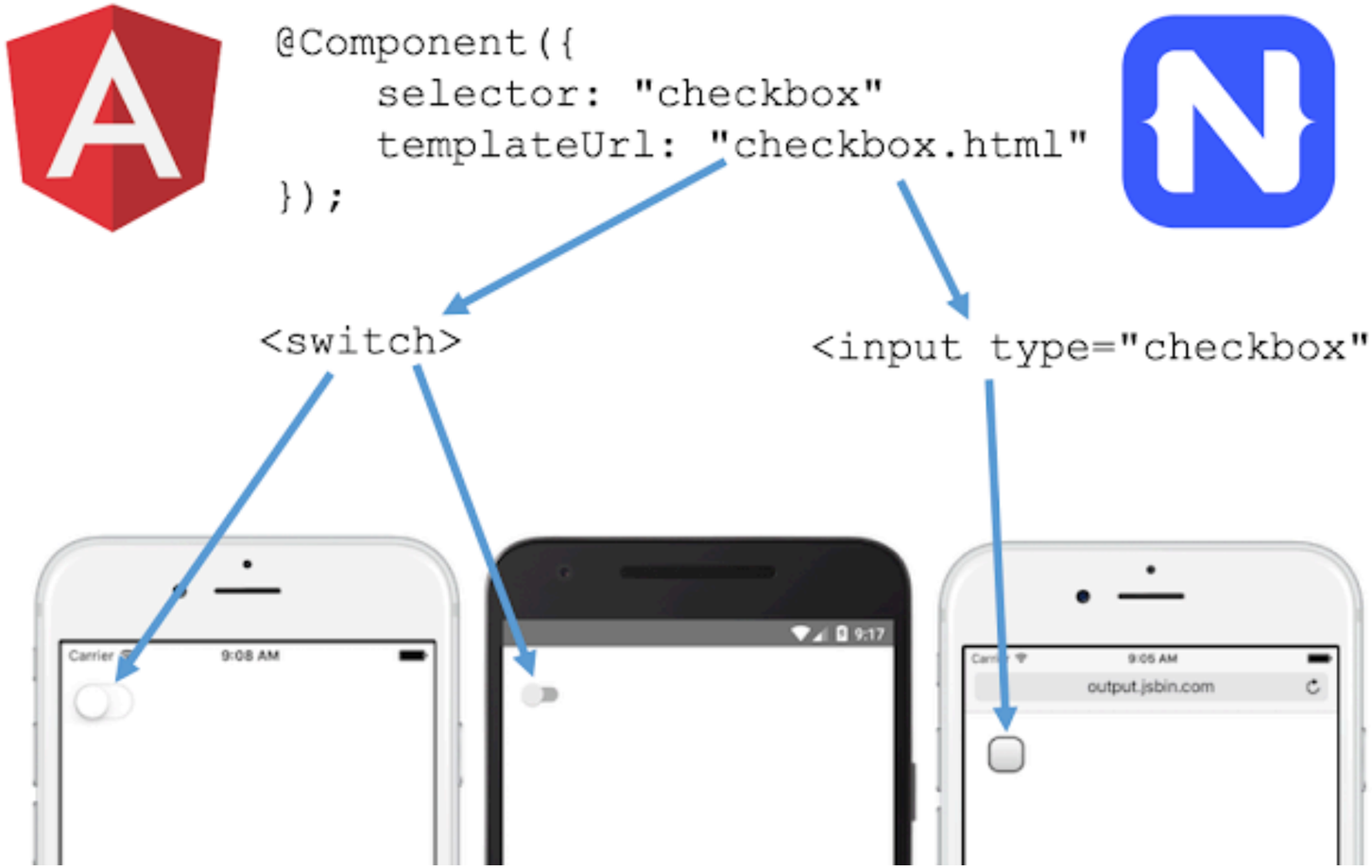
AngularFire

github.com/angular/angularfire2

improving
It's what we do.

Things that play nice with Angular

Angular + NativeScipt = Cross Mobile



improving
It's what we do.

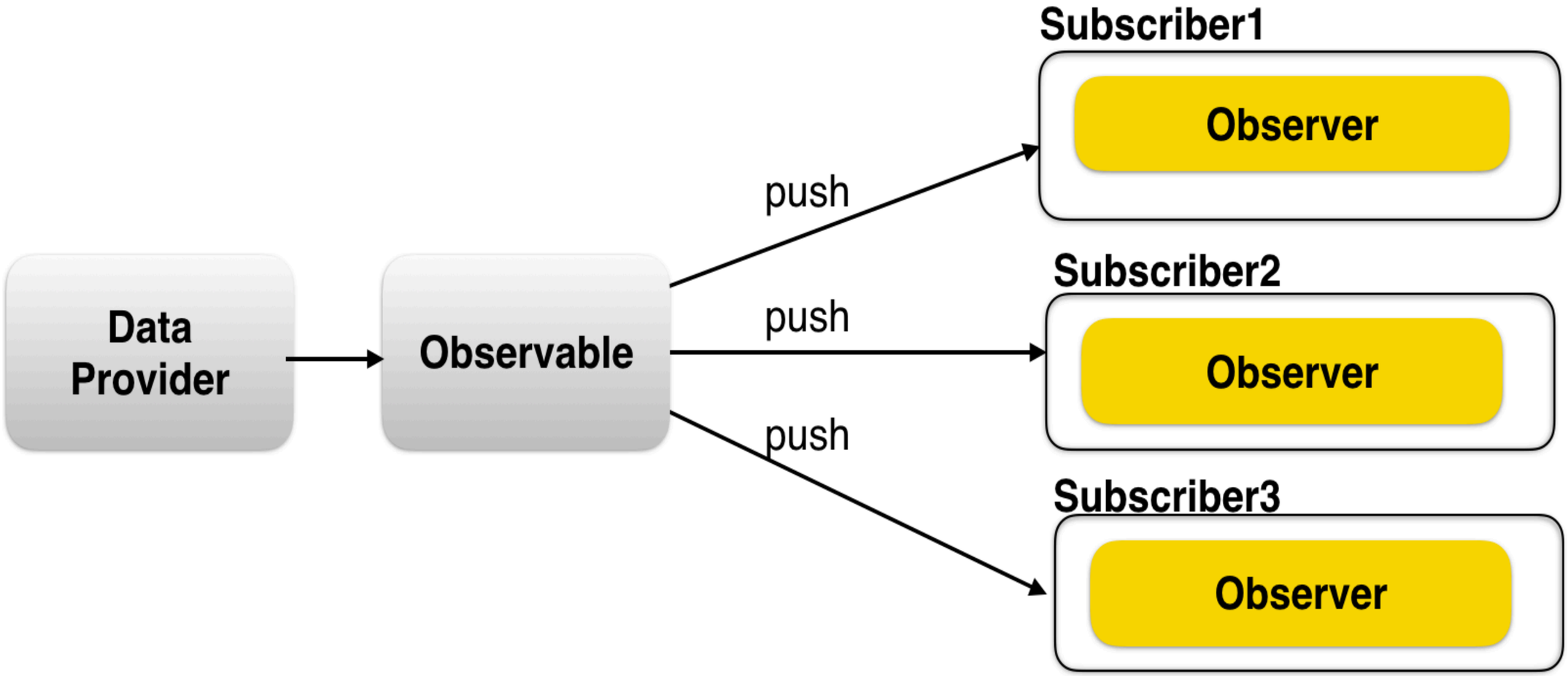
Rxjs - the JS implementation of

ReactiveX

http://reactivex.io/rxjs/

improving
It's what we do.

Observable Streams



UserEvent

ServerCall

SystemEvent

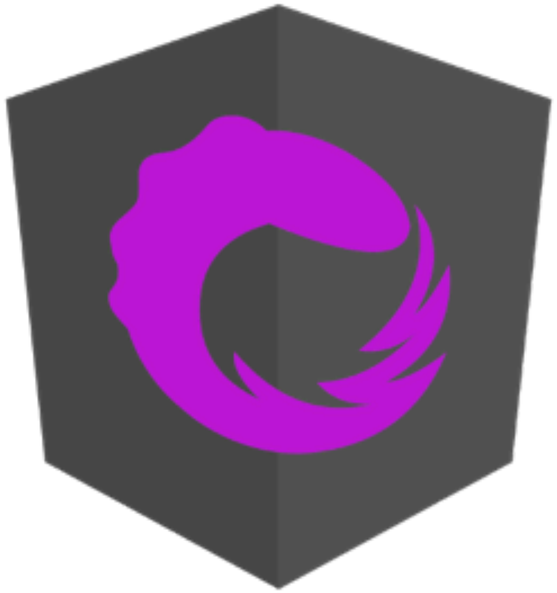
DataModel

View1

View2

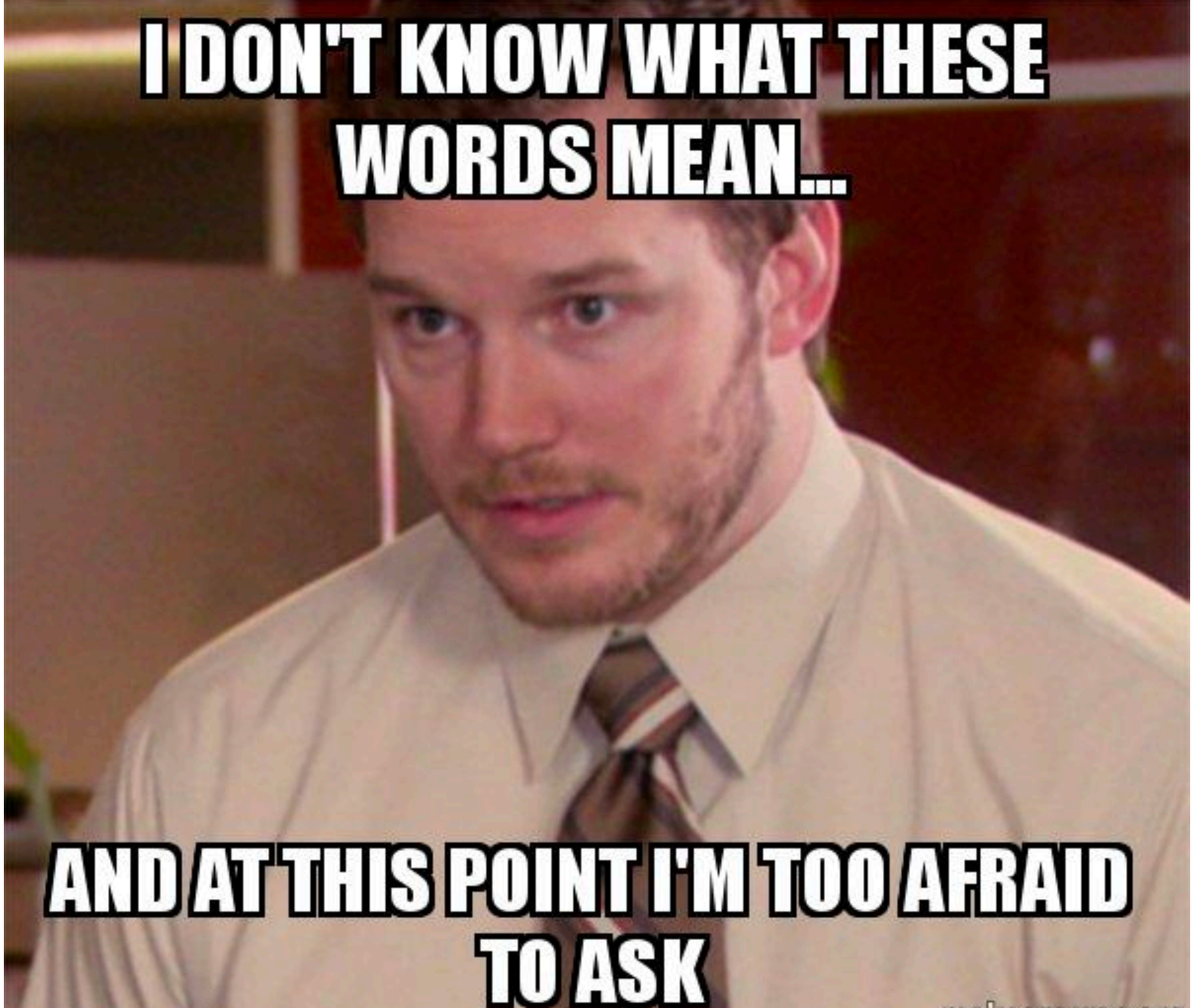
View3

Ngrx



"RxJS powered state management for Angular applications, inspired by Redux"

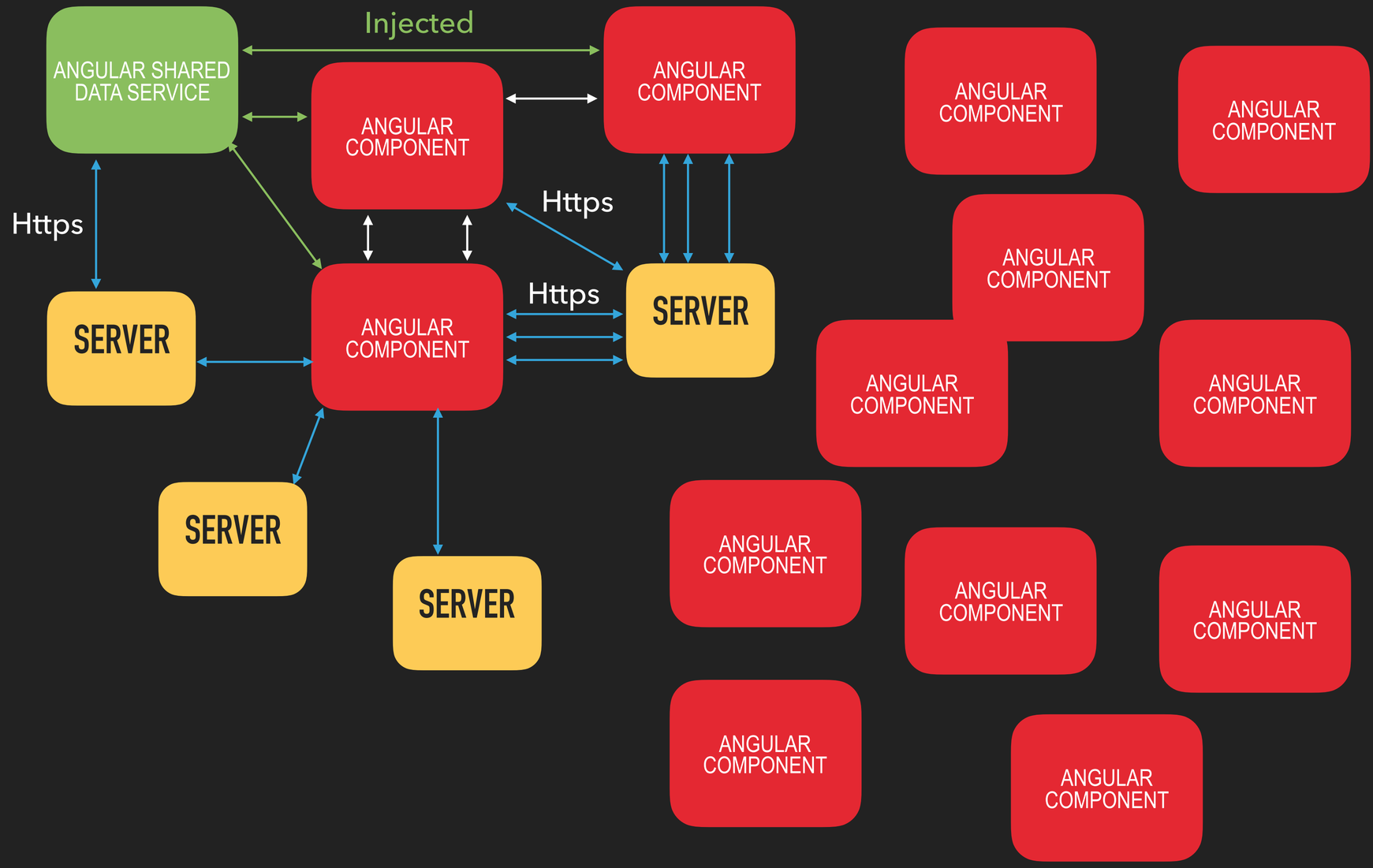
improving
It's what we do.


I DON'T KNOW WHAT THESE WORDS MEAN...
AND AT THIS POINT I'M TOO AFRAID TO ASK

ngrx supercharges the redux pattern with RxJS

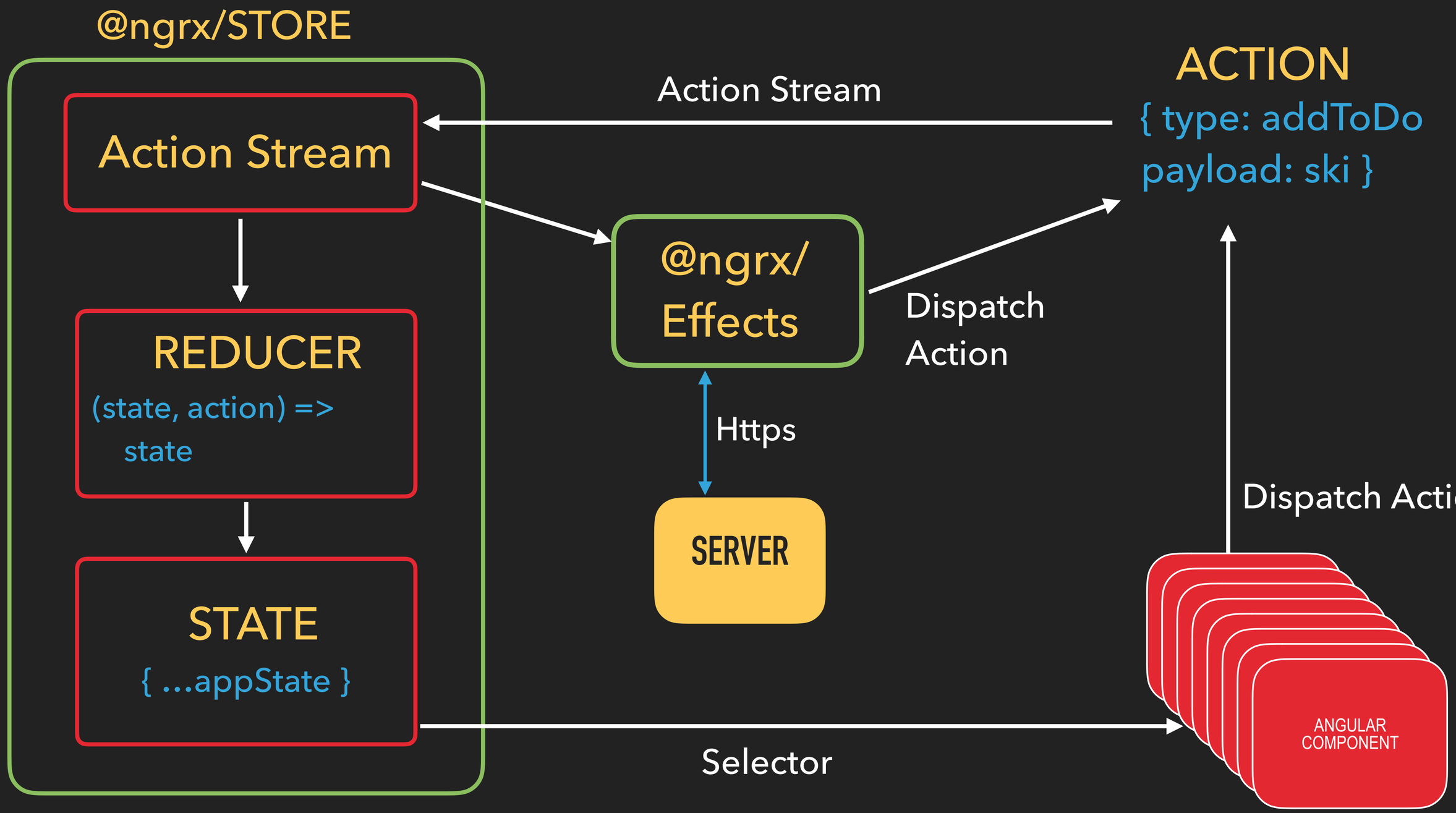
improving
It's what we do.

THE PROBLEM


Injected
ANGULAR SHARED DATA SERVICE
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
ANGULAR COMPONENT
ANGULAR COMPONENT
ANGULAR COMPONENT

THE SOLUTION:

DATA FLOW


@ngrx/STORE
Action Stream
REDUCER
(state, action) =>
state
STATE
{ ...appState }
Action Stream
ACTION
{ type: addToDo
payload: ski }
@ngrx/
Effects
Dispatch
Action
Https
SERVER
Dispatch Action
ANGULAR
COMPONENT
Selector

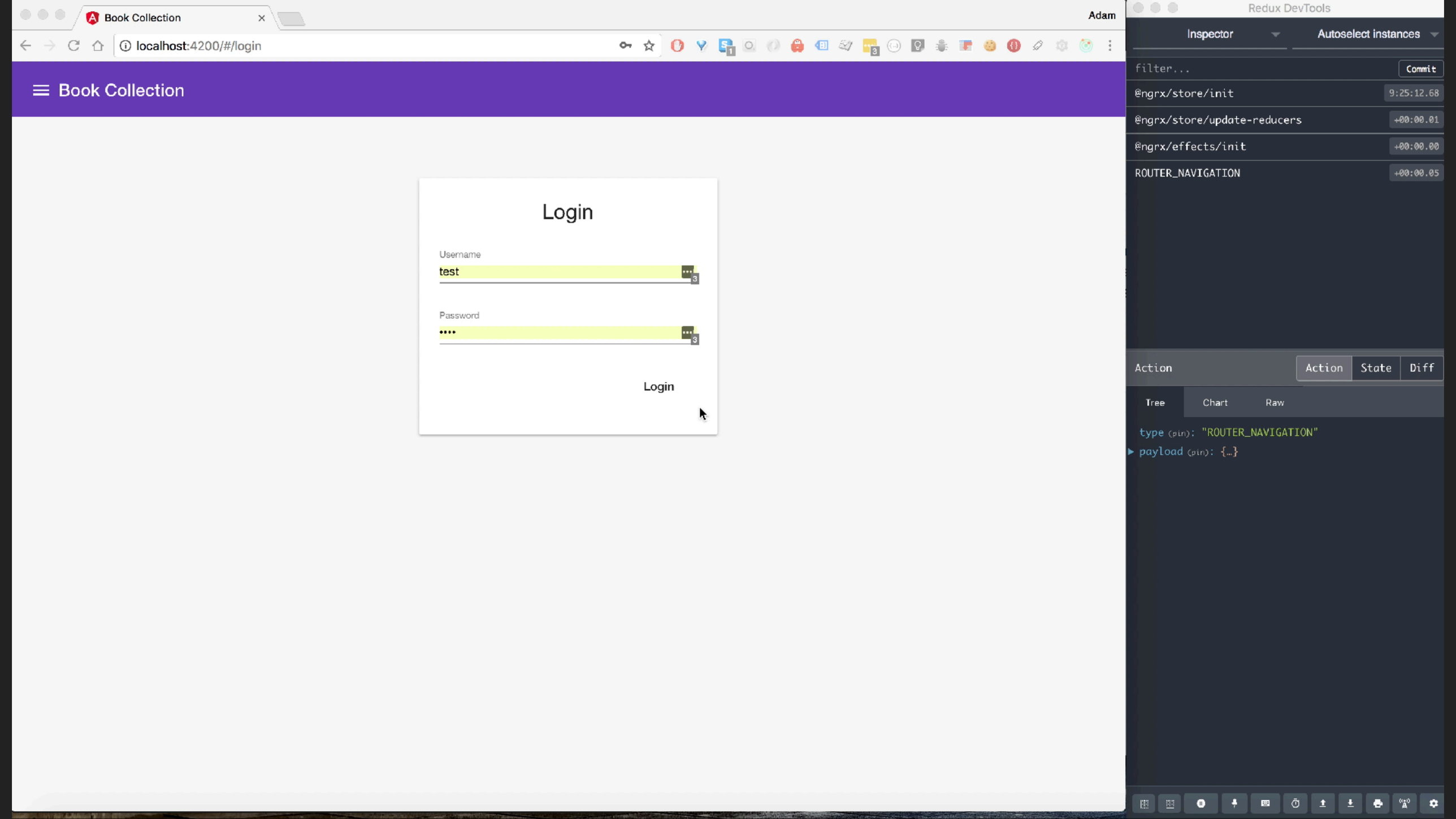
BENEFITS OF THE REDUX PATTERN

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 State Reproduction
- Deterministic View Renders
- State updates are transactional
- Testing is easier
- More performant - Angular change detection OnPush setting


Book Collection
Adam
localhost:4200/#/login
Book Collection
Login
Username
test
Password
Login
Redux DevTools
Inspector
Autoselect instances
filter...
Commit
@ngrx/store/init
9:25:12.68
@ngrx/store/update-reducers
+00:00.01
@ngrx/effects/init
+00:00.00
ROUTER_NAVIGATION
+00:00.05
Action
Action
State
Diff
Tree
Chart
Raw
type (pin): "ROUTER_NAVIGATION"
payload (pin): {…}

The Action

- plain Javascript object
- only required property is ‘type’

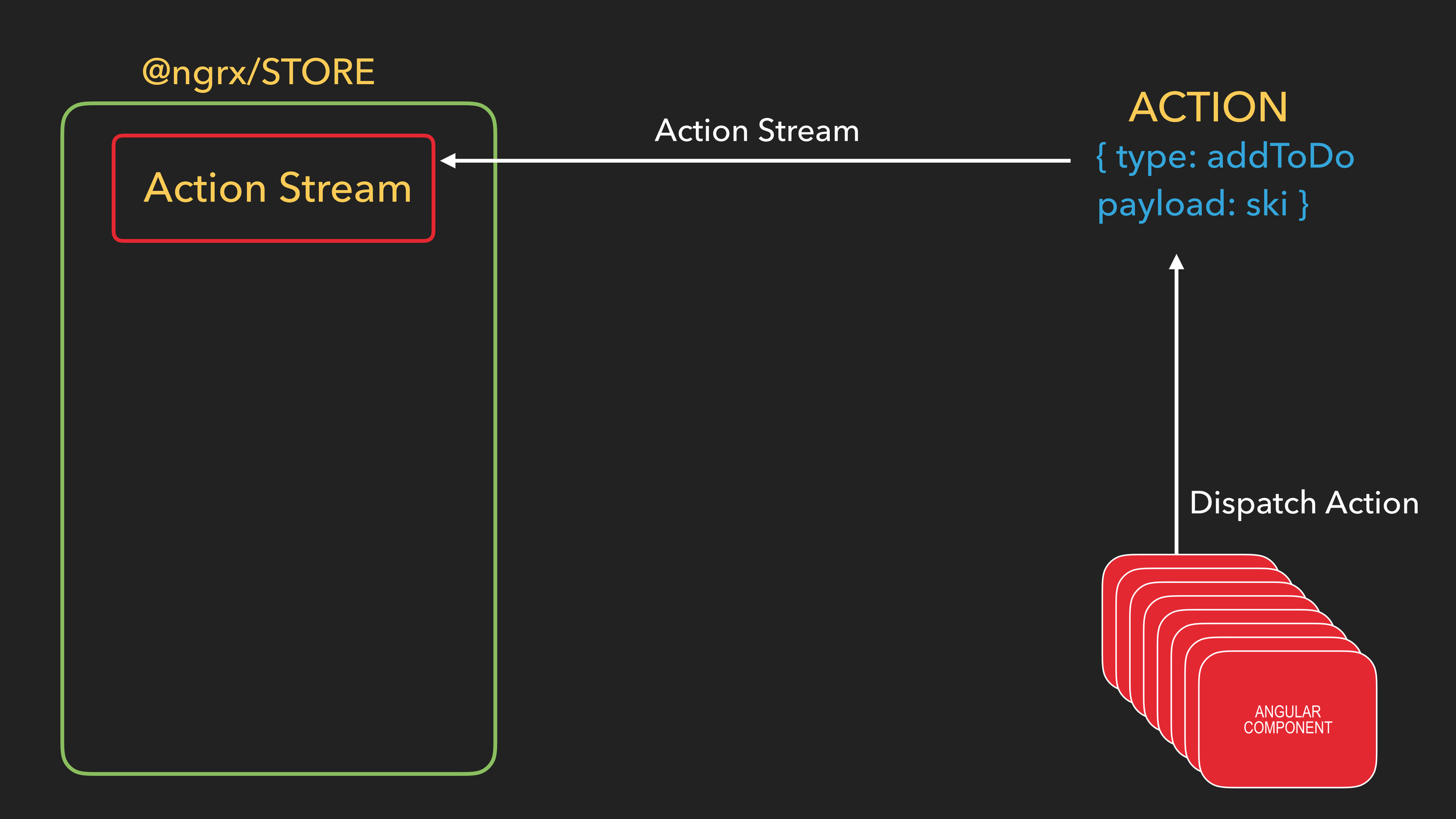
```
{ type: '[Layout] Open Sidenav' }
```

The Action

layout.actions.ts

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

export enum LayoutActionTypes {
  OpenSidenav = '[Layout] Open Sidenav',
  CloseSidenav = '[Layout] Close Sidenav',
}

export class OpenSidenav implements Action {
  readonly type =
LayoutActionTypes.OpenSidenav;
}

export class CloseSidenav implements Action {
  readonly type =
LayoutActionTypes.CloseSidenav;
}

export type LayoutActionsUnion = OpenSidenav |
CloseSidenav;
```

```
new LayoutActions.OpenSidenav()

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


@ngrx/STORE
Action Stream
Action Stream
ACTION
{ type: addToDo
payload: ski }
Dispatch Action
ANGULAR
COMPONENT

Dispatching an Action

- the dispatcher is a method on the store
- to use, inject the store as a service into your component

```
this.store.dispatch(
  { type: ‘[Layout] Open Sidenav’ }
);
```

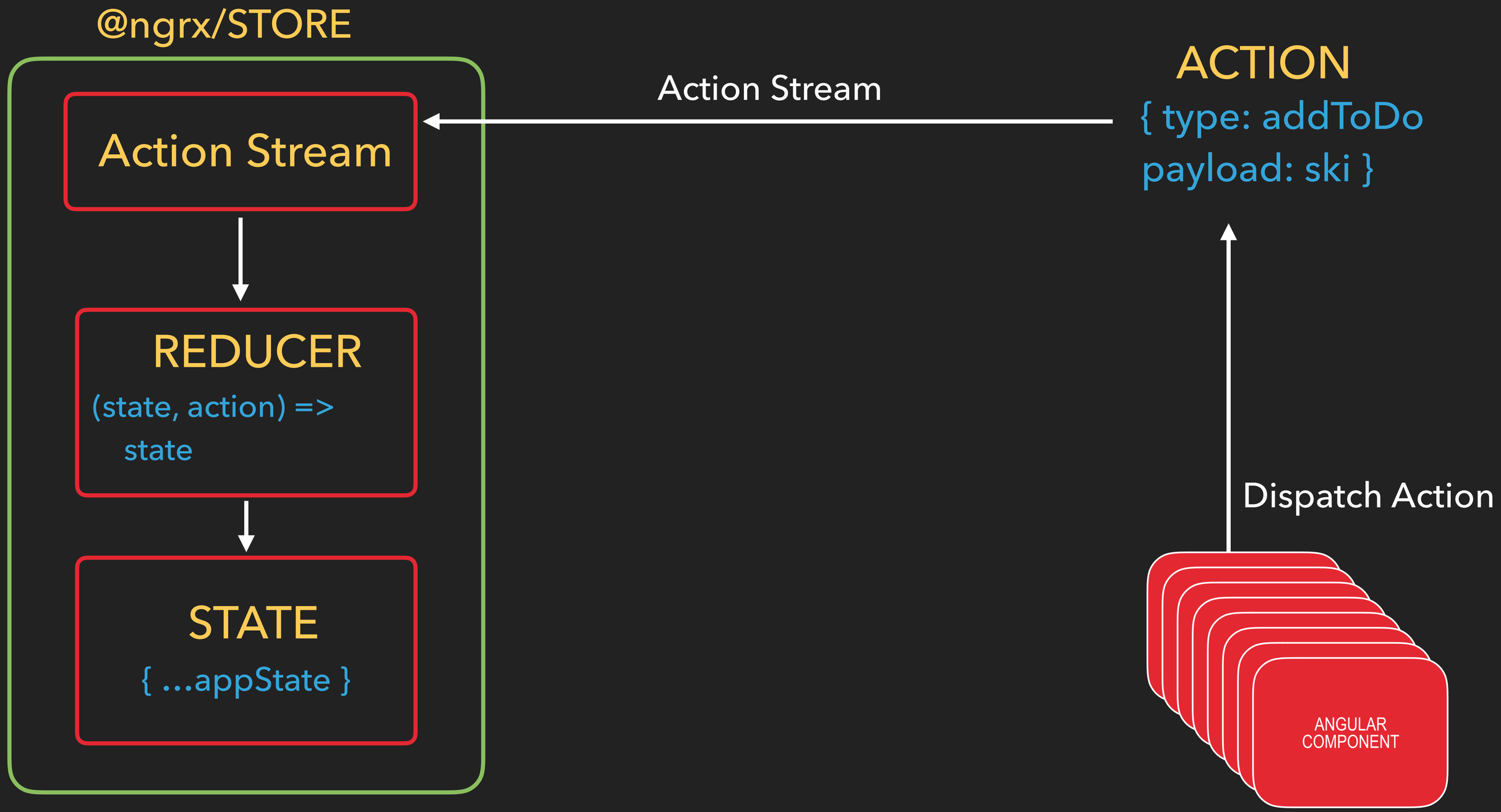
Dispatching an Action

app.component.html

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

app.component.ts

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


@ngrx/STORE
Action Stream
REDUCER
(state, action) =>
state
STATE
{ ...appState }
Action Stream
ACTION
{ type: addToDo
payload: ski }
Dispatch Action
ANGULAR
COMPONENT

The Reducer

- a pure function
- only reducers update (not mutate) state
- each slice of state has a corresponding reducer

The Reducer

layout.reducer.ts

```
export function reducer(
  state: LayoutState = initialState,
  action: LayoutActionsUnion
): LayoutState {
  switch (action.type) {
    case LayoutActionTypes.CloseSidenav:
      return {
        showSidenav: false,
      };

    case LayoutActionTypes.OpenSidenav:
      return {
        showSidenav: true,
      };

    default:
      return state;
  }
}
```

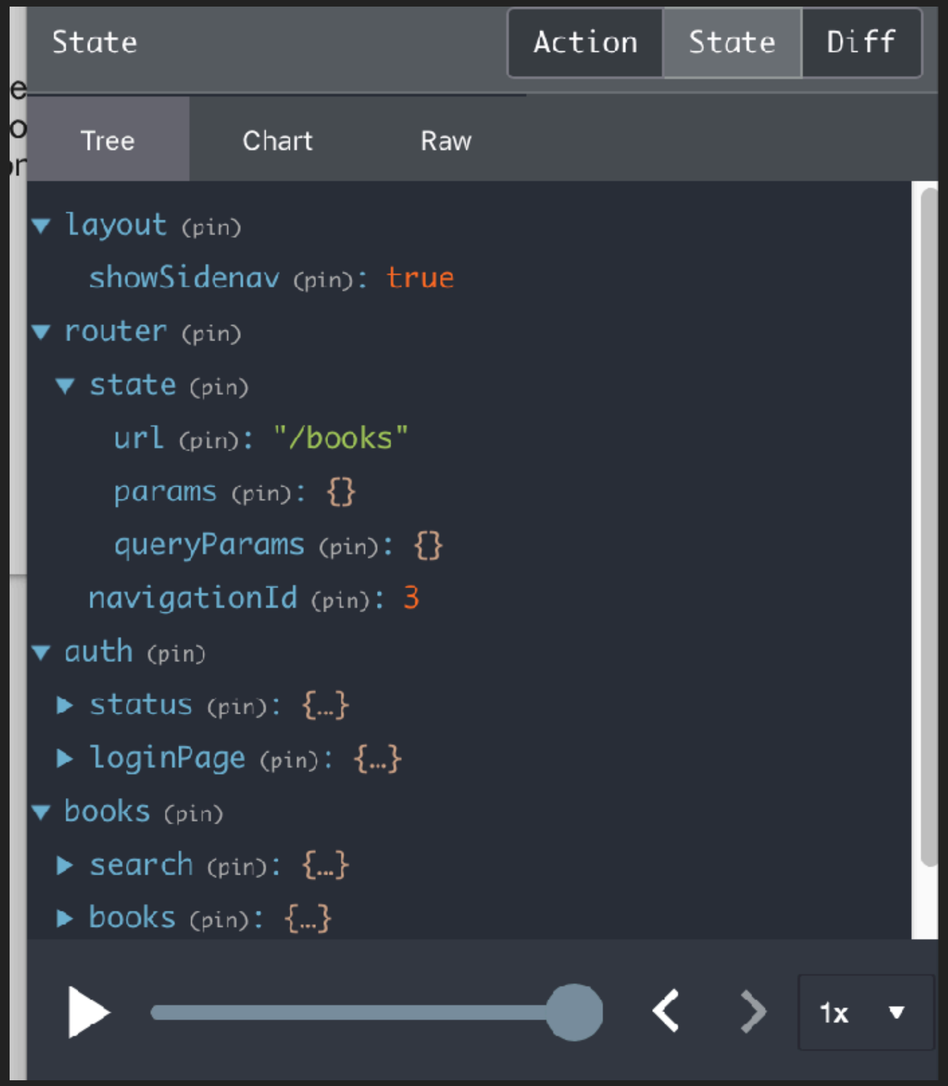
The State

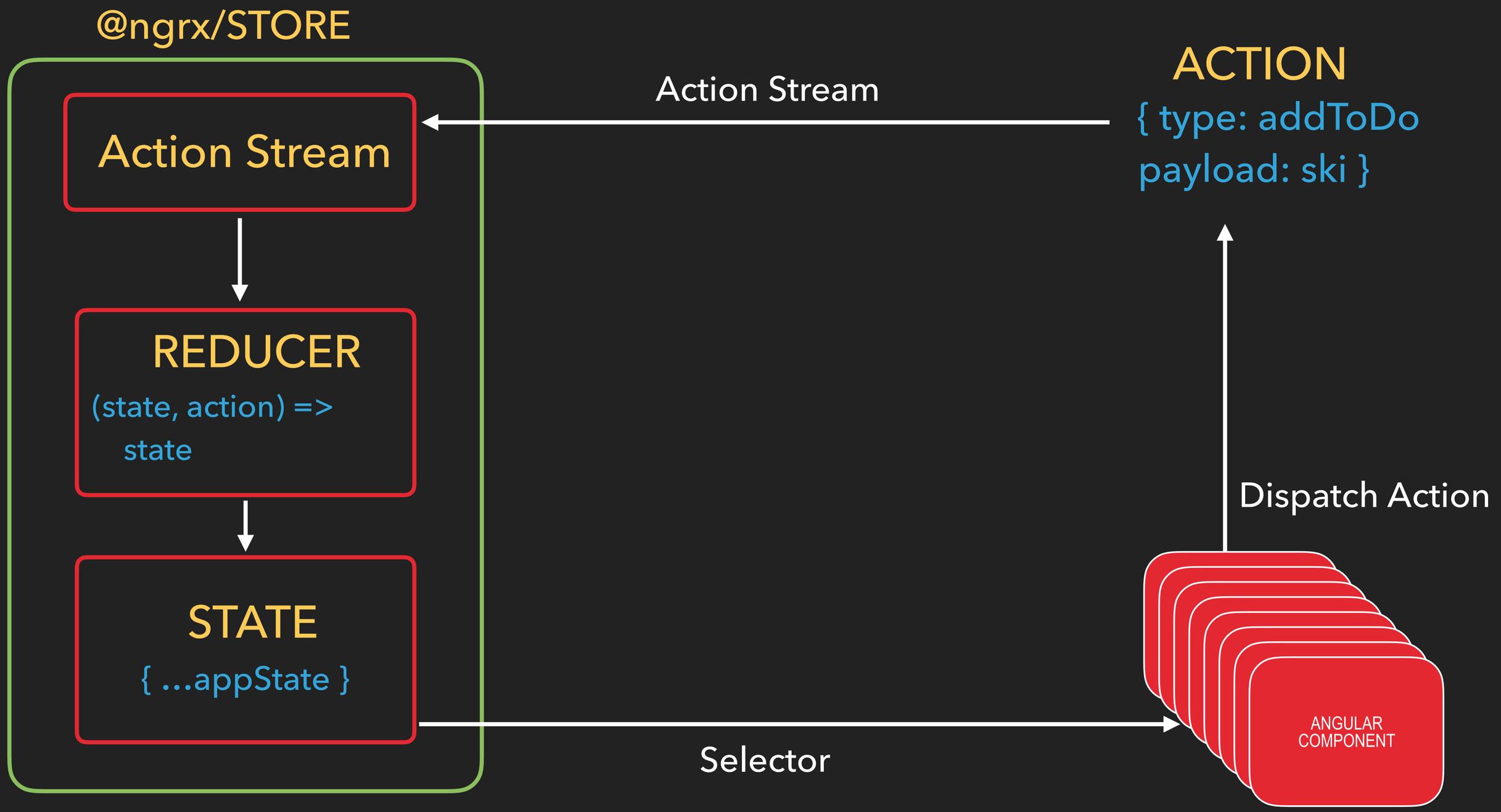
layout.reducer.ts

```ts
export interface LayoutState {
  showSidenav: boolean;
}

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

```ts
export function reducer(
  state: LayoutState = initialState,
  action: LayoutActionsUnion
): LayoutState {
  switch (action.type) {
    case LayoutActionTypes.CloseSidenav:
      return {
        showSidenav: false,
      };

    case LayoutActionTypes.OpenSidenav:
      return {
        showSidenav: true,
      };

    default:
      return state;
  }
}
```

The State


State
Action
State
Diff
Tree
Chart
Raw
layout (pin)
showSidenav (pin): true
router (pin)
state (pin)
url (pin): "/books"
params (pin): {}
queryParams (pin): {}
navigationId (pin): 3
auth (pin)
status (pin): {…}
loginPage (pin): {…}
books (pin)
search (pin): {…}
books (pin): {…}
1x


@ngrx/STORE
Action Stream
REDUCER
(state, action) =>
state
STATE
{ ...appState }
Action Stream
ACTION
{ type: addToDo
payload: ski }
Dispatch Action
Selector
ANGULAR
COMPONENT

The Selector

- callback function to filter Observable stream from the Store
- are composable
- are memoised

The Selector

Store State

```
{
    layout: {
        showSidenav: true
    },
    router: {...},
    auth: {...},
    books: {...}
}
```

The Selector

Store State

```
{
    layout: {
        showSidenav: true
    },
    router: {...},
    auth: {...},
    books: {...}
}
```

```
const getLayoutState =
  createFeatureSelector<LayoutState>('layout');
```

The Selector

Store State

```
{
    layout: {
        showSidenav: true
    },
    router: {...},
    auth: {...},
    books: {...}
}
```

```
const getLayoutState =
  createFeatureSelector<LayoutState>('layout');

const getShowSidenav = createSelector(
  getLayoutState,
  fromLayout.getShowSidenav
);
```

The Selector

Store State

```
{
    layout: {
        showSidenav: true
    },
    router: {...},
    auth: {...},
    books: {...}
}
```

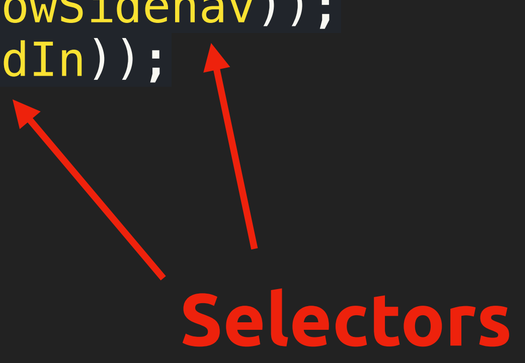
```
export class AppComponent {
  showSidenav$: Observable<boolean>;
  loggedIn$: Observable<boolean>;

  constructor(private store: Store<fromRoot.State>) {

    this.showSidenav$ = this.store.pipe(select(getShowSidenav));
    this.loggedIn$ = this.store.pipe(select(getLoggedIn));
  }
```

```
const getLayoutState =
  createFeatureSelector<LayoutState>('layout');

const getShowSidenav = createSelector(
  getLayoutState,
  fromLayout.getShowSidenav
);
```

Using Selectors in the Component

app.component.ts

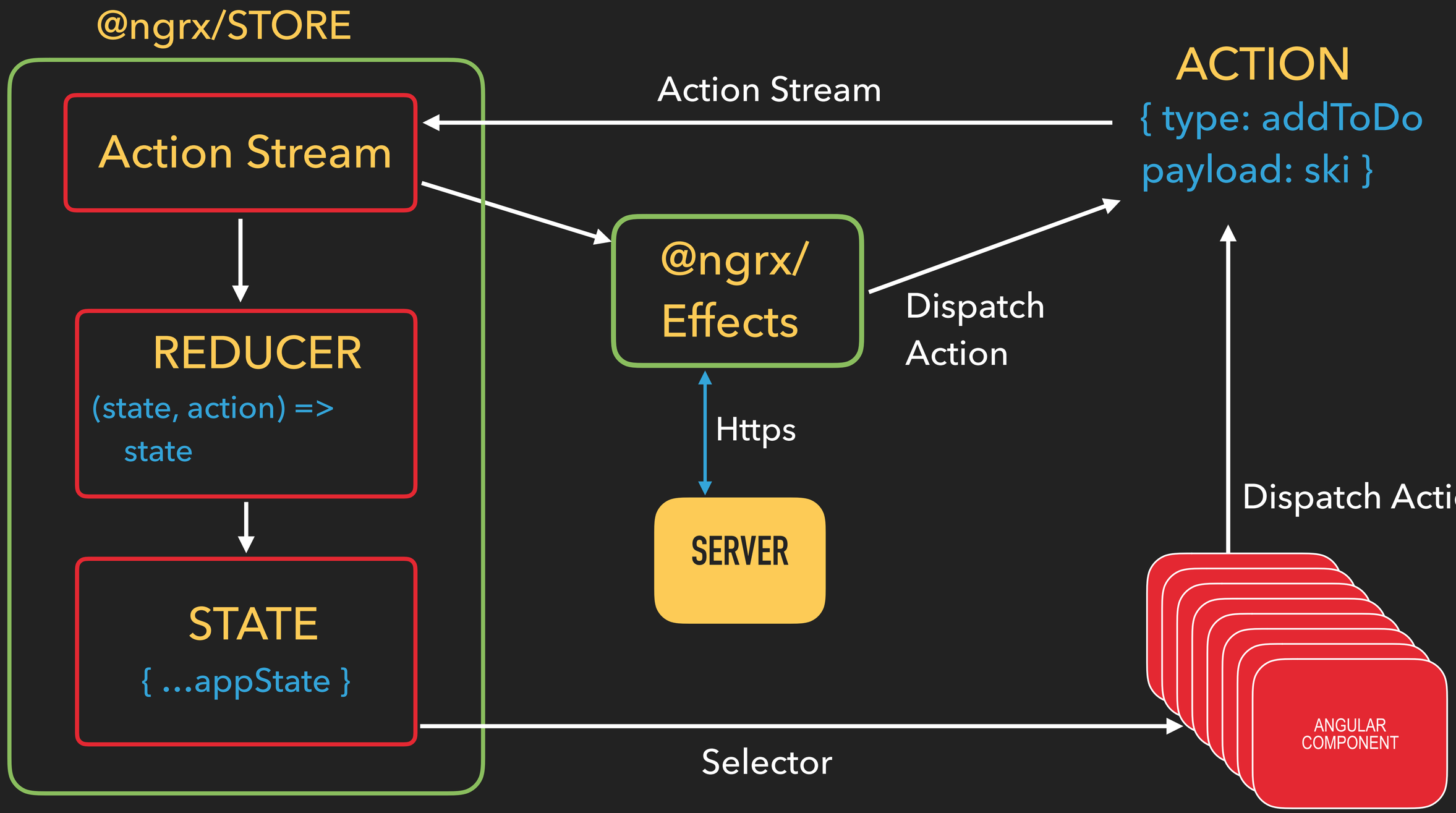
```
export class AppComponent {
  showSidenav$: Observable<boolean>;
  loggedIn$: Observable<boolean>;

  constructor(private store: Store<fromRoot.State>) {

    this.showSidenav$ = this.store.pipe(select(getShowSidenav));
    this.loggedIn$ = this.store.pipe(select(getLoggedIn));
  }
```



Subscribing to the returned Observable

app.component.html

```
<bc-sidenav [open]="showSidenav$ | async">
    <bc-nav-item (navigate)="closeSidenav()" *ngIf="loggedIn$ | async"
routerLink="/" icon="book" hint="View your book collection">
        My Collection
    </bc-nav-item>
    <bc-nav-item (navigate)="closeSidenav()" *ngIf="loggedIn$ | async"
routerLink="/books/find" icon="search" hint="Find your next book!">
        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>
```


@ngrx/STORE
Action Stream
REDUCER
(state, action) =>
state
STATE
{ ...appState }
Action Stream
ACTION
{ type: addToDo
payload: ski }
@ngrx/
Effects
Dispatch
Action
Https
SERVER
Dispatch Action
ANGULAR
COMPONENT
Selector

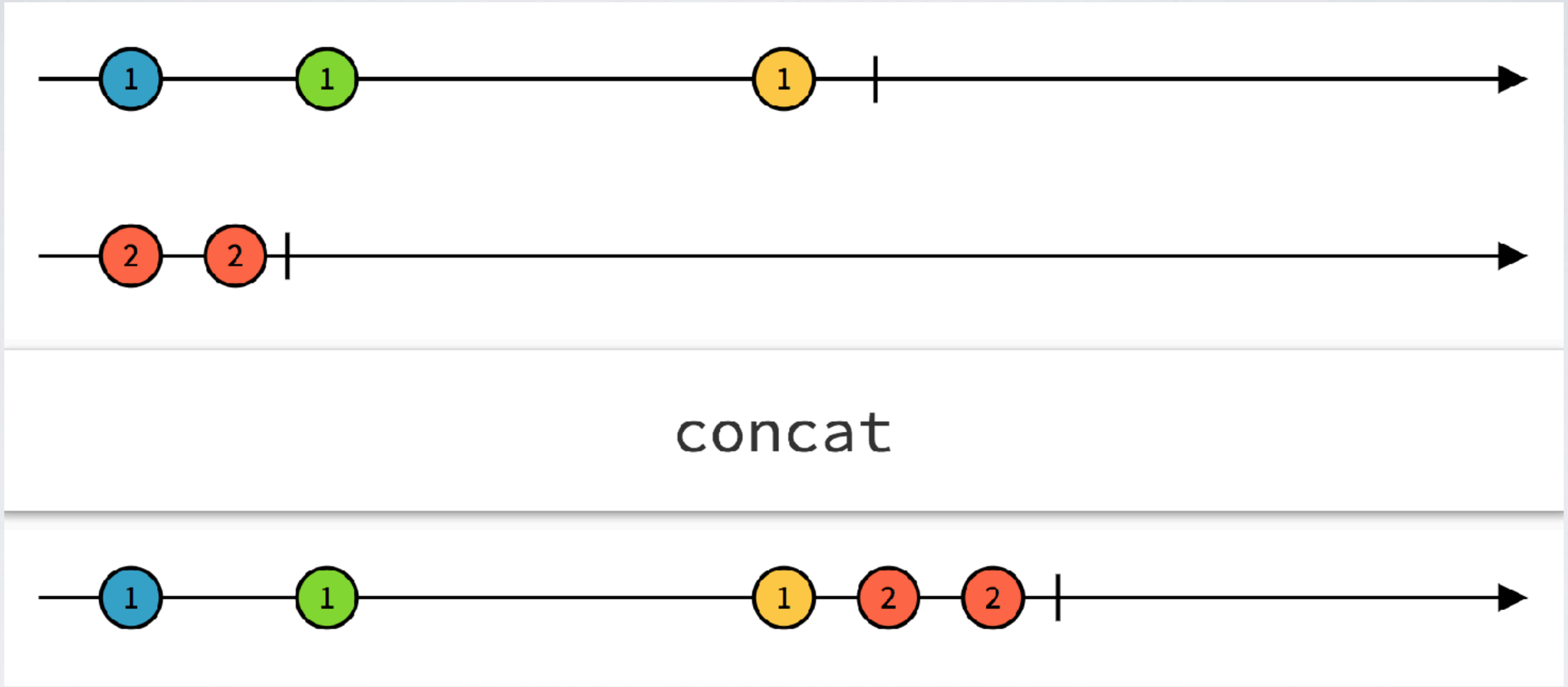
The Effect

```
@Injectable()
export class BookEffects {
  @Effect()
  search$: Observable<Action> = this.actions$.pipe(
    ofType<Search>(BookActionTypes.Search),
    debounceTime(this.debounce || 300, this.scheduler || asyncScheduler),
    map(action => action.payload),
    switchMap(query => {
      if (query === '') {
        return empty();
      }

      const nextSearch$ = this.actions$.pipe(
        ofType(BookActionTypes.Search),
        skip(1)
      );

      return this.googleBooks
        .searchBooks(query)
        .pipe(
          takeUntil(nextSearch$),
          map((books: Book[]) => new SearchComplete(books)),
          catchError(err => of(new SearchError(err)))
        );
    })
  );
```


CALLBACK WITHIN A CALLBACK
WITHIN A CALLBACK
WITHIN A CALLBACK

Another 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)])
    );
```



rxmarbles.com

More performant change detection: OnPush

```
@Component({
 changeDetectionStrategy.OnPush
})
```

!! Mutating the state, squirrel is the same reference

```
squirrel.age = 12;
squirrel.energyLevel = low;
```

```
squirrel: any = {
  tail: 'bushy',
  color: 'red-brown',
  age: 2,
  energyLevel: 'insane',
  favoriteFood: 'pizza'
};
```

!! Giving squirrel a new reference to a new object

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

Nx: Angular + Ngrx cli

nrwl.io/nx

Ngrx might be the right tool if...

- User workflows are complex
- Your app has a large variety of user workflows (ex: 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 serve a single view

Stay in touch!

@ajpasternack
github.com/apasternack
<adam.pasternack@gmail.com>

STIR TREK

2018 May 4
Columbus OH

Adam Pasternack

If you, your company, or someone your know is thinking about using Angular or Ngrx, need some help getting started, or are encountering problems with your app, or you need some Angular Developers, please reach out!

improving
It's what we do.