

REDUX & ANGULAR 2.0



Nir Kaufman

Nir Kaufman

Head of Angular Development @ **500Tech**

- AngularJS evangelist
- International speaker
- Guitar player

**Photoshop*



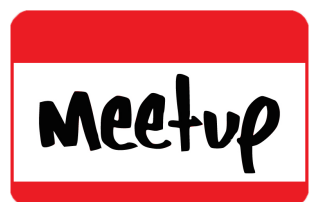
500Tech

WE DEVELOP, CONSULT AND TRAIN

 ANGULAR,  REACT &  NODE.



ANGULARJS IL



THE CHALLENGE

SPA BECOME
INCREASINGLY
COMPLICATED

THERE IS NOTHING
WRONG WITH THE
MVC PATTERN

IF YOU ARE
BUILDING A **CRUD**
APPLICATION

Customers

Filter customers...



First name	Last name	Email	Status
Gabrielle	Patel	gabrielle@patel.com	Imported
Brian	Robinson	brian@robinson.com	Customer
Eduardo	Haugen	eduardo@haugen.com	Contacted
Koen	Johansen	koen@johansen.com	Imported
Alejandro	Macdonald	alejandro@macdonald.com	Contacted
Angel	Karlsson	angel@karlsson.com	Contacted
Yahir	Gustavsson	yahir@gustavsson.com	Contacted
Haiden	Svensson	haiden@svensson.com	Imported
Emily	Stewart	emily@stewart.com	Contacted
Corinne	Davis	corinne@davis.com	Customer
Ryann	Davis	ryann@davis.com	NotContac
Yurem	Jackson	yurem@jackson.com	ClosedLo
Kelly	Gustavsson	kelly@gustavsson.com	Contacted
Eileen	Walker	eileen@walker.com	NotContac
Katelyn	Martin	katelyn@martin.com	NotContac

Edit customer

First name Last name Email * Birth day Gender ☐ Female ☒ MaleStatus

Save

Cancel



BUT WE ARE
PUSHING THE
ENVELOPE AS MUCH
AS WE CAN




Alexander Pierce

Online











Search...

MAIN NAVIGATION

Dashboard 




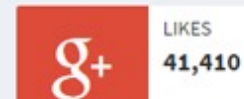
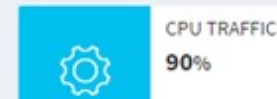
Dashboard v1

Dashboard v2

Layout Options  4Widgets  newCharts UI Elements Forms Tables Calendar  3Mailbox  12Examples Multilevel 

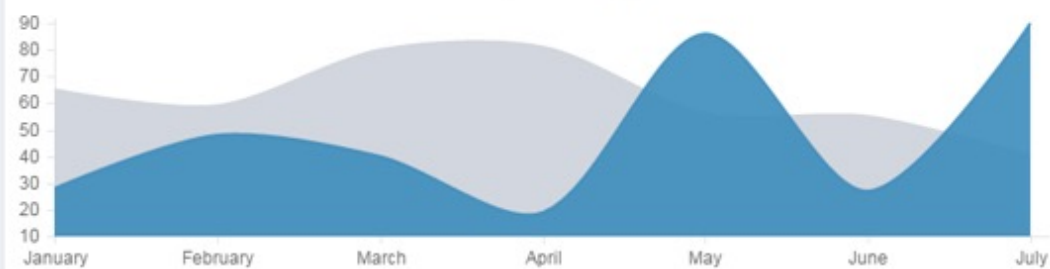
Documentation

LABELS

 Important Warning InformationDashboard Version 2.0[Home](#) > [Dashboard](#)

Monthly Recap Report

Sales: 1 Jan, 2014 - 30 Jul, 2014



Goal Completion

 17%
\$35,210.43
TOTAL REVENUE 0%
\$10,390.90
TOTAL COST 20%
\$24,813.53
TOTAL PROFIT 18%
1200
GOAL COMPLETIONS

Visitors Report


8390
VISITS
30%
REFERRALS
70%
ORGANIC

INVENTORY

5,200

50% Increase in 30 Days



MENTIONS

92,050

20% Increase in 30 Days



DOWNLOADS

114,381

70% Increase in 30 Days



DIRECT MESSAGES

163,921

40% Increase in 30 Days

Direct Chat

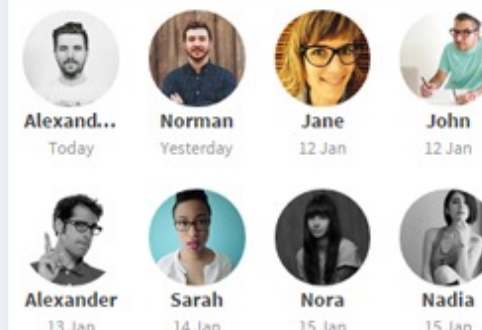
Alexander Pierce 23 Jan 2:00 pm
Is this template really for free? That's unbelievable!

23 Jan 2:05 pm Sarah Bullock
You better believe it!

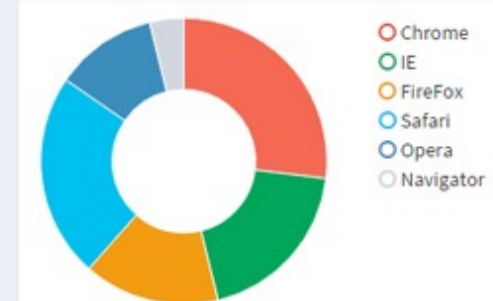
Alexander Pierce 23 Jan 5:37 pm
Working with AdminLTE on a great new app! Wanna join?

Latest Members

8 New Members

[View All Users](#)

Browser Usage



United States of America

 12%

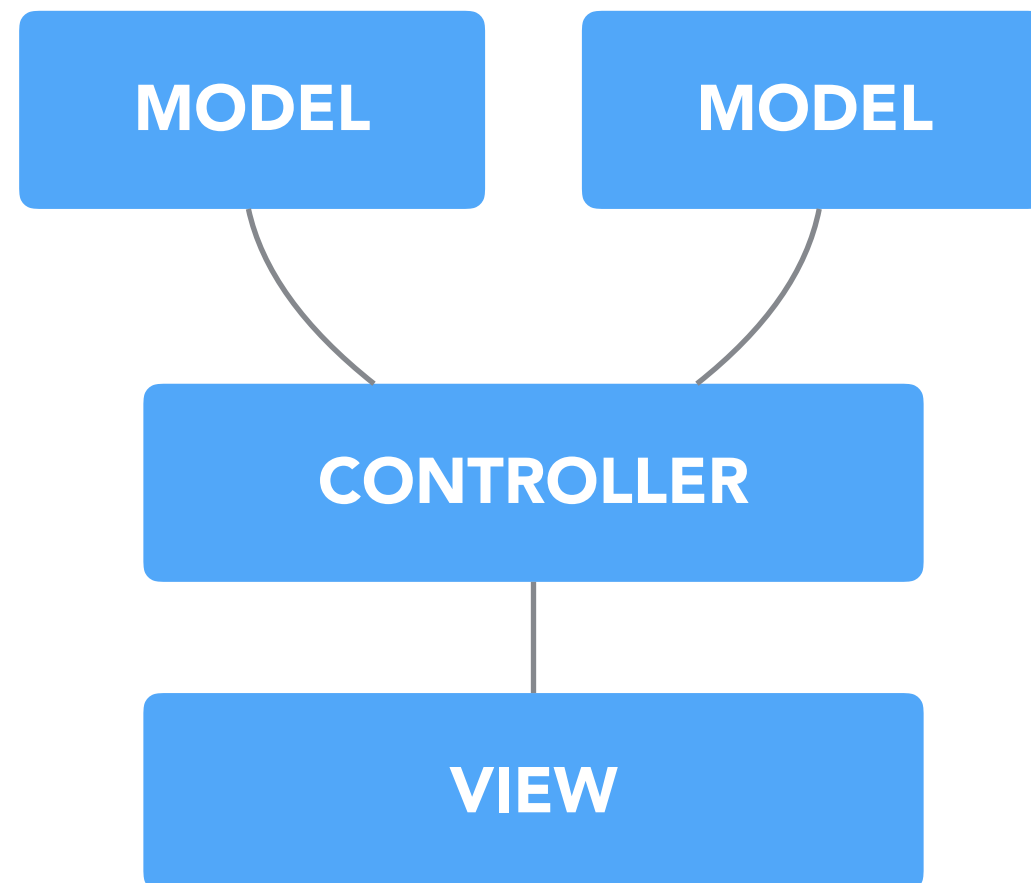
MANAGING AN
EVER-CHANGING STATE
IS A HARD TASK

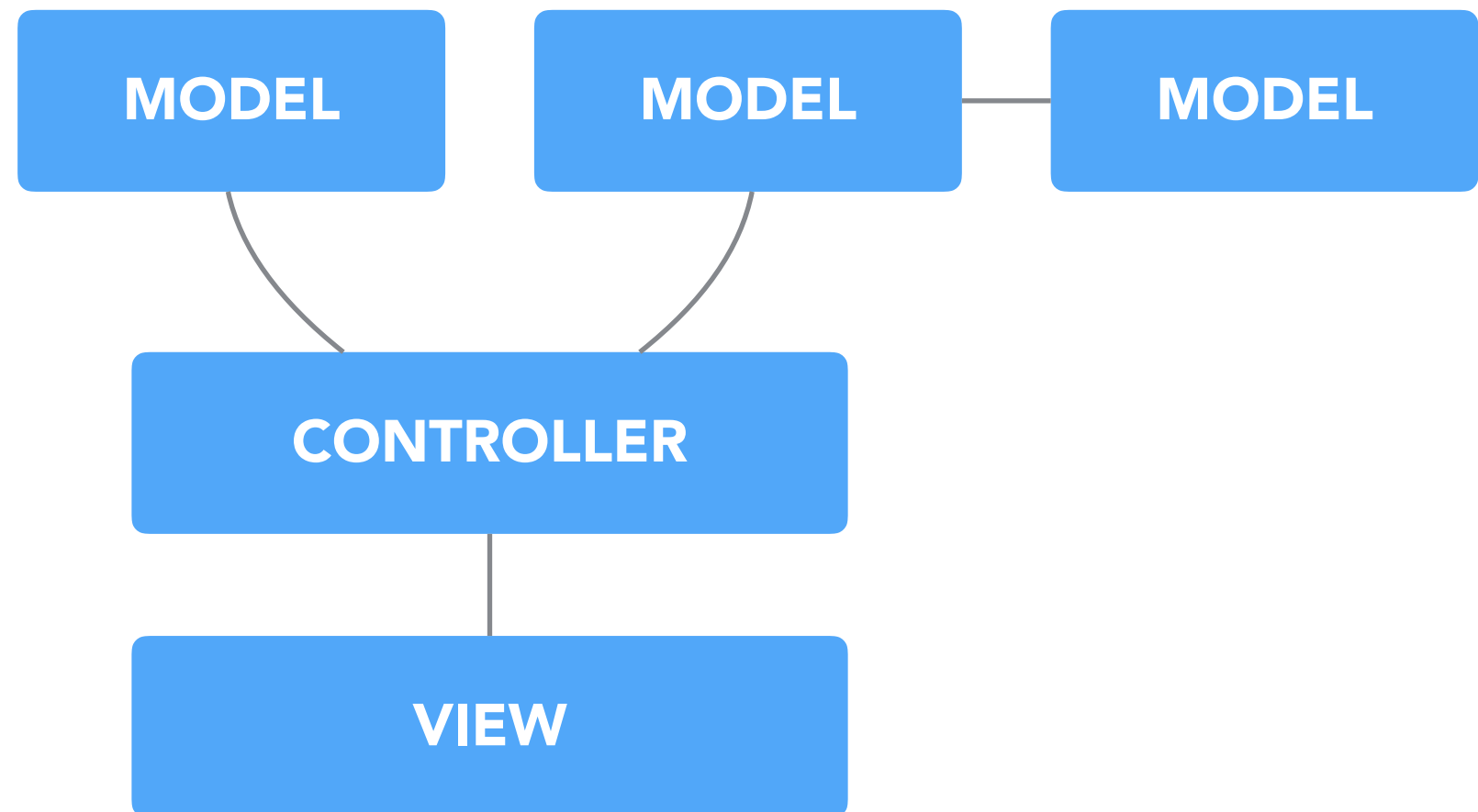
EVERYTHING IS
CONNECTED TO
EVERYTHING

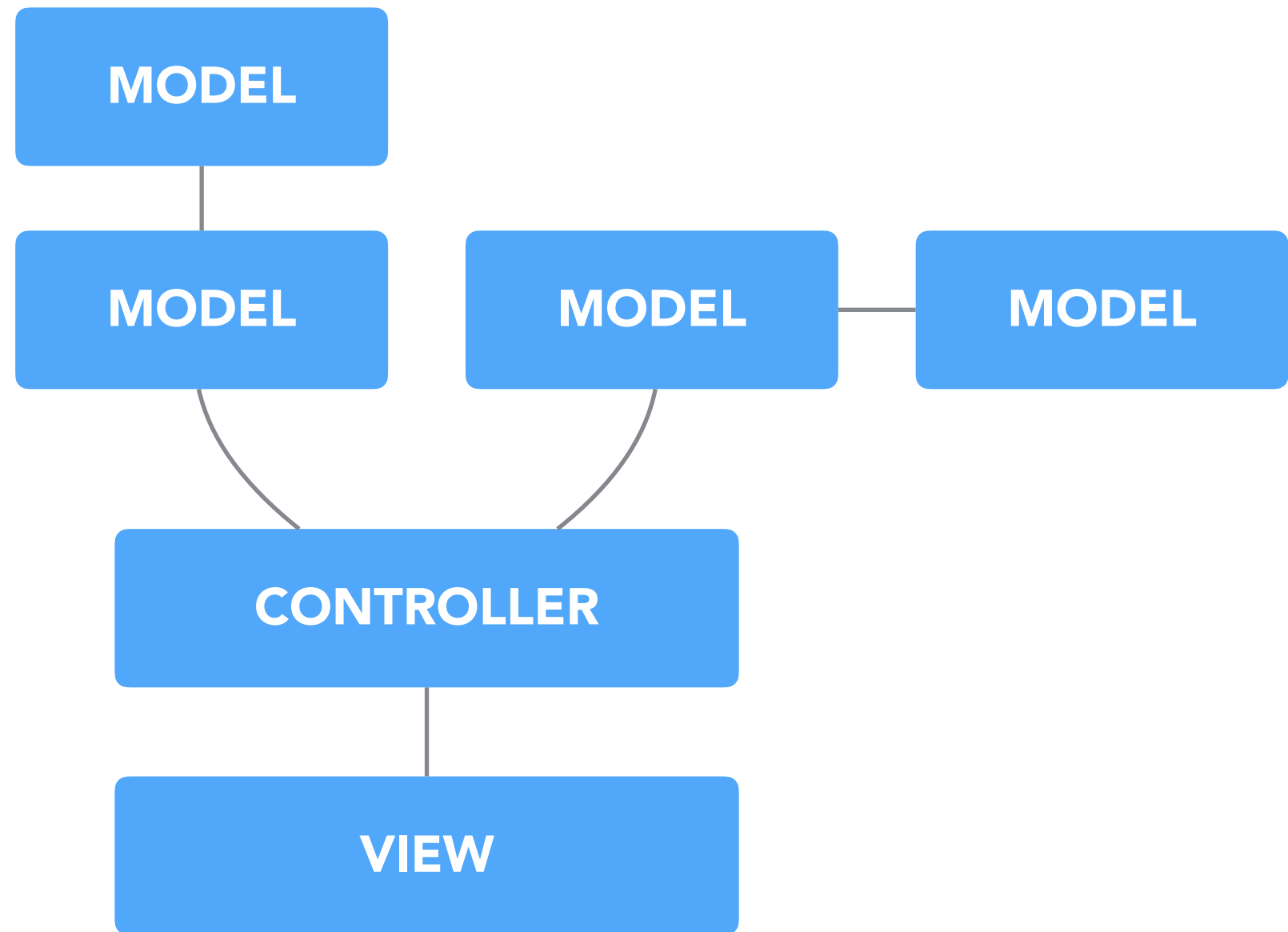
VIEW

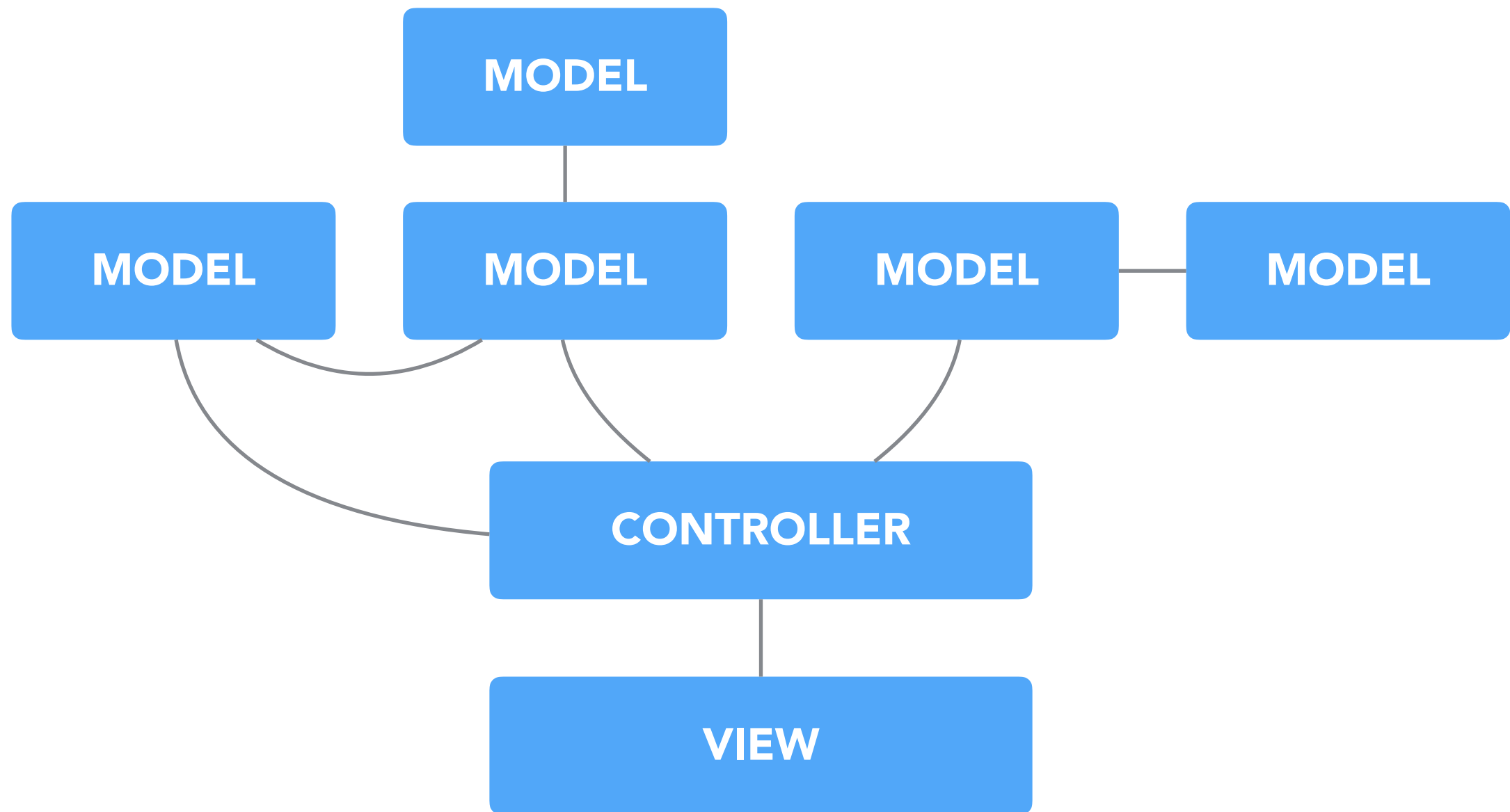
CONTROLLER

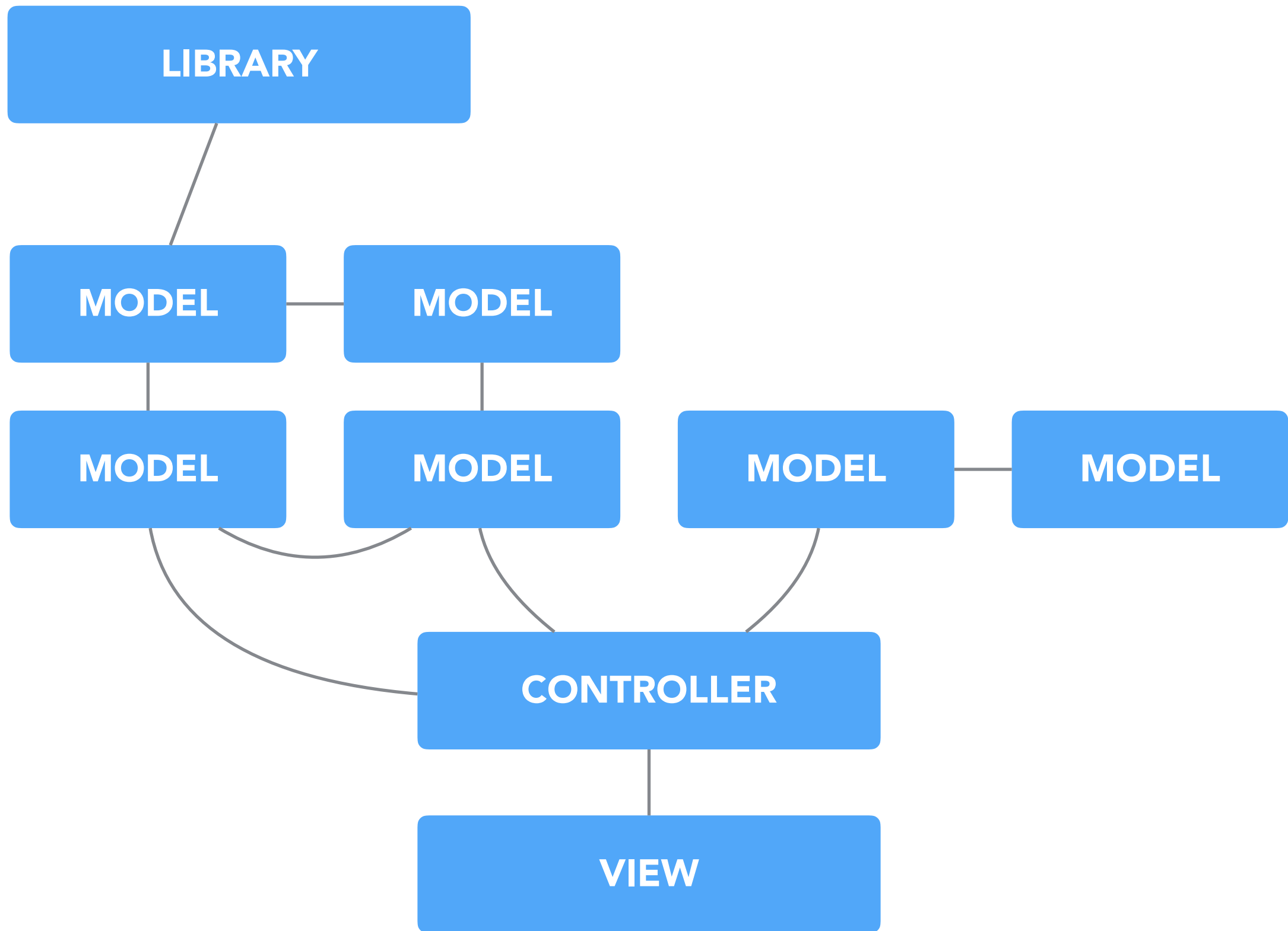
VIEW

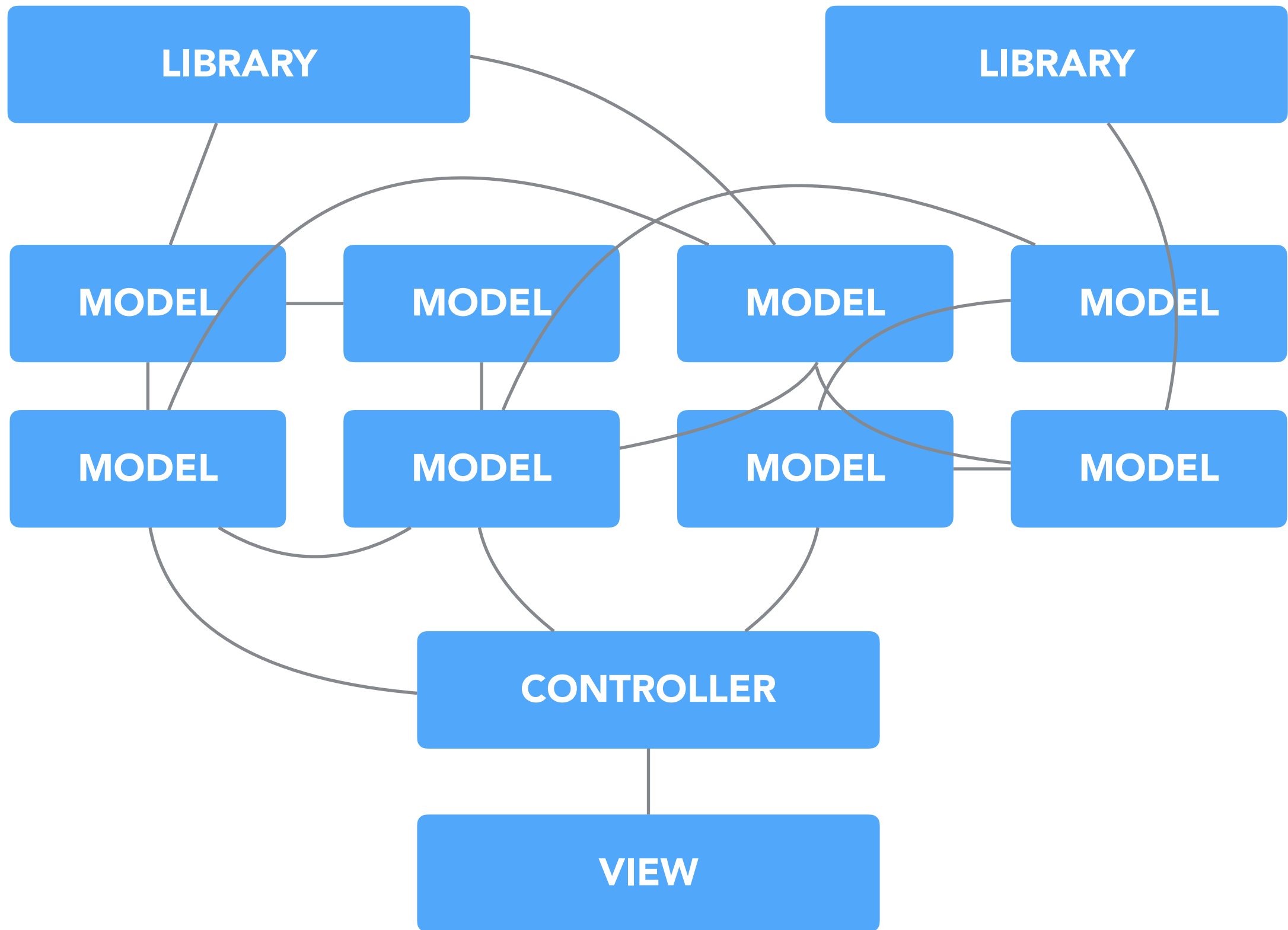












CHANGING SOMETHING
BREAKS SOMETHING
SOMEWHERE



ENTER REDUX



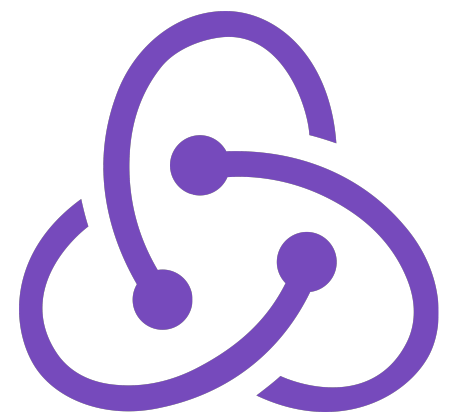
PLAY ALONG



<http://tinyurl.com/hq23lsa>

<https://github.com/nirkaufman/redux-playground>

REDUX IS A LIBRARY
FOR IMPLEMENTING A
DESIGN PATTERN



REDUX ATTEMPTS TO
MAKE STATE MUTATIONS
PREDICTABLE

INSPIRED BY
FLUX, CQRS &
EVENT SOURCING

REDUX INTREDUCE

THREE PRINCIPLES

SINGLE SOURCE OF TRUTH

*the state of your whole application is stored in
an object tree within a single **store***



THE TRUTH IS OUT THERE

Stateful components

```
class SideBarComponent {  
    private visible: boolean;  
  
    toggle(){  
        this.visible = !this.visible  
    }  
}
```


Stateful components

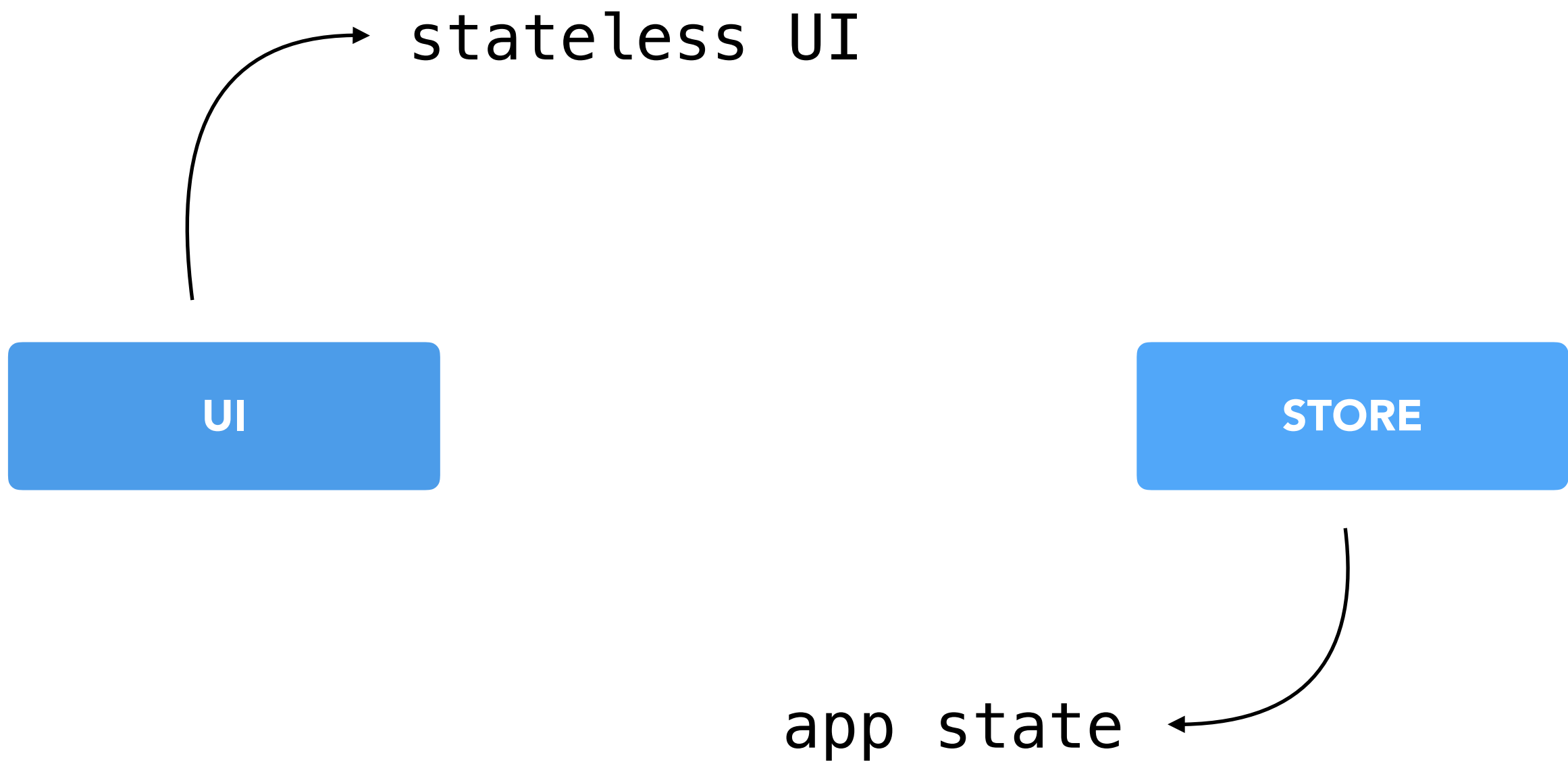
```
class TabsComponent {  
  
    private activeTab:Tab;  
  
    activateTab(tab) {  
        this.activeTab = tab;  
    }  
}
```

Data Models

```
class Accounts {  
  
    private accounts: Account[];  
  
    getAccounts() {  
        return this.accounts;  
    }  
}
```

Application state

```
const state = {  
  tabs: [],  
  accounts: [],  
  sidebar: {}  
};
```

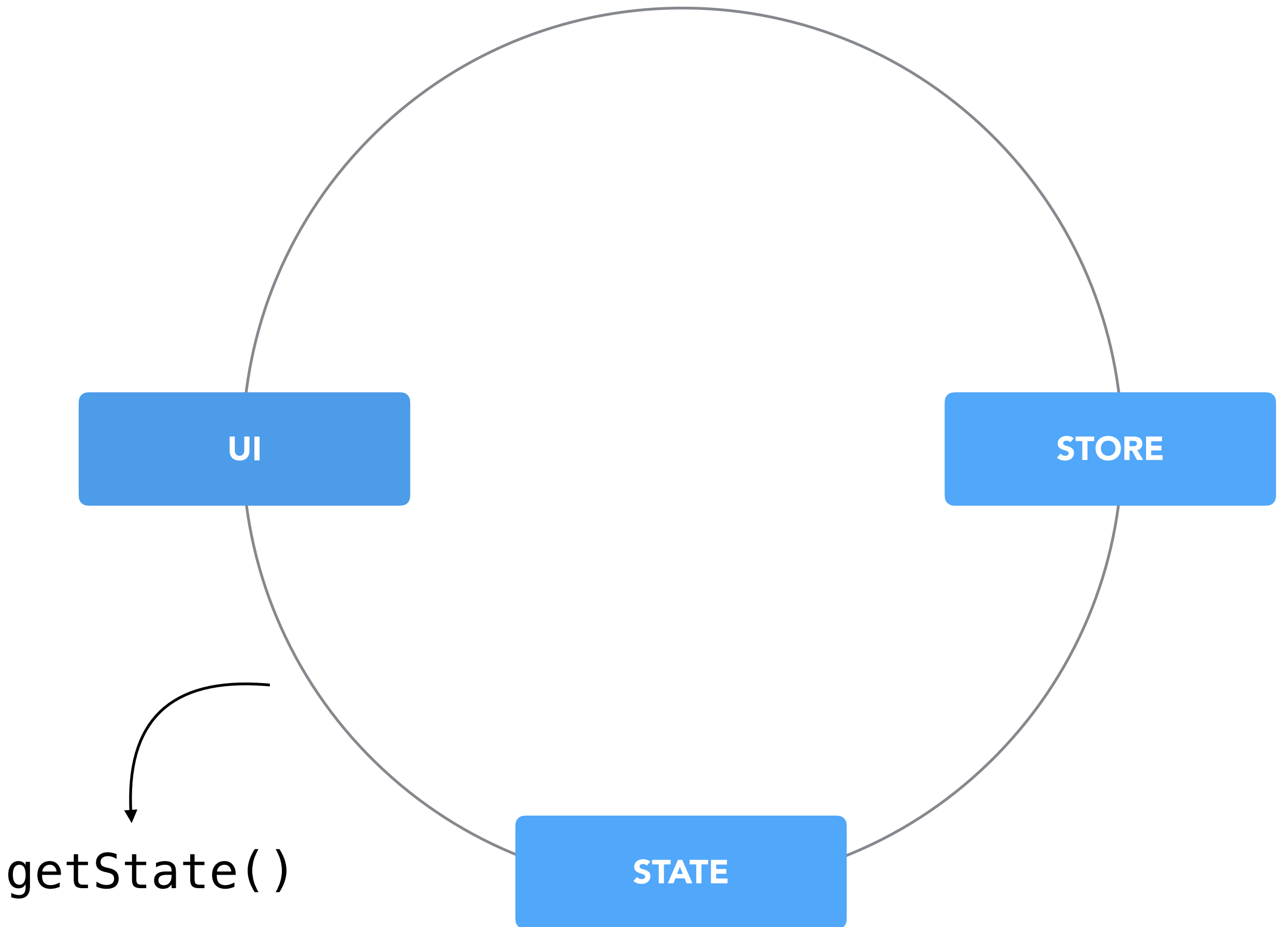


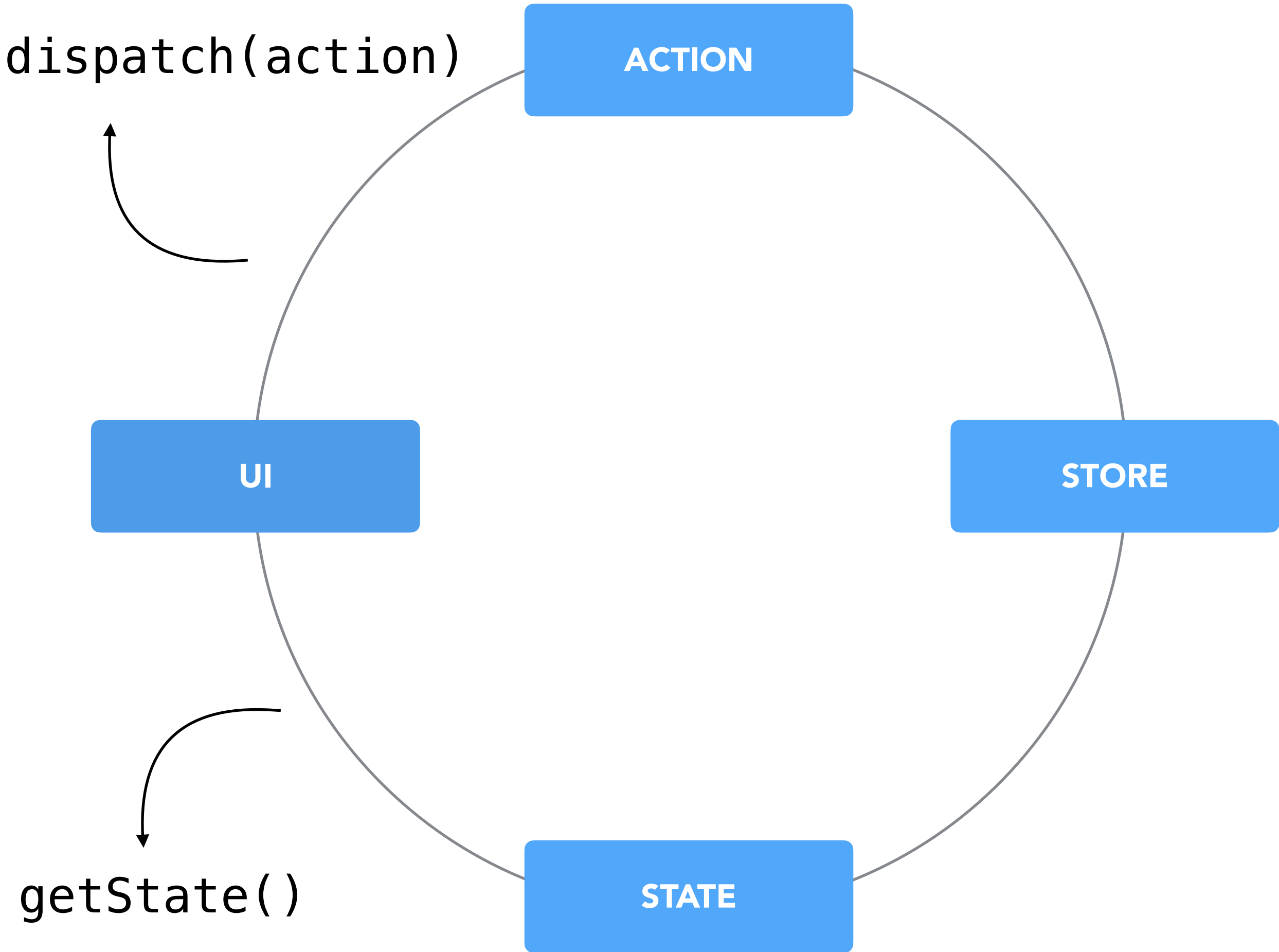
THE STATE IS READ ONLY

*the only way to mutate the state is to emit an **action**, an object describing what happened*

Read-only State

```
class Store {  
    private state: Object;  
  
    getState(){  
        return this.state;  
    }  
}
```





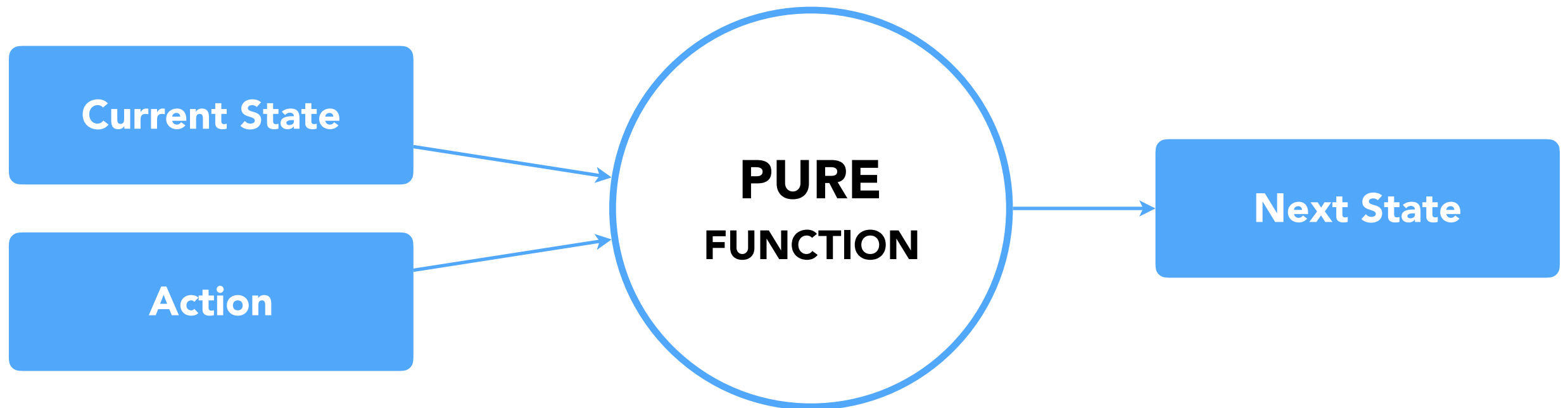
PURE FUNCTIONS

*to specify how the state tree is transformed by actions, you write **pure functions**.*

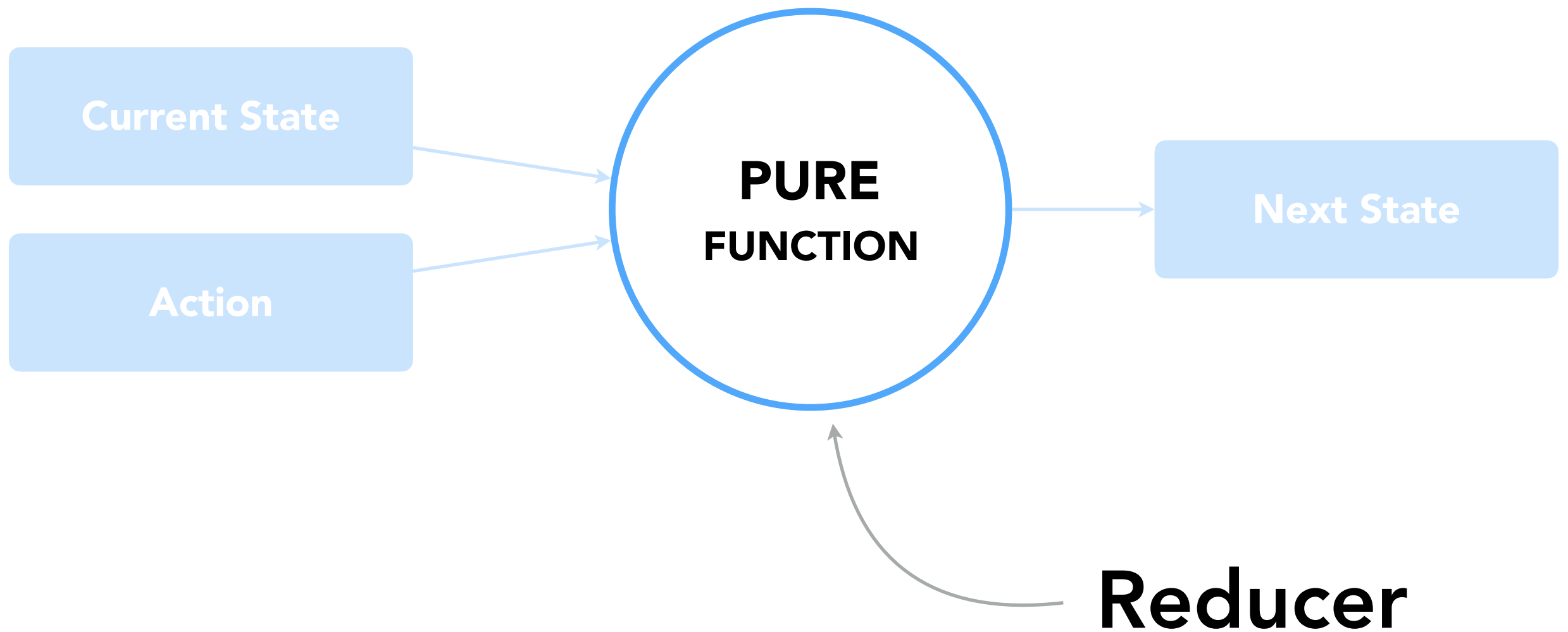
PURE FUNCTION

*return value is only determined by its input values,
without observable **side effects**.*

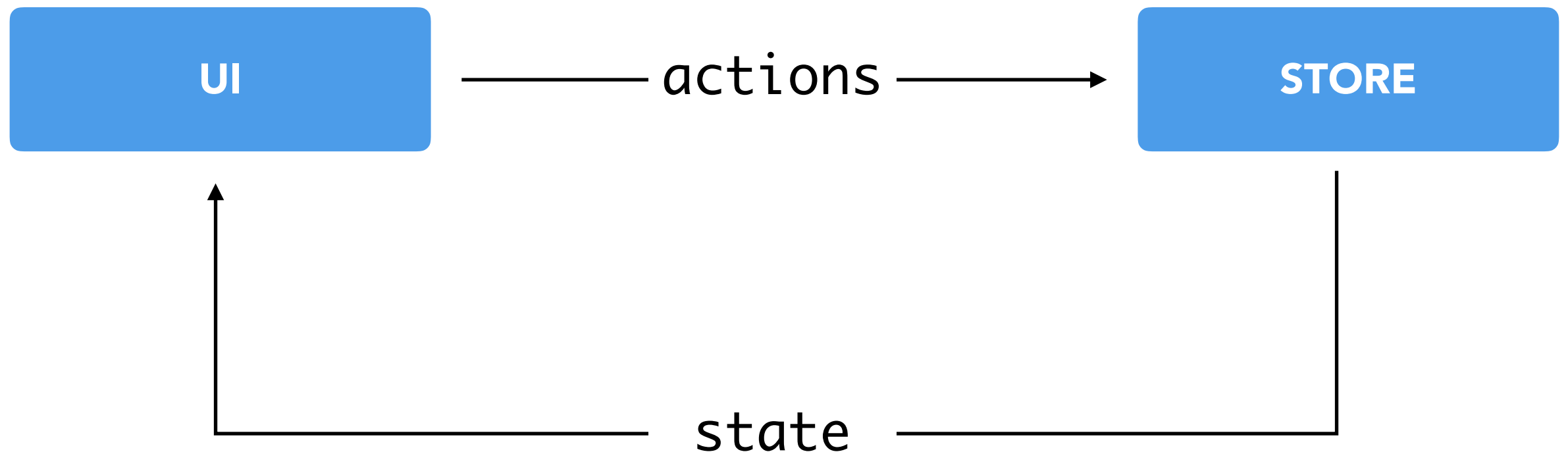
Calculate the next state



Calculate the next state



Uni directional data flow



ENTER THE STORE

THE STORE IS THE
HEART OF **REDUX**

TO CREATE A STORE
WE NEED A **REDUCER**


```
import { createStore } from 'redux';
```

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

```
import { createStore } from 'redux';
```

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

REDUCE METHOD

*applies a function against an accumulator and each value of the array (from left-to-right) to **reduce it to a single value.***

Reduce in action

```
function sum (previousVal, currentVal) {  
    return previousVal + currentVal;  
}
```

```
[0, 1, 2, 3, 4].reduce(sum);
```

```
// => 10
```

EVENT SOURCING

*capture all changes to an application state as a
sequence of events.*

Simple counter app

```
function counter (state, action) {  
  switch (action) {  
    case 'up':  
      return state + 1;  
    case 'down':  
      return state - 1;  
    default:  
      return state;  
  }  
}  
  
['up', 'up', 'down'].reduce( counter, 0 );
```

THE REDUCER RETURNS
THE NEXT **STATE**

BASED ON A SEQUENCE
OF **ACTIONS**

THE SAME SEQUENCE
OF **ACTIONS**

WILL PRODUCE THE
SAME **STATE**

PREDICTABLE STATE CONTAINER

STORE API

`dispatch(action)`

`subscribe(listener)`

`getState()`

`replaceReducer(reducer)`

HANDS ON!

*implementing a working store
in less than **30 lines of code.***

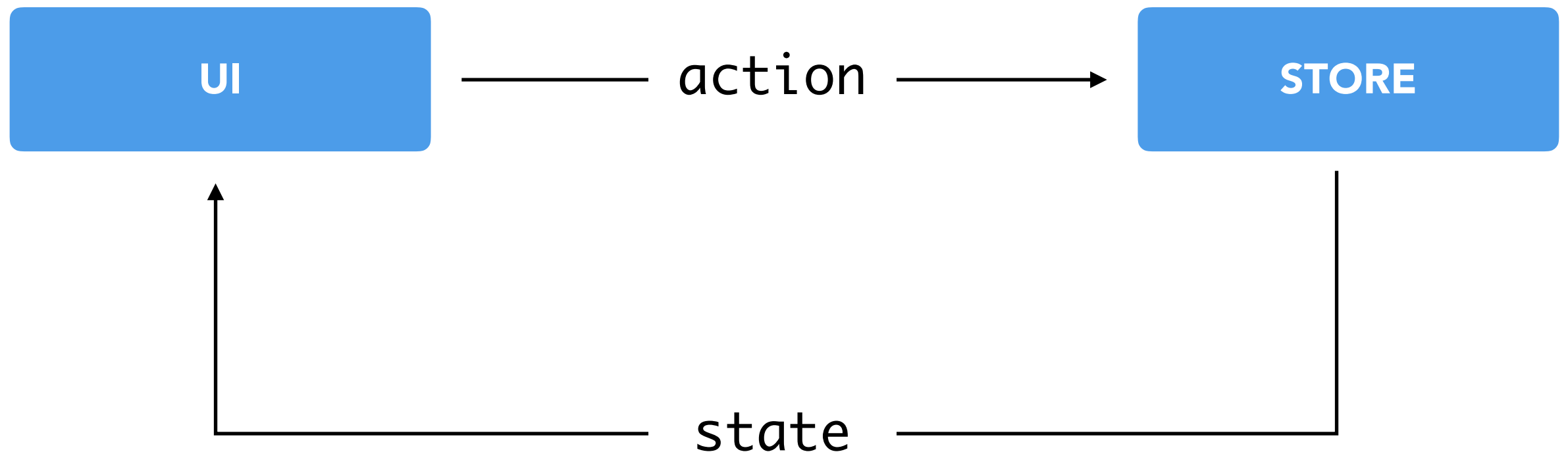
```
function createStore(reducer) {  
  let state = null;  
  const listeners = [];  
  
  function getState() {  
    return state;  
  }  
  
  function dispatch(action) {  
    state = reducer(state, action);  
    listeners.forEach( listener => listener() )  
  }  
  
  function subscribe(listener) {  
    listeners.push(listener);  
    return function unsubscribe() {  
      let index = listeners.indexOf(listener);  
      listeners.splice(index, 1)  
    }  
  }  
  
  return { getState, dispatch, subscribe }  
}
```

ASYNCHRONOUS DATA FLOW

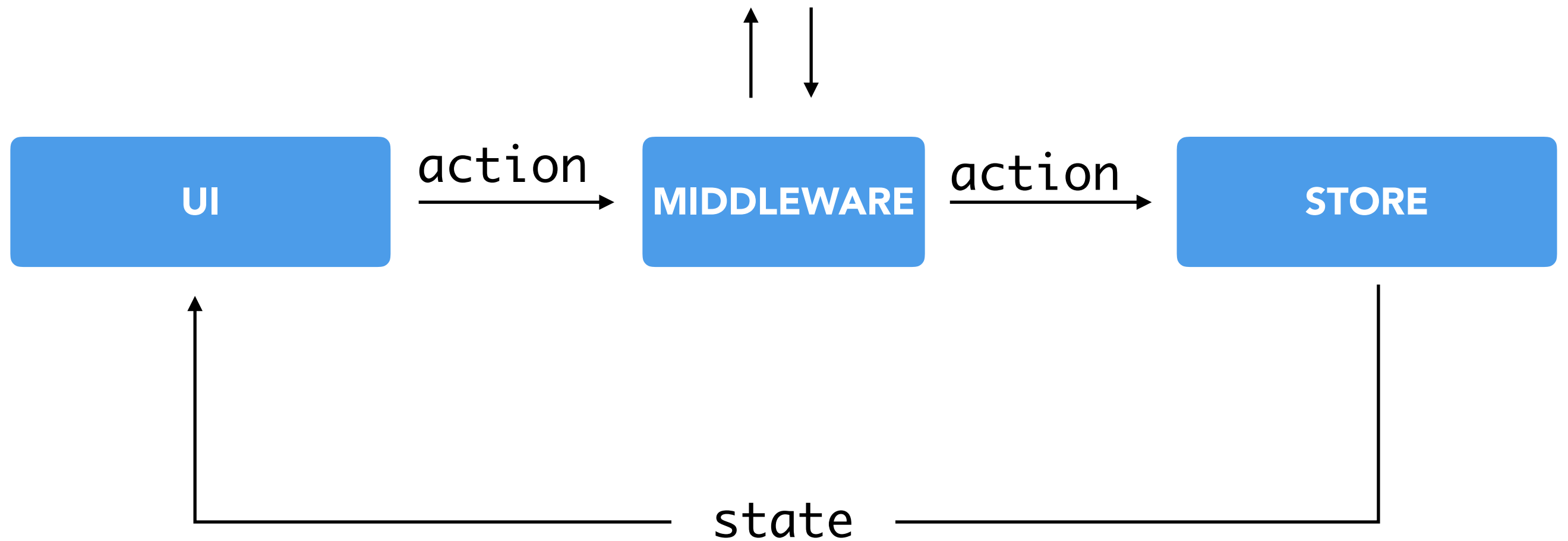
MIDDLEWARE

extension point between dispatching an action, and the moment it ***reaches the reducer.***

Async flow with middlewares



Async flow with middlewares



Middleware

```
export const middleware = store => next => action => {  
  return next(action)  
};
```

- get the current **state** from the **store**
- pass an action to the **next** middleware
- access the provided **action**

ANGULAR & REDUX

ANGULAR IS A NEW
PLATFORM FOR BUILDING
COMPLEX MODERN SPA'S





GET THE CODE



<http://tinyurl.com/h4bqmut>

<https://github.com/nirkaufman/angular2-redux-workshop.git>

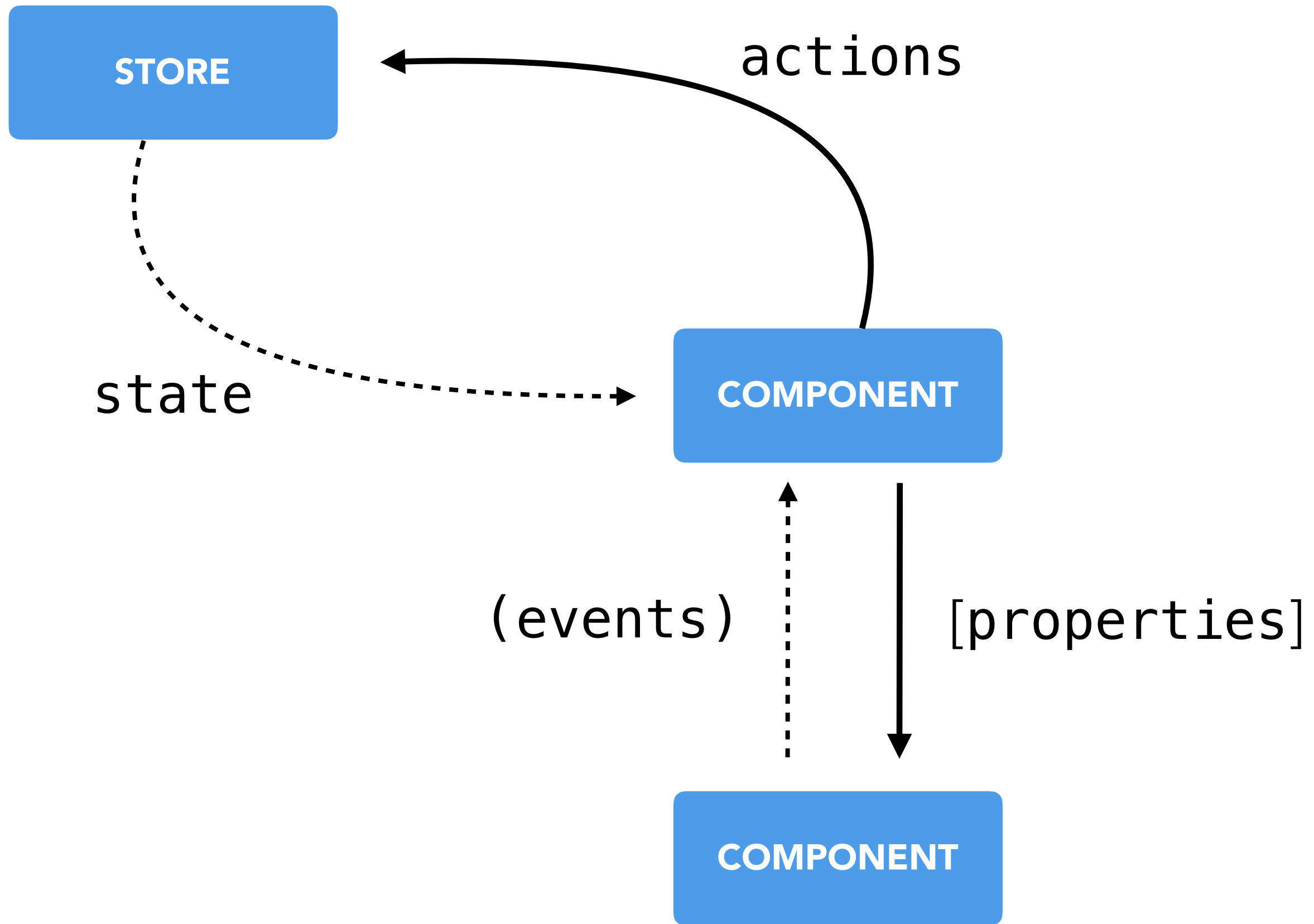


`git checkout master`

AN ANGULAR APP IS A
TREE OF COMPONENTS

**WE MAP PROPERTIES
TO THE STATE**

WE DISPATCH **ACTIONS**
IN REACTION TO **EVENTS**





```
git checkout 01_project-structure
```

ANGULAR 2.0 ENCOURAGING AN **OOP APPROACH**

TO USE DEPENDENCY
INJECTIONS WITH REDUX

WE WRAP STUFF IN
PROVIDERS

```
import {createStore} from "redux";
import {RootReducer} from '../reducers/root';

export class Store {

  private store = createStore(rootReducer);

  get state() {
    return this.store.getState();
  }

  dispatch(action){
    this.store.dispatch(action)
  }
}
```

**WE COMBINE MULTIPLY
REDUCERS TO ONE
ROOT REDUCER**

combineReducers in action

```
import {combineReducers} from 'redux';  
  
export const RootReducer = combineReducers({  
  app: (state = 0) => state  
});
```

**WE NEED TO REGISTER
OUR STORE PROVIDER
ON THE MODULE**


```
@NgModule({  
  declarations: [AppComponent],  
  imports      : [BrowserModule, HttpClientModule],  
  providers    : [Store],  
  bootstrap    : [AppComponent]  
})
```

NOW WE CAN INJECT
IT TO OUR COMPONENT
AND GET THE STATE!

```
export class AppComponent {  
    constructor(store: Store) {  
        console.log(store.state);  
    }  
}
```



```
git checkout 02_wiring
```

LIVE DEMO

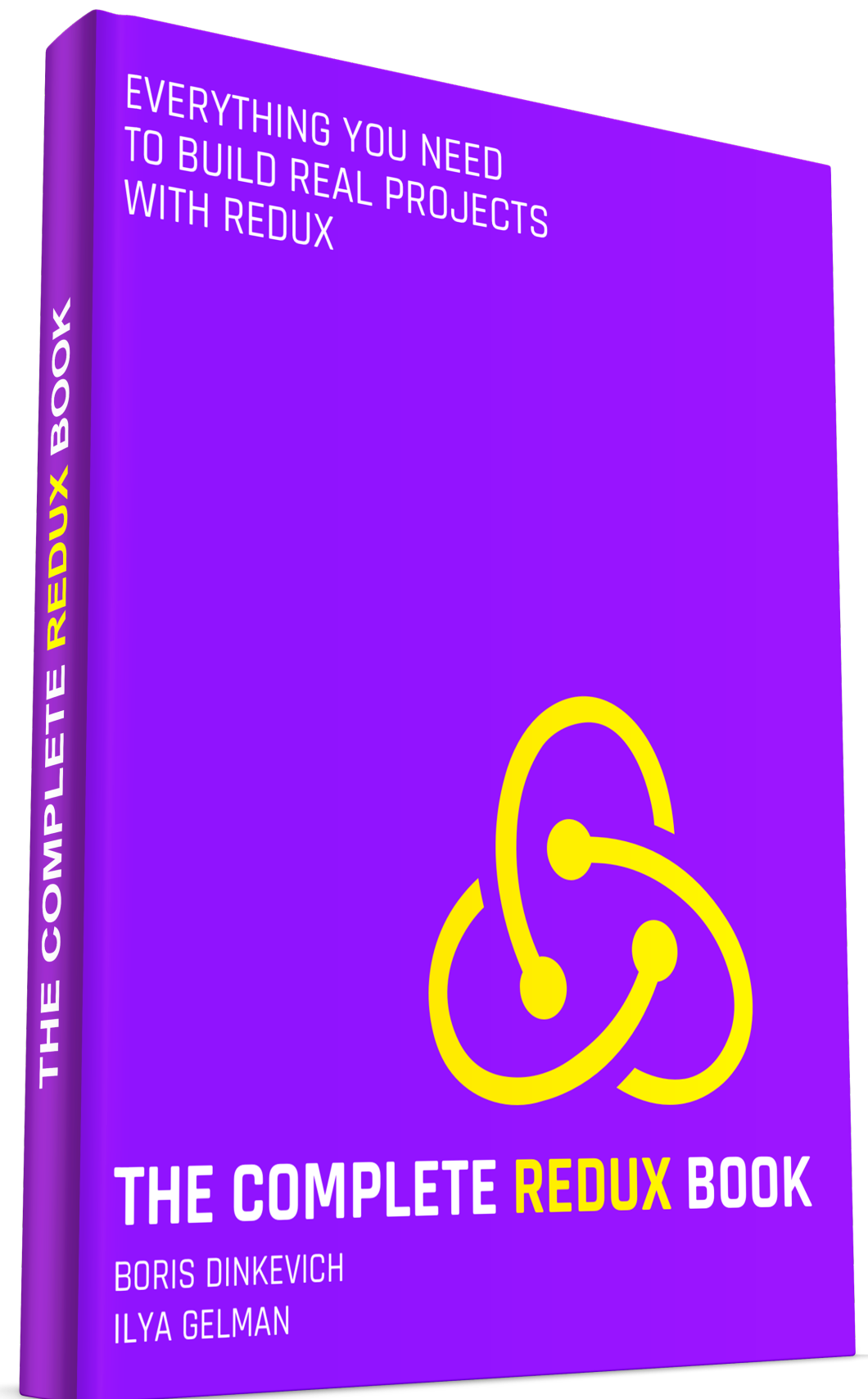
<https://github.com/nirkaufman/oscon-redux-angular-workshop>

NEXT STEPS

THE COMPLETE REDUX BOOK



<https://leanpub.com/redux-book>



RESOURCES

REDUX

<http://redux.js.org/>

<https://egghead.io/series/getting-started-with-redux>

CQRS & EVENT SOURCING

<https://msdn.microsoft.com/en-us/library/dn568103.aspx>

<https://msdn.microsoft.com/en-us/library/dn589792.aspx>

ANGULAR 2

[angular-2-change-detection-explained.html](#)

<https://github.com/ngrx/store>

<https://github.com/angular-redux/ng2-redux>

NIR KAUFMAN

Head of Angular Development @ **500Tech**



 @nirkaufman

 github.com/nirkaufman

 meetup.com/Angular-AfterHours/

keep in touch!

nir@500tech.com