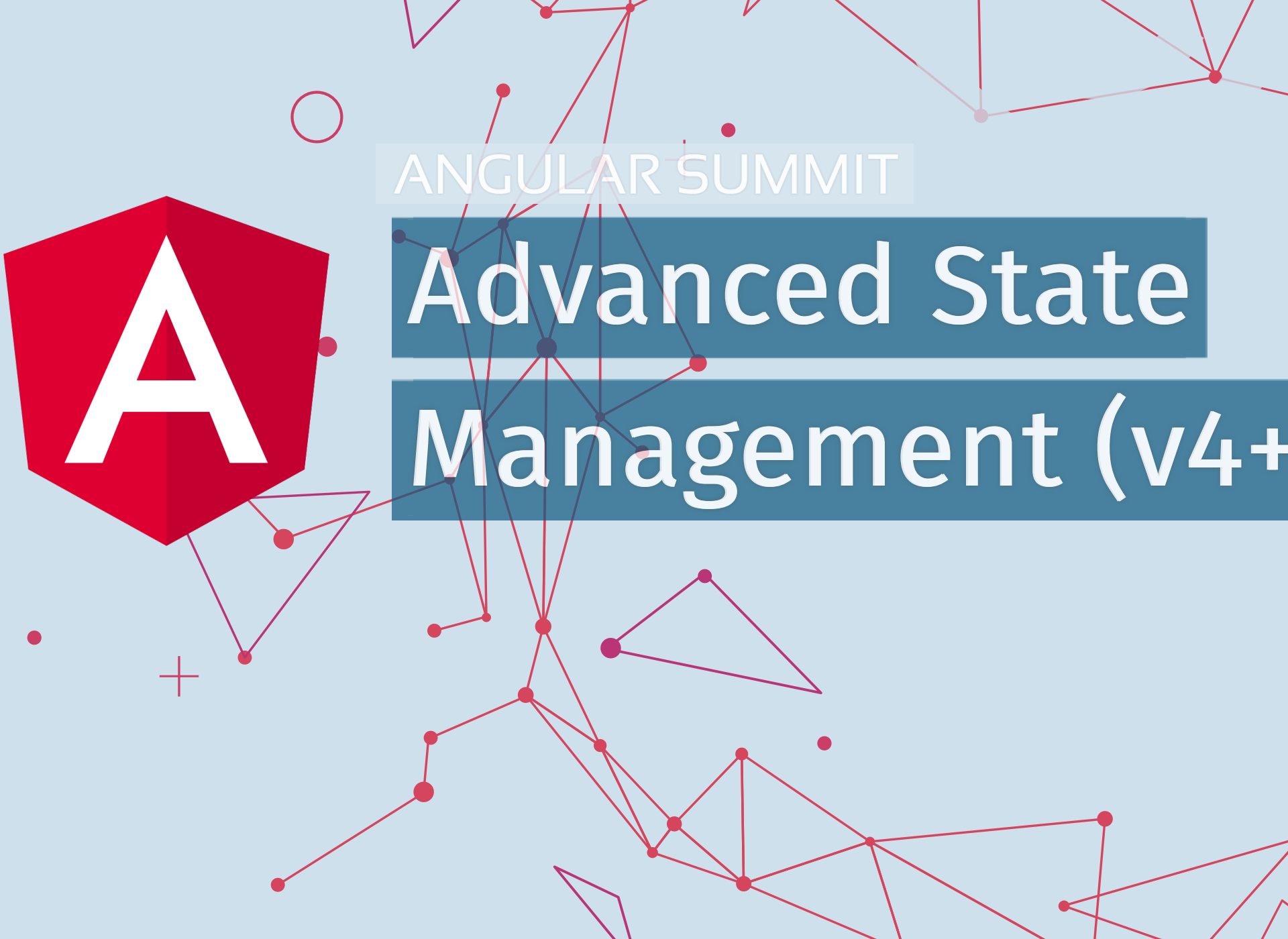

ANGULAR SUMMIT
Advanced State
Management (v4+


Google Developer Expert

Master of Ceremonies

Google Developers Experts Writer Ed

Blogger

gsans

Coding is fun | Coded something awesome today? | #Angular & #JavaScript are the new kings! M...

Aug 14 · 8 min read

GraphQL and the amazing Apollo Client

Explore an Application built using React and Angular 2

International Speaker


NG
CONF


AC
ANGULAR
CONNECT


Riga
DevDay


FullStack
The conference on JavaScript, Node and Internet of Things
13-15 JUL
2016


DON'T MISS:
droidcon
LONDON
27-28 OCT
2016
LONDON


VOXXED DAYS
BELGRADE


js.talks();


QCon
LONDON


Jfokus


INTERNET &
MOBILE
WORLD 2016


{ }
Frontend
Union
Conf


ff


JAZOON
TechDays


GDG DevFest
The Netherlands


GDG DevFest
Ukraine 2016


GDG DevFest
London 2016

Angular Trainer (v4+)

Community Leader

NG-CRUISE

— PRESENTED BY NG-CONF —

ANGULAR CRUISE

PRESENTED BY NG-CONF

As Jeff Cross' beard has stated,

we're officially jumping the shark

May 29, 2017!

SIGNUP

SPONSOR

Angular

Features

- Latest Web Standards
- Simple
- Lightning fast
- Works everywhere

new in Angular

Angular Version

```
// index.html
<my-app ng-version="4.1.0">
  <div>
    <h2>Hello Angular 4! 👋</h2>
  </div>
</my-app>
```

Semantic Versioning

X . Y . Z

MAJOR MINOR PATCH

Semantic Versioning

- v4 March 2017
- v5 Sept/Oct 2017
- v6 March 2018
- v7 Sept/Oct 2018

Angular 4

- Improved compiler (AOT)
- uses TypeScript 2.3
  - StrictNullChecks
- new bundles
  - @angular/platform-server
  - @angular/animations

Angular 4

- Add/update meta tag
- New EmailValidator
- Deprecated
  - template, OpaqueToken
  - ng-template, InjectionToken

ngIf changes

```
// Angular 2
<div *ngIf="flightInfo">{{flightInfo.name}}</div>
<div *ngIf="!flightInfo">Loading...</div>

// Angular 4
<div *ngIf="flightInfo; else noInfo">{{flight.name}}
<ng-template #noInfo>
  <div>Loading...</div>
</ng-template>
```

async pipe changes

```
// Angular 2
<div *ngIf="flightInfo$ | async">
  {{(flightInfo$ | async)?.name}}
</div>

// Angular 4
<div *ngIf="flightInfo$ | async; let flight">
  {{flight.name}}
</div>

<div *ngIf="flightInfo$ | async as flight">
  {{flight.name}}
</div>
```

demo

Redux

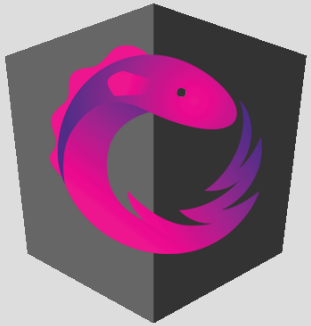
Angular Data Layer

DATA CLIENTS

GraphQL

Firebase

STATE MANAGEMENT



ngrx

Redux

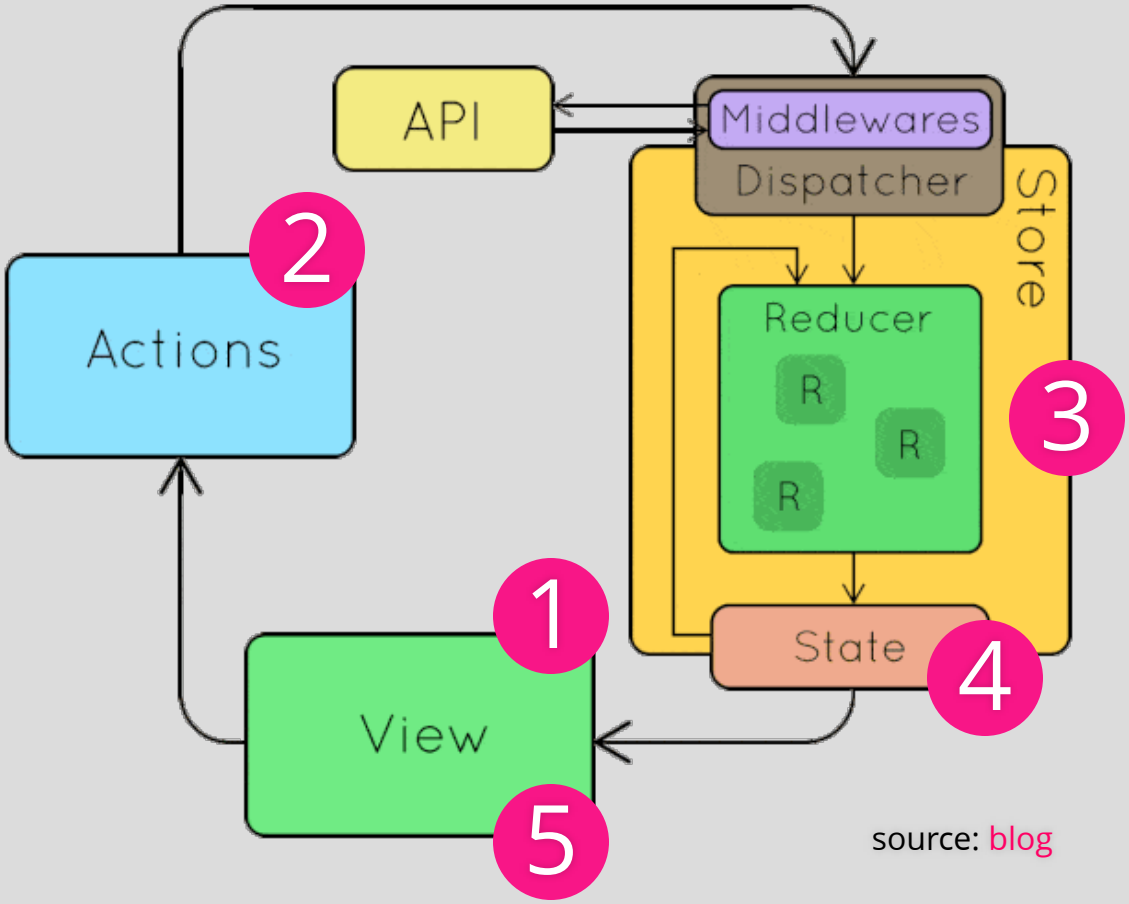
Dan Abramov

@gaearon

Main Principles

- Unidirectional data flow
- Single Immutable State
- New states are created without ***side-effects***

Unidirectional data flow



source: blog

Single Immutable State

- Helps tracking changes by reference
- Improved Performance
- Enforce by convention or using a library. Eg: Immutable.js

Immutable by Convention

- New array using ***Array Methods***
  - map, filter, slice, concat
  - Spread operator (ES6) [...arr]
- New object using ***Object.assign*** (ES6)

Using Immutable.js

```
let selectedUsers = Immutable.List([1, 2, 3]);
let user = Immutable.Map({ id: 4, username: 'Spiderman'}):

let newSelection = selectedUsers.push(4, 5, 6); // [1, 2, 3, 4, 5, 6];
let newUser = user.set('admin', true);
newUser.get('admin') // true
```

Reducers

- ***Reducers*** create new states in response to ***Actions*** applied to the current ***State***
- Reducers are ***pure functions***
- Don't produce ***side-effects***
- Composable

Example: pure function

```
// function foo(x) { return x+1; }
let foo = x => x+1;

// pure function
foo(1); // 2
foo(1); // 2
```

Example: side-effect

```
let flag = false;
let foo = x => {
  flag = !flag;  // side effect
  return flag ? x+1: 0;
}

// not pure function
foo(1); // 2
foo(1); // 0
```

Middlewares

- Sit between ***Actions*** and ***Reducers***
- Used for logging, storage and ***asynchronous operations***
- Composable

Performance

Change Detection

source: blog

Change Detection

source: blog

ngrx

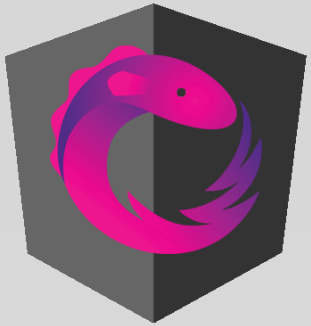
Angular Data Layer

DATA CLIENTS

GraphQL

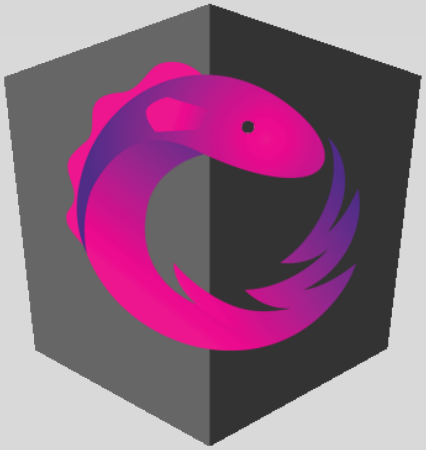
Firebase

STATE MANAGEMENT



ngrx

Redux



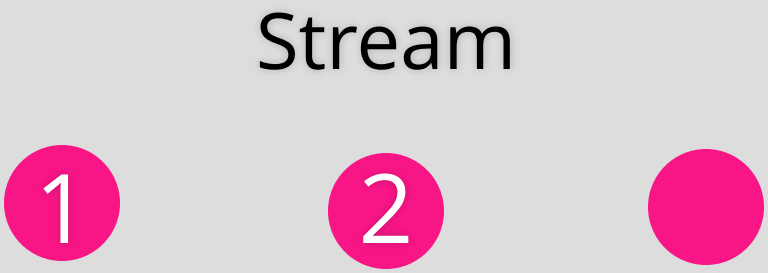
Rob Wormald

@robwormald

ngrx/store

- Re-implementation of Redux on top Angular and RxJS 5
- ngrx suite: store, effects, router, db

RxJS 5


Stream
1
2

Observable

```
//Observable constructor
let simple$ = Rx.Observable.create(observer => {
  try {
    //pushing values
    observer.next(1);
    observer.next(2);
    observer.next(3);

    //complete stream
    observer.complete();
  }
  catch(e) {
    //error handling
    observer.error(e);
  }
});
```

Subscribe

```
/*
a$ ---1---2---3|
*/

let a$ = Rx.Observable.of(1,2,3);

let subscription = a$.subscribe({
  next:  x => console.log(x),
  error: x => console.log('#'),
  complete: () => console.log('|')
});
```

Unsubscribe

```
let subscription = twits$.subscribe(
  twit => feed.push(twit),
  error => console.log(error),
  () => console.log('done')
);

setTimeout(() => subscription.unsubscribe(), 5000);
```

Solution Architecture

demo

Components Tree

source: blog

Components Tree

```
<app>
  <add-todo>
    <input><button>Add todo</button>
  </add-todo>
  <todo-list>
    <ul>
      <todo id="0" completed="false"><li>buy milk</li></todo>
    </ul>
  </todo-list>
  <filters>
    Show: <filter-link><a>All</a><filter-link> ...
  </filters>
</app>
```

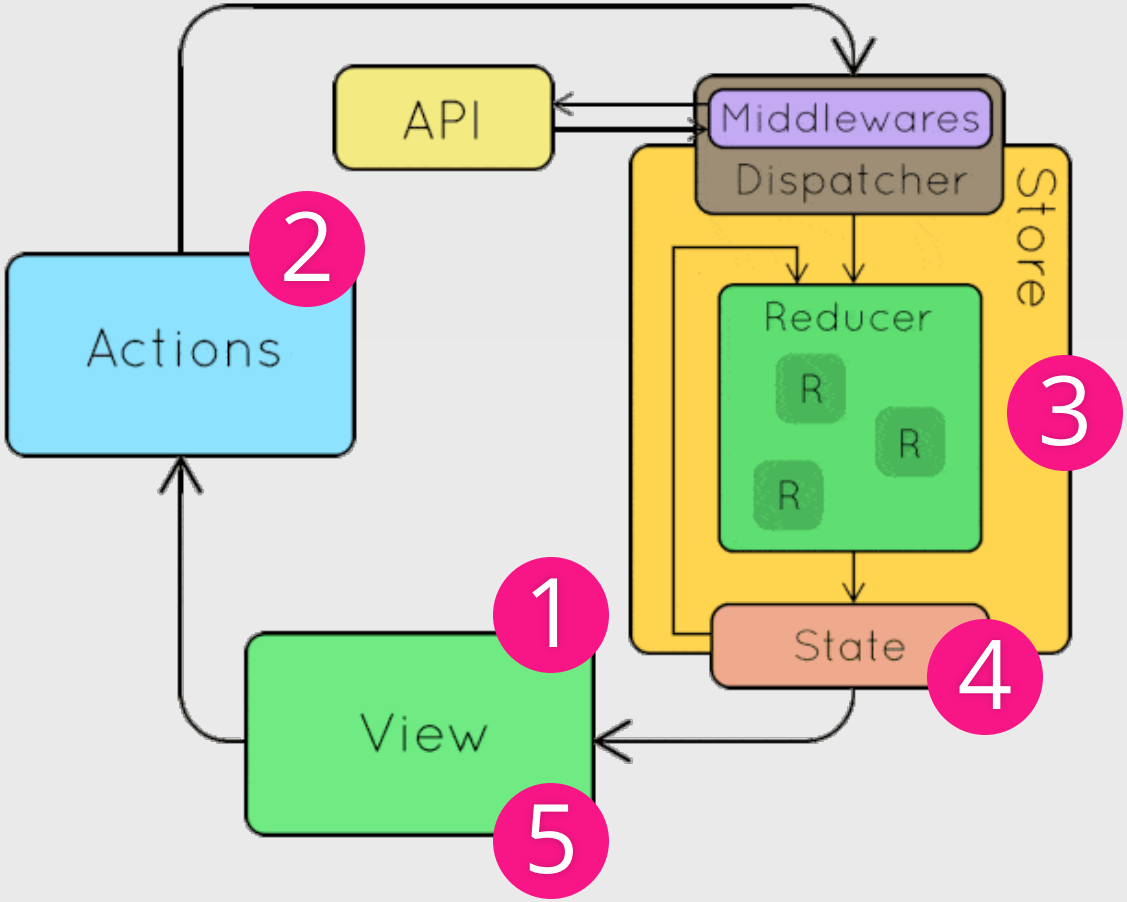
Setup

```
import { App } from './app';
import { StoreModule } from "@ngrx/store";
import { rootReducer } from './rootReducer';

@NgModule({
  imports: [
    BrowserModule,
    StoreModule.provideStore(rootReducer)
  ],
  declarations: [ App ],
  bootstrap: [ App ]
})
export class AppModule {}

platformBrowserDynamic().bootstrapModule(AppModule);
```

Adding a new Todo

1. *Component subscribes*
2. *Component dispatches*

   ADD_TODO action
3. *Store executes rootReducer*
4. *Store notifies Component*
5. *View updates*



Subscribing to the Store

```
@Component({
  template:
      `<todo *ngFor="let todo of todos | async">{{todo.text}}</todo>`
})

export class App implements OnDestroy {
  public todos: Observable<Todo>;

  constructor(
    private _store: Store<TodosState>
  ) {
    this.todos = _store.let(
      state$ => state$.select(s => s.todos);
    );
  }
}
```

ADD_TODO Action

```
// add new todo
{
  type: ADD_TODO,
  id: 1,
  text: "learn redux",
  completed: false
}
```

todos Reducer

```
const initialState = [];

const todos = (state = initialState, action:Action) => {
  switch (action.type) {
    case TodoActions.ADD_TODO:
      return state.concat({
          id: action.id,
          text: action.text,
          completed: action.completed });
    default: return state;
  }
}

// {
//  todos: [], <-- todos reducer will mutate this key
//  currentFilter: 'SHOW_ALL'
// }
```

currentFilter Reducer

```
const currentFilter = (state = 'SHOW_ALL', action: Action) => {
  switch (action.type) {
    case TodoActions.SET_CURRENT_FILTER:
      return action.filter
    default: return state;
  }
}

// {
//  todos: [],
//  currentFilter: 'SHOW_ALL' <-- filter reducer will mutate this key
// }
```

rootReducer

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

const combinedReducer = combineReducers({
  todos: todos,
  currentFilter: currentFilter
});

export rootReducer = (state, action) => combinedReducer(state, action);
```

New State

```
{
  todos: [{
    id: 1,
    text: "learn redux",
    completed: false
  }],
  currentFilter: 'SHOW_ALL'
}

// {
//  todos: [], <-- we start with no todos
//  currentFilter: 'SHOW_ALL'
// }
```

Stateless Components

Stateless Todo Component

```
// <todo id="1" completed="true">buy milk</todo>
@Component({
  inputs: ['id', 'completed'],
  template: `
    <li (click)="onTodoClick(id)"
      [style.textDecoration]="completed?'line-through':'none'">
      <ng-content></ng-content>
    </li>`,
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class Todo {
  constructor(
    private _store: Store<TodosState>,
    private todoActions:TodoActions
  ) { }

  private onTodoClick(id){
    this._store.dispatch(this.todoActions.toggleTodo(id));
  }
}
```

Redux Dev Tools

Features

- Save/Restore State
- Live Debugging
- Time travel
- Dispatch Actions

demo

Why use Redux?

Main Benefits

- Simplified Development
- Avoids complex dependencies
- Great Performance
- Developer Experience

ANGULAR SUMMIT

Thanks!