



# Using new ngrx/entity and schematics

ANGULAR SUMMIT

# Google Developer Expert



# International Speaker



*Spoken at 82 events in  
24 countries*

# Blogger



## Google Developers Experts



Writer ▾



Applause from Houssein Djirdeh, Chau Nguyen, and 273 others

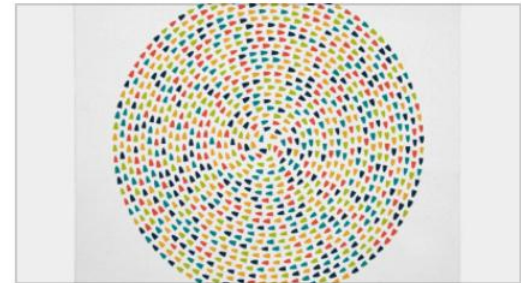
### Angular—Testing Guide (v4+)

Nine easy-to-follow examples using TestBed,



Applause from Seyed Mostafa Meshkati, Coskun Deniz, and 141 others

### Angular — Supercharge your Router transitions using new animation features (v4.3+)



Applause from Aaron Frost, Angular Academy, and 61 others

### Angular — Applying Motion principles to a listing



# Trainer











# Angular Academy

[About](#)

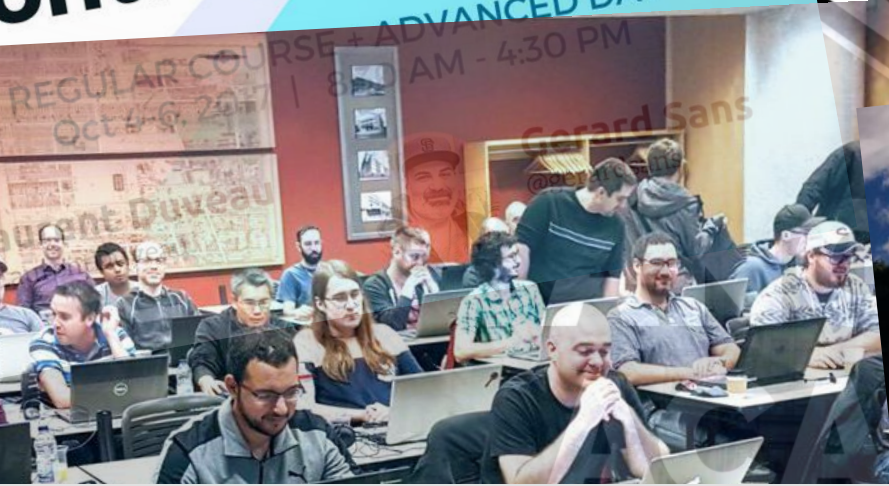
[Course outline](#)

[Register](#)

[Discounts](#)

**Angular Academy  
Montreal (English)**

REGULAR COURSE + ADVANCED DAY  
Oct 4-6, 2017 | 8:30 AM - 4:30 PM



Learn Angular and TypeScript Now!



# *Redux*

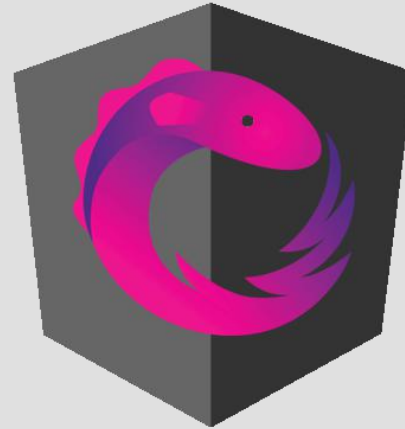
---



# State Management



Redux



ngrx



***Dan Abramov***

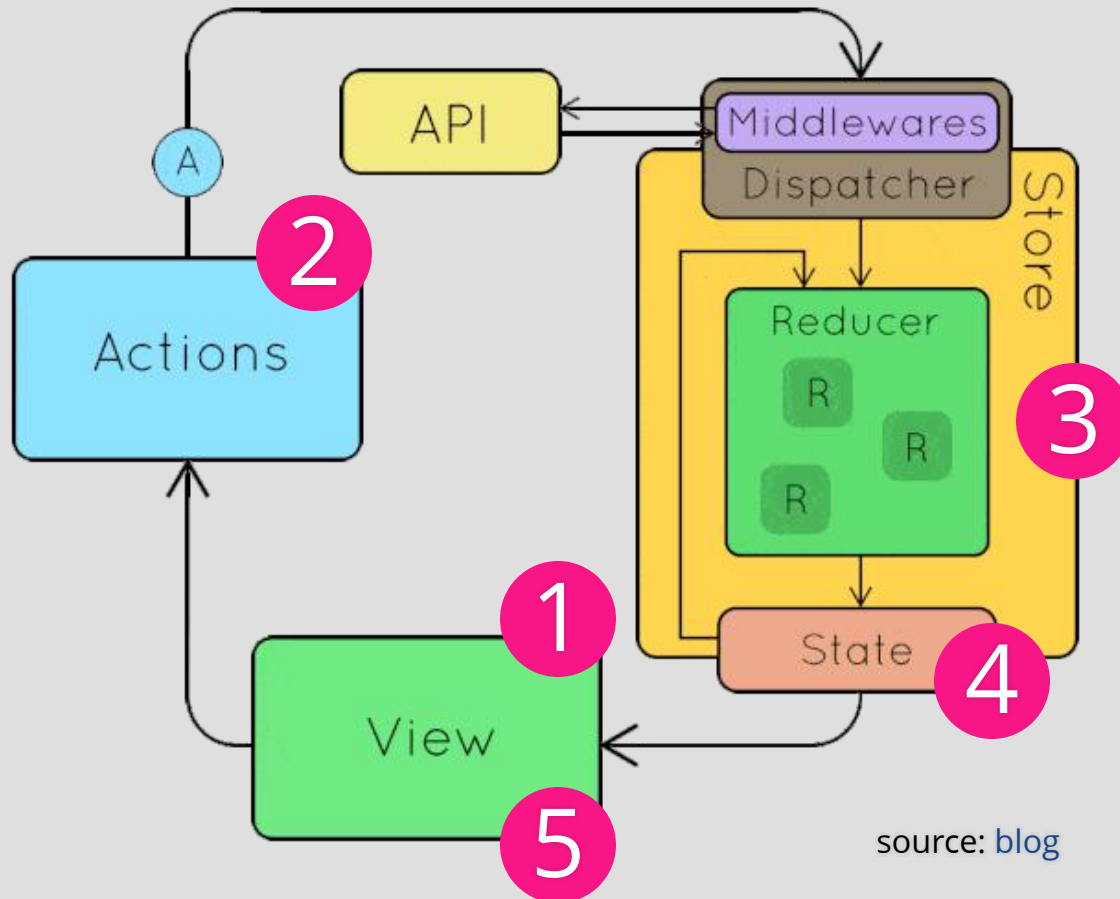
@gaelaron



# Redux Principles

- *Unidirectional data flow*
- *Single Store*
- *No side-effects*

# Unidirectional data flow





*Developer  
Experience*

# extension.remotedev.io

## Redux DevTools Extension

on gitter PRs welcome backers 13 sponsors 3

The image shows a composite view of the Redux DevTools Extension interface and a web browser. On the left, the Redux DevTools interface is visible, featuring a 'Chart' tab and an 'Autoselect' dropdown. A diagram shows a state tree with a root node 'todos' branching into three child nodes: 'todos[0]', 'todos[1]', and 'todos[2]'. Below this, a list of tasks is displayed: 'What needs to be done?' (checked), 'Add extension's enhancer' (unchecked), 'Install extension' (unchecked), and 'Use Redux' (checked). A progress bar at the bottom indicates '2 items left' and includes filters for 'All', 'Active', and 'Completed'. On the right, a web browser window displays the URL 'zalmoxisus.github.io/redux-devtools-extension/examples/todomvc/'. The browser shows a tutorial for the 'Redux TodoMVC example' with a list of tasks: 'What needs to be done?' (checked), 'Add extension's enhancer' (unchecked), 'Install extension' (unchecked), and 'Use Redux' (checked). The browser also displays a 'Log monitor' tab with an 'Autoselect instances' dropdown. The log shows two entries for 'ADD\_TODO': the first entry shows an action with text 'Install extension' and a state with 'todos' containing 2 items; the second entry shows an action with text 'Install extension' and a state with 'todos' containing 2 items. The browser's bottom bar includes tabs for 'Elements', 'Console', 'Sources', 'Network', 'Timeline', 'Profiles', 'Resources', 'Security', 'Redux', and a search icon.



# Features

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

***Todo App***





# Todo App powered by @ngrx/entity

Add a new todo

Learn Du

All

Active

Completed

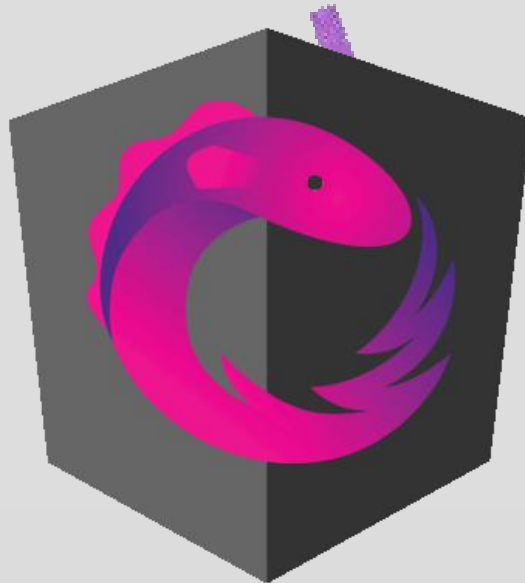
☐ Try stroopwafe 🍪 🗑️

☒ Learn @ngrx/entity 🥗 🗑️

Made in Amsterdam by @gerardsans

***demo***

***ngrx***



v5



# What's new in v5?

- *@ngrx/schematics (@angular-devkit)*
- *'Pipeable' select operator (lettable)*
- *Support custom createSelector*
- *RxJS 5.5*
- *UpsertOne/Many*



Ben Lesh

Following

RxJS 5+ Lead, Software Engineer at Google, RxWorkshop.com. Views are my own  
Jan 10

Remember to use patch imports to avoid blowing up your bundle size.

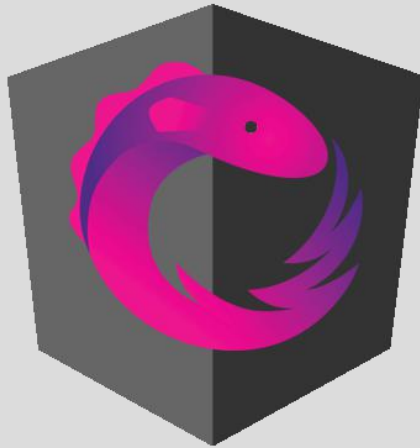
Top JavaScript Libraries & Tech to Learn in 2018  
Eric Elliott

👏 20K 💬 67

This is actually inaccurate now, as of rxjs 5.5 people should try to use “pipeable” (aka “lettable”) operators which is a functional programming approach. They are more tree-shakeable, and will ultimately reduce the overall output bundle size substantially. It also becomes much easier to create your own operators, as they are just higher-order functions.

**Rob Wormald**

@robwormald



# @ngrx/platform

- *Suite including*
  - *@ngrx/store\**
  - *@ngrx/effects*
  - *@ngrx/router-store*
  - *@ngrx/store-devtools\**
  - *@ngrx/entity\**
  - *@ngrx/schematics\**



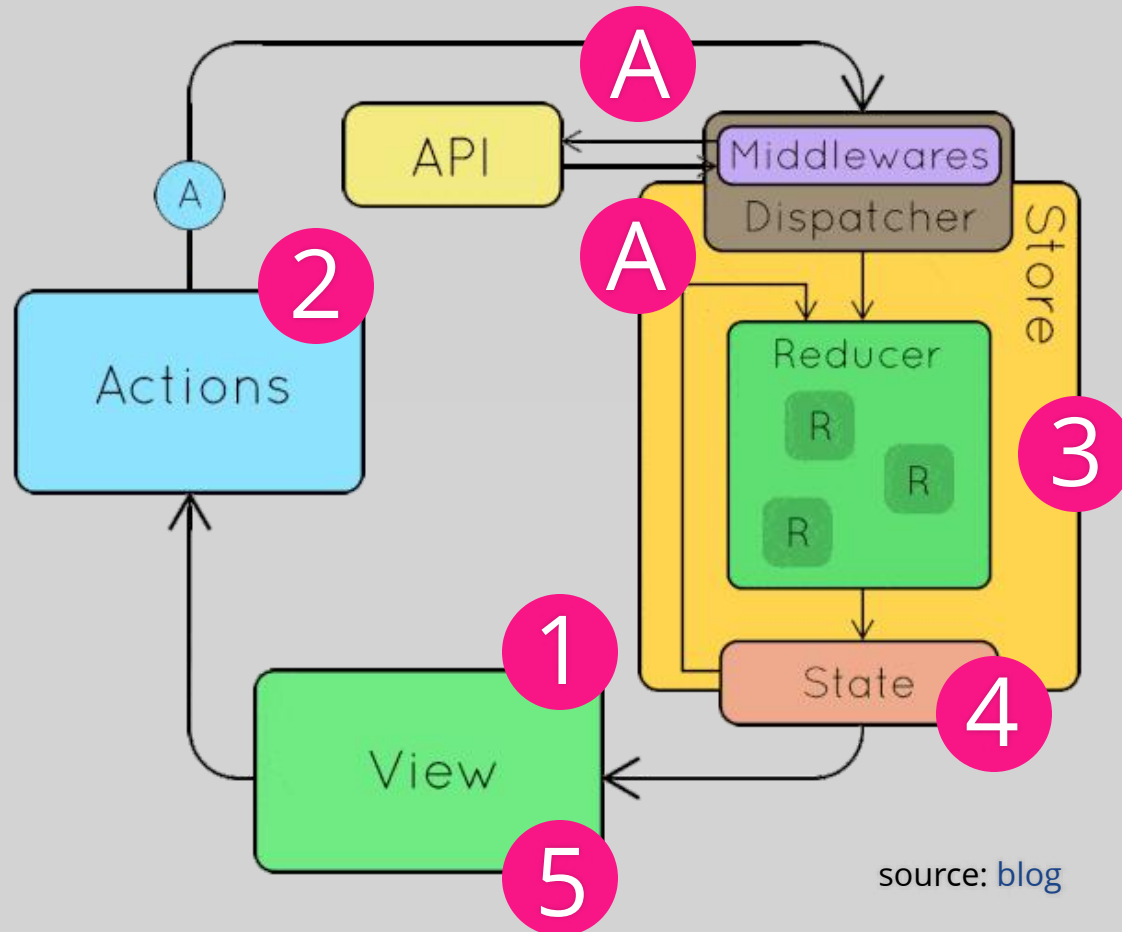
# ngrx/store

- *Re-implementation of Redux on top Angular and RxJS 5.*
- *Colocation Support*
  - *StoreModule.forRoot()*
  - *StoreModule.forFeature()*
- *Payload Type checking*

# ngrx/effects

- *Asynchronous Actions Middleware*
- *Executes side-effects*
- *Colocation Support*
  - *EffectsModule.forRoot()*
  - *EffectsModule.forFeature()*

# Asynchronous Actions



*ngrx/entity*



# ngrx/entity

- *Library to manage Lists*
- *High performance*
  - *EntityState<T>*
  - *Ids lookup and Entities Map*
- *Reducer Helper (EntityAdapter)*
  - *getInitialState*
  - *Add/update/remove: All, One, Many*
  - *getSelectors: Ids, Entities, All, Total*

# ngrx/schematics

- *Scaffolding templates for ngrx*
- *Provides commands for*
  - *Setting initial Store and Effects*
  - *Actions, Reducers, Entities, Features*
  - *Containers + Store injected*
- *Automates creation + registration*

# ngrx/schematics commands

```
> ng new ngrx-entity-todo
```














```
> npm install @ngrx/schematics --save-dev
```

```
> npm install @ngrx/{store, effects, entity, store-devtools} --save
```

```
> ng generate store State --root --module app.module.ts --collection @ngrx
```

```
> ng generate effect App --root --module app.module.ts --collection @ngrx
```

```
> ng generate entity Todo -m app.module.ts --collection @ngrx/schematics
```

- ▲  src
  - ▲  app
    - ▲  reducers
      - ▲  currentFilter
        -  currentFilter.actions.ts
        -  currentFilter.model.ts
        -  currentFilter.reducer.ts
      - ▲  todo
        -  todo.actions.ts
        -  todo.model.ts
        -  todo.reducer.spec.ts
        -  todo.reducer.ts
      -  index.ts



# Filter Actions

```
// src/app/reducers/currentFilter/currentFilter.actions.ts
import { Action } from '@ngrx/store';

export enum CurrentFilterActionTypes {
  SetCurrentFilter = '[Filter] Set current filter'
}

export class SetCurrentFilter implements Action {
  readonly type = CurrentFilterActionTypes.SetCurrentFilter;

  constructor(public payload: { filter: string }) {}
}

export type CurrentFilterActions = SetCurrentFilter;
```

# Filter Reducer

```
// src/app/reducers/currentFilter/currentFilter.reducer.ts
export const initialState: string = 'SHOW_ALL';

export function reducer(
  state = initialState,
  action: CurrentFilterActions): string
{
  switch (action.type) {
    case CurrentFilterActionTypes.SetCurrentFilter:
      return action.payload.filter
    default:
      return state;
  }
}
```

# Todos Actions

```
// src/app/reducers/todo/todo.reducer.ts
export enum ActionTypes {
  AddTodo = '[Todo] Add Todo', UpdateTodo = '[Todo] Update Todo', ...
}

let currentId = 1;
export class AddTodo implements Action {
  readonly type = ActionTypes.AddTodo;
  constructor(public payload: { todo: Todo }) {
    payload.todo.id = currentId++;
  }
}

export type TodoActions = LoadTodos | AddTodo | UpsertTodo | AddTodos
| UpsertTodos | UpdateTodo | UpdateTodos | DeleteTodo | DeleteTodos |
```

# Todos Model

```
// src/app/reducers/todo/todo.model.ts
import { EntityState } from '@ngrx/entity';

export interface Todo {
  id: number;
  completed: boolean;
  text: string;
}

export interface Todos {
  todos: EntityState<Todo>;
}
```

# Composing Reducers

```
import * as todos from './todo/todo.reducer';
import * as currentFilter from './currentFilter/currentFilter.reducer';

export interface Filter {
  currentFilter: string;
}

export interface Todos {
  todos: EntityState<Todo>;
}

export interface TodosState extends Todos, Filter { }

export const reducers: ActionReducerMap<TodosState> = {
  todos: todos.reducer,
  currentFilter: currentFilter.reducer
};
```

# New Pipeable Selectors

```
// ngrx v4
// export const getTodos = state$ => state$.select(s => s.todos);
// export const getCurrentFilter = state$ => state$.select('currentFi

export const getTodos = state$ => state$.pipe(
  select('todos'),
  map(todoEntity.selectAll)
);

export const getCurrentFilter = state$ => state$.pipe(
  select('currentFilter')
);
```



# AppModule Setup

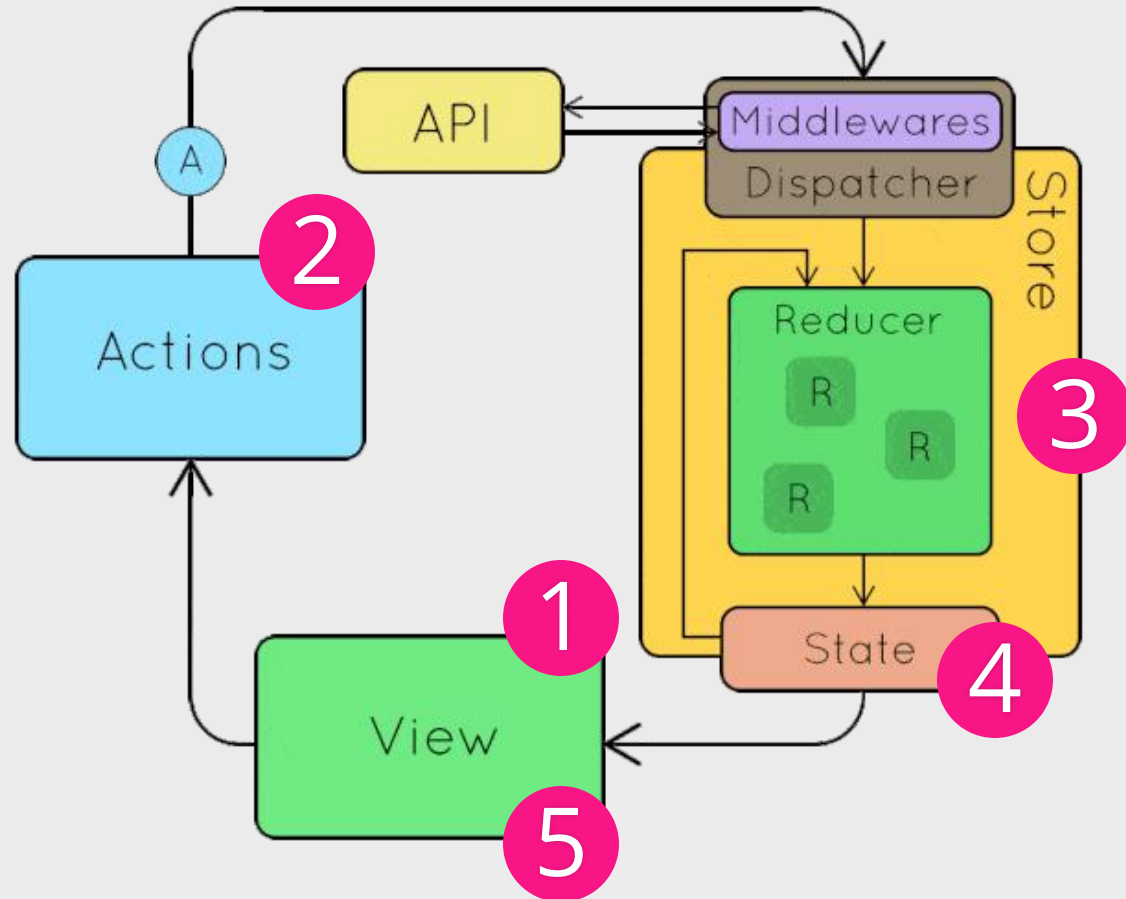
```
import { StoreModule } from "@ngrx/store";
import { StoreDevtoolsModule } from '@ngrx/store-devtools';
import { AppComponent } from './app.component';
import { reducers, metaReducers } from './reducers';

@NgModule({
  imports: [
    BrowserModule,
    StoreModule.forRoot(reducers, { metaReducers }),
    !environment.production ? StoreDevtoolsModule.instrument() : [],
  ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule {}
```

***Let's use it!***

# Adding a new Todo

1. *Component subscribes*
2. *Component dispatches*  
`TodoActionTypes.AddTodo`
3. *Store executes rootReducer*
4. *Store notifies Component*
5. *View updates*



# Subscribing to the Store

```
@Component({
  template: `
    <todo *ngFor="let todo of todos | async | visibleTodos:currentFilter"
      {{todo.text}}
    </todo>`
})
export class App implements OnDestroy {
  constructor(private _store: Store<TodosState>) { }

  public ngOnInit() {
    this.todos = this._store.let(getTodos);
    this._store.let(getCurrentFilter).subscribe((filter: string) => {
      this.currentFilter = filter;
    })
  }
}
```

# app.component.ts

```
@Component({
  template: `<input #todo (keydown.enter)="addTodo(todo1)">`
})
export class AppComponent {
  private addTodo(input) {
    if (input.value.length === 0) return
    const todo: Todo = {
      id: null,
      text: input.value,
      completed: false,
    }
    this._store.dispatch(new todoActions.AddTodo({ todo }))
    input.value = ''
  }
}
```

# TodoActionTypes.AddTodo

```
// add new todo
{
  type: '[Todo] Add Todo',
  payload: {
    todo: {
      id: 1,
      text: 'Learn Dutch',
      completed: false
    }
  }
}
```



# todos Reducer

```
export const adapter: EntityAdapter<Todo> = createEntityAdapter<Todo>
export let initialState: State = adapter.getInitialState();

export function reducer(state = initialState, action: TodoActions): S
  switch (action.type) {
    case TodoActionTypes.AddTodo: {
      return adapter.addOne(action.payload.todo, state);
    }
    ...
    default: { return state; }
  }
}

export const {
  selectIds, selectEntities, selectAll, selectTotal
} = adapter.getSelectors();
```

# State

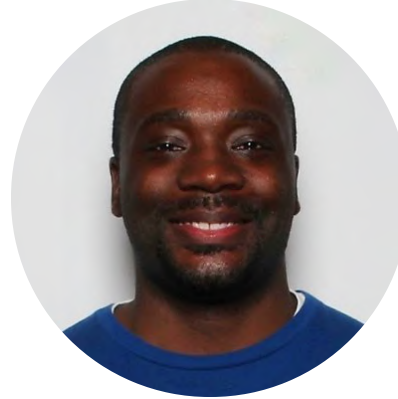
```
{
  todos: {
    ids: [ 1 ],
    entities: {
      '1': {
        id: 1,
        text: 'Learn Dutch',
        completed: false
      }
    }
  },
  currentFilter: 'SHOW_ALL'
}
```

***demo***

***More***



**Mike Ryan**  
@MikeRyanDev



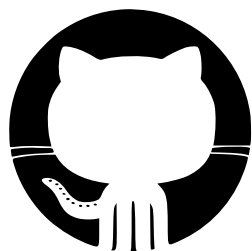
**Brandon Roberts**  
@brandontroberts



**Rob Wormald**  
@robwormald



**Todd Motto**  
@toddmotto



***gsans/ngrx-entity-todo-app***



