ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Professional NgRx

6 - NgRx on Steroids

@ngrx/entity

@ngrx/entity

- ngrx as maintainer
- Helper methods for states with "entity" character
- Provides performant data structure
- EntityAdapter as main element
- Ignores side-effects, e.g. backend communication
- Partial updates of entities (not only the whole object)

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Creating the EntityAdapter

- createEntityAdapter
  - Factory method
  - Define id property
  - Define comparator function for sorting

Creating the EntityAdapter

```
const adapter = createEntityAdapter<Holiday>({});
```

Entity's type

Creating the EntityAdapter

```
const adapter = createEntityAdapter<Holiday>({

  selectId: (holiday) => holiday.id


});
```

Entity's type

Defining the id (optional)

Creating the EntityAdapter

```
const adapter = createEntityAdapter<Holiday>({
  selectId: (holiday) => holiday.id,
  sortComparer: (holiday1, holiday2) =>
    holiday1.title.localeCompare(holiday2.title),
});
```

Entity's type

Defining the id (optional)

Sorting function (optional), disable with false

Defining the EntityState

```
export interface HolidaysState extends EntityState<Holiday> {}
```

Defining the EntityState

```
export interface HolidaysState extends EntityState<Holiday> {

  loadStatus: LoadStatus;

  favouriteIds: number[];

}
```

Additional properties

Internal structure

```
export interface HolidaysState {
  entities: { [id: string]: Holiday | undefined };
  ids: string[] | number[];
  loadStatus: LoadStatus;
  favouriteIds: number[];
}
```

Fast lookup

For Sorting

Initializing the EntityState

```
const initialState: HolidaysState = adapter.getInitialState();
```

Initializing the EntityState

```
const initialState: HolidaysState = adapter.getInitialState({
  favouriteIds: [],
  loadStatus: "not loaded",
});
```

Initializing the EntityState

```
const initialState: HolidaysState = adapter.getInitialState({
  favouriteIds: [],
  loadStatus: "not loaded",
});
```

Reducer

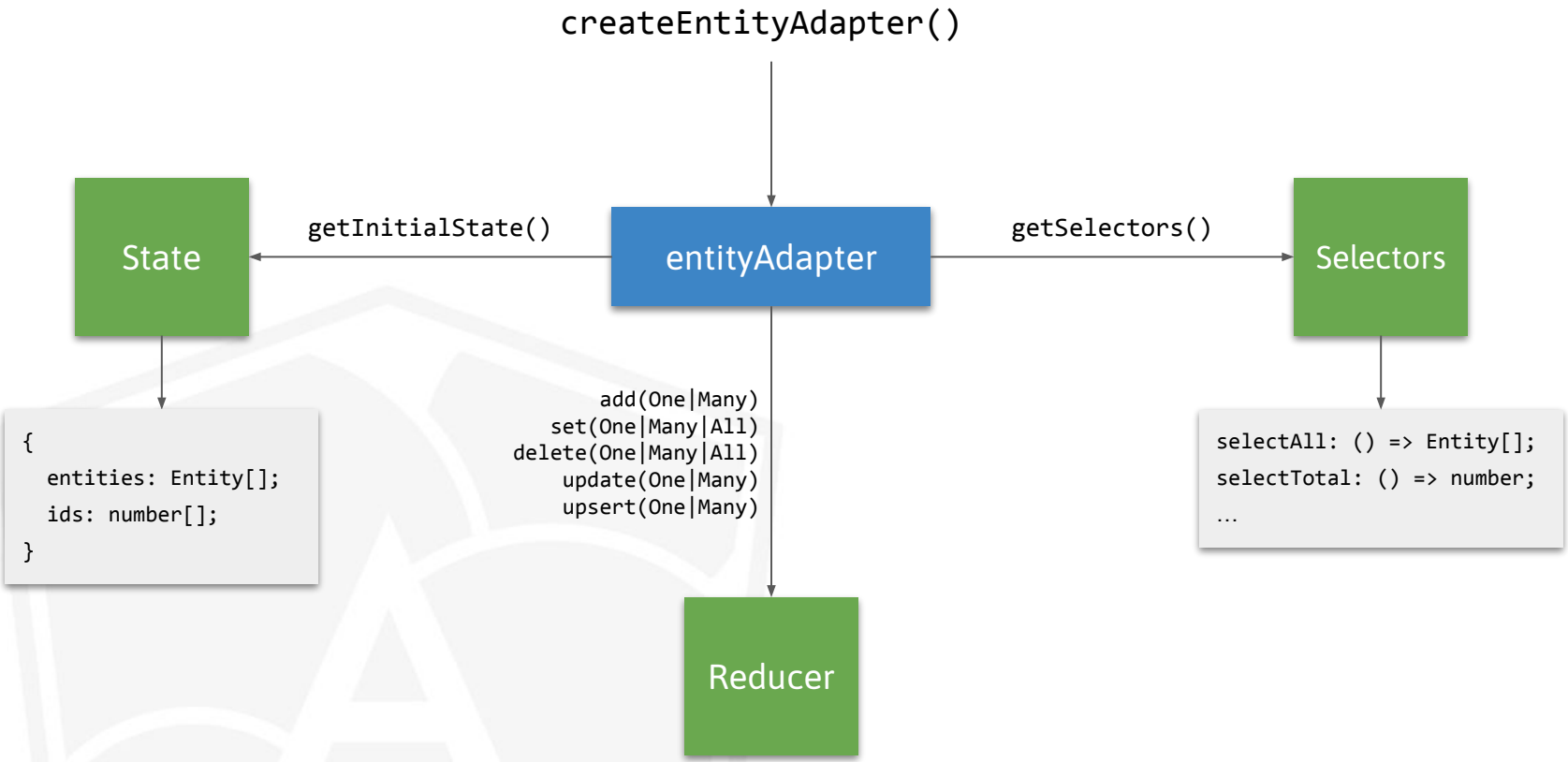
```
createReducer<HolidaysState>(
  initialState,
  on(loaded, (state, { holidays }) => adapter.setAll(holidays, state))
);
```

Reducer

```
createReducer<HolidaysState>(
  initialState,
  on(loaded, (state, { holidays }) => ({
    ...adapter.setAll(holidays, state),
    loadStatus: "loaded",
  }))
);
```

EntityAdapter's methods

- Initialize the state
- Select entities
- Crud functions in reducer
  - add[One|Many]
  - set[One|Many|All]
  - delete[All|One|Many]
  - update[One|Many]
  - upsert[One|Many]



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

@ngrx/data

@ngrx/data

Pros

- Generic solution including API communication
- Suited for "mass production"
- Provides extension points for customization
- Significantly reduces boilerplate code

Cons

- Design for a very specific use case
- Customization very hard to learn
- Limited flexibility
- Naming conventions not always comprehensible
- Issues with lazy loaded modules (overriding services)

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Features

- Loading Status
- Change Tracker
- Optimistic updates (undo is built-in)
- Filtering

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Demo

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Elf

- Opinionated state management
- Features can be enabled as you go
  - And helper functions
- Not enforcing strictness like NgRx
  - Store could be available publicly
  - Isn't NgRx 😉

Elf - enable Features

- Differentiation between Entity, UIProperties, and generic
- selectedId
- undo/redo
- localStore
- immer integration
- loadStatus
- caching
- pagination

Dynamic NgRx