ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Professional NgRx

7 - Beyond NgRx

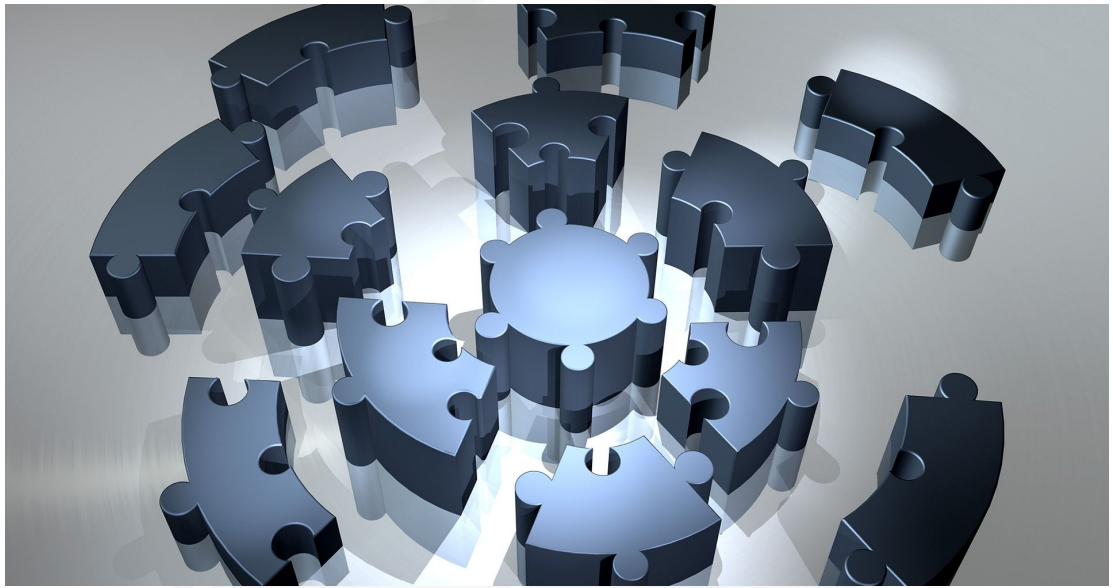
Elf

- 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

@ngrx/ component-store



Main facts

- (Little) sibling to @ngrx/store
- Local / component state management
- State vanishes with component (by default)
- Push-based / Reactive
- All logic included in single service

Use cases

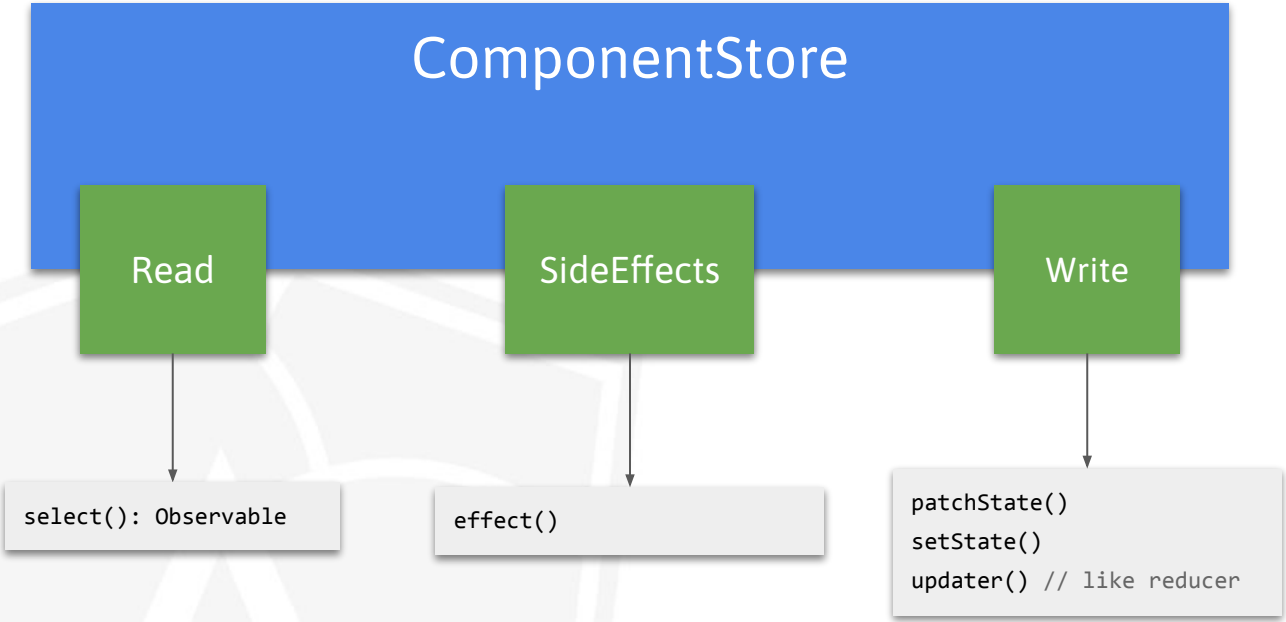
- Multiple instances of same components
- Complicated, local state logic
- Different static logic
- ~Independent, simple global state

Defining the state

```
export interface HolidaysState {
  holidays: Holiday[];
  favouriteIds: number[];
}

@Injectable()
export class HolidaysStore extends ComponentStore<HolidaysState> {
  constructor(private httpClient: HttpClient, private config: Configuration) {
    super({ holidays: [], favouriteIds: [] });
  }
}
```

API



Selectors

```
export interface HolidaysState {
  holidays: Holiday[];
  favouriteIds: number[];
}

@Injectable()
export class HolidaysStore extends ComponentStore<HolidaysState> {
  constructor(private httpClient: HttpClient, private config: Configuration) {
    super({ holidays: [], favouriteIds: [] });
  }

  readonly holidays$ = this.select(({ holidays, favouriteIds }) =>
    holidays.map((holiday) => ({
      ...holiday,
      isFavourite: favouriteIds.includes(holiday.id),
    }))
  );
}
```

Actions

```
@Injectable()
export class HolidaysStore extends ComponentStore<HolidaysState> {
  // ...

  addFavourite(holidayId: number) {
    this.patchState((state) => ({
      favouriteIds: [...state.favouriteIds, holidayId],
    }));
  }

  removeFavourite(holidayId: number) {
    this.patchState((state) => ({
      favouriteIds: state.favouriteIds.filter((id) => id !== holidayId),
    }));
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Effects

```
@Injectable()
export class HolidaysStore extends ComponentStore<HolidaysState> {
  // ...

  readonly load = this.effect((i$: Observable<void>) => {
    return i$.pipe(
      switchMap(() =>
        this.httpClient.get<Holiday[]>(this.#baseUrl).pipe(
          tapResponse((holidays) => {
            const finalHolidays = holidays.map((holiday) => ({
              ...holiday,
              imageUrl: `${this.config.baseUrl}${holiday.imageUrl}`,
            }));
            this.#setHolidays(finalHolidays);
          }, console.error)
        )
      )
    );
  });

  #setHolidays = (holidays: Holiday[]) => this.patchState({ holidays });
}
```

Using the ComponentStore

```
@Component({
  templateUrl: './holidays.component.html',
  standalone: true,
  imports: [...],
  providers: [HolidaysStore],
})
export class HolidaysComponent {
  holidays$ = this.holidaysStore.holidays$;

  constructor(private holidaysStore: HolidaysStore) {
    this.holidaysStore.load();
  }

  addFavourite(id: number) {
    this.holidaysStore.addFavourite(id);
  }

  removeFavourite(id: number) {
    this.holidaysStore.removeFavourite(id);
  }
}
```

Dynamic NgRx