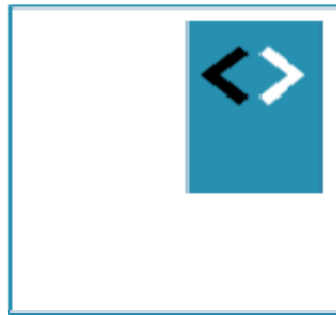# Angular Advanced Module @ngrx/effects

Peter Kassenaar –
info@kassenaar.com

What is @ngrx/effects for?

- `@ngrx/effects` provides an API to model event sources as actions. Effects:

  - *Listen* for actions dispatched from `@ngrx/store`

  - *Isolate side effects* from components

  - Provide *new sources* of actions to reduce state based on *external interactions* such as network requests, web socket messages and time-based events.
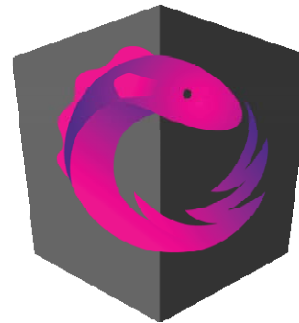
For Instance (when to use ngrx/effects):
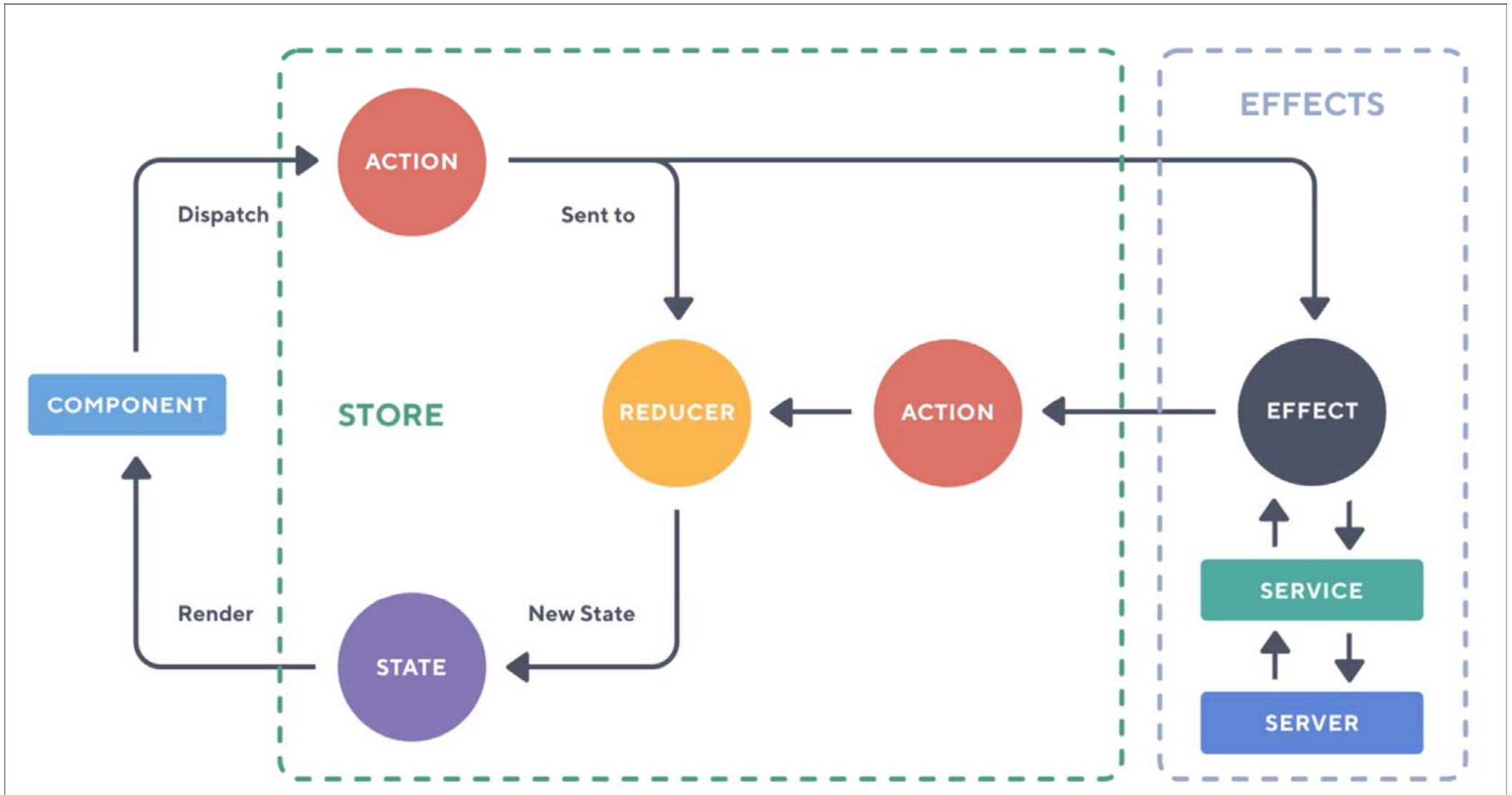
- Talking to a RESTFul server == a side effect with implications on the store.

    - Hence the name, `ngrx/effects`

    - So, it is a side effects model for `ngrx/store`

- Listen for `ngrx/store` actions

- Isolate effects from components and reducers

- Communicate outside of Angular, notify the store when changes are complete

    - Perfect for Asynchrounous operations

Effects flow
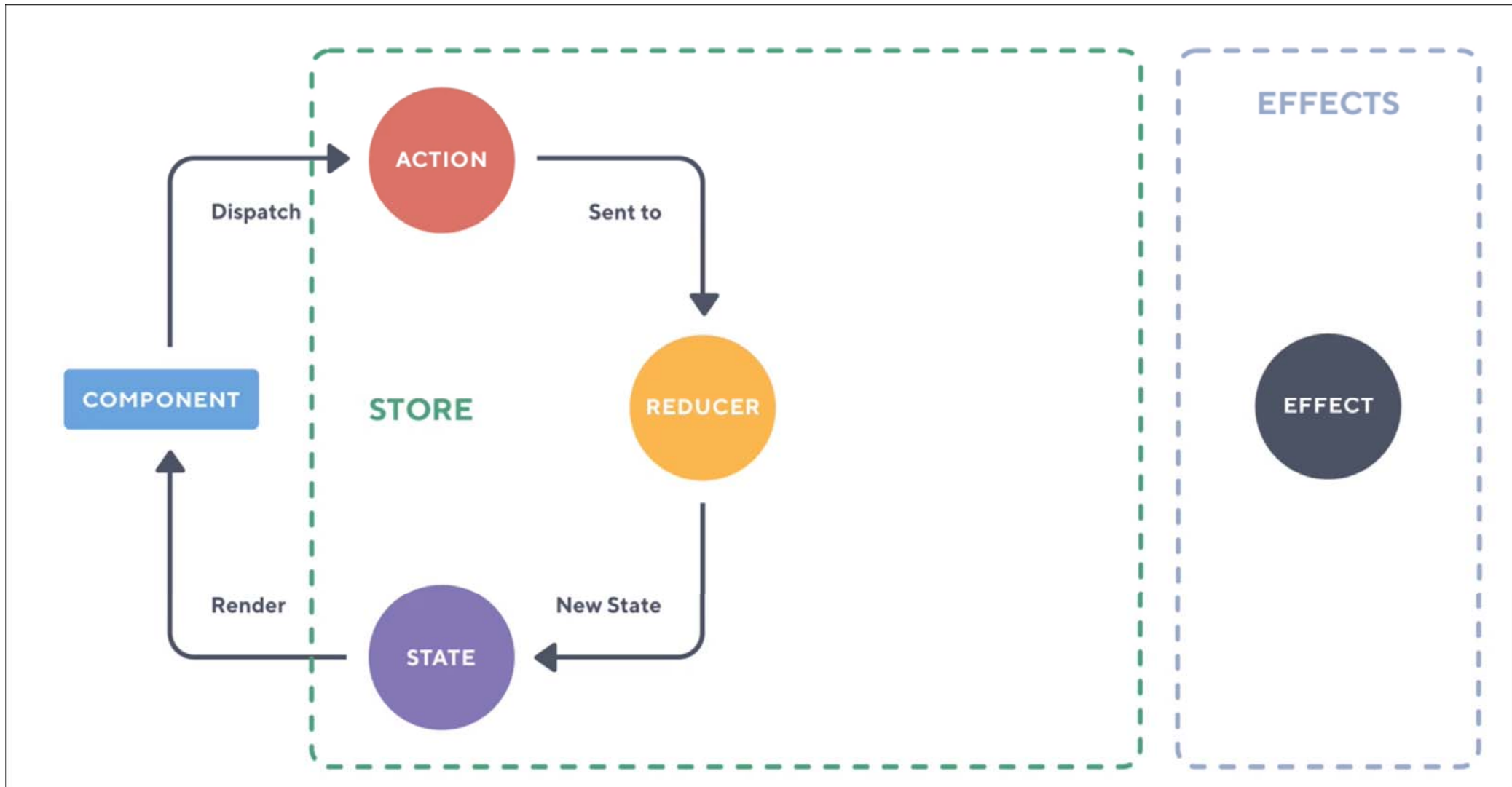
Effects flow

So, an effect...

1. Listens to store `Actions`.

   - YOU define which action an effect listens to

2. Performs an action (such as talking to a webserver)

3. Dispatches the result to the reducer as a new

   `Action`.

4. The reducer in turn updates the store

Adding @ngrx/effects

```
npm install @ngrx/effects --save
```

OR

```
yarn add @ngrx/effects
```

Adding effects to the module

- Import `EffectsModule` from `@ngrx/effects`

- Use `EffectsModule.forRoot()` for root module

- Use `EffectsModule.forFeature()` for feature module

```
// Effects
import {EffectsModule} from '@ngrx/effects';

// exported effects from general index file
import { effects } from './effects/index';

@NgModule({
   …
   imports     : [
      …
      EffectsModule.forRoot(effects),   // array of effects
   ],
   …
})
export class AppModule {
}
```

Export all effects from index.ts file

Not necessary – but seen often: export all files from a folder from an index.ts-file

```
// effects/index.ts, export everything from this folder
// TODO: add extra/new effects to this file to auto-export them
import {CitiesEffects} from './citiesEffects';

export const effects: any[] = [CitiesEffects];

export * from './citiesEffects';
```

Example effect

```typescript
import {Injectable} from '@angular/core';
import {Actions, Effect} from '@ngrx/effects';
import {CityService} from '../services/city.service';

import * as cityActions from '../actions/cities.actions'
import {map, switchMap, catchError} from 'rxjs/operators';

@Injectable()
export class CitiesEffects {
  constructor(private action$: Actions,
          private cityService: CityService) {
  }

  @Effect()
  loadCities$ = this.action$.ofType(cityActions.LOAD_CITIES_EFFECT) // 1. Receive action from dis
    .pipe
      switchMap(() => {
        return this.cityService.loadCitiesViaEffect()        // 2. Talk to API
          .pipe(
            map(cities => new cityActions.LoadCities(cities)), // 3. Dispatch new action
            catchError(err => {
              console.log(err);
              throw err.json();                    // 4. Optional (but best practice): catch errors
            })
          )
      })
    )
```
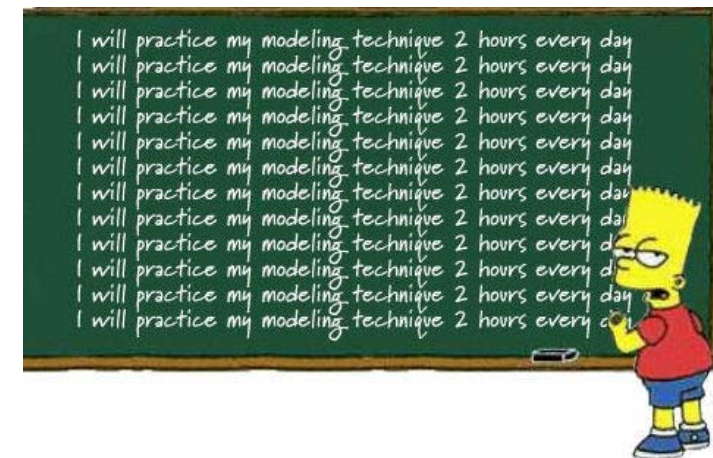
Difference with service-based approach

- The Service *does not* subscribe.

- Instead it just fetches content from an URL and returns it to the effect

- ...which in turn dispatches a new Action to update the store.

```
loadCitiesViaEffect() {
    return this.http.get(BASE_URL).pipe(
        tap(res => console.log('just received', res))
    ).pipe(
        catchError(err => {
            console.log(err);
            throw err.json();
        })
    )
}
```

Workshop

- Start from `/230-ngrx-store-effect`

  - Use `cities.json`, or another `.json`-file you create yourself

  - Implement the `@Effect()`'s for Adding and
    Removing a city

- OR: Create a blank project:

  - Add `@ngrx/effects` to the project and to the module

  - Create an `@Effect()` to load an
    external resource (your `.json`-file)

  - Notify the store once the resource
    is loaded and update the UI.

More info on @ngrx/effects

- https://github.com/ngrx/platform/blob/master/docs/effects/README.md

- https://www.youtube.com/watch?v=f97ICOaekNU

- https://angular-2-training-book.rangle.io/handout/state-management/ngrx/side_effects.html

Angular ngrx Redux Quick Start Tutorial