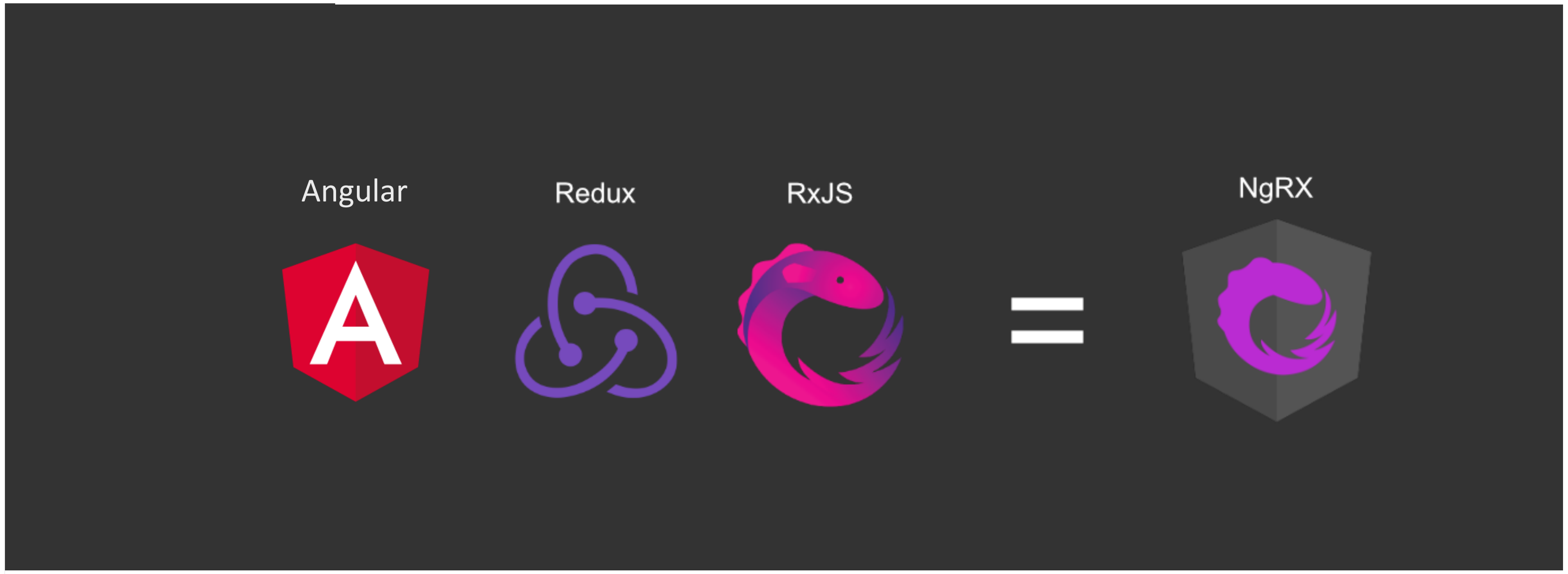
State Management

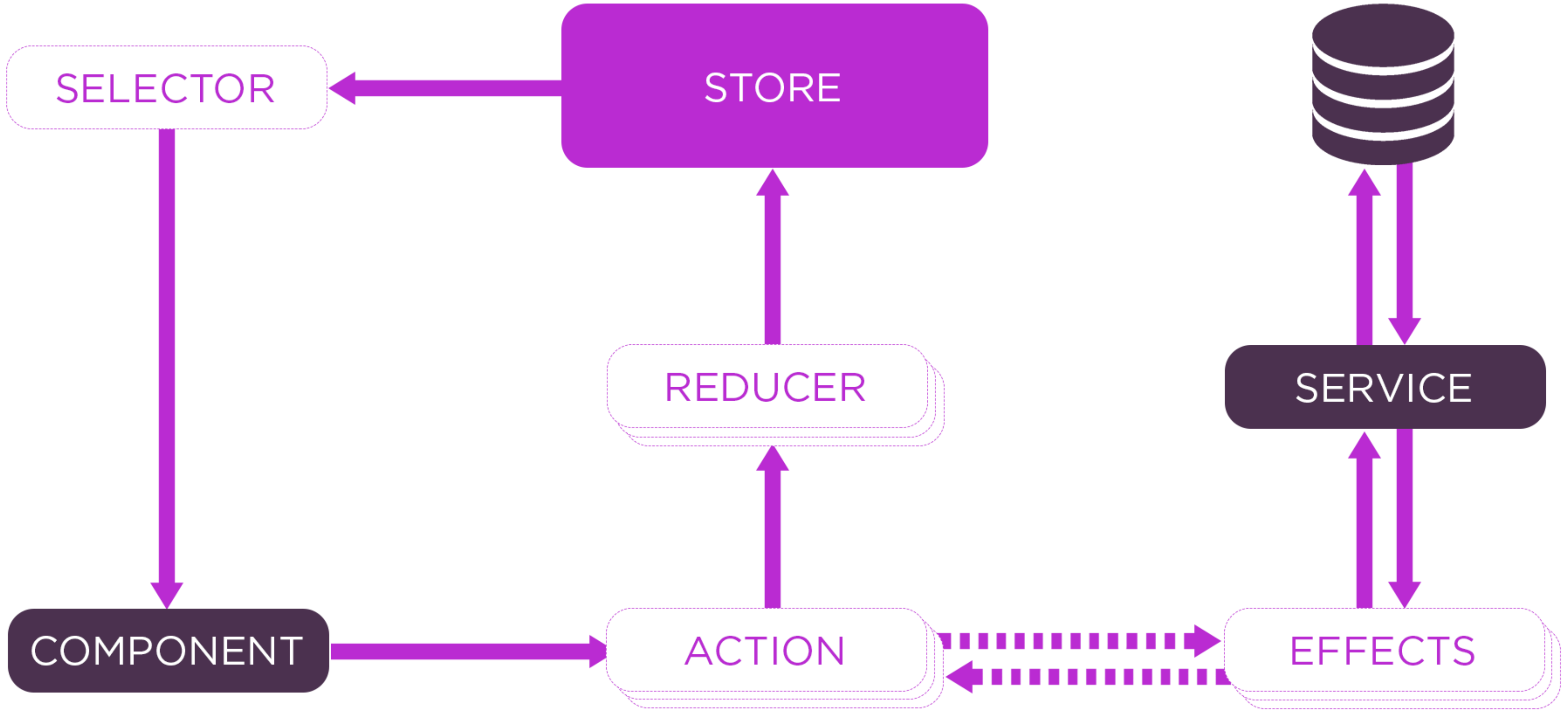
IIHT

NGRX

- A group of Angular libraries for reactive extensions.
- **Ngrx**/Store implements the Redux pattern
  - Using the well-known RxJS observables



NGRX STATE MANAGEMENT LIFECYCLE



Installation

- https://ngrx.io/guide/store/install
  - npm install @ngrx/store -S

Getting Started

- $ ng new angular-redux-rxjs
- $ cd src/app
- $ mkdir components
- $ cd components
- $ ng g c counter
- $ cd ..
- $ touch counter.actions.ts
- $ touch counter.reducer.ts

Getting Started

counter.component.html

```html
<button (click)="increment()">Increment</button>

<div>Current Count : {{count$ | async}}</div>

<button (click)="decrement()">Decrement</button>

<button (click)="reset()">Reset</button>

<br/>
<input #n (keyup)="reset(n.value)" type="number" />
```

Getting Started

counter.component.ts

```
import { Component } from '@angular/core';
import { select, Store } from '@ngrx/store';
import { Observable } from 'rxjs';
import {increment , decrement , reset} from ’../../counter.actions’
…
export class CounterComponent {

  count$ : Observable<number>;
  constructor(private store: Store<{count : number}>) {
    this.count$ = store.pipe(select('count'));
  }
  increment(){this.store.dispatch(increment({payload: {incrementNumber : 1}}))}
  decrement(){this.store.dispatch(decrement())}
  // reset(){this.store.dispatch(reset())}
  reset(num = 0) {
    this.store.dispatch(reset({
      payload: { num: num }
    }))

}
```

Getting Started

counter.actions.ts

```
import {createAction ,props} from '@ngrx/store'


export const increment = createAction(
                'increment’,
                props<{payload: {incrementNumber : any;}
      }>());

export const decrement = createAction('decrement');

// export const reset = createAction('reset');
export const reset = createAction('reset', props<{payload: {num:number}}>());
```

Getting Started

counter.reducer.ts

```
import {createReducer, on} from '@ngrx/store';

import {increment, decrement, reset} from './counter.actions'

export const initialState = 0;

const _counterReducer = createReducer(
    initialState,
    on(increment, (state ,action)=> state + action.payload.incrementNumber),
    on(decrement, (state)=> state -1 ),
    // on(reset, (state)=> 0)
    on(reset, (state, action) => action.payload.num)

)
export function counterReducer(state, action){
    return _counterReducer(state, action)
}
```

Getting Started

app.module.ts

```
import { StoreModule } from '@ngrx/store';
import { counterReducer } from './counter.reducer';
import { CounterComponent } from './components/counter/counter.component';
…

@NgModule({
  …
  imports: [
    StoreModule.forRoot({ count: counterReducer }),
    BrowserModule,
    …
  ],
  …
})
export class AppModule { }
```

Getting Started

app.component.html

```
<app-counter></app-counter>
```