signalStore

Async Programming

rxMethods

rxMethod

A method that generates another method

TS app.component.ts

```ts
import { rxMethod } from '@ngrx/signals/rxjs-interop';

export class AppComponent {
  readonly foo = rxMethod<number>(…);
}
```

TS app.component.ts

```ts
doSomething() {
  this.foo(42);
}
```

rxMethod

- Accepts a function as parameter
- The function accepts an observable as parameter
- The function needs to return an observable
- Similar to **effects** in NgRX

TS app.component.ts

```ts
import { rxMethod } from '@ngrx/signals/rxjs-interop';

export class AppComponent {

  readonly foo = rxMethod<number>(a$ => {
    return a$.pipe(...)
  });

}
```

rxMethod

- Parameter invoked during invocation of rxMethod
- Has **injection context**!
- Therefore – rxMethod must be called in Injection Context!

TS app.component.ts

```ts
import { rxMethod } from '@ngrx/signals/rxjs-interop';
export class AppComponent {

  readonly foo = rxMethod<number>(a$ => {
    const a = inject(Service);    // works!
    return a$.pipe(...)
  });
}
```

rxMethod

- The input observable is triggered whenever the method is called
- The output observable is subscribed to.

TS app.component.ts

```ts
import { rxMethod } from '@ngrx/signals/rxjs-interop';
export class AppComponent {

  readonly foo = rxMethod<number>(input$ => {
    const output$ = input$.pipe(...);
    return output$;
  });
}
```

rxMethod

- Output observable should be created from input observable
- The method logic should be implemented using rxjs operators
- Side-effects should be done using the “tap” operator

TS app.component.ts

```
import { rxMethod } from '@ngrx/signals/rxjs-interop’;
import { filter, tap, map } from ’rxjs’;
export class AppComponent {
  readonly foo = rxMethod<number>(input$ => {
    const output$ = input$.pipe(
          filter(…)
          tap(….),
          map(…)
          );
    return output$;
  });
}
```

rxMethod

3 Ways to invoke

TS app.component.ts

```ts
import { rxMethod } from '@ngrx/signals/rxjs-interop';
export class AppComponent {
  readonly a = 42;
  readonly b$ = of(42);
  readonly c = signal(42);

  readonly foo = rxMethod<number>(...);
}
```

TS app.component.ts

```ts
doSomething() {
  this.foo(this.a);          // with value
  this.foo(this.b$);         // with observable
  this.foo(this.c);          // with signal
}
```