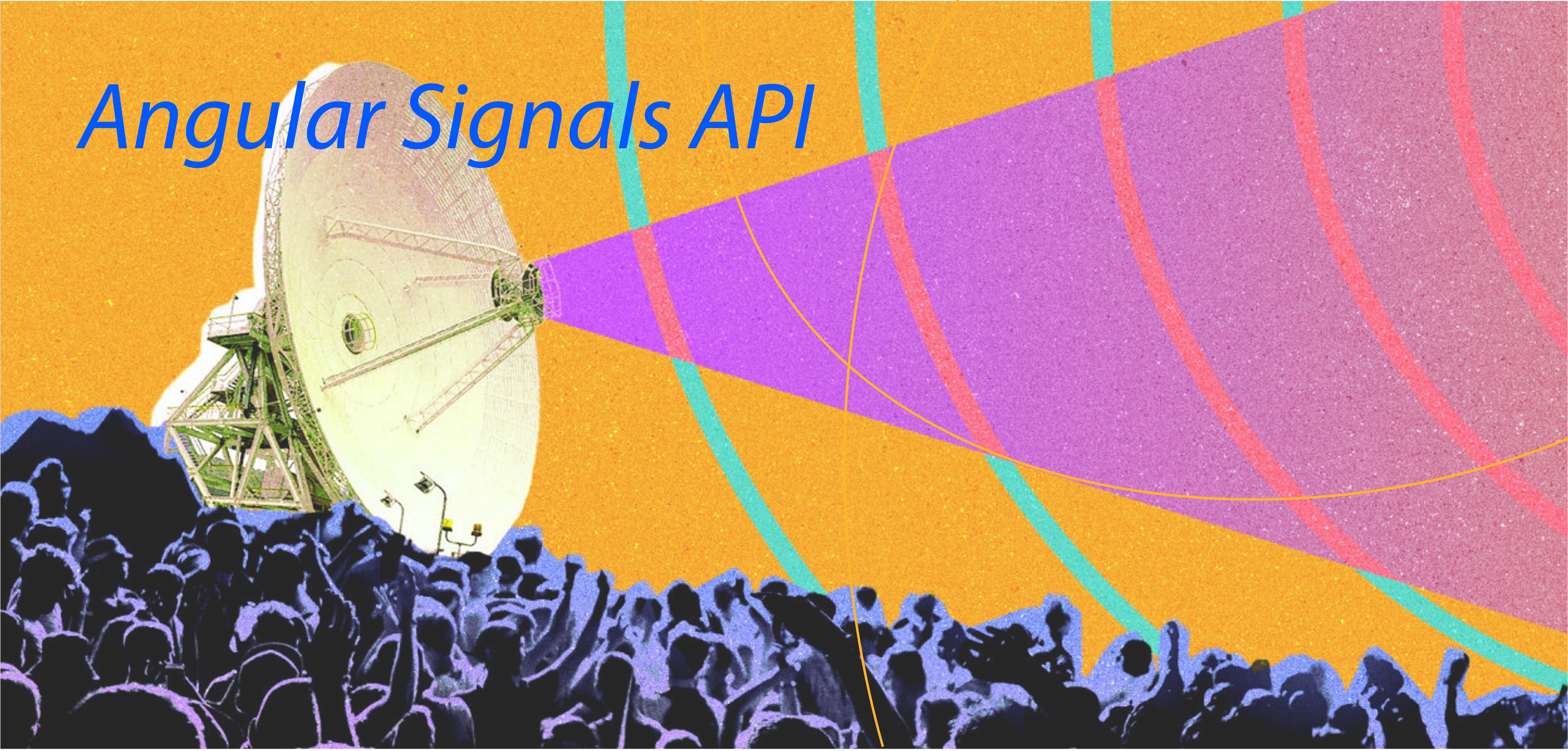
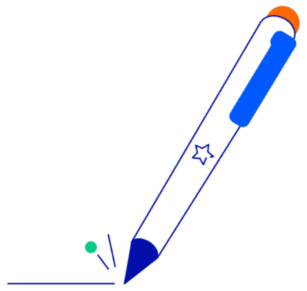
Angular Signals API



Rabobank

What are Signals?

A signal is **a primitive that holds a value that could be read, update or notify consumers when that value changes**. Signals can contain any value, from simple primitives to complex data structures.



Writable signals

Let update their values directly

Computed signals

Derives its value from other signals

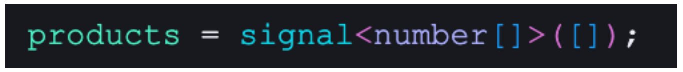
Effects

Runs operations when signals values are changes

Rabobank

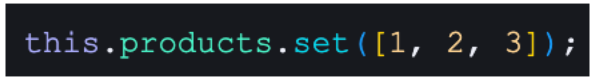
Writable signals

- Could be created by calling the **signal()** function that always requires an an initial value



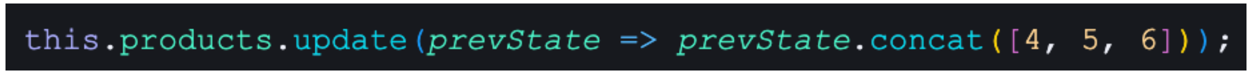
```
products = signal<number[]>([]);
```

- The new value could be set by calling the ***.set()*** method



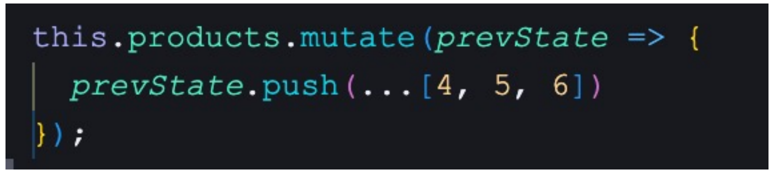
```
this.products.set([1, 2, 3]);
```

- The value could be updated based on the previous one by calling the ***.update()*** method



```
this.products.update(prevState => prevState.concat([4, 5, 6]));
```

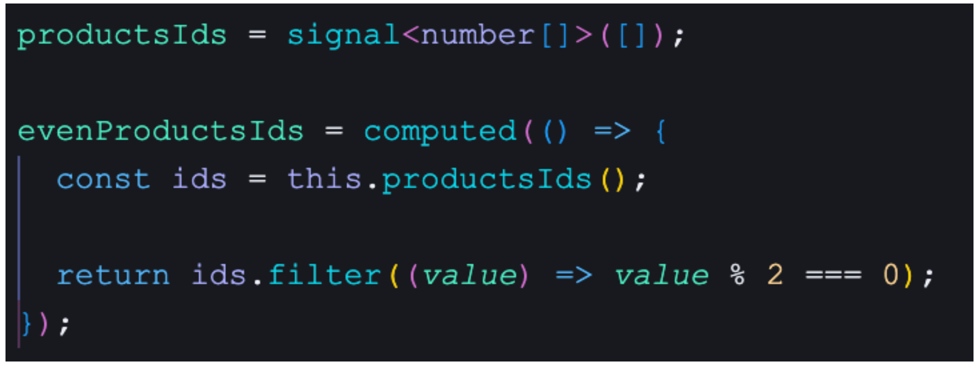
- The value could be mutated based on the current value by calling the ***.mutate()*** method



```
this.products.mutate(prevState => {
  prevState.push(...[4, 5, 6])
});
```

Computed signals

- Could be created by calling the **computed()** function that awaits a callback as argument
- Depends on the value from a writable signal
- Gets called/notified as soon as the signal it depends on gets updated



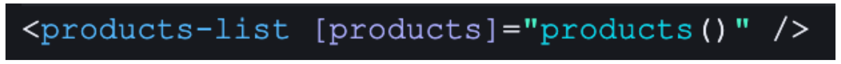
```
productsIds = signal<number[]>([]);

evenProductsIds = computed(() => {
  const ids = this.productsIds();

  return ids.filter((value) => value % 2 === 0);
});
```

Rabobank

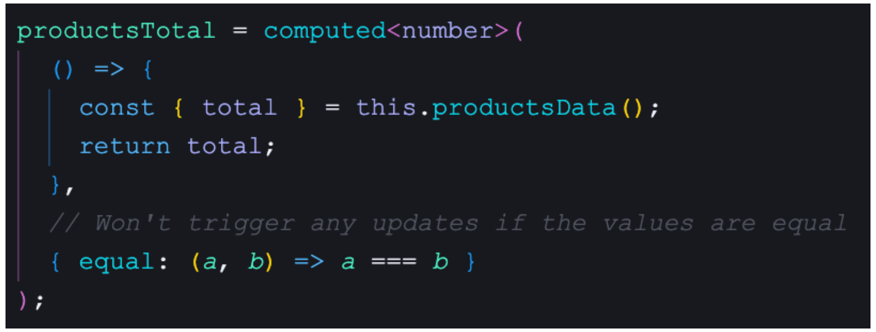
Writable & Computed signals

- Could be read in the template by calling them

- Could have a predicate function as equality option. If the new and previous values are equal none of the consumers will be notified.



```
<products-list [products]="products()" />
```

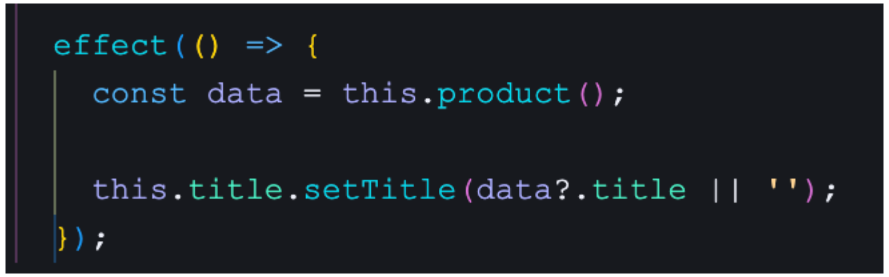


```
productsTotal = computed<number>(
  () => {
    const { total } = this.productsData();
    return total;
  },
  // Won't trigger any updates if the values are equal
  { equal: (a, b) => a === b }
);
```

Rabobank

Effects

- Could be created by calling the **effect()** function
- Runs every time the tracked signal notifies on its update
- Runs at least one time



```
effect(() => {
  const data = this.product();

  this.title.setTitle(data?.title || '');
});
```

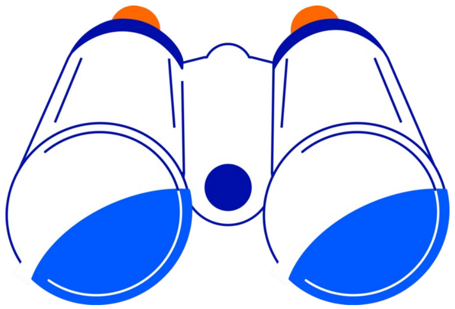
Rabobank

RxJS Interop

Angular provides additional package @angular/core/rxjs-interop which provides useful utilities to integrate Angular Signals with RxJS Observables.

toSignal

Creates *a signal* which tracks the value of an Observable



toObservable

Creates *an Observable* which tracks the value of a signa

Rabobank

toSignal

- Creates a signal which tracks the value of an Observable.

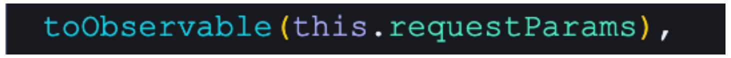
```
title = toSignal(this.route.title);
```

- Behaves similar to async pipe in templates

- Automatically unsubscribes from the given Observable once the component gets destroyed

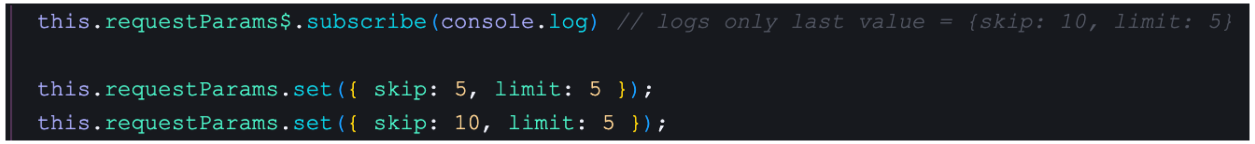
toObservable

- Creates an Observable which tracks the value of a signal



```
toObservable(this.requestParams),
```

- Observable which depends on a signal emits value once the signal is settled



```
this.requestParams$.subscribe(console.log) // logs only last value = {skip: 10, limit: 5}

this.requestParams.set({ skip: 5, limit: 5 });
this.requestParams.set({ skip: 10, limit: 5 });
```

Rabobank

Demo

Rabobank

Useful links

- Angular Signals Implementation
- Signals documentation
- RxJS Interop for Signals
- Angular Architects signals overview
- Demo repository

Rabobank

Questions

Contacts

Team: The Jugglers

Email: anton.cherepov@rabobank.nl

Rabobank