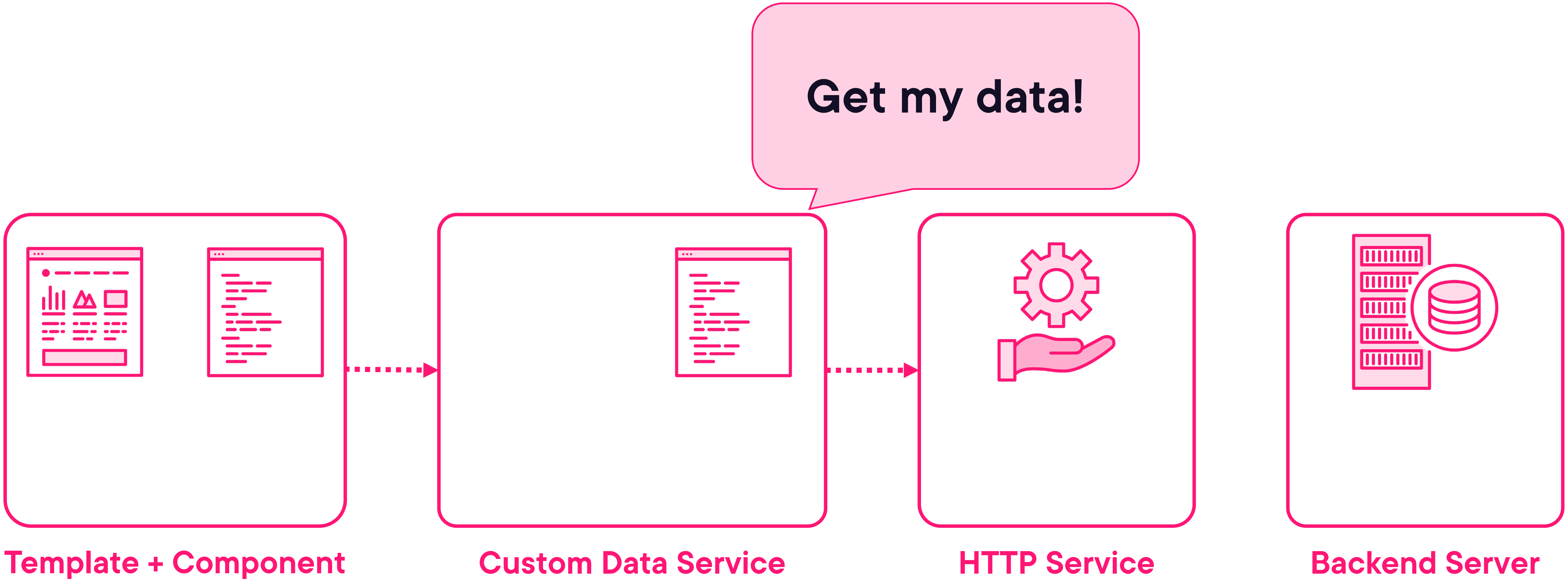
RxJS + Signals: Better Together

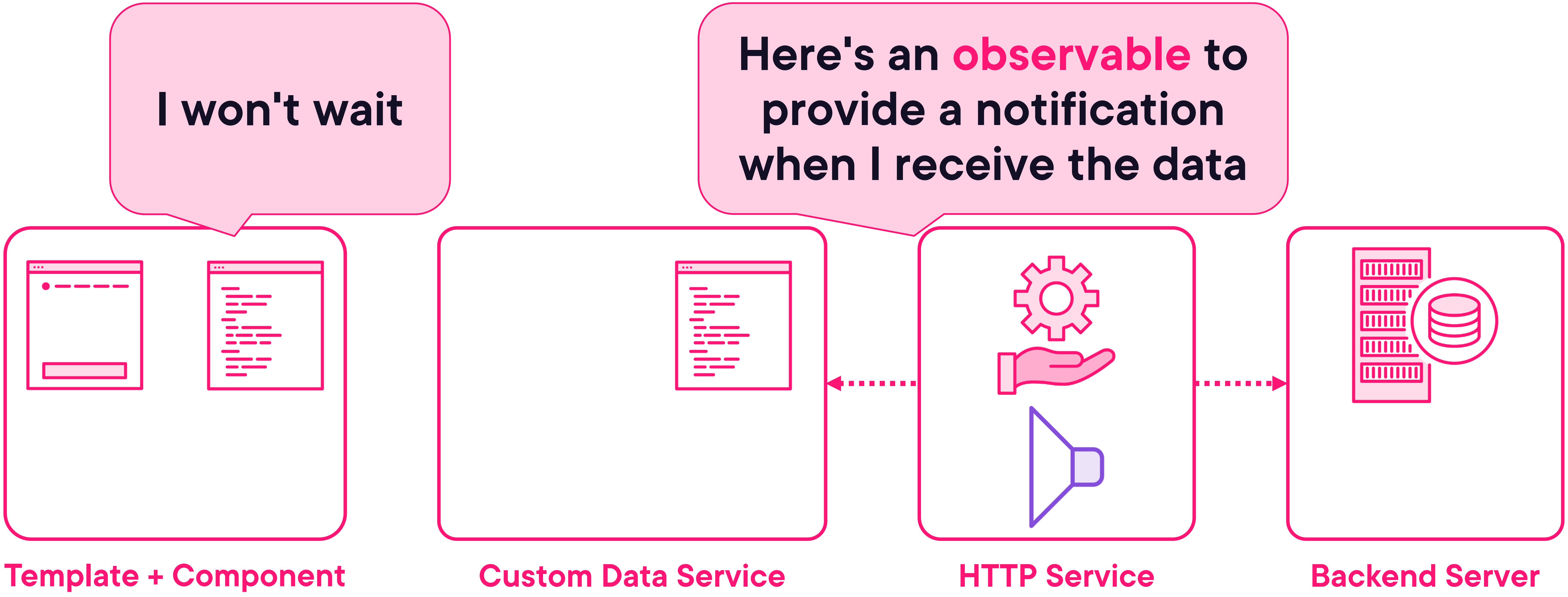
Deborah Kurata
Developer

https://www.youtube.com/@deborah_kurata

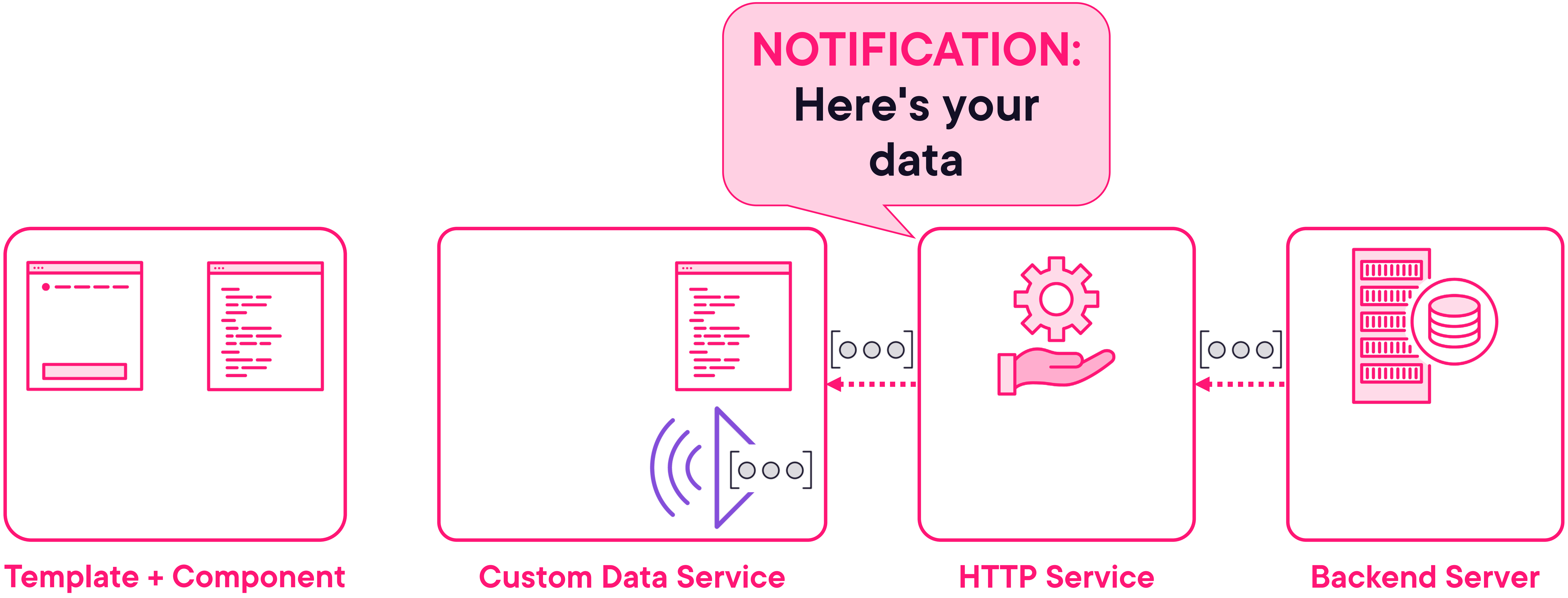
RxJS + Signals: Better Together



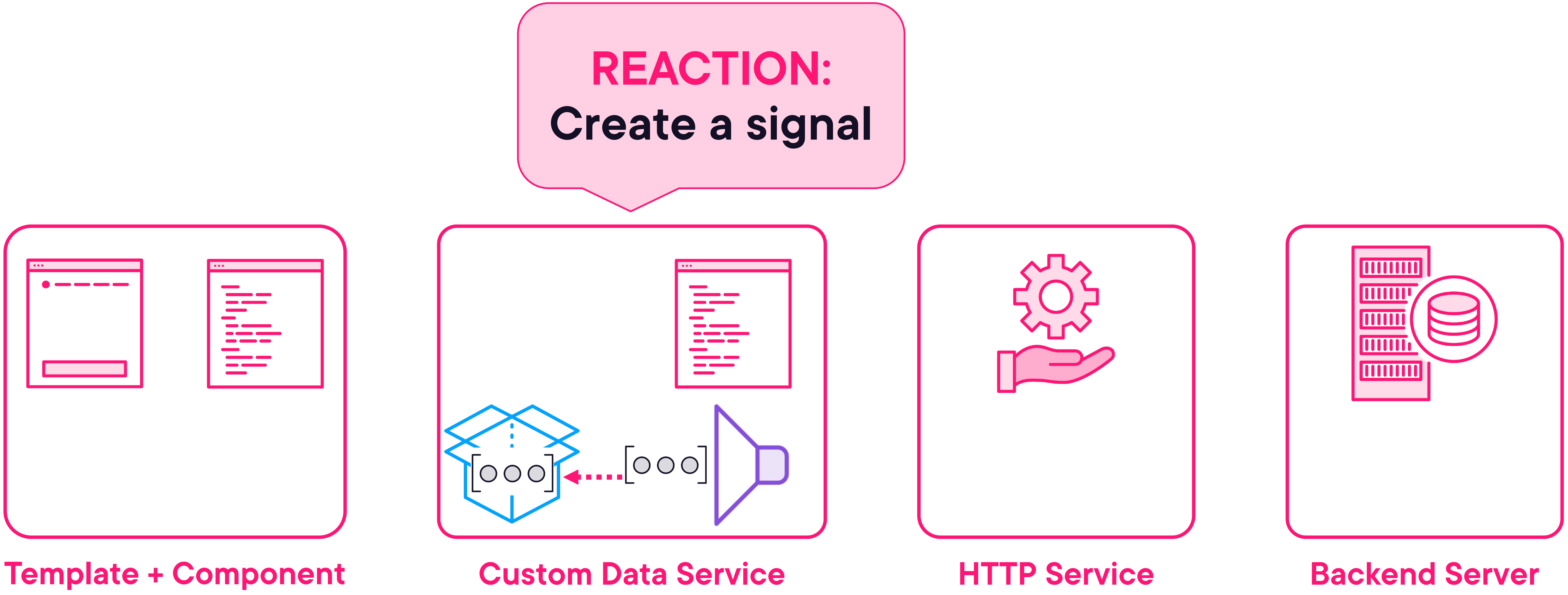
RxJS + Signals: Better Together



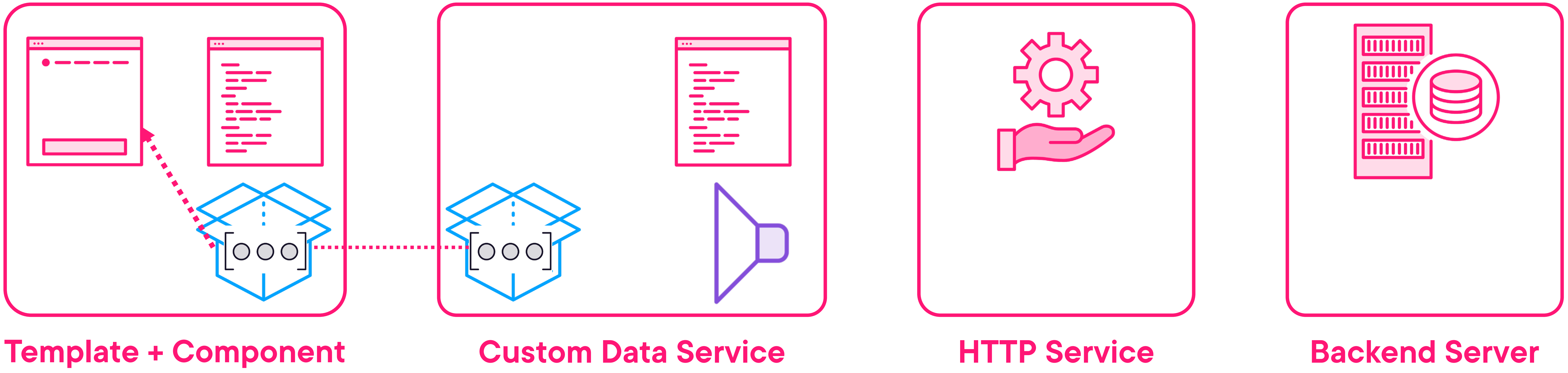
RxJS + Signals: Better Together



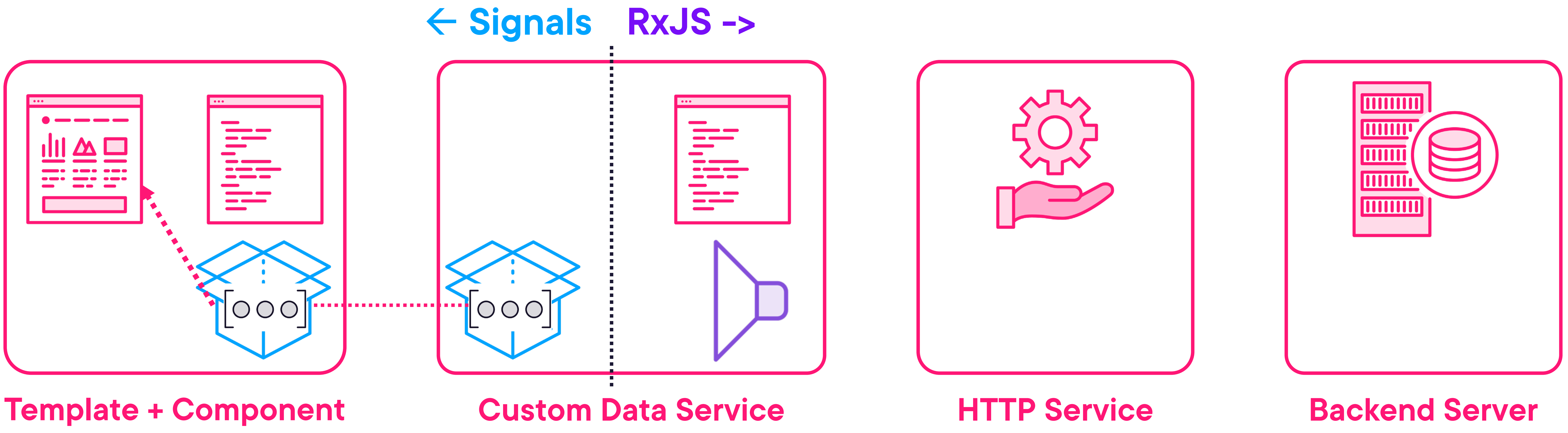
RxJS + Signals: Better Together



RxJS + Signals: Better Together

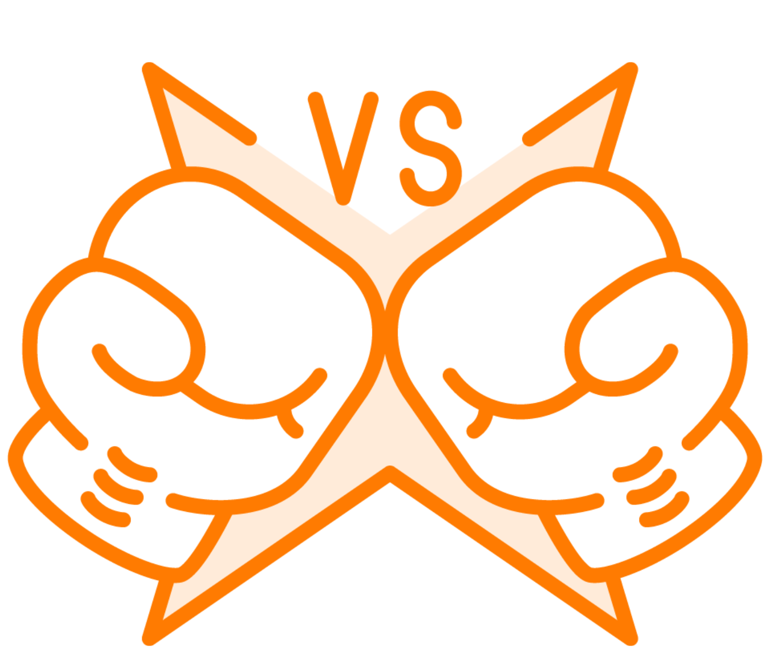


RxJS + Signals: Better Together

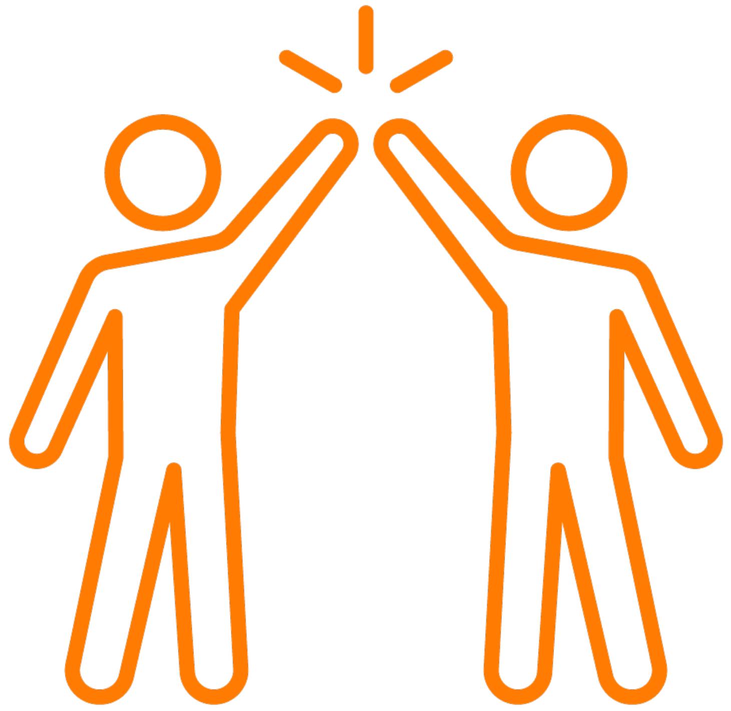


Signals	RxJS
Basic state management	Async operations / HTTP requests
Reactivity: Computed signals	Operations on the response
Improved change detection / performance	Reactivity: User actions and other events, when we need every notification

RxJS + Signals: Better Together



Signals vs. Observables



Better together

Overview

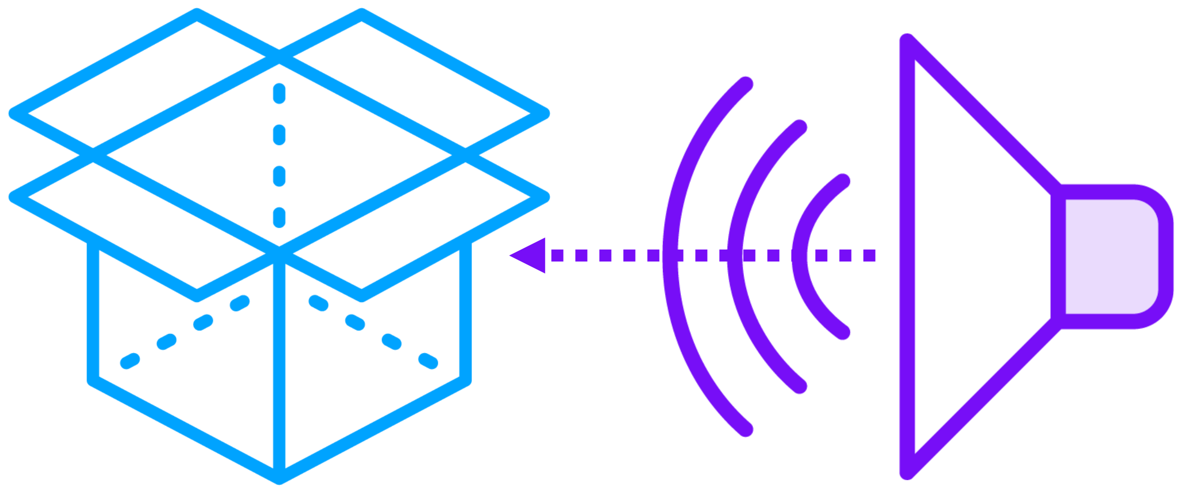
Create a readonly signal from an observable

Error handling

Compare a Subject with a signal

Create an observable from a signal

Create a Signal from an Observable: `toSignal`



Holds the emitted value from the provided observable

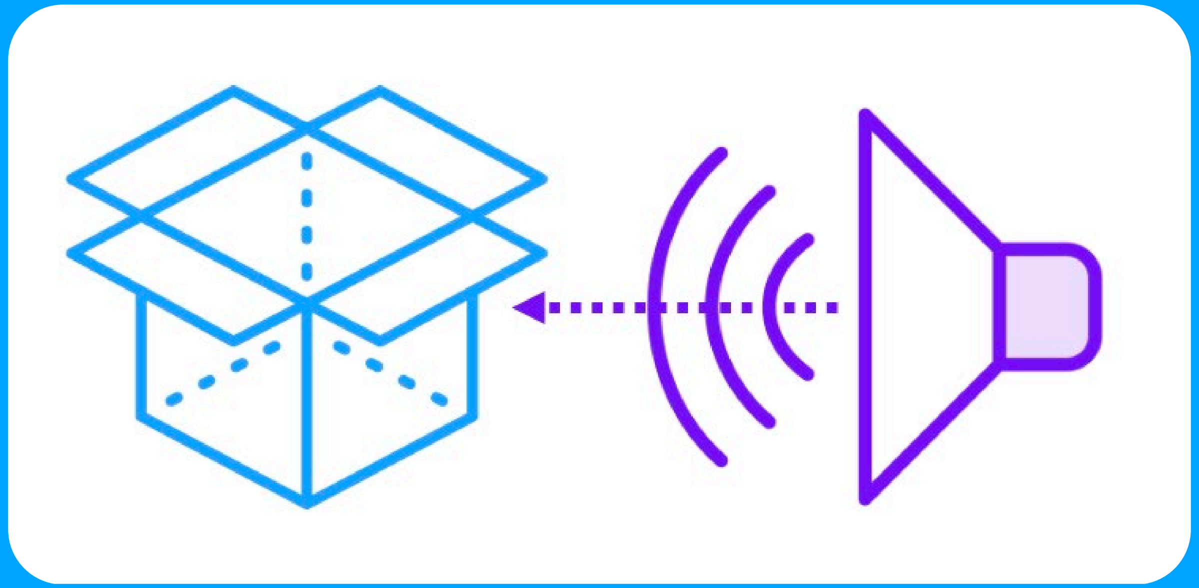
```
product = toSignal(this.products$);
```

Provides synchronous access to the values emitted from that observable

Always contains the most recent emitted value

Automatically subscribes and unsubscribes

Signals created using `toSignal` are read only!



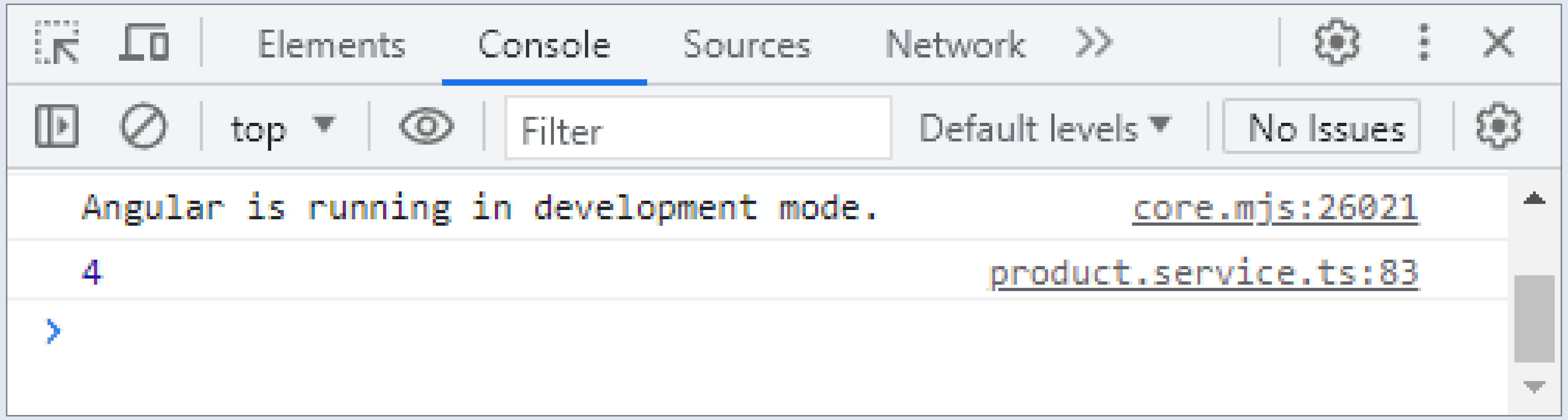
toSignal

```
product = toSignal(this.products$);
```

```
product = toSignal(this.products$,
  { initialValue: [] as Product[] });
```

toSignal

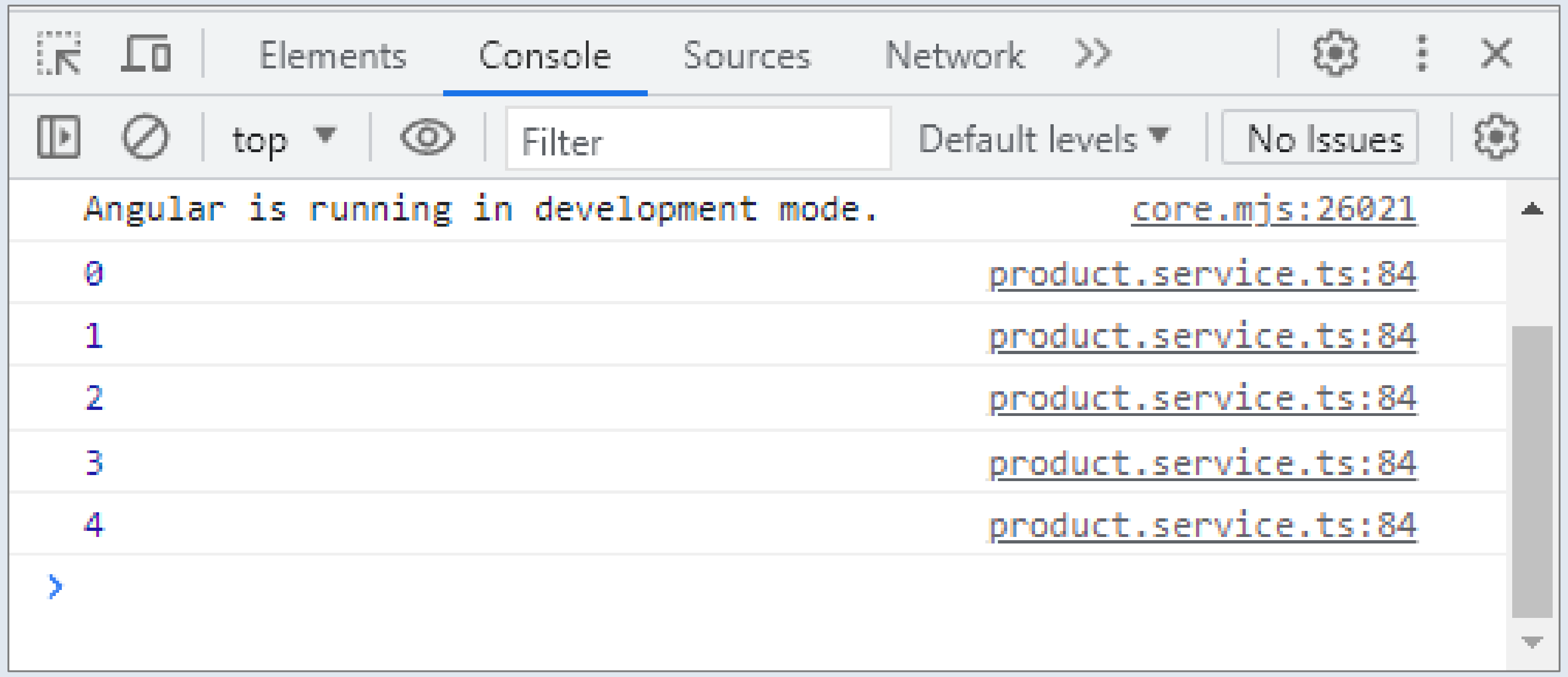
```
o$ = of(1, 2, 3, 4);

s = toSignal(this.o$, { initialValue: 0 });

e = effect(() => console.log(this.s()));
```



```
Angular is running in development mode.        core.mjs:26021
4                                         product.service.ts:83
```

toSignal

```
o$ = of(1, 2, 3, 4)
        .pipe(delay(this.randomDelay()));

s = toSignal(this.o$, { initialValue: 0 });

e = effect(() => console.log(this.s()));
```



```
Angular is running in development mode.          core.mjs:26021
0                                             product.service.ts:84
1                                             product.service.ts:84
2                                             product.service.ts:84
3                                             product.service.ts:84
4                                             product.service.ts:84
```

Demo

Use `toSignal` to create a signal from an observable

Use that signal in the template

If signals are just simple **containers for values**, how can they **generate an error**?

Signal Errors: toSignal

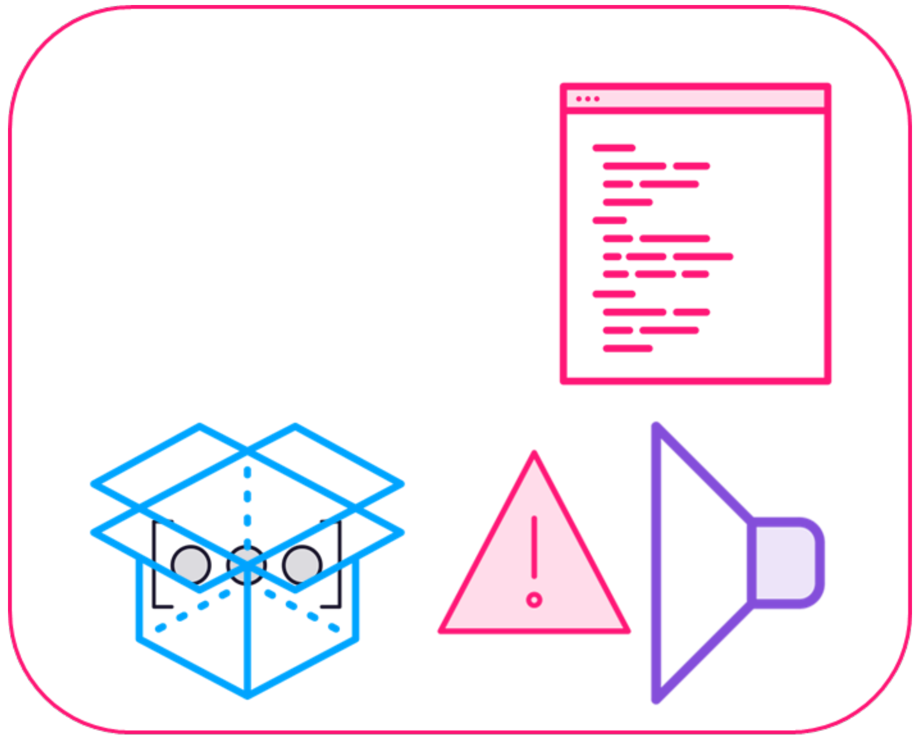
```
private products$ = this.http.get<Product[]>(this.productsUrl)
  .pipe(
    catchError(err => this.handleError(err))
  );

products = toSignal(this.products$,
                    { initialValue: [] as Product[] });
```

```
handleError(err: HttpErrorResponse): Observable<never> {
  const formattedMessage = this.errorService.formatError(err);
  return throwError(() => formattedMessage);
}
```

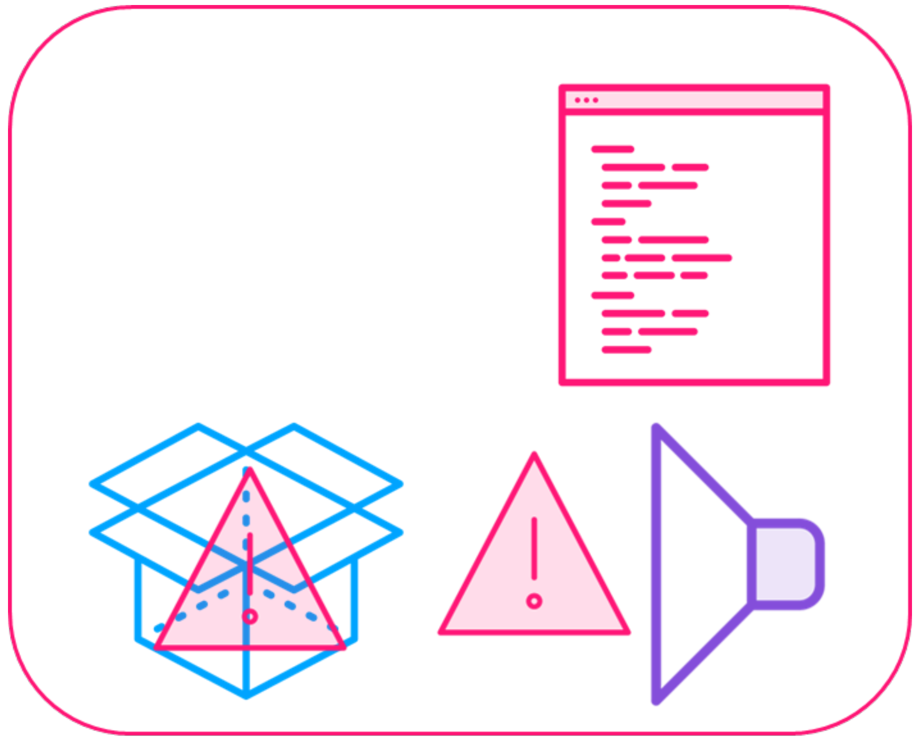
Signal Errors: computed

```
count = signal(3);
validate = computed(() => {
  if (this.count() === 3) {
    throw 'Validation error!';
  }
});
```

Error Handling Options



Catch in the observable pipeline, create a replacement observable, pass valid data to the signal



Catch using try...catch

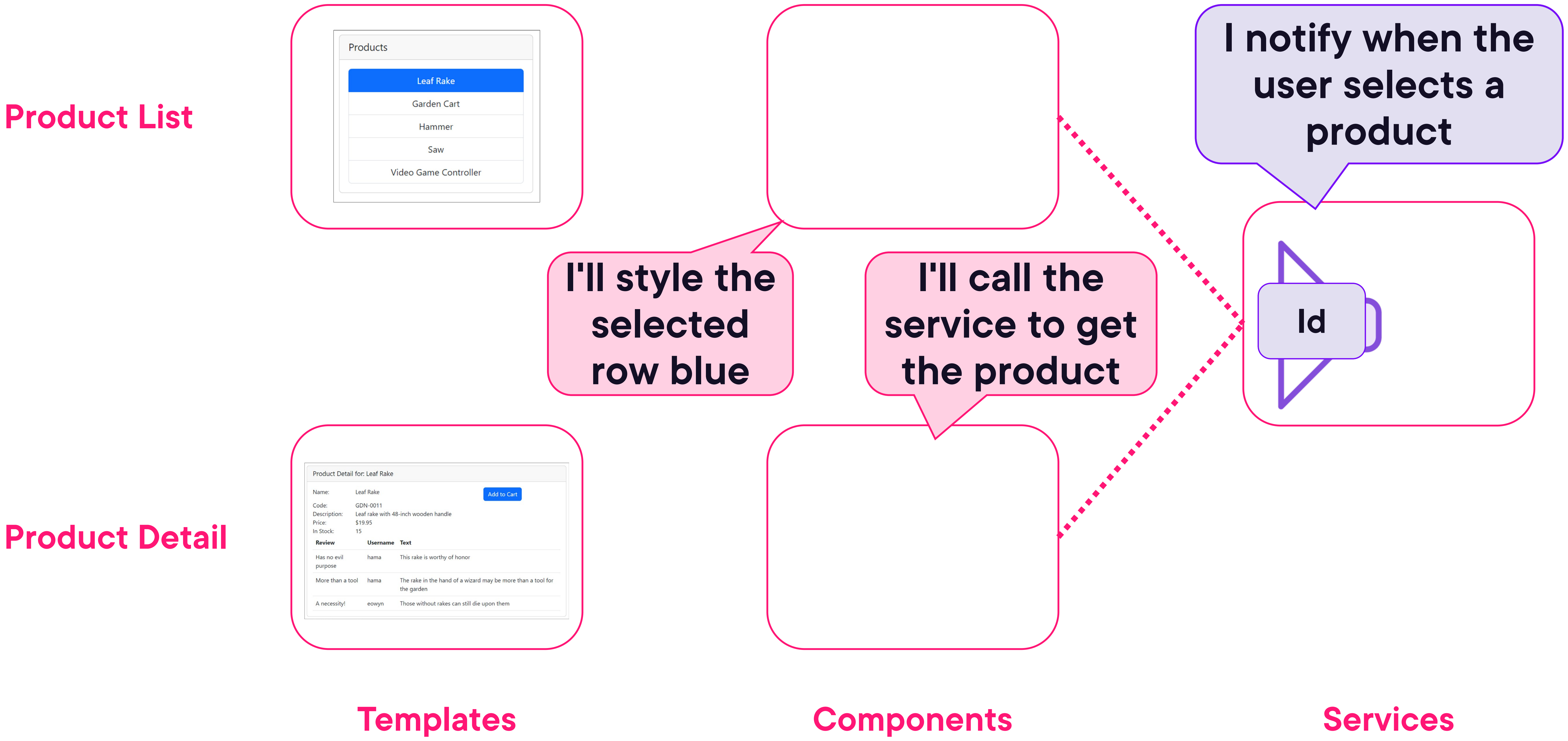
Demo

Handle errors using try…catch

Demo

Handle errors using the RxJS pipeline

Reacting to User Actions with a Subject


Product List
Product Detail
I notify when the user selects a product
I'll style the selected row blue
I'll call the service to get the product
Id
Templates
Components
Services

Products

- Leaf Rake
- Garden Cart
- Hammer
- Saw
- Video Game Controller

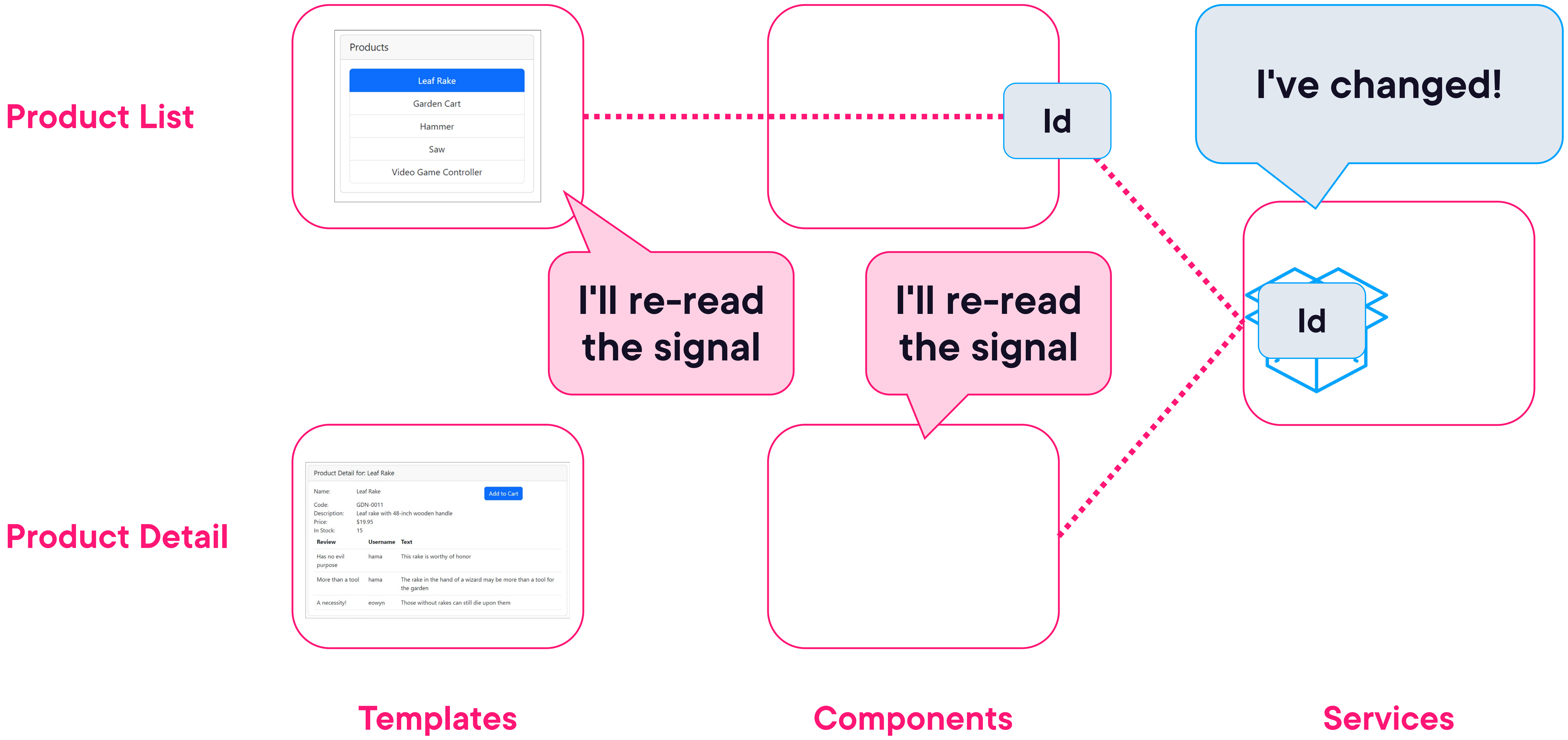
Product Detail for: Leaf Rake

Name:	Leaf Rake
Code:	GDN-0011
Description:	Leaf rake with 48-inch wooden handle
Price:	$19.95
In Stock:	15

Add to Cart

Review	Username	Text
Has no evil purpose	hama	This rake is worthy of honor
More than a tool	hama	The rake in the hand of a wizard may be more than a tool for the garden
A necessity!	eowyn	Those without rakes can still die upon them

Reacting to User Actions with a Signal



Products

- Leaf Rake
- Garden Cart
- Hammer
- Saw
- Video Game Controller

Product Detail for: Leaf Rake

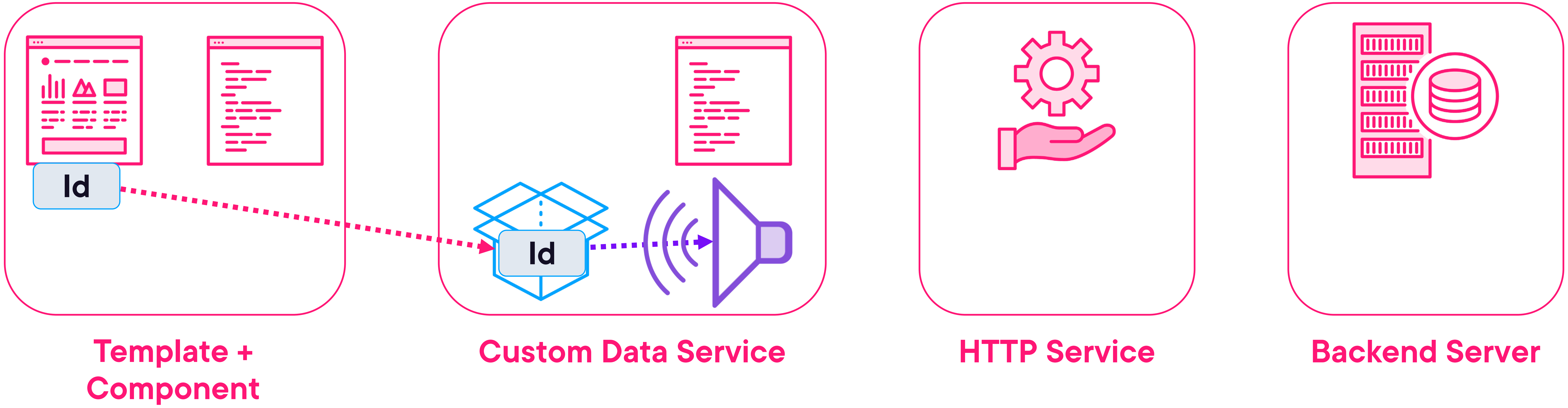
Name:	Leaf Rake
Code:	GDN-0011
Description:	Leaf rake with 48-inch wooden handle
Price:	$19.95
In Stock:	15

Review	Username	Text
Has no evil purpose	hama	This rake is worthy of honor
More than a tool	hama	The rake in the hand of a wizard may be more than a tool for the garden
A necessity!	eowyn	Those without rakes can still die upon them

Demo

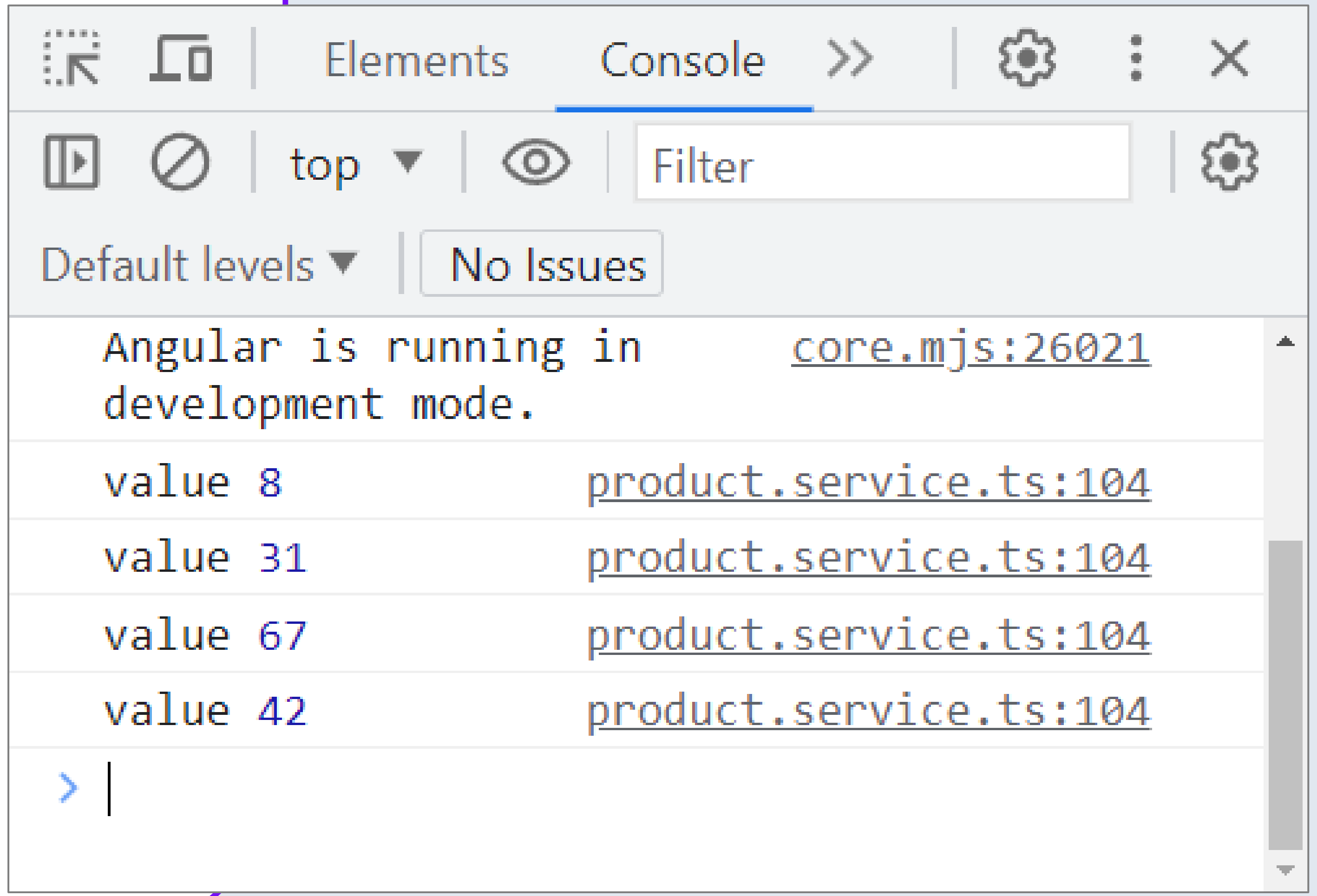
Replace a BehaviorSubject with a signal

Creating an Observable from a Signal (`toObservable`)



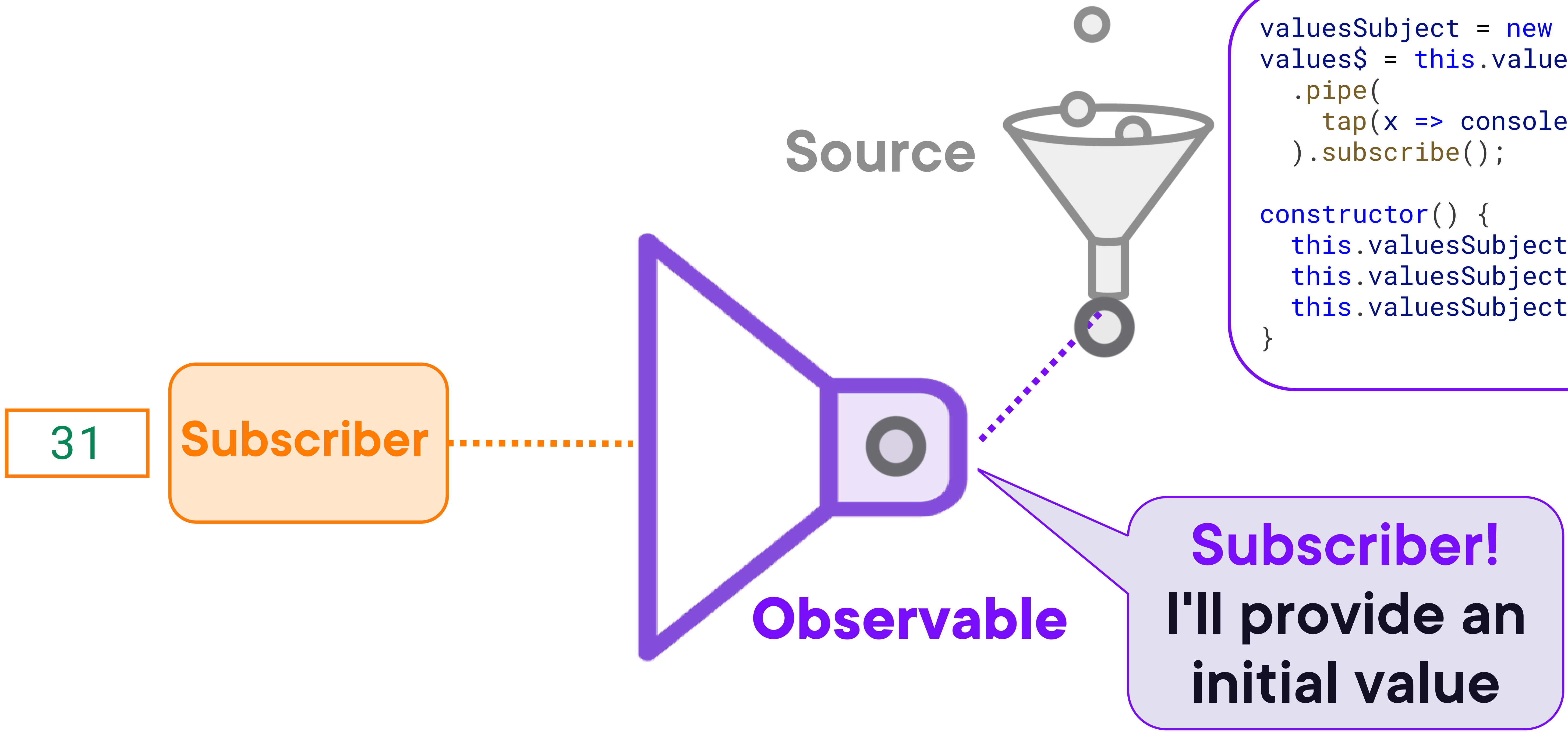
With Subject, Notifications Are Asynchronous

```
valuesSubject = new BehaviorSubject(8);
values$ = this.valuesSubject.asObservable()
  .pipe(
    tap(x => console.log('value', x))
  ).subscribe();

constructor() {
  this.valuesSubject.next(31);
  this.valuesSubject.next(67);
  this.valuesSubject.next(42);
}
```



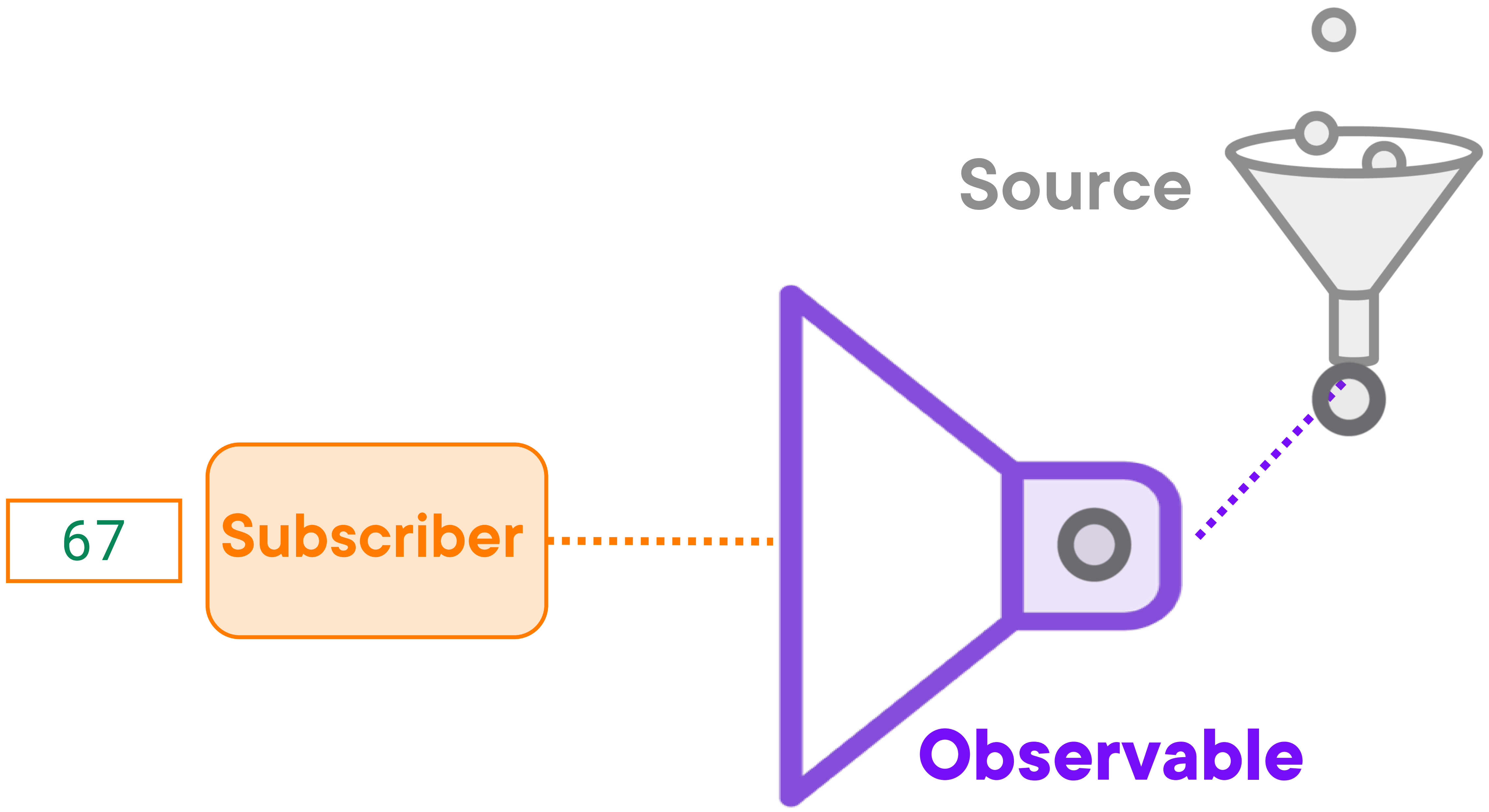
```
Angular is running in        core.mjs:26021
development mode.
value 8                 product.service.ts:104
value 31                product.service.ts:104
value 67                product.service.ts:104
value 42                product.service.ts:104
```

With Subject, Notifications Are Asynchronous



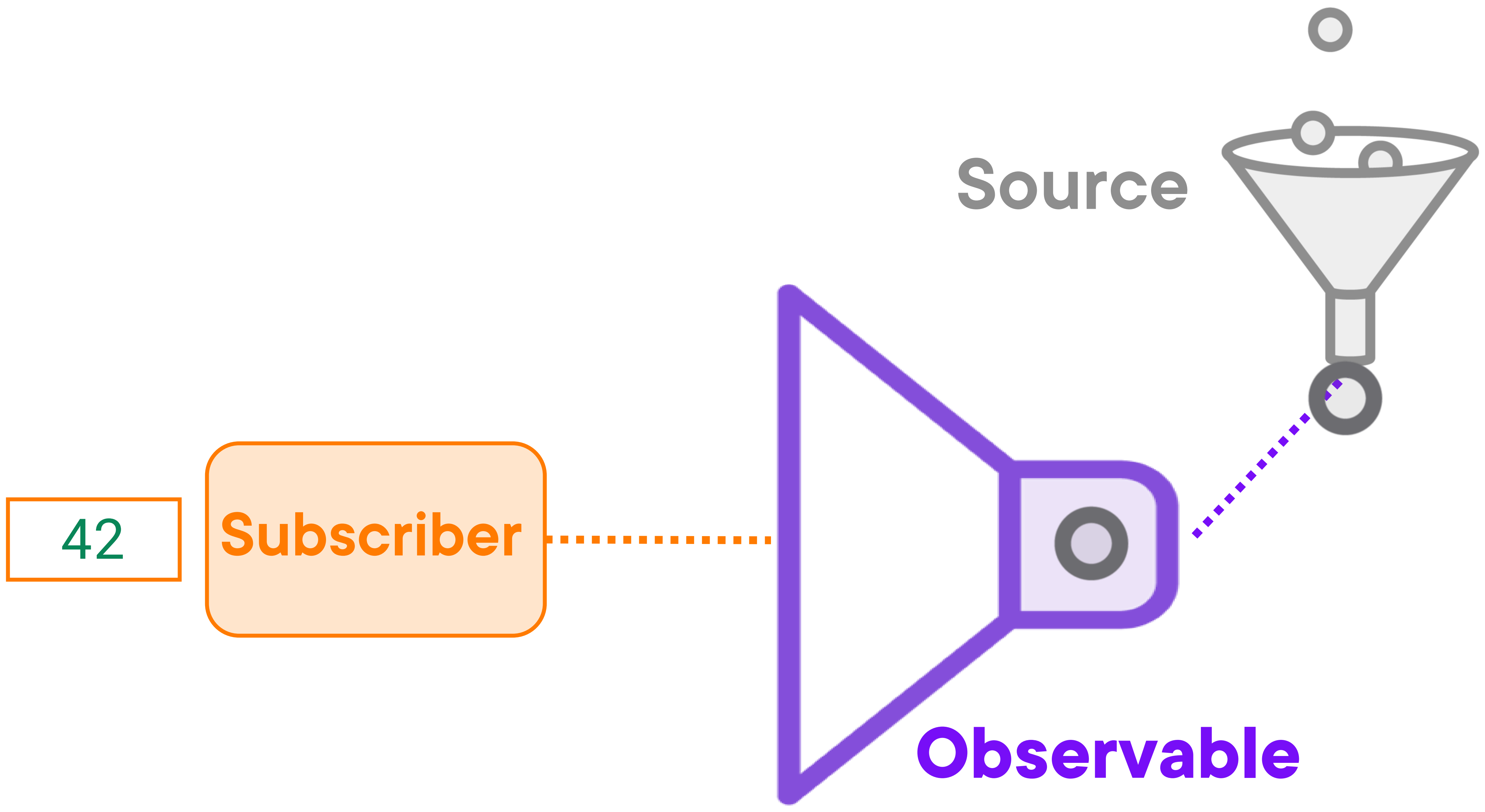
```
valuesSubject = new BehaviorSubject(8);
values$ = this.valuesSubject.asObservable()
  .pipe(
    tap(x => console.log('value', x))
  ).subscribe();

constructor() {
  this.valuesSubject.next(31);
  this.valuesSubject.next(67);
  this.valuesSubject.next(42);
}
```

With Subject, Notifications Are Asynchronous



```
valuesSubject = new BehaviorSubject(8);
values$ = this.valuesSubject.asObservable()
  .pipe(
    tap(x => console.log('value', x))
  ).subscribe();

constructor() {
  this.valuesSubject.next(31);
  this.valuesSubject.next(67);
  this.valuesSubject.next(42);
}
```

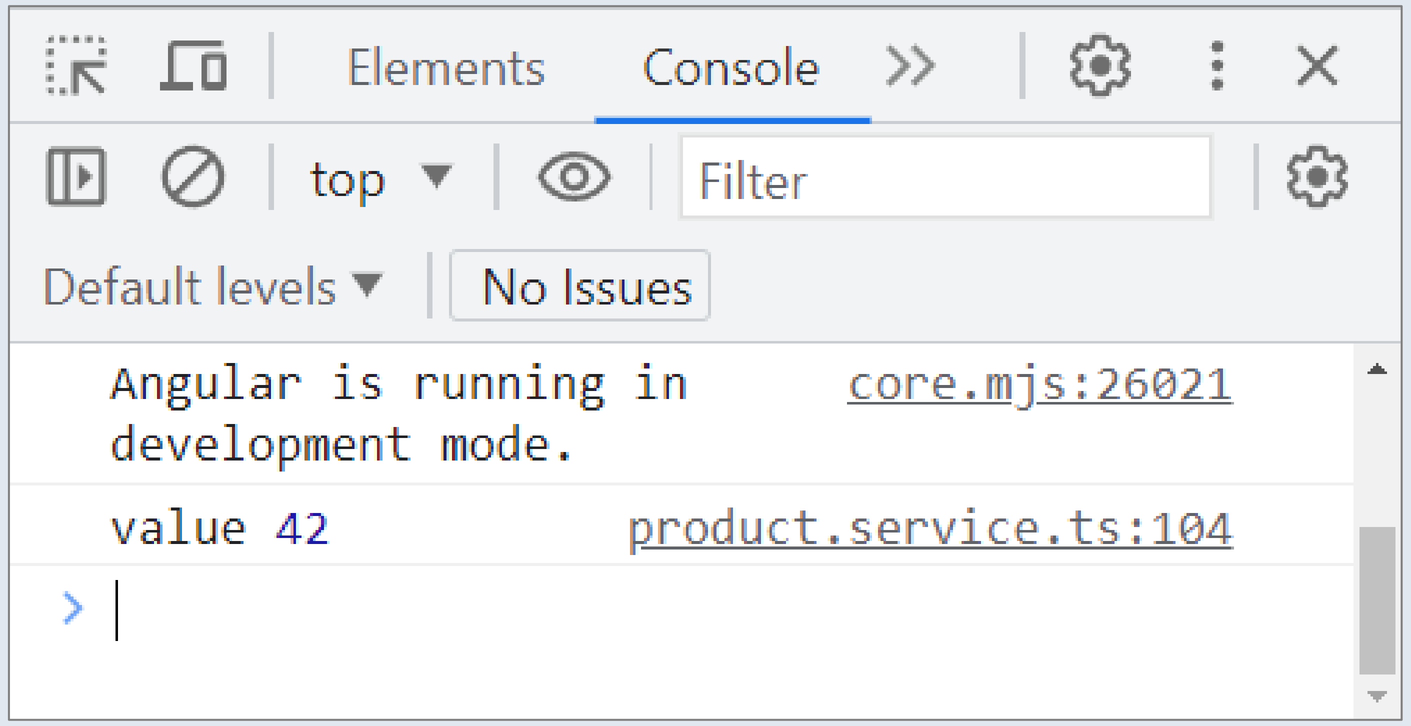
With Subject, Notifications Are Asynchronous



```
valuesSubject = new BehaviorSubject(8);
values$ = this.valuesSubject.asObservable()
  .pipe(
    tap(x => console.log('value', x))
  ).subscribe();

constructor() {
  this.valuesSubject.next(31);
  this.valuesSubject.next(67);
  this.valuesSubject.next(42);
}
```

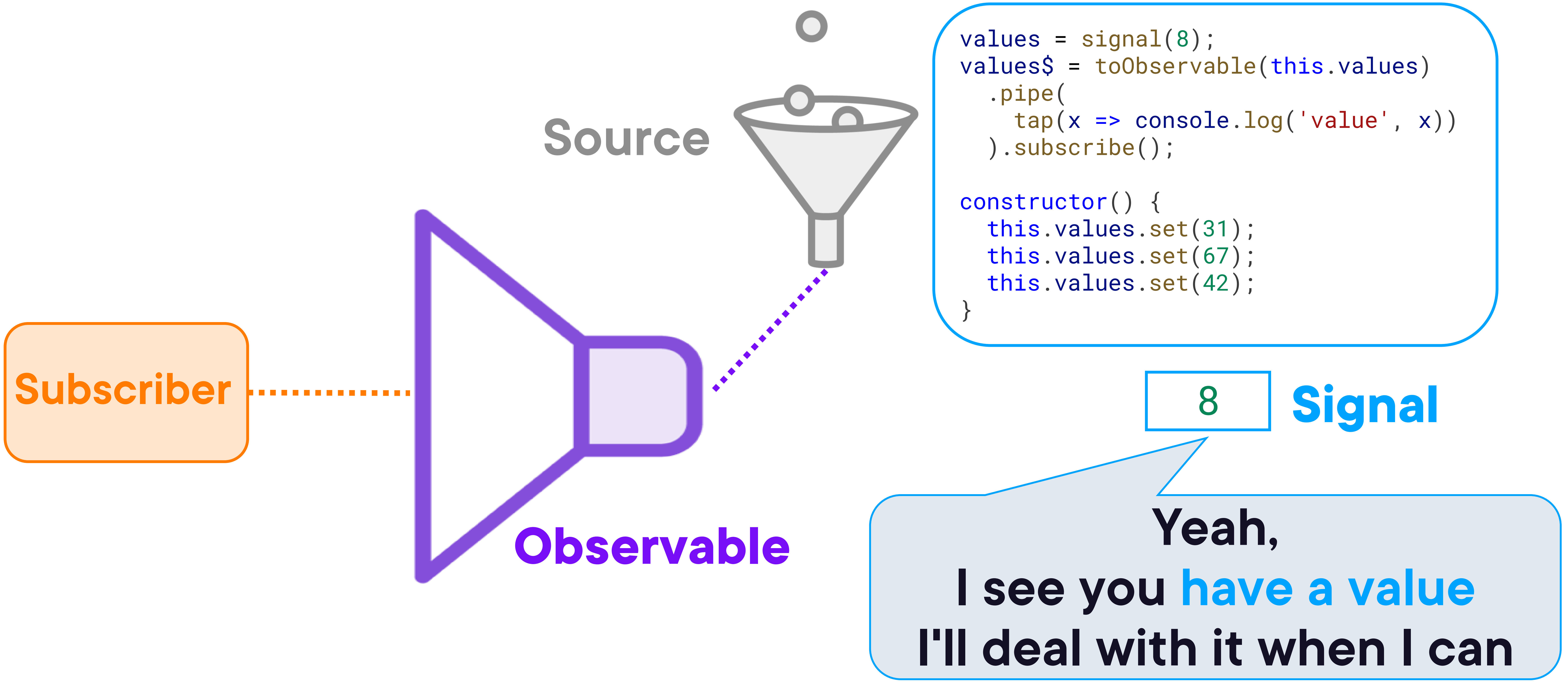
With toObservable, Notifications Are Scheduled

```
values = signal(8);
values$ = toObservable(this.values)
  .pipe(
    tap(x => console.log('value', x))
  ).subscribe();

constructor() {
  this.values.set(31);
  this.values.set(67);
  this.values.set(42);
}
```



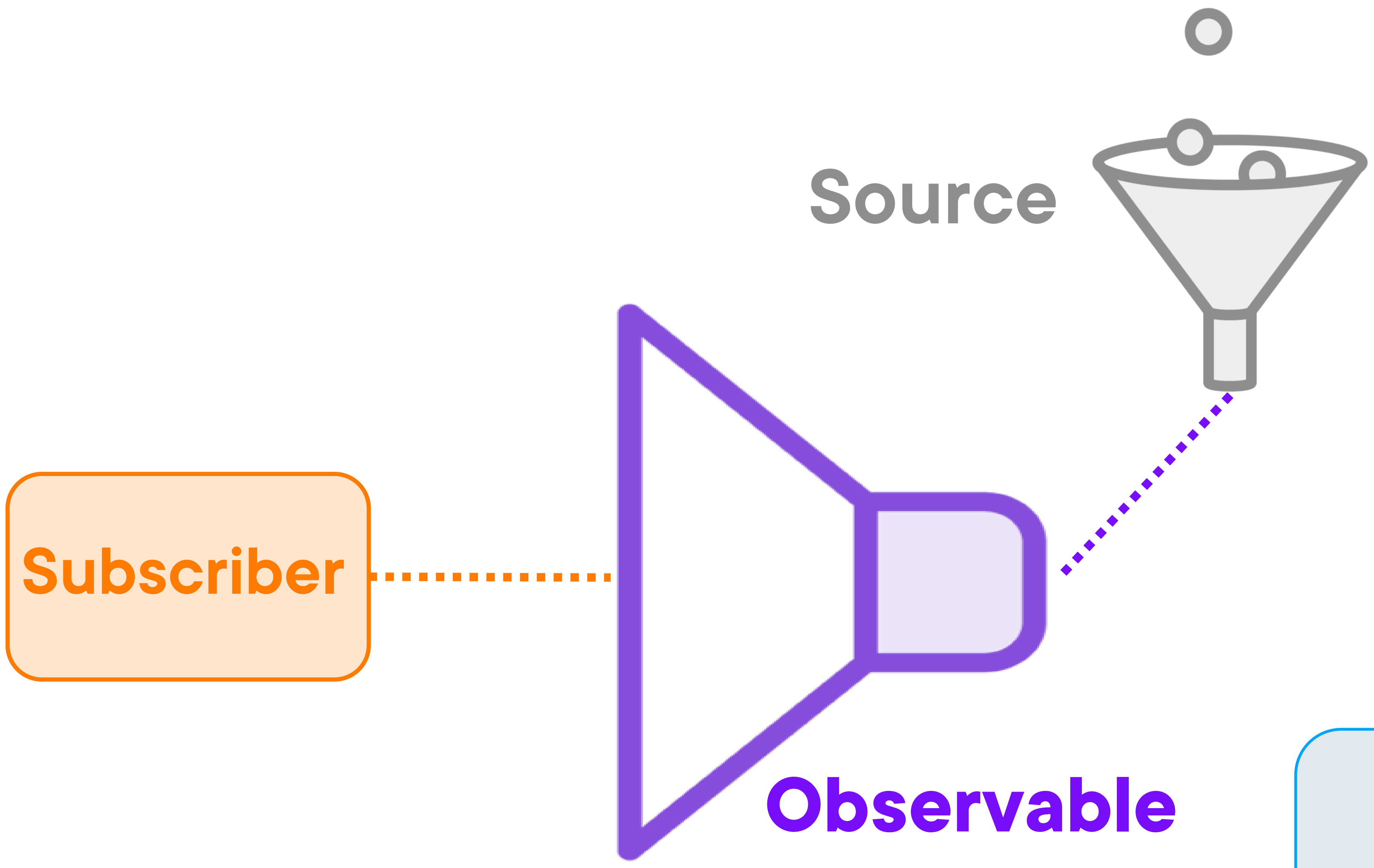
```
Angular is running in          core.mjs:26021
development mode.
value 42                 product.service.ts:104
>
```

With `toObservable`, Notifications Are Scheduled



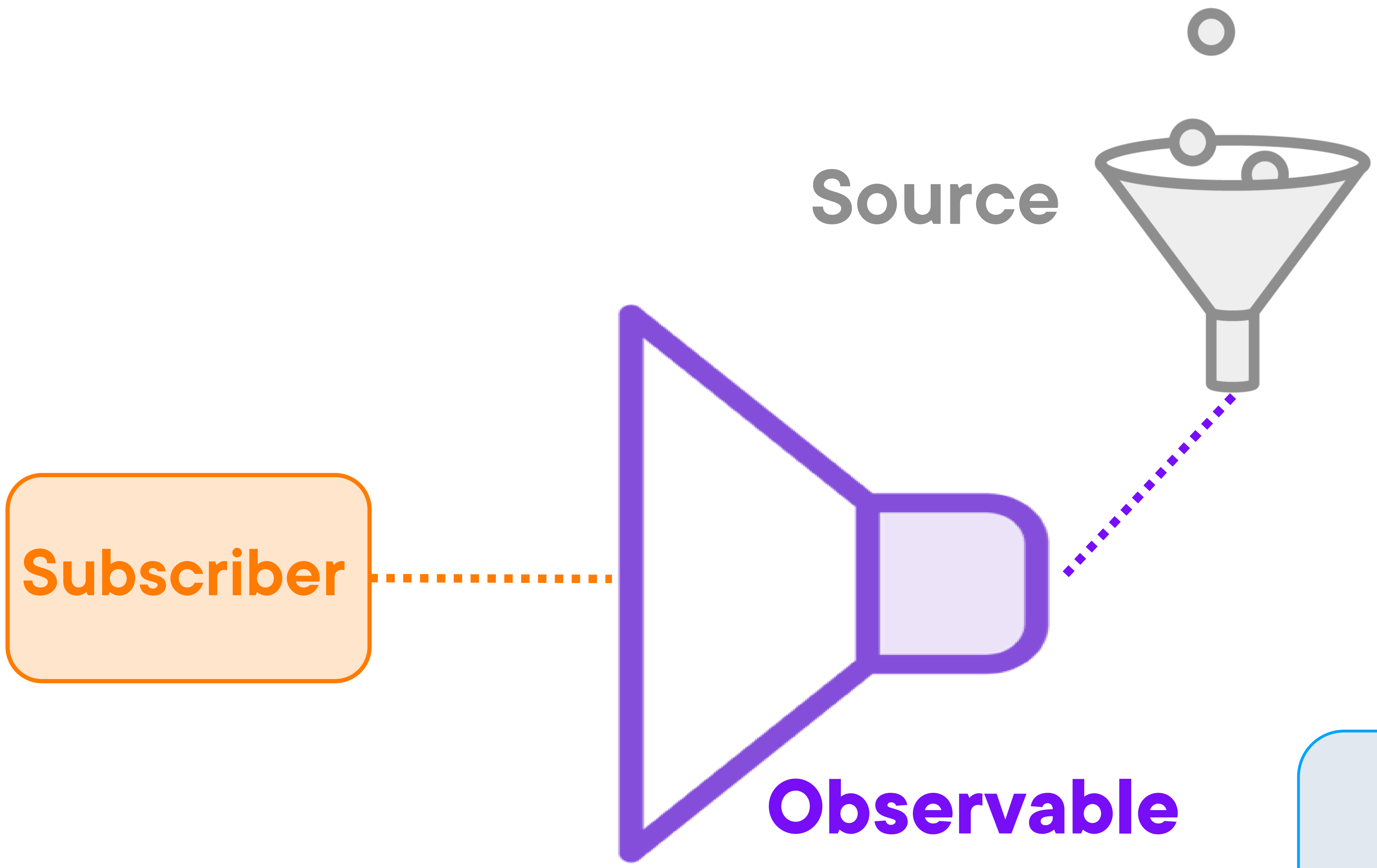
```
values = signal(8);
values$ = toObservable(this.values)
  .pipe(
    tap(x => console.log('value', x))
  ).subscribe();

constructor() {
  this.values.set(31);
  this.values.set(67);
  this.values.set(42);
}
```

With `toObservable`, Notifications Are Scheduled

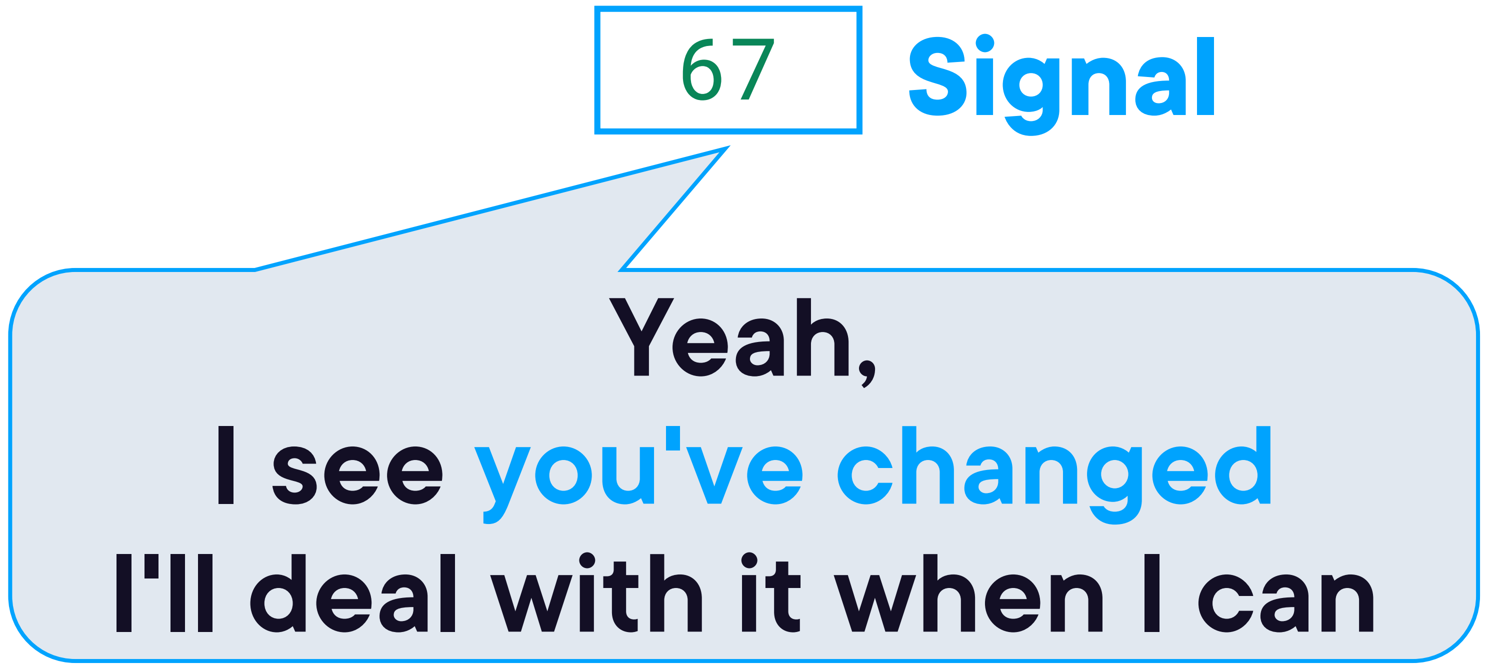


```
values = signal(8);
values$ = toObservable(this.values)
  .pipe(
    tap(x => console.log('value', x))
  ).subscribe();

constructor() {
  this.values.set(31);
  this.values.set(67);
  this.values.set(42);
}
```

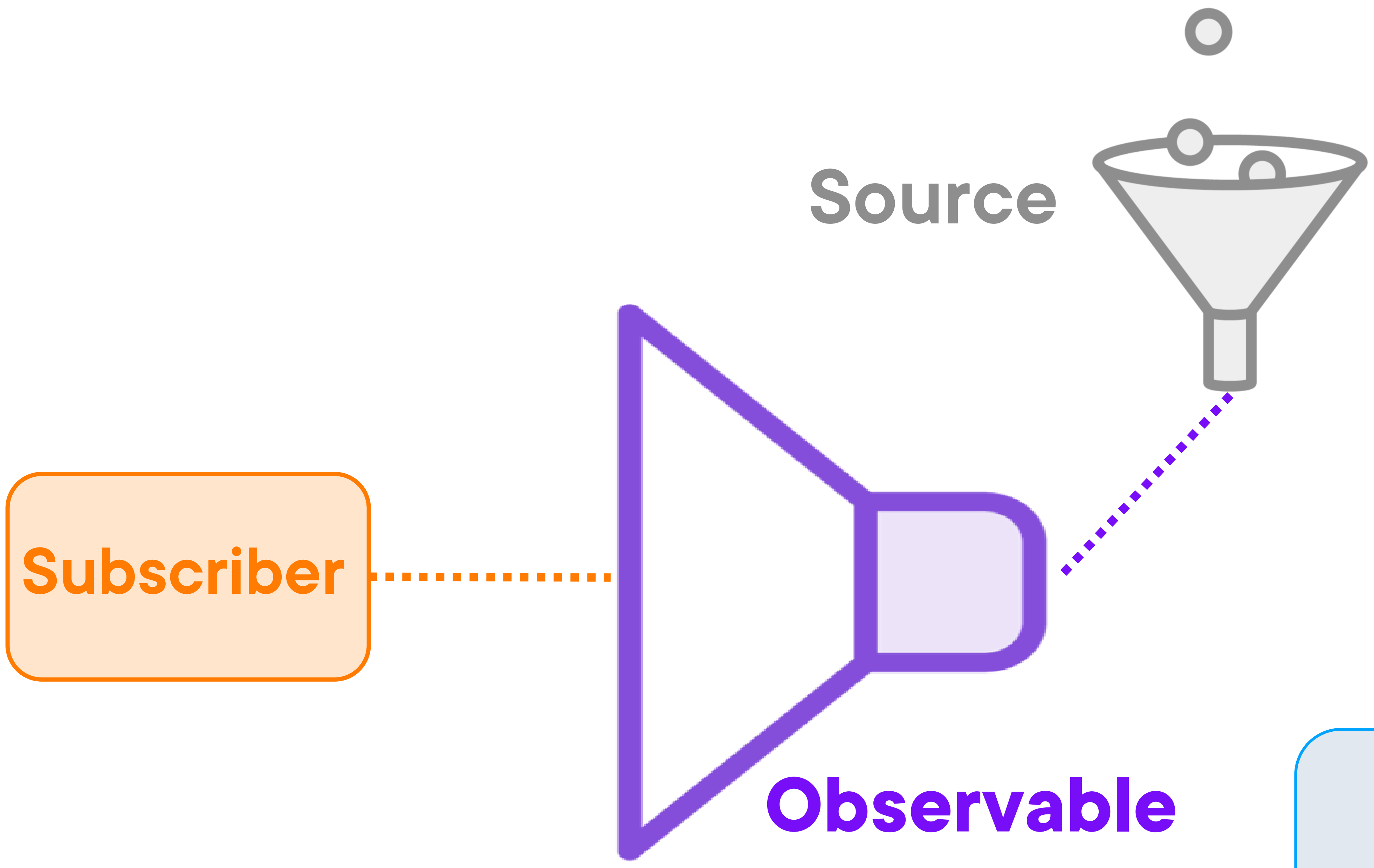
With `toObservable`, Notifications Are Scheduled



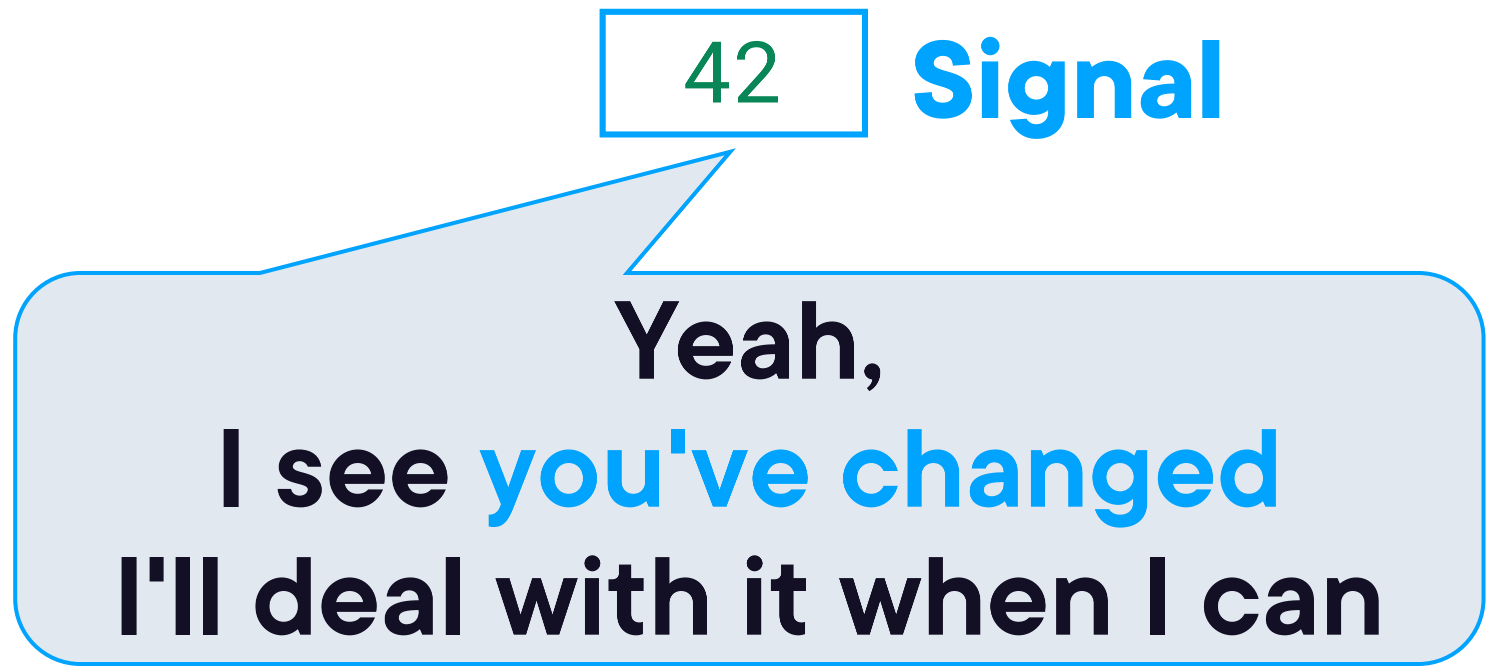
```
values = signal(8);
values$ = toObservable(this.values)
  .pipe(
    tap(x => console.log('value', x))
  ).subscribe();

constructor() {
  this.values.set(31);
  this.values.set(67);
  this.values.set(42);
}
```



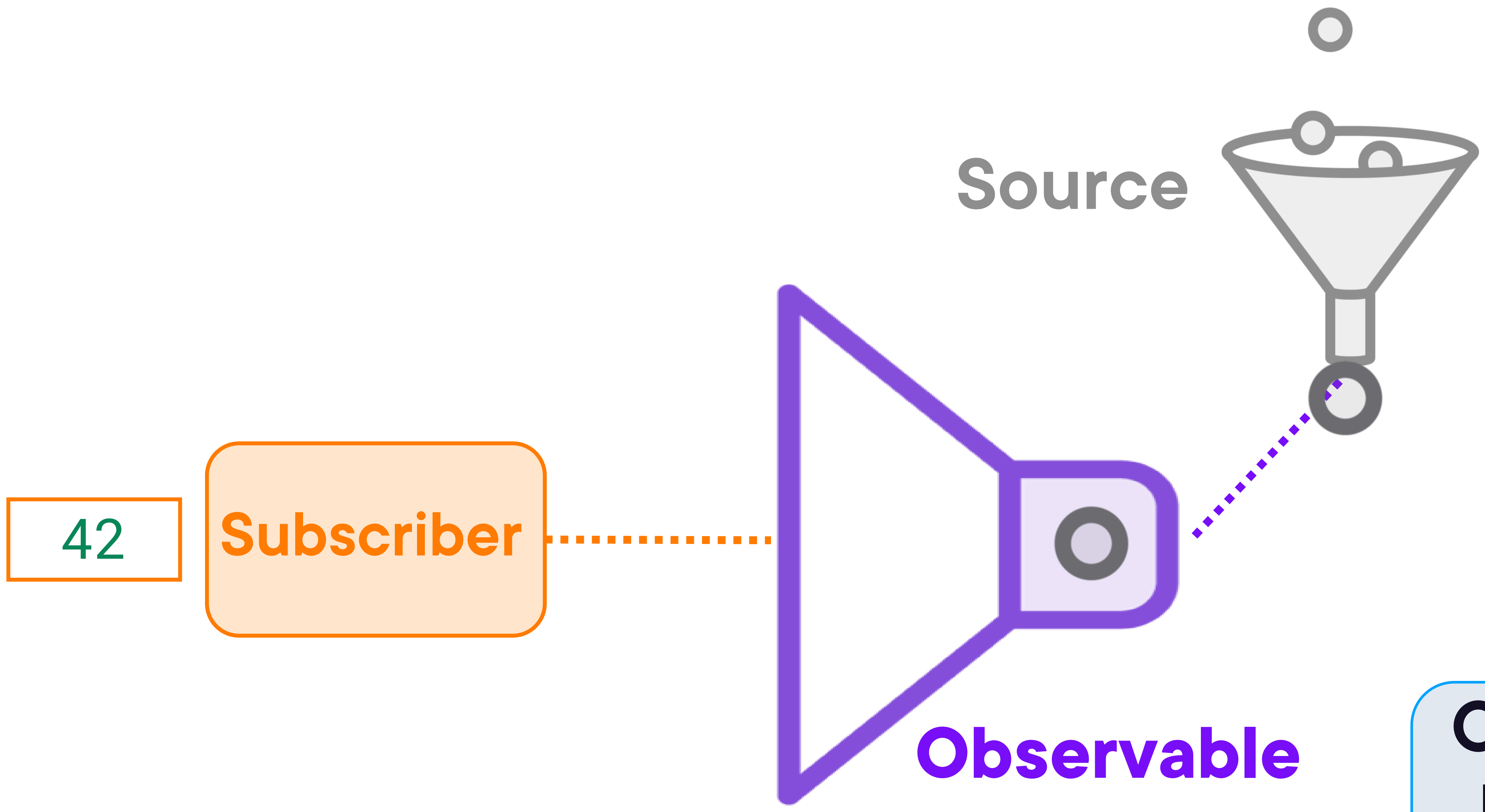
With `toObservable`, Notifications Are Scheduled



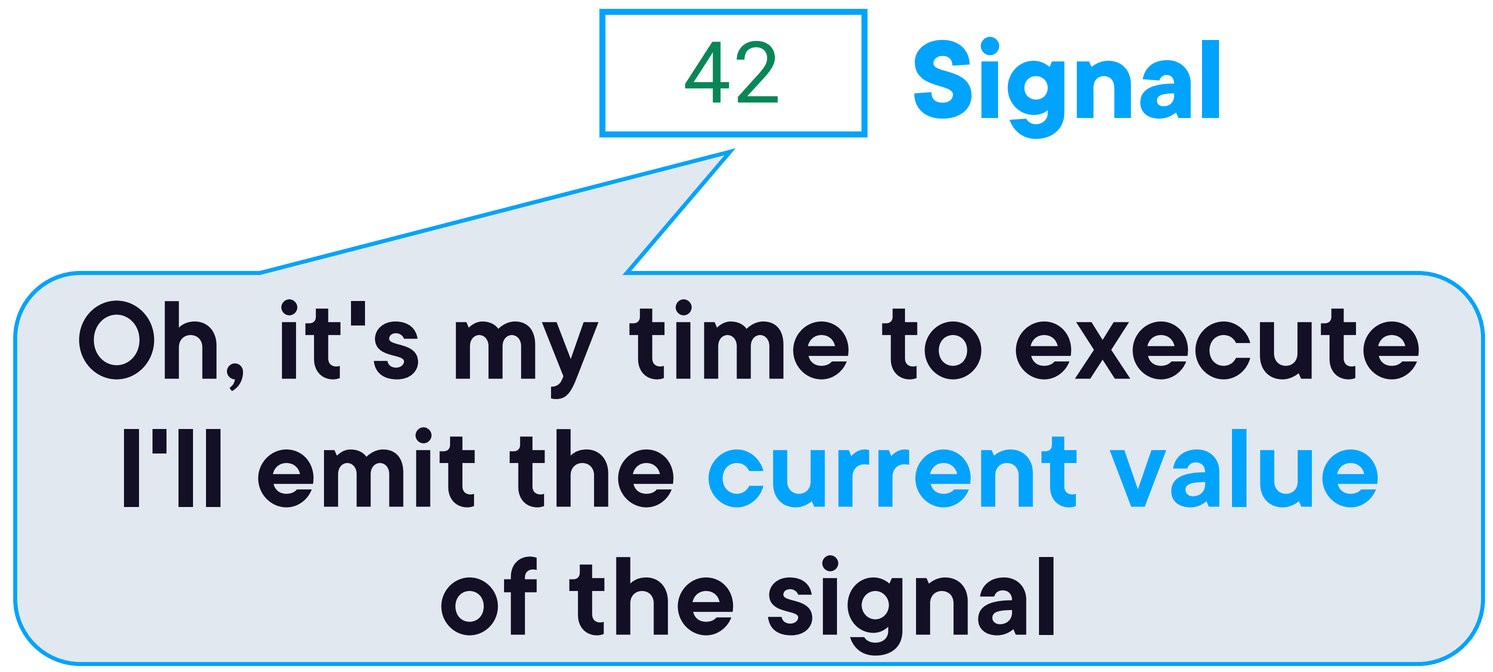
```
values = signal(8);
values$ = toObservable(this.values)
  .pipe(
    tap(x => console.log('value', x))
  ).subscribe();

constructor() {
  this.values.set(31);
  this.values.set(67);
  this.values.set(42);
}
```



With `toObservable`, Notifications Are Scheduled



```
values = signal(8);
values$ = toObservable(this.values)
  .pipe(
    tap(x => console.log('value', x))
  ).subscribe();

constructor() {
  this.values.set(31);
  this.values.set(67);
  this.values.set(42);
}
```



Oh, it's my time to execute
I'll emit the current value
of the signal

Demo

Use `toObservable` to react to changes in a signal

Demo

Expose only signals from the service

Demo

Only read signals from the service

Key Point

Type Narrowing with Objects

```
product?: Product;
title = '';
...
constructor() {
  if (this.product) {
    this.title = `Detail: ${this.product.name}`;
  } else {
    this.title = 'Detail';
  }
  console.log(this.title);
}
```

I'm checking here if it's undefined

So here it must be defined

That means I can dot into the product's name

Key Point

Type Narrowing with Signals

```
product = signal<Product | undefined>(undefined);
title = '';
...
constructor() {
  if (this.product()) {
    this.title = `Detail: ${this.product().name}`;
  } else {
    this.title = 'Detail';
  }
  console.log(this.pageTitle);
}
```

I'm **reading** the signal here
if it has a value,
I'll execute the if block

But here
I'm **reading it again**
and it could be undefined

Key Point

Type Narrowing with Signals

```
product = signal<Product | undefined>(undefined);
title = '';

constructor() {
  if (this.product()) {
    this.title = `Detail: ${this.product().name}`;
  } else {
    this.title = 'Detail';
  }
  console.log(this.pageTitle);
}
```

```
Object is possibly 'undefined'. ts(2532)
```

How do we fix it??

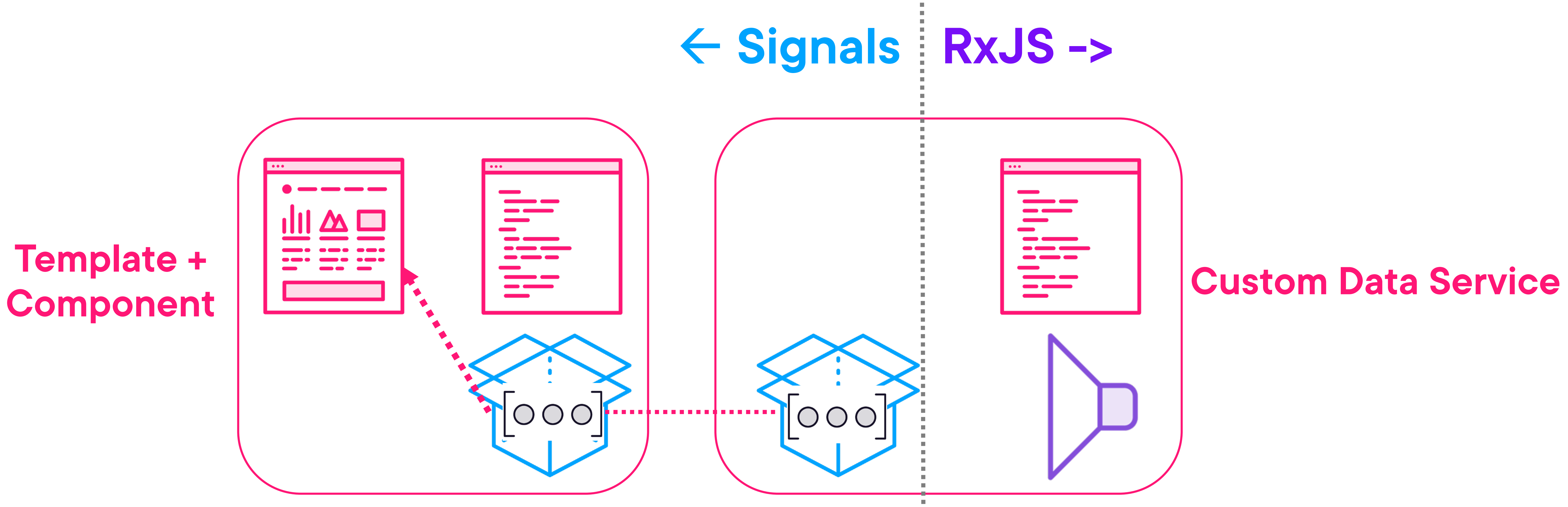
Key Point

Type Narrowing with Signals

```
product = signal<Product | undefined>(undefined);
title = '';

constructor() {
  const p = this.product();
  if (p) {
    this.title = `Detail: ${p.name}`;
  } else {
    this.title = 'Detail';
  }
  console.log(this.pageTitle);
}
```

I'm **reading** the signal here

RxJS + Signals: Better Together



Basic state management

Reactivity: Computed signals

Improved change detection/ performance

Async / HTTP requests

Operations on the response

Reactivity: User actions and other events when we need every notification

Handle Errors

Writable signals hold a value and won't generate an error

Handle errors in

- Computed signals
- Signals created from `toSignal`
- Effects

Use try…catch

Use the observable pipeline

Wrapper Object for Error Handling

```
export interface Result<T>
{
  data: T | undefined;
  error?: string;
}
```

```
private products$ = this.http.get<Product[]>(this.url)
  .pipe(
    map(p => ({ data: p } as Result<Product[]>)),
    catchError(err => of({
      data: [],
      error: err
    } as Result<Product[]>))
  );
```

toSignal
toObservable
Subject

toSignal creates a signal

- Holds the latest value from an observable

```
 product = toSignal(this.products$,
   { initialValue: [] as Product[] });
```

toObservable creates an observable

- Emits the current value of the signal
- When the signal changes, `toObservable` is scheduled to run
- It may not emit notification of every change

```
result$ = toObservable(this.selectedProductId)
    .pipe(...);
```

Use a Subject if you need every notification

For More Information

Demo code

- https://github.com/DeborahK/angular-rxjs-signals-fundamentals

Code from the slides

- https://stackblitz.com/edit/rxjs-signals-m12-deborahk

"Unlocking the Power of Angular Signals + RxJS"

- https://youtu.be/nXJFhZdbWzw

https://youtube.com/@deborah_kurata