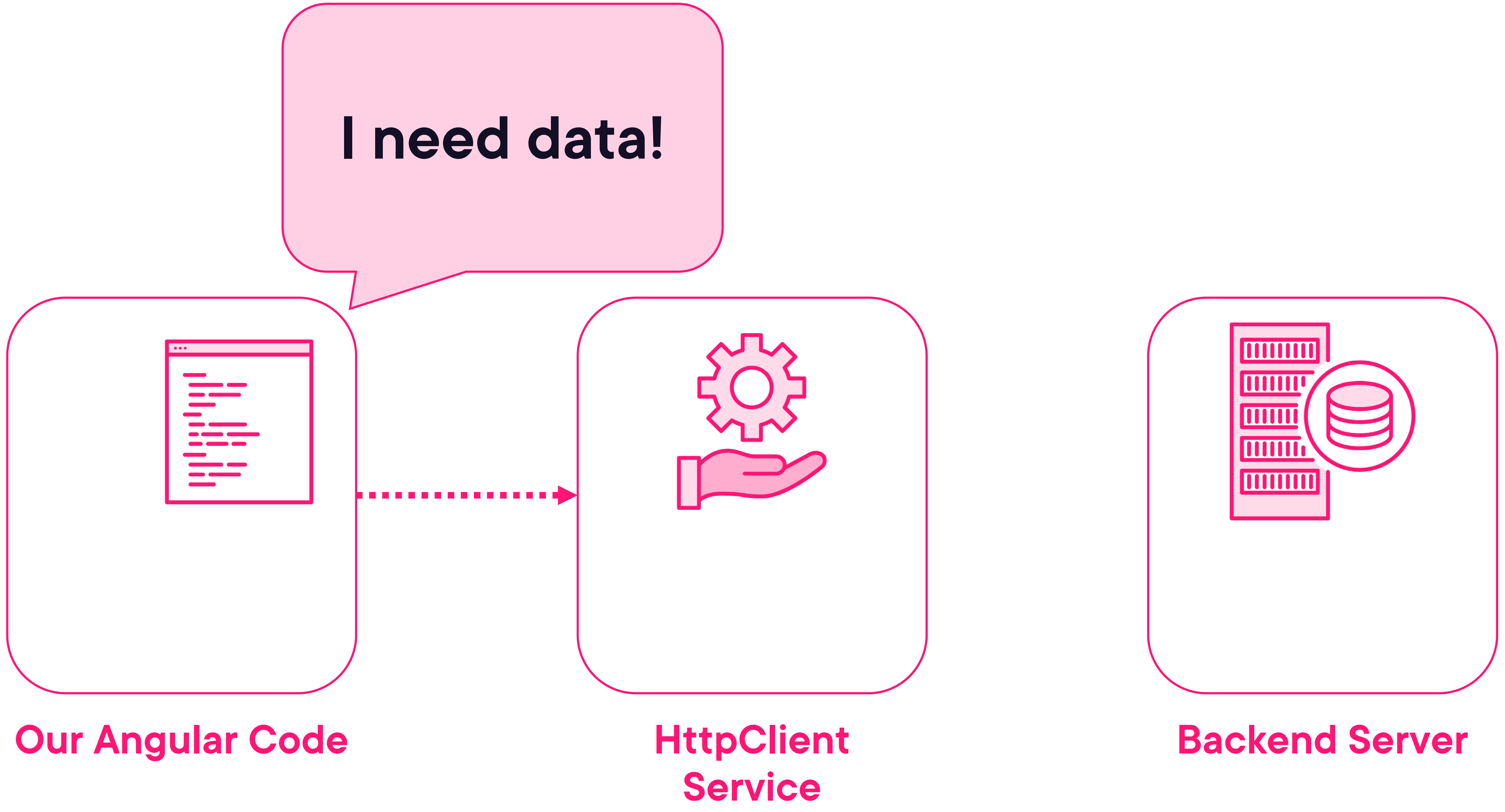
Retrieving Data with HTTP and Observables

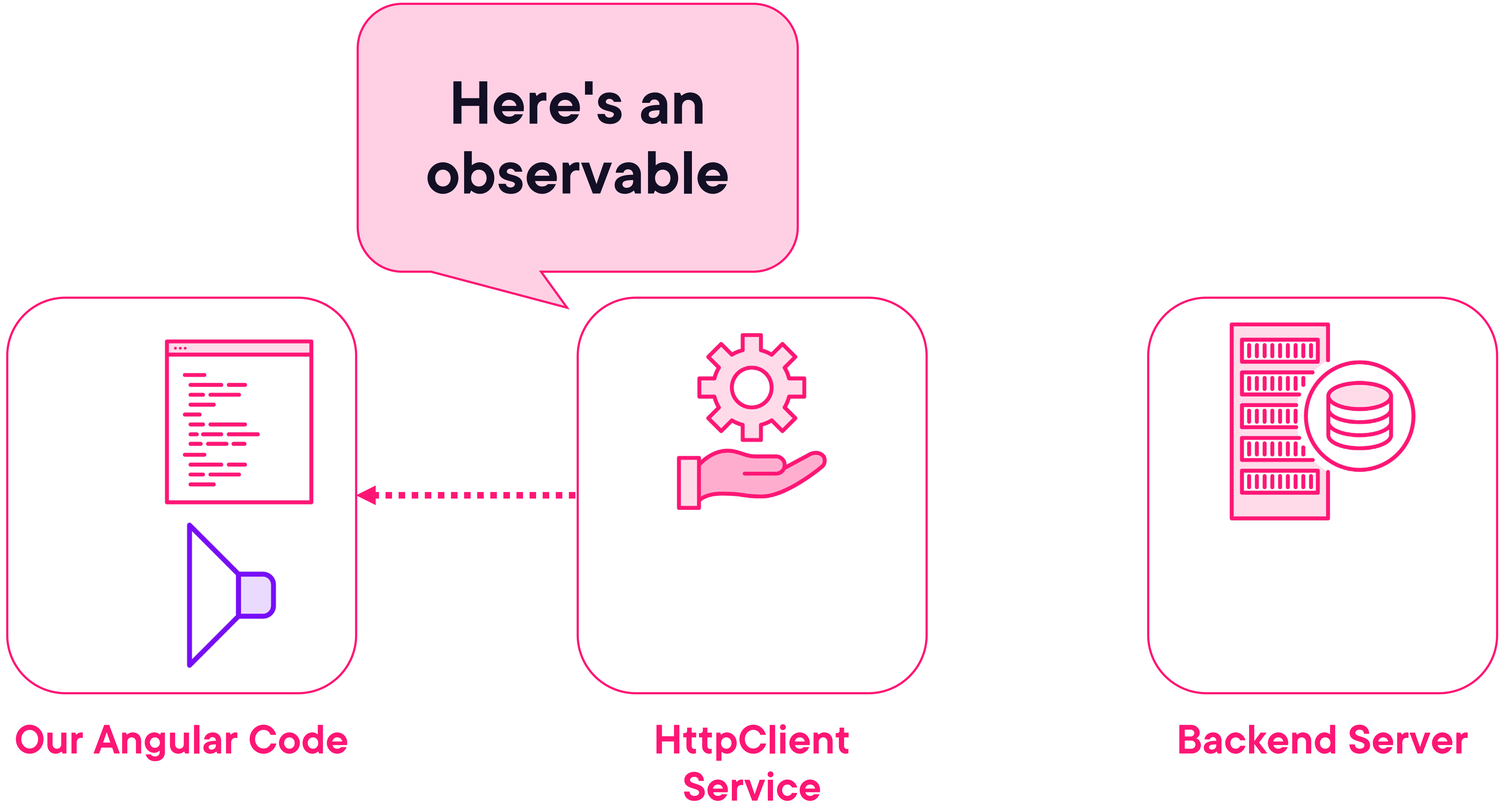
Deborah Kurata
Developer

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

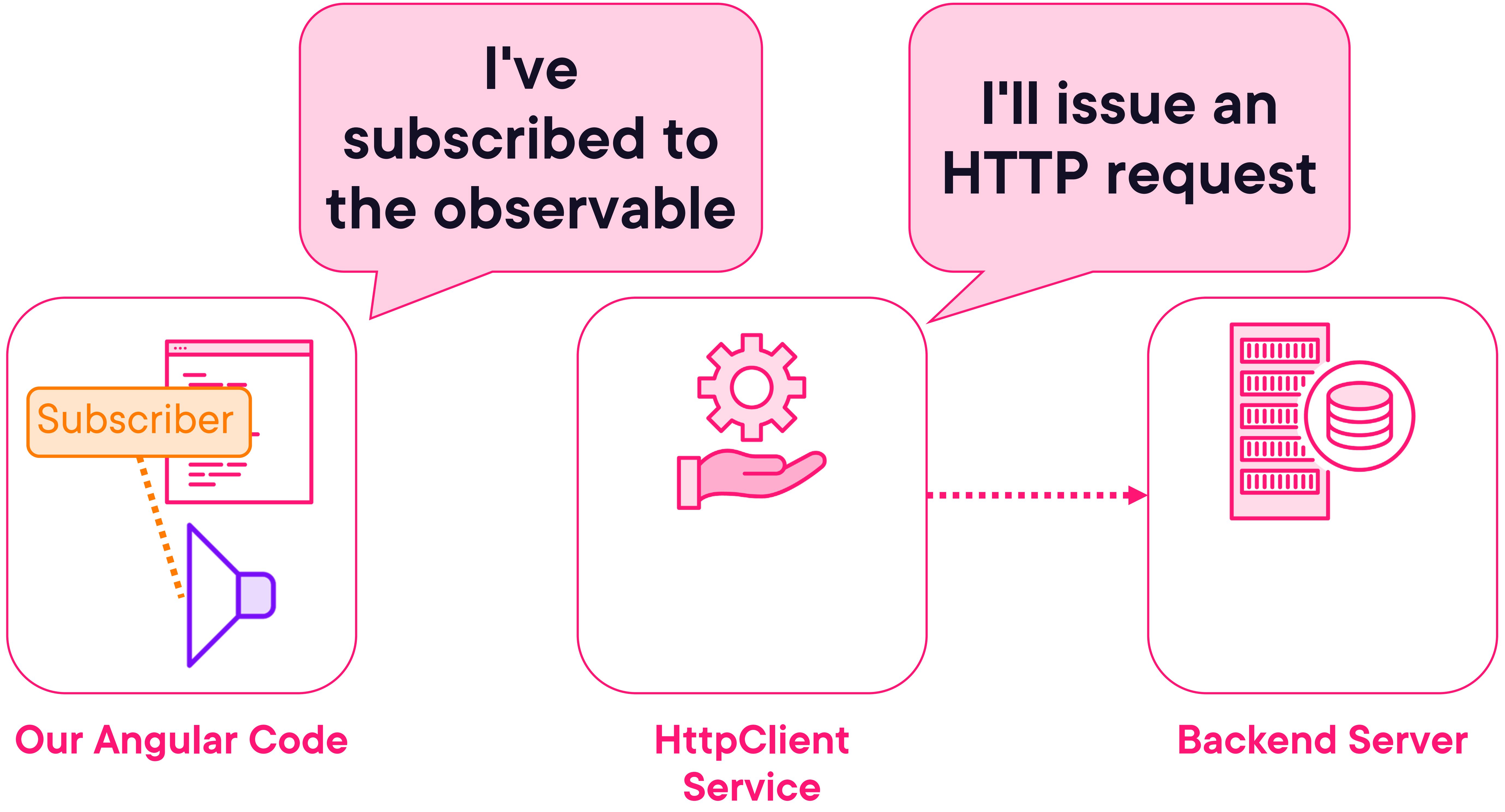
HttpClient Service



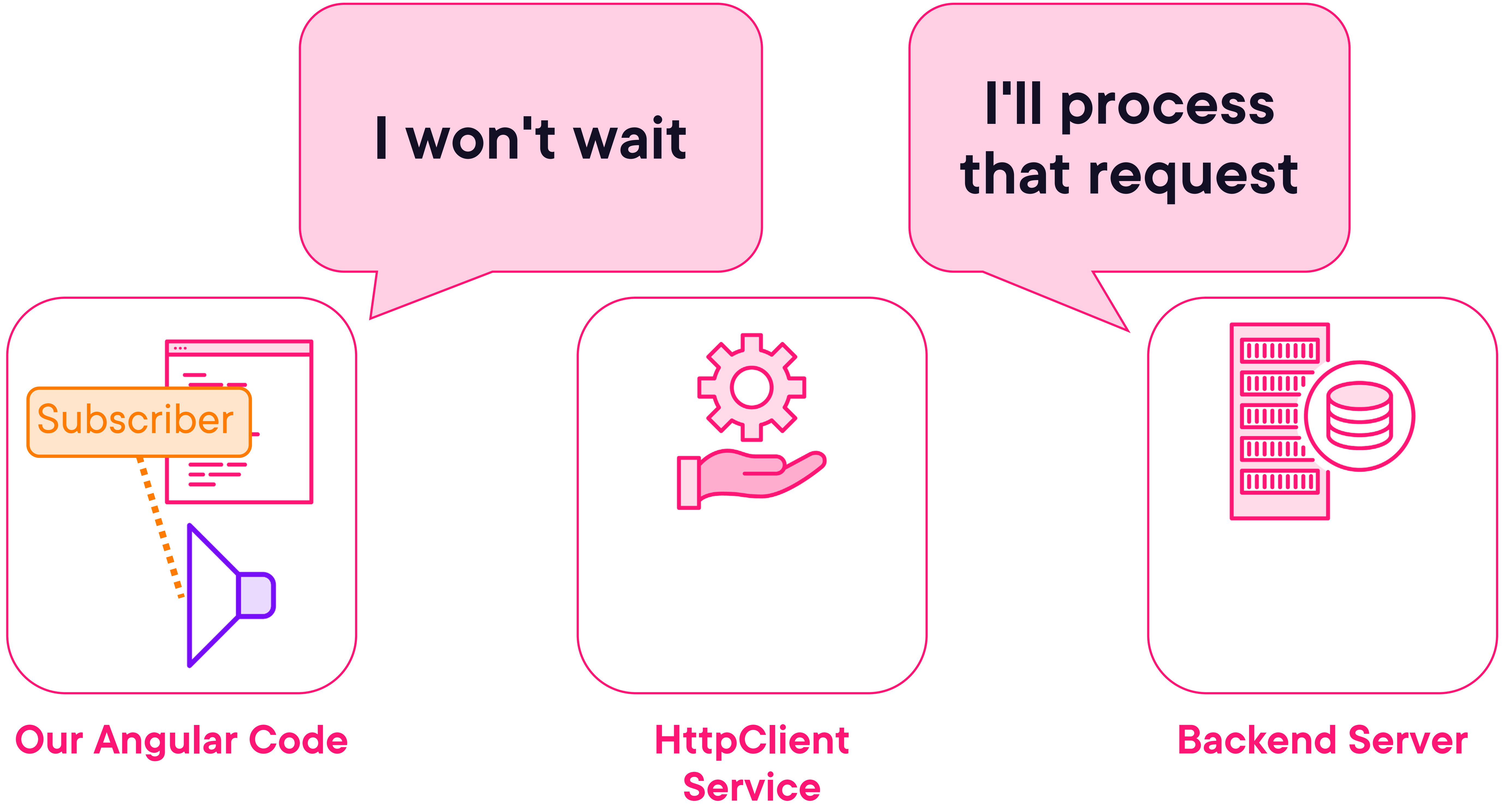
HttpClient Service



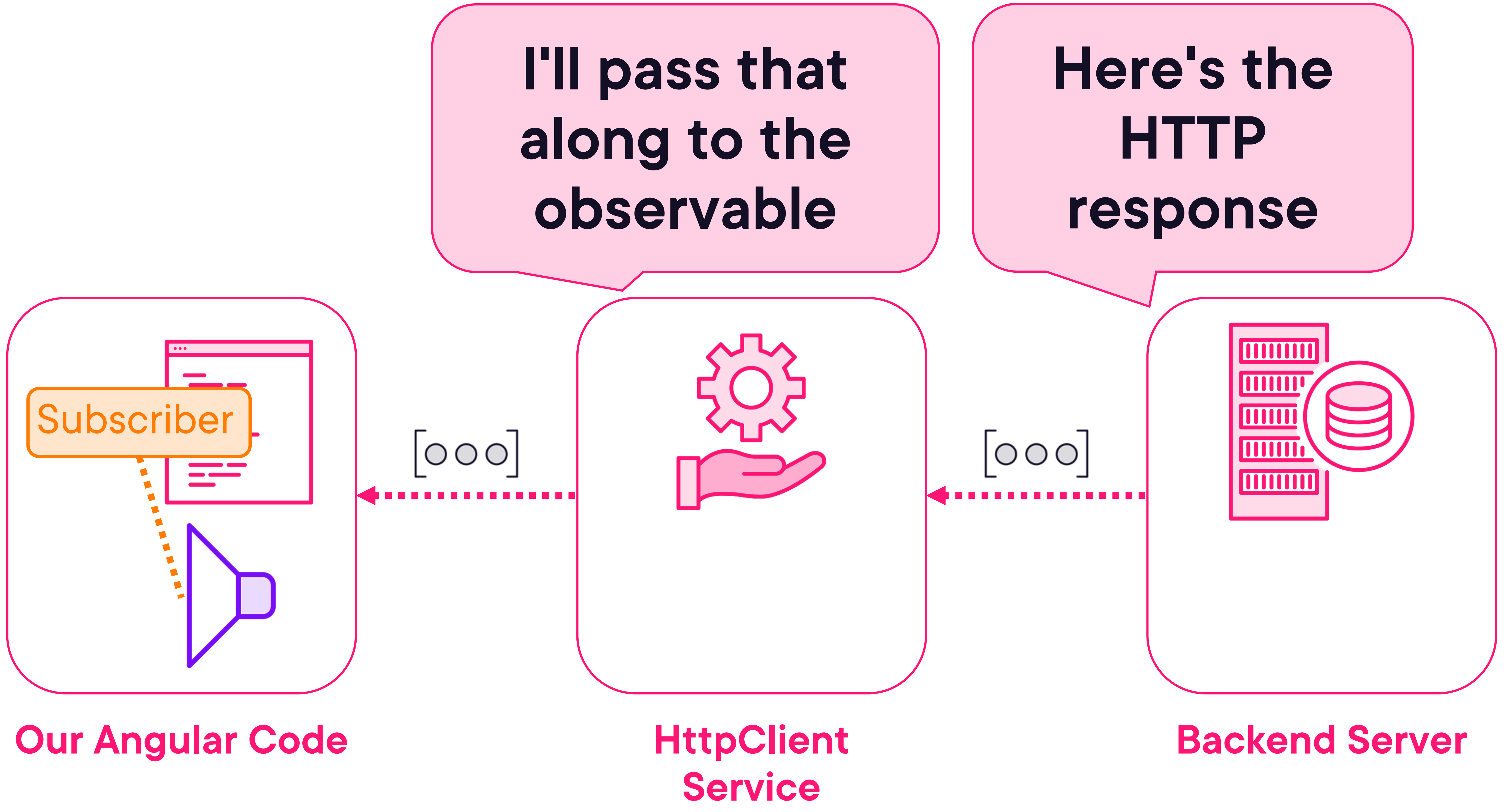
HttpClient Service



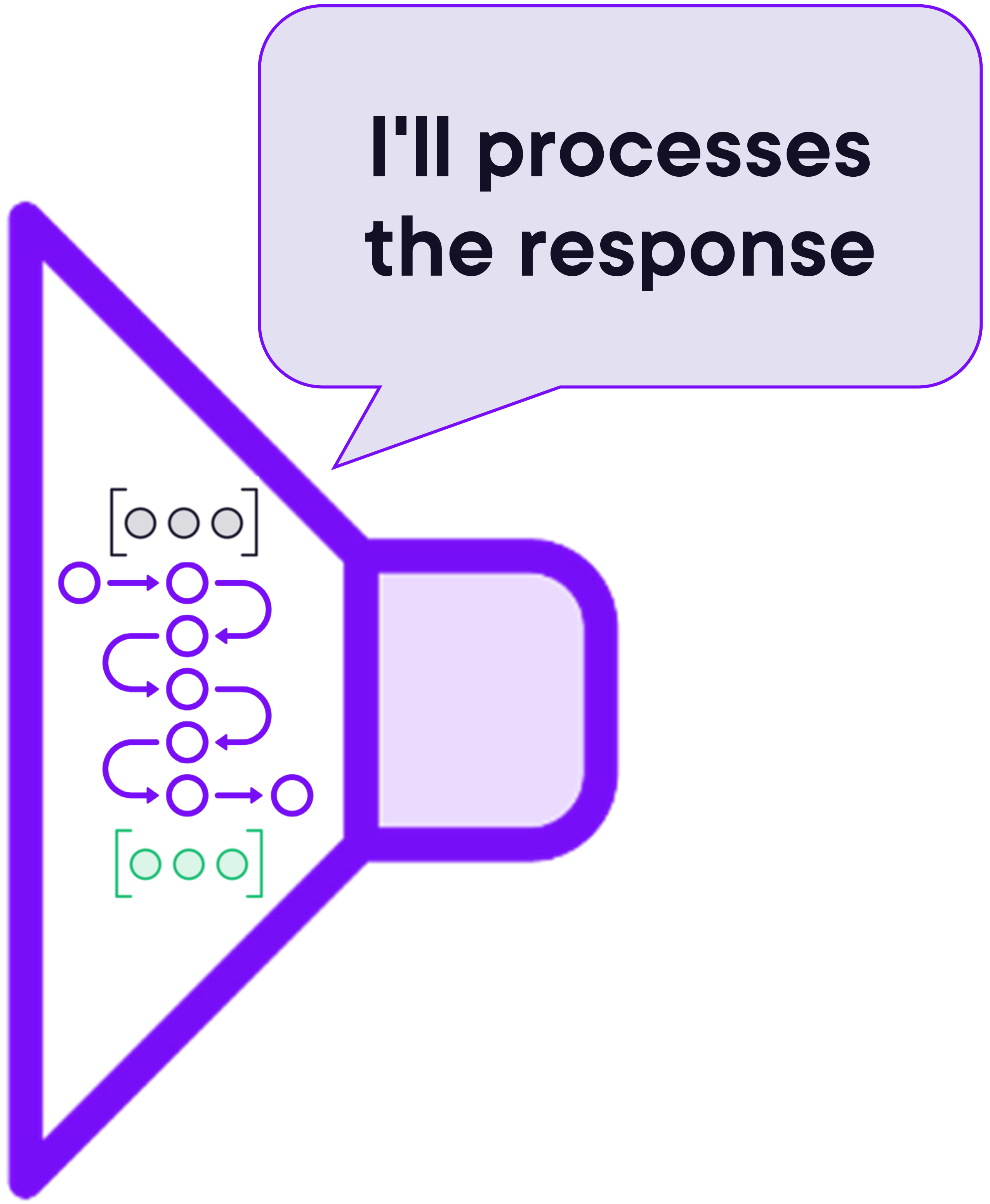
HttpClient Service



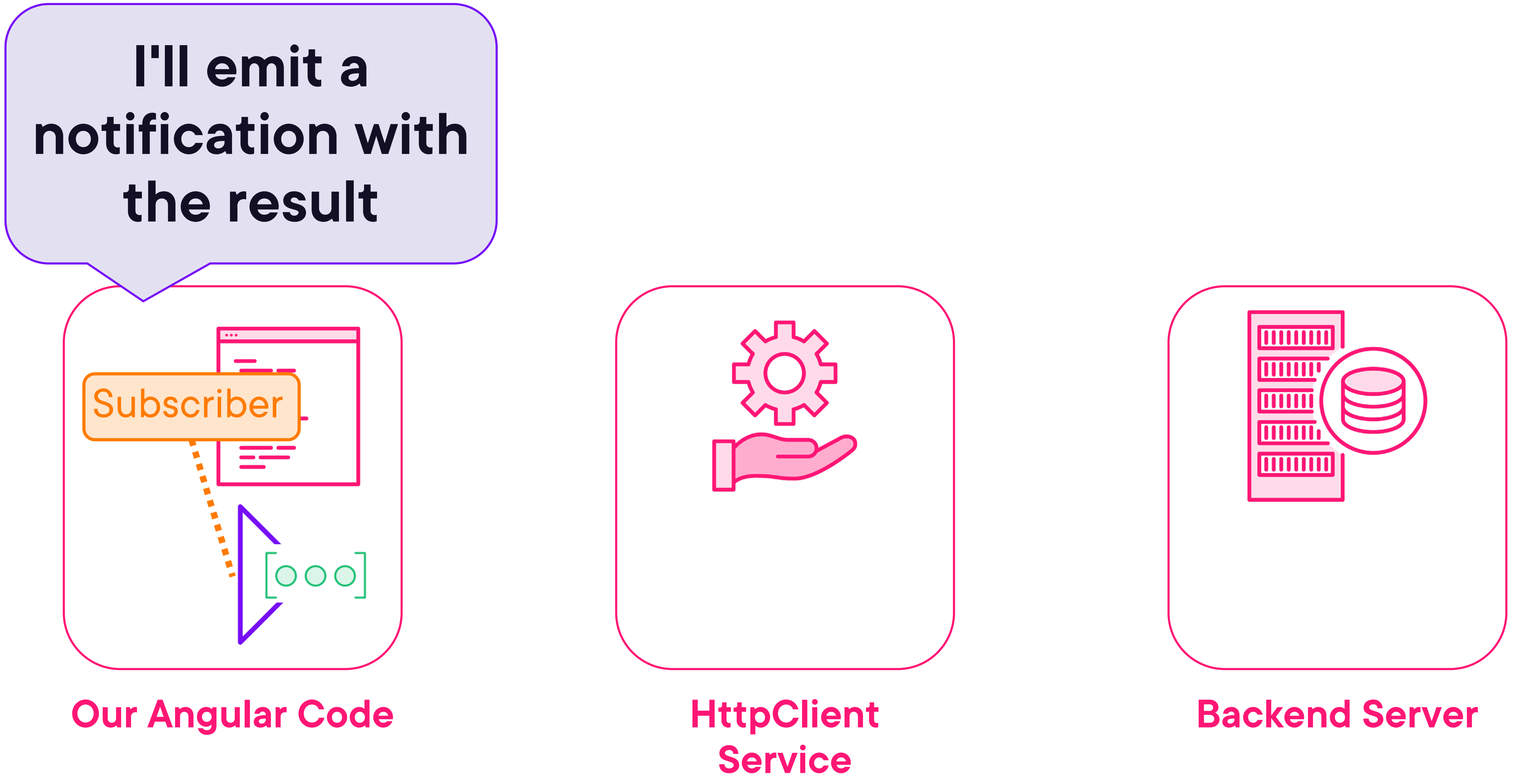
HttpClient Service



HttpClient Service



HttpClient Service



Benefits of Using Observables for HTTP Requests

Notifications

Callback functions

Operators

Error handling

Retry

Cancellation

Overview

Set up the sample application

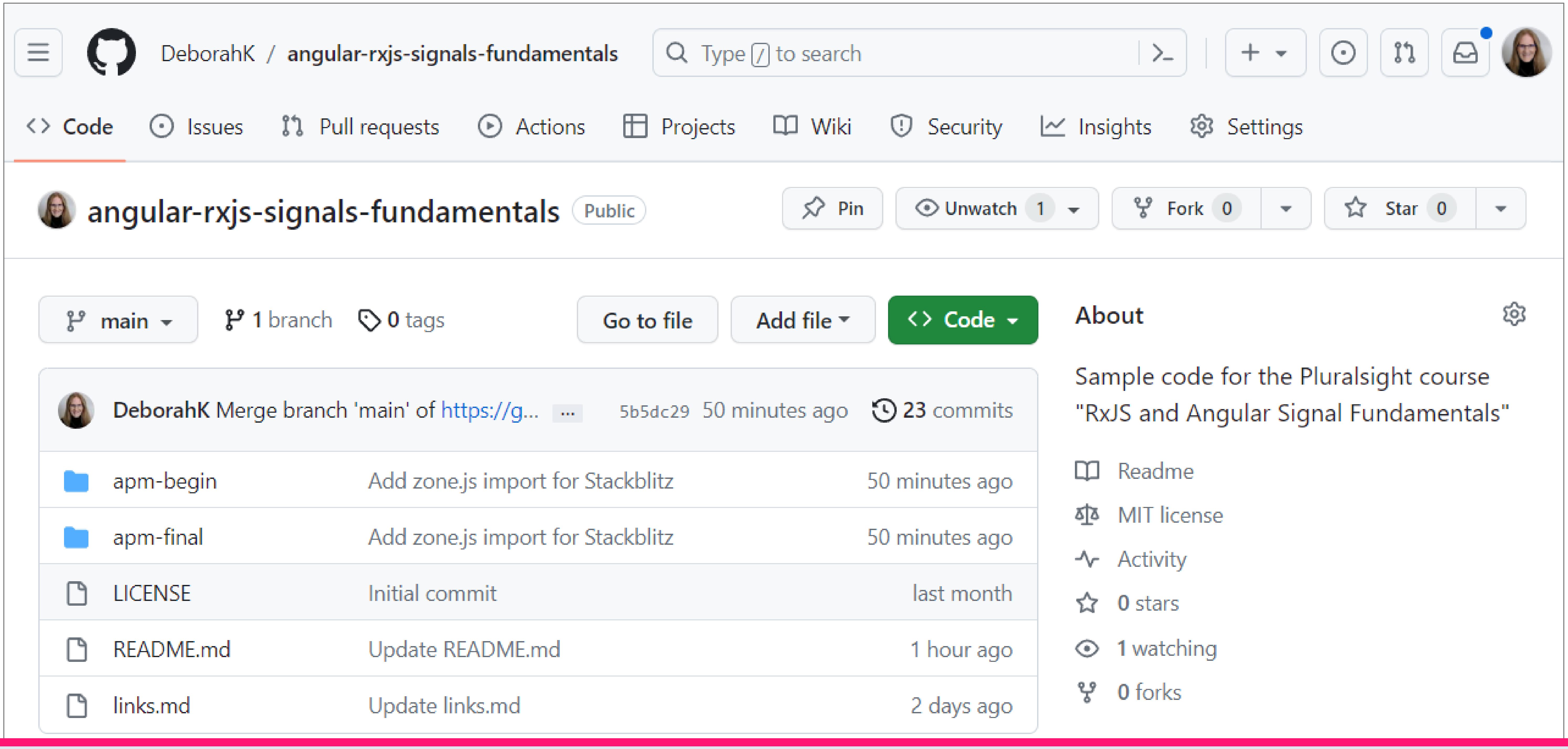
Dissect the code for retrieving data via HTTP and observables

Write code to retrieve data via HTTP

Demo

Overview of the sample application

GitHub Repository



angular-rxjs-signals-fundamentals Public

DeborahK Merge branch 'main' of https://g... 5b5dc29 50 minutes ago 23 commits

apm-begin	Add zone.js import for Stackblitz	50 minutes ago
apm-final	Add zone.js import for Stackblitz	50 minutes ago
LICENSE	Initial commit	last month
README.md	Update README.md	1 hour ago
links.md	Update links.md	2 days ago

About

Sample code for the Pluralsight course "RxJS and Angular Signal Fundamentals"

Readme
MIT license
Activity
0 stars
1 watching
0 forks

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

Coding Along (Optional)

Fork to create a copy of my GitHub repository
Clone that copy to your desktop

Download the code from GitHub as a **zip** file and unzip it

Use **StackBlitz**
https://stackblitz.com/github/DeborahK/angular-rxjs-signals

Gentle Introduction to Git/GitHub: https://youtu.be/pICJdbC7j0Q

Demo

Set up the sample application

- Stackblitz
- GitHub

Demo

Code walk through

Retrieving Data

```
getProducts(): Observable<Product[]> {
   return this.http.get<Product[]>(this.productUrl)
     .pipe(
       tap(data => console.log(data))
     );
 }
```

Key Point

Create a service to encapsulate HTTP requests

Why?

To share the retrieved data with any component or other service

Retrieving Data

```
getProducts(): Observable<Product[]> {
   return this.http.get<Product[]>(this.productUrl)
     .pipe(
       tap(data => console.log(data))
     );
 }
```

Retrieving Data

```
getProducts(): Observable<Product[]> {
   return this.http.get<Product[]>(this.productUrl)
     .pipe(
       tap(data => console.log(data))
     );
 }
```

```
 sub!: Subscription;
 products: Product[] = [];
```

```
this.sub = this.productService.getProducts().subscribe(
   products => this.products = products
);
```

Retrieving Data

```
getProducts(): Observable<Product[]> {
   return this.http.get<Product[]>(this.productUrl)
     .pipe(
       tap(data => console.log(data))
     );
 }
```

```
sub!: Subscription;
products: Product[] = [];
```

```
this.sub = this.productService.getProducts()
  .pipe(
    tap(data => console.log(data))
  ).subscribe(
     products => this.products = products
  );
```

Retrieving and Mapping Data

```
{
 "customers": [
   {
    "id": 1,
    "name": "microsoft",
    "address": "…"
   },
   {
    "id": 2,
    "name": "google",
    "address": "…"
   },
   {
    "id": 1,
    "name": "amazon",
    "address": "…"
   }
 ],
 "c
 "s
}
```

```
[
  {
    "id": 1,
    "name": "microsoft",
    "address": "…"
  },
  {
    "id": 2,
    "name": "google",
    "address": "…"
  },
  {
    "id": 1,
    "name": "amazon",
    "address": "…"
  }
```

```
getCustomers(): Observable<Customer[]> {
  return this.http.get<CustomerData>(this.url)
    .pipe(
      map(data => data.customers)
    );
}
```

Key Point

Strongly type the data and observable using the generic type parameters

```
getCustomers(): Observable<Customer[]> {
   return this.http.get<CustomerData>(this.url)
     .pipe(
       map(data => data.customers)
     );
 }
```

Why?

Minimizes code errors
Helps the compiler help us

Key Point

Take advantage of the observable pipeline

```
getCustomers(): Observable<Customer[]> {
   return this.http.get<CustomerData>(this.url)
     .pipe(
       map(data => data.customers)
     );
 }
```

Why?

To manipulate the item, handle errors, or gather related data before emitting the item

Demo

Retrieving data

- Retrieve all products (service)

Demo

Retrieving data

- Subscribe to the returned observable (component)

Demo

Retrieving data

- Retrieve a single product by id

Retrieving Data

HTTP request/response is asynchronous
- Issue an HTTP request
- Some time later, receive the response

Angular's HttpClient service is the intermediary
- Custom data service <-> backend server

When issuing an HTTP request, the HttpClient service returns an observable
- Subscribe to this observable

Returned response is emitted to the provided observable

Use the observable pipeline or observer
- React and process the emission

Retrieving Data (Procedural)

```
getProducts(): Observable<Product[]> {
   return this.http.get<Product[]>(this.productUrl)
     .pipe(
       tap(data => console.log(data))
     );
 }
```

```
this.sub = this.productService.getProducts()
  .pipe(
    tap(data => console.log(data))
  ).subscribe(
     products => this.products = products
  );
```

Observer vs. operator

```
this.sub = this.productService.getProducts()
  .pipe(
    tap(data => console.log(data))
  ).subscribe(
     products => this.products = products
  );
```

```
this.sub = this.productService.getProducts()
  .pipe(
    tap(data => this.products = products)
  ).subscribe();
```

Best Practices

Encapsulate HTTP requests in a service

Strongly type the data and observable

Take advantage of the observable pipeline

For More Information

Demo code

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

"Gentle Introduction to Git and GitHub"

- https://youtu.be/pICJdbC7j0Q

Angular documentation

- https://angular.io/guide/understanding-communicating-with-http

"RxJS Mapping: Mapping Retrieved Data"

- https://youtu.be/c7z-rsKcvZw

"Simplify with Angular Standalone Components"

- https://youtu.be/c8YGsPx0zVk

Up Next:

Handling HTTP Errors with Observables