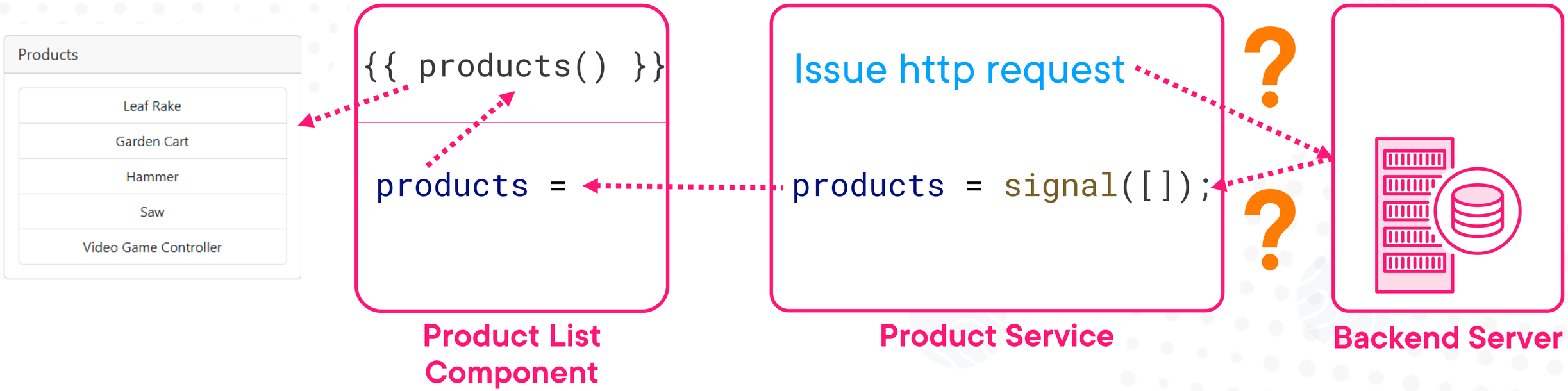
Retrieving Data into a Signal Using an Observable

Deborah Kurata

Developer | Content Creator | MVP | GDE

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

Retrieving Data



Retrieving Data (HttpClient)

Before Angular v19

Define http request
Subscribe to issue the request
Returns an observable

Observable emits returned data
Set emitted data into signal
Unsubscribe

```
products = signal([]);
```

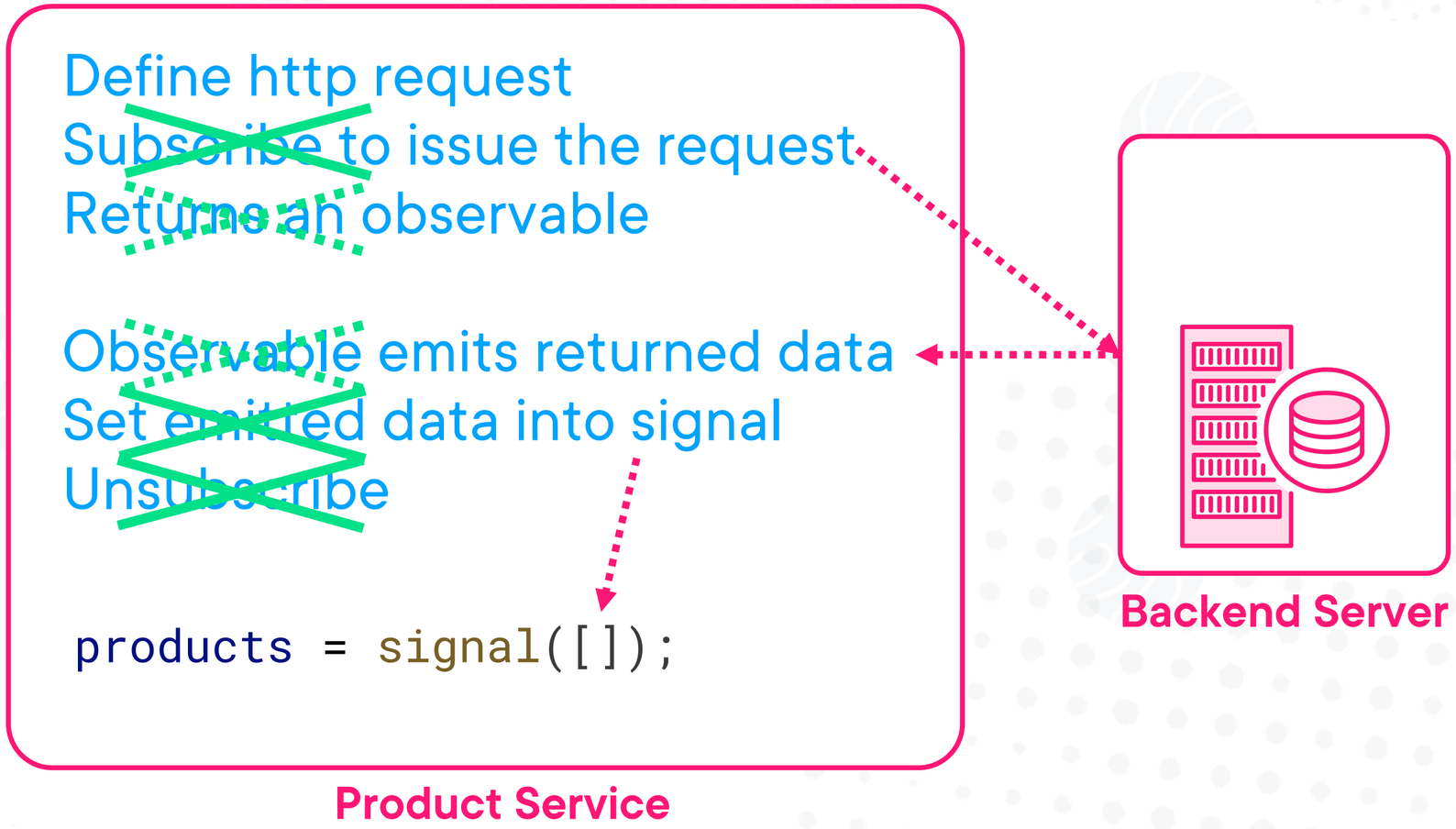
Product Service

Backend Server

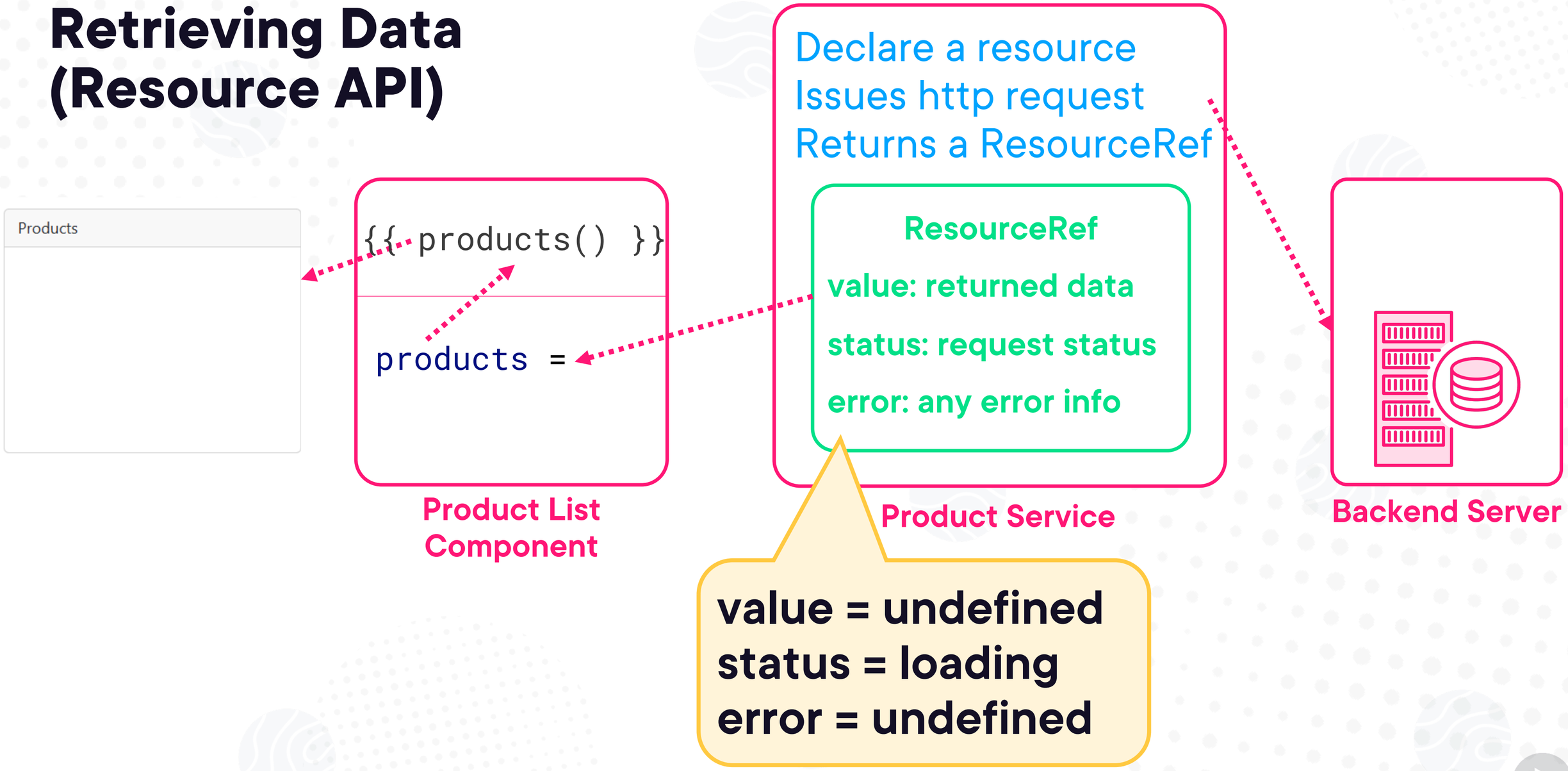
Retrieving Data (Resource API)

Before Angular v19

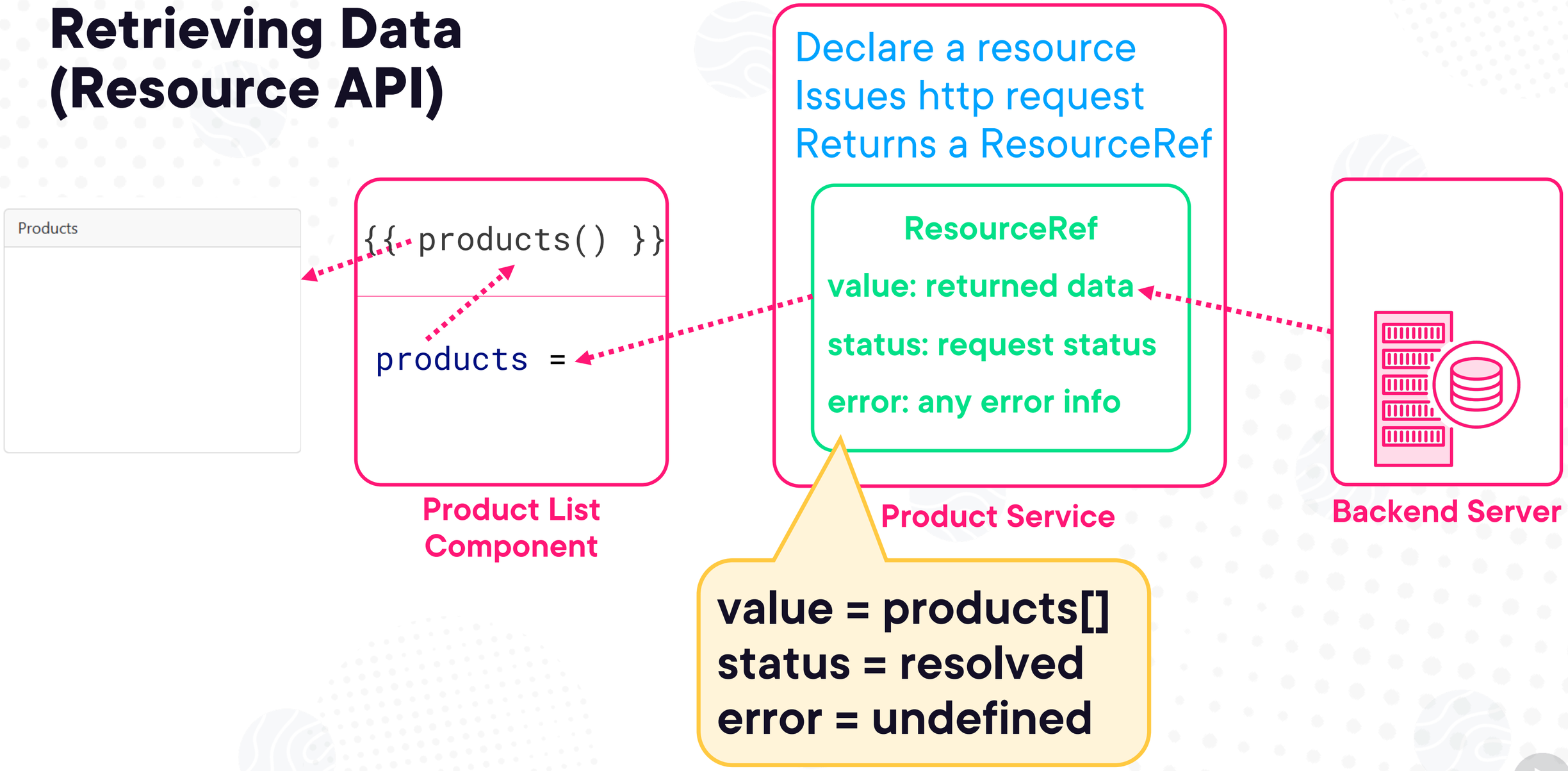
With the Resource API Angular v19+



Retrieving Data (Resource API)



Retrieving Data (Resource API)

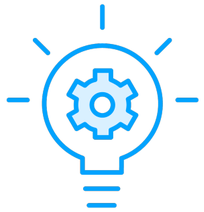


The Resource API is
experimental
and could change
before it's stable

Types of Resources

`resource()`: Lower level; works with promises

`httpResource()`: Works with a URL string



`rxResource()`: Works with observables

rxResource(): stream

```
productsResource = rxResource({
  stream: () => this.http.get<Product[]>(this.url)
});
```

```
products = this.productsResource.value;
error = this.productsResource.error;
```

rxResource(): defaultValue

```
productsResource = rxResource({
  stream: () => this.http.get<Product[]>(this.url) ,
  defaultValue: []
});
```

```
products = this.productsResource.value;
error = this.productsResource.error;
```

rxResource(): params

```
productResource = rxResource({
  params: this.selectedProduct,
  stream: (p) =>
    this.http.get<Product>(`${this.url}/${p.params?.id}`)
});
```

```
products = this.productsResource.value;
error = this.productsResource.error;
```

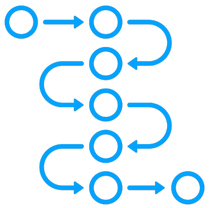
rxResource(): Pipeline

```
productResource = rxResource({
  params: this.selectedProduct,
  stream: (p) =>
    this.http.get<Product>(`${this.url}/${p.params?.id}`).pipe(
      map(p => ({
        ...p,
        productName: p.productName.toUpperCase()
      })),
      tap(p => console.log(JSON.stringify(p)))
    )
});
```

Retrieving Data Using `rxResource()`

Retrieve a single set of data

Reactively retrieve data when a signal changes



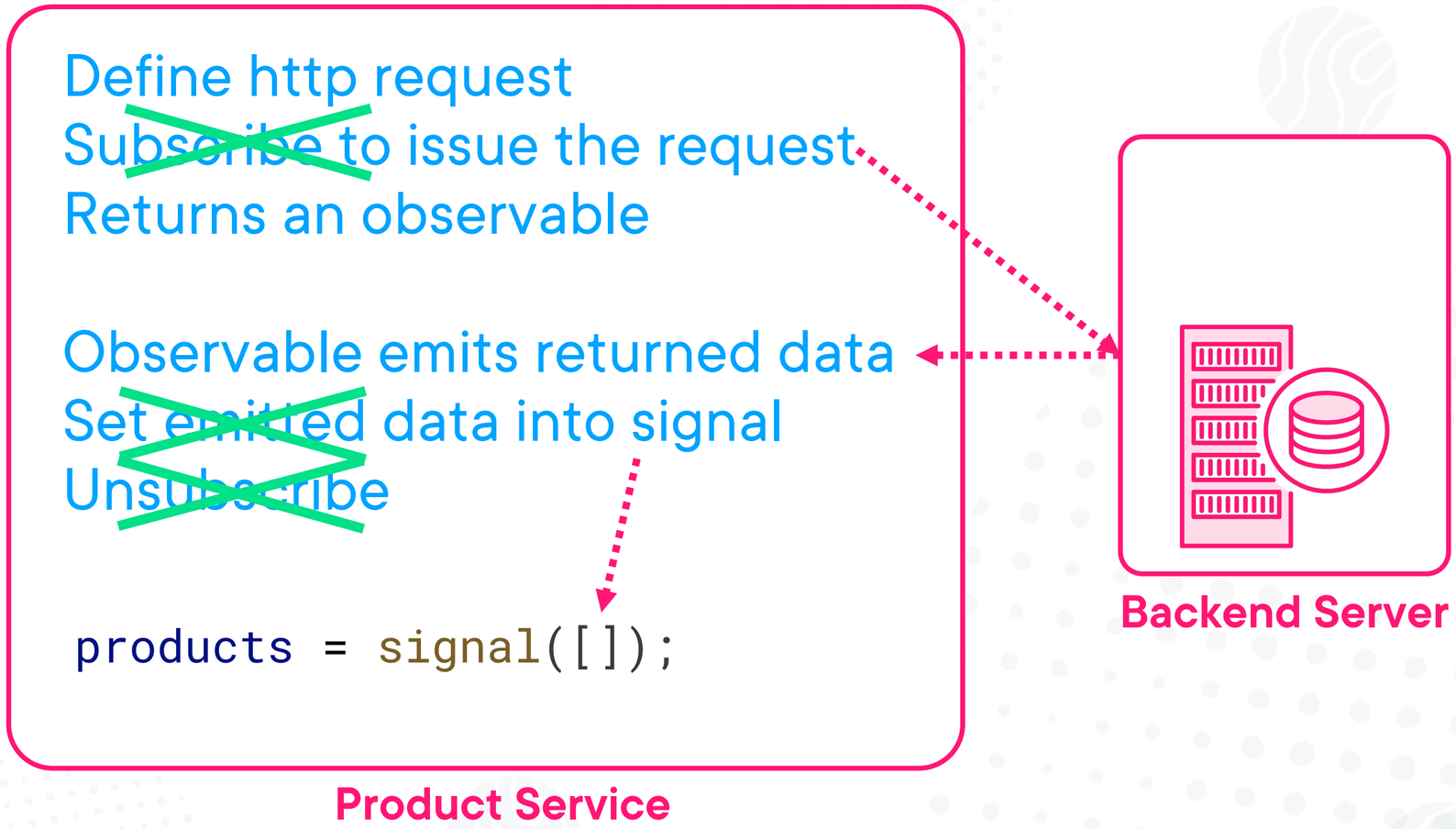
Use RxJS creation functions and pipeable operators to retrieve and compose data

Providing HttpClient (app.config.ts)

```
import { provideHttpClient } from '@angular/common/http';
...

export const appConfig: ApplicationConfig = {
  providers: [
    ...
    provideHttpClient()
  ]
};
```

rxResource() vs. HttpClient



Retrieving Data

`rxResource()`	VS.	`httpResource()`
Pass in a function that returns an observable		Pass in a function that returns a string URL
Provides value, isLoading, and error signals		Provides value, isLoading, and error signals
Build a pipeline of operators		No pipeline; use computed properties
Fits in well if you already have observables		

GitHub

https://github.com/DeborahK/angular-rxjs-ps-course

Beginning sample application files:

apm-begin

Final (completed) sample application files:

apm-end

File with clickable links to additional information:

MOREINFO.md