Caching Observables

Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

Alarm
Net
HDD
IR
Esc
Fn
Enter
100-240VAC

Caching Observables

Retain retrieved data locally

Reuse previously retrieved data

Stored in memory or external

Module Overview

Why caching?

Patterns for data caching

RxJS Features

`shareReplay`

Retrieving Data

Product List Component

```
constructor(private productService: ProductService) { }

ngOnInit() {
 this.productService.getProducts()
  .subscribe(
    products => this.products = products
  );
}
```

Product List Template

```
<div *ngIf="products$ | async as products">
<table>
  <tr *ngFor="let product of products">
    <td>{{ product.productName }}</td>
    <td>{{ product.productCode }}</td>
  </tr>
</table>
```

Advantages of Caching Data

Improves responsiveness

Reduces bandwidth and network consumption

Reduces backend server load

Reduces redundant computations

Classic Caching Pattern

Product Service

```
private products: Product[];

getProducts(): Observable<Product[]> {
  if (this.products) {
    return of(this.products);
  }
  return this.http.get<Product[]>(this.productsUrl)
    .pipe(
      tap(data => this.products = data),
      catchError(this.handleError)
    );
}
```

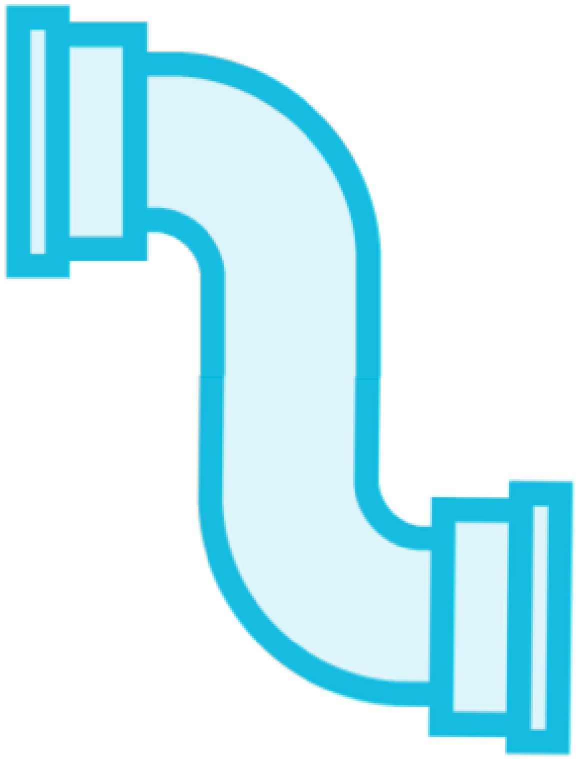
Declarative Caching Pattern

Product Service

```
private productsUrl = 'api/products';

products$ = this.http.get<Product[]>(this.productsUrl)
  .pipe(
    shareReplay(1),
    catchError(this.handleError)
  );
```

RxJS Operator: `shareReplay`



Shares the stream with other subscribers

Replays the defined number of emissions on subscription

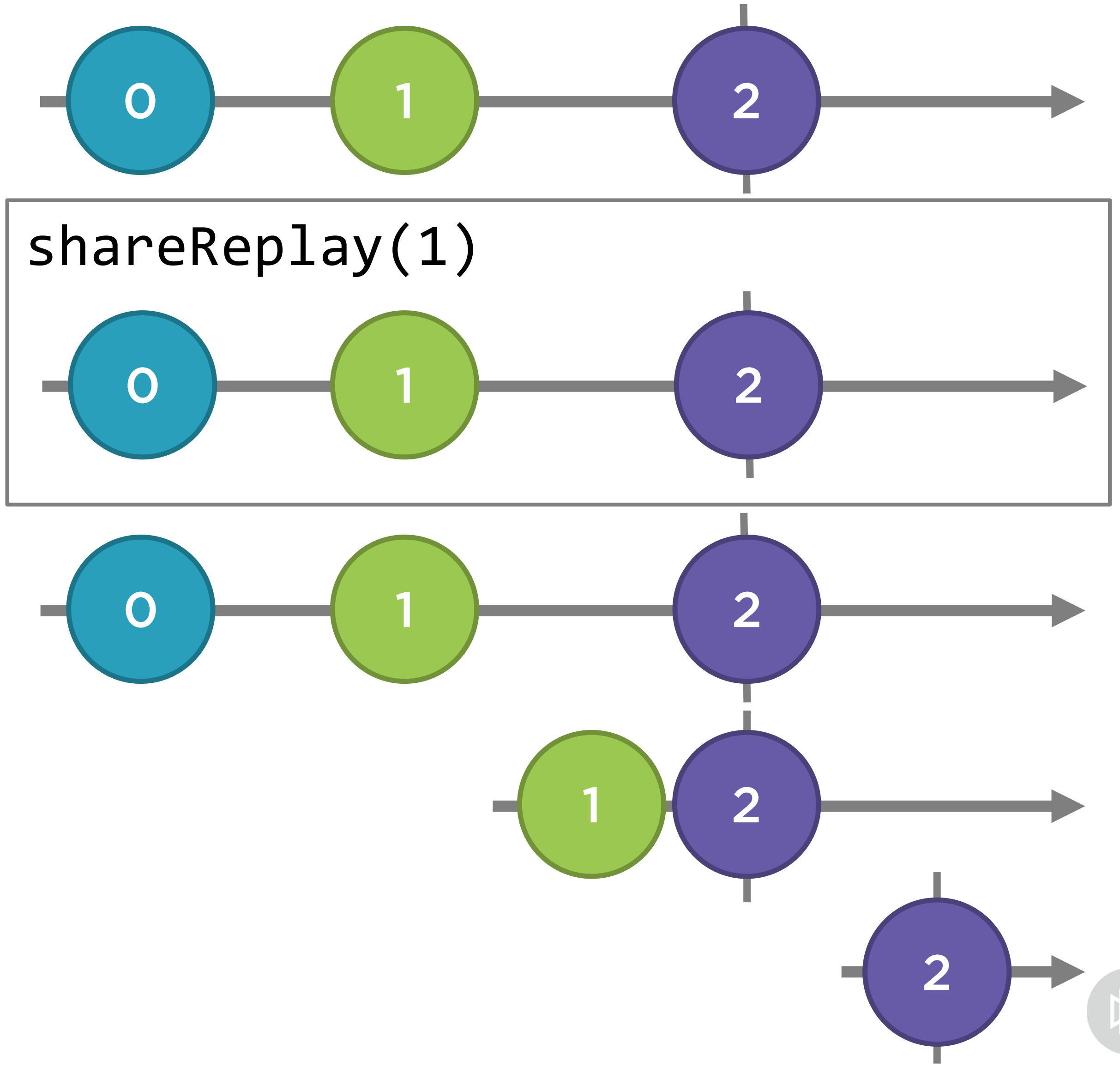
```
shareReplay(1)
```

Used for

- Caching data in the application

Marble Diagram: shareReplay

```
a$ = interval(1000)
 .pipe(
  take(3),
  shareReplay(1)
 )
```



```
a$.subscribe(console.log);
```

```
a$.subscribe(console.log);
```

```
a$.subscribe(console.log);
```

RxJS Operator: shareReplay

`shareReplay` is a multicast operator

Returns a Subject that shares a single subscription to the underlying source

Takes in an optional buffer size, which is the number of items cached and replayed

On a subscribe, it replays a specified number of emissions

The items stays cached forever, even after there are no more subscribers

Demo

Caching data with `shareReplay`

Caching Observables

Use `shareReplay` on any stream you wish to share and replay to all new subscribers

```
productCategories$ = this.http.get<ProductCategory[]>(this.url)
  .pipe(
    tap(data => console.log('categories', data)),
    shareReplay(1),
    catchError(this.handleError)
  );
```

Cache Invalidation

Evaluate:

- Fluidity of data
- Users' behavior

Consider

- Invalidating the cache on a time interval
- Allowing the user to control when data is refreshed
- Always getting fresh data on update operations