Reactively Retrieving Data

Deborah Kurata

Developer | Content Creator | MVP | GDE

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

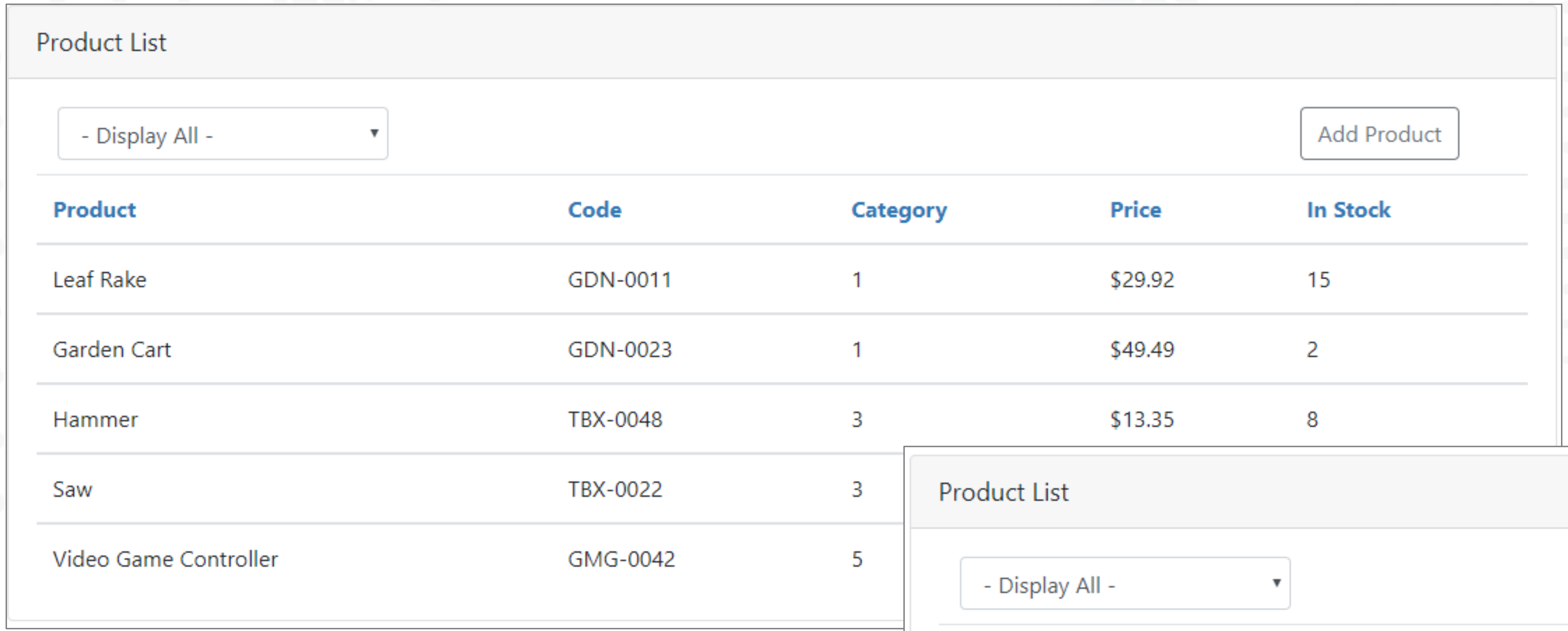
Difficult to retrieve?

Required multiple data sets?

Arrived in a format challenging to work with?

This is where RxJS shines!

Map Category Ids to Category Names

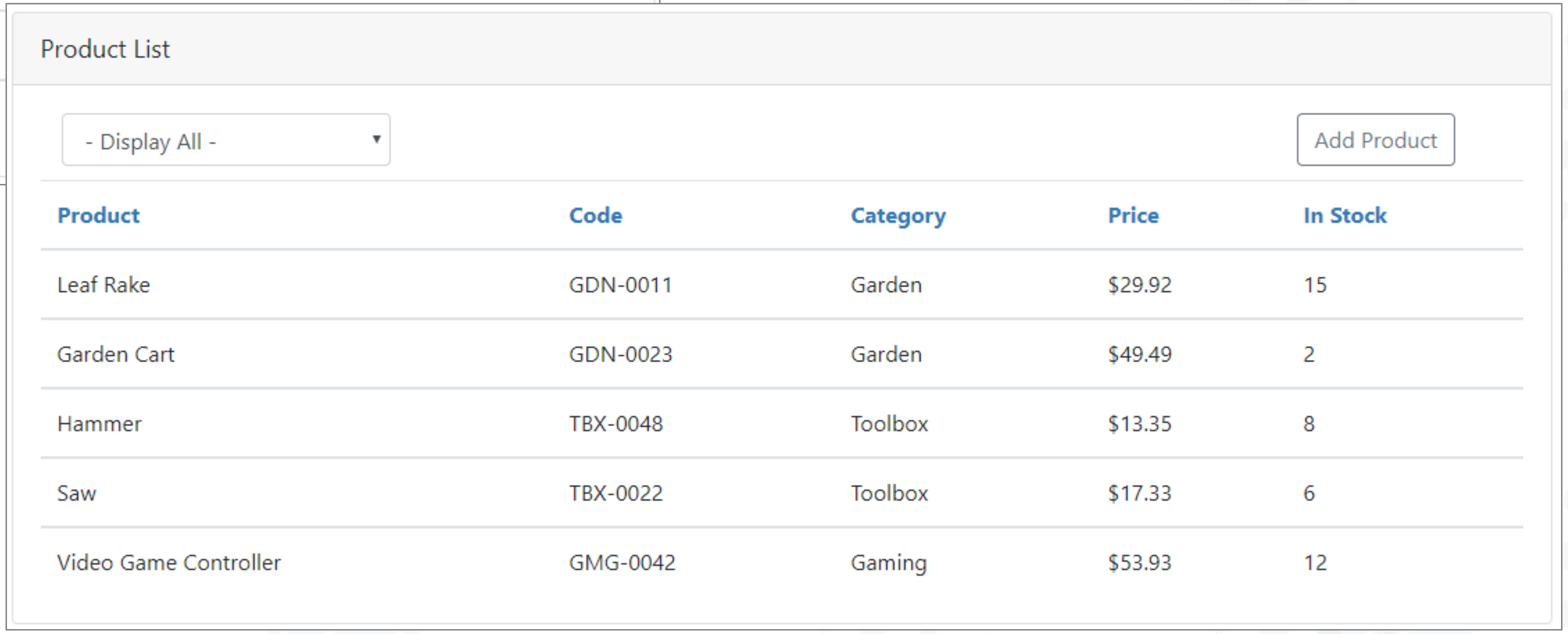


Product List

- Display All -

Add Product

Product	Code	Category	Price	In Stock
Leaf Rake	GDN-0011	1	$29.92	15
Garden Cart	GDN-0023	1	$49.49	2
Hammer	TBX-0048	3	$13.35	8
Saw	TBX-0022	3		
Video Game Controller	GMG-0042	5		



Product List

- Display All -

Add Product

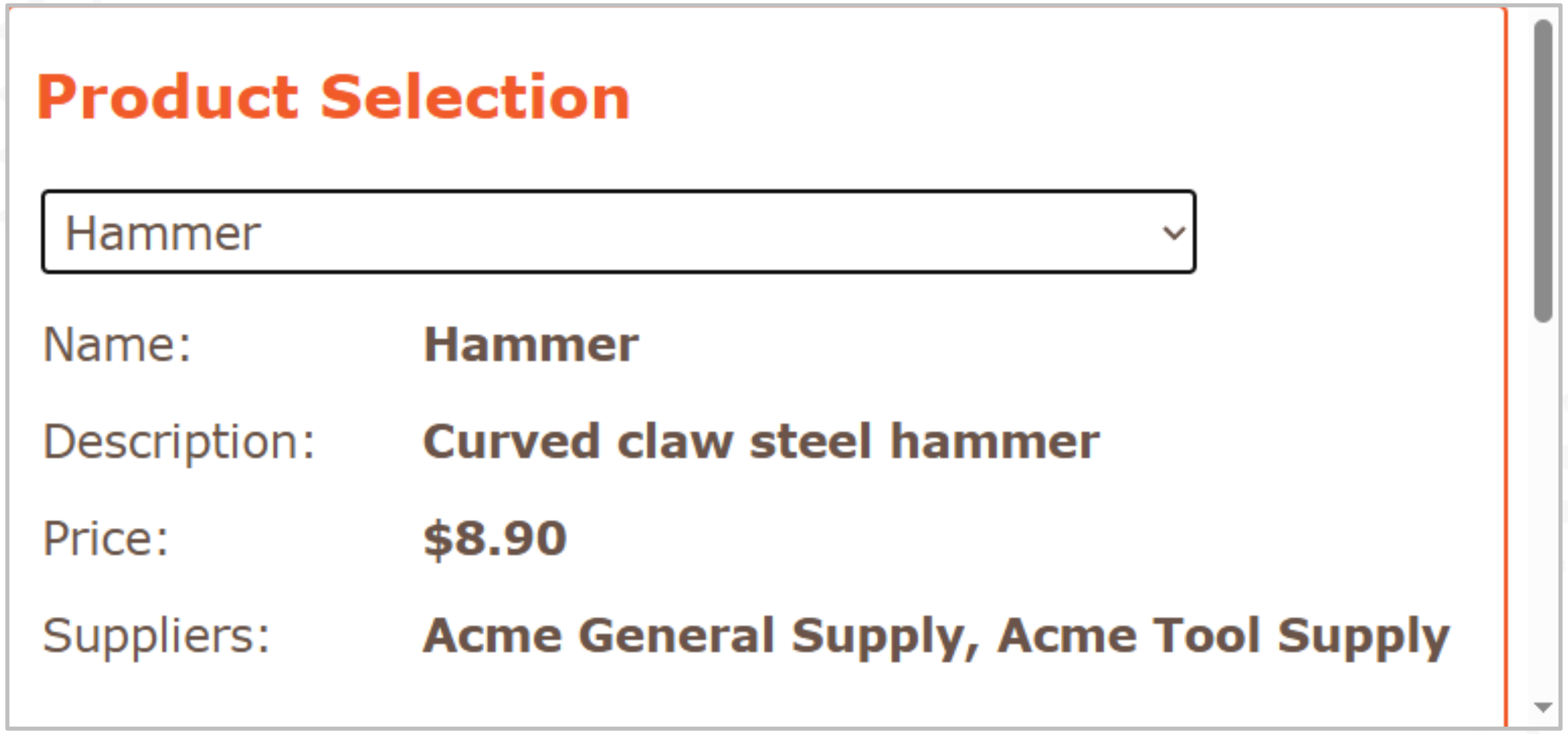
Product	Code	Category	Price	In Stock
Leaf Rake	GDN-0011	Garden	$29.92	15
Garden Cart	GDN-0023	Garden	$49.49	2
Hammer	TBX-0048	Toolbox	$13.35	8
Saw	TBX-0022	Toolbox	$17.33	6
Video Game Controller	GMG-0042	Gaming	$53.93	12

Map Category Ids to Category Names

```
productsWithCategory$ = combineLatest([
  this.http.get<Product[]>('/api/products'),
  this.http.get<Category[]>('/api/categories')
]).pipe(
  map(([products, categories]) =>
    products.map(product => ({
      ...product,
      category: categories.find(c => product.categoryId === c.id)?.name
    } as Product))
  )
);
```

```
productsResource = rxResource({
  stream: () => this.productsWithCategory$,
  defaultValue: []
});
```

Display Suppliers


Product Selection

Hammer

Name:	Hammer
Description:	Curved claw steel hammer
Price:	$8.90
Suppliers:	Acme General Supply, Acme Tool Supply

Suppliers

```
export interface Product {
  id: number;
  productName: string;
  description: string;
  price: number;
  supplierIds?: number[];
}
```

```
export interface Supplier {
  id: number;
  name: string;
  cost: number;
  minQuantity: number;
}
```

forkJoin()

```
supplierIds = computed(() =>
  this.productService.selectedProduct()?.supplierIds);

suppliers = forkJoin(this.supplierIds().map(
  supplierId => this.http.get<Supplier>(`${this.url}/${supplierId}`)
));
```

```
[3,6,12]
```

```
[ {id: 3, name: 'SYS'}
  {id: 6, name: 'Acme'}
   {id:12, name: 'ABC'} ]
```

RxJS + `rxResource()`
Transform and compose complex sets of data

Use the Resource API to …

Retrieve directly into a signal

Access resource properties (isLoading, error)

Skip subscribing and unsubscribing

Reactively retrieve data when dependent signals change

```
productsResource = rxResource({
  stream: () => this.http.get<Product[]>(this.url).pipe(
    map(items => items.sort((a, b) =>
      a.productName < b.productName ? -1 : 0))
  ),
  defaultValue: []
});

productsResourceHTTP = httpResource<Product[]>(() =>
  this.url, { defaultValue: [] }
);
```

```
productsResource = rxResource({
  stream: () => this.http.get<Product[]>(this.url).pipe(
    map(items => items.sort((a, b) =>
      a.productName < b.productName ? -1 : 0))
  ),
  defaultValue: []
});
```

Select rxResource() ...

To process retrieved data using an observable pipeline

When working with complex data

If you have existing observables

Use HttpClient Instead

```
products$ = this.http.get<Product[]>(this.url).pipe(
  startWith([]),
  catchError(error => {
    console.error('Error loading products', error);
    return of([]);
  })
);
```

When you cannot use experimental techniques

- The Resource API is currently experimental

To issue POST, PUT, or other mutable HTTP requests

- The Resource API is not suitable for mutable requests

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