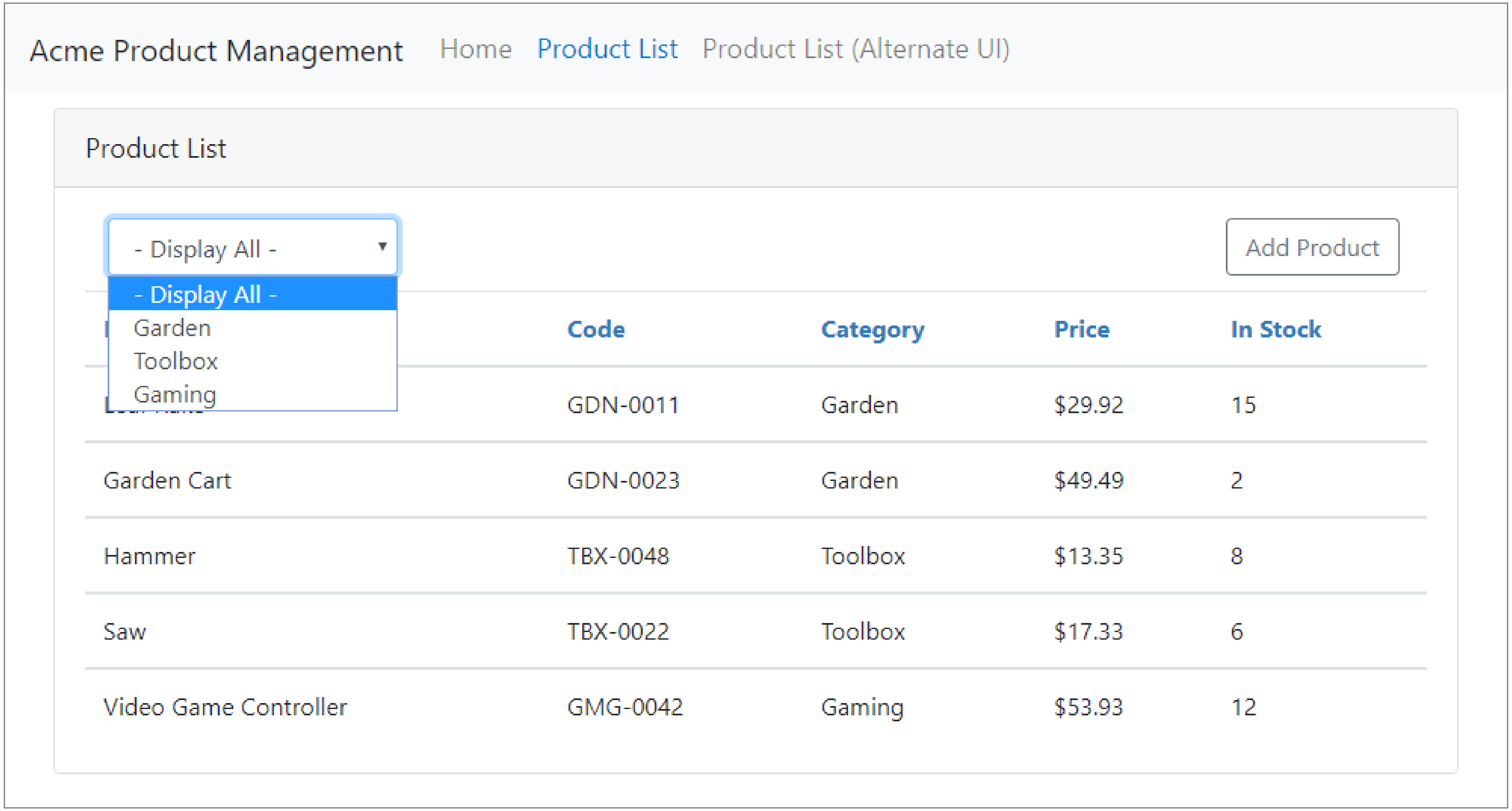
Reacting to Actions

Deborah Kurata

CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/



Acme Product Management Home Product List Product List (Alternate UI)

Product List

[illegible]	Code	Category	Price	In Stock
[illegible]	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

Module Overview

Filtering a stream

Data stream vs. action stream

Subject and BehaviorSubject

Reacting to actions

Starting with an initial value

RxJS Features

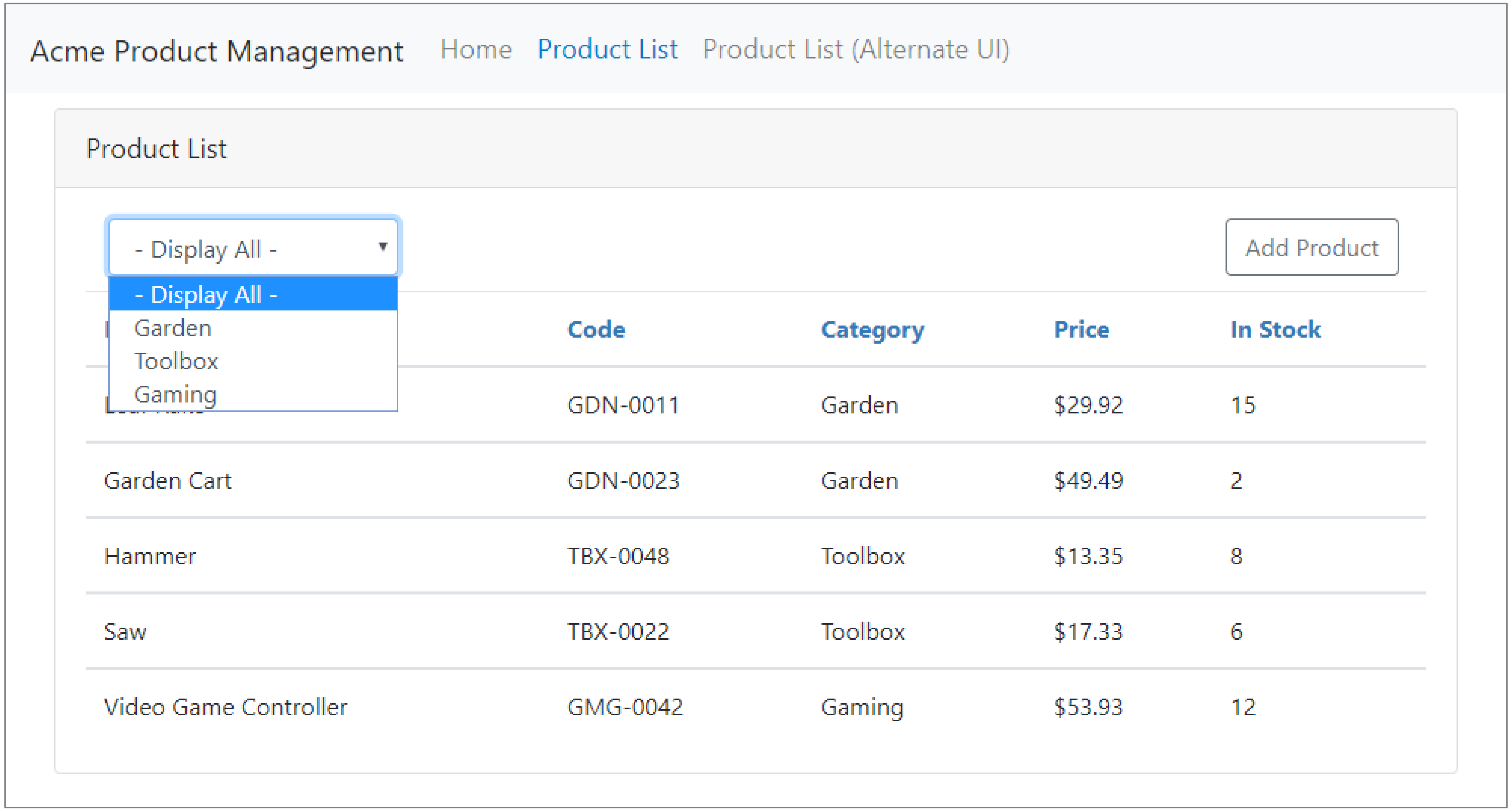
`filter`

`startWith`

`Subject`

`BehaviorSubject`

Filtering a Stream



Acme Product Management   Home   Product List   Product List (Alternate UI)

Product List

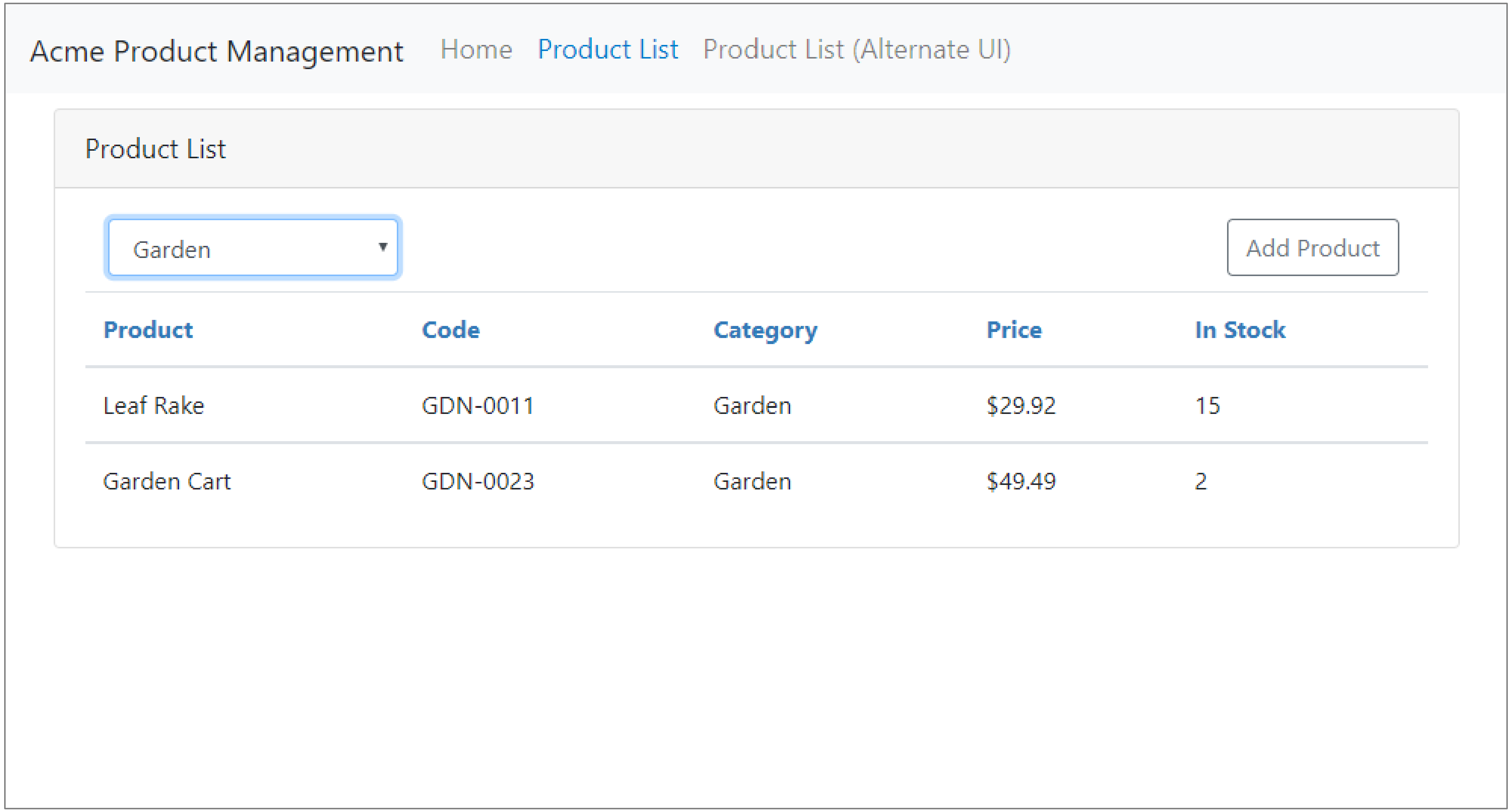
- Display All -

- Display All -
Garden
Toolbox
Gaming

Add Product

	Code	Category	Price	In Stock
[illegible]	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

Filtering a Stream

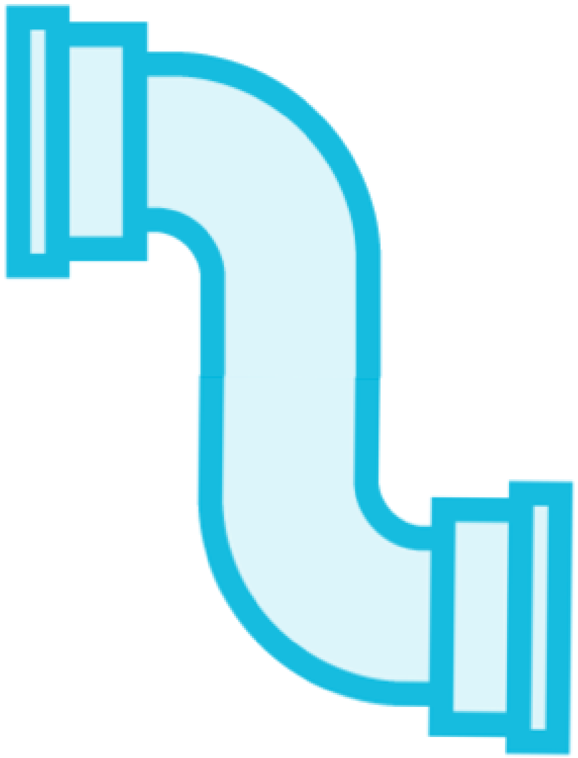


Product List

Garden

Product	Code	Category	Price	In Stock
Leaf Rake	GDN-0011	Garden	$29.92	15
Garden Cart	GDN-0023	Garden	$49.49	2

RxJS Operator: `filter`



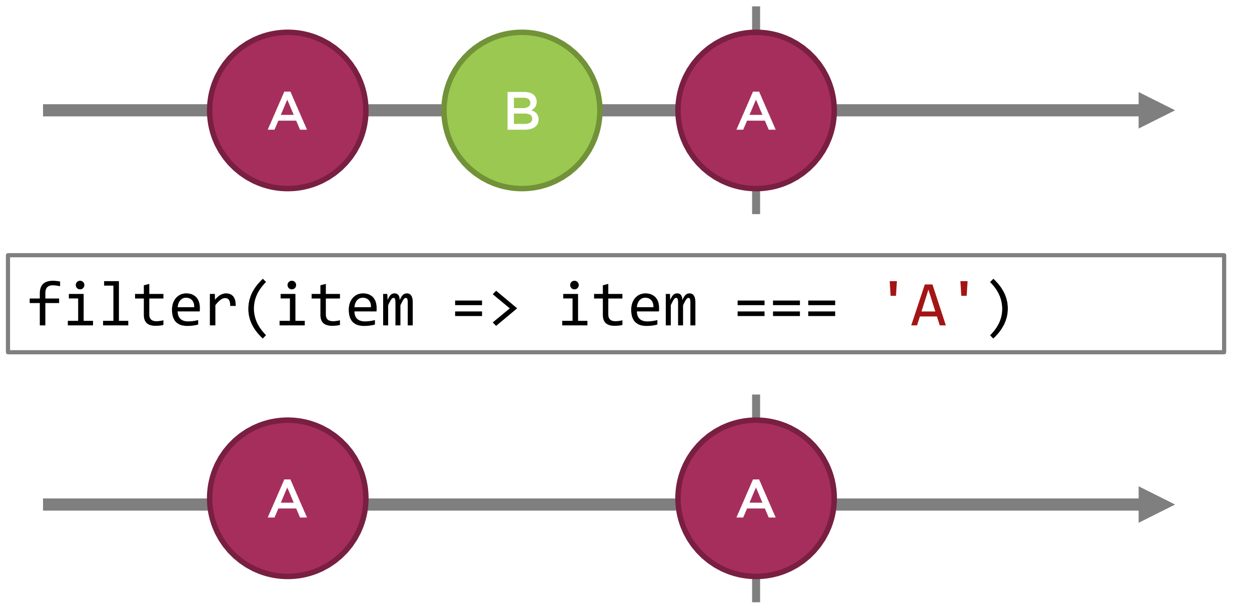
Filters to the items that match criteria specified in a provided function

```
filter(item => item === 'Apple')
```

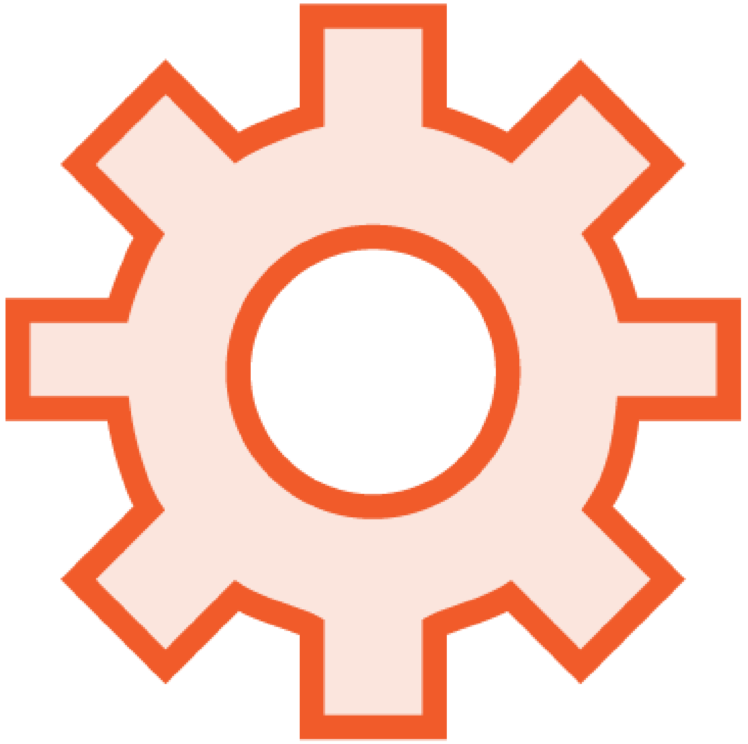
Similar to the array filter method

Marble Diagram: `filter`

```
of('A', 'B', 'A')
  .pipe(
    filter(item => item === 'A'),
  );
```



RxJS Operator: `filter`



`filter` is a transformation operator

- Takes in an input stream, subscribes
- Creates an output stream

When a source item is emitted

- Item is evaluated as specified by the provided function
- If the evaluation returns true, item is emitted to the output stream

Demo

Filtering a stream: Demo I

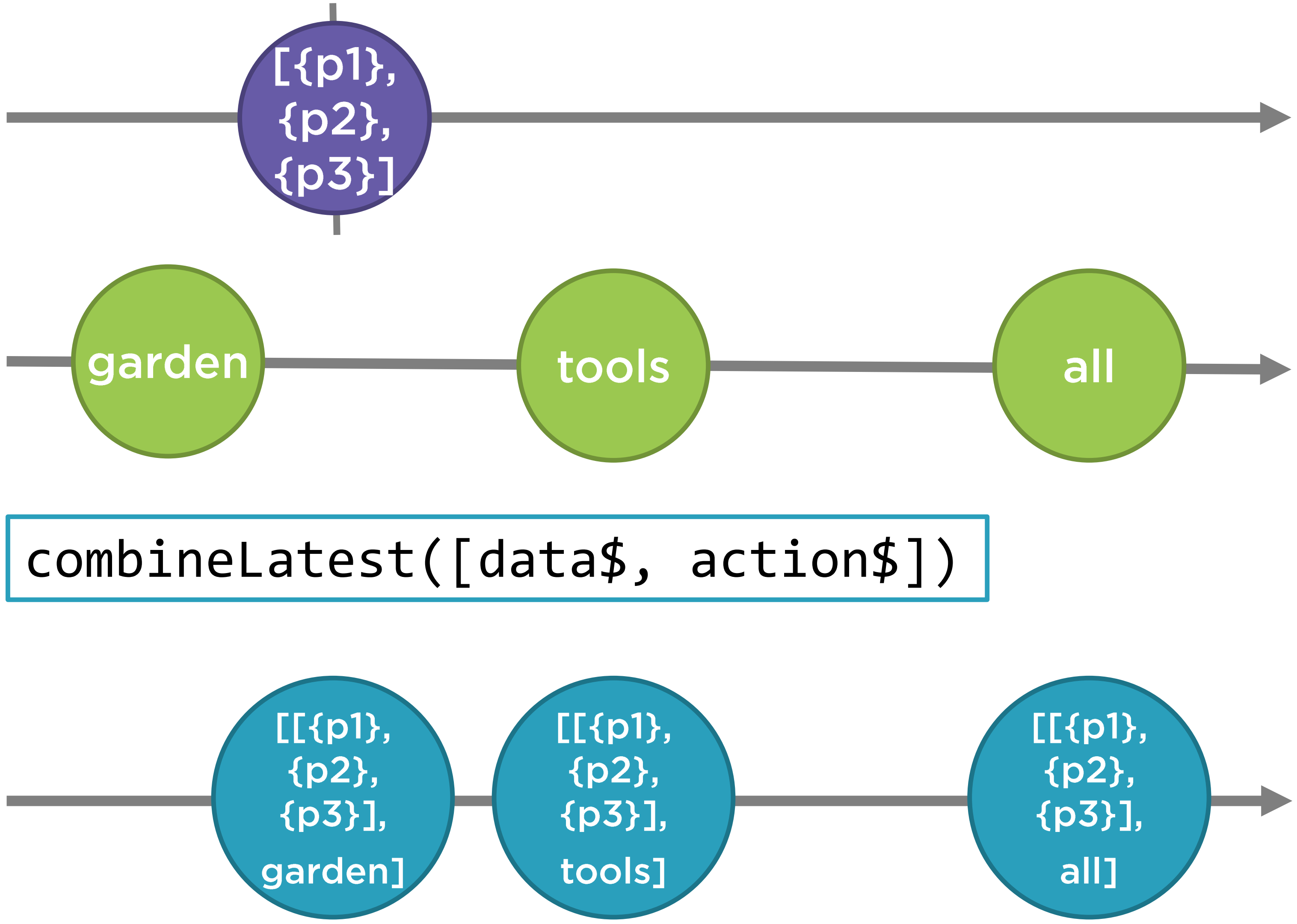
- Hard-coded category

Demo

Filtering a stream: Demo II

- Dropdown list of categories

Data Stream vs. Action Stream



[{p1}, {p2}, {p3}]
garden
tools
all
[[{p1}, {p2}, {p3}], garden]
[[{p1}, {p2}, {p3}], tools]
[[{p1}, {p2}, {p3}], all]

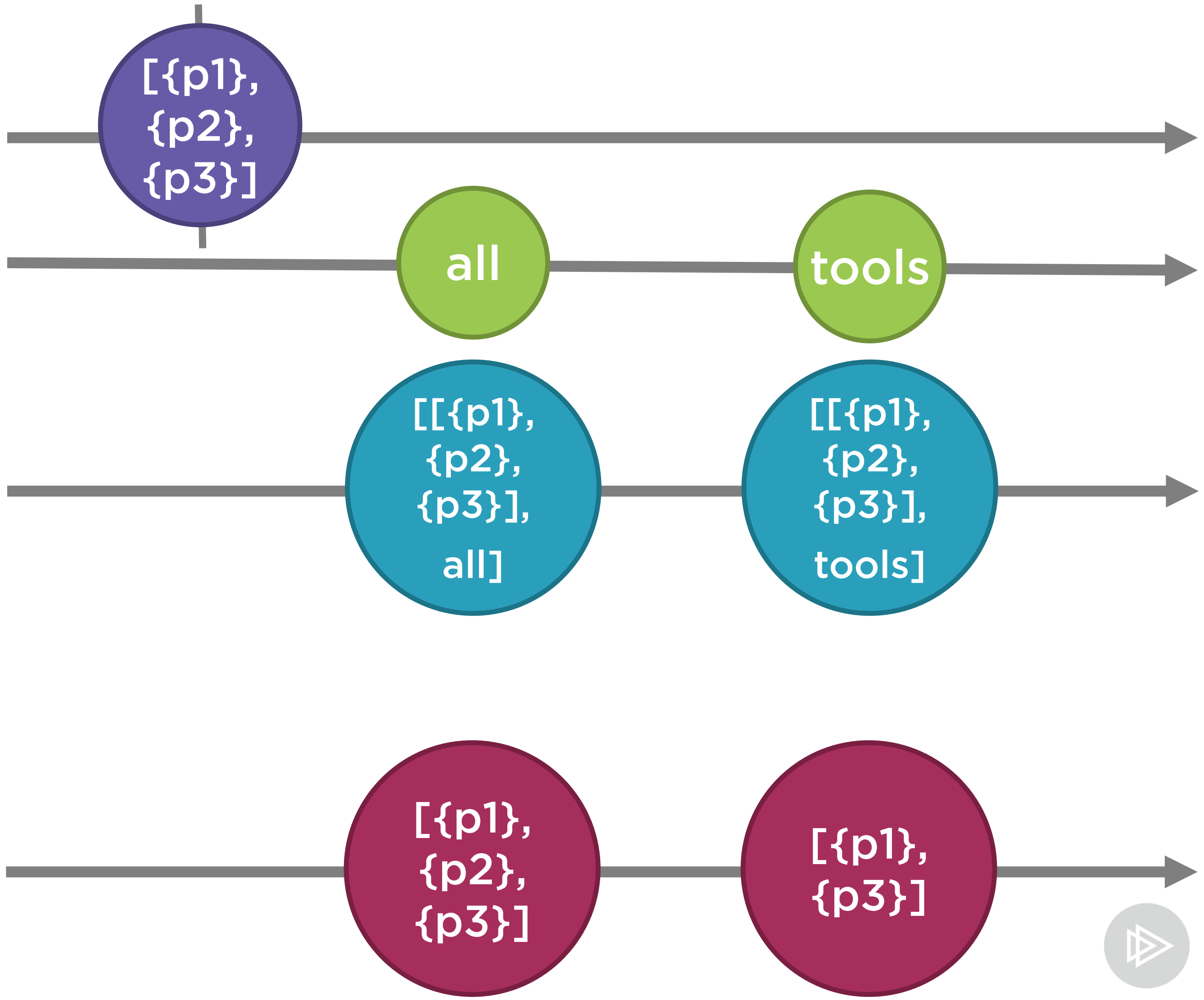
```
combineLatest([data$, action$])
```

Data Stream vs. Action Stream

```
products$ = combineLatest([
  this.productService.products$,
  this.action$



])
.pipe(
  map(([products, category]) =>
   products.filter(product =>
     product.category === category)
  )
);
```



Creating Streams

Data Stream

```
products$ = this.http.get<Product[]>(this.productsUrl)
```

Action Stream

```
action$ = ???
```

- Use a built-in stream
- `fromEvent`
- Subject/BehaviorSubject

Observable and Observer

Observable Stream

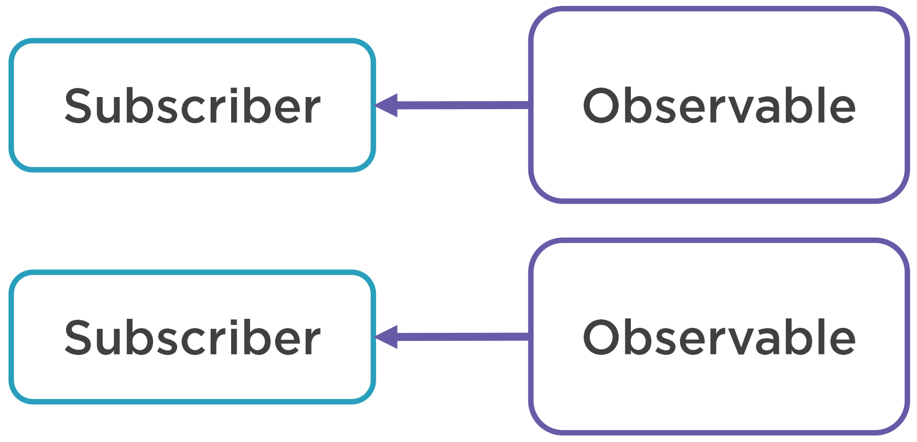
Observer:

next()

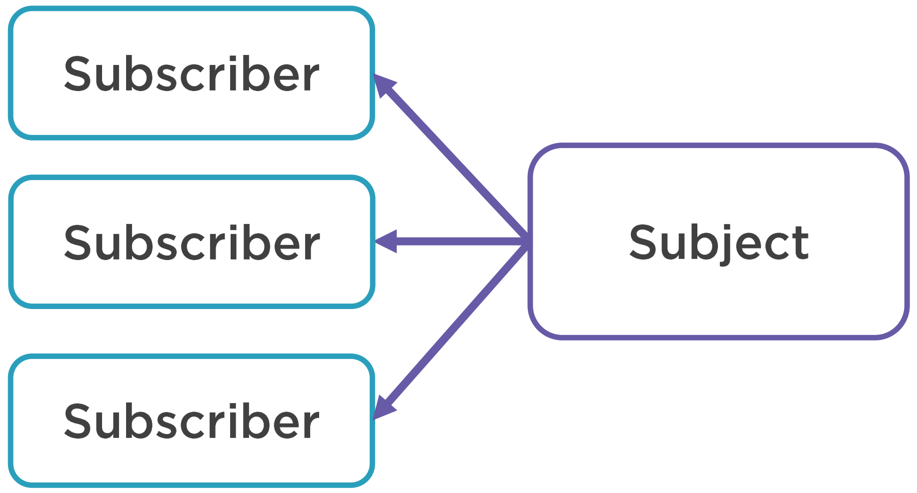
error()

complete()

Unicast vs. Multicast



Observable is unicast



Subject is multicast

Subject

```
private categorySelectedSubject = new Subject<number>();
categorySelectedAction$ = this.categorySelectedSubject.asObservable();
```

```
onSelected(categoryId): void {
  this.categorySelectedSubject.next(+categoryId);
}
```

```
products$ = combineLatest([
  this.productService.products$,
  this.categorySelectedAction$
])
.pipe(
  map(([products, categoryId]) =>
   products.filter(product =>
     categoryId ? product.categoryId === categoryId : true)
  )
);
```

BehaviorSubject

```
private categorySelectedSubject = new BehaviorSubject<number>(0);
categorySelectedAction$ = this.categorySelectedSubject.asObservable();
```

```
onSelected(categoryId): void {
  this.categorySelectedSubject.next(+categoryId);
}
```

```
products$ = combineLatest([
  this.productService.products$,
  this.categorySelectedAction$
])
.pipe(
  map(([products, categoryId]) =>
   products.filter(product =>
     categoryId ? product.categoryId === categoryId : true)
  )
);
```

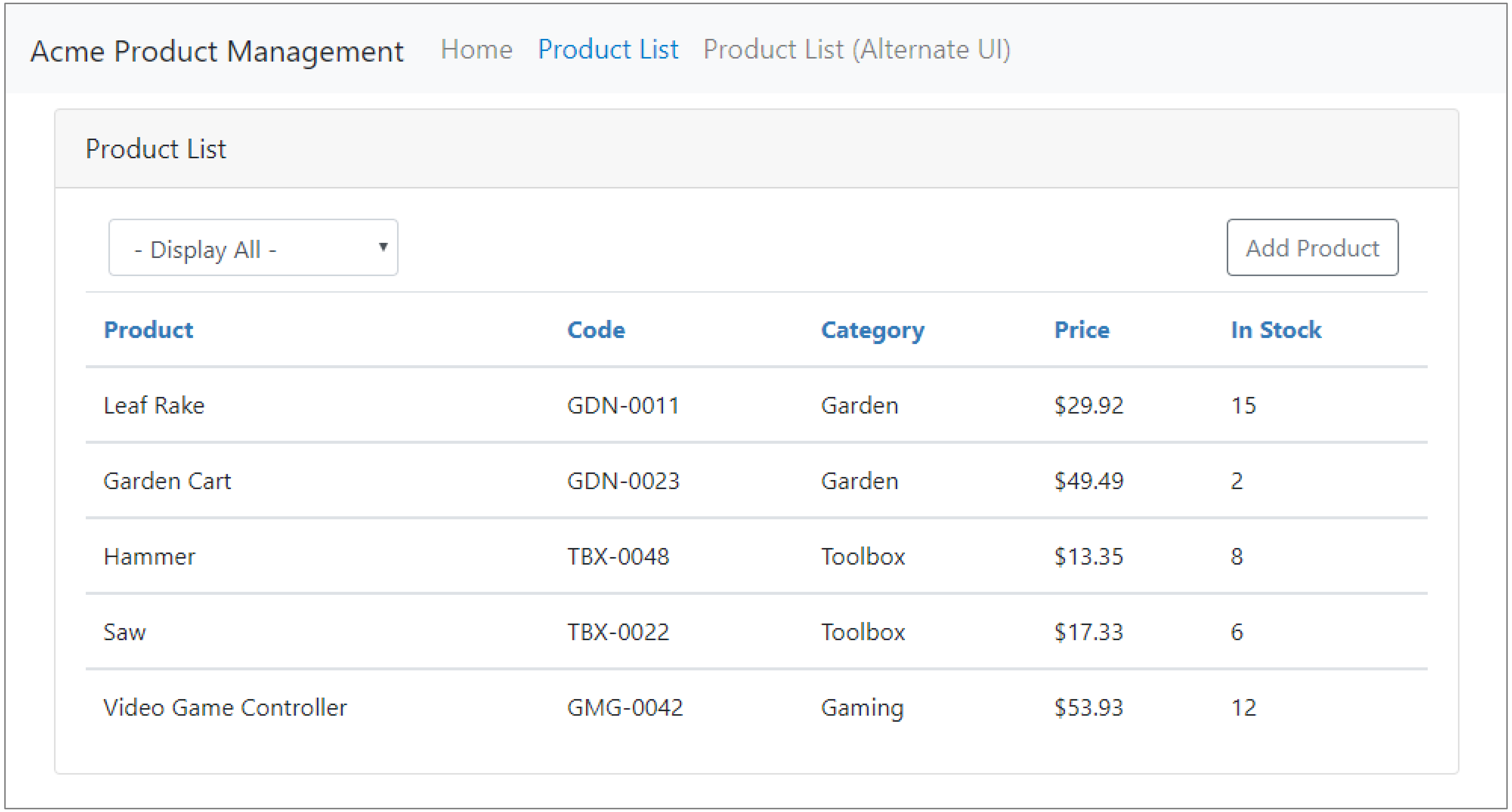
Reacting to Actions

Create an action stream (Subject/BehaviorSubject)

Combine the action stream and data stream

Emit a value to the action stream when an action occurs

Starting with an Initial Value



Acme Product Management Home Product List Product List (Alternate UI)

Product List

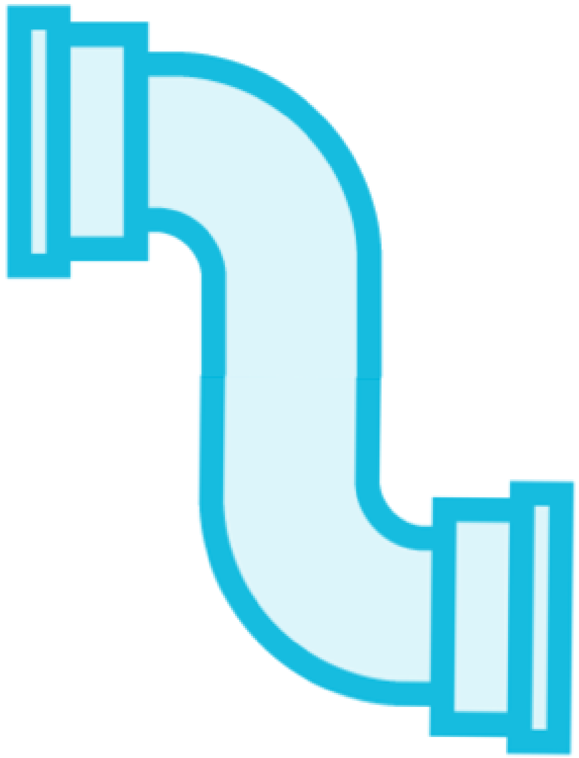
Product	Code	Category	Price	In Stock
Leaf Rake	GDN-0011	Garden	$29.92	15
Garden Cart	GDN-0023	Garden	$49.49	2
Hammer	TBX-0048	Toolbox	$13.35	8
Saw	TBX-0022	Toolbox	$17.33	6
Video Game Controller	GMG-0042	Gaming	$53.93	12

Starting with an Initial Value

```
this.categorySelectedAction$.pipe(startWith(0))
```

```
private categorySelectedSubject = new BehaviorSubject<number>(0);
categorySelectedAction$ = this.categorySelectedSubject.asObservable();
```

RxJS Operator: `startWith`

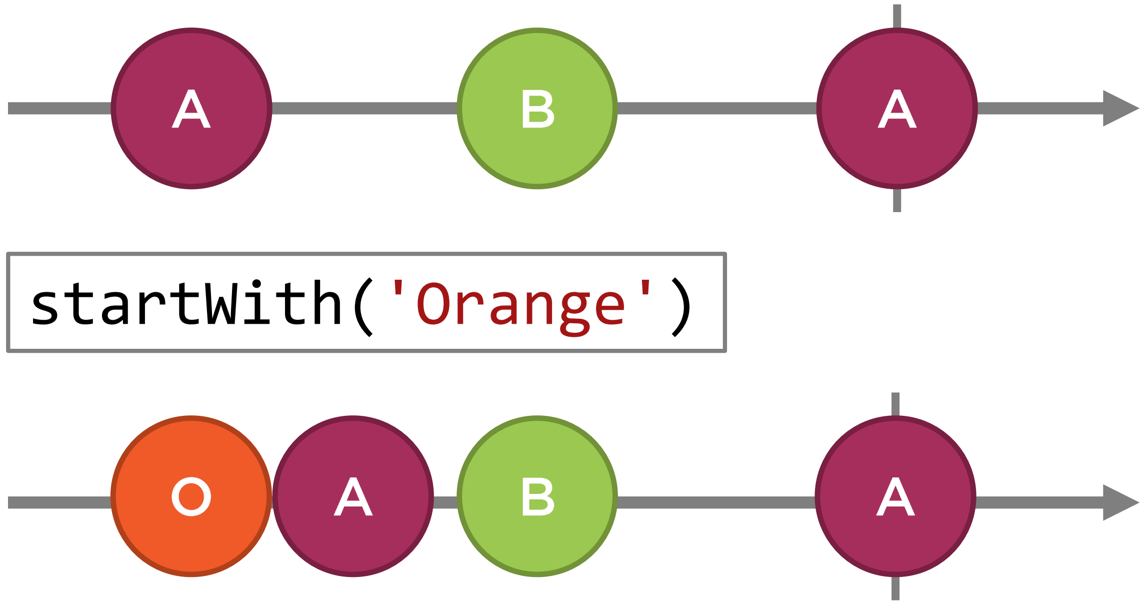


Provides an initial value

```
startWith('Orange')
```

Marble Diagram: `startWith`

```
of('A', 'B', 'A')
  .pipe(
    startWith('O'),
  );
```



RxJS Operator: `startWith`

`startWith` is a combination operator

- Takes in an input stream, subscribes
- Creates an output stream

When a source item is emitted

- If it's the first item, it emits the specified initial value(s), then …
- It emits the item to the output stream

Initial value(s) must be the same type as the input Observable

Reacting to Actions

Create an action stream (Subject/BehaviorSubject)

```
selSubject = new Subject<number>();
selectedAction$ = this.selSubject.asObservable();
```

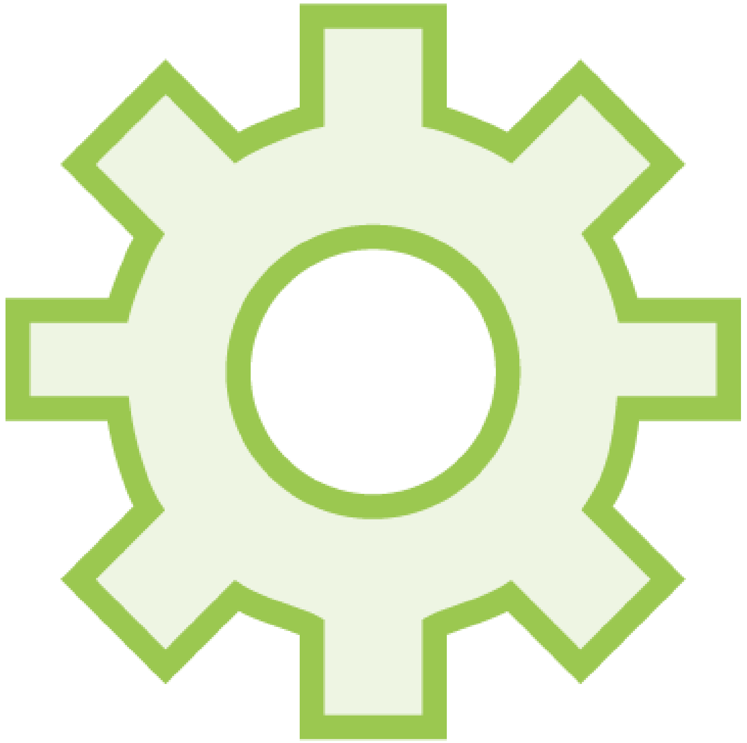
Combine the action and data streams

```
products$ = combineLatest([
  this.productService.products$,
  this.selectedAction$
]).pipe(...);
```

Emit a value to the action stream when an action occurs

```
onSelected(id): void {
  this.selSubject.next(+id);
}
```

RxJS Features



`filter`: **Only emits items that match criteria**

```
filter(item => item === 'Apple')
```

`startWith`: **Defines an initial value emitted before the input stream values**

```
startWith('Orange')
```

Subject/BehaviorSubject

Subject: **Special type of Observable that is both an Observable and an Observer**

```
selectedSubject = new Subject<number>();
```

BehaviorSubject: **Special type of Subject that emits an initial value**

```
selectedSubject = new BehaviorSubject<number>(0);
```