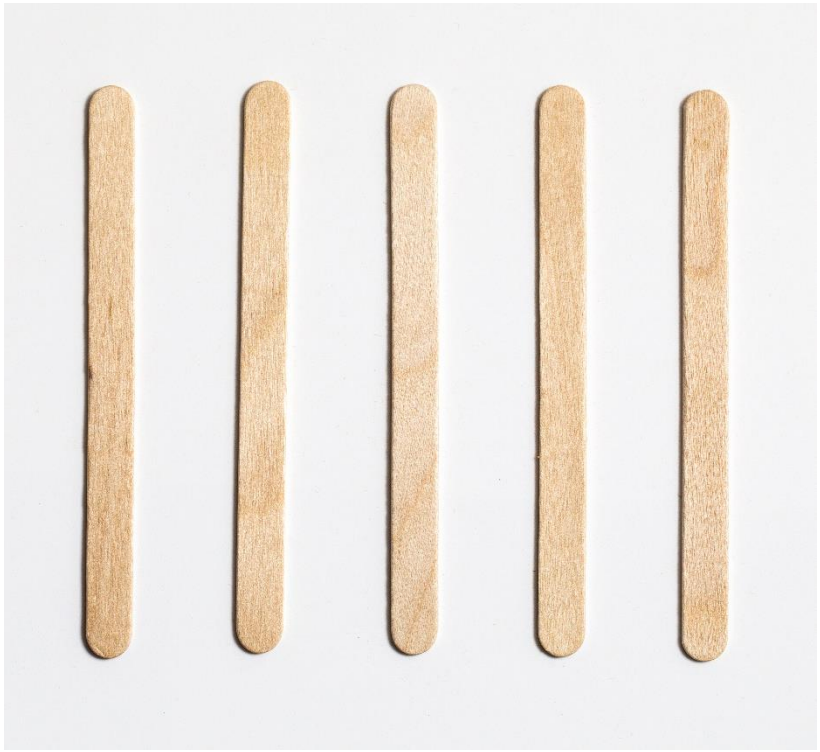
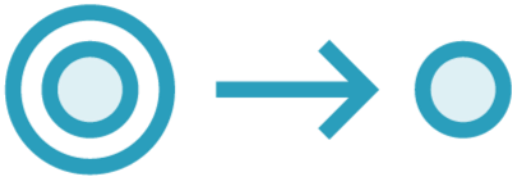
Combining Streams

Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

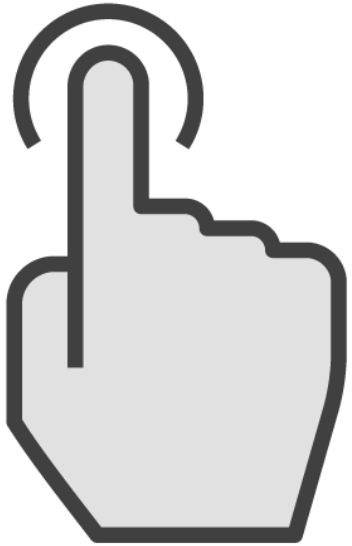


Why Combine Observable Streams?



Map id to a string

Work with multiple data sources



React to actions

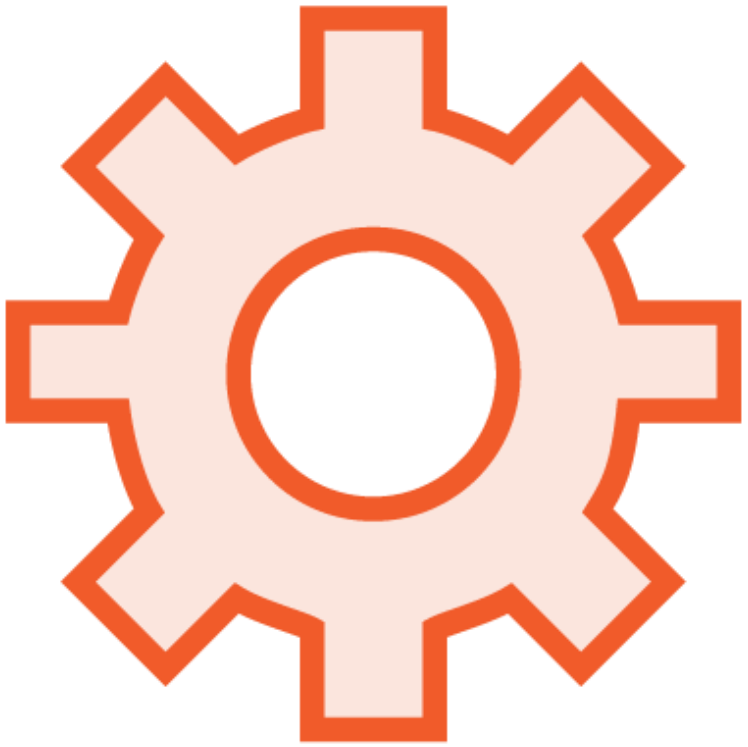
Simplify template code

Module Overview

Combination operators

Combining streams to map an id to a string

RxJS Features

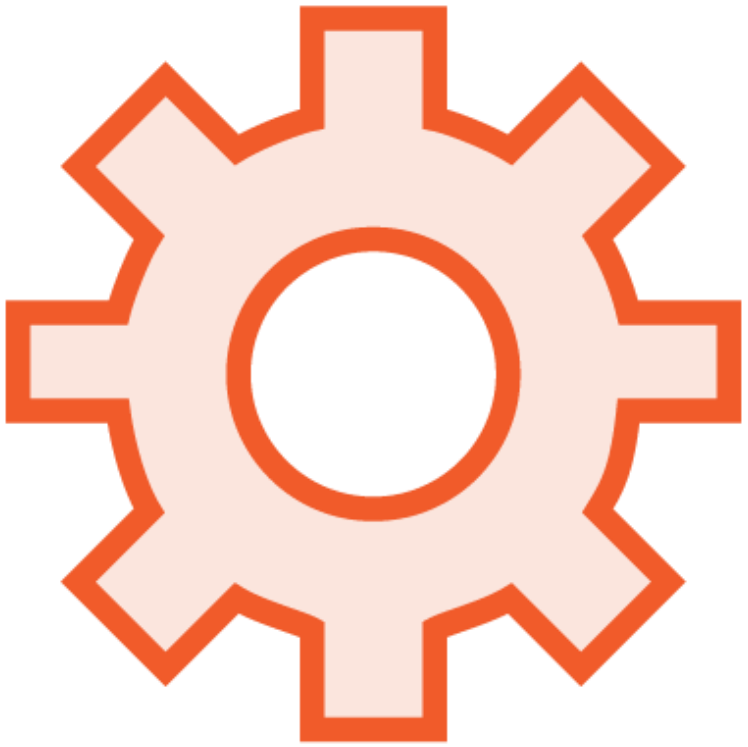


`combineLatest`

`forkJoin`

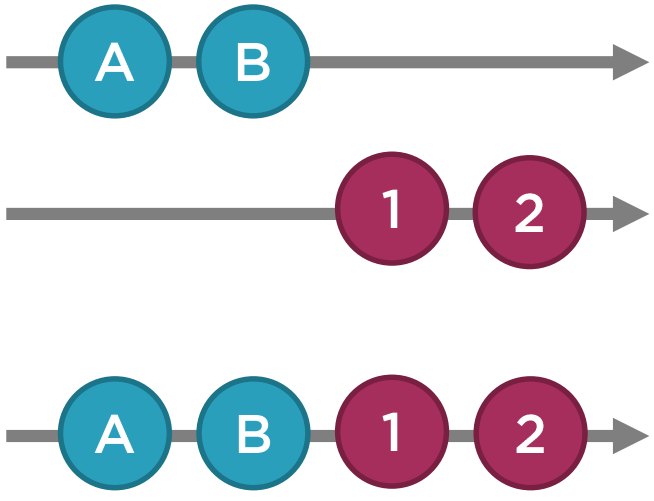
`withLatestFrom`

Combination Operators/Functions

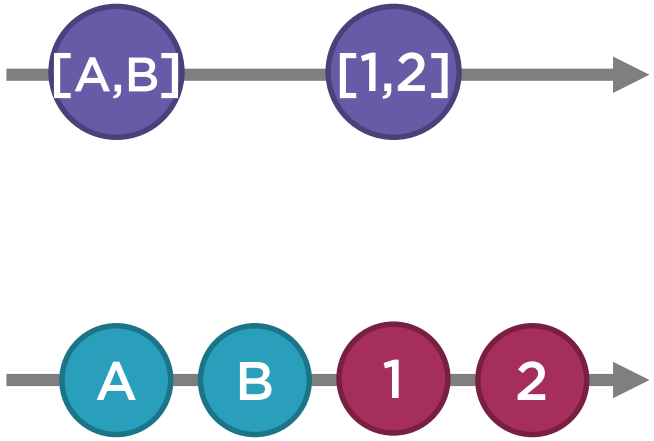


Combine information from multiple Observable streams

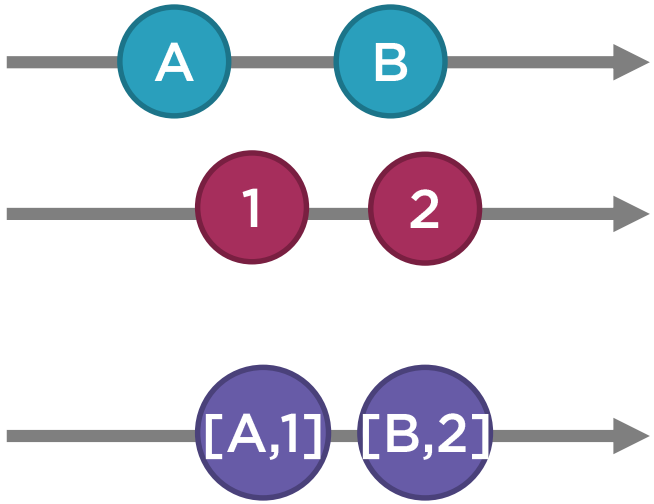
Types of Combination Operators/Functions



Combine to a single stream



Flatten higher-order Observables



Emit a combined value

RxJS Creation Function: `combineLatest`

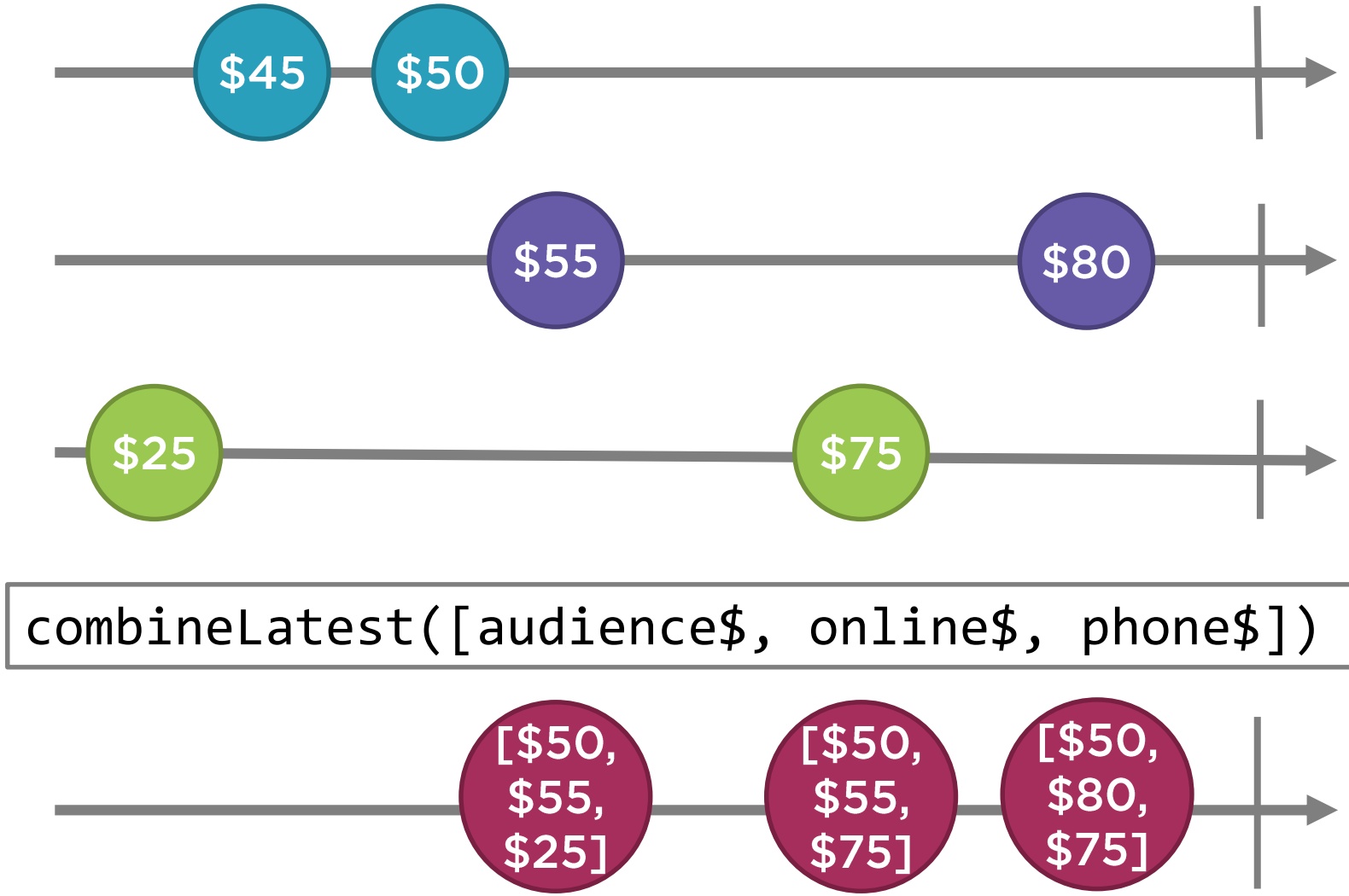
Creates an Observable whose values are defined:

- Using the latest values from each input Observable

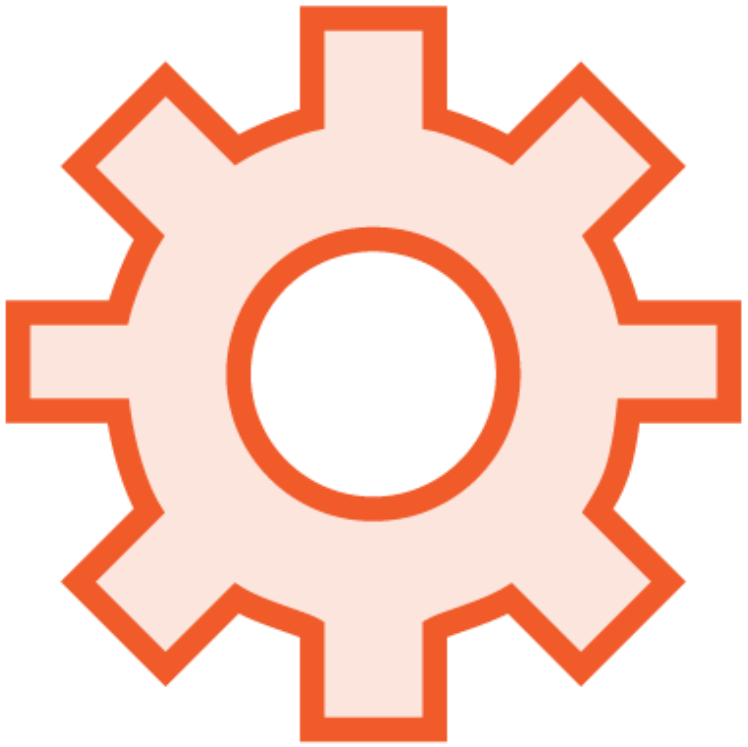
```
combineLatest([a$, b$, c$])
```

Static creation function, not a pipeable operator

Marble Diagram: `combineLatest`



RxJS Creation Function: `combineLatest`



`combineLatest` **is a combination function**

- Takes in a set of streams, subscribes
- Creates an output stream

When an item is emitted from any stream

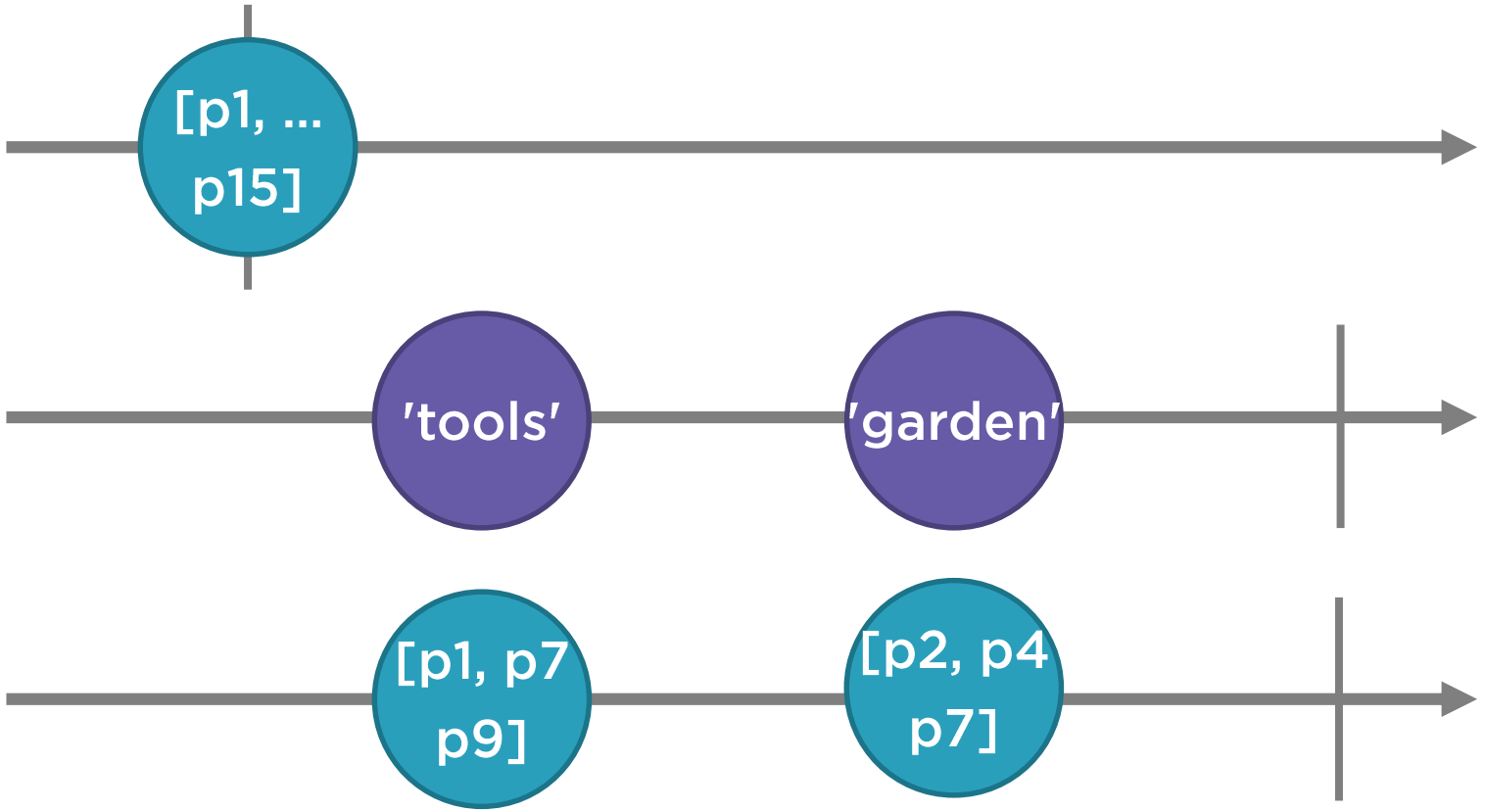
- If all streams have emitted at least once
- Emits a value to the output stream

Completes when all input streams complete

Emitted value combines the latest emitted value from each input stream into an array

Use `combineLatest`

To re-evaluate state when an action occurs



RxJS Creation Function: `forkJoin`

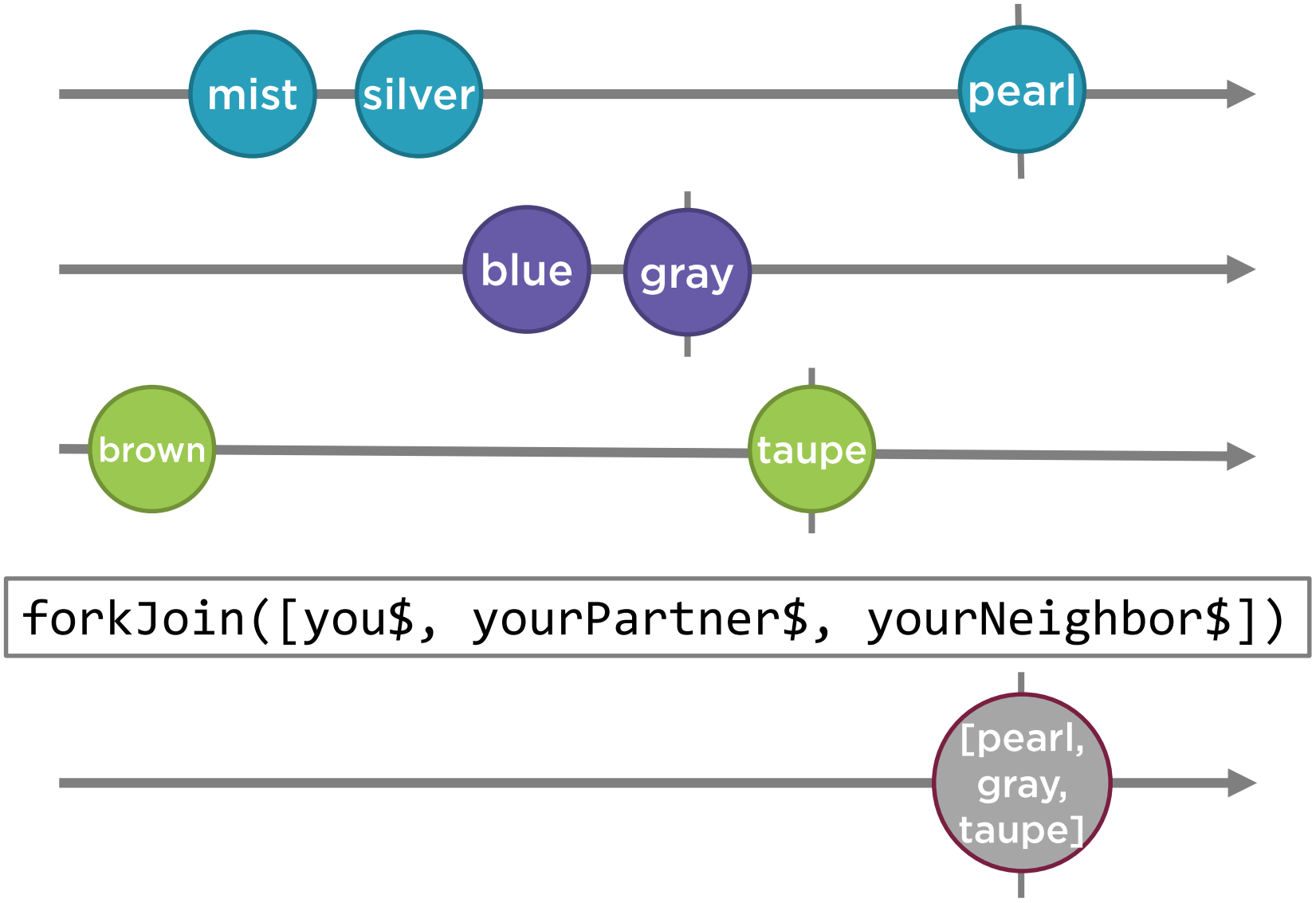
Creates an Observable whose value is defined

- Using the last value from each input Observable

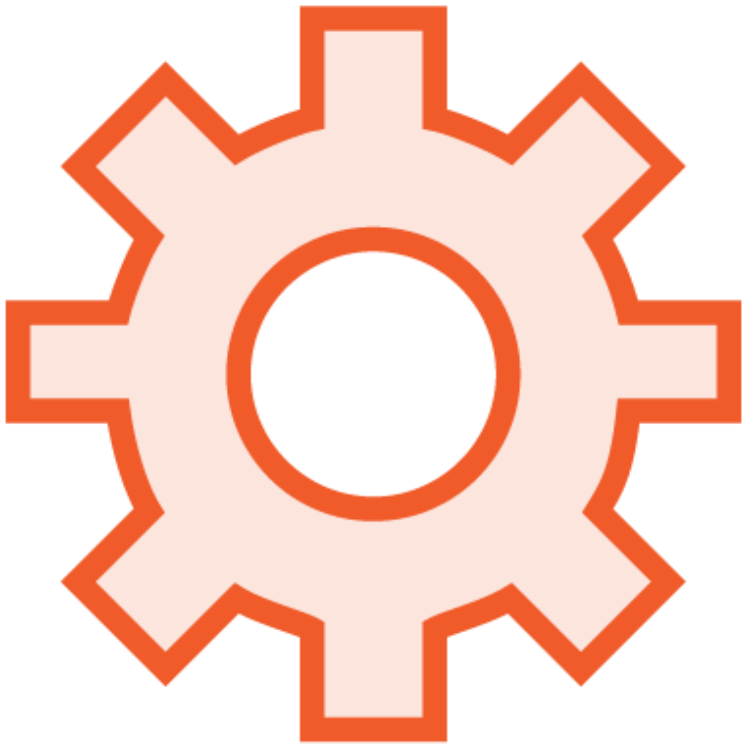
```
forkJoin([a$, b$, c$])
```

Static creation function, not a pipeable operator

Marble Diagram: `forkJoin`



RxJS Function: `forkJoin`



`forkJoin` **is a combination function**

- Takes in a set of streams, subscribes
- Creates an output stream

When all input streams complete

- Emits a value to the output stream
- And completes

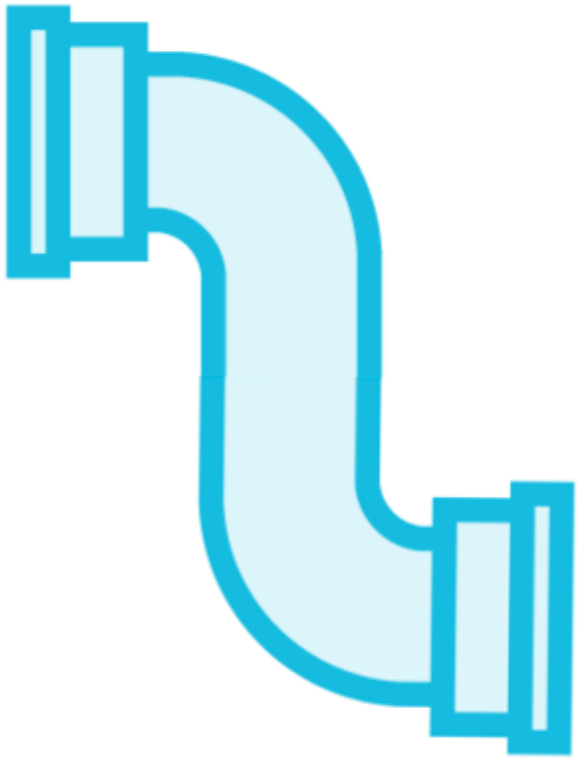
Emitted value combines the last emitted value from each input stream into an array

Use `forkJoin`

To wait to process any results until all streams are complete

Don't use when working with streams that don't complete

RxJS Operator: `withLatestFrom`



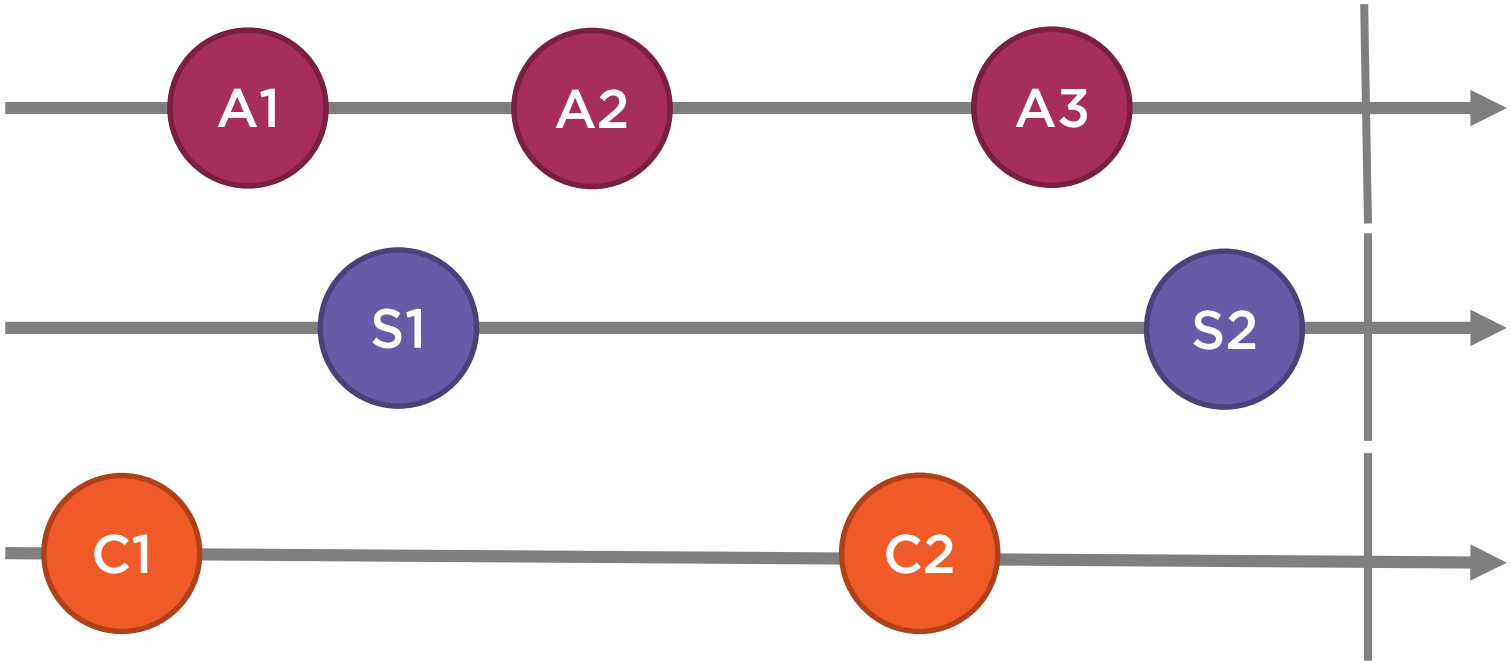
Creates an Observable whose values are defined

- Using the latest values from each input Observable
- But only when the source stream emits

```
a$.pipe(withLatestFrom(b$, c$))
```

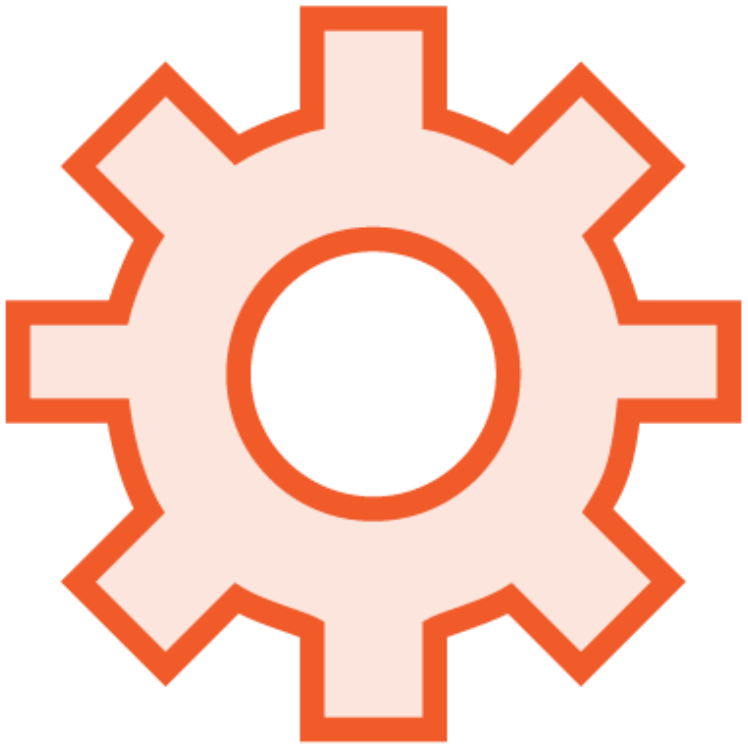
Pipeable operator

Marble Diagram: `withLatestFrom`




```
apples$.pipe(withLatestFrom(sticks$, caramel$))
```

RxJS Operator: `withLatestFrom`



`withLatestFrom` **is a combination operator**

- Takes in a set of streams, subscribes
- Creates an output stream

When an item is emitted from source stream

- If all streams have emitted at least once
- Emits a value to the output stream

Completes when the source stream completes

Emitted value combines the latest emitted value from each input stream into an array

Use `withLatestFrom`

To react to changes in only one stream

To regulate the output of the other streams

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

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	$17.33	6
Video Game Controller	GMG-0042	5	$53.93	12

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

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

Mapping an Id to a String

```
export interface Product {
 id: number;
 productName: string;
 productCode?: string;
 description?: string;
 price?: number;
 categoryId?: number;
}
```

```
export interface Product {
 id: number;
 productName: string;
 productCode?: string;
 description?: string;
 price?: number;
 categoryId?: number;
 category?: string;
}
```

Mapping an Id to a String

```
products$=this.http.get<Product[]>(this.url)
 .pipe(
   map(products =>
    products.map(product => ({
      ...product,
      price: product.price * 1.5,
      category: ???
   }) as Product)
 ));
```

```
category: this.getCategory(product.categoryId)
```

```
productCategories$ = this.http.get<ProductCategory[]>(this.productCategoriesUrl);
```

Combining the Streams

```
productsWithCategory$ = combineLatest([
 this.products$,
 this.productCategories$
]);
```

[product[], category[]]

```
productsWithCategory$ = forkJoin([
 this.products$,
 this.productCategories$
]);
```

[product[], category[]]

```
productsWithCategory$ = this.products$
 .pipe(
   withLatestFrom(this.productCategories$)
 );
```

[product[], category[]]

Mapping an Id to a String

```
product$ = this.http.get<Product[]>(this.url);
```

```
productCategories$ = this.http.get<ProductCategory[]>(this.productCategoriesUrl);
```

```
productsWithCategory$ = combineLatest([
 this.products$,
 this.productCategories$
])
 .pipe(
   map(([products, categories]) =>
    products.map(product => ({
      ...product,
      price: product.price * 1.5,
      category: categories.find(
       c => product.categoryId === c.id
      ).name
   }) as Product))
 );
```

Mapping an Id to a String

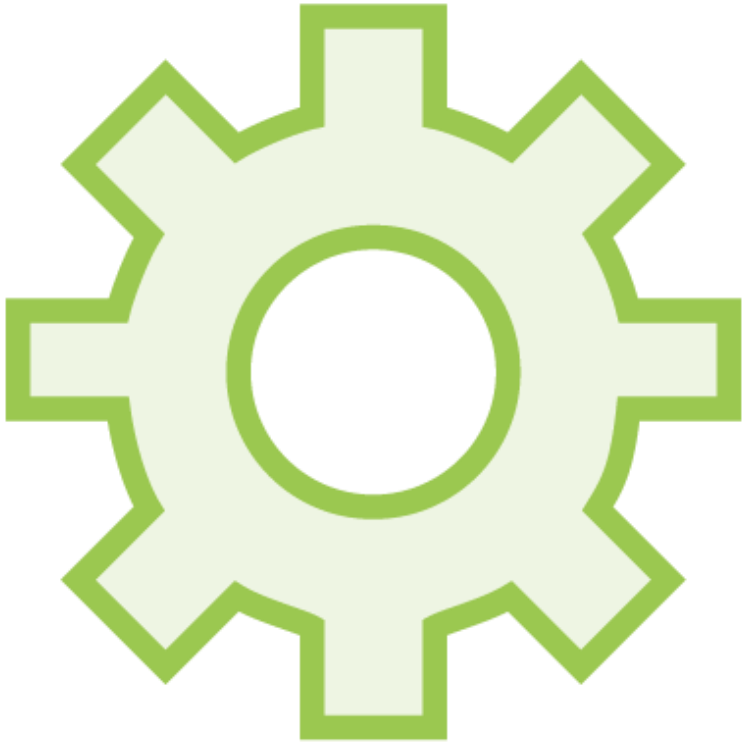
```
productsWithCategory$ = combineLatest([
 this.products$,
 this.productCategories$
])
 .pipe(
   map(([products, categories]) =>
    products.map(product => ({
     ...product,
     price: product.price * 1.5,
     category: categories.find(
      c => product.categoryId === c.id
     ).name
   }) as Product))
 );
```

[product[], category[]]

Demo

Combining streams
to map an id to a string

Combining Observable Streams



combineLatest: Emits any time a new value is emitted from **any** of them

```
combineLatest([a$, b$, c$])
```

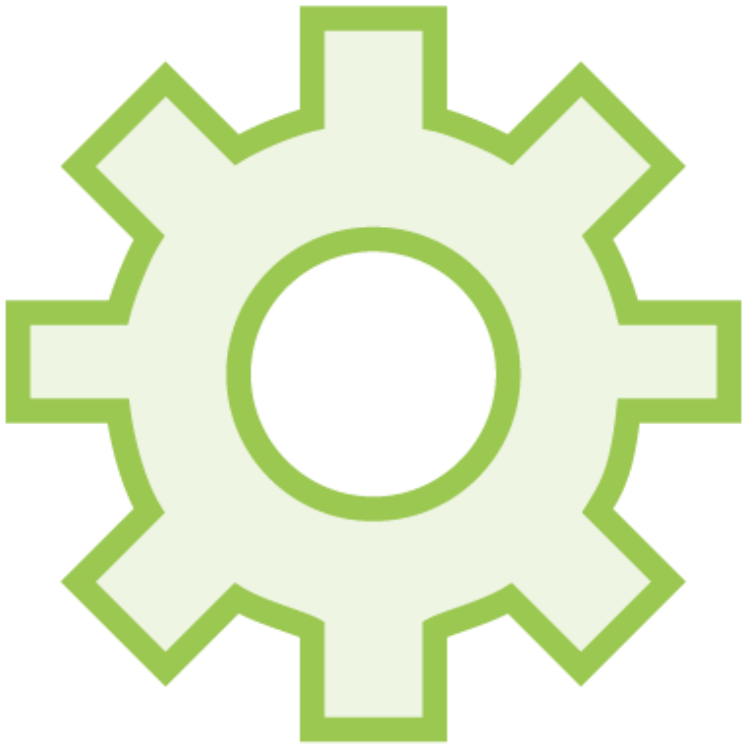
forkJoin: Emits only the **last** emitted values

```
forkJoin([a, b$, c$])
```

withLatestFrom: Emits any time a new value is emitted from the **source** stream

```
a$.pipe(withLatestFrom(b$, c$))
```

Emitted Item



```
combineLatest([a$, b$, c$])

forkJoin([a$, b$, c$])

a$.pipe(withLatestFrom(b$, c$))
```

```
[a1, b1, c1]
```

Mapping an Id to a String

Combine the streams

Map the items

```
productsWithCategory$ = combineLatest(
 this.products$,
 this.productCategories$
)
 .pipe(
   map([products, categories] =>
    products.map(product => ({
     ...product,
     category: categories.find(
      c => product.categoryId === c.id
     ).name
   }) as Product))
 );
```