Final Words

Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

Goals for This Course

- Add clarity
- Evaluate current patterns
- Improve state management
- Merge RxJS streams
- Minimize subscriptions
- Improve UI performance

Key points, tips and common issues

RxJS Operators

Import from 'rxjs/operators'

```
import { map, catchError } from 'rxjs/operators';
```

Specify operators within the pipe method

```
of(2, 4, 6)
  .pipe(
     map(item => item * 2),
     tap(item => console.log(item))
  );
```

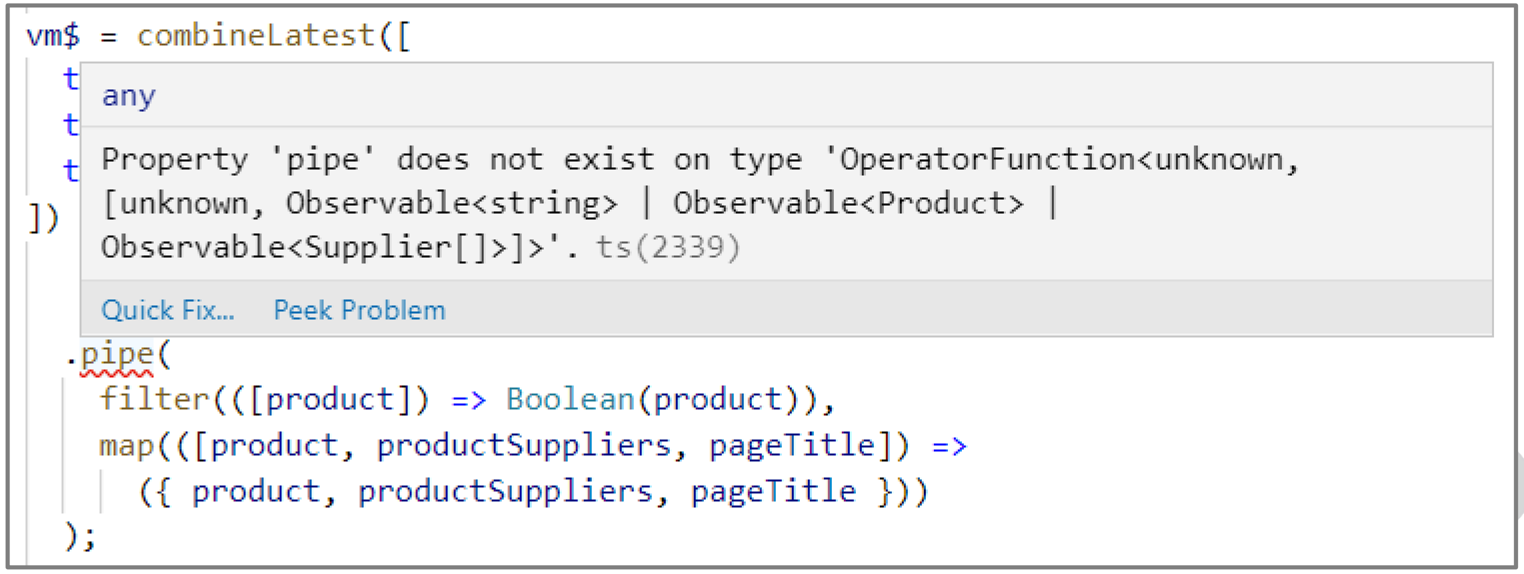
RxJS Creation Functions

Import from 'rxjs'

```
import { combineLatest, of } from 'rxjs';
```

Be careful not to import the operator with the same name

```
import { combineLatest } from 'rxjs/operators';
```



```
vm$ = combineLatest([
  t
  t
  t
])
```

```
any
Property 'pipe' does not exist on type 'OperatorFunction<unknown,
[unknown, Observable<string> | Observable<Product> |
Observable<Supplier[]>]>'. ts(2339)
```

Quick Fix...   Peek Problem

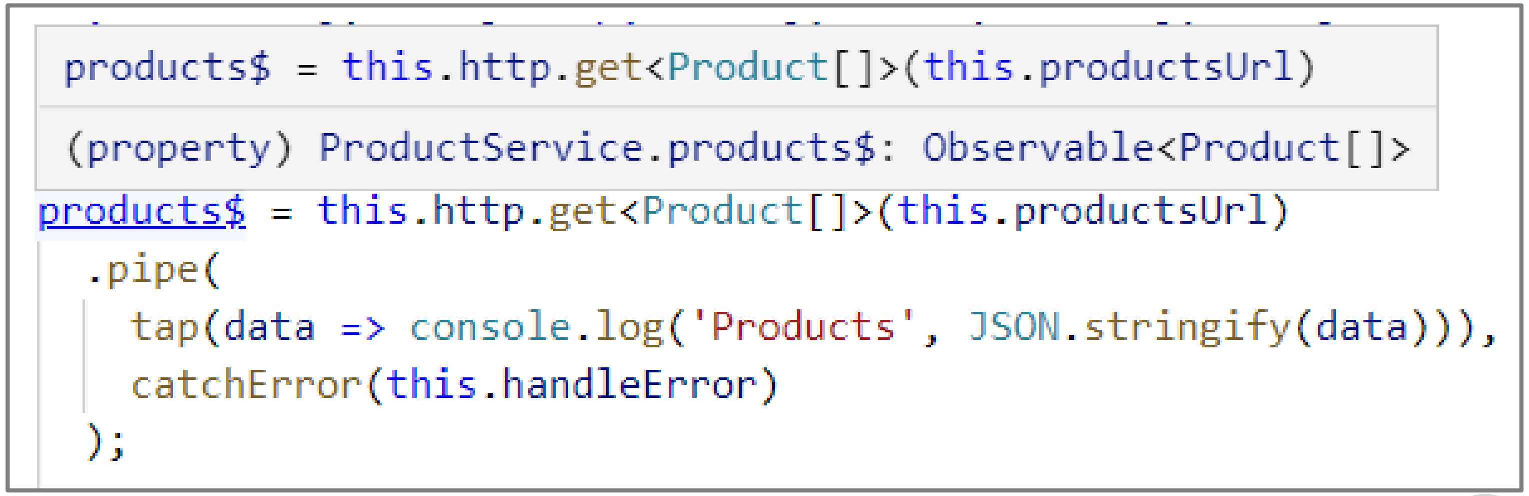
```
  .pipe(
    filter(([product]) => Boolean(product)),
    map(([product, productSuppliers, pageTitle]) =>
      ({ product, productSuppliers, pageTitle }))
  );
```

Debugging Observables

Use the `tap` operator

```
tap(data => console.log(JSON.stringify(data)))
```

Hover over the Observable to view the type



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

(property) ProductService.products$: Observable<Product[]>
products$ = this.http.get<Product[]>(this.productsUrl)
  .pipe(
    tap(data => console.log('Products', JSON.stringify(data))),
    catchError(this.handleError)
  );
```

Data Streams

Emits one item, the response

After emitting the response, the stream completes

The response is often an array

To transform the array elements:
- Map the emitted array
- Map each array element
- Transform each array element

Action Streams

Only emits if it is active

If the stream is stopped, it won't emit

An unhandled error causes the stream to stop

Catch the error and replace the errored Observable

- Don't replace an errored action Observable with EMPTY
- Replace with a default or empty value

Combination Operators (Array)

Don't emit until each source stream emits at least once

- `combineLatest`
- `forkJoin`
- `withLatestFrom`

Action stream created with a `Subject` does not immediately emit

When combining with an action stream, consider using a `BehaviorSubject` since it emits a default value

Complete Notifications

Some functions/operators require the input Observable(s) complete before they emit

- `forkJoin`
- `toArray`

Take care when using these functions/operators with action streams, which often don't complete

```
this.selectedProduct$
 .pipe(
  switchMap(selectedProduct =>
   from(selectedProduct.supplierIds)
    .pipe(
     mergeMap(id => this.http.get<Supplier>(`${this.url}/${id}`)),
     toArray()
 )));
```

A Few More Terms

Cold Observable

- Doesn't emit until subscribed to
- Unicast

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

Hot Observable

- Emits without subscribers
- Multicast

```
productSubject = new Subject<number>();
this.productSubject.next(12)
```

Learning More

Pluralsight courses

- Angular Component Communication
- Angular NgRx: Getting Started
- RxJS: Getting Started
- Learning RxJS Operators by Example Playbook

RxJS documentation

- rxjs.dev

@deborahkurata