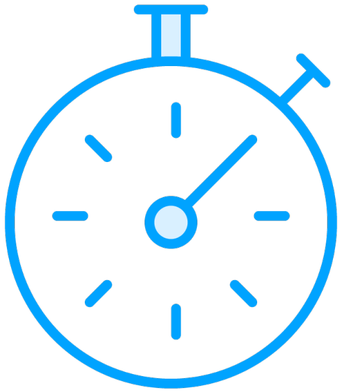
Debouncing and Transforming User Input

Deborah Kurata

Developer | Content Creator | MVP | GDE

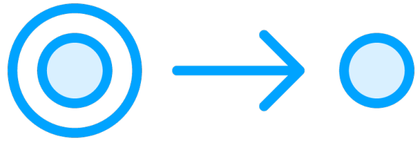
@deborahkurata | https://www.youtube.com/@deborah_kurata

Handling User Input with RxJS



Debounce keystrokes

Filter noisy input



Transform values

Search Feature

Issuing an HTTP request every time a user presses a key is inefficient

Often want to transform what's typed before passing it as a query parameter

Debouncing

Consolidating events
that happen within
a specified short amount of time

Debouncing

Typing: T

Emission:

Debouncing

Typing: T
H

Emission:

Debouncing

Typing: T H E

Emission: THE

Debouncing

Debouncing is time-based
Requires asynchronous processing

Signals are synchronous
They don't understand the concept of time

Observables understand time

GitHub

https://github.com/DeborahK/angular-rxjs-ps-course

Beginning sample application files:

apm-begin

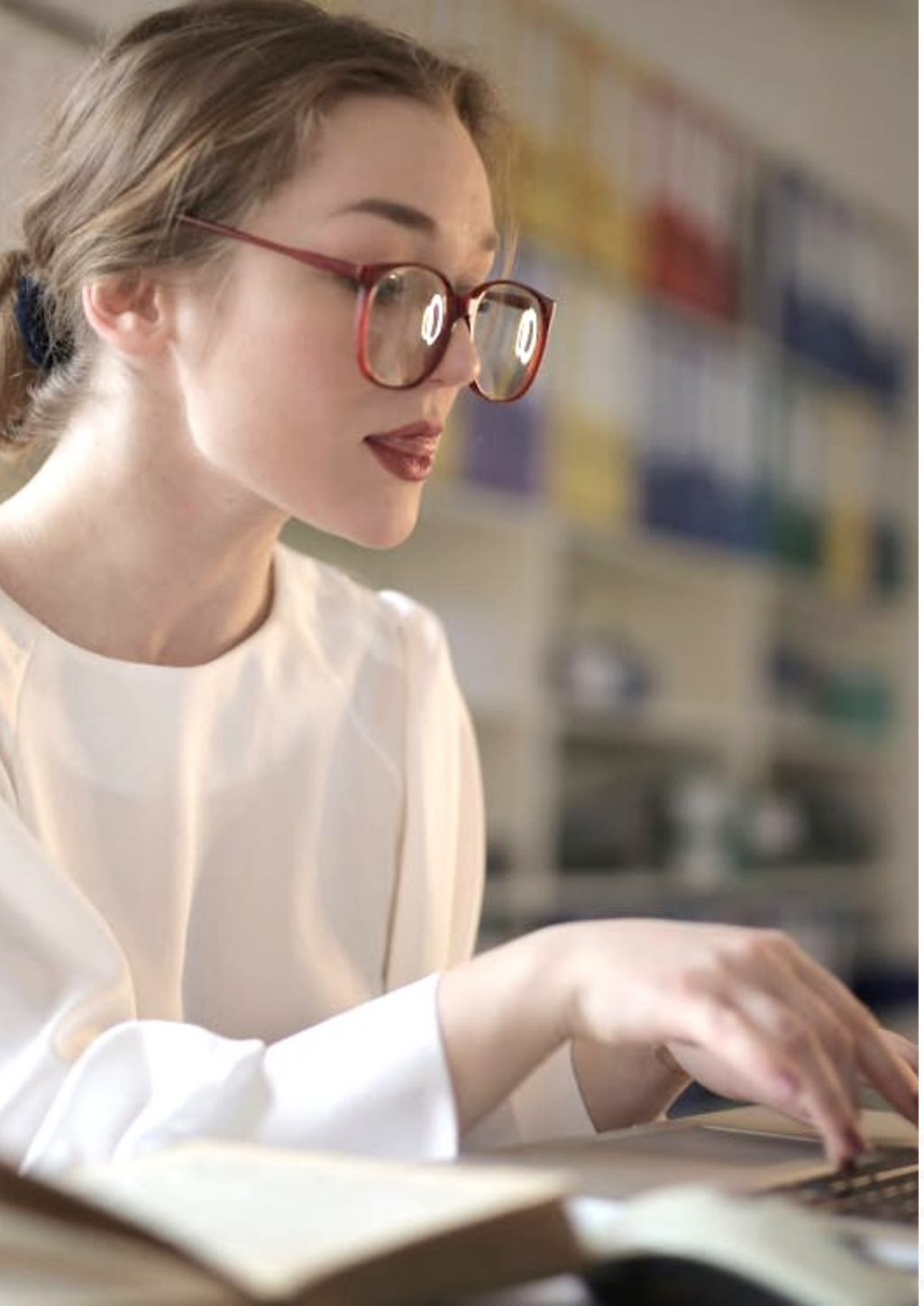
Final (completed) sample application files:

apm-end

File with clickable links to additional information:

MOREINFO.md

RxJS continues to play an important role in today's Angular



Use RxJS to React to DOM Events

React to keyboard, mouse, network or media events

Define hotkeys for power users

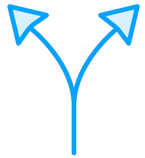
Move elements in a game

Control audio/video playback

Detect loss of network and reduce data flow

...

Use RxJS with Angular's Observables



Watch routing parameters, data, or events

React to value and status changes in reactive forms

Issue HTTP requests with HTTPClient

- Especially when you can't yet use the experimental Resource API
- To send POST, PUT or other mutative HTTP requests

Use RxJS Operators to Compose and Transform Complex Data

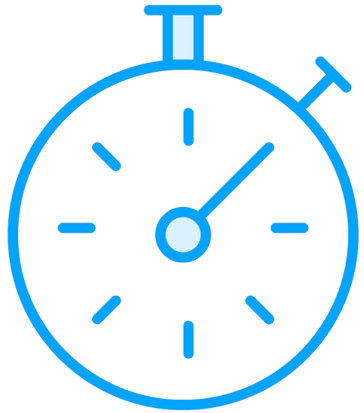
Merge multiple datasets

Replace foreign keys with names from another data source

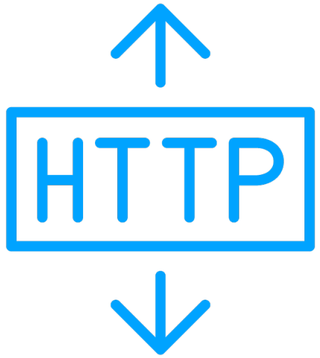
Flatten nested structures

Reshape backend responses into a more useful format

Use RxJS to Process User Input

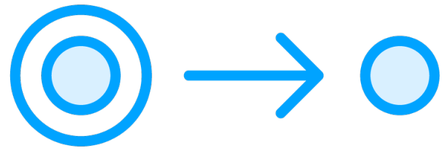


Debounce data entry



Prevent duplicate HTTP calls

Filter invalid or unnecessary values



Convert input: uppercase or trim

Don't Need Observables to React to State Changes

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

selectedProduct$ = this.productSelectedAction$.pipe(
  switchMap(id => this.http.get<Product>(`${this.url}/${id}`)),
  tap(product => console.log('selectedProduct', product)),
);

selectedProductChanged(selectedProductId: number): void {
  this.productSelectedSubject.next(selectedProductId);
}
```

```
selectedProduct = signal<Product | undefined>(undefined);

productResource = httpResource<Product | undefined>(() =>
     `${this.url}/${this.selectedProduct()?.id}`);
```

When to Use a Signal vs. an Observable?

> When working with state, a signal is great!

> For an event-driven stream, an observable is keen!

Thank you!

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