Reacting to Changes

Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

Module Overview

Watching

Reacting

Reactive Transformations

Watching

```
(property) AbstractControl.valueChanges: Observable<any>
```

A multicasting observable that emits an event every time the value of the control changes, in the UI or programmatically.

```
const phoneC

phoneControl.valueChanges.subscribe();
```

```
const phoneControl =
this.customerForm.get('phone');

phoneControl.statusChanges.subscribe();
```

Watching

```
(property) AbstractControl.valueChanges: Observable<any>
```

A multicasting observable that emits an event every time the value of the control changes, in the UI or programmatically.

```
const phoneC

phoneControl.valueChanges.subscribe();
```

```
const phoneControl = this.customerForm.get('phone');

phoneControl.statusChanges.subscribe();
```

Watching

```
this.myFormControl.valueChanges.subscribe(value =>
   console.log(value));
```

```
this.myFormGroup.valueChanges.subscribe(value =>
   console.log(JSON.stringify(value)));
```

```
this.customerForm.valueChanges.subscribe(value =>
   console.log(JSON.stringify(value)));
```

Watching

Sign Up!

First Name — First Name (required)

Last Name — Last Name (required)

Email — Email (required)

Confirm Email — Confirm Email (required)

Phone — Phone

Send Notifications — Email / Text

Rating

Send me your catalog

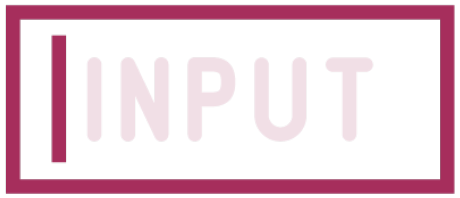
Save

Reacting

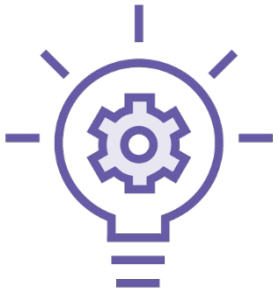
Adjust validation rules

Handle validation messages

Modify user interface elements

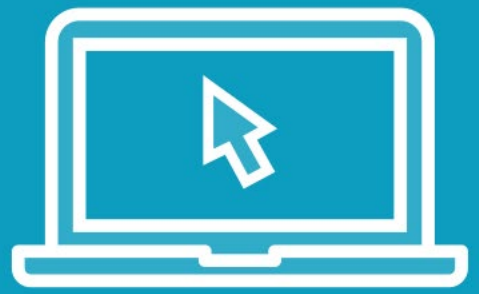


Provide automatic suggestions



...

Demo

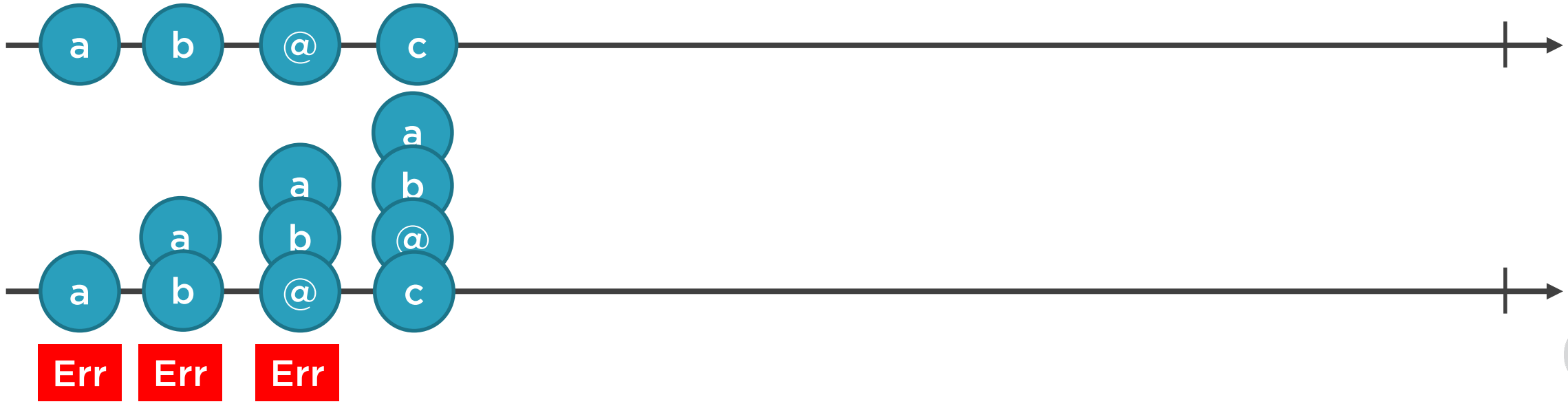


Displaying Validation Messages

Reactive Transformations

debounceTime

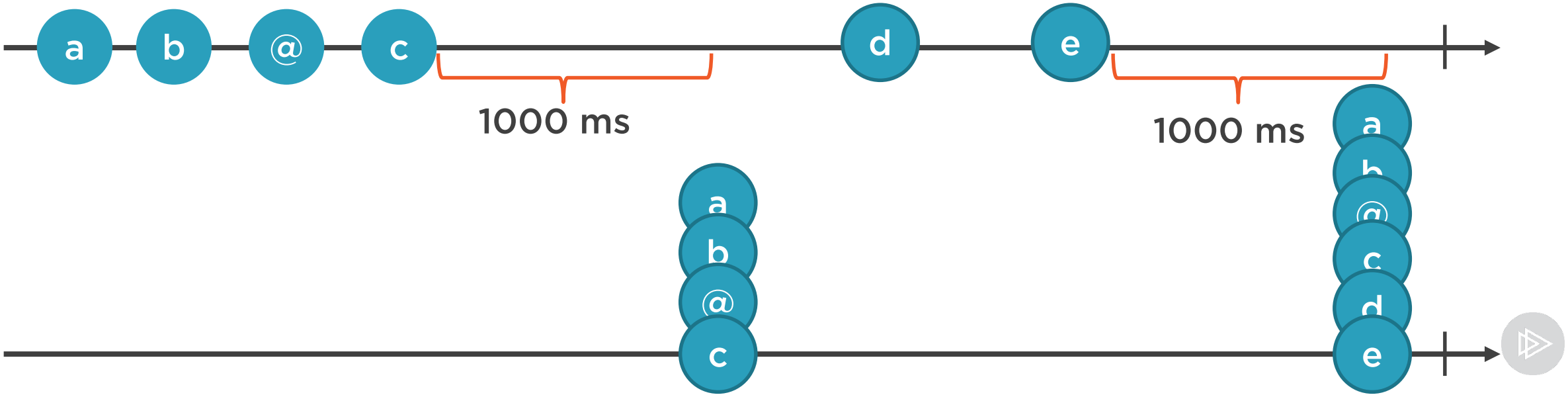
- Ignores events until a specific time has passed without another event
- `debounceTime(1000)` waits for 1000 milliseconds (1 sec) of no events before emitting another event



Reactive Transformations

debounceTime

- Ignores events until a specific time has passed without another event
- `debounceTime(1000)` waits for 1000 milliseconds (1 sec) of no events before emitting another event



Reactive Transformations

throttleTime

- Emits a value, then ignores subsequent values for a specific amount of time

distinctUntilChanged

- Suppresses duplicate consecutive items

Checklist: Watching

Use the valueChanges Observable property

Subscribe to the Observable

```
this.myFormControl.valueChanges.subscribe(
    value => console.log(value)
);
```

Checklist: Reacting

```
this.myFormControl.valueChanges.subscribe(
  value => this.setNotification(value)
);
```

- Change validation rules
- Handle validation messages
- Adjust user-interface elements
- Provide automatic suggestions
- And more

Checklist: Reactive Transformations

import the operator

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

Use the operator

```
this.myFormControl.valueChanges.pipe(
  debounceTime(1000)
).subscribe(
  value => console.log(value)
);
```

Summary

Watching

Reacting

Reactive Transformations

```
<div class="form-group">
  <label class="col-md-2 col-form-label"
         for="emailId">Email</label>
  <div class="col-md-8">
    <input class="form-control"
         id="emailId" type="email"
         placeholder="Email (required)"
         required
         email
         [(ngModel)]="customer.email"
         name="email"
         #emailVar="ngModel"
        [ngClass]="{'is-invalid':
            (emailVar.touched ||
             emailVar.dirty) &&
             !emailVar.valid }" />
    <span class="invalid-feedback">
      <span *ngIf="emailVar.errors?.required">
         Please enter your email address.
      </span>
      <span *ngIf="emailVar.errors?.email">
         Please enter a valid email address.
      </span>
    </span>
  </div>
</div>
```

Template-driven

```
<div class="form-group">
  <label class="col-md-2 col-form-label"
         for="emailId">Email</label>
  <div class="col-md-8">
    <input class="form-control"
           id="emailId" type="email"
           placeholder="Email (required)"
           formControlName = "email"
          [ngClass]="{'is-invalid':
                              emailMessage}" />
    <span class="invalid-feedback">
       {{ emailMessage }}
    </span>
   </div>
</div>
```

Reactive