Reacting to Changes

Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

Module Overview

Watching

Reacting

Reactive Transformations

Watching

- `valueChanges` property emits events on value changes
- `valueChanges` is an `Observable<any>`
- Observable is a collection of events that arrive asynchronously over time
- Subscribe to the observable to watch the events
- `statusChanges` property emits events on validation changes

Watching

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

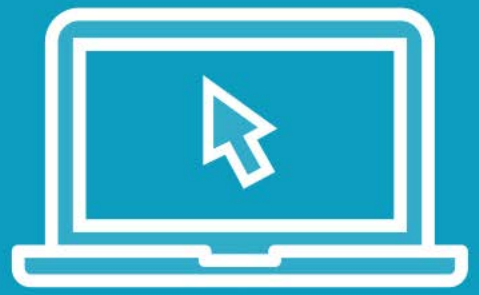
```
this.myFormGroup.valueChanges.subscribe(value =>
   console.log(JSON.stringify(value)));
```

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

Reacting

- Validation rules
- Validation messages
- User interface elements
- Automatic suggestions
- And more ...

Demo

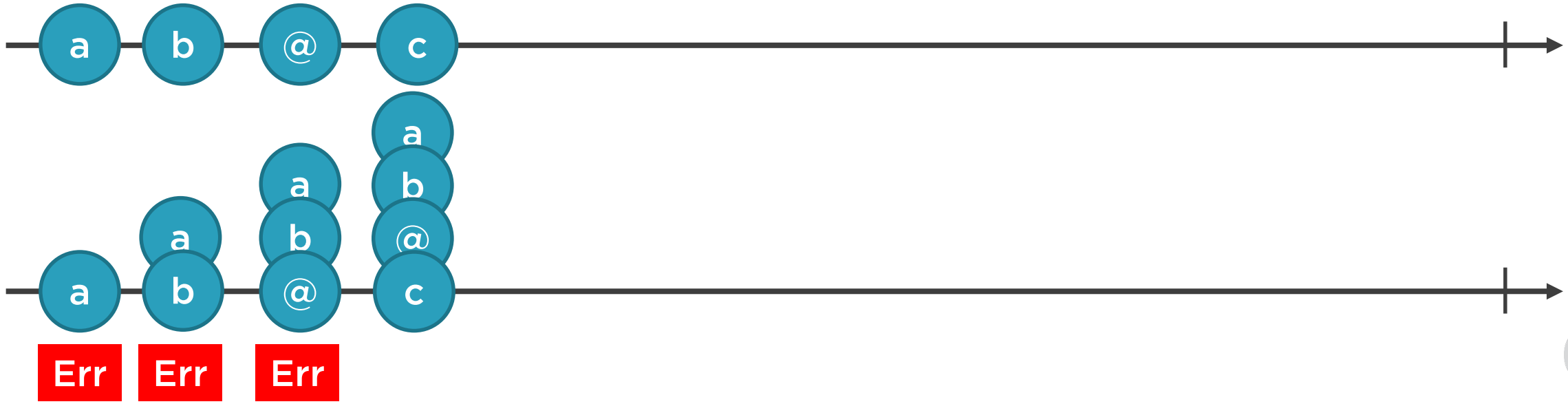


Displaying Validation Messages

Reactive Transformations

debounceTime

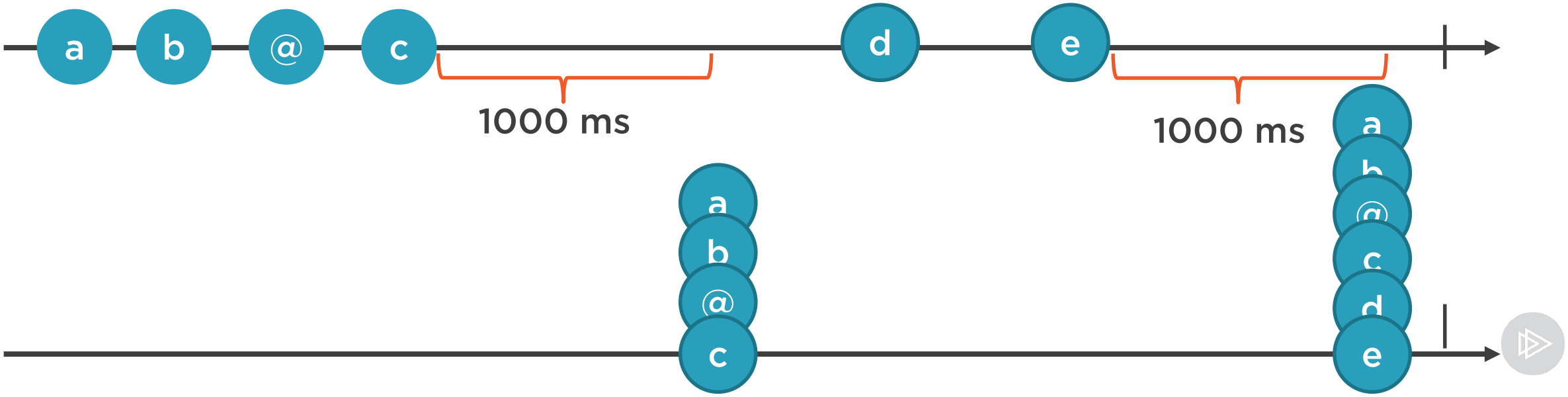
- Ignores events until a specific time has passed without another event
- `debounceTime(1000)` waits for 1000 milliseconds (1 sec) of no events before emitting another event



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

https://github.com/ReactiveX/rxjs/tree/master/src/operator

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

Add the operator

```
import 'rxjs/add/operator/debounceTime';
```

Use the operator

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

Summary

Watching

Reacting

Reactive Transformations

```
<div class="form-group"
    [ngClass]="{'has-error': (emailVar.touched
     || emailVar.dirty) && !emailVar.valid }">
  <label class="col-md-2 control-label"
         for="emailId">Email</label>
  <div class="col-md-8">
    <input class="form-control"
         id="emailId" type="email"
         placeholder="Email (required)"
         required
         pattern="[a-z0-9._%+-]+@[a-z0-9.-]+"
         [(ngModel)]="customer.email"
         name="email"
         #emailVar="ngModel" />
    <span class="help-block"
          *ngIf="(emailVar.touched
       || emailVar.dirty) && emailVar.errors">
      <span *ngIf="emailVar.errors.required">
         Please enter your email address.
      </span>
      <span *ngIf="emailVar.errors.pattern">
         The confirmation does not match the
email address.
      </span>
    </span>
  </div>
</div>
```

Template-driven

```
<div class="form-group"
     [ngClass]="{'has-error': emailMessage}">
  <label class="col-md-2 control-label"
         for="emailId">Email</label>

  <div class="col-md-8">
    <input class="form-control"
           id="emailId" type="email"
           placeholder="Email (required)"
           formControlName = "email" />
    <span class="help-block"
          *ngIf="emailMessage">
       {{ emailMessage }}
    </span>
   </div>
</div>
```

Reactive