



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

## Angular's Future with Signals



# Agenda

#1  
Motivation  
& Basics

#2  
DEMO

#3  
RxJS/NGRX  
Interop

#4  
Signal  
Components



# Motivation & Basics



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Change Detection (CD) in Angular



# Drawbacks

Zone.js:  
Magic

Zone.js:  
~100K

Cannot patch  
async/await

coarse  
grained CD

e.g. Components are always checked as a  
whole, even if only a tiny fraction changed



# How Do Other Frameworks Solve This?



SVELTE



SOLID



qwik

Signals!



@ManfredSteyer

# How Will Angular Solve This?

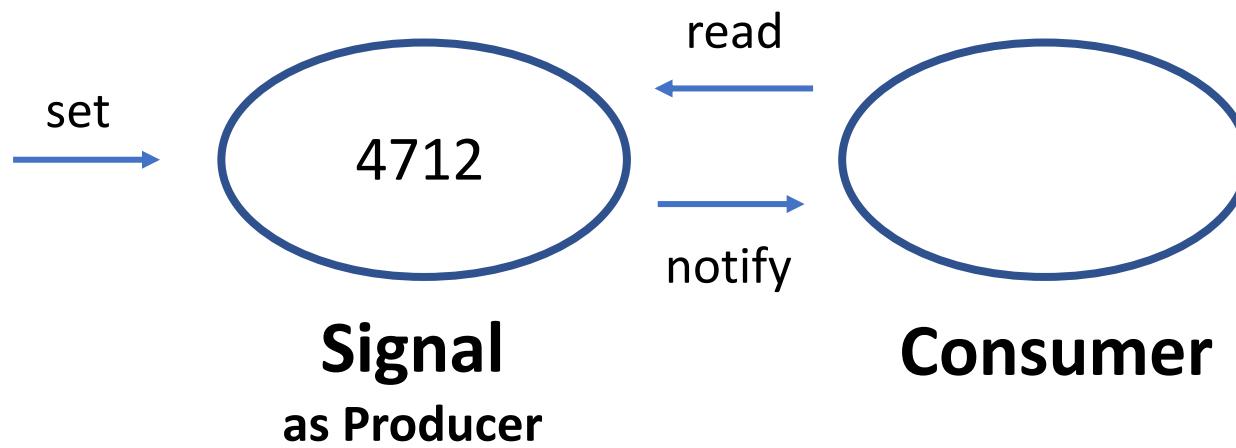


Signals!



@ManfredSteyer

# Signals: Simple Reactivity



# Component Without Signals

```
flights: Flight[] = [];

const flights = await this.flightService.findAsPromise(from, to);
this.flights = flights;

<div *ngFor="let f of flights">
    <flight-card [item]="f" />
</div>
```



# Component With Signals

```
flights = signal<Flight[]>([]);

const flights = await this.flightService.findAsPromise(from, to);
this.flights.set(flights);

<div *ngFor="let f of flights()">
    <flight-card [item]="f" />
</div>
```



# RxJS and NGRX Interop



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# @angular/core/rxjs-interop

toObservable(signal)

toSignal(observable\$)

takeUntilDestroyed()



# DEMO

# Outlook:

## Signal Components

(Since Angular 17.1 / .2 / .3)



# Signal Based Components (V17.1)

```
@Component({
  selector: 'app-temperature',
  template: `
    <p>C: {{ celsius() }}</p>
    <p>F: {{ fahrenheit() }}</p>
  `,
})
export class TemperatureComponent {
  celsius = signal(0);
  fahrenheit = computed(() => this.celsius() * 1.8 + 32);
}
```



# Signal Inputs (readonly, V17.1)

```
@Component({
  selector: 'app-temperature',
  template: `
    <p>C: {{ celsius() }}</p>
    <p>F: {{ fahrenheit() }}</p>
  `,
})
export class TemperatureComponent {
  // celsius = input(0); // InputSignal<number>
  // celsius = input.required(0); // InputSignal<number>
  celsius = input.required<number>(); // InputSignal<number>
  fahrenheit = computed(() => this.celsius() * 1.8 + 32);
}
```

```
<!-- parent component template -->
<app-temperature [celsius]="25" />
```



@ManfredSteyer

# Why Signal Inputs

- Will automatically mark **OnPush** components as dirty
- Type safety
  - Required inputs do not require initial values! 😊
    - or tricks to tell TypeScript that an input always has a value
  - Transforms are automatically checked
- Values can be easily derived upon changes using **computed()**
- Monitoring thru **effect()** instead of **ngOnChanges()**

# Signal View Queries (V17.2)

```
@Component({
  template: ' <div #el></div> <my-component />'
})
export class TestComponent {

  // query for a single result by a string predicate
  divEl = viewChild<ElementRef>('el'); // Signal<ElementRef|undefined>
  // query for a single result by a type predicate
  cmp = viewChild(MyComponent); // Signal<MyComponent|undefined>
}
```



# Signal Content Queries (V17.2)

```
@Component({...})
export class TestComponent {

    // query by a string predicate
    headerEl = contentChild<ElementRef>('h'); // Signal<ElementRef|undefined>

    // query by a type predicate
    header = contentChild(MyHeader); // Signal<MyHeader|undefined>
}
```



# Signal Outputs (V17.3)

```
@Component({...})
export class TemperatureComponent {
  tempChange = output<number>(); // OutputEmitterRef<number>

  // or
  tempChangeSubject = new Subject<number>();
  tempChange = outputFromObservable(this.tempChangeSubject); // OutputEmitterRef<number>
}
```

```
<!-- parent component template -->
<app-temperature (tempChange)="handleTempChange($event)" />
```



# Why Signal Outputs

- The API is conceptually **aligned** with the new APIs for signal inputs
- Simpler API and removed complexity that isn't relevant for outputs
- Automatic clean-up of outputs upon destruction (no unsubscribing)
- Improved type safety for new outputs & classical EventEmitters

# Signal Model (V17.3)

```
@Component({...})  
export class TemperatureComponent {  
    temperature = model(0); // writeable InputSignal<number> & OutputEmitterRef<number>  
}
```

```
<!-- parent component template -->  
<app-temperature [(temperature)]="myTemperature" />
```

```
<!-- parent component template with signal -->  
<app-temperature [temperature]="temperature" (temperatureChange)="updateTemperature($event)" />
```



# Signal Components

Work w/o  
Zone.js

Interop with  
traditional  
components

Fine-grained  
CD

No need to  
update code!



# Conclusion

Fine-grained  
CD

Zone-less  
Future

Convertible to  
Observables  
and vice versa!

No need to  
unsubscribe!

No need to  
update code!

