RxAngular
template syntax vs control flow

ngLeipzig#43 - 9th January 2024

Finatix

Finatix

About me

- Christian Illies
- 36yrs, married, 1 child, 1 dog
- Frontend software developer
- 10+ years XP
- Focussed on Angular since AngularJS/Angular2
- @kloener

Finatix

Agenda

1. Browser Scheduling

2. Angular Control Flow

3. RxAngular

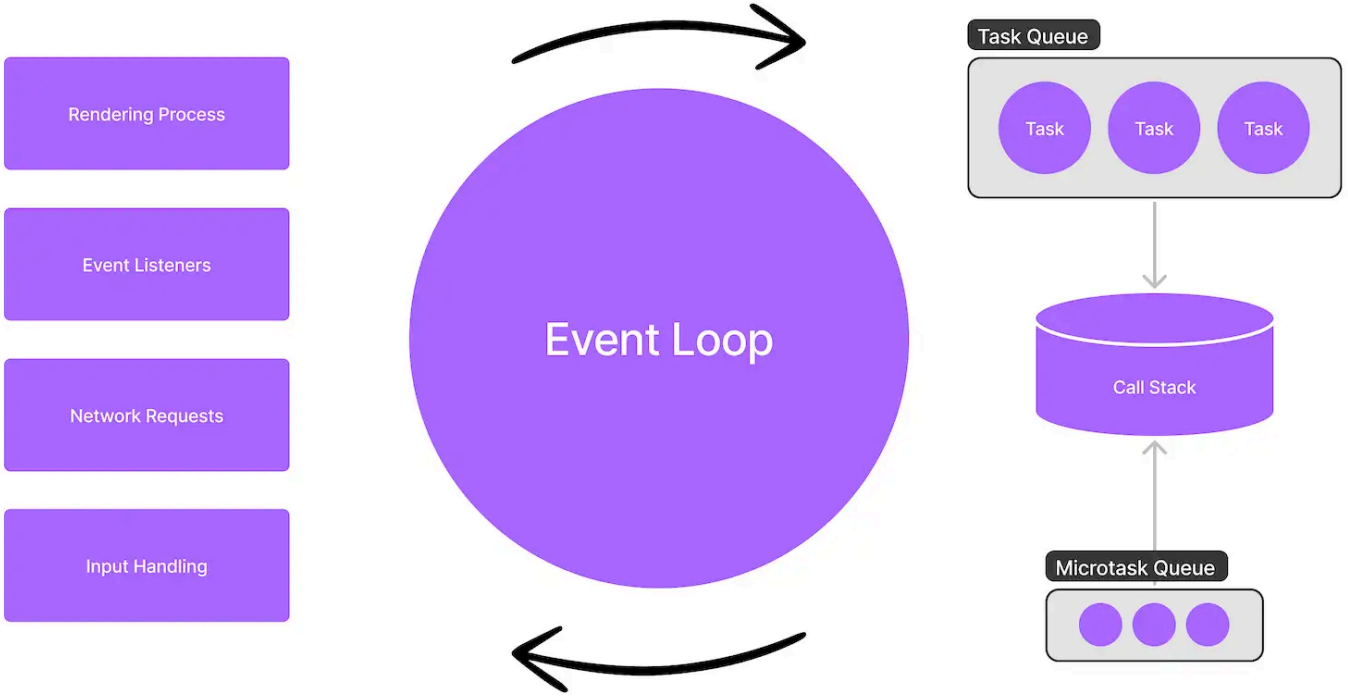
4. Comparison

Browser scheduling

How to schedule tasks?

Finatix

Browser scheduling



Browser scheduling

setTimeout

- MacroTask
- High prio, heavy work

requestIdleCallback

- MacroTask
- Low prio, heavy work

queueMicrotask & Promises

- MicroTask
- High prio, easy work

requestAnimationFrame

- Between Micro- and MacroTask Queues

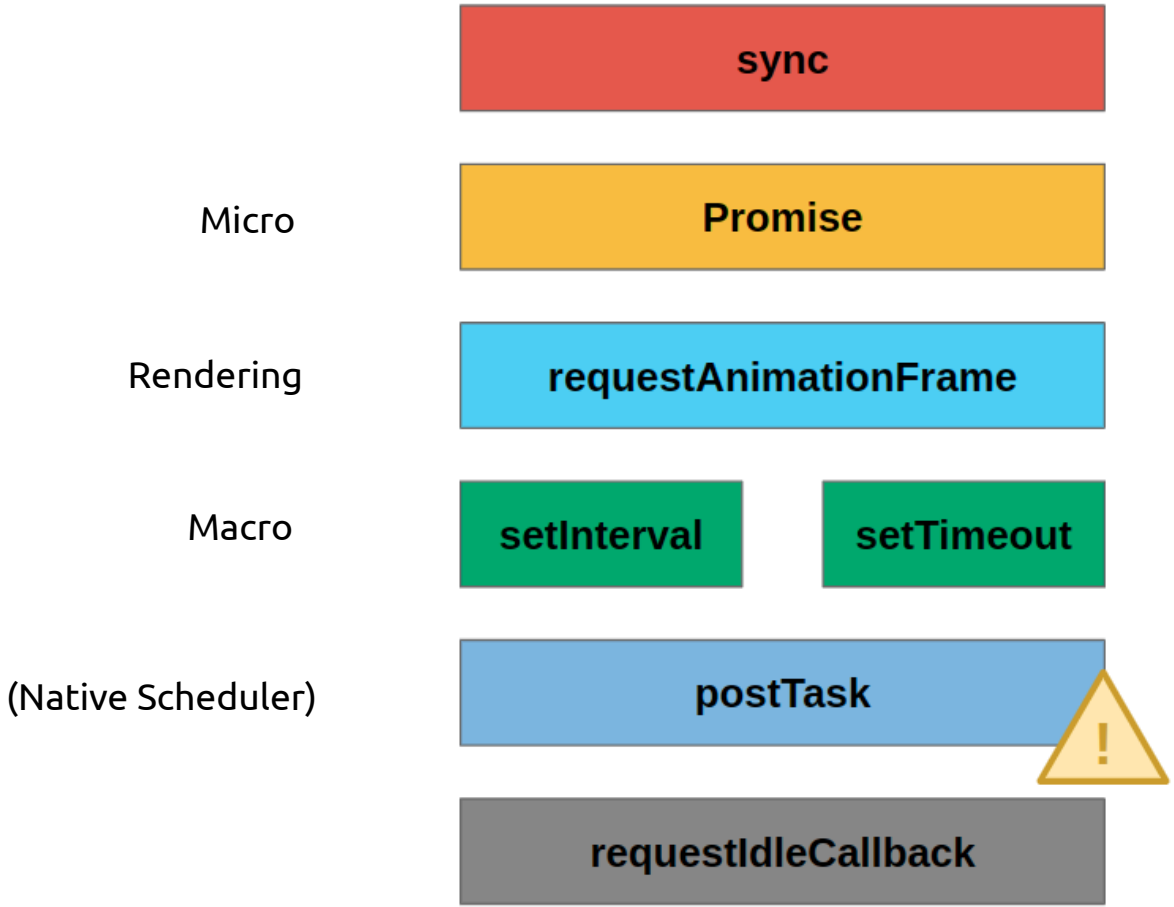
postMessage

- MacroTask
- u. a. für Webworker

scheduler.postTask

- MacroTask
- Native scheduling & prioritizing

Browser scheduling



Angular Control Flow

@if @defer @for

Finatix

Angular - Control Flow

@if

- New Angular Syntax for „if“ conditions
- No config for scheduling

@defer

- Lazy load components
- Load/Render Scheduling by conditions like viewport, interactions, idle, …

@for

- New Angular Syntax for „for“ conditions
- No config for scheduling

Angular only uses „requestAnimationFrame“ for scheduling/rendering,
see Github Source

Finatix

Angular - @for

- Render list all at once
- No scheduling

```
<ul #ul>
  @for(item of items; track item.id) {
   <li> … </li>
  }
</ul>
```

Finatix

Angular - @defer

- Render list but defer loading/rendering of each item
- Renders all items once the condition passes
- Schedule by viewport, idle, interactions, …

```
<ul #ul>
  @for(item of items; track item.id) {
    @defer (on viewport(ul); on hover; prefetch on idle) {
      <li> … </li>
    }
  }
</ul>
```

Finatix

RxAngular

CDK and Template directives

Finatix

RxAngular

- `npm i -S @rx-angular/template @rx-angular/cdk`

- Enhances rendering performance and enables non-blocking rendering in Angular
- Tasks can be scheduled by their requirements (tooltip, list, popups, …)
- Provides directives as a drop-in replacement for angular's structural directives

- Directives schedule rendering tasks of your template
- CDK provides the RxStrategyProvider to schedule tasks of your code

- Concurrent render strategies: immediate, userBlocking, normal, low, idle

RxAngular - RxFor

```
<ul>
  <li *rxFor=„
      let item of items;
      trackBy: ‚id';
      strategy: ‚normal‘;
      renderCallback: itemsRendered$">
    …
  </li>
</ul>
```

Finatix

RxAngular - RxLet

- ngIf Async Trick:
  - Typings are hard to handle due to null and undefined
  - Inefficient change detection (Evaluation of the whole template)
  - New but same values (1 => 1) still trigger change detection

```
<ng-container *ngIf="observableNumber$ | async as n">
  <app-number [number]="n"></app-number>
  <app-number-special [number]="n"></app-number-special>
</ng-container>
```

- RxLet
  - Allows fine-grained control of change detection on a per directive basis.
  - Contextual state: suspense, error, complete, …

```
<ng-container *rxLet=„observableNumber$; let n">
  <app-number [number]="n"></app-number>
  <app-number-special [number]="n"></app-number-special>
</ng-container>
```

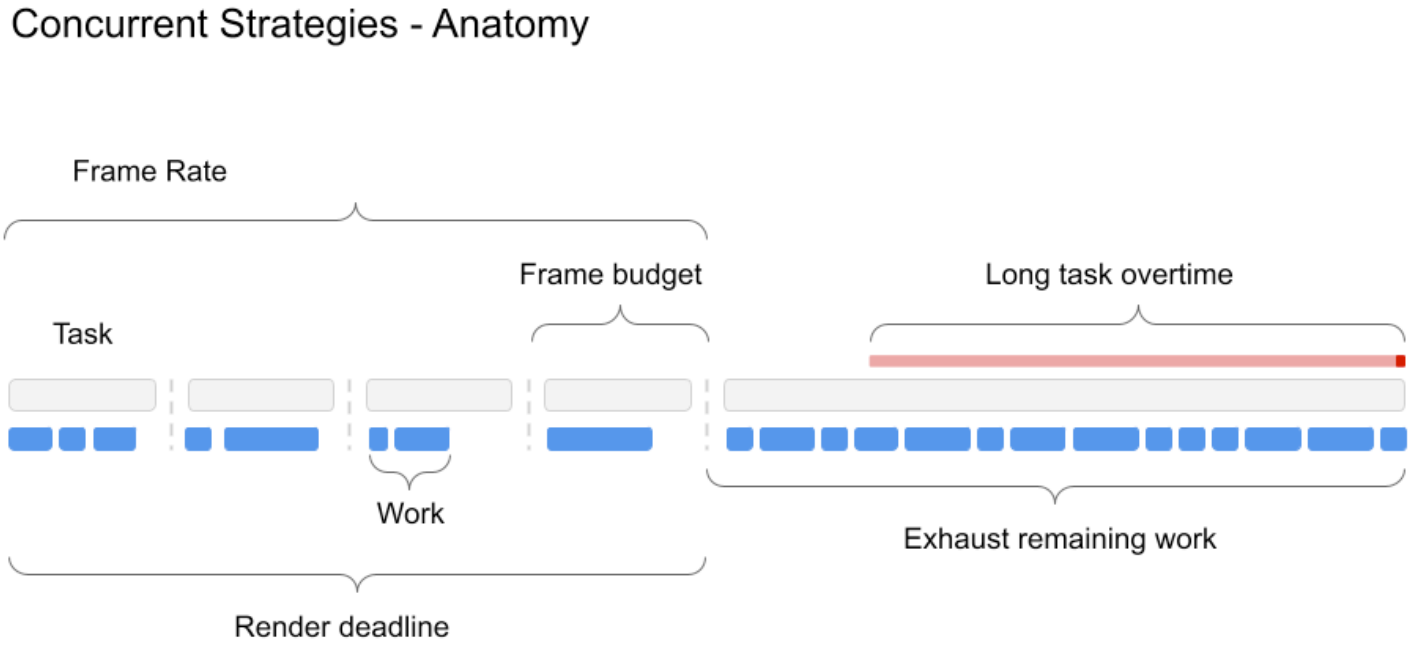
RxAngular - RxStrategyProvider

```
strategyProvider = inject(RxStrategyProvider);
```

```
  this.strategyProvider
    .schedule(() => this.workA(), { strategy: 'idle' })
    .subscribe();

  this.strategyProvider
    .schedule(() => this.workB(), { strategy: 'low' })
    .subscribe();

  this.strategyProvider
    .schedule(() => this.workC(), { strategy: 'normal' })
    .subscribe();

  this.strategyProvider
    .schedule(() => this.workD(), { strategy: 'userBlocking' })
    .subscribe();

  this.strategyProvider
    .schedule(() => this.workE(), { strategy: 'immediate' })
    .subscribe();
```

Finatix

RxAngular - Strategies



RxAngular - Strategies

e. g. Tooltips

e. g. Dropdown Menu

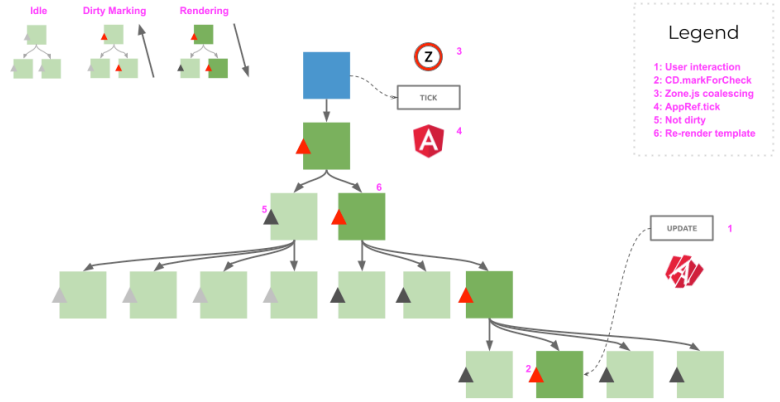
e. g. Lists and Cards

e. g. Lazy dialogs

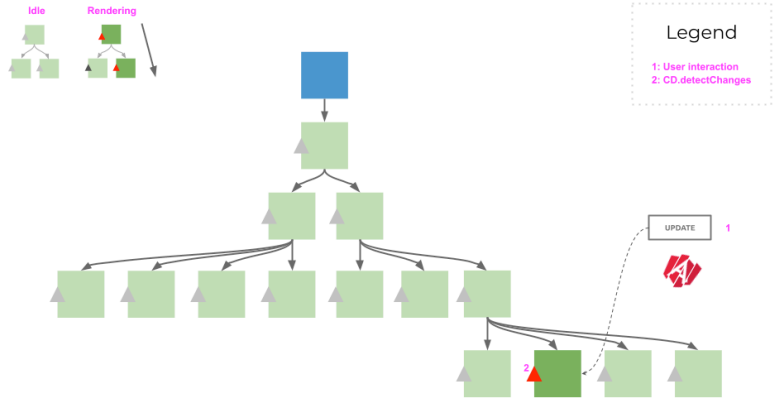
e. g. Background sync

RxAngular - Strategies

- Non-concurrent strategies
  - Native, uses `markForCheck` and re-evaluates from the root of the component tree
  - Local, uses `detectChanges` and re-evaluates from the affected component and its children.
  - Both use `requestAnimationFrame` for scheduling



Native Strategy



Local Strategy

RxAngular - Testing

- Testing can be hard
- Override global primary render strategy using RX_RENDER_STRATEGIES_CONFIG token
- Provide Inputs for strategies
- Or wait for the elements to be rendered using renderCallback

Comparison

Demo

Finatix

Thank you!

Questions?

Ressources

- Samples Repository

- Angular
- Angular @defer docs

- RxAngular
- RxAnguler Template Directives
- RxAngular StrategyProvider

- Others
- Explanation of the Event Loop
- WebAPI: Scheduler