Angular Change Detection

...

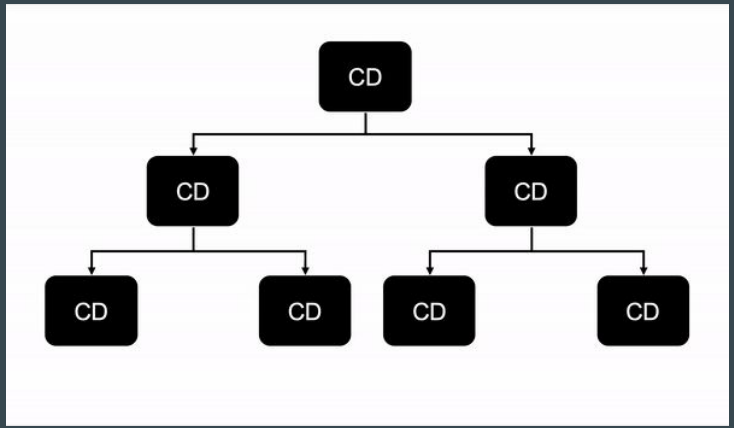
~~The good~~, the bad, and the ugly

(there is no good :')

Angular Change Detection

A mechanism used to update view and reflect view model changes on the UI

The Angular Change Detection checks for all components from **top to bottom**



Change Detection strategies

When does a component re-render?

Default

- There is an DOM/Browser event (click, setTimeout, ajax call, web socket)
- Deep check on every prop on the component

OnPush

- The `Input` reference changes
- An event originated from the component or one of its children
- An observable linked to the template via the **push pipe** (async) emits a new value
- Explicit local Change Detection cycle (detectChanges)

Gotchas

Default

- Easy to implement
- Slow

OnPush

- Immutability
- “faster”

But wait, I lied...

None of these events actually trigger a change detection cycle

- The `Input` reference changes
- An event originated from the component or one of its children
- An observable linked to the template via the **push pipe** emits a new value

They just mark the component **dirty**, meaning that a re-render will occur sometimes in the future

The Push pipe (async)

Every time the subscribed observable emits a value, it will mark the component dirty, by calling **ChangeDetectorRef.markForCheck**

Source code: https://github.com/angular/angular/blob/8.2.x/packages/common/src/pipes/async_pipe.ts

So who actually runs a Change Detection cycle?

Zone.js & NgZone

Zone.js is an execution context that helps developers intercept and keep track of async operations.

NgZone is the Angular wrapper of Zone.js and it is used to automatically trigger a global Change Detection cycle when an async operation completes.

NgZone.onMicroTaskEmpty subscription => https://github.com/angular/angular/blob/8.2.x/packages/core/src/application_ref.ts#L523
ApplicationRef.tick() => https://github.com/angular/angular/blob/8.2.x/packages/core/src/application_ref.ts#L633

Very detailed explanation: https://medium.com/angular-in-depth/i-reverse-engineered-zones-zone-js-and-here-is-what-ive-found-1f48dc87659b

Putting all together

1. Event occurred inside the component: the component **and its ancestors** are marked as dirty
2. The same event is intercepted by the NgZone, that runs a global change detection cycle
3. The CD cycle, starting from the root component and following the component tree top to bottom, will check if any component need to be re-rendered

4. If yes, executes a re-render
4. If no, goes to the next leaf

(assume a OnPush change detection strategy)

Manually trigger change detection

Global

ApplicationRef.tick: starts from the root component and walks down the component tree

Local(component) - ChangeDetectorRef

detectChanges: executes a synchronous local change detection cycle

markForCheck: doesn't actually run a re-render, but it only marks the component **and its ancestor** as dirty, meaning that they will be re-rendered on the next change detection cycle

Solution? (for our dreams)

Remove Zone.js, detach and implement a local change detection behaviour for each component

Pros

- Extreme performance, since components re-render only when is really needed.

Cons:

- Most(all?) 3rd party libraries rely on zones+global CD and they could misbehave

Solution? (for our lives)

Detach change detection on selected components and implement a local change detection behaviour

Pros

- Extreme performance for selected components

Cons:

- Since detaching a view, will detach also all the children, all the children components must implement a local change detection behaviour.
- The components that use a 3rd party library, will keep the default behaviour

Q & A

Demo source:
https://github.com/crushjz/angular-8-changedetection-talk