Reactive Angular met RxJS

Change Detection

Peter Kassenaar –
info@kassenaar.com

What is Change Detection

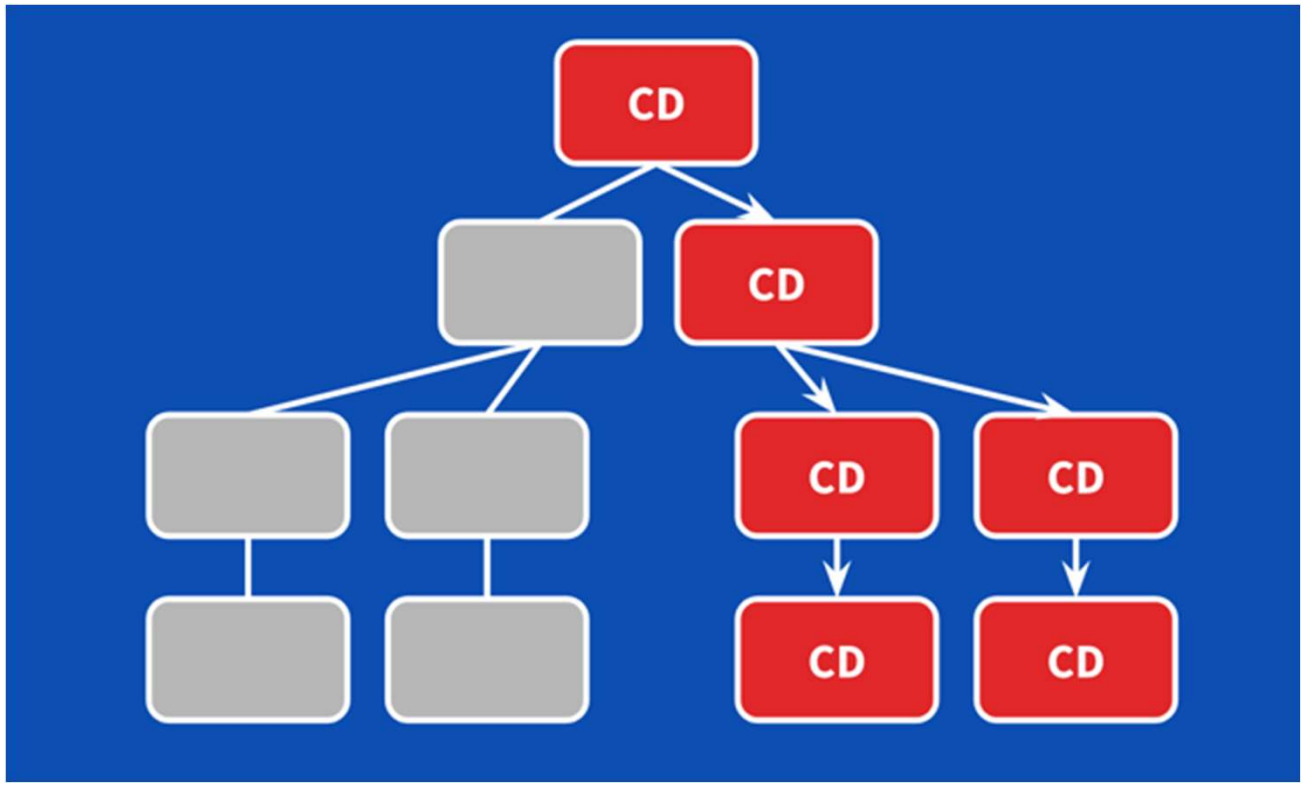
- Angular performs *Change Detection* every time something changes:
  - In the View (mouseclick, mousemove, etc)
  - In the controller (data change, network request, subscription, etc)
- The view gets updated with changes in the model,
- The model gets updated with changes in the view
  - Changes are propagated to child components!

Where’s the problem?

CD can be costly in bigger apps with lots of nested components!

Extra – about Change detection

- Angular uses `zone.js` to perform change Detection
- Angular has some strategies to optimize CD



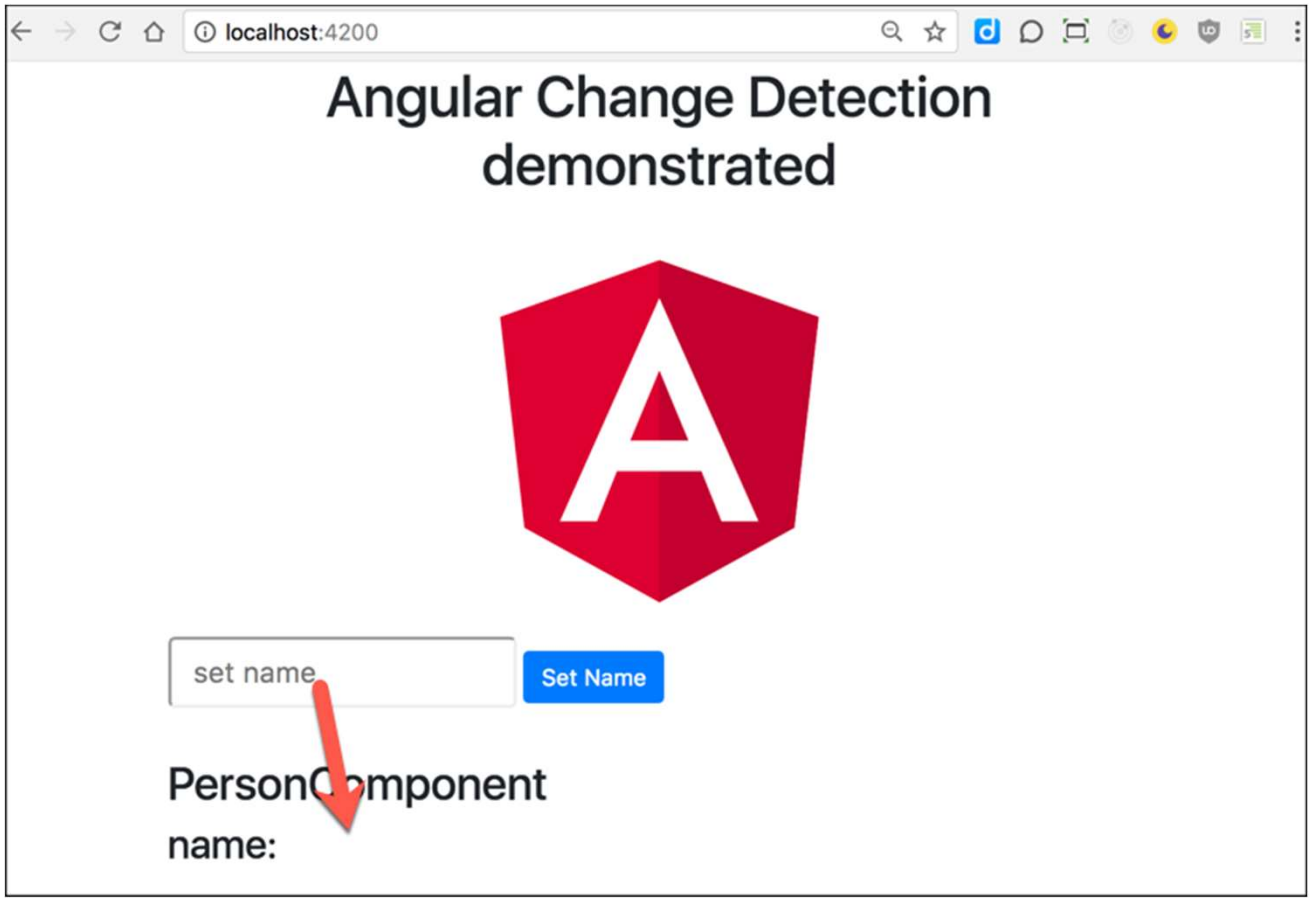
Using CD

- But we can do better.
- *Only* call CD when we want to:
- `changeDetection: ChangeDetectionStrategy.OnPush` in component decorator
- CD runs only when a new value is pushed to the component
  - NOT when properties of an object change

Using custom CD strategy is one of the biggest and easiest performance wins in Angular

Example

- Example `../800-change-detection`



Demo

- General rule: nested components are inside *one* view.
  - If the parent runs a change detection cycle, changes are propagated to the children.
  - Unless they have `.onPush()` activated
- `…/800-change-detection`
- (un) comment various lines

Defining Custom Change Detection

ANGULAR FEATURES DOCS RESOURCES EVENTS BLOG Search

Defining custom change detection

To monitor changes that occur where `ngOnChanges()` won't catch them, you can implement your own change check, as shown in the *DoCheck* example. This example shows how you can use the `ngDoCheck()` hook to detect and act upon changes that Angular doesn't catch on its own.

The *DoCheck* sample extends the *OnChanges* sample with the following `ngDoCheck()` hook:

DoCheckComponent (ngDoCheck)

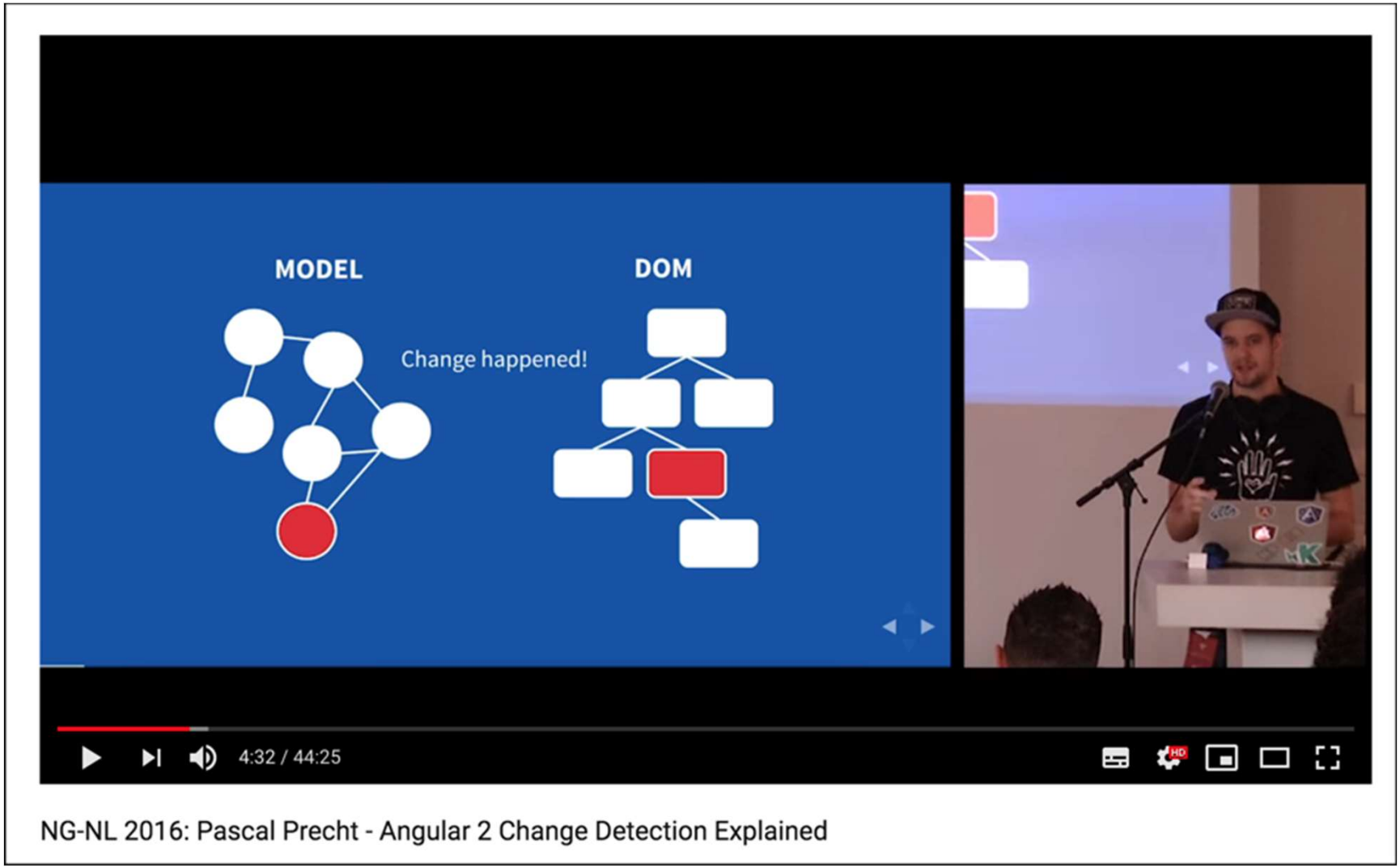
```
ngDoCheck() {

  if (this.hero.name !== this.oldHeroName) {
    this.changeDetected = true;
    this.changeLog.push(`DoCheck: Hero name changed to "${this.hero.name}" from
"${this.oldHeroName}"`);
    this.oldHeroName = this.hero.name;
  }

  if (this.power !== this.oldPower) {
    this.changeDetected = true;
    this.changeLog.push(`DoCheck: Power changed to "${this.power}" from
"${this.oldPower}"`);
    this.oldPower = this.power;
  }

  if (this.changeDetected) {
```


https://angular.io/guide/lifecycle-hooks#defining-custom-change-detection

Alligator.io

https://alligator.io/angular/change-detection-strategy/

More Info



https://www.youtube.com/watch?v=CUxD91DWkGM

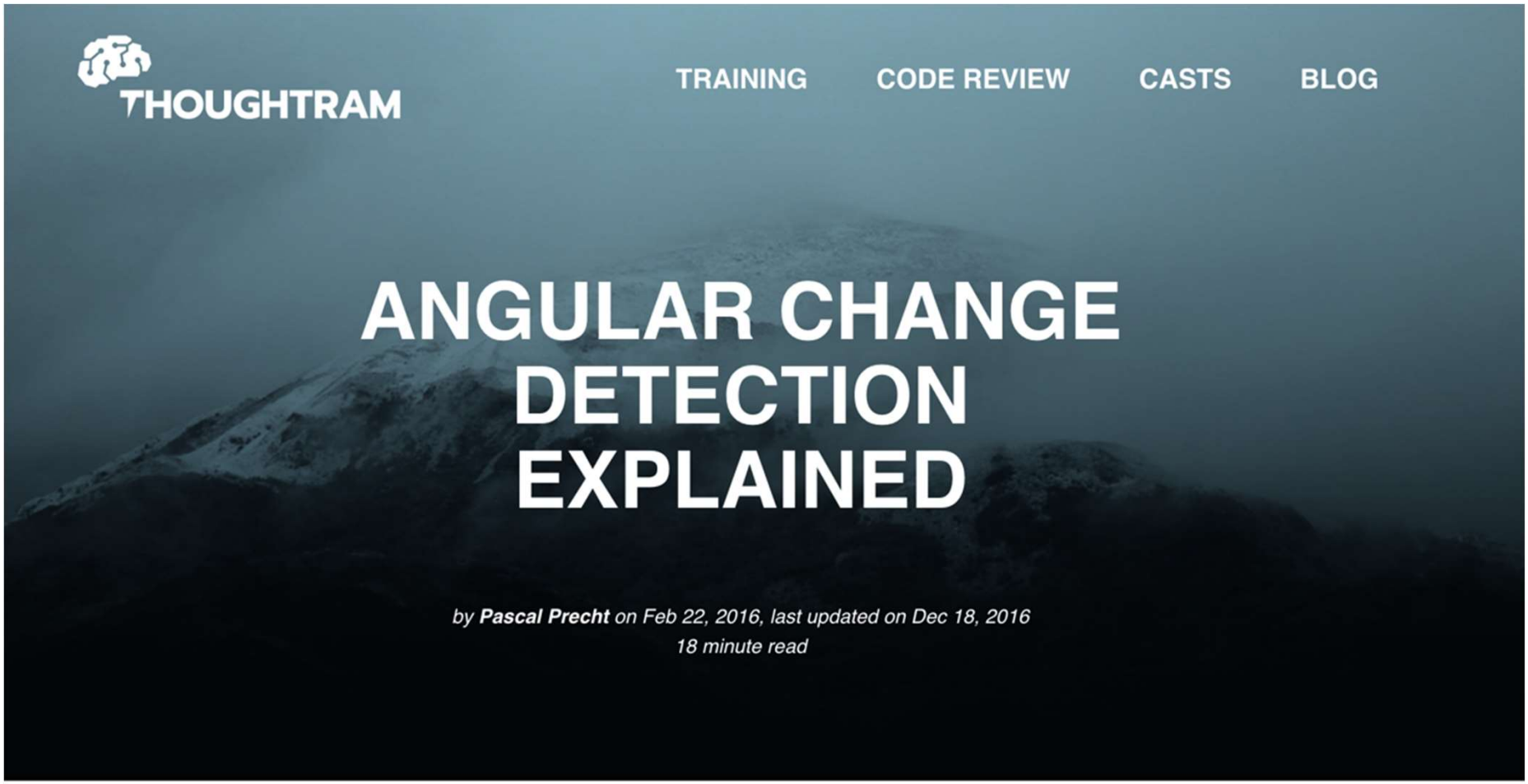
More on Change Detection

https://blog.angularindepth.com/a-gentle-introduction-into-change-detection-in-angular-33f9ffff6f10

Change Detection – deep dive

https://blog.angularindepth.com/everything-you-need-to-know-about-change-detection-in-angular-8006c51d206f

Thoughtram



https://blog.thoughtram.io/angular/2016/02/22/angular-2-change-detection-explained.html

Using zones

When and how to use zones.js

What is zone.js?

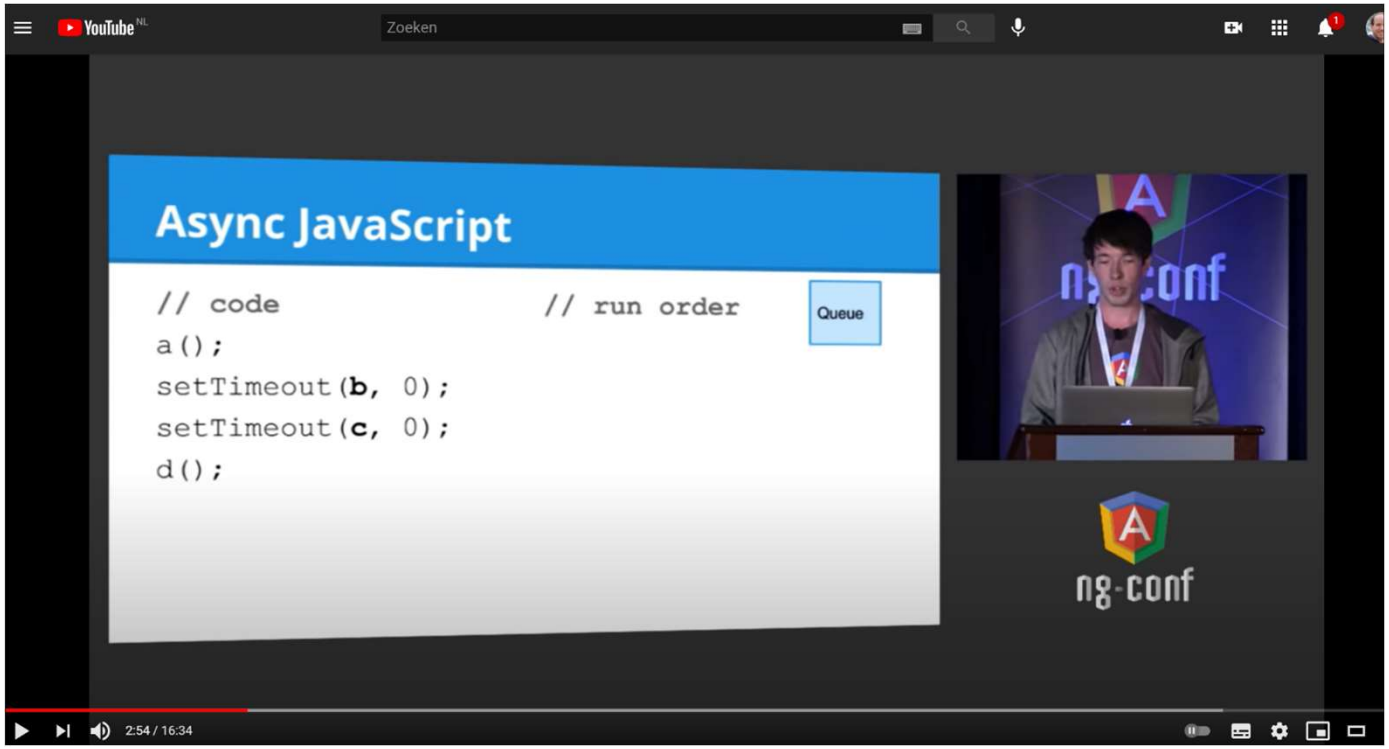
*"In **Angular**, Zone.js is used to detect when certain async operations occur to trigger a change detection cycle."*

"Zone.js is an execution context that helps developers intercept and keep track of async operations. Zone works on the concept of associating each operation with a zone."

https://blog.bitsrc.io/how-angular-uses-ngzone-zone-js-for-dirty-checking-faa12f98cd49

Zones

- Originally in Dart
- Ported to JavaScript by Brian Ford



https://www.youtube.com/watch?v=3lqtmUscE_U

Zones in Angular

- By default included in Angular
- To capture all async operations (`click`, `scroll`, `mouseover`, `keyup`, `keydown`, etc) and re-render the component when a change happens
- So you don't have to do that yourself.
- It mostly works out-of-the-box…
- …Until it doesn't

https://angular.io/guide/zone

ANGULAR FEATURES DOCS RESOURCES EVENTS BLOG Search

NgZone

A zone is an execution context that persists across async tasks. You can think of it as thread-local storage for JavaScript VMs. This guide describes how to use Angular's NgZone to automatically detect changes in the component to update HTML.

Fundamentals of change detection

To understand the benefits of `NgZone`, it is important to have a clear grasp of what change detection is and how it works.

Displaying and updating data in Angular

In Angular, you can display data by binding controls in an HTML template to the properties of an Angular component.

src/app/app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
    <h1>{{title}}</h1>
    <h2>My favorite hero is: {{myHero}}</h2>
```

From the Angular documentation

Detecting changes with plain JavaScript

To clarify how changes are detected and values updated, consider the following code written in plain JavaScript.

```
<html>
  <div id="dataDiv"></div>
  <button id="btn">updateData</button>
  <canvas id="canvas"></canvas>
  <script>
    let value = 'initialValue';
    // initial rendering
    detectChange();

    function renderHTML() {
      document.getElementById('dataDiv').innerText = value;
    }

    function detectChange() {
      const currentValue = document.getElementById('dataDiv').innerText;
      if (currentValue !== value) {
        renderHTML();
      }
    }

    // Example 1: update data inside button click event handler
    document.getElementById('btn').addEventListener('click', () => {
      // update value
      value = 'button update value';
      // call detectChange manually
      detectChange();
    });
```

https://angular.io/guide/zone

Zone.js detect changes for you and rerenders

- You [only] need to inject/use Zone yourself if you
  - Want to do change detection manually
  - When a 3rd party API creates a change that zone.js doesn't handle
- For instance:

There are still some third party APIs that Zone does not handle. In those cases, the `NgZone` service provides a `run()` method that allows you to execute a function inside the Angular zone. This function, and all asynchronous operations in that function, trigger change detection automatically at the correct time.

```
export class AppComponent implements OnInit {
  constructor(private ngZone: NgZone) {}
  ngOnInit() {
    // New async API is not handled by Zone, so you need to
    // use ngZone.run() to make the asynchronous operation in the Angular zone
    // and trigger change detection automatically.
    this.ngZone.run(() => {
      someNewAsyncAPI(() => {
        // update the data of the component
      });
    });
  }
}
```

Shortcut:

"Use zone.js manually if you expect a change detection cycle and rerender to occur, but it doesn't happen"

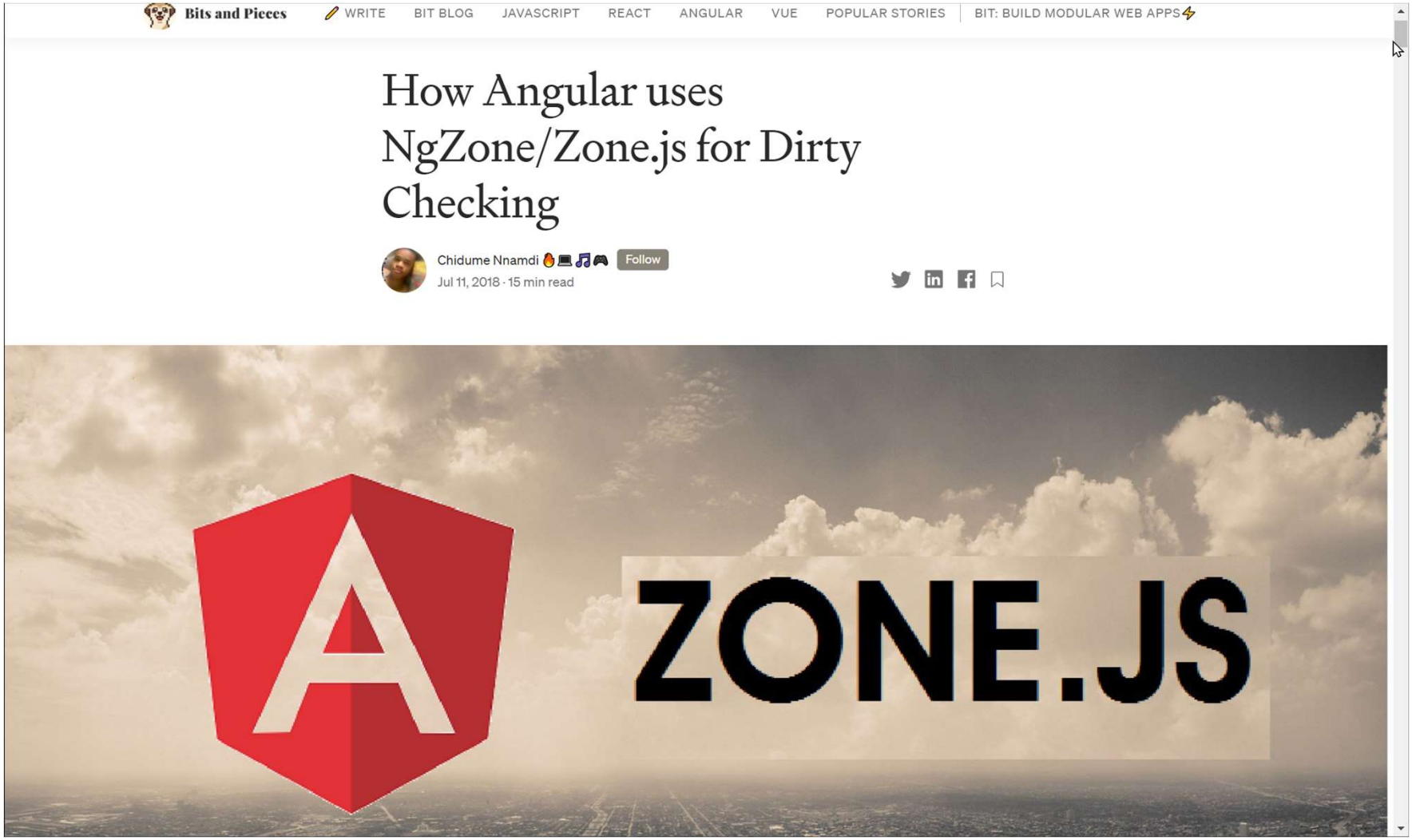
No rerender?

- Then inject `ngZone` and run that task inside a zone
- Or use `zone.tick()` or `zone.run()` to update the DOM after the unsupported async operation has completed.
- You *want* to run a task outside Angular? (for performance reasons or the like): use `runOutsideAngular()`

By default, all asynchronous operations are inside the Angular zone, which triggers change detection automatically. Another common case is when you don't want to trigger change detection. In that situation, you can use another `NgZone` method: `runOutsideAngular()`.

```
export class AppComponent implements OnInit {
  constructor(private ngZone: NgZone) {}
  ngOnInit() {
    // You know no data will be updated,
    // so you don't want to trigger change detection in this
    // specified operation. Instead, call ngZone.runOutsideAngular()
    this.ngZone.runOutsideAngular(() => {
      setTimeout(() => {
        // update component data
        // but don't trigger change detection.
      });
    });
  }
}
```

Deep dive


Bits and Pieces
WRITE
BIT BLOG
JAVASCRIPT
REACT
ANGULAR
VUE
POPULAR STORIES
BIT: BUILD MODULAR WEB APPS
How Angular uses NgZone/Zone.js for Dirty Checking
Chidume Nnamdi
Follow
Jul 11, 2018 · 15 min read
ZONE.JS

Thoughtram

Understanding Zones

At NG-Conf 2014, Brian gave an excellent talk on zones and how they can change the way we deal with asynchronous code. If you haven't watched this talk yet, give it a shot, it's just ~15 minutes long. APIs might be different nowadays, but the semantics and underlying concepts are the same. In this article we'd like to dive a bit deeper into how zones work.

The problem to be solved

Let's recap really quick what zones are. As Brian stated in his talk, they are basically **an execution context** for asynchronous operations. They turn out to be really useful for things like error handling and profiling. But what exactly does that mean?

In order to understand the execution context part of it, we need to get a better picture of what the problem is that zones are trying to solve. Let's first take a look at the following JavaScript code.

```
foo();
bar();
baz();

function foo() {...}
function bar() {...}
function baz() {...}
```

Nothing special going on here. We have three functions `foo`, `bar` and `baz` that are executed in sequence. Let's say we want to measure the execution time of this code. We could

https://blog.thoughtram.io/angular/2016/01/22/understanding-zones.html

What Is Zone.js and How Can I Use It?

arguments passed during the invocation of *setTimeout*, the way of scheduling and invoking the *setTimeout* (or any other async API).

ZoneTask are classified into three types:

- *MacroTask — setTimeout(), cleatTimeout(), setInterval()……*
- *MicroTask — Promise.then()*
- *EventTask — element.addEventListener()*

ZoneTask has a member named type which can have a value of either *microTask*, *macroTask* or *eventTask*. Structure of *ZoneTask* is as belows :

```
class ZoneTask<T extends TaskType> implements Task {
    // Refers to the type if Task's can be `MicroTask`, `MacroTask` or `EventTask`
    type: T;

    // Refers to the source of task
    source: string;

    // Refers to the function which executes the logic of invoking the `callback`
    invoke: Function;

    // Refers to the callback which is supposed to be called as a part of `Task`
    callback: Function;

    // Refers to the extra data for `Task`
    data: TaskData|undefined;

    // Refers to the function which executes the logic of the scheduling the task
    scheduleFn: ((task: Task) => void)|undefined;

    // Refers to the function which executes the logic of the cancelling the task
    cancelFn: ((task: Task) => void)|undefined;
```

https://medium.com/swlh/what-is-zone-js-and-how-can-i-use-it-63ce08a55962