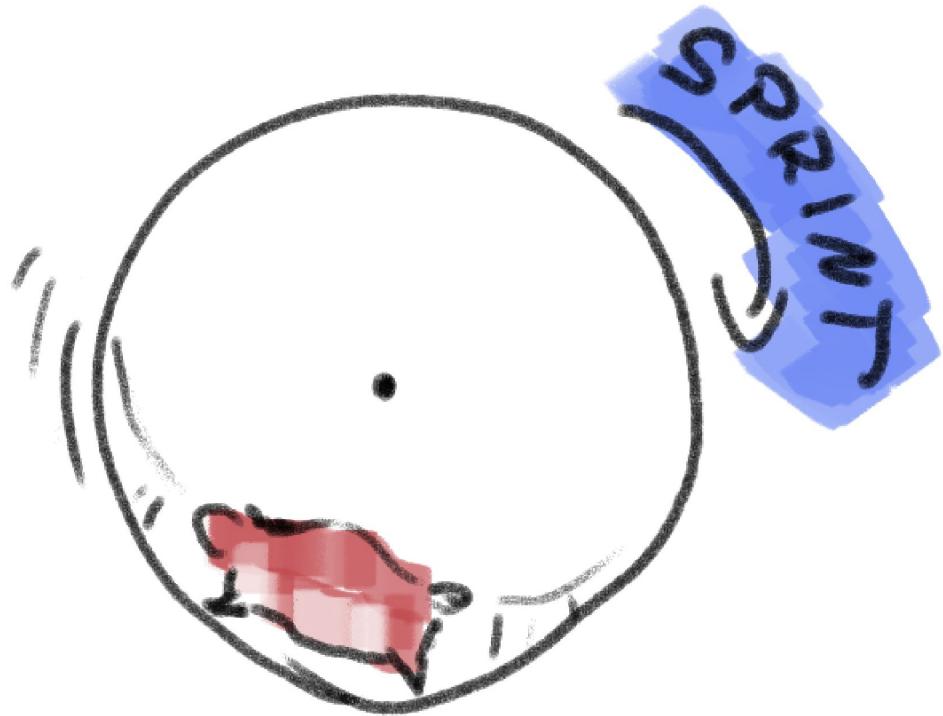


• • •

Sport vs Extreme Sport

Experience != Knowledge

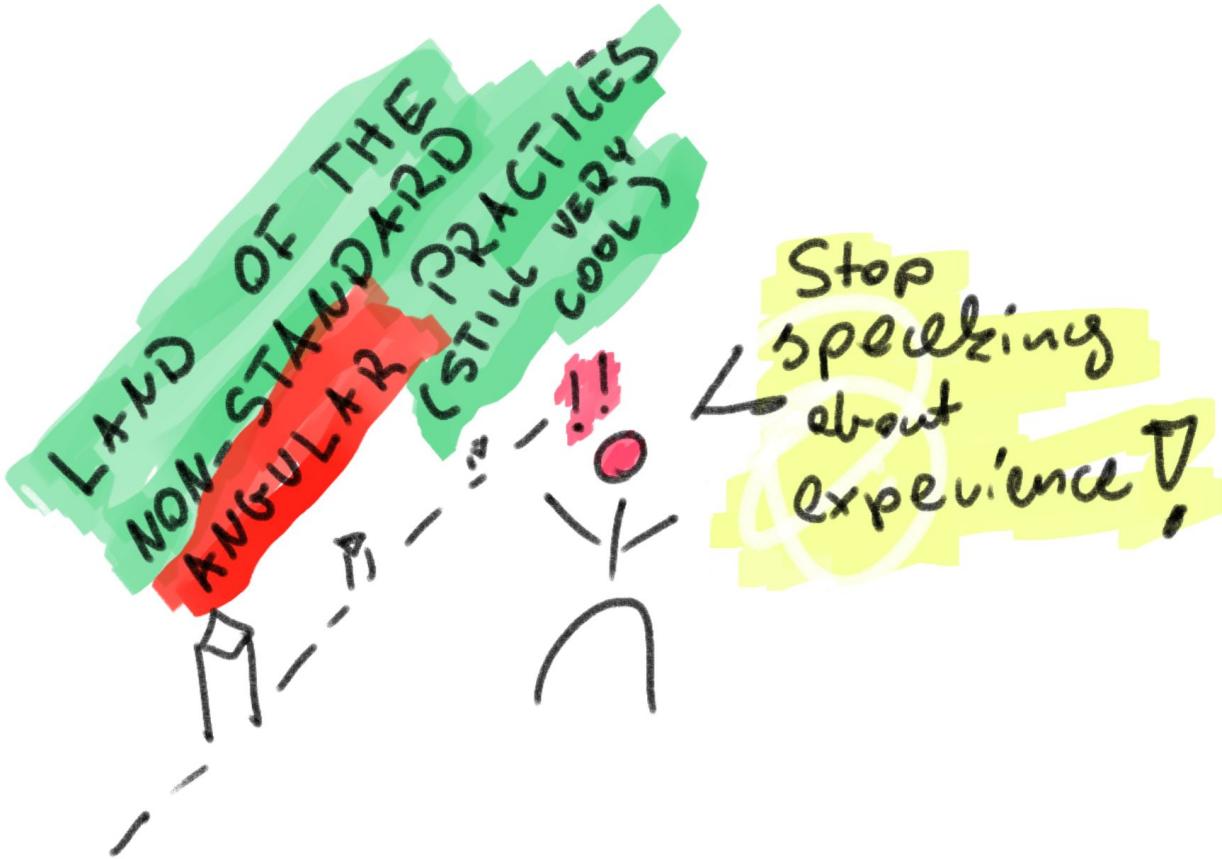




How to Become an Angular Pro Developer in 2022-11 Easy Steps to Follow.

These are some of the most important things you should follow while working on the Angular apps, if you're a Pro Developer. Create your...





How to become an

EXTREME

Angular programmer
in 30 minutes



Kamil Gałek

Frontend Developer
BigPicture



BigPicture

mythical-angular.dev



Search or jump to...

Pull requests Issues Codespaces Marketplace Explore



galcz05 / recipes Public

Pin

Unwatch 2

Fork 0

Star 0

Code Issues 2 Pull requests Actions Projects Wiki Security Insights Settings

master

1 branch

0 tags

Go to file

Add file

Code



galcz05 Daikon and Carrots

0beaef9 2 days ago 24 commits



src

new channels and recipes

4 months ago



README.md

Daikon and Carrots

2 days ago



image.gif

new idea for this repo

3 years ago

README.md



Drinks

Limoncello

Watermelon Agua Fresca

Meat

About

No description, website, or topics provided.

Readme

0 stars

2 watching

0 forks

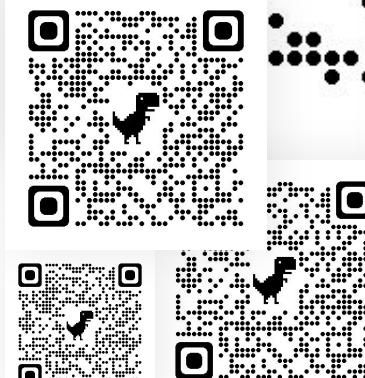
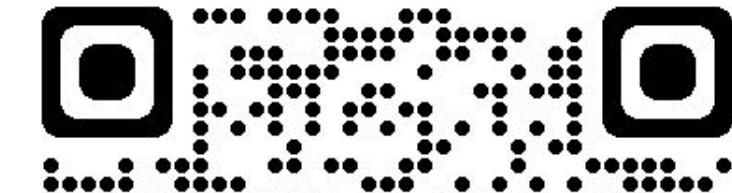
Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

30 min + 5 min for questions
but I'll be here so if you have any question I'll
try to answer it





How to become an

EXTREME

Angular programmer
in 30 minutes

Is it even possible?

No, it's not

WHAT?!

SO WHY
I'M HERE?



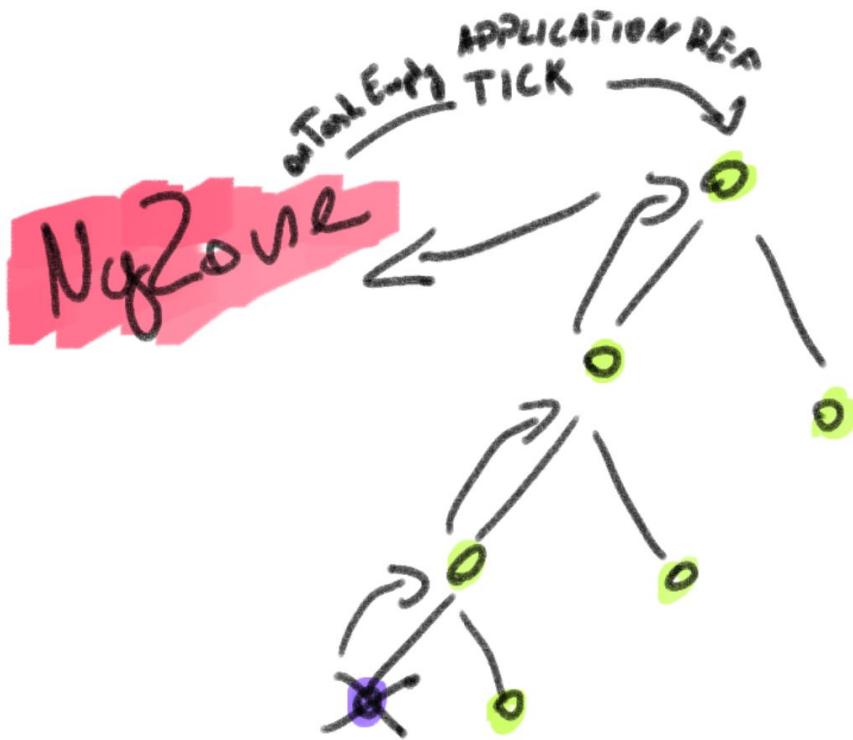
How to become an

EXTREME

Angular programmer
in 3-5 years

Please don't go out...
30 min it's a good time to start

NgZone

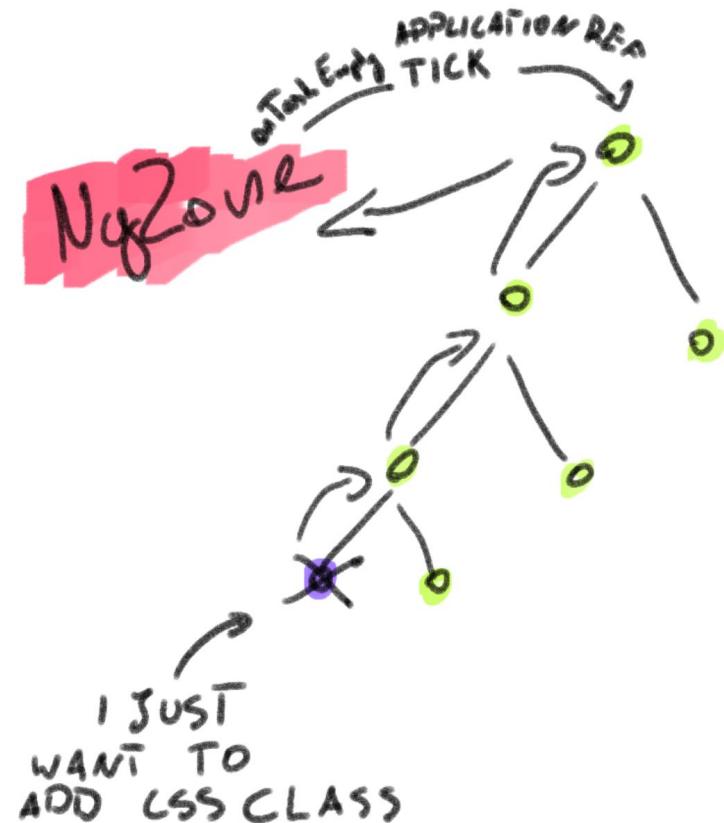


The screenshot shows a code editor interface with the following details:

- Title Bar:** angular > main
- Project Explorer:** Shows the directory structure of the angular project, including folders like di, hydration, i18n, interface, linker, metadata, reflection, render, render3, sanitization, signals, testability, util, view, zone, and various application-related files.
- Current File:** application_ref.ts
- Code Editor:** The file content is displayed, starting with a public readonly property `isStable` and its implementation. It includes comments about the `EnvironmentInjector` used to create the application, and a constructor that initializes `_zone`, `_injector`, and `_exceptionHandler`. The code then handles microtask empty events by running the zone and ticks, and defines an observable `isCurrentlyStable` that checks if the zone is stable and has pending macrotasks.
- Status Bar:** Shows the file path (angular > packages > core > src > application_ref.ts), file statistics (778:53 LF UTF-8 2 spaces*), and the TypeScript version (5.0.0).

```
public readonly isStable: Observable<boolean>;  
/*  
 * The `EnvironmentInjector` used to create this application.  
 */  
get injector(): EnvironmentInjector {  
    return this._injector;  
}  
/** @internal */  
constructor(  
    private _zone: NgZone,  
    private _injector: EnvironmentInjector,  
    private _exceptionHandler: ErrorHandler,  
) {  
    this._onMicrotaskEmptySubscription = this._zone.onMicrotaskEmpty.subscribe({  
        next: () => {  
            this._zone.run(() => {  
                this.tick();  
            });  
        }  
    });  
  
    const isCurrentlyStable = new Observable<boolean>((observer: Observer<boolean>) => {  
        this._stable = this._zone.isStable && !this._zone.hasPendingMacrotasks &&  
            !this._zone.hasPendingMicrotasks;  
        this._zone.runOutsideAngular(() => {  
            observer.next(this._stable);  
            observer.complete();  
        });  
    });  
}
```

What's the problem?



How to deal with NgZone...?

The screenshot shows a dark-themed code editor interface with a project structure on the left and a code editor on the right.

Project Structure:

- ng-inject-lightning-talk
- main
- Project
- src
 - app-routing.module.ts
 - assets
 - util
 - useHostBinding.spec.ts
 - useHostBinding.ts
 - useHostListen.spec.ts
 - useHostListen.ts
 - useOnChanges.spec.ts
 - useOnChanges.ts
 - useOnDestroy.spec.ts
 - useOnDestroy.ts
 - useZonelessHostListen.spec.ts
 - useZonelessHostListen.ts
 - favicon.ico
 - index.html
 - main.ts
 - styles.css
 - .editorconfig
 - .gitignore
 - angular.json
 - package.json
 - package-lock.json
 - README.md
 - tsconfig.app.json
 - tsconfig.json
 - tsconfig.spec.json
 - yarn.lock- External Libraries
- Scratches and Consoles
 - Scratches
 - scratch.txt
 - scratch_1.css
 - scratch_2.json
 - scratch_3.js

Code Editor (main.ts):

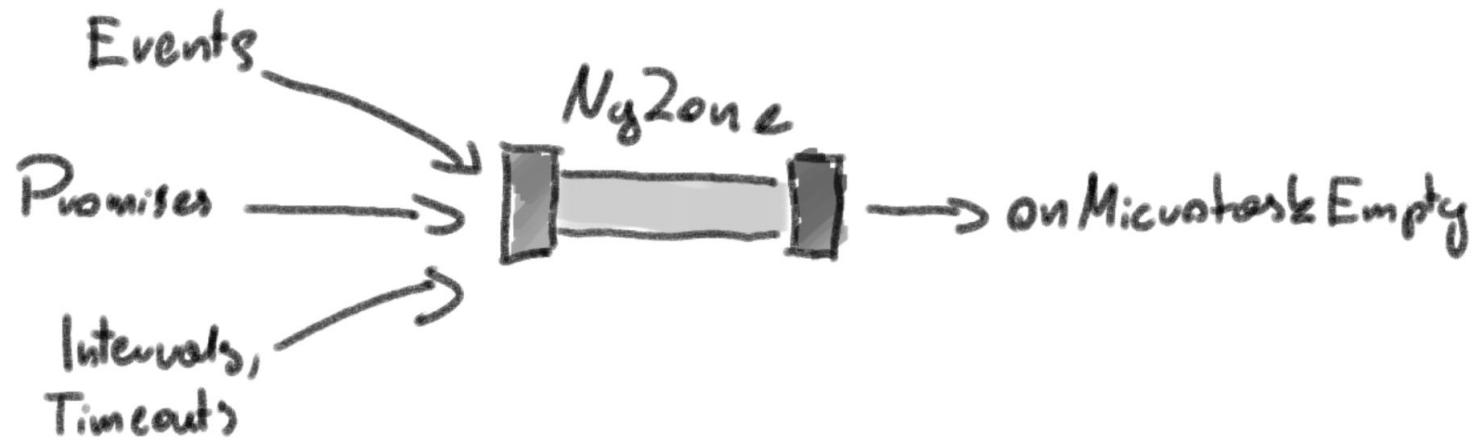
```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';

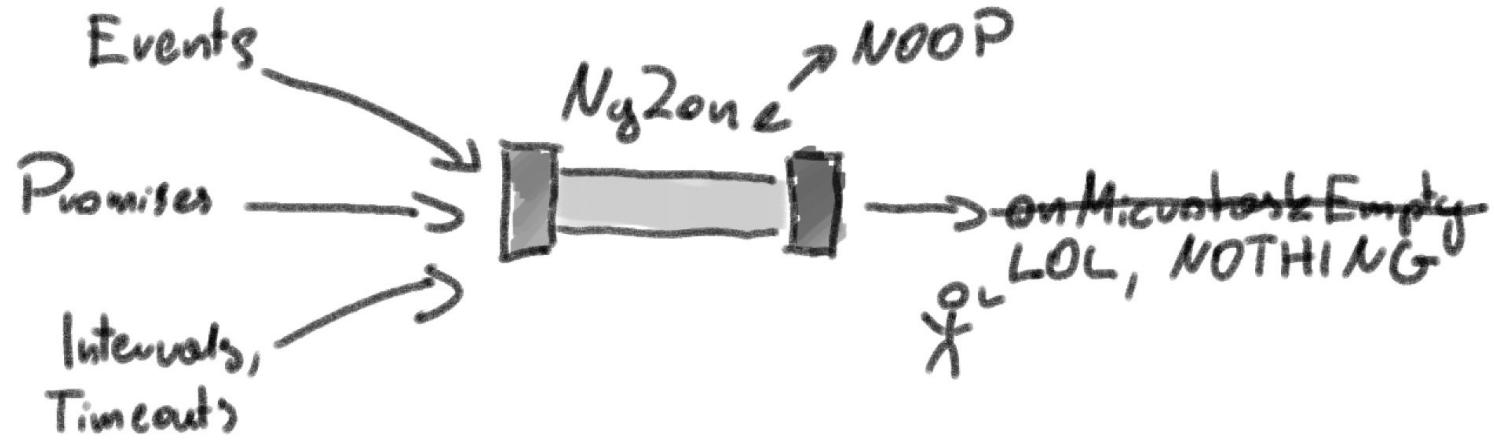
platformBrowserDynamic().bootstrapModule(AppModule, {
  ngZone: 'noop'
}) ...
.catch(err => console.error(err));
```

Angular CLI Server

File status indicators: 11:1 LF UTF-8 2 spaces* TypeScript

noop vs NgZone





The screenshot shows a dual-pane code editor interface with two tabs, both titled "ng_zone.ts".

Left Tab (Line 89 to 119):

```
* @publicApi
*/
export class NgZone {
  readonly hasPendingMacrotasks: boolean = false;
  readonly hasPendingMicrotasks: boolean = false;

  /**
   * Whether there are no outstanding microtasks or macrotasks.
   */
  readonly isStable: boolean = true;

  /**
   * Notifies when code enters Angular Zone. This gets fired first on
   */
  readonly onUnstable: EventEmitter<any> = new EventEmitter(false);

  /**
   * Notifies when there is no more microtasks enqueued in the current
   * This is a hint for Angular to do change detection, which may end
   * For this reason this event can fire multiple times per VM Turn.
   */
  readonly onMicrotaskEmpty: EventEmitter<any> = new EventEmitter(false);

  /**
   * Notifies when the last 'onMicrotaskEmpty' has run and there are
   * implies we are about to relinquish VM turn.
   * This event gets called just once.
   */
  readonly onStable: EventEmitter<any> = new EventEmitter(false);

  /**
   * forkInnerZoneWithAngularBehavior() > delayChangeDetectionForEventsDelegate()
  
```

Right Tab (Line 480 to 508):

```
* Provides a noop implementation of NgZone which does nothing.
* to framework to perform rendering.
*/
export class NoopNgZone implements NgZone {
  readonly hasPendingMicrotasks: boolean = false;
  readonly hasPendingMacrotasks: boolean = false;
  readonly isStable: boolean = true;
  readonly onUnstable: EventEmitter<any> = new EventEmitter();
  readonly onMicrotaskEmpty: EventEmitter<any> = new EventEmitter();
  readonly onStable: EventEmitter<any> = new EventEmitter();
  readonly onError: EventEmitter<any> = new EventEmitter();

  run<T>(fn: (...args: any[]) => T, applyThis?: any, applyArgs?: any)
    return fn.apply(applyThis, applyArgs);
}

runGuarded<T>(fn: (...args: any[]) => any, applyThis?: any, applyArgs?: any)
  return fn.apply(applyThis, applyArgs);
}

runOutsideAngular<T>(fn: (...args: any[]) => T): T {
  return fn();
}

runTask<T>(fn: (...args: any[]) => T, applyThis?: any, applyArgs?: any)
  return fn.apply(applyThis, applyArgs);
}

forkInnerZoneWithAngularBehavior() > delayChangeDetectionForEventsDelegate()
```

The status bar at the bottom indicates the file is 398:22, LF, UTF-8, 2 spaces*, TypeScript 5.0.0.

no operations

rxjs/dist/esm/internal

...

zone.js

Stat size: **119.62 KB**

Parsed size: **119.62 KB**

Gzipped size: **23.96 KB**

Path: **./node_modules/zone.js/fesm2015/zone.js**

pipe.js noop.js

operators

share.js

mergeInternals.js

zone.js

Runtime vs Bundle Optimizations

load. Users have different performance expectations for each of these contexts, so performance goals are defined based on the context and [UX research on how users perceive delays](#).



The 4 parts of the RAIL performance model

Focus on the user

Make users the focal point of your performance effort. The table below describes key metrics of how users perceive performance delays:

User perception of performance delays

0 to 16 ms	Users are exceptionally good at tracking motion, and they dislike it when animations aren't smooth. They perceive animations as smooth so long as 60 new frames are rendered every second. That's 16 ms per frame, including the time it takes for the browser to paint the new frame to the screen, leaving an app about 10 ms to produce a frame.
0 to 100 ms	Respond to user actions within this time window and users feel like the result is immediate. Any longer, and the connection between action and reaction is broken.
100 to 1000 ms	Within this window, things feel part of a natural and continuous progression of tasks. For most users on the web, loading pages or changing views represents a task.
1000 ms or more	Beyond 1000 milliseconds (1 second), users lose focus on the task they are performing.
10000 ms or more	Beyond 10000 milliseconds (10 seconds), users are frustrated and are likely to abandon tasks. They may or may not come back later.

Users perceive performance delays differently, depending on network conditions

On this page

Focus on the user

Goals and guidelines

Response: process events in under 50ms

50 ms or 100 ms?

Animation: produce a frame in 10 ms

Idle: maximize idle time

Load: deliver content and become interactive in under 5 seconds

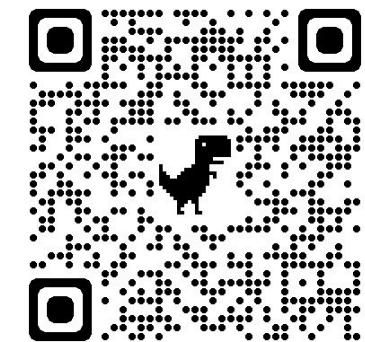
Tools for measuring RAIL

Chrome DevTools

Lighthouse

WebPageTest

Summary



Dependency Injection

A screenshot of a dark-themed code editor, likely Visual Studio Code, displaying a TypeScript file named `example.service.ts`. The file contains the following code:

```
import { Injectable } from '@angular/core';
@Injectable({
  providedIn: "root"
})
export class ExampleService {
  doStuff(): void {
    // code...
  }
}
```

The editor interface includes a title bar with project and workspace dropdowns, a tab bar with an Angular CLI Server entry, and a status bar at the bottom showing the file path, encoding, and other details.

Providing vs Injecting

Null



Platform



Application



Lazy Route



Modules



Directives

Null



Platform



Application



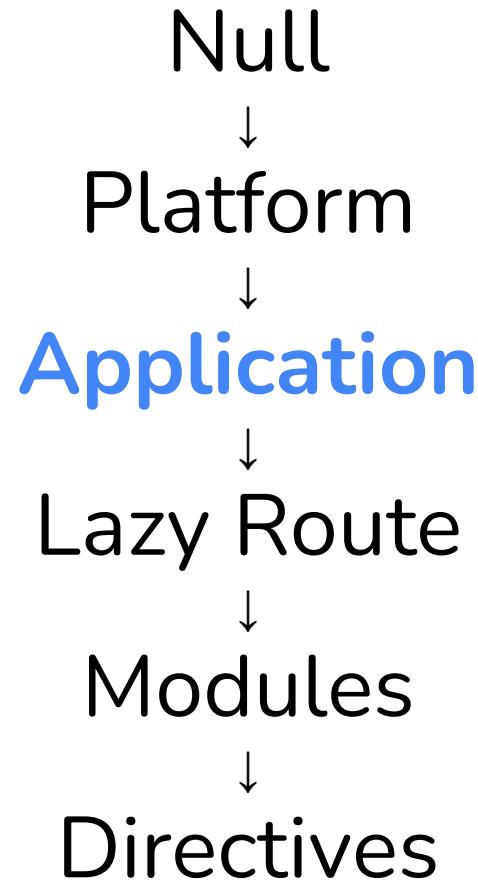
Lazy Route

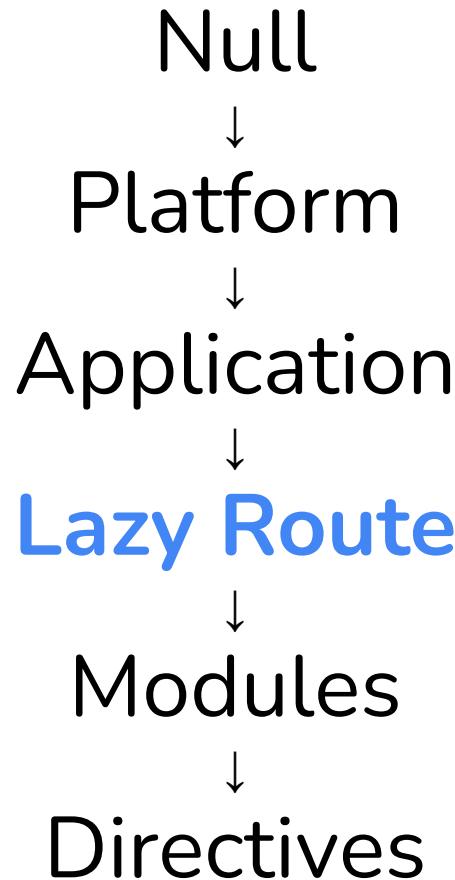


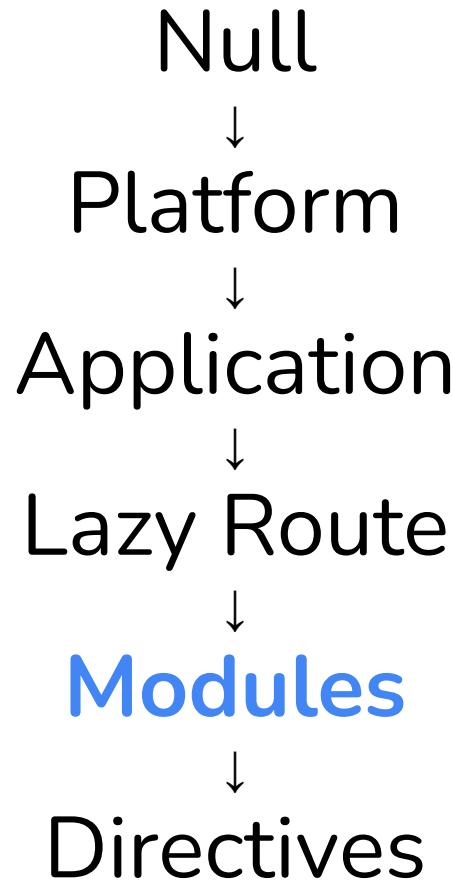
Modules

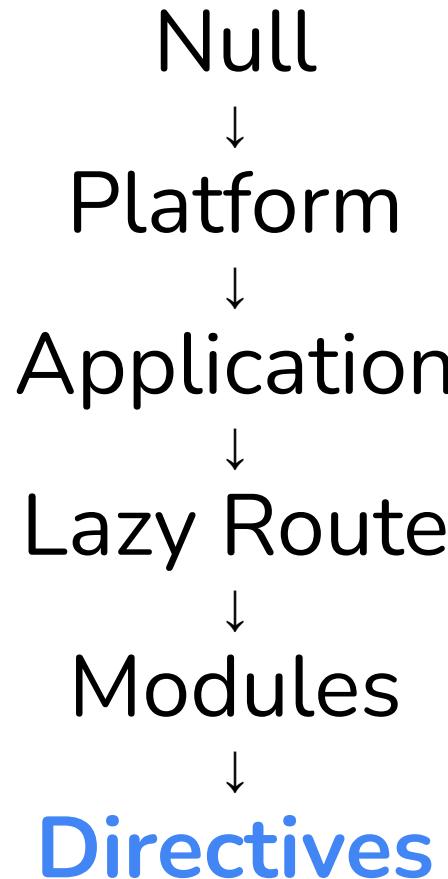


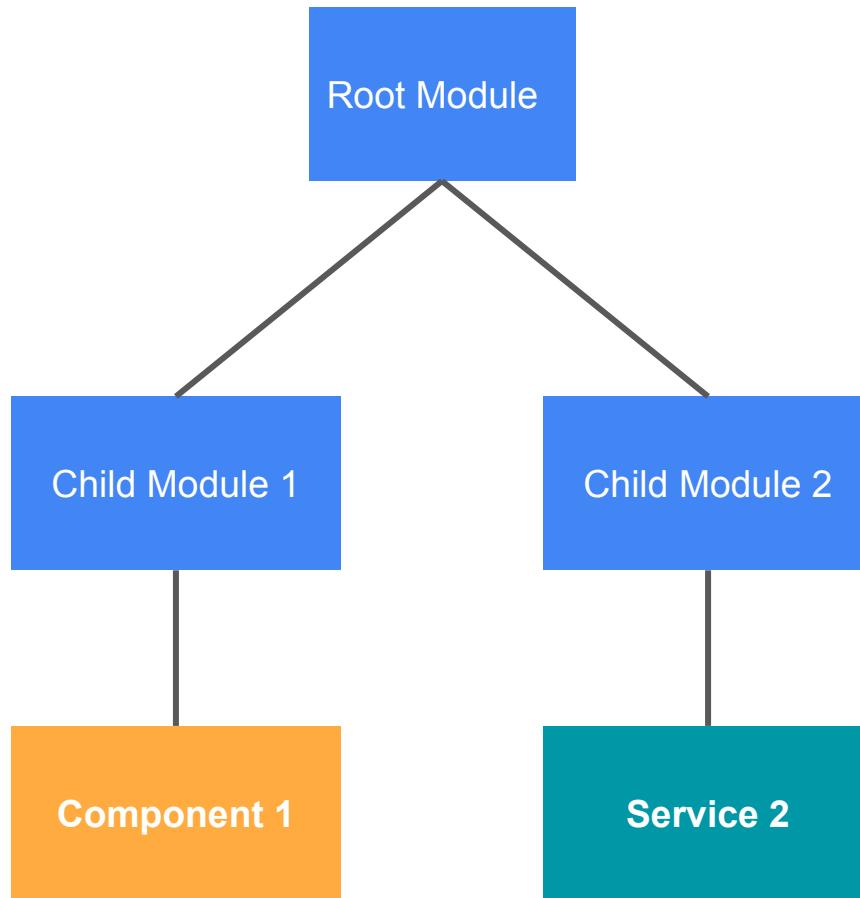
Directives

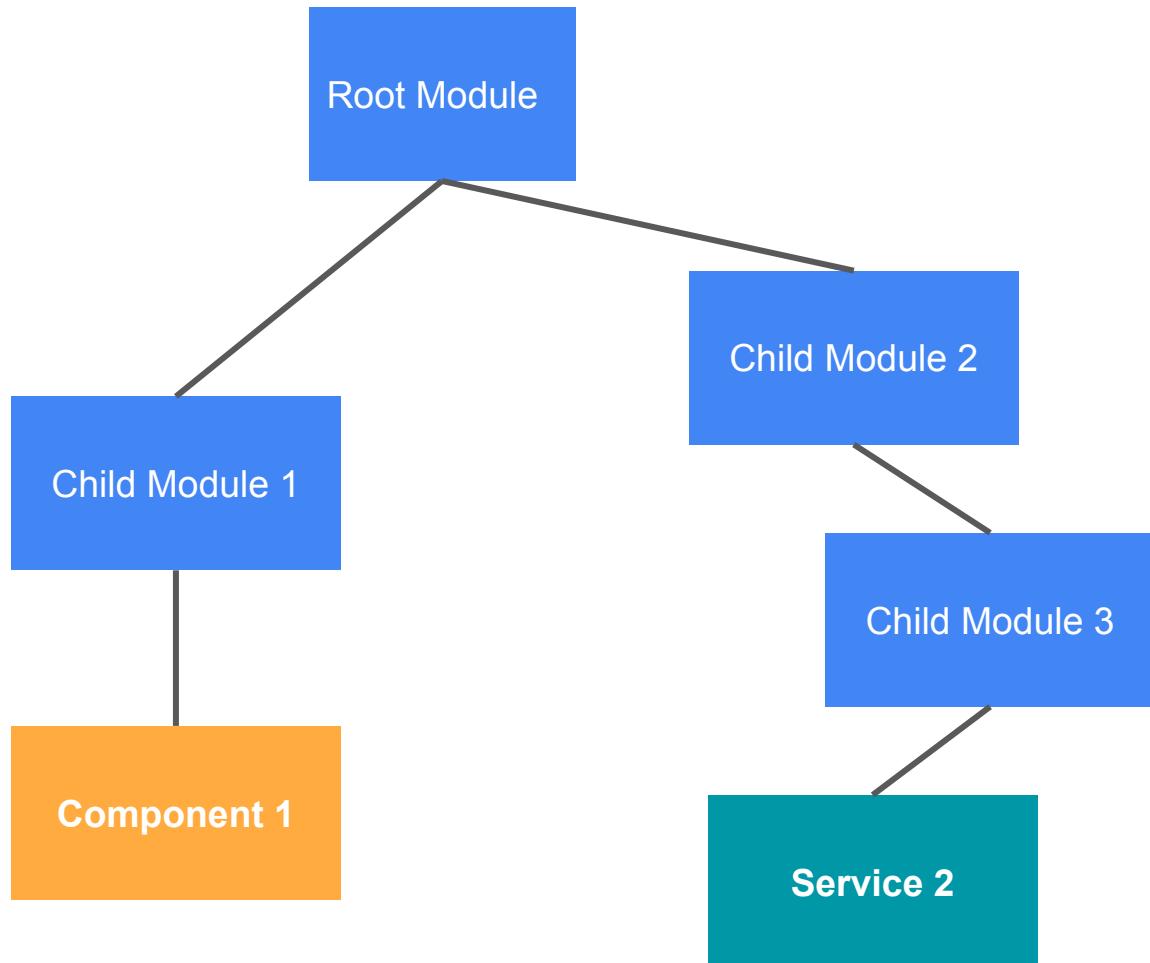












inject()



Search Medium

Write



galczo5

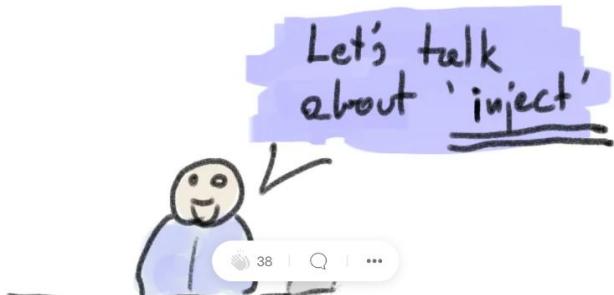
Mar 7 · 8 min read · Listen



I'm preparing myself for the lightning talk, so here you have 5 cool usages of inject function in Angular

Spring conferences are coming. It's the perfect time to prepare a lightning talk subject. In this article I will describe the code from one of my subjects: 5 cool usages of `inject()` function from Angular.

Code, unit tests, and a simple app for manual testing you can find in the repository: github.com/galczo5/ng-inject-lightning-talk.



galczo5

36 Followers

[Edit profile](#)

More from Medium

Piyali Das

[Angular Theme Integration using Dynamically Load CSS](#)



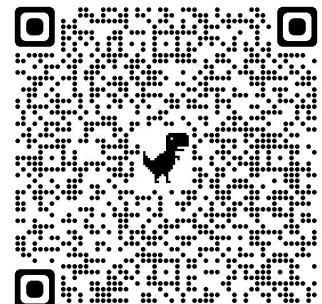
Kellycochane

[Why you should stop using “traditional” margin and padding when styling with...](#)



cengkuru michael in Bootcamp

[Integrating ChatGPT API into Angular: A 3-Step Guide for Conversational AI...](#)



Guillaume Ferber

[Maximizing Performance with Angular: Tips and Tricks for Efficient Code](#)

[Help](#) [Status](#) [Writers](#) [Blog](#) [Careers](#) [Privacy](#) [Term](#)
[Text to speech](#)

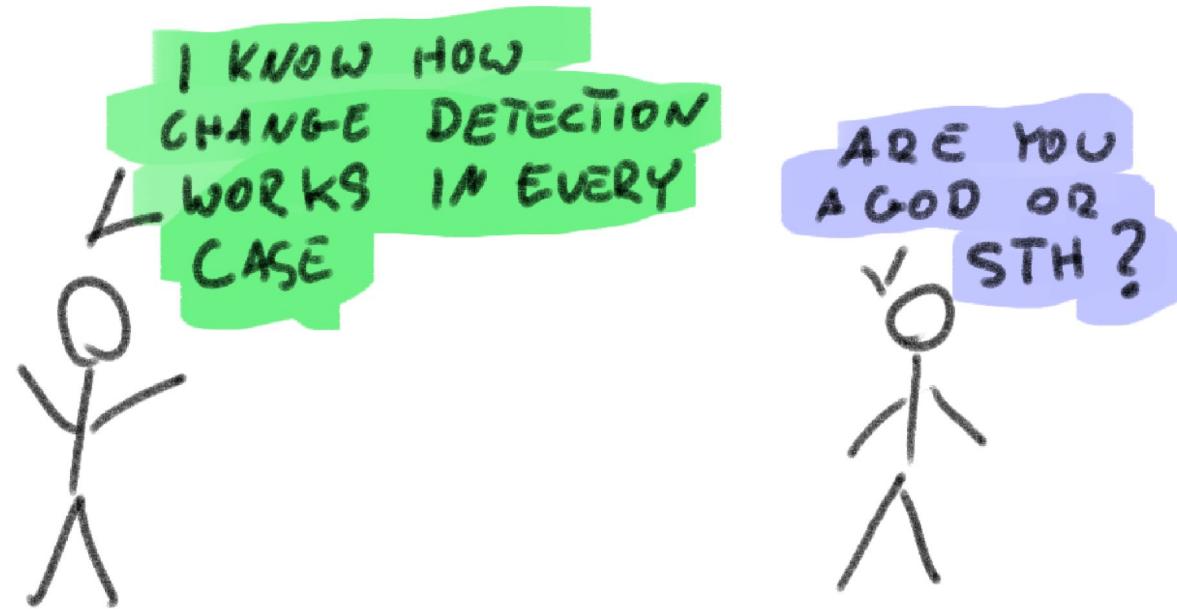
```
// Example 1
// Template: Table of cat facts
@Component({ /* ... */ })
export class Component {
    catFacts$ = useHttpResource('GET', 'https://catfact.ninja/facts');
}

// Example 2
// Template: Table of cat facts
@Component({ /* ... */ })
export class Component {

    @Input()
    id!: string;

    catFacts$ = useOnInit(this, "id")
        .pipe(
            useCatFacts(), // Pass id to useCatFacts: https://catfact.ninja/fact
            useMessageErrorHandler() // Create dynamically component that will d
        );
}
```

Change Detection



Reduce number of change detection cycles



Learn Find Job Stay connected Contribute

ARTICLES



Angular

React

JavaScript

Angular in Depth course series the deep dive you've been waiting for

[Enroll now](#)

These 5 articles will make you an Angular Change Detection expert

This article is a list of articles related to change detection that you need to read to become an expert in change detection. It also offers an extra list of articles aimed at dispelling common confusions.

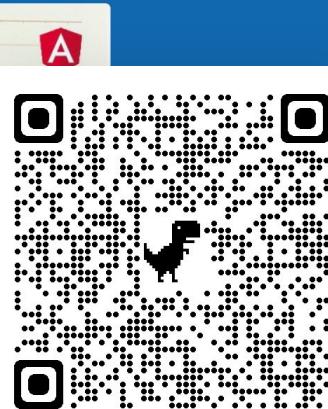
Angular

Change-Detection

2 October 2017

5 min read

1 comment



“Doing less stuff is faster
than doing more stuff”

Renderer2

The screenshot shows a code editor interface with the Angular project structure open. The left sidebar displays the 'Structure' of the `Renderer2` class, which includes methods like `createElement`, `createText`, `destroyNode`, and various styling and attribute manipulation methods. The right panel shows the `api.ts` file, which contains the implementation for these methods. The code uses JSDoc-style comments to describe each method's parameters and behavior.

```
194 /**
195 * Implement this callback to remove the value from the element.
196 * @param el The element.
197 * @param style The name of the style.
198 * @param flags Flags for style variations to remove.
199 */
200 abstract removeStyle(el: any, style: string, flags: number);
201 /**
202 * Implement this callback to set the value of a property on the element.
203 * @param el The element.
204 * @param name The property name.
205 * @param value The new value.
206 */
207 abstract setProperty(el: any, name: string, value: any);
208 /**
209 * Implement this callback to set the value of a node.
210 * @param node The node.
211 * @param value The new value.
212 */
213 abstract setValue(node: any, value: string): void;
214 /**
215 * Listen to an event on the entire window or document.
216 */
217 /**
218 * Implement this callback to start an event listener.
219 * @param target The context in which to listen for events.
220 * @param event The event type.
221 */
222
```

Second one is toggling the background style in half of the elements.

Last button is toggling the background style in only one element from the list.

Do the tests with the dev tools open. There are logs with a precise time measurement for each action.

```
[STYLE.BACKGROUND] = "COLOR"
THIS.RENDERER.SETSTYLE(THIS.ITEM.NATIVEELEMENT, 'BACKGROUND', COLOR);
THIS.ITEM.NATIVEELEMENT.STYLE.BACKGROUND = COLOR;

CHANGE ALL CHANGE HALF
CHANGE ONE
CHANGE ALL CHANGE HALF CHANGE ONE
CHANGE ALL CHANGE HALF
CHANGE ONE
```

LOREM IPSUM DOLOR SIT AMET ENIM.

With my setup, I had results like in the table below.

Directive	Renderer2	Native
Change all	277ms	5155ms
Change half	164ms	2597ms
Change one	54ms	11ms

Conclusion

There is no one the best way to add styles to

Directive	Renderer2	Native
Change all	277ms	5155ms
Change half	164ms	2597ms
Change one	54ms	11ms

Why changing the styles in one element is more efficient with the `Renderer2`? It's easy. There is no overhead of framework checks etc.



Rendevu 2

vs

[style]

BOTH!

Observable Input

ng-inject-lightning-talk main

Angular CLI Server

app.component.ts

```
1 import {Component, Input} from '@angular/core';
2 import {Observable} from "rxjs";
3 ...
4 @Component({
5   selector: 'app-root',
6   template: `
7     <h1>Quick test</h1>
8     <p>
9       <button routerLink="/">Home page</button>
10    </p>
11    <router-outlet></router-outlet>
12  `
13})
14 export class AppComponent {
15
16   @Input()
17   thisMethodNeedChangeDetectionToPassValue!: string;
18
19   @Input()
20   thisMethodDoesNotNeedChangeDetectionToPassValue!: Observable<string>;
21
22}
23
```

File tree:

- src
- app
- app.component.ts

Bottom status bar: 23:1 LF UTF-8 2 spaces* TypeScript

Precise changes in small piece of DOM only
when needed

Signals!

Encapsulation...?

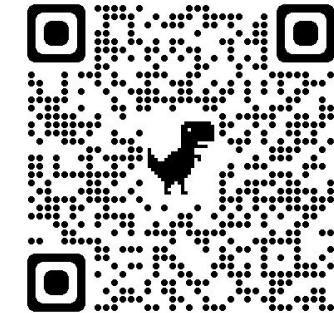
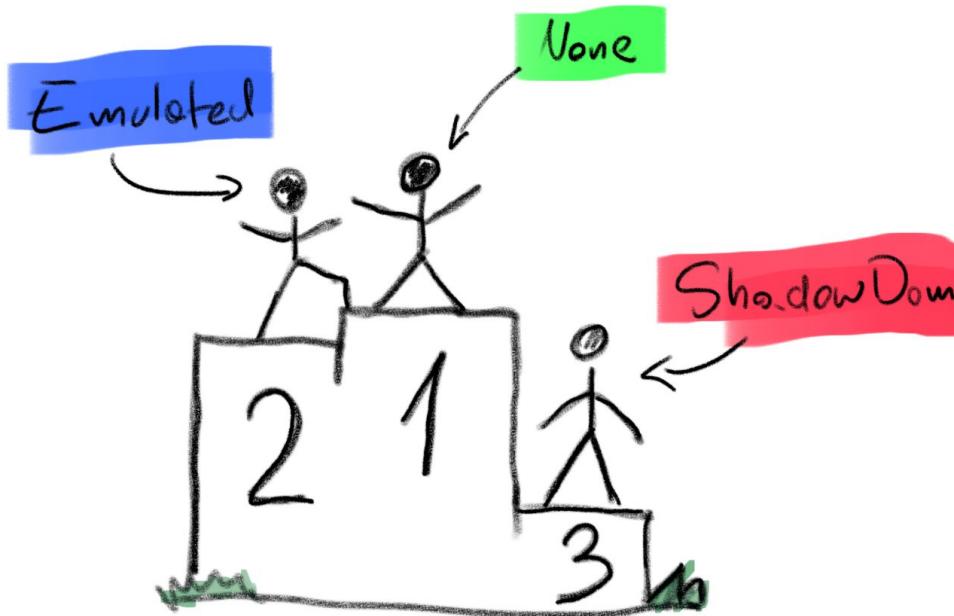
Encapsulation type	Total	Avg iteration time
None	12496ms	124,9ms
Emulated	13027ms	130,2ms
ShadowDom	21352ms	213,5ms

There is almost no difference between the `Emulated` and disabled encapsulation. The difference and the rest of the encapsulation types is huge.

Conclusion

It scares me a little that I wrote an article about encapsulation and its performance. It's just a small part of the whole text. I just wanted to describe differences well :D

Encapsulation type	Total	Avg iteration time
None	12496ms	124,9ms
Emulated	13027ms	130,2ms
ShadowDom	21352ms	213,5ms



Unit tests

What's the problem with Karma?



I CAN'T
GO FASTER
↙ MAYBE ASK JEST

Karma vs Jest

Mythical Angular

[Home](#) | [All posts](#) | [Tags](#) | [GitHub](#)

esbuild against TestBed

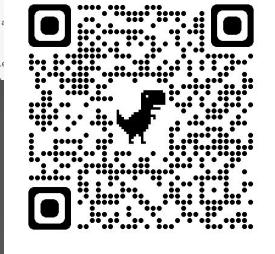
2022-07-24

Isn't the unit testing the most basic technique and one of the more important aspects of programming these days? During my days as a .NET developer, I learned a few things about unit testing:

1. Unit tests have to be **FAST**. If they are not fast, you're going to be irritated, and you'll waste a lot of time.
2. You have to **ISOLATE** your unit tests from the source code. If not, changes in the source code may affect tests and change the requirement behind the test.
3. Tests have to be **REPEATABLE**. No need to explain, random tests are not trustworthy.
4. You shouldn't have to check the code manually. Tests should be **SELF-VALIDATING**. If you have to check it manually even after a unit test, then it's not a good unit test.
5. Tests should be **THOROUGH**.

All of these rules are known and understood.

Unfortunately, I have a problem: so the test code is compiled.



It's pretty obvious and easy to Karma as a test runner, since the browser and

Mythical Angular

[Home](#) | [All posts](#) | [Tags](#) | [GitHub](#)

It looks that TestBed works with esbuild

2022-08-08

In previous post "[esbuild against TestBed](#)" I said that `TestBed` is not working when we use esbuild to bundle tests. It seems like it's possible to work around the problems.

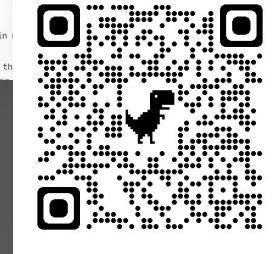
I said also that I'm not sure what to think about the `TestBed`, is it worth using it or not. Every day, I'm closer to getting my own opinion. But this is a subject for the next post.

Today we will focus on how to compile our tests with esbuild and still be able to use `TestBed`.

Repository

As always, code you can find in:

I had to decompose and change the `transformer`, but the compilation



Mythical Angular

[Home](#) | [All posts](#) | [Tags](#) | [GitHub](#)

Short: I had some time, so I checked how time-consuming is TesBed

2022-08-09

This is going to be short. I cannot stop thinking about my last results from the [previous article](#). I was able to use esbuild instead of ts-jest, and it was still slow.

It looked like using TestBed is time-consuming. So I decided to check it, and I'm ready to show you the results.

The test

For testing, I used the same repre-

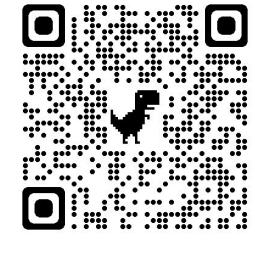
The test is simple. Each `it` con-

`import { TestBed } from '@angular/core'`

tests.

me case, tests are checking

`copy`



In addition to the code, I've created a repository with example unit tests using the `TestBed`. Here it is: [ng-es-build-unit-tests](#).

Test results

Today I will put the results of the tests before the code. There is only one initial condition. Because I don't have a caching mechanism in my code, I decided to disable cache in Jest with `ts-jest`. Thanks to that, tests are comparable.

I have only 16 tests, all of them are using `TestBed` to create a component.

No. ts-jest esbuild no compilation (working on result from esbuild)

1	15.86s	13.90s	10.45s
2	16.07s	11.09s	10.48s
3	15.98s	11.79s	13.57s

As you can see, results are interesting. It's obvious that esbuild is faster than ts-jest. The problem is that, even if I skip the compilation, the tests are slow.

You may say, that 10s or 12s is not that long. Let's take a look at the numbers. In my case, one test took me 0.75s.

So, for an app that contains 300 components, 300 injectables and 100 directives if you want to have a test for every single object, you will spend about 525s. It's almost 9 minutes!. And having 700 tests, it is not that unusual.

We all heard about testing pyramid. In theory, we should be able to have a huge number of unit tests, because they are fast.

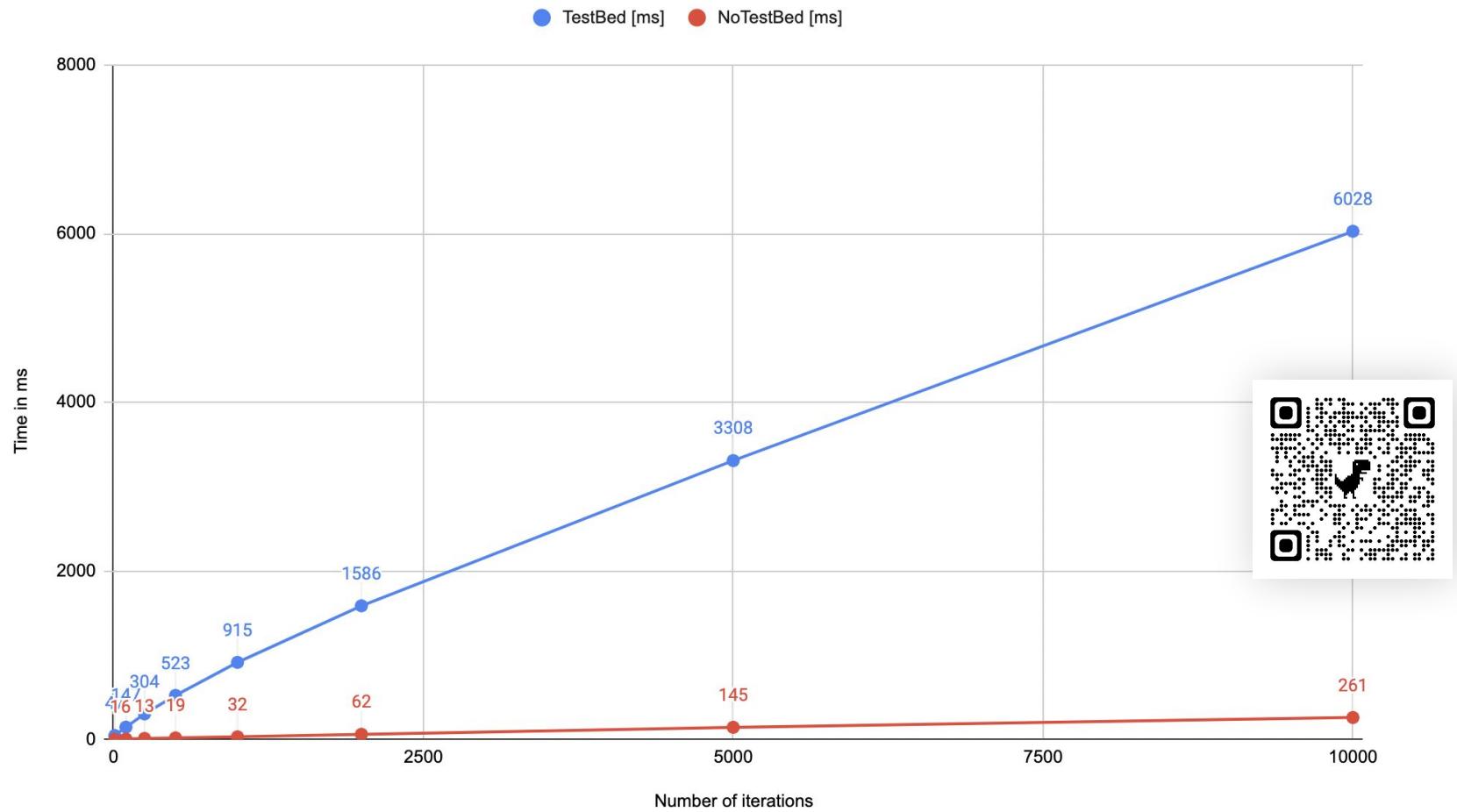
No. ts-jest esbuild no compilation (working on result from esbuild)

1	15.86s	13.90s	10.45s
2	16.07s	11.09s	10.48s
3	15.98s	11.79s	13.57s



The code

After the previous section, you know that when you use `TestBed` there is not a lot of the difference between the `ts-jest` and `esbuild`. Anyway, I want to show you how I implemented it, despite there is no sense of using it.





galczo5

Mar 18 · 13 min read · Listen



Mythical Angular — Chat GPT can write tests without TestBed, you should too



Hi GPT! Are you ready for some frontend dev coding?



Hello! While I am not capable of writing code directly, I can certainly provide guidance and suggestions on frontend development. What do you need help with?



Do you know the Angular framework?



Yes, I am familiar with the Angular framework. Angular is a popular JavaScript framework for building web applications, and it is often used for building complex single-page applications. It was developed by Google and is known for its robustness, modularity, and scalability. What do you need help with in Angular?

Me: Hi GPT! Are you ready for some frontend dev coding?

Chat GPT: Hello! While I am not capable of writing code directly, I can certainly provide guidance and suggestions on frontend development. What do you need help with?



galczo5

44 Followers

[Edit profile](#)

More from Medium



FAM in Level Up Coding



[When and why to use ng-container in Angular?](#)



Altimetrik Poland Tech Blog



[How to Develop Angular Libraries Locally?](#)



Preston Lamb in ngnconf



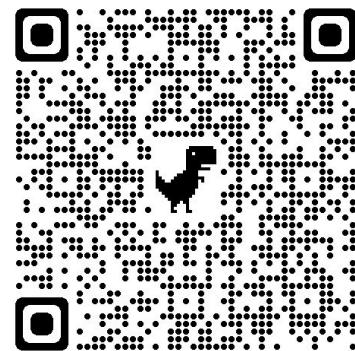
[Using Observables as C](#)



Tomasz Wi... in The Aize



[Creating a loading indicator using RxJs and the withLoading pattern](#)



It all sounds cool,
but I have to warn you!

Top 3 important
aspects of Angular to focus
and projects to follow

Change Detection

NgZones, Working without zone.js, Reducting change detection cycles with Renderer2, ChangeDetectorRef

Dependency Injection

Component/Directive providers, Platform
Providers, Replacing built-in providers

Reactivity

Observable Inputs, rx-angular, Signals



Kamil Gałek

Frontend Developer
BigPicture