The Complete Angular Performance Guide For 2021

Frustrated with your slow Angular app? Here's how to double or even triple the speed of your Angular application.

Daniel Kreider Follow

Jun 14 · 13 min read

. . .

Need to make your Angular app perform faster RIGHT NOW? Then click here to view and download the complete Angular performance checklist.

Otherwise, if you want to dive deep...

...into all the mechanics and logic of how we improve the performance of an Angular application, then read on.

> We've got so much to cover and you're in for a real ride!

Introduction

Imagine doubling or tripling the performance of your angular application.

What if your Angular application loaded twice as fast?

Or once it had loaded, it performed many times faster than it currently does?

We can even take it a step further...

How would you feel if you knew exactly where to go looking to find the performance bottlenecks in your Angular application?

Whenever your boss shows up to tell you that the Angular application is slow you know exactly where to start and the steps to follow to make your

Angular app perform faster.

> Wouldn't that feel amazing?

So...

What can you do to improve the performance of your Angular application? And fix Angular performance issues?

Aside from hiring an Angular consultant to help you, here's the in-depth-guide to improve and optimize your Angular performance and making sure it loads blazing fast. Every. Single. Time.

In this complete Angular performance guide I'm going to show you the step-by-step process to make your Angular app load faster, run faster and perform faster while making your team, your boss and your users elated with its performance.

I'm also giving you an Angular performance checklist that you can download free and use anytime you want. This Angular performance checklist was created to help you tackle your Angular application like a football-tackling-rock-star.

If you want to download the Angular Performance checklist, then click on this link. Otherwise, keep reading and I will give you the step-by-step formula to make your Angular application perform faster than it ever has.

Why are angular apps slow?

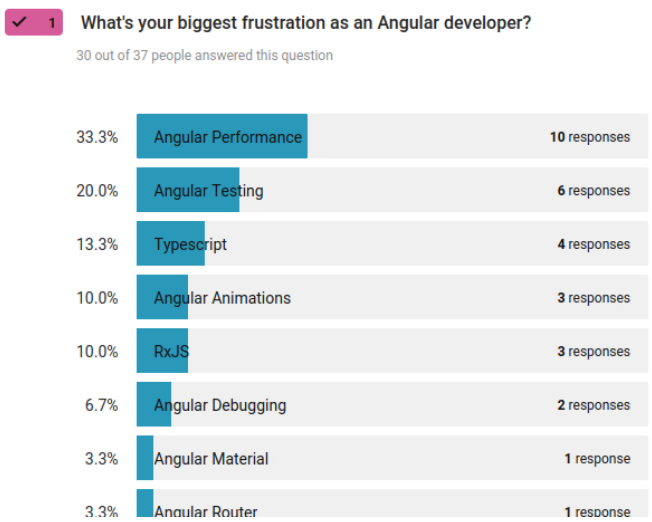
Is it because Angular is a crappy project?

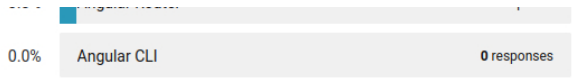
Or because it was poorly engineered by the Angular team from Google?

I recently surveyed a group of angular Developers and asked them what their biggest frustration with Angular is.

Want to guess what answer got the most votes?

Angular Performance.





Many Angular developers are frustrated with Angular because they say it is slow.

Some think that it's hard to build a fast Angular application. Especially given that the bundle sizes of its biggest rivals — React & Vue.js — are usually about half as small and take about half the time to parse and run the JavaScript.

So does this mean that your Angular application is just doomed to perform like a turtle?

Nope. Not if you apply what I'll show you in this article, follow the steps and put in the work.

Yes buddy, you have to learn how to make your Angular application load faster, run faster and perform faster.

So where do you start?

Dicing the Angular performance problem

We need to define a few things before we dive into the code of your Angular application.

First, there are two performance bottle-necks that your Angular application might have.

- **Angular Load Performance**
- **Angular Runtime Performance**

Angular load performance refers to how long it takes for a browser to download your application, load it into the browser, parse the JavaScript and get it displayed to your user.

Once it's loaded, the next type of performance is runtime performance. Is your Angular app snappy and fast as users interact with it? Or does it slow down at times? Is it a memory pig? Or is it light and runs faster even on small mobile devices?

And as you've probably already guessed, the most common Angular performance issue is in the first option — Angular load performance.

But, no Angular application is immune to runtime performance. Runtime performance is usually caused by not properly unsubscribing from a RxJS observable, binding thousands of elements to a list without using a trackbyFn and overloading the change detection cycle.

This Angular performance guide is split into two sections — a section on Angular load performance and a section on Angular runtime performance.

We'll first dive into the mechanics of making your Angular application load faster. And then, last of all, we'll discuss how to make sure it performs fast once it's been loaded.

Getting started: How to measure and profile the performance of your Angular application

The first step in improving the performance of your Angular app is to **measure** how long it takes to load and bootstrap inside the browser.

Why?

Some developers know that they need to optimize their Angular app but they have no idea how much or where exactly to start. And so they make these wild stabs at the code or build configuration and then refresh the app to see if it appears to load faster than it did before. This kind of optimization approach is nothing but a clown show and deserves every kind of criticism it can get. It's like trying to bake a cake without the necessary measuring cups. Or re-arranging the chairs on the deck of the Titanic before it sank.

So what's the smarter approach?

Measure. Measure. Measure.

Somehow, we as developers don't always notice the extra second or two that will bother the users. At least that's been my experience.

For example, one of my developer buddies is a Django developer. He kept insisting that his Django application was faster than my Angular app.

When we were together he'd pull out a browser and show me just how much faster it was...

And how my Angular application was a turtle compared to his Django stuff...

And on and on and on... until...

We started actually counting the milliseconds by using the browser's developer tool. After profiling the performance of my Angular app vs his Django app we discovered that they were averaging about the same load time.

Which one was faster? I don't remember. 😁

Obviously my friends internal timer wasn't properly counting and proves why a developer MUST carefully profile the performance of his Angular app and know EXACTLY how many milliseconds it takes to load the thing. By counting the milliseconds that it takes to load your Angular app you will be able to know exactly how much you're improving load speed and performance as you attempt to optimize. Yup, count the milliseconds.

How to make your Angular app load faster ⚡

1. Split your application into lazy-loaded modules.

Years ago when I was still green behind the ears and hardly knew anything about Angular I created a LARGE angular application and pilled the whole thing into one module — the `app.module.ts` file.

That was a mistake that cost me a lot of performance issues. If you're not familiar with Angular modules, then check out this talk by the Angular comedian Shai Reznik.

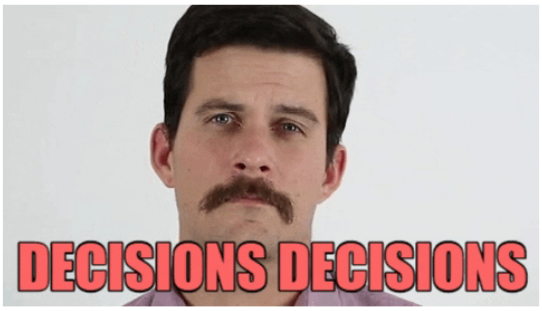
By default, NgModules are eagerly loaded, which means that your Angular application loads ever single NgModules gets loaded, even if they are not immediately necessary. So to make your Angular application load faster, you need to first split the big beast into smaller pieces.

The Angular docs do a great job of explaining how to set up lazy loading modules. You can find all the examples and best practices here.

2. Make sure to use and properly configure your preloading strategies

Once we have our application split into lazy-loaded modules, we need to optimize how those modules are loaded. The Angular framework gives us two basic preloading strategies.

So which one should you use?



Well...

It all depends on who uses your Angular application.

If your users are employees on a high-speed fiber-optic connection, then use a preload-all strategy and load the entire Angular app right away.

But if your Angular application has mobile users on slow 3G connections than a conservative approach is wiser. There are other preloading strategies available for you to install and configure and if you want to know more than read this article where I explain Angular preloading strategies in-depth.

3. Don't over-look the app.module.ts file

Recently I was reviewing the code for a business application built with Angular. I knew the project like the back of my hand. Or at least I thought I did, because I was the main developer and had written 95% of the code for the project.

I opened the app.module.ts file... to take a peak... and I was shocked! 😲

It was busting with imports and libraries that the project had since outgrown.

So what did I do? I yanked those imports out and trimmed the project up, nice and clean, just like a hipster-house-puppy.

Yes buddy, if your Angular app bundle is too large the first place I'd go hunting is in the app.module.ts.

It's a great way to quickly optimize your Angular application. And who knows — you might scare up some weird stuff. 🙈

4. Use an app shell

What is an app shell?

An app shell is a way to rapidly render a portion of your Angular application. It was designed to boost the speed of your Angular application by creating a static rendered page (or a skeleton that is common for all pages) that is sent to the client. The browser then loads the rest of your Angular application and switches to it automatically when finished.

An app shell is designed to show a meaningful first paint allowing your application to appear quickly before it loads and parses the rest of the JavaScript files.

The Angular docs have a great guide about app shells and how to use them here.

5. Use Brotli.

Brotli is the quickest way to boost the load time of any Angular application.

In case you've never heard of Brotli, it's a way to compress your build files and serve them to your users as smaller files. It was initially developed by Google in 2013 and since then has gained a lot of traction. Maybe you've heard of the well known Gzip standard? Brotli was introduced as a successor to Gzip and has gained more popularity recently. On average you can expect JavaScript files compressed with Brotli to be roughly 15% smaller, HTML files are around 20% smaller, and CSS files are around 16% smaller.

To be clear, there is no special relationship between Angular and Brotli. You can use Brotli to compress anything. And it works just as well with other frontend libraries and frameworks like React or Vue.js and more.

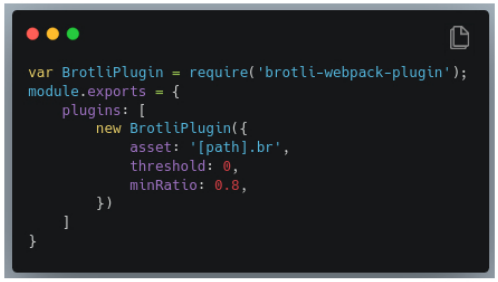
But since you and I are Angular people, here's how to create a custom build step that automatically compresses the production files with Brotli.

The first step is to install the custom-webpack package. It's crucial that you install the version that matches the version of your Angular application so head over to the package page and grab the installation instructions for your specific version of Angular.

The second package you need to install is brotli-webpack-plugin. Here's the command.

```
npm i brotli-webpack-plugin --save-dev
```

And now that we've got our dependencies installed we need to create a `webpack.config.json` in the root directory of our Angular application. It'll look like this.



```
var BrotliPlugin = require('brotli-webpack-plugin');
module.exports = {
    plugins: [
        new BrotliPlugin({
            asset: '[path].br',
            threshold: 0,
            minRatio: 0.8,
        })
    ]
}
```

Our last change will be to the angular.json file. We'll configure Angular to use our custom webpack builder and webpack config file.

```
"architect": {
        "build": {
          "builder": "@angular-builders/custom-
webpack:browser",
          "options": {
            "customWebpackConfig": {
              "path": "./webpack.config.js"
            },
          }
          ...
        }
```

And that's how to use Brotli with Angular. Next time you create a production build, the build files of your Angular application will automatically get compressed with the Brotli algorithm which allows you to serve smaller files to your user's browser.

If you want to learn more than I've got an entire article that explains Brotli and Angular. You can find it here.

How to make your Angular app perform faster

1. Use a trackBy function to improve ngFor performance.

The Angular framework is powered by a library called Zone.js that triggers change detection anytime a DOM event occurs.

Angular also has another nifty feature called ngFor. Just hand it an array of information to render and watch it whiz...

> ...until it SNAPS and POPS and BLOWS UP in your face! 🤯

Because Zone.js triggers a new render every time a DOM event occurs that means your list is being re-rendered when a button is clicked, etc... and of course you never see it because the list's data hasn't changed.

This might be fine if your list never grows. But small lists tend to become large lists, and large lists will generate performance issues unless you're a wise Angular developer. And buddy, that's what I intend to make ya — a shrewd and wise Angular developer that knows how to make your Angular apps perform!

So how do we solve this problem?

We use a trackBy function.

To describe the Angular trackBy function in a nutshell, it is an optional function that can be used with Angular's ngFor. Angular trackBy is used to define how to track changes for an item in a list. How about we dive into some code. What does it take to create our own trackBy function? And avoid expensive re-rendering operations?

Below is a basic example of the Angular trackBy function. The first step is to add a trackBy function to our components Typescript file, like this.

```
trackByItems(index: number, item: Item): number { return item.id; }
```

And then, we'll modify our ngFor to use the new trackBy function.

```
<ul>
    <li *ngFor="let item of items; index as i; trackBy: trackByFn">
        {{ item.value }}
    </li>
</ul>
```

And that is how to use a trackBy function to make your lists perform faster.

2. Tweaking Angular's change detection settings to dodge unnecessary change detection cycles.

As I already said, Angular uses Zone.js to detect events like mouse clicks, keyboard press and so forth. When Zone.js detects an event, it checks the data bindings and updates any stale data bindings that way the data show to the user is always fresh, new and correct. That is change detection in a nutshell. It might be a cool feature but it's got limits... and... if you abuse Angular change detection it'll get cranky.

So how do we trace change detection performance issues?

Enable Angular's Debug Tools

Open the `main.ts` file (used to bootstrap your Angular application) and edit these lines of code...

```
platformBrowserDynamic().bootstrapModule(AppModule) .catch(err =>
console.error(err));
```

...to look like this.

```
platformBrowserDynamic().bootstrapModule(AppModule).then(module =>
enableDebugTools(module.injector.get(ApplicationRef).components[0]))
.catch(err => console.error(err));
```

Profiling Change Detection Time

Now that we've got the debug tools enabled, we can use them to discover how long change detection is taking.

We'll run our Angular application with `ng serve --open`.

As soon as it launches in your browser, open the console in Developer Tools and type the following command to measure how long the last change detection cycle lasted.

```
ng.profiler.timeChangeDetection()
```

For a very basic Angular application you can expect a change detection cycle of 0.01–0.05 milliseconds. And although opinions vary, I would recommend that you never let your change detection cycle go beyond 10ms.

So how do you fix bad change detection performance?

Let's say we have an Angular component that displays a list of 5,000 random numbers.

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

@Component({
  selector: 'app-number-list',
  template: `
    <ul>
        <li *ngFor="let number of numbers">{{ number }}</li>
    </ul>
  `,
  styleUrls: ['./number-list.component.css']
})
export class NumberListComponent implements OnInit {

  numbers: number[] = [];

  constructor() { }

  ngOnInit(): void {
    this.generateNumbers();
  }

  generateNumbers(): void {
    for(let i = 0; i < 5000; i++) {
      let number = Math.random();
      this.numbers.push(number);
    }
  }
}
```

How well do you think it will perform?

Well, when I checked the change detection cycle it was taking at least 12–14ms. 😮

This is a simple example of a bad-performing component. Lists are a bad culprit for long change detection cycles and a good way to fix a long list is

to use a virtual scrolling strategy. The Angular Material CDK has a great virtual scrolling package worth checking into.

3. Monitor the slowness of your HTTP calls.

Maybe a sluggish API server is making your Angular application slow? And if it is, how would you know?

Well, you can use an HTTP interceptor to monitor any slow HTTP calls. Once you've created your HTTP interceptor, use it to monitor how long outbound HTTP requests are taking to complete.

A deeper explanation with lots of examples requires its own article so if you want a complete, step-by-step guide on how to monitor HTTP requests then click here.

Conclusion

And that, my friend, is an explanation of the multitude of ways that you can double or even possibly triple or quadruple the performance of your Angular application.

Angular is a fast evolving framework — which is one of the reasons I enjoy Angular.

It's built by a great group of people that want to know the frustrations Angular developers have. The Angular team has already shed a lot of sweat to improve the performance of Angular applications and we can expect that they'll continue to do so.

Sure, Angular might have a performance quirk, but I still prefer it over some other libraries and frameworks out there.

If you enjoyed this article or found it useful please bang the 👏 button and follow me for more cool articles like this one.

Follow Me: GitHub, Medium, Personal Blog

. . .

Originally published at https://danielk.tech.

Angular | Angular2 | Typescript | JavaScript | Web Development

956 5

WRITTEN BY

Daniel Kreider

https://danielk.tech

Follow

Geek Culture

A new tech publication by Start it up (https://medium.com/swlh).

Follow

More From Medium

React, the Pros and the Cons

Sharad Satsangi in Nerd For Tech

1334. Find the City With the Smallest Number of Neighbors at a Threshold Distance- Step by step…

Noobnoob

How to Create a Service-Oriented React Web-App

Frank Zickert | Quantum Machine Learning in codeburst

Handy JavaScript Tricks —Part 2

John Au-Yeung in JavaScript in Plain English

React By Example: Part 2

John Tucker in Frontend Weekly

Why Should We Be Careful When Using the const JavaScript Keyword?

John Au-Yeung in Level Up Coding

How to Build a Datepicker Component with Vue.js 3?

John Au-Yeung in Dev Genius

Should I use const?

Alex Bostock in Nerd For Tech

Learn more.

Medium is an open platform where 170 million readers come to find insightful and dynamic thinking. Here, expert and undiscovered voices alike dive into the heart of any topic and bring new ideas to the surface. Learn more

Make Medium yours.

Follow the writers, publications, and topics that matter to you, and you'll see them on your homepage and in your inbox. Explore

Write a story on Medium.

If you have a story to tell, knowledge to share, or a perspective to offer — welcome home. It's easy and free to post your thinking on any topic. Start a blog

Medium

About Write Help Legal