**18 Quick Tips for Improving AngularJS Performance**

By Cody Arsenault

Updated on March 7, 2019



[AngularJS](https://angularjs.org/) is an open source JavaScript framework developed and maintained by Google. The tool provides everything you need to create and manage dynamic frontends for web applications. Its modular approach to web design and massive support community make AngularJS a popular tool among professional developers. In fact, AngularJS powers some of the web’s most [high traffic](https://www.keycdn.com/blog/high-traffic) websites including Google and Virgin America. This guide will serve as an introduction to AngularJS and offer tips on how to **improve AngularJS performance**.

**What is AngularJS?**[**#**](https://www.keycdn.com/blog/angular-performance#what-is-angularjs)

AngularJS was created to simplify the complex process of building and managing JavaScript applications. Based on the [Model-View-Controller](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller), or MVC, programming structure, AngularJS is especially useful for creating single page web apps. With a JavaScript library based on standard JS and HTML, AngularJS automatically takes care of things like [DOM](https://www.keycdn.com/blog/blocking-the-dom) manipulation and [AJAX](https://www.keycdn.com/support/ajax-programming) glue that would otherwise have to be coded by developers. The tool provides modular building blocks of JavaScript for developers to mix, match and test. AngularJS can quickly be added to any HTML page with a simple tag.

**The Pros and Cons of AngularJS[#](https://www.keycdn.com/blog/angular-performance" \l "the-pros-and-cons-of-angularjs)**

A few features set apart AngularJS from its competition including:

* Simplified two-way data binding. AngularJS allows you to bind data to HTML using expressions, and AngularJS directives let developers extend their HTML functionality to create new constructs. Things like DOM manipulation and data binding code get condensed into simple elements that can be quickly and easily embedded in HTML templates.
* Since AngularJS was designed to be a highly versatile framework, it can be used to create almost any type of web application. If you’re building a dynamic single page app, there’s likely no better alternative.
* AngularJS is part of the MEAN software bundle, which also includes [MongoDB](https://www.mongodb.com/" \t "_blank), [Express.js](https://www.keycdn.com/support/express-cdn-integration), and [Node.js](https://nodejs.org/en/). Therefore, it allows you to manage both the front and back end of projects using only JavaScript. Alternatively, Ruby on Rails makes a great complimentary back end. ASP.NET and C# also pair well with AngularJS.
* Because AngularJS was built with a **functionality-first mindset**, it’s best suited for a top-down development process. The modular nature of AngularJS makes it easy to divide the labor in large-scale projects among different teams. It also greatly simplifies the testing and debugging process. Because they prioritize using a minimal amount of code, AngularJS applications tend to be compact and easy to edit.

Nonetheless, there are some things that you should take into consideration when deciding if AngularJS is the right fit for your project.

* First and foremost, AngularJS is considered to be very “opinionated,” which means that it imposes structure on developers. For novice and even expert programmers, this is generally good news. AngularJS was designed to be as user-friendly as possible, so its tools are fairly intuitive. However, developers who crave more flexibility may find themselves having to “work around” the framework.
* For some projects, using AngularJS might be overkill. Lightweight frameworks such as [Backbone.js](http://backbonejs.org/" \t "_blank)may be a better option for static websites. AngularJS also isn’t equipped to handle data-intensive DOM manipulation since it relies on “dirty checking” to manage DOM changes, which means that any alterations to variables trigger a DOM update. Although that’s not an issue for many websites, it could cause applications like GUI editors and video games to lag.
* AngularJS also struggles to support high-traffic photo galleries, which is why Instagram wasn’t built on the framework. You can work around these performance issues, but it might be better to go with an alternative like [React](https://reactjs.org/). Otherwise, AngularJS is capable of supporting forms with high levels of user interaction; after all, it does power Gmail.

**AngularJS Optimization Tips**[**#**](https://www.keycdn.com/blog/angular-performance#angularjs-optimization-tips)

AngularJS has plenty of built-in optimization tools, but performance complaints still plague the framework. If you don’t have the massive infrastructure that Google has, you might need to implement some best practices to improve your AngularJS application’s performance.

Whether you know you’re in need of a performance boost, or if you just want to see if there’s room for improvement, here are some tips for getting your AngularJS apps up to speed:

**1. Keep an eye on your digest cycle**[**#**](https://www.keycdn.com/blog/angular-performance#1-keep-an-eye-on-your-digest-cycle)

The digest cycle of your AngularJS app is a good indicator of its performance. Think of the digest cycle like a loop that checks for changes to variables being monitored. The shorter the digest cycle, the faster your application will run.

**2. Limit your watchers**[**#**](https://www.keycdn.com/blog/angular-performance#2-limit-your-watchers)

Speaking of which, any time you introduce data-bindings, you create more $$watchers and $scopes, which prolongs the digest cycle. Too many $$watchers can cause lag, so limit their use as much as possible.

**3. Use one-time binding, if possible**[**#**](https://www.keycdn.com/blog/angular-performance#3-use-one-time-binding-if-possible)

If you’re using an older version of AngularJS, you may be able to take advantage of one-time binding. To do so, just add a double-colon before the value. If applied correctly, the value will resolve once and then disappear from the watchers list. It’s important to note that the one-time binding feature, which was introduced in AngularJS 1.3, is **not available in Angular 4.0**.

**4. Use scope.$evalAsync[#](https://www.keycdn.com/blog/angular-performance" \l "4-use-scope-evalasync)**

If you try to manually activate the digest cycle while it’s already running, you could get an error. To prevent this from happening, use scope.$evalAsync instead of $apply when you need to initiate the digest cycle. It queues up operations to be executed at the end of the current cycle without setting off a new one.

**5. Use Chrome DevTools like CPU Profiler and Timeline**[**#**](https://www.keycdn.com/blog/angular-performance#5-use-chrome-devtools-like-cpu-profiler-and-timeline)

Both the DevTools Profiler and the Timeline tool can help you find performance bottlenecks to guide your optimization efforts. Read our in-depth guide on [Chrome DevTools](https://www.keycdn.com/blog/chrome-devtools).

**6. Limit DOM access**[**#**](https://www.keycdn.com/blog/angular-performance#6-limit-dom-access)

Accessing the DOM can get expensive, so keep your DOM trees small. Don’t modify the DOM if you can help it, and don’t set any inline styles to avoid JavaScript reflow.

**7. Disable CSS class and comment directives**[**#**](https://www.keycdn.com/blog/angular-performance#7-disable-css-class-and-comment-directives)

When creating a directive, you can designate it to be used as an element, attribute, CSS class or comments. If you don’t need CSS class and comment directives, disable them for a performance boost.

**8. Disable debug data**[**#**](https://www.keycdn.com/blog/angular-performance#8-disable-debug-data)

Tools like [Batarang](https://github.com/angular/batarang" \t "_blank) and [Protractor](http://www.protractortest.org/#/) rely on the data about binding and scopes that AngularJS attaches to DOM elements. Therefore, when you’re done debugging, disable the extra data so that it doesn’t drag your application down.

**9. Use Lodash[#](https://www.keycdn.com/blog/angular-performance" \l "9-use-lodash)**

[Lodash](https://lodash.com/) lets you quickly rewrite your application’s logic to improve upon the built-in AngularJS methods and enhance your application’s performance. If your web app doesn’t use Lodash, you can rewrite the methods yourself using native JavaScript.

**10. Use ng-if or ng-switch instead of ng-show**[**#**](https://www.keycdn.com/blog/angular-performance#10-use-ng-if-or-ng-switch-instead-of-ng-show)

The directive ng-show simply toggles the CSS display on or off for a specified element.

To remove an element from the DOM, you must use ng-if or ng-switch.

**11. Avoid ng-repeat when possible**[**#**](https://www.keycdn.com/blog/angular-performance#11-avoid-ng-repeat-when-possible)

Overuse of the ng-repeat directive can drastically drive down performance. Fortunately, there are alternatives. For instance, rather than employing ng-repeat to render a global navigation, you could make your own by using the $interpolate provider to render your template against an object before converting it into a DOM node.

**12. Use $watchCollection[#](https://www.keycdn.com/blog/angular-performance" \l "12-use-watchcollection)**

When you’re using $watch, two parameters are great, but three’s a crowd. Adding a third parameter forces AngularJS to run deep checking, which eats up a lot of resources. The developers were nice enough to include a work around: $watchCollection. It behaves as a third parameter for $watch, yet it just checks the first layer of each object’s properties, so it doesn’t slow things down as much.

**13. Use $cacheFactory[#](https://www.keycdn.com/blog/angular-performance" \l "13-use-cachefactory)**

If you need to store data that you might need to recalculate later, use the $cacheFactory directive. It works like any other memoization method.

**14. Use console.time[#](https://www.keycdn.com/blog/angular-performance" \l "14-use-console-time)**

If you’re having problems debugging, console.time (Chrome DevTools) is an excellent tool for measuring execution times and other performance benchmarks.

**15. Debounce ng-model**[**#**](https://www.keycdn.com/blog/angular-performance#15-debounce-ng-model)

Debouncing inputs using the ng-model directive can limit the digest cycle. For example, applying ng-model-options="{debounce:200}" ensures that the digest cycle doesn’t run more than once every 200ms.

**16. Use $filter**[**#**](https://www.keycdn.com/blog/angular-performance#16-use-filter)

AngularJS runs DOM filters twice during each digest cycle: first to detect changes, and then to update values that have changed. To save some time, the $filter provider allows you to preprocess data before it gets sent to the View and thus skips the DOM parsing process.

**17. Tight Scoping**[**#**](https://www.keycdn.com/blog/angular-performance#17-tight-scoping)

Keep your variables scoped tightly so that the JavaScript garbage collector can free up some memory every now and then.

**18. Pagination or infinite scroll**[**#**](https://www.keycdn.com/blog/angular-performance#18-pagination-or-infinite-scroll)

If all else fails, you can lower the number of elements that get looped over by implementing pagination or infinite scroll. AngularJS even has a directive called ngInfiniteScroll for that purpose.

It’s always more efficient to employ best practices from the beginning rather than to keep going back and making changes. Before you start coding, think carefully about how you can limit bindings, watchers and expensive directives like ng-repeat. Consult the official [AngularJS docs](https://docs.angularjs.org/guide" \t "_blank) for troubleshooting and additional help getting started.

**AngularJS Performance & Testing Tools**[**#**](https://www.keycdn.com/blog/angular-performance#angularjs-performance-testing-tools)

You can find a number of tools dedicated to improving the testing and performance of AngularJS apps. Here are a few of the best options:

**1. Protractor**[**#**](https://www.keycdn.com/blog/angular-performance#1-protractor)

[Protractor](http://www.protractortest.org/#/) comes straight from the Angular team. This software bundle allows you to run automated end-to-end testing with ease. Because Protractor is built on top of webDriverJS and Selenium server, it boasts all of their features, which means you can use the Selenium grid feature to simultaneously run tests in multiple browsers. You can write your own test suites using Jasmine or Mocha.

**2. WebDriverIO[#](https://www.keycdn.com/blog/angular-performance" \l "2-webdriverio)**

As an implementation of W3C webDriver API, [WebDriverIO](http://webdriver.io/guide/plugins/webdriverjs-angular.html" \t "_blank) is more flexible than WebDriverJS. Its command line interface makes setting up tests easy enough for non-programmers to figure out. WebDriverIO users benefit from excellent support and an active developer community.

**3. NightwatchJS[#](https://www.keycdn.com/blog/angular-performance" \l "3-nightwatchjs)**

[NightwatchJS](http://nightwatchjs.org/) is also a custom implementation of W3C webdriver API. Easy to extend and customize, this tool comes with its own testing framework and assertions mechanisms, yet it lacks the level of support that WebDriverIO and Protractor have.

**4. TestingWhiz[#](https://www.keycdn.com/blog/angular-performance" \l "4-testingwhiz)**

Dynamic commands allow [TestingWhiz](https://www.testing-whiz.com/how-to-perform-test-automation-of-angularjs-application" \t "_blank) to sync with varying server wait times to provide accurate end-to-end testing of Angular apps. The codeless scripting feature makes TestingWhiz highly popular among non-programmers.

**5. Batarang[#](https://www.keycdn.com/blog/angular-performance" \l "5-batarang)**

The [Batarang](https://chrome.google.com/webstore/detail/angularjs-batarang/ighdmehidhipcmcojjgiloacoafjmpfk" \t "_blank) tool is a Chrome Extension created by the Angular team to make debugging easier. It boasts several handy features, but its most useful for keeping track of performance benchmarks.

**Improving AngularJS Performance - An Ongoing Process**[**#**](https://www.keycdn.com/blog/angular-performance#improving-angularjs-performance-an-ongoing-process)

As a final note, also considering using a CDN to speed up your AngularJS assets. This will optimize the load time for visitors around the globe thanks to smart caching.

Although there are dozens of JavaScript frameworks these days, AngularJS remains a favorite for many, so it won’t be going anywhere anytime soon. To get the most out of your applications, you should make AngularJS optimization a regular habit. Fine-tuning your application’s performance might allow you to provide more content while using less code, which frees up resources that could be better spent elsewhere.

# 9 Ways to Improve AngularJS Performance

### To get maximum benefits from new and updated JavaScript frameworks, you must keep optimizing performance on regular basis.

[**[](https://dzone.com/users/2947492/pjain-08.html)**](https://dzone.com/users/2947492/pjain-08.html)**by**

[**Samaira Sandberg**](https://dzone.com/users/2947492/pjain-08.html)

**·**

**Jun. 15, 17 ·**[**Performance Zone**](https://dzone.com/apm-tools-performance-monitoring-optimization)**· Opinion**

**Like** [**(15)**](https://dzone.com/articles/9-ways-to-improve-angularjs-performance)

**Comment (3)**

**Save**

[**Tweet**](https://dzone.com/articles/9-ways-to-improve-angularjs-performance)

**34.09k Views**

Join the DZone community and get the full member experience.

[**JOIN FOR FREE**](https://dzone.com/articles/9-ways-to-improve-angularjs-performance)

AngularJS is the most widely used web app framework nowadays, and its popularity graph is constantly rising after the arrival of its much-awaited version AngularJS 4.0. But we can see that almost every Angular expert still struggles with various issues that directly correlate to AngularJS performance, even though it has a lot of its own optimizations.

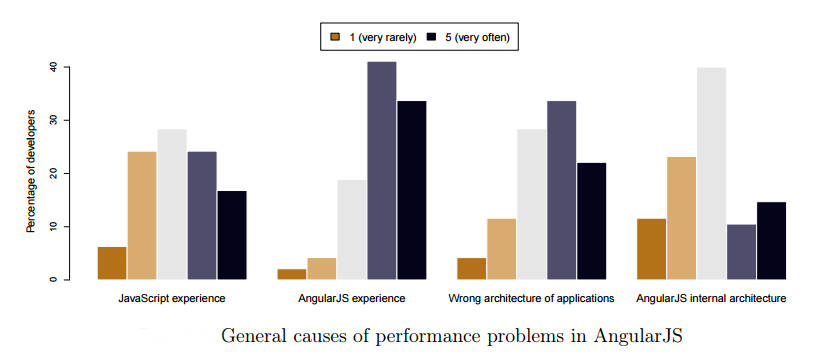
Today, online businesses are greatly affected by the performance of web technologies that they use for their respective projects. Hence, it becomes necessary to dig into all of the factors that are hurting their business growth.

But there might be cases in which the incorrect use of AngularJS methods restricts your application's stance in the market, so AngularJS performance optimization becomes an important need of every [AngularJS development](https://www.valuecoders.com/hire-developers/hire-angularjs-developers?utm_source=Dzone%2FPrakhar&utm_medium=Angularjs%20Performance" \t "_blank) expert. That’s why we're listing nine ways to improve AngularJS performance in this blog.

**Related Reading:**[**Top 10 Advantages of Using AngularJS**](https://www.valuecoders.com/blog/technology-and-apps/advantages-of-angular-js-app-development/)

Recently, a survey was conducted by Brazilian computer scientists who found interesting facts about what causes performance problems for AngularJS programmers. There is not much information about how AngularJS performs in the real world. But research based on surveys provide some pieces of evidence for it.

The survey came up with the following results:

[](https://www.markupbox.com/blog/wp-content/uploads/2017/06/angular1.jpg)

* 45% voted that the source code problem that affects performance.
* Only 8% admitted to actually making changes.
* Some respondents blamed the architecture of AngularJS.
* Some of them blamed unnecessary two-way binding.

After so much discussion on AngularJS performance, now it’s time to see the nine ways through which you can improve AngularJS performance.

AngularJS performance can be easily measured by its digest cycle. The digest cycle can be taken as a loop. During this cycle, Angular checks if there is any change to all the variables watched by all $scopes. If $scope.myVar is defined in a controller and marked for watching, then Angular will monitor the changes on myVar in each loop iteration.

## 1. Batarang Tool to Benchmark Watchers

Batarang is a great dev tool from the Angular team that lowers your debugging efforts. Although it has many new features, some of them help you profile and track the performance of your AngularJS performance. Moreover, the watch tree determines which scopes are not destroyed as it seems to be if there is any increase in the memory.

## 2. Use of Native JavaScript or Lodash

Lodash boosts your application performance by simply rewriting some of the basic logic instead of relying on inbuilt AngularJS methods. If Lodash is not included in your application, then you would probably need to re-write everything in native JavaScript.

## 3. Chrome Dev Tool Profiler to Identify Performance Bottlenecks

This one is a handy tool that gives you the option to select which profile type you want to create. Record Allocation Timeline, Take Heap Snapshot, and Record Allocation Profile are used for memory profiling. After this performance optimization, your app will fully render in less than two seconds and users can freely interact with it then.

## 4. Minimize Watchers

AngularJS completely revolves around its digest cycle. Whenever a digest cycle is triggered, it loops over every binding to detect model changes. The amount of time taken in each digest cycle can be reduced by reducing the number of watchers. It also reduces application memory footprints.

## 5. ng-if Is Better Than ng-show

The ng-show directive toggles the CSS display property on a particular element, while the ng-if directive actually removes the element from DOM and re-creates it if needed. Further, the ng-switch directive is an alternative to the ng-if with the same performance benefits.

## 6. Don't Use ng-repeat

This is the biggest win for any app if it is not using the ng-repeat directive, so it is advisable to avoid ng-repeat and build the HTML using JavaScript. For vocalizing applications, the use of ng-if results in the addition of unnecessary watchers. The use of the ng-bind-html directive is a better solution to get rid of this problem.

## 7. Use $watchCollection (Includes Third Parameter)

Using $watch with two parameters is good — but using $watch with three parameters, i.e. $watch(‘value’,function(){},true), makes Angular perform deep checking (to check every property of the object). But this could be expensive. Hence, to resolve such a performance issue, Angular provides $watchCollection(‘value’, function(){}) which acts almost alike $watch with 3rd parameter except it only checks first layer of object’s properties at low cost.

## 8. Use console.time for Debugging Issues

If your application is struggling with debugging issues and effecting Angular performance, then use console.time, which is a great API.

## 9. Debounce ng-model

You can debounce the input with ng-model. For example, to de-bounce the search input like GOOGLE, you have to use ng-model-options=”{debounce:250}”. This makes the digest cycle get triggered no more than once per 250ms due to the changes in this input model.

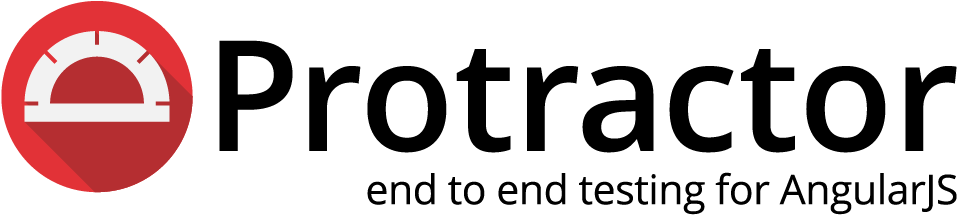
Development time is at a premium nowadays, so you need a comprehensive framework such as AngularJS to run your business quickly out of the box.

After researching a lot, we have collected some other important tools to enhance AngularJS performance.

Check out the following list of four AngularJS performance enhancing tools.

## 1. Protractor

[Protractor](http://www.protractortest.org/#/)is the most powerful automated end-to-end Angular testing tool, which was developed by the Angular team. Protractor is built by combining some great technologies like NodeJS, Selenium, webDriver, Mocha, Cucumber, and Jasmine.

[](https://www.markupbox.com/blog/wp-content/uploads/2017/06/protractor-logo.png)

## 2. GulpJS

[GulpJS](http://gulpjs.com/) is used for automating repetitive tasks, the streaming build system, and linting JavaScript using JSHint or ESLint.

[](https://www.markupbox.com/blog/wp-content/uploads/2017/06/gulp-1.png)

## 3. TestingWhiz

[TestingWhiz](http://www.testing-whiz.com/)is one of the most user-friendly test automation tools because of its codeless scripting feature. TestingWhiz offers an end-to-end testing solution to test AngularJS applications. It has a variety of test commands to easily create tests related to AngularJS. With a more dynamic wait command, TestingWhiz synchronizes with the varying server wait times for AngularJS components.

[](https://www.markupbox.com/blog/wp-content/uploads/2017/06/testingwhiz-version-2-0-released.jpg)

## WebdriverIO

[WebdriverIO](http://webdriver.io/) lets you control a browser or a mobile application with just a few lines of code. Your test code will look simple, concise, and easy to read. The integrated test runner lets you write asynchronous commands in a synchronous way so that you don’t need to care about how to handle a promise to avoid racing conditions. Additionally, it takes away all the cumbersome setup work and manages the Selenium session for you.

### [webdriverio](https://www.markupbox.com/blog/wp-content/uploads/2017/06/webdriverio.png)

## ****Conclusion****

As we have seen, [web development](https://www.valuecoders.com/?utm_source=Dzone%2FPrakhar&utm_medium=Angularjs%20Performance) is changing rapidly due to the introduction of many new and updated JavaScript frameworks. To get maximum benefits from these frameworks, you must keep optimizing performance on regular basis.

AngularJS is a powerful framework to build web applications like no other. By improving AngularJS performance, developers become able to do more with less coding. Even the risk of producing “spaghetti code” is greatly reduced.

# 11 Tips to Improve AngularJS Performance

April 28th 2015

[**TWEET THIS**](https://twitter.com/intent/tweet?text=11%20Tips%20to%20Improve%20AngularJS%20Performance&url=https%3A%2F%2Fhackernoon.com%2F11-tips-to-improve-angularjs-performance-2f2010a98479&via=akras14&hashtags=angularjs,t%C4%B1p,javascript)

I am new to Angular (even though I am not new to the web development), so please take everything that I am about to say with a grain of salt. That being said, I watched a lot of talks and read a lot of articles relevant to Angular performance, and this post is the summary of my findings.

### 1. Minimize/Avoid Watchers

Usually, if your Angular app is slow, it means that you either have too many watcher, or those watchers are working harder then they should.

Angular uses dirty checking to keep track of all the changes in app. This means it will have to go through every watcher to check if they need to be updated (call the digest cycle). If one of the watcher is relied upon by another watcher, Angular would have to re-run the digest cycle again, to make sure that all of the changes has propagated. It will continue to do so, until all of the watchers have been updated and app has stabilized.

Even though running JavaScript in modern browsers is really fast, in Angular it is fairly easy to add so many watchers that you app will slow down to a crawl.

Keep in mind the following when implementing or refactoring an Angular app.  
<http://www.codelord.net/2014/06/17/angular-performance-101-slides/>

**Watches are set on:**

* $scope.$watch
* {{ }} type bindings
* Most directives (i.e. ng-show)
* Scope variables scope: { bar: '='}
* Filters {{ value | myFilter }}
* ng-repeat

**Watchers (digest cycle) run on:**

* User action (ng-click etc). Most built in directives will call $scope.applyupon completion which triggers the digest cycle.
* ng-change
* ng-model
* $http events (so all ajax calls)
* $q promises resolved
* $timeout
* $interval
* Manual call to $scope.apply and $scope.digest

This was the biggest win for our app. I am not going to go into too much details, but I found the article bellow to be extremely helpful.

In addition to infinite scroll, make sure to use track by when possible.

[AngularJS](https://docs.angularjs.org/api/ng/directive/ngRepeat#tracking-and-duplicates)

For example a unique step id, is a good value to track by when doing an ng-repeat.

<li ng-repeat="Task in Tasks track by Task.Id></li>

### 3. Use Bind once when possible

Angular 1.3 added :: notation to allow one time binding. In summary, Angular will wait for a value to stabilize after it’s first series of digest cycles, and will use that value to render the DOM element. After that, Angular will remove the watcher forgetting about that binding.

[AngularJS](https://code.angularjs.org/1.3.15/docs/guide/expression#one-time-binding)

See the Pen [AngularJS — Bind Once Example](http://codepen.io/akras14/pen/QbWJRg/) by Alex ([@akras14](http://codepen.io/akras14)) on [CodePen](http://codepen.io/).

If you are on pre 1.3 version of Angular you can use this library to achieve similar results:

[Pasvaz/bindonce](https://github.com/Pasvaz/bindonce)

### 4. Use $watchCollection instead of $watch (with a 3rd parameter)

$watch with only 2 parameters, is fast. However, Angular supports a 3rd parameter to this function, that can look like this: $watch('value', function(){}, true). The third parameter, tells Angular to perform deep checking, meaning to check every property of the object, which could be very expensive.

To address this performance issue, angular added $watchCollection('value', function(){}). $watchColleciton acts almost like $watch with a 3rd parameter, except it only checks the first layer of object’s properties, thus greatly improving the performance.

Official doc:

[AngularJS](https://code.angularjs.org/1.3.15/docs/api/ng/type/$rootScope.Scope#$watchCollection)

Helpful blog post:

[Scope $watch() vs. $watchCollection() In AngularJS](http://www.bennadel.com/blog/2566-scope-watch-vs-watchcollection-in-angularjs.htm)

### 5. Avoid repeated filters and cache data whenever possible

One time binding does not seem to play well with filters. There seems to be work arounds to make it work, but I think it’s cleaner and more intuitive to simply assign the needed value to a variable (or set it as a property on an object, if you are dealing with a lot of variables).

For example, instead of:

{{'DESCRIPTION' | translate }}

You can do:

* In JavaScript $scope.description: $translate.instant('DESCRIPTION')
* In HTML {{::description}}

Or instead of: {{step.time\_modified | timeFormatFilter}}

* in JavaScript

var timeFormatFilter = $filter(‘timeFormatFilter’); step.time\_modified = timeFormatFilter(step.time\_modified);

var timeFormatFilter = $filter(‘timeFormatFilter’);

step.time\_modified = timeFormatFilter(step.time\_modified);

* In HTML {{::Path.time\_modified}}

### 6. Debounce ng-model

If you know there is going to be a lot of changes coming from an ng-model, you can de-bounce the input.

For example if you have a search input like Google, you can de-bounce it by setting the following ng-model option: ng-model-options="{ debounce: 250 }.

This will ensure that the digest cycle due to the changes in this input model will get triggered no more then once per 250ms .

[AngularJS](https://docs.angularjs.org/api/ng/directive/ngModelOptions)

### 7. Use ng-if instead of ng-show (but confirm that ng-if is actually better for your use case)

ng-show will render an element, and use display:none to hide it,  
ng-if will actually removes the element from DOM, and will re-create it, if it’s needed.

You may need ng-show for an elements that toggles on an off often, but for 95% of the time, ng-if is a better way to go.

### 8. Use console.time to benchmark your functions

console.time is a great API, and I found it particularly helpful when debugging issues with Angular performance. I placed a number of those calls through out my code, to help me confirm that my re-factoring was in fact improving the performance.

[Console.time()](https://developer.mozilla.org/en-US/docs/Web/API/Console/time)

The API looks as such:

console.time(“TimerName”);

//Some code

console.timeEnd(“TimerName”);

And here is a simple example:

console.time(“TimerName”);

setTimeout(function(){

console.timeEnd(“TimerName”);

}, 100);

//In console $: TimerName: 100.324ms

Note: If console.time is not precise enough for your needs, you can get a more accurate reading using performance.now(). You will have to do your own math, if you choose to take this path.

[https://docs.google.com/presentation/…](https://docs.google.com/presentation/d/15XgHRI8Ng2MXKZqglzP3PugWFZmIDKOnlAXDGZW2Djg/edit#slide=id.g10d2b49c1_0143)

totalTime = 0; count = 0;

var someFunction = function() {

var thisRunStartTime = performance.now();

totalTime += performance.now() — thisRunStartTime;

console.log(“Average time: “ + totalTime/count);

### 9. Use native JavaScript or Lodash for slow functions

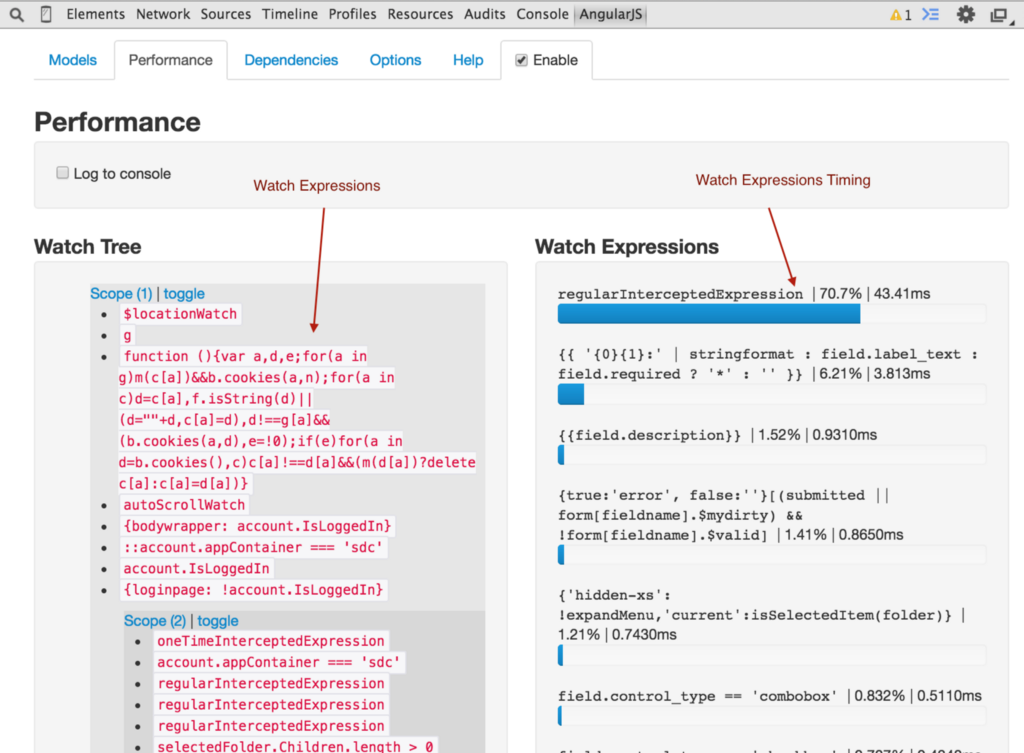
Our app was already using lodash, so there was no overhead for me to use it in my optimization. If lodash was not include, I would probably try to re-write everything in native JavaScript.

In my tests I got a significant performance boost by simply re-writing some of the basic logic with lodash, instead of relying on built-in Angular methods (which have to account for much more generic use cases).

Maintainer of Lodash John-David Dalton is also a co-creator of <https://jsperf.com/>, and he is all about the performance. So I trust him and his library when it comes to speed.

### 10. Use Batarang to benchmark your watchers

Batarang is a great tool from the Angular team, and it was very helpful in my debugging efforts. It has a lot of useful features, but the one that was the most relevant to this use-case is the performance tab.



Make sure to get the stable version, which seems to work for the majority of users.  
<https://chrome.google.com/webstore/detail/angularjs-batarang-stable/niopocochgahfkiccpjmmpchncjoapek>

Watch this video to get more insight into the Batarang.

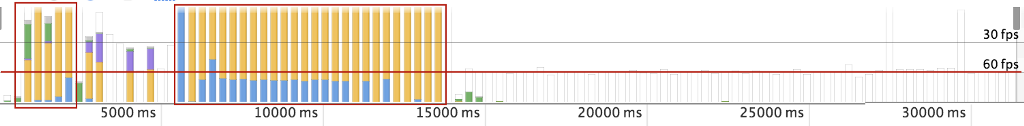
### 11. Use Chrome Timeline and Profiler to identify performance bottlenecks

I like to think of myself as a Chrome Dev Tools power user. But it’s not often that I get a to use the Timeline and Profiler views. In this project, both were extremely helpful.

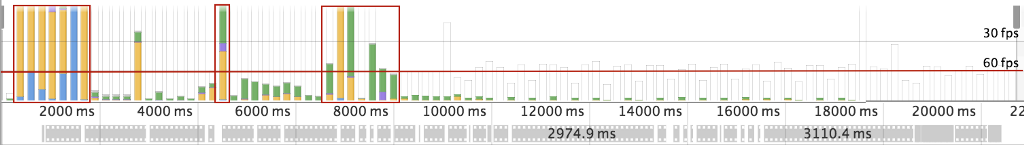
**Pro Tip:** If you use console.time API (see tip #8), the time period will get highlighted on your timeline snapshot. So you can examine the exact time period that you care about the most.

[How to Use the Timeline Tool | Web | Google Developers](https://developer.chrome.com/devtools/docs/timeline#user-produced-timeline-events)

The timeline view, and the magic 60fps line is crucial. When I started on our project, the app was rendering full steam for 15 seconds or more, becoming almost completely unresponsive to the user.



After performance optimization, the app now fully renders in less then 2 seconds (note that the time scale is different), allowing users to freely interact with the user interface after a relatively short delay.

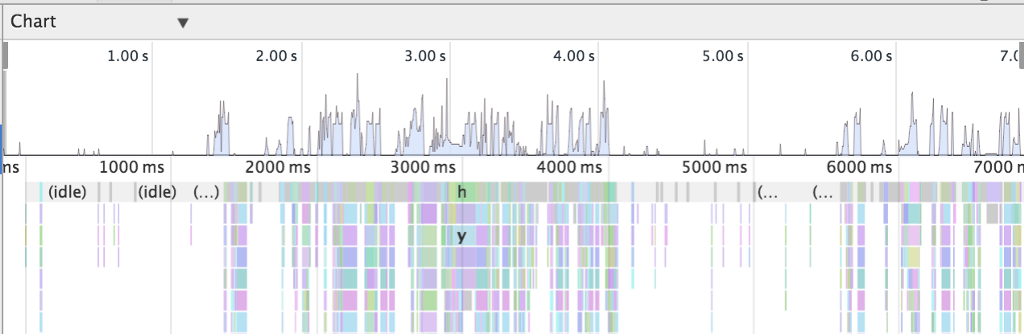


It is clear from looking at the image that the app could be further optimized. But even as is, I am very happy with the improvements to the user experience.

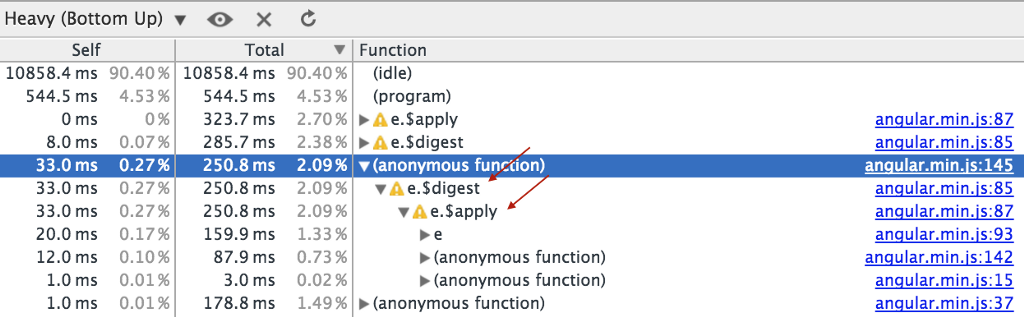
To get more experience with Timeline view, check out these web audits by Paul Irish:

Finally, I wanted to mentioned the Profiling tab in Chrome Dev tools, and the JavaScript CPU profiler in particular. It has 3 views:

**1. Chart view is similar to the timeline, but it makes it a bit easier to jump to the source code of the function of interest.**

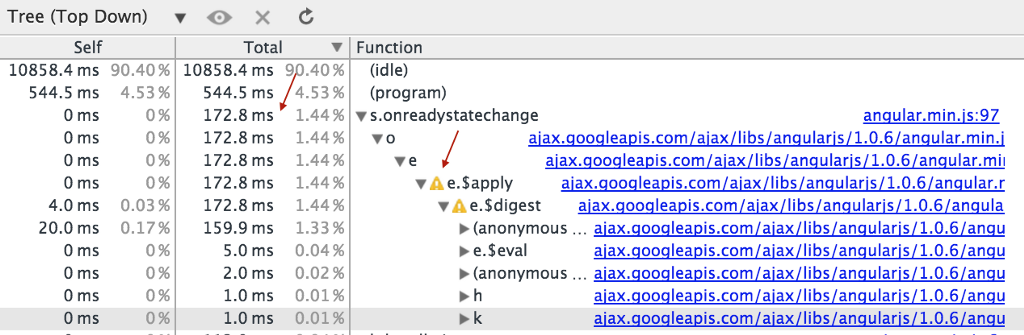


**2. Heavy (Bottom up view)**  
This view identifies heavy user functions, and shows you the reverse call stack to help pinpoint origination of the function. Note how $digest comes before the $apply, indicating the reverse order.



**3. Tree (Top Down)**  
Exposes the functions from which the heavy consumption originated, and then you can drill down to find the offending function.

Also note the yellow triangle with a “!”, if you however over it, it will identify a potential optimization problem.



[Analyze Runtime Performance | Web | Google Developers](https://developer.chrome.com/devtools/docs/cpu-profiling)

P.S. [If you found these tips helpful, please consider reviewing this post on Amazon](https://www.alexkras.com/recommends/angular-performance)

Originally published at [www.alexkras.com](https://www.alexkras.com/11-tips-to-improve-angularjs-performance/) on April 28, 2015.