**Angular A Platform for Building Mobile and Desktop Web Applications.**



[1.Introduction 4](#_Toc25155569)

[1.1 Features 4](#_Toc25155570)

[1.2 Principal 5](#_Toc25155571)

[1.3 Advantage 5](#_Toc25155572)

[2.Stakeholders 5](#_Toc25155573)

[2.1 Overview 5](#_Toc25155574)

[2.2 Stakeholders Analysis 7](#_Toc25155575)

[2.3 Power Interest Grid 8](#_Toc25155576)

[3.Context View 9](#_Toc25155577)

[3.1 System scope 9](#_Toc25155578)

[3.2 Dependencies: 11](#_Toc25155579)

[4.Development view 13](#_Toc25155580)

[4.1 Main Components of Angular 14](#_Toc25155581)

[4.2 Four core features 16](#_Toc25155582)

[5.Performance Perspective 17](#_Toc25155583)

[5.1 Measurement of the performance 18](#_Toc25155584)

[5.2 The way to improve performance 18](#_Toc25155585)

[6. Deployment View 21](#_Toc25155586)

[6.1 Third-party Software Requirements 22](#_Toc25155587)

[6.2 Specialist knowledge 23](#_Toc25155588)

[7.Evolution Perspective 24](#_Toc25155589)

[8. Functional view 27](#_Toc25155590)

[8.1 Angular module 27](#_Toc25155591)

[8.2 $Scope 29](#_Toc25155592)

[8.3 Extensibility 33](#_Toc25155593)

[9.Technical-Debt 33](#_Toc25155594)

[9.1 Static Analysis Result 34](#_Toc25155595)

[9.2 Static Analysis 36](#_Toc25155596)

[10.Conclusion 39](#_Toc25155597)

[11.References 41](#_Toc25155598)

# 1.Introduction

AngularJS was born in 2009 and was founded by Misko Hevery and later acquired by Google. It is an excellent front-end JS framework that has been used in Google's many products. AngularJS has many features, the most central of which are: MVC, modularization, automated two-way data binding, semantic labeling, dependent injection, and so on.

## 1.1 Features

* **MVC**

AngularJS absorbs the basic principles of traditional MVC and is closer to MVVM (Model-View-View Model)

* **Modularization**

In AngularJS, a template is an HTML file that extends the mapping of HTML content from model to view.

* **Dependent injection**

Angular has a built-in dependency injection subsystem that helps developers develop, understand, and test applications more easily

* **data binding**

Can help you avoid writing a lot of initial code to save time, that is, handle listening and querying DOM operations

* **semantic labeling**

Can be used to create custom labels. Can be used to decorate elements or manipulate DOM attributes

## 1.2 Principal

Does not recommend developers to manually operate the DOM, which is still operating DOM in the bottom layer

## 1.3 Advantage

Angular minimizes DOM operations on the page

Let JavaScript development focus on business logic

More reasonable code structure

Lower maintenance costs

Combine page structure and data with simple instructions

Componentized programming with custom instructions

# 2.Stakeholders

As the definition above, stakeholders in Angular are interested in and influence Angular. More specifically, stakeholders can make a huge impact on Angular. Let's look at Angular's stakeholders, which include developers, maintainers, users, and other very important roles. After that we will have a deeper understanding and try to give the Power Interest Graph of Angular.

## 2.1 Overview

1. **Acquirers**

Acquirers, or called as investors, is the first part of the chain which decides the survival of tech companies. They are more concerned about the profit model of the product and the specific operation cost and market profit of the product. Angular's predecessor was Angular. AngularJS was born in 2009 and was created by **Misko Hevery** and others and later acquired by **Google**. Angular is built by a team of engineers who share a passion for making web development feel effortless, and now Angular is managed by them and their collaborators. Actually they are a large group of people, so we just list some of them.

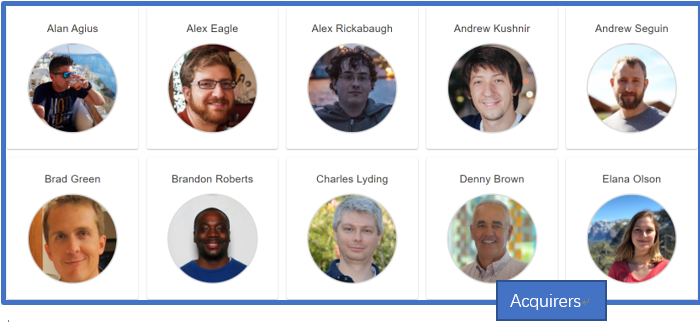


Figure 1: Acquirers of Angular

1. **Developers**

As an open-source program on GitHub, dozens of developers would like to take part in this job. We can easily find them on GitHub. To make it clearer who contributed the most to Angular, here we list the contributors sorted by number of commits.

|  |  |  |  |
| --- | --- | --- | --- |
| Developer | Commits | Additions | Deletions |
| [petebacondarwin](https://github.com/petebacondarwin) | 1120 | 406,538 | 247,475 |
| [gkalpak](https://github.com/gkalpak) | 933 | 96,310 | 65,985 |
| [vsavkin](https://github.com/vsavkin) | 952 | 165,431 | 117,393 |
| [vicb](https://github.com/vicb) | 916 | 160,889 | 182,997 |
| [IgorMinar](https://github.com/IgorMinar) | 856 | 244,201 | 221,320 |

Table 1: Most contributed developers in trems of commit number

1. **Maintainers**

Maintainers are responsible for managing the development and evolution of Angular. They discuss the development direction of Angular. What’s more, they accept and review pull-requests. We can view the main contributors on GitHub. They are the same as the Developers above.

1. **Users**

Angular is an MVVM framework used by **Google** and other front-end engineers in major projects.

1. **Suppliers**

**GitHub** can be regarded as the supplier of Angular owing to it coordinated the process of Angular’s development. Besides, **JavaScript** is the language of Angular, which can be also considered as the supplier.

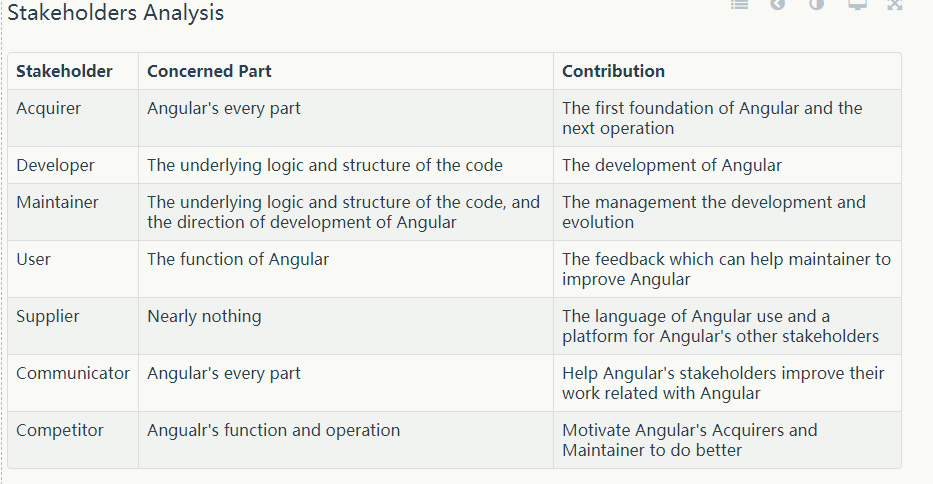
1. **Communicators**

Angular has some large communities on **GitHub**, **Gitter**，\*Stack Overflow\* and so on. Every stakeholder of Angular can get help and find more about Angular here.

1. **Competitors**

Obviously, there are a lot of companies which also engaged in front-end development, such as **Vue** and **React**.

## 2.2 Stakeholders Analysis

**Table 2: Stakeholders Analysis

## 2.3 Power Interest Grid

The following figure shows the Power Interest Grid. Power Interest Grid contains the main stakeholder categories and more detailed explanation will be listed.

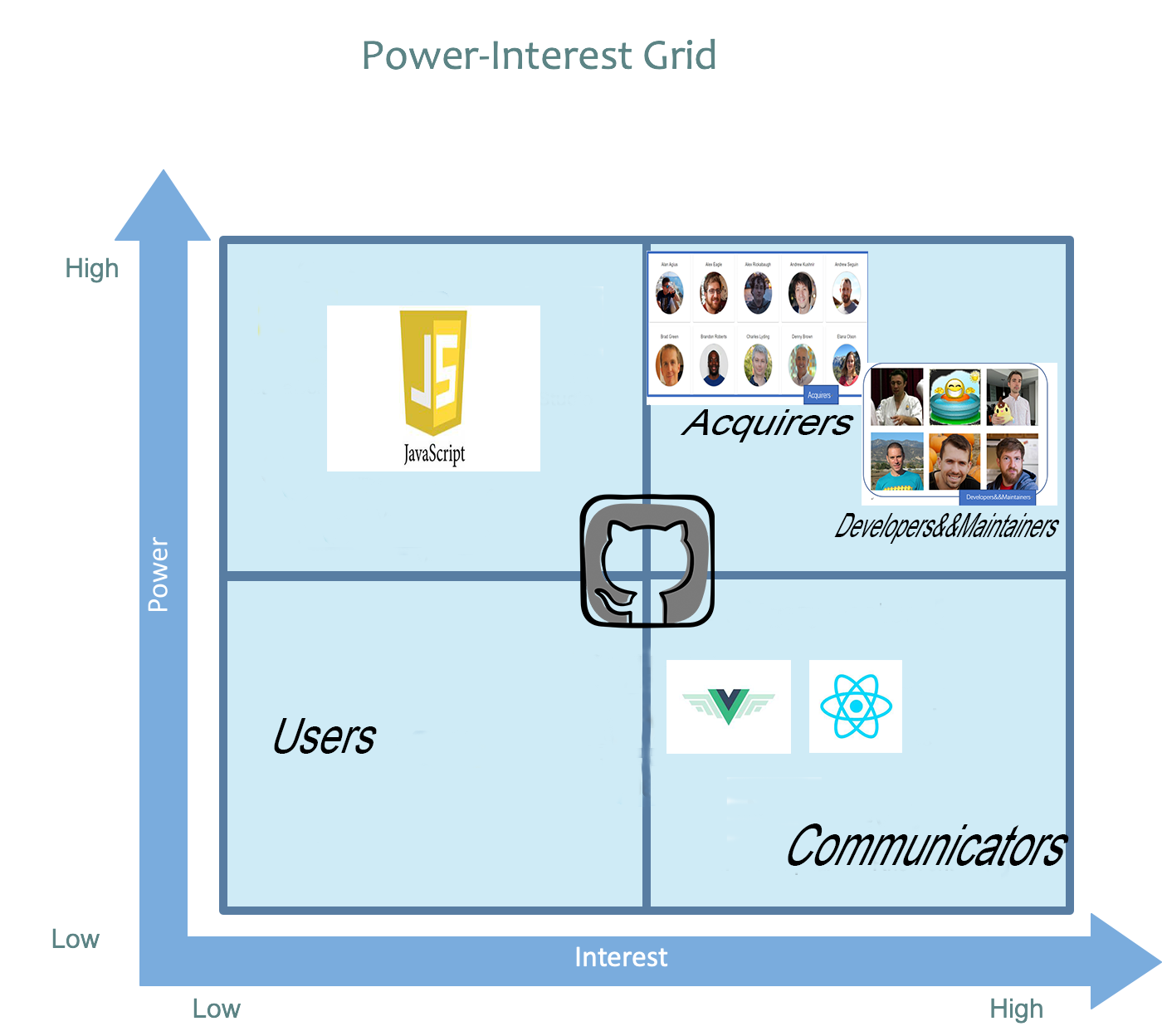


Figure 2: Power Interest Grid

* **Low power and low interest**

Users, GitHub are stakeholders who do not has any control over Angular, and do not have significant roles in the development of Angular.

* **Low power and high interest**

GitHub is the supplier of Angular and follow the latest development of Angular. Besides, the communities of Angular on GitHub, Gitter and Stake Overflow are the communicators. They are active in the discussion of Angular, but they do not have significant power to directly change the Angular. What’s more, the Competitors of Angular will be greatly affected by the changes in Angular, but they have nearly no power to influence it.

* **High power and low interest**

JavaScript are stakeholders who directly affect the development of Angular because JavaScript is the language Angular uses. But JavaScript won’t pay attention on Angular.

* **High power and high interest**

Core developers, maintainers and acquirers have significant interest and power in developing Angular.

# 3.Context View

We will discuss all external entities that Angular interacts with and the context in which they interact with each other in this section. Through this analysis, we want to get an overview of the ecosystem in which Angular resides through which we can identify dependencies and users of interest.

## 3.1 System scope

According to the introduction of [Angular at its website](https://angular.io/), Angular is defined as ‘**Angular is a development platform for building mobile and desktop web applications using TypeScript/JavaScript and other languages.**’ The scope of the software is clearly defined here. It is constrained to focus on one task, which is develop web applications, and on a specific platform.

When we focus on the dependencies and interactions between the system and the environment. We can summarize several features of Angular.

* **DEVELOP ACROSS ALL PLATFORMS**

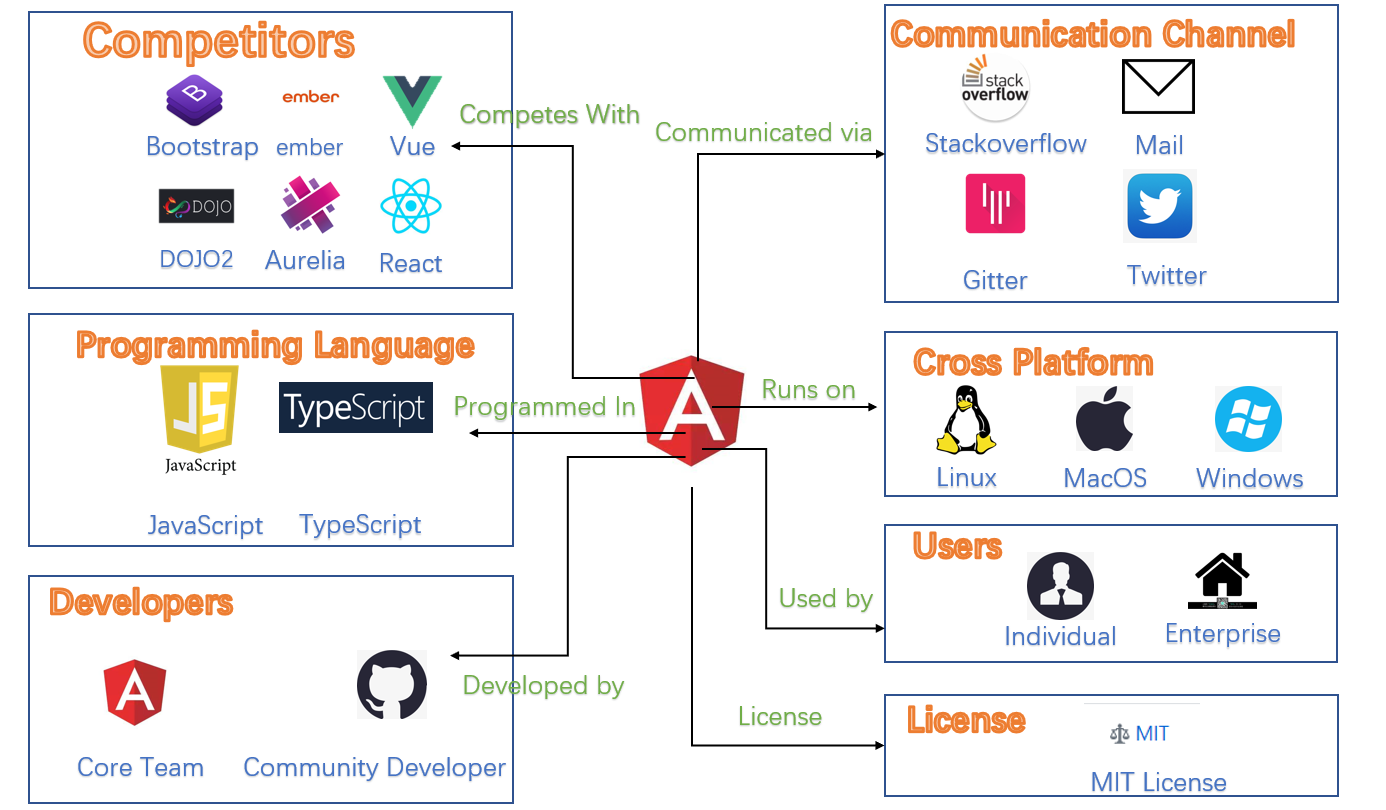
Code created by Angular can be easily run on different platforms, such as Web, mobile web, Windows, Linux, macOS.

* **SPEED&PERFORMANCE**

Through the rendering capabilities of Web Worker and server, angular has good performance and responsiveness on the web platform. With the help of RxJS, immutable.js, Angular can adapt to the needs of massive data.

* **External Entities**

The connections between Angular and its environment is summarized in the figure.



The external entities can be split into three groups.

1. **Competitors**

Angular is of course not the only platform that provides front-end functionalities. When Angular was initially created, products with the same function already exist on the market, but they have their own advantages and disadvantages. The following is a list of competitors that provide in some way the same functionalities that Angular provides.

* [React](https://github.com/facebook/react)
* [Vue](https://github.com/vuejs/vue)
* [Dojo2](https://github.com/dojo/core)
* [Ember](https://github.com/emberjs/ember.js)
* [Aurelia](https://github.com/aurelia/framework)
* [BootStrap](https://github.com/twbs/bootstrap)

1. **Software Platform & Programming languages & Dependencies**

**Programming languages**: Angular is almost entirely written in TypeScript and JavaScript and some other languages useful in Angular.

**Platforms**: Angular is well tested for compatibility on all three major operating systems (Linux, Windows, OSX), and it’s also available on mobile.

## 3.2 Dependencies:

**Language Dependencies**: As mentioned above, Angular is based on a specific number of languages. Naturally, in the process of running, the foundation of the environment of these languages is needed. That is, Angular relies on these environments in terms of language.

**Package Dependencies**:

The official website on Angular's official website lists the packages needed to configure the angular environment. The packages listed in the [dependencies](https://angular.io/api/core/ResolvedReflectiveFactory#dependencies) section of  package.json  are essential to run the Angular projects.

The [dependencies](https://angular.io/api/core/ResolvedReflectiveFactory#dependencies) section of can be divided into three types:

**Angular packages**: Some Angular core packages have certain rules in naming. Such as Angular core and optional modules; their package names begin with @ angular /.

**Support packages**: Some necessary third-party libraries are used to complete the normal application of Angular. For example, some of the feature packs in Node.js.

**Polyfill packages**: Polyfills plug gaps in a browser's JavaScript implementation.

More information is listed on their [own website](https://angular.io/guide/npm-packages) to introduce their npm-packages.

1. **Development & Community**

**Development**: A platform that plays an important role during development is GitHub, which is used for code versioning, issue tracking and project management. All code gets tested and integrated via continuous tools.

**Community**: The main communication channel used in the development of Angular is through their GitHub repository and their mailing list. More open discussion is done via Stack Overflow and the Angular Gitter channel. The Angular developers also provide support to users through their own website.

1. **Users**

The users of Angular can be divided into two subcategories. The individual community and enterprise.

**Individual community**

The individual community are the types of users that uses Angular as hobby for research. They do not intend to make money by using Angular. Such users are developers and universities.

**Enterprise**

Enterprise are the users who do use Angular as a tool in their company to help their product. Some major companies that use Angular commercially are:

* PayPal
* Netflix
* Lego
* Lambda Test

1. **License**

The Angular license closely follows the [MIT license](https://github.com/angular/angular/blob/master/LICENSE).

# 4.Development view

Angular JS is not just a class library, but a complete framework. Angular is an MVC framework that provides you with many API functions. You can achieve more complex effects without writing many native js, such as animation, $. animate. It avoids you interacting with multiple class libraries and requires familiarity with the tedious work of multiple interfaces. It is designed by Google Chrome developers and leads the next generation of Web application development. Maybe we won't use Angular JS in five or 10 years, but its design essence will continue to be used.

Knowing the developers of Angular JS, you're sure to be excited about the ability of Angular JS custom instructions (which function like custom controls on the. NET platform). Custom instructions allow you to extend HTML tags and features. Instructions can be reused and used across projects. Large Web projects can be created using Angular JS. You can split the project into multiple modules and one module into multiple module files. At the same time, you can organize these documents according to your usage habits.

## 4.1 Main Components of Angular

1. **Module**

Angular applications are modular, and they are packaged together by modules. Each Angular application has at least one module, and most applications have a number of feature modules that are used to implement specific functions.

1. **Component**

The component is responsible for controlling a small area on the screen, which we call a view.

1. **Template**

A template is an HTML file, but it is not a standard HTML file. It uses some template syntax. The template syntax allows the template to have its own logical relationship and enable simple data interaction with the component.

1. **Metadata**

Metadata tells Angular how to handle a class.

1. **Databinding**

Data binding is the most common data processing mode in Angular. Data binding plays an important role in the interaction of templates with corresponding components, as well as in the communication between parent and child components.

1. **Directive**

Since the Angular template is dynamic, you need to implement the conversion to the DOM via instructions. There are two types of instructions: structure instructions, attribute instructions.

1. **Service**

A service is a broad category that includes: values, functions, or characteristics required by an application. Almost anything can be a service. A typical service is a class with a focused, clear purpose. It should do something specific and do it well.Component classes should be kept lean. The component itself does not get data from the server, does not perform validation input, and does not write logs directly to the console. They delegate these tasks to the service. So the service is errands, the service is generally used to process the business logic, is injected into the component, the service is a global singleton. This means that the services injected into all components are the same.

1. **Denpendency injection**

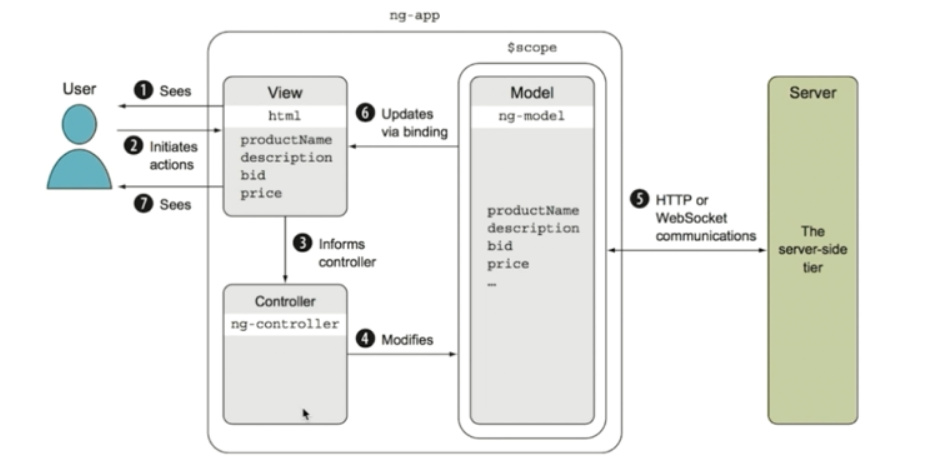
Dependency injection is a way to provide a new instance of a class, and it is also responsible for handling all the dependencies required by the class. Most dependencies are services. Angular uses dependency injection to provide new components and the services they need.

## 4.2 Four core features

* **MVC**

Angular follows the engineering MVC pattern and encourages presentation, data and logic loose coupling, and through dependency injection. Angular uses traditional server-side services in client-side web applications, such as view-independent control. Because of this, the back end reduces a lot of burden.

Angular is a typical MVC framework



* **Two-way binding**

The AngularJS framework not only applies HTML but also extends HTML to accommodate dynamic content through two-way data binding. Two-way data binding allows automatic synchronization between models and views.

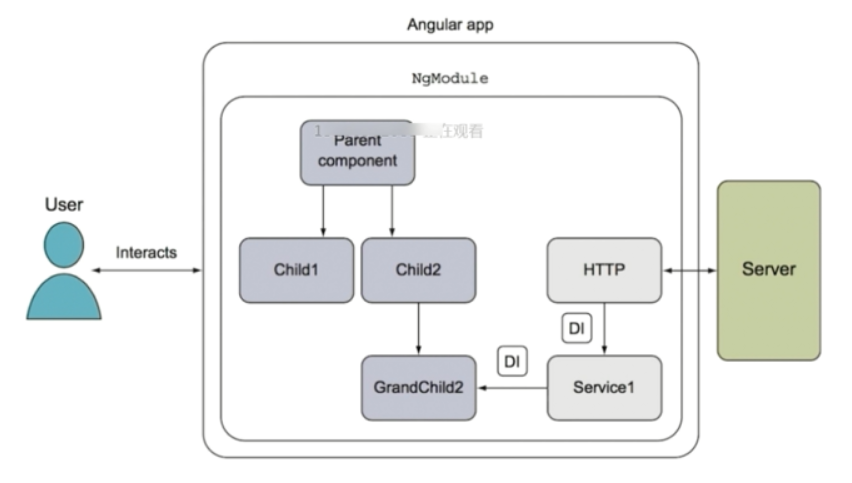
* **Dependency injection**

This is a design pattern, that is, an object depends on other objects without manual creation, so when the object is created, the objects it depends on are automatically created and injected by the framework.

* **Modular design:**

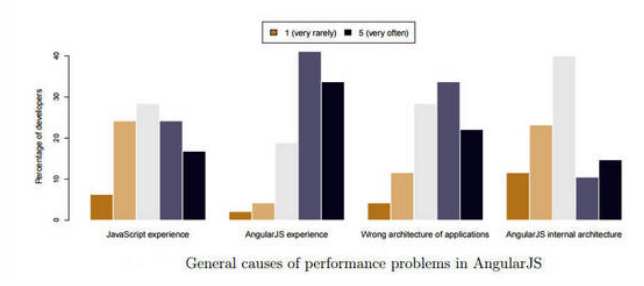
Obeying the high cohesion low coupling law.

Angular is a component tree



# 5.Performance Perspective

In this section, we discuss performance of Angular as an important factor which provide a more detailed view of its architecture concerning how Angular enables highly scalable web development. Performance of a big project, especially it is related to web application development, usually refers to its ability to satisfy users' requests simultaneously.



From the result, 45% of voters said it was due to performance issues affected by source code issues. Only 8% of voters admitted that they actually made changes. Some interviewees blamed the architecture of AngularJS. Some of them blame unnecessary two-way bindings. So we may could draw the conclusion that if we want to improve the performance, we need to design the Angular project in a smart way.

## 5.1 Measurement of the performance

The performance of AngularJS can be measured simply by its digest period. The digest cycle can be used as a loop. In this cycle, Angular checks all variable changes with all $scopes. If $scope.myVar is defined in the controller and marked as observed, then Angular will monitor the myVar update, which is checked once per iteration.

## 5.2 The way to improve performance

1. **Benchmark Watcher with the Batarang tool**

For teams using Angular, Batarang is a great development tool that can reduce your stress on debugging. Although there may be many new features, they are mainly used to help you describe and track the performance of your AngularJS. In addition, it is through the monitoring tree to determine which range is not destroyed, for example, by checking whether the memory usage has increased to determine whether to destroy.

1. **Use Native JavaScript or Lodash**

Lodash improves application performance by simply overwriting some of the basic logic rather than relying on the built-in AngularJS approach. If your application doesn't include Lodash, you may need to rewrite all the code in Native JavaScript.

1. **Chrome development tool Profiler to identify performance bottlenecks**

This is a handy tool that lets you choose which profile type to create. Record the allocation time point, get the heap snapshot, and record the assigned configuration file for memory profiling. After this performance optimization, your application will be fully rendered in less than two seconds, and users can interact with it at will.

1. **Minimize observers**

AngularJS is completely around its digest cycle. Whenever a digest cycle is triggered, it loops through each binding to detect model changes. By reducing the number of observers, you can reduce the time spent in each digest cycle. It also reduces the memory footprint of the application.

1. **ng-if is better than ng-show**

The ng-show directive toggles the CSS display property on a specific element, while the ng-if directive actually removes the element from the DOM and recreates it when needed. In addition, the ng-switch directive is an alternative to ng-if, which has the same performance.

1. **Don't use ng-repeat**

Not using the ng-repeat directive is the biggest win for the application, so it is recommended to avoid using ng-repeat and build HTML using JavaScript. For vocal applications, using ng-if leads to an increase in unnecessary observers. Using the ng-bind-html directive is a better solution to get rid of this problem.

1. **Use $watchCollection (with the third argument)**

It's good to use $watch with two arguments - but $watch with three arguments when using***$watch('value',function(){},true)***, which allows Angular to perform an in-depth check (to check Each property of the object). But the cost can be very expensive. Therefore, in order to solve such a performance problem, Angular provides the ***$watchCollection(‘value’, function(){})*** directive, which is almost identical to the third argument, but checks the first layer of the object's properties at low cost.

1. **Use console.time for debugging problems**

If your application is trying to debug problems and affect Angular performance, you need to use console.time , which is a nice API.

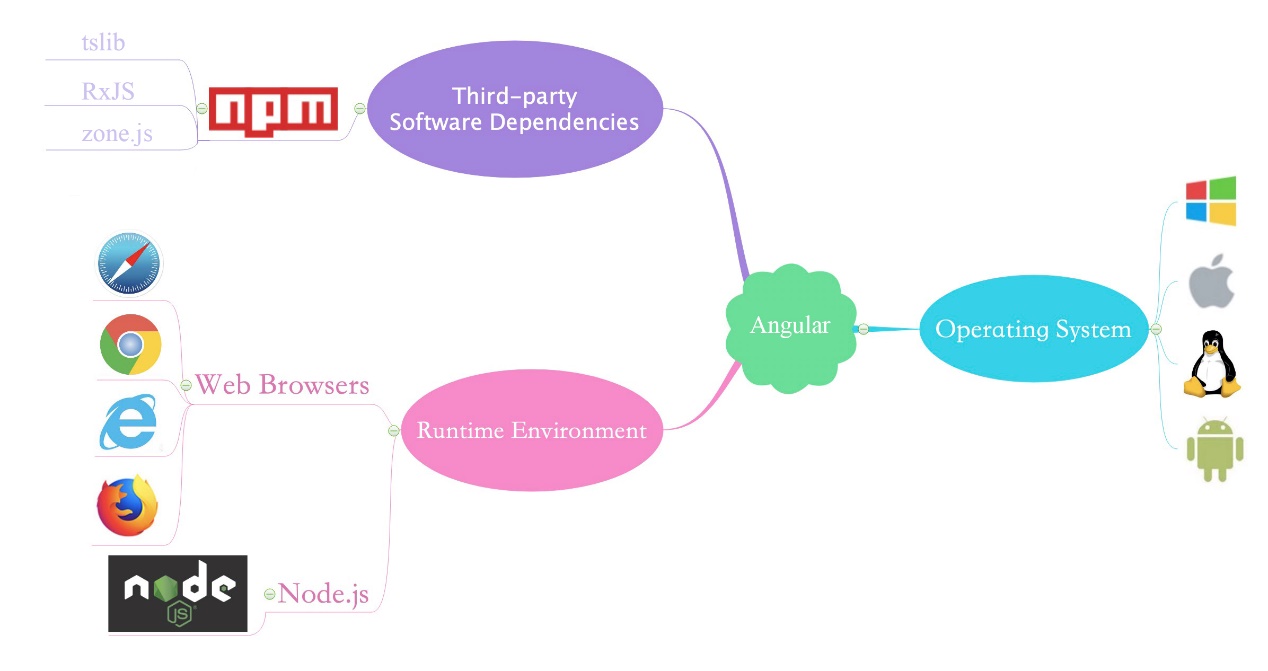
1. **Anti-binding ng-model**

You can unbind the input ng-model . For example, for anti-binding search input like Google, you must use ng-model-options=”{debounce:250}”. This allows it to detect a trigger at least every 250ms during the digest cycle.

# 6. Deployment View

According to Rozanski and Woods, the deployment view describes the environment where the system will be deployed.

In this case, Angular runs on the server side. It depends on several third-party software to realize its functions to render the web page on the client side. For the client side, customers can choose any web browser if it supports the corresponding version of JavaScript and HTML that Angular uses. While on the server side, Angular is a cross-platform product since it runs on Node.js, which is a cross-platform runtime environment in JavaScript for developing web applications on the server side. Angular supports a series of calls to the large number of [API libraries](https://angular.cn/api) with many features. However, you will need to install the necessary third-party libraries and packages for API functions before using these APIs.



*Deployment environment and dependencies of Angular*

## 6.1 Third-party Software Requirements

* **Node.js**: Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. It is like JVM for Java.
* **npm**: Short for Node Packaged Module. npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js.
* **tslib**: It is a runtime library for TypeScript that contains all the TypeScript helper functions. Since the Angular framework is also used in the part of the TypeScript process.
* **RxJS**: RxJS is an asynchronous programming library, and it implements event-based programming through observable sequences.
* **zone.js**: Zone.js implements the concept of Zones in JavaScript, inspired by Dart. Zone is the context of the execution process, and can be passed between asynchronous tasks. You can think of it as a Java-like TLS thread local storage technology, but it is also used in the JavaScript language for the key technologies in the Angular framework.
* **Other Json package for specific features**: When we are not satisfied with the features that come with Angular, we want to add new features. JS packages that implement new features need to be added by the npm management tool.

What's more, here are some recommended web browsers, Chrome, Firefox, Safari, and IE10 including later versions. And Facebook has also offered us some useful tools to develop using Angular. For example, we can use "ng new my-app" to build a development environment for Angular automatically without too much complex configuration.

## 6.2 Specialist knowledge

To use Angular, the following specialist knowledge is required.

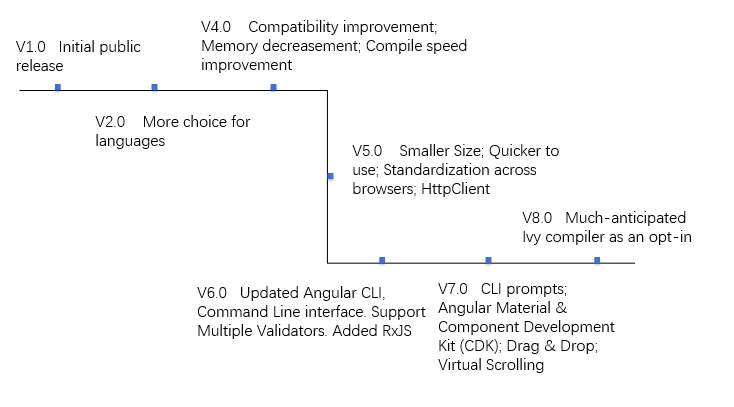
* Basic knowledge of programming in JavaScript.
* Familiar with web programming: HTML, CSS, Web Model, etc.

If you have basic knowledge in both fields, you are good to go in the realm of Angular.

# 7.Evolution Perspective

This section summarizes and analyzes feature evolution of Angular by comparing its changes among different releases, which helps us understand its core design and architecture.

Angular's initial public open-sourced version was released on October, 2010. But Angular's 1.x version is Angular's predecessor, AngularJS, a front-end MVC framework that is very different from today's Angular framework. And after that, it varies greatly between different releases. The latest version(v8.2.12) of Angular was released on Oct 18, 2019, and there are still changes that have landed in master but are not yet released. Angular's official website states that the latest 9.0.0 version will be officially launched in October/November 2019, and the 10.0.0 version will be officially launched in May 2020. The following figure shows the timeline of Angular version releases based on its [changelog](https://github.com/angular/angular/blob/master/CHANGELOG.md) and [version release posts](https://blog.angular.io/tagged/release%20notes).



*Feature evolution of Reac*

Angular uses semantic versioning to mark its major releases, minor releases, and bug fix patches. Here, we will mainly discuss the changes between its major versions.

It must be explained in advance that Angular has evolved from Angular1 (IE AngularJS) to Angular8, which does not have the version number of Angular3. It was skipped because the Angular team wanted to sync the angular version with the angular router version.

V1.0(1.0-1.7) Initial public release (v1.0). Provide a framework for client-side model–view–controller (MVC) and model–view–ViewModel (MVVM) architectures.

V2.0(2.0-2.4) More choice for languages. Any of the languages from ES5, ES6, TypeScript or Dart to write Angular 2 code. Added Mobile oriented. Some other minor changes and bug fixes.

V4.0(4.0-4.4) Compatibility improvement. Memory of generated code for your components decreasement, compilation speed improvement, better bug fixes alert. Support for more edition of Typescript. Added HttpClient, a library for making HTTP Requests. Some other minor changes and bug fixes.

V5.0(5.0-5.2). Memory of Angular decreasement. Performance improvement. Added Multiple names to components and directives while exporting. Deprecated @angular/http in Angular and replaced with @angular/common/http library. Standardization across browsers improvement. Some other minor changes and bug fixes.

V6.0(6.0-6.1)Updated Angular CLI, Command Line interface.Updated Component Development Kit and Angular Material. Support for Multiple Validators. Support for o several functions that can be used to create new observables via RxJS. Some other minor changes and bug fixes.

V7.0(7.0-7.2) Added CLI prompts. Added Angular Material & Component Development Kit (CDK). Improvements in CDK and Angular Material that include refresh, virtual scrolling, large lists of data, dynamic loading and unloading of parts of the DOM. Added Drag & Drop. Added Virtual Scrolling. Some other minor changes and bug fixes.

V8.0(8.0-8.2) Addedmuch-anticipated Ivy compiler as an opt-in. Added differential loading of modern JavaScript, dynamic imports for lazy routes, support for web workers, TypeScript 3.4 support, and Angular Ivy as an opt-in preview.

# 8. Functional view

## 8.1 Angular module

A module is a functional block used to combine related components, instructions, services, and so on. It can be understood that each part to be used in an application is classified according to a certain law to form a complete application.

Angular modules can be generated by the angular.module(name, requires) method:

·name: the name of the module;

·requires: the list of dependencies of the module (can be injected into the list of objects in the module), because angular.module(name) is to get the module, so if the module does not have a module to depend on, this parameter needs to be set to an empty array[] .

angular.module(name) is equivalent to the getter method of the AngularJS module, which is used to get a reference to the module.

1. **Benefits of using modules:**

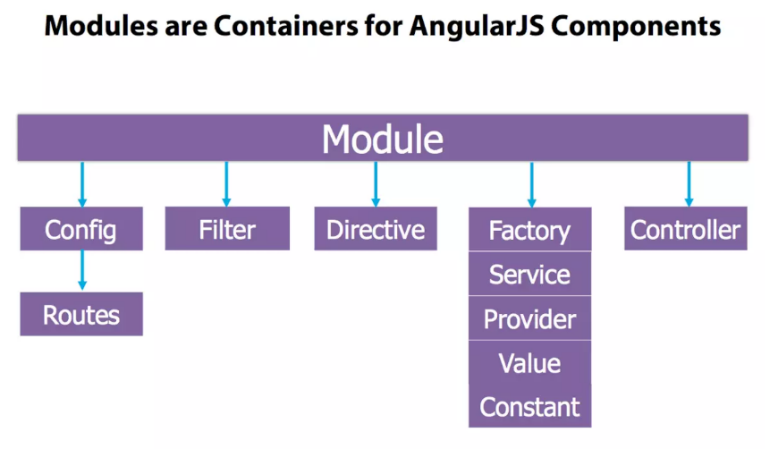
·Keep the global namespace clean;

·Writing test code is easier and keeps it clean so that it's easier to find features that are isolated from each other;

·Easy to reuse code between different applications;

·Enables the app to load parts of the code in any order.

1. **The parts contained in Module:**



***Structured Angular module***

It is recommended to break the application into multiple modules:

·Service Module: for service claims

·Instruction module: for instruction declaration

·Filter module: for filter declarations

·Application layer module: This module depends on the above module

The reason for this decomposition is that it is often necessary to ignore the initialization code in the test, which is often difficult to test. It is easy to ignore in the test by putting it in a separate module. Testing can also be more focused by only loading modules.

1. **Module loading and dependencies**

Load

Module loading is divided into two phases: config and run.

config: During the module loading phase, AngularJS configures the module during the provider registration and configuration process. Throughout the AngularJS workflow, this phase is the only part that can be modified before the app launches. Only providers and constants can be injected into config.

AngularJS will execute these functions in the order in which they are written and registered, which means that we can't inject a provider that has not yet been registered (the only exception is the constant() method, which is always executed before all configuration blocks).

The .config() function takes a single argument: configFunction, which AngularJS executes when the module loads.

run: is executed after the injector is created, it is the first method executed in the AngularJS application (closest to the main method concept). Only instances and constants can be injected into the run. Usually used to register global event listeners.

Rely

A module can list the list of modules it depends on. These dependent modules need to be loaded before the module is loaded. That is, the configuration block of the dependent module is executed before the configuration block of the module. Each module can only be loaded once, even if multiple modules depend on it.

## 8.2 $Scope

1. **Concept**

A scope is an object that points to an application model. The scope has a hierarchical structure, a root scope, multiple sub-scopes, different locations, and different roles. Scopes can monitor expressions and pass events. Apply a link between HTML (view) and JavaScript (controller).



1. **Features**
2. $scope provides some tool methods $watch(), $apply()

$watch() is used to listen for model changes, and it will prompt you when the model changes.

Expression: $watch(watchExpression, listener, objectEquality);

Its parameters:

watchExpression: The object to listen to, which can be an angular expression such as 'name', or a function such as function(){return $scope.name}.

listenern: A function or expression that will be called when watchExpression changes. It takes three arguments: newValue (new value), oldValue (old value), scope (scoped reference).

objectEquality: whether deep listener, if set to true, tells Angular to check for changes to each property in the monitored object. If you want to monitor the individual elements of an array or the properties of an object instead of a normal value, then you should use it.

$apply() is used to propagate changes to the model. An AngularJS external controller (DOM events, external callback functions such as jQuery UI space, etc.) must call $apply after calling the AngularJS function.

1. $scope can propagate events for an object, similar to a DOM event.
2. $scope is not only the basis of MVC, but also the basis for implementing two-way data binding. The scope provides an expression execution context, for example, the expression {{username}} itself is meaningless. It only makes sense with the username attribute specified by the scope $scope.
3. $scope is a POJO (Plain Old JavaScript Object).
4. $scope is a tree structure parallel to the DOM tag.
5. The child $scope object inherits the properties and methods on the parent $scope.
6. **Usage**

·Provides observers with the ability to monitor changes to the data model

·Can notify the entire app of changes to the data model

·Can be nested to isolate business functions and data

·Provide a context execution environment for expressions

When you link an element of the view to a variable in the $scope object, AngularJS creates an internal clock so that you can see any change in the variable. The function we are talking about is $scope.watch().

While our application is running, AngularJS calls the function $scope.$Digest(). This checks all the clocks and verifies if any of the observed variables have changed; if one of them has changed, the corresponding listening function is called. The listener function executes the code defined within it.

For example: if our variable is a chain that has been linked to spam and changes, the watch will call a listener function (and this will change the content of the spam with the new chain within our variable). We can easily say that the function $digest() is responsible for activating the update.

$apply() takes a function or chain of expression AngularJS, and then calls the function $digest() so that all our watchers are reviewed.

When should we use it? When we change a variable, AngularJS automatically takes control of what needs to be done. For example ng-click, ng-class, etc. are events that are contained in $scope.$apply(). If we use setTimeout () we will create a new shift that calls a function when the time runs out, AngularJS does not know about this new shift, so anything we do in the function will not be reflected in the DOM, and that’s when you should use $scope.$apply ().

For example:

setTimeout(function (){

$scope.$apply(function (){

$scope.domtext = "i've waited enough, i'm calling this one";

});},5000);

Another way:

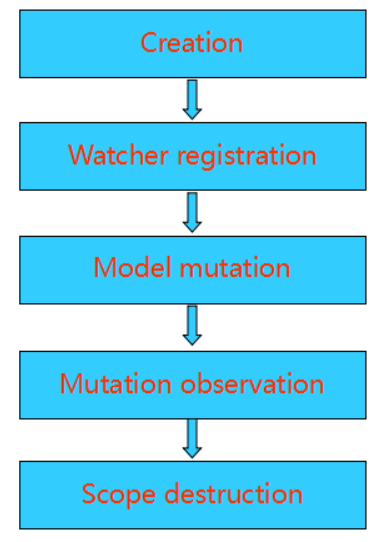
setTimeout(function (){

$scope.domtext = "i've waited enough, i'm calling this one";

$scope.$apply();},5000);

If the code is not within the $apply() function, like the second example, the error will be thrown out of AngularJS, and if there is an error in your application, it will not work and notification will not be given. $apply() has a try / catch in it, so the error is always detected and the $digest() will continue, this continuation of the function $digest() is done due a “finally” clause.

1. **Life circle**



1. Create - The more scope is created and injected by the injector when the app starts. During the template connection phase, some instructions create their own scope.
2. Register Observer - During the template connection phase, the scope's listener will be registered. This is also used by the listener to identify model state changes and update the view.
3. Model state changes - Update model state must occur in the scope.$apply method to be observed. The Angular framework encapsulates the $apply process without worrying about it.
4. Observe the model state - At the end of $apply, angular performs the $digest process from the root scope and spreads to the child scope. The expression or method observed during this process checks if the state of the model changes and performs an update.
5. Destroy the scope - When the subscope is no longer needed, the scope is destroyed by scope.$destroy() and the resource is reclaimed.

## 8.3 Extensibility

Custom directives are used in AngularJS to extend the functionality of HTML. Custom directives are defined using "directive" function. A custom directive simply replaces the element for which it is activated. AngularJS application during bootstrap finds the matching elements and do one time activity using its compile() method of the custom directive then process the element using link() method of the custom directive based on the scope of the directive.

# 9.Technical-Debt

After analyzing Angular, we can now have a deeper understanding of the source code. Some changes have taken place in the system. Therefore, we must do something to identify the features added to it. Technical debt is the concept that reflects the additional development work that results from the implementation of the error. Long-term adjustments are required when using a code that is easy to implement in the short term instead of a total solution. There are two implementation methods, static code analysis and manual code checking. And this section is not only related to the code, but also to low documentation or code coverage.

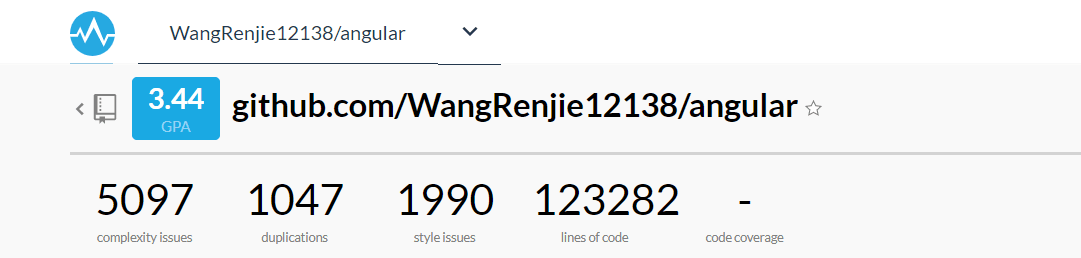
## 9.1 Static Analysis Result

As a method in technical debt analysis, static code analysis often involves an assessment of code style, complexity, repeatability, security, and the like. For cost reasons, we chose two free static code analysis tools, CODEBEAT and CodeFactor. These two tools can check the code separately to rate them from A to F. They can lead to code complexity, code issues, and code duplication.

1. **CODEBEAT**

After the analysis, the total rating of the Angular project was 3.44 points. (The total score is 4 points)

The analysis showed that there were 5,097 complexity issues, 1047 duplications, and 1990 style issues in Angular123282lines of codes.



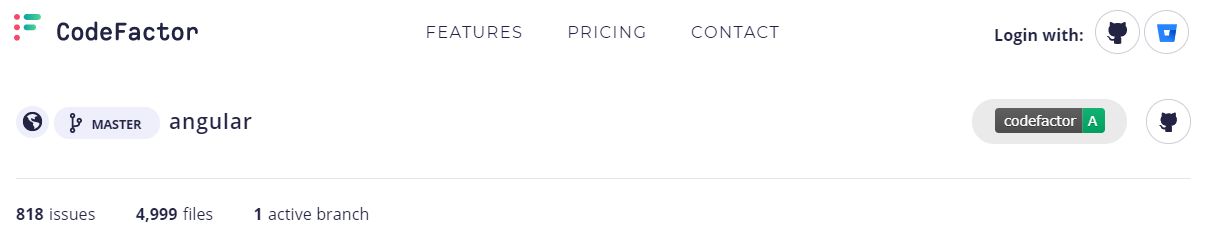
CODEBEAT listed all the files associated with the above issues and gives a rating from A to F, where F is the worst case. The figure below shows part of the list of files containing problematic code.



1. **CODEFACTOR**

CODEFACTOR and CODEBEAT have simliar function. In CODEFACTOR, Angular received A's rating.

The analysis showed that there were 818 issues in Angular4999files. Among them, the three most issues are Complexity, Style, and Maintainability.



## 9.2 Static Analysis

We can see that the number of Angular issues is huge in CODEBEAT and CODEFACTOR. This is because CODEBEAT and CODEFACTOR have different criteria for the issue. Next we will analyze the analysis results of Angular in CODEFACTOR.

1. **Analysis by Category**

We selected the three categories with the most issuesiss for analysis.

1. **Complex Code**

Long or complex methods may lead to code that is difficult to understand and change. Such code may also attract abusive use of comments when a small method or variable are viable alternatives. The key to understand about small methods is good naming. If you have a good name for a method you don't need to look at the body. These problems can be solved bt refactoring.

Refactoring：

* Extract Method
* Replace Temp With Query
* Replace Conditional with Polymorphism
* Replace Method with Method Object
* Decompose Conditional
* Preserve Whole Object
* Introduce Parameter Object

1. **Style Issues**

In Angular, the three most popular style issues are:

* **Calls to 'console.log' are not allowed.**

The measures we can take for this kind of problem is Banning the use of specified console methods because In general, console methods aren’t appropriate for production code.

* **Unnecessary semicolon**

Enforces consistent semicolon usage at the end of every statement. Although the complier can ignore this kind of problems, the maintainers of Angular still need to pay attention to them.

* **Unexpected unit(length-zero-no-unit)**

Lengths refer to distance measurements. A length is a dimension, which is a number immediately followed by a unit identifier. However, for zero lengths the unit identifier is optional. The length units are: em, ex, ch, vw, vh, cm, mm, in, pt, pc, px, rem, vmin, and vmax.

1. **Maintainability issues**

There are a lot of variables are defined but never used in Angular. Variables that are declared and not used anywhere in the code are most likely an error due to incomplete refactoring. Such variables take up space in the code and can lead to confusion by readers.

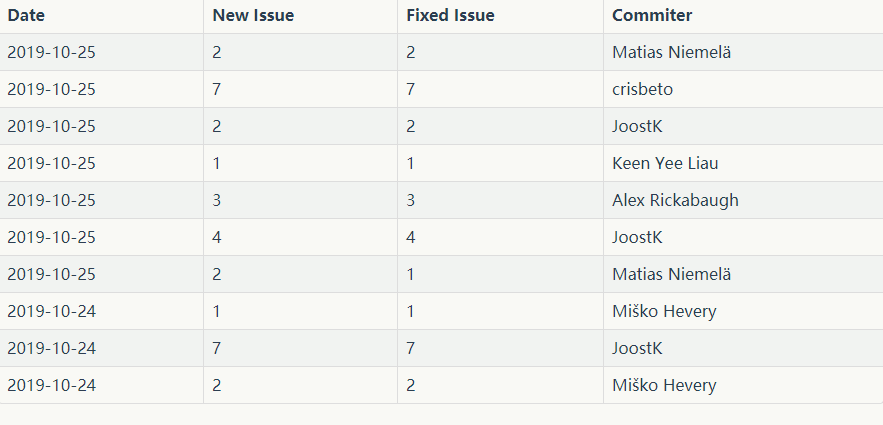
There are some rule details about this kind of issues. This rule is aimed at eliminating unused variables, functions, and parameters of functions.A variable is considered to be used if any of the following are true:

* It represents a function that is called (doSomething())
* It is read (var y = x)
* It is passed into a function as an argument (doSomething(x))
* It is read inside of a function that is passed to another function (doSomething(function() { foo(); }))

1. **Analysis by Time**

After analyzing the types of issues that have occurred the most, we will analyze the time of the issue. We can see that there are less and less issues occured in the past year. If there is an issue occurs, it will be fixed in a short time.

Based on the results given by CODECACTOR, we can see that even though Angular maintains a high frequency update close to every day, Angular developers and maintainers attach great importance to Technical Debt and are consciously solving this problem. We have selected the last ten updates with new issues.

****

*Table 2: Recent Issue Change Situation*

As far as Angular's performance is concerned, we have reason to believe that Angular has a good response to Technical Debt.

# 10.Conclusion

Angular is written in JavaScript and TypeScript, designed to decouple the logic of your application from DOM manipulation and is designed to make dynamic web page updates a platform for easily building web and mobile applications.

First of all, in the **Stakeholder Analysis**, we discussed various roles of stakeholders involved in Angular. The core members of Angular, which take on the responsibilities of development and evolution, play numerous different roles of Angular. What’s more, they maintain and promote Angular with the help of community contributors (most of the community contributors come from GitHub).

Secondly, the **Context View** shows the responsibilities and external entities used by Angular. Therefore, the reader can clearly understand the interaction in the Angular environment.

Third, we explored in detail the core composition, important features, organizational forms of Angular and its superiority as a framework in the **Development view**. Some of its novel uses and differences have also been elaborated.

Next, the **deployment view** gives a description of the React deployment environment. Detailed instructions for various third-party software requirements to help programmers set up the development environment.

Moreover, we learned about the important role of modules in Angular and Angular's Extensibility in the **Functional view**. We also explored the core position of $Scope in Angular and its usage and features.

From the **Performance Perspective**, we discuss the factors that will influence the Angular development and also give you some effective advice on how to improve the performance of the Angular. In a other word, the usage standard.

In the **Technical Debt section**, we used tools such as codefactor and codebeat to analyze technical debts within Angular. We find the problems in Angular and give some methods to solve them. However, Angular still has a good response to Technical Debt in the analysis by time.

From the E**volution Perspective**, we summarize and analyze Angular's functional evolution by comparing Angular's changes between versions, which helps us understand its core design and architecture.

Since its inception in 2009, Angular has been thriving in the field of web application development. Its ability to span all platforms and focus on speed and performance makes it popular with web application developers. Angular has become a very popular framework and is used in many large platforms such as z ***Zhihu***, ***Smartisan***, ***Aliyun***, ***Worktile***, ***Teambition*** and ***Ele*** (which is a famous take out platform) Etc. We firmly believe that Angular can prosper in the future.

With this valuable opportunity, we can focus on Angular's architectural analysis in 12 weeks. In the process, we gradually learned Angular through multiple meetings and iterative revisions to the document. In addition, we sincerely hope that this document will benefit users, developers and those interested in Angular.

# 11.References

1. <https://www.imooc.com/article/29750>
2. <https://www.jianshu.com/p/aa2446510c97>
3. <https://www.cnblogs.com/airen123/p/9479399.html>
4. <https://blog.csdn.net/Mr_xiaoshulin/article/details/86029436>
5. <https://github.com/liyihann/react/blob/master/项目体系结构文档.md#7>
6. [Wikipedia. Stakeholder (corporate)[EB/OL］.Stakeholder (corporate) - Wikipedia](http://wikipedia.moesalih.com/Stakeholder_(corporate))
7. <http://www.apkbus.com/blog-907513-68272.html>
8. <https://www.cnblogs.com/wlpower/p/6406890.html>
9. <https://hackernoon.com/why-should-you-use-angularjs-key-features-and-reasons-f2be0da0cba9>
10. <https://www.sitepoint.com/10-reasons-use-angularjs/>
11. <https://blog.csdn.net/n950814abc/article/details/80047992>