

## Architecture

### Version 1.0 vs. Version 2.0

Angular 2.0 shows a substantial change in the structure as compared to version 1.0. The architecture of Angular v1 is based on MVC whereas the architecture of Angular v2 is based on service/controller. There is very less possibility to upgrade the Angular v1 to v2, mainly developers have to rewrite the entire application code.

### Version 2.0 vs. Version 4.0

The upgrade of the version from 2.0 to 4.0 has reduced its bundled file size by 60%. The code generated is reduced and has accelerated the application development. Here the developed code can be used for prod mode and debug.

## JavaScript and TypeScript

### v 1.0 vs. v 2.0

Angular v1.0 use JavaScript to build the application while Angular v2.0 uses the Typescript to write the application. TypeScript is a superset of JavaScript which helps to build more robust and structured code. Dart can be used by developers along with TypeScript in version 2.0.

### v 2.0 vs. v 4.0

Angular v4.0 is compatible with newer versions TypeScript 2.1 and TypeScript 2.2. This helps with better type checking and also enhanced IDE features for Visual Studio Code.

## Mobile Support

Angular 2.0 has made it possible to accomplish the native applications for a mobile platform like React Native. Angular 2.0 gives us the two layers: application layer and the rendering layer. As need, any view can be rendered in runtime for the required component.

## Component-based UI

### 1.0 vs. 2.0

The controller concept which was present in Angular v1.0 is eliminated in Angular v2.0. Angular v2.0 has changed to component based UI. This helps a developer to divide the applications in terms of components with desired features and enable to call required UI. These have helped to improve the flexibility and reusability as compared to Angular v1.0

## SEO Friendly

### 1.0 vs. 2.0

With Angular v1.0 developing the search engine friendly Single Page Applications was the major difficulty. But this bottleneck was eliminated in Angular v2.0. **AngularJS development services** build SEO friendly Single Page Applications by rendering the HTML at the server side.

2.0 vs. 4.0

## Features of Angular version 4.0

### • View engine with less code

The view engine is introduced in Angular 4 where the produced code of components can be reduced up to 60%. The bundles are reduced to thousands of KBs.

### • Router ParamMap

Before Angular 4, simple object structures used to store route parameters.

These Parameters were assessed by simple standard JavaScript syntax.

Syntax

```
(parameterObject['parameter-name'])
```

But now in Angular 4, these parameters are available as a map. To use these parameters simple methods are called.

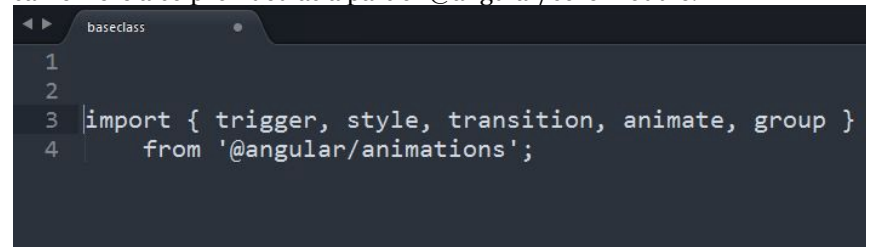
Method call

```
(parameterMap.get('parameter-name'))
```

This adds to the type security. Old values were unsafe in regards to the type as these values could take any type possible. But now, these values are string or array of strings.

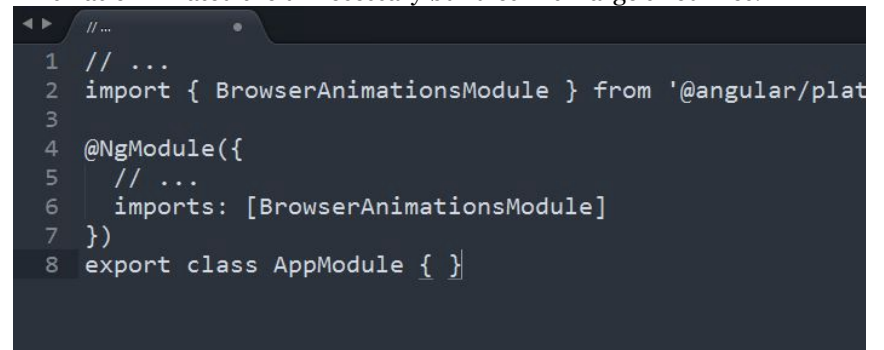
### • Animation

Up till AngularJS the code required for the animation part was always included in the application in spite of the fact that animation is actually used or not. The functions required for the same were also provided as a part of @angular/core module.



```
1
2
3 import { trigger, style, transition, animate, group }
4   from '@angular/animations';
```

But now in Angular 4, the animation is part of a separate package. This has eliminated the unnecessary bundles with large sized files.



```
1 // ...
2 import { BrowserAnimationsModule } from '@angular/plat
3
4 @NgModule({
5   // ...
6   imports: [BrowserAnimationsModule]
7 })
8 export class AppModule { }
```

Animation can also be avail from module BrowserAnimationsModule from @angular/platform-browser/animations.

### • ngIf with a new else statement

Information which is dependent of one another is displayed with the help of “conditional rendering”. If any condition is not met, then resultant element and all child elements are not added to the DOM-tree.

```
<!--ngif with else-->
<div *ngIf="hungry" ; else elseBlock ">I Am Hungry </div>
<ng-template #elseBlock>I am not Hungry</ng-template>

<!-- ngIf with then -->

<div *ngIf="hungry" ; then thenBlock ">I Am Hungry.</div>
<ng-template #thenBlock>I will eat.</ng-template>

<!-- ngIf with then and else block -->
<div *ngIf="hungry; then thenBlock else elseBlock ">I Am Hungry.</div>
<ng-template #thenBlock>I will eat.</ng-template>
<ng-template #elseBlock>I am not Hungry</ng-template>
```

Commonly there was additionally a requirement for a contrasting case, making it important to figure a similar condition a different way using another \*ngIf. We were required to use if statement all the time. While dealing with big and multiple lines of codes it was very difficult to maintain and go through these awful implications. But with Angular 4 this was solved with the help of ‘else’.

### • Smaller and faster

Angular 4 has made the code file size smaller and improved the speed of the application.

### • Pipes

The first letter of each word is changed to Uppercase with the use of Pipe. Pipe is a new title case introduced in Angular 4. It takes

first letter of word to uppercase and remaining is kept in a small case.

```
<p>{{ 1234.56 | currency : 'USD' }}</p>
<!--will display '$1234.56'-->
```

The first letter of each word is changed to Uppercase with the use of Pipe. Pipe is a new title case introduced in Angular 4. It takes first letter of word to uppercase and remaining is kept in a small case.

### • Template

- In Angular 4, the <ng-template> tag ought to be utilized rather than the <template> tag as it has been expostulated and set apart as inadmissible. Likewise, it’s presently conceivable to utilize else syntax in the template. It will pop warning on the off chance that you utilize the deprecated layout some place when you are at Angular 4, so it will be anything but difficult to spot them. It’s a breaking change; with the semantic versioning rules, this has been incorporated in Angular 4.

```
<ng-template #myTemplate> Welcome to AngularMinds</ng-template>
```

- Better and efficient debugging with the use of source maps.
- Enable to develop powerful Single Page Application (SPA) which is 100% SEO friendly.

### Http:

```
var http = injector.get(Http);
http.get('cuisines').subscribe((res:Response) => listMenu(res));
```