1. **Introduction to Angular 2 [A Most Popular JS Framework]**
2. Angular 2 is mobile oriented & better in performance.
3. Angular 2 provides more choice for languages.
4. Angular 2 implements web standards like components.
5. AngularJS 2.0 is not easy to setup as AngularJS 1.x.
6. Angular 1.x controllers and $scope are gone.
7. Different ways to define local variables.
8. Structural directives syntax is changed.
9. Angular 2 uses camel case syntax for built-in directives.
10. Angular 2, directly uses the valid HTML DOM element properties and events.
11. One-way data binding directive replaced with [property].
12. Two-way data binding: ng-model replaced with [(ngModel)]
13. Way of Bootstrapping Angular Application is changed:
14. Ways of Dependency Injection is Changed- syntax changed.
15. Way of routing is Changed- syntax changed.

**Angular 2 is a most popular framework for developing mobile apps**.  It is also for desktop as well mobile applications.  The Angular 2 is development phase you know very well and hopefully you love.

The [**Angular 2**](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html) is focusing on **data-binding**, extensible **HTML** and on application test-ability but it is still in design and prototyping stage.

**Angular framework** helps us to build client applications in **HTML** and **JavaScript**.

**Angular 2** is so **simpler**, **faster**, **modular** and instrumented design.

**Angular 2** targeting to modern browsers and it is developing using **ES6** (**The ES6 is called ECMAScript version 6**). It also support to **ECMAScript** version 5(**ES5**).

You don’t worry about the versions of ECMAScript. The compiler manages to the versioning related problems.

All the **Angular 2 framework** code is already being written in ECMAScript 6.

The set of modern browsers are

1.       Chrome

2.       Firefox

3.       Opera

4.       Safari

5.       IE Version10 and 11.

**Stayed Informed** – [13 Best Advantages for Angular2 - [Angular2vs. Angular1]](http://www.code-sample.com/2016/06/angular-2-vs-angular-1-performance.html)

**Stayed Informed** – [12 Best Advantages of Angular 2 &ReactJs [Angular2 vs. ReactJs]](http://www.code-sample.com/2016/07/angular-2-vs-react.html)

On mobiles, it is supporting to the list of Chrome on **Android**, **iOS** 6+, **Windows Phone** 8+ and **Fire-Fox** mobile and also trying to support to older versions of **Android**.

The **Angular 2** is using [**Traceur**](http://www.code-sample.com/2015/03/what-is-traceur-compiler.html)compiler to generate the nice **ECMAScript** 5/6 that runs everywhere you want to do.

**Angular 2** team working with [Traceur compiler](http://www.code-sample.com/2015/03/what-is-traceur-compiler.html) team to provide the support to build some extensions. This set of extensions called “**ES 6 +A**”.

**Angular 2 Frequently Q&A :-**

[**What is ECMAScript ES5/ES6?**](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html)

The **ECMAScript** is a scripting language which is developed by **Ecma International Org.**

Currently **ECMAScript**available in multiple versions that are **ES5 and ES6** and both of versions fully supported to **Chrome**, **Firefox**, **Opera**, **Safari**, and **IE** etc.

[**What is Traceur compiler?**](http://www.code-sample.com/2015/03/what-is-traceur-compiler.html)

The “**Traceur”**is a JavaScript**compiler**. The **Traceur**compiler is very popular now days use to allow use to use the features from the future. This compiler is fully supported to **ES5**, **ES6** and also to **vNext**.

The main goal of **Traceur**compiler is to inform to design of new JavaScript features and wrote the programming code of new efficient and good manners.

[**What is Advantages of Angular 2?**](http://stackoverflow.com/questions/34114593/angular-vs-angular-2/35175128#35175128)

1.       There is many more advantage of Angular 2.

2.       The Angular 2 has better performance.

3.       The Angular 2 has more powerful template system.

4.       The Angular 2 provide simpler APIs, lazy loading and easier to application debugging.

5.       The Angular 2 much more testable.

6.       The Angular 2 provides to nested level components.

7.       The Angular 2 execute run more than two programs at the same time.

* Angular2 is quite different
* Angular 2 provides the possibility to use different languages like TypeScript But AngularJS uses only JavaScript.
* TypeScript is OOP (Wow)
* Angular 2 is based on components
* The use of components make Angular2 more testable
* Angular2 is faster
* In Angular2 no more $Scope and controllers
* In Angular2 the use of dependency injection is enhanced

The Angular 2 architecture diagram identifies the eight main building blocks as.

1.       [Module](http://www.code-sample.com/2016/04/cannot-find-module-angular2-core.html)

2.       [Component](http://www.code-sample.com/2016/06/angular-2-components.html)

3.       [Template](http://www.code-sample.com/2016/06/angular-2-template-components.html)

4.       [Outputs](http://www.code-sample.com/2016/06/angular-2-outputs.html)

5.       [Data Binding](http://www.code-sample.com/2016/06/angular-2-template-components.html)

6.       [Directive](http://www.code-sample.com/2016/06/angular-2-directives-components.html)

7.       Service

8.       [Dependency Injection](http://www.code-sample.com/2016/04/dependency-injection-in-angular-2.html)

The Angular 2 framework consists of several libraries, the some of them working as core and some are optional.

**References,**

[https://angular.io/docs/ts/latest/guide/architecture.html#!#component](https://angular.io/docs/ts/latest/guide/architecture.html#%21#component)

<http://code.ciphertrick.com/2016/03/15/angular-2-architecture-development-setup/>

**2. Angular 2 Component Constructor vs. OnInit event**

The **constructor** is a typescript feature. The constructor is only related to class instantiation and it’s nothing to do with [Angular 2](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html) and it is use to some initialization processing with respect to class hierarchies for the newly created instance.

The **ngOnInit** event is an [Angular 2 life-cycle](http://www.code-sample.com/2016/06/angular-2-components-life-cycle.html) event/ method that are called after the first ngOnChanges. The ngOnInit method is use to parameters defined with [@Input](http://www.code-sample.com/2016/06/angular-2-components.html) otherwise the constructor is OK.

**3. 13 Best Advantages for Angular2 - [Angular2 vs. Angular1]**

**“What”? “Why”? and “Advantages of Angular2”?**

Why should you use Angular2 ? What are the Advantages of Angular2 ?

The core differences and many more advantages on [Angular2 vs. Angular 1](http://www.code-sample.com/2016/06/angular-2-vs-angular-1-performance.html) as following,

1.      Angular2 has better performance.

2.      Angular2 has more powerful template system.

3.      Angular2 provide simpler APIs, lazy loading and easier to application debugging.

4.      [Angular2](http://www.code-sample.com/2011/11/angularjs-tutorial-step-by-step.html) much more testable

5.      Angular2 provides to nested level components.

6.      Angular2 execute run more than two programs at the same time.

7.      Angular1 is controllers and $scope based but Angular2 is component based.

8.      The Angular2 structural directives syntax is changed like ng-repeat is replaced with \***ngFor** etc.

9.      In Angular2, local variables are defined using prefix (**#**) hash. You can see the below \*ngFor loop Example.

There are more advantages over performance, template system, application debugging, testing, components and nested level components.  
**For Examples as,**

**Angular 1 Controller:-**

**var** app = angular.module("userApp", []);

app.controller("productController", **function**($scope) {

$scope.users = [{ name: "Anil Singh", Age:**30**, department :"IT" },

{ name: "Aradhya Singh", Age:**3**, department :"MGMT" }];

});

**Angular 2 Components using TypeScript:-**

Here the @Component annotation is used to add the metadata to the class.

**import** { Component } from 'angular2/core';

@Component({

selector: 'usersdata',

template: `<h3>{{users.name}}</h3>`

})

**export** **class** UsersComponent {

users = [{ name: "Anil Singh", Age:**30**, department :"IT" },

{ name: "Aradhya Singh", Age:**3**, department :"MGMT" }];

}

**Bootstrapping in Angular 1 using ng-app,**

angular.element(document).ready(**function**() {

angular.bootstrap(document, ['userApp']);

});

**Bootstrapping in Angular 2,**

**import** { bootstrap } from 'angular2/platform/browser';

**import** { UsersComponent } from './product.component';

bootstrap(UserComponent);

The Angular2 structural directives syntax is changed like **ng-repeat** is replaced with **\*ngFor** etc.

**For example as,**

//Angular 1,

<div ng-repeat="user in users">

Name: {{user.name}}, Age : {{user.Age}}, Dept: {{user.Department}}

</div>

//Angular2,

<div \*ngFor="let user of users">

Name: {{user.name}}, Age : {{user.Age}}, Dept: {{user.Department}}

</div>

**4. Angular 2 Components Life cycle**

In [Angular 2](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html) [components](http://www.code-sample.com/2016/06/angular-2-components.html) life-cycle, there are several events occur to complete this life-cycle.

Components Life-cycle events

|  |  |
| --- | --- |
| **Events** | **Description** |
| ngOnChanges | Before Ng on init event, the data-bound input property value changes. |
| ngOnInit | After the first ngOnChanges event, the ngOnInit event fire. |
| ngDoCheck | During every Angular change detection cycle ngDoCheck event fire. |
| ngAfterContentInit | After projecting content into the component ngAfterContentInit event fire. |
| ngAfterContentChecked | After every check of projected component content the ngAfterContentChecked event fire. |
|  |  |
| ngAfterViewInit | After initializing the component's views and child views the ngAfterViewInit event fire. |
| ngAfterViewChecked | After every check of the component's views and child views the ngAfterViewChecked event fire. |
| ngOnDestroy | Just before Angular destroys the directive or component the ngOnDestroy event fire. |
|  |  |

For more detail, you can go below links

<https://angular.io/docs/ts/latest/guide/lifecycle-hooks.html>

**5. Angular 2 Components**

In [Angular 2](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html), the components are the main way to build or specify HTML elements and business logic on the page.

In AngularJs 1, we are handling using scope, directives and controllers but all those concepts are using in a single combined that is called components.

The component is the core functionality of Angular 2 app but we need to know to pass the data in to the components to configure them.

To build an [Angular 2](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html) application you define a set of [components](http://www.code-sample.com/2016/06/angular-2-components.html), for every HTML elements, views, and route.

[Angular 2](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html) applications must have a root component that contains all other components. That means all Angular 2 applications have a component tree.

Application è [Component](http://www.code-sample.com/2016/06/angular-2-components.html) è Component1 and Component2

**Example of Components**

import { Component} from 'angular2/core';

import {  bootstrap} from 'angular2/platform/browser';

@Component({

    selector: 'my-app',

    providers: [Service],

    template: '<div>Hello my name is {{name}}!</div>'

})

class MyApp {

    constructor(service: Service) {

        this.name = service.getName();

        setTimeout(() => this.name = 'Anil Singh,', 1000);

    }

    }

    class Service {

      getName() {

        return 'Hello';

    }

}

bootstrap(App);

In the above example, you can see the [components](http://www.code-sample.com/2016/06/angular-2-components.html) have combination of selector, providers and template.

The component will be created using tag <my-app></my-app> in our HTML page and after our application constructor called and will return to “Hello my name is Anil Singh!”.

***Angular 2 Component Summary***

·         Angular 2 Component meta-data annotation is used to register the components.

·         Angular 2 components are used to create UI widgets.

·         Angular 2 components are used to split to application into smaller parts.

·         Only one component is used per DOM element.

·         In the Angular 2 components, @View, template and templateUrl are mandatory in the components.

**6. Angular 2 Directives [Components, Structural, Attribute Directives]**

 There are 3 types of directives in Angular 2.

1.     **Components** **Directives** - directives with a template

2.     **Structural Directives** - change the DOM layout by adding and removing DOM elements.

3.     **Attribute Directives** - change the appearance or behavior of an element, component, or other directive.

**What are components directives?**

A component is a directive with a template and the @Component decorator is actually a @Directive decorator extended with template oriented features.

1.     To register a component, we use @Component meta-data annotation.

2.     The directives are used to add behavior to existing DOM elements.

3.     The directives are used to design a reusable component.

4.     Only one component can be present per DOM element.

5.     Multiple directives are used per DOM element.

6.     The directive does not have @View etc.

**What are structural directives?**

The Structural directives are responsible for HTML layout and It is using Angular 2 for reshape the DOM's structure and also removing, or manipulating elements.

**What are attribute directives?**

Attribute directives are used to change the behavior, appearance or look of an element on a user input or via data from the service.

**For example as,**

**import** {Component, View} **from** 'angular2/core'';

**@Component**({

selector: 'user-detail'

})

**@View**({

template: "<div> <h1>{{userName}}</h1> <p>{{phone}}</p>"

})

**class** **userDetail** {

constructor(public userName: string, public phone: string) {}

}

<user-detail></user-detail>

I hope you are enjoying with this post! Please share with you friends!! Thank you!!!

**7. Angular 2 template**

A template is a HTML view that tells [Angular 2](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html), how to render your components in the views.

The Angular 2 templates are very similar to Angular 1 but Angular 2 have some small syntactical changes.

You can see the changes as below

·         **{}**: Is use to rendering the HTML elements.

·         **[]**: Is use to binding properties.

·         **()**: Is use to handling your events.

·         **[()]**: Is use to data binding.

·         \*:  Is use to asterisk Operations like \*ngFor="#item of items”

For more detail, you can go

<https://angular.io/docs/ts/latest/guide/template-syntax.html>

<http://learnangular2.com/templates/>

**8. Angular 2 components vs directives**

Angular 2 components vs directives

|  |  |
| --- | --- |
| [**@Components**](http://www.code-sample.com/2016/04/angular-2-components-vs-directives.html) | [**@Directive**](http://www.code-sample.com/2016/04/angular-2-components-vs-directives.html) |
| 1.       @Component meta-data annotation is used  to register the components. | @Directive meta-data annotation is used  to register the directives. |
| 2.       The components are used to create UI widgets. | The directives are used to add behavior to existing DOM elements. |
| 3.       The components are used to split to application into smaller parts. | The directives are use to design a reusable components. |
| 4.       Only one component is used per DOM element. | More than one directive are used per DOM element. |
| 5.       In the components, @View, template and templateUrl are mandatory in the components. | The directive do not have @View etc. |

**Example for using Component.**

    import {Component, View} from 'angular2/core';

    @Component({

       selector: 'hello-world'

    })

    @View({

       template: "<h1>Hello  {{angular}}</h1>"

    })

    class hello {

        constructor(public angular: string) {}

    }

    <hello-world></hello-world>

**Example for using Directive.**

import {Component, View} from 'angular2/core'';

@Component({

    selector: 'user-detail'

})

@View({

    template: "<div> <h1>{{userName}}</h1> <p>{{phone}}</p>"

})

class userDetail {

    constructor(public userName: string, public phone: string) {}

}

<user-detail></user-detail>  
  
For more detail, go to link  
<http://stackoverflow.com/questions/32680244/directive-v-s-component-in-angular2>

**9. Angular 2 @Inputs**

As you know, [Angular 2 component](http://www.code-sample.com/2016/06/angular-2-components-life-cycle.html) is the core components of applications but you need to know how to pass data into components to dynamically.

For the same, you need to define an input (use like **@Input** decorator) for a component.

[**Howto pass data into components?**](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html) You can see the below example for passing the user data in to the components.

For example,

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

@Component({

  selector: “user-info”,

  template: “<div> Hello, This is {{ userInfo.name}}</div>”

})

export class UserInfo {

  @Input() userInfo;

  constructor() { }

}

<user-info [userInfo]="currentUser"></user-info>

The components <user-info></user-info> is use to render the user information on the view.

**10. Angular 2 Outputs**

**In Angular 2, if you want to bind an event on an element, you can use the new** [**Angular 2**](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html) **events** i.e.

<button (click)="addUser()">Click</button>

The method addUser() will be called when user clicked on button.

[***What happen if you want to create a custom event?***](http://www.code-sample.com/2016/06/angular-2-outputs.html)

**Now come to the outputs, if you want to create your custom event in** [**Angular 2**](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html) **that time we will use to new** [**@Output**](http://www.code-sample.com/2016/06/angular-2-outputs.html) **decorator.**

Example

import { Component} from 'angular2/core';

import {  bootstrap} from 'angular2/platform/browser';

@Component({

    selector: 'my-app',

    providers: [Service],

    template: '<div>Hello my name is {{name}}!</div>'

})

class MyApp {

    constructor(service: Service) {

        this.name = service.getName();

        setTimeout(() => this.name = 'Anil Singh,', 1000);

    }

    }

    class Service {

      getName() {

        return 'Hello';

    }

}

bootstrap(App);

**In the above example, we will need to import Output and Event-Emitter to create our new custom event.**

import { Component , Output, EventEmitter} from 'angular2/core';

import {  bootstrap} from 'angular2/platform/browser';

@Component({

    selector: 'my-app',

    providers: [Service],

    template: '<div>Hello my name is {{name}}!</div>'

})

class MyApp {

    constructor(service: Service) {

        this.userClicked.emit(this.user);

        this.name = service.getName();

        setTimeout(() => this.name = 'Anil Singh,', 1000);

    }

 }

 class Service {

      getName() {

        return 'Hello';

    }

   @Output() userClicked = new EventEmitter();

}

bootstrap(App);

**Now when we are using the** [**components**](http://www.code-sample.com/2016/06/angular-2-components.html) **anywhere in our application, we can bind the our custom event i.e.**

<my-app (userClicked)="userClicked($event)"></my-app>

**11. Create Angular 2 Autocomplete Examples**

//IMPORT ANGULAR 2 CORE COMPONENTS.

import {Component} from '[angular2/core](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html)';

//INITIALIZATIONS OF SELECTOR AND TEMPLATE.

@Component({

    selector: '[autocomplete](http://www.code-sample.com/2016/07/create-angular-2-autocomplete-dropdown.html)',

    template: 'autocomplete-filter.html'

});

//THIS IS AUTOCOMPLETE COMPONENTS APP CLASS.

export class ComponentApp {

    //COUNTRIES LIST

    public countries = [{"name": "Brazil", "code": "BR"},

                        {"name": "Canada", "code": "CA"},

                        {"name": "France", "code": "FR"},

                        {"name": "India", "code": "IN"},

                        {"name": "Mexico", "code": "MX"},

                        {"name": "Nepal", "code": "NP"},

                        {"name": "New Zealand", "code": "NZ"},

                        {"name": "Reunion", "code": "RE"},

                        {"name": "Singapore", "code": "SG"},

                        {"name": "Spain", "code": "ES"},

                        {"name": "Sri Lanka", "code": "LK"},

                        {"name": "United Kingdom", "code": "GB"},

                        {"name": "United States", "code": "US"}];

    //CREATED THE VARIABLES FILTEREDITEMS AND REFELEMENT

    public filterKey = '';

    public filteredItems = [];

}

//FILTER DATA METHOD.

var filter = function(){

    if (this.filterKey !== ''){

        this.filteredItems = this.countries.filter(function(e){

            return (e.toLowerCase().substr(0, this.filterKey.length) ==

this.filterKey.toLowerCase()) == true;

        }.bind(this));

    }

    else{

        this.filteredItems = [];

    }

}

//SELCTION ITEM METHOD.

var select = function(item){

    this.filterKey = item;

    this.filteredItems = [];

}

//COUNTRY AUTOCOMPLETE HTML: autocomplete-filter.html

<div class="autocomplete-container">

    <div class="autocomplete-filter-section">

        <input type="text" [(ngModel)]=filterKey (keyup)=filter()>

        <label for="country">Country</label>

    </div>

    <div class="suggestions" \*ngIf="filteredItems.length > 0">

        <div \*ngFor="#item of filteredItems">

            <div>

                <a (click)="select(item)">**{{**item**}}**</a>

            </div>

        </div>

    </div>

</div>  
  
The output look like below pic.

**12. Angular 2 components css styles and styleUrls**

Hello everyone, I am going to share the CSS styling in [Angular2](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html) [components](http://www.code-sample.com/2016/06/angular-2-components.html). The Angular 2 components styling can be

1.     Inline styles

2.     CSS Style URLs and

3.     Template inline styles

The [Angular 2 components](http://www.code-sample.com/2016/06/angular-2-components-life-cycle.html) allow us to define both type of css that are inline css and styleUrls and the detail about it as given below.

Components Inline CSS Styles

        @Component({

  selector: 'customers',

  templateUrl: 'customers.html',

styles: [

    .customer {

       padding:0.3em;

       background-color: #f5f5f;

       box-shadow: inset 1px 1px 1px rgba(0,0,1,0.2);

       border-radius:1px;

       border: solid 1px #c1c1c;

    }

]

         })

Components CSS styleUrls

@Component({

  selector: 'customers',

  templateUrl: 'customers.html',

  styleUrls: ['customers.css']

})

//customers.css

.customer {

    padding:0.3em;

    background-color: #f5f5f;

    box-shadow: inset 1px 1px 1px rgba(0,0,1,0.2);

    border-radius:1px;

    border: solid 1px #c1c1c;

}

Components Template inline css styles

<style>

 .customer {

    padding:0.3em;

    background-color: #f5f5f;

    box-shadow: inset 1px 1px 1px rgba(0,0,1,0.2);

    border-radius:1px;

    border: solid 1px #c1c1c;

  }

</style>

<div class="customer">

  <div (click)="toggle()">

           {{IsVisible ? true : false }} {{CustomerUID}}

  </div>

  <div [hidden]="!IsVisible">

          <customer></customer>

  </div>

</div>

13. Angular 1 and Angular 2 Integration

1. Angular [frameworks](http://www.code-sample.com/2016/02/angular-1-and-angular-2-integration.html) provide the support of mixing code of Angular 1 and Angular 2 in the same application.

2. We can write the mixing components of Angular 1 and Angular 2 in the same view.

3. We can [inject services](http://www.code-sample.com/2016/02/angular-1-and-angular-2-integration.html) across frameworks of Angular 1 and Angular 2 in the same application.

4. Both Angular 1's and Angular 2's data [binding](http://www.code-sample.com/2016/02/angular-1-and-angular-2-integration.html) works across frameworks in the same view.

### [What are major changes in Angular 2?](http://www.code-sample.com/2016/02/major-changes-in-angular-2.html)

For more detail about it. go to below link

<http://angularjs.blogspot.in/2015/08/angular-1-and-angular-2-coexistence.html>

**14. What are major changes in Angular 2?**

The [Angular 2](http://www.code-sample.com/2016/02/major-changes-in-angular-2.html) provides lots off benefits over AngularJs 1. i.e.

1.       Angular 2 have better performance then Angular 1.

2.       Angular 2 is more powerful templating then Angular 1.

3.       Angular 2 provided [Lazy loading](http://www.code-sample.com/2016/02/major-changes-in-angular-2.html).

4.       Angular 2 provided simpler APIs.

5.       Easier debugging.

6.       Improvement over [Dependency Injection](http://www.code-sample.com/2016/02/major-changes-in-angular-2.html).

7.       Angular 2 is much more testable the Angular 1.

8.       Angular 2 provided to nesting level component.

9.       Angular 2 provided the way that is two or more systems work together as a combined system.

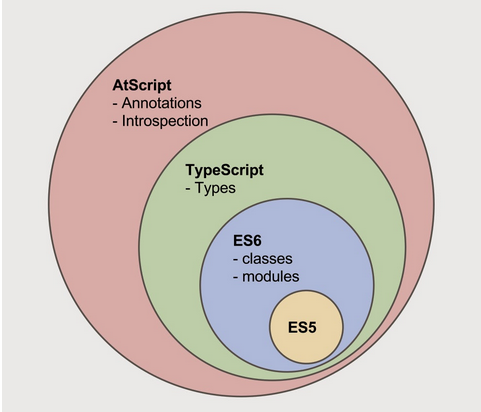
For more detail about it. go to below link

[*http://angularjs.blogspot.in/2015/08/angular-1-and-angular-2-coexistence.html*](http://angularjs.blogspot.in/2015/08/angular-1-and-angular-2-coexistence.html)

* [Integration of Angular 1 and Angular 2](https://lnkd.in/eRfM7ur)
* [What are major changes in Angular 2?](http://www.code-sample.com/2016/02/major-changes-in-angular-2.html)
* [Why Angular 2? Why not Angular 1.x+?](http://www.code-sample.com/2015/03/why-angularjs-2.html)
* [New Features of Angular 2](http://www.code-sample.com/2015/03/whats-new-in-angularjs-20.html)
* [First app with Angular 2 and ES6](http://www.code-sample.com/2015/06/quick-start-angular-2-with-es6-example.html)
* [Forms in Angular 2](http://www.code-sample.com/2015/07/angularjs-2-forms-validation.html)
* [ng-for loop in Angular 2 example](http://www.code-sample.com/2015/07/ng-for-loop-in-angular-2.html)
* [What is ECMAScript ES6?](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html)
* [What is Traceur compiler?](http://www.code-sample.com/2015/03/what-is-traceur-compiler.html)

**15. What is ECMAScript (ES6) in Angular 2?**

The [ECMAScript](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html) is known as now ES6. The [ES6](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html) is version 6. The ES6 is a scripting language and it developed by Ecma International org.

[](http://4.bp.blogspot.com/-061Zv308VJg/VSUN-VX3Y-I/AAAAAAAAExc/Do6nOnuY0ew/s1600/ES6-TypeScript-AtScript.png)

The [JavaScript](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html) is an implementation of ES6. The ES6 features are fully supported to latest browsers(chrome, Firefox etc.)  
  
Go to in detail <http://en.wikipedia.org/wiki/ECMAScript>

A basic simple example with live demo of Add two numbers using ES6 as given below.

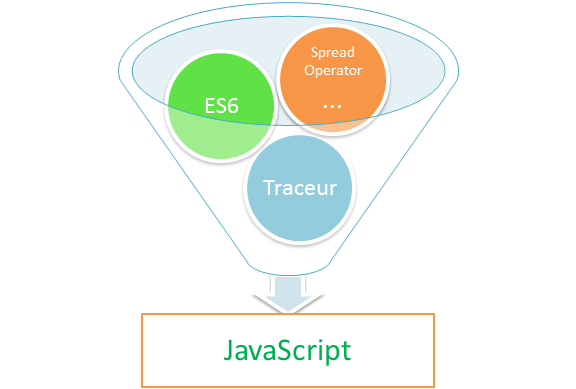
*let AddTwoNum = (num1,num2) => num1+num2;*

*console.log(AddTwoNum(4,3));*

For the output go to link <http://www.es6fiddle.com/i8fi4ths/>

**16. Angular 2 Compiler | What is Traceur compiler in Angular 2 ?**

The [Traceur](http://www.code-sample.com/2015/03/what-is-traceur-compiler.html) is a JavaScript compiler. The Traceur compiler used to allow us to use the features from the future. The Traceur compiler is fully supported to [ECMAScript](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html)(ES6) and ES.vNext also.

[](http://4.bp.blogspot.com/-1JmLIdDVgCU/VSUPX2DYFYI/AAAAAAAAEx0/vbdnbxbxBBk/s1600/Traceur+compiler.png)

The main goal of Traceur compiler is to inform the designs of new JavaScript features and also allow us to write the code in better manners and it also prefer, tell us to use design patterns.

Now the days Traceur compiler are broadly used in [Angularv2.0](http://www.code-sample.com/2015/03/why-angularjs-2.html) because Angular v2.0 are fully used to [ES5](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html) and ES6.  
  
  
***The output as given below link***  
<http://www.javascriptoo.com/traceur>  
<https://code.google.com/p/traceur-compiler/wiki/GettingStarted>

**17. angular 2 development environment**

<https://angular.io/docs/ts/latest/quickstart.html>

<https://www.youtube.com/watch?v=0scZXATiGSY>

18. [Cannot find module 'angular2/core' (Run the live error solution)](http://www.code-sample.com/2016/04/cannot-find-module-angular2-core.html)

<http://stackoverflow.com/questions/34488669/cannot-find-module-angular2-core>

**19. Cannot find module 'angular2/angular2'**

I am developing an angular2 aap with MVC5. I have register.ts file and the error detail as given below.

    import {Component, View} from '[angular2/angular2](http://www.code-sample.com/2016/04/cannot-find-angular-2-angular-2.html)';

    @Component({

          selector: 'register'

    })

    @View({

         templateUrl: '../scripts/register.html'

    })

    export class register {

    }

My bootstrap.ts file as given below

    import {bootstrap} from 'angular2/angular2';

    import {register} from '/components/register';

    bootstrap(login);

**The solution as given below.**

It just changed to module to as given below

import {Component, View} from '[angular2/core](http://www.code-sample.com/2016/04/cannot-find-angular-2-angular-2.html)'

and also changed bootstrap to as given below

import {bootstrap} from 'angular2/platform/browser'

For more detail, go to below links

<http://stackoverflow.com/questions/34697466/cannot-find-module-angular2-angular2>

<http://stackoverflow.com/questions/33332394/angular-2-typescript-cant-find-names>

**20.** [**Why Angular v2.0? Why not Angular 1.x+?**](http://www.code-sample.com/2015/03/why-angularjs-2.html)

Every AngularJs developer little bit confused and ask to angular and other community [why angularjs 2.0](http://www.code-sample.com/2015/03/why-angularjs-2.html)? why not angular 1.x+?

1. [First app with Angular 2 and ES6](http://www.code-sample.com/2015/06/quick-start-angular-2-with-es6-example.html)
2. [Integration of Angular 1 and Angular 2](https://lnkd.in/eRfM7ur)
3. [What are major changes in Angular 2?](http://www.code-sample.com/2016/02/major-changes-in-angular-2.html)
4. [New Features of Angular 2](http://www.code-sample.com/2015/03/whats-new-in-angularjs-20.html)
5. [What is ECMAScript ES6?](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html)
6. [What is Traceur compiler?](http://www.code-sample.com/2015/03/what-is-traceur-compiler.html)

Today's I attend ngConf/ng-conf meeting I asked same question and the community speaker explained in detail and finally I got my answer. The Answers are as given below

* In Angular v2.0, we are using classes more than in Angular v1.x+ and the ES6 has a better classes syntax than ES5.
* The Goal of Angular v2.0 it should be declarative.
* The  Syntax of Angular v2.0 are human-readable and human-writable.
* The Angular v2.0 deals with server data in form of a JSON and its is structural types on the JSON as well as on the Angular API.
* The Angular v2.0 will be written in ES6 and it compiled into ES5 using "*traceur*". The Angular users can write your code using ES6 or ES5 both are working fine in v2.0.
* The Angular v2.0 have the ability to attach the "meta-data" to Angular framework  "Types" or your code.
* I can say, AngularJs 2.0 is Predictable.
* I can say, AngularJs 2.0 is Tool able etc.

Go to below link for [ES6](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html) and [Traceur](http://www.code-sample.com/2015/03/what-is-traceur-compiler.html).

[What is ECMAScript ES6?](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html)

[What is Traceur compiler?](http://www.code-sample.com/2015/03/what-is-traceur-compiler.html)

For example how can say, the Angular v2.0 is simpler then Angular 1.x+. The example as given below.

This example demo using for-each in angular version v1.x vs. v2.0.   
  
AngularJs 1.x : foreach

|  |  |
| --- | --- |
| 1  2  3  4 | //Angularjs 1.x example  <li ng-repeat="user in users">    <h2>{{user.name}}</h2>  </li> |
|  |  |

AngularJs 2.0 : foreach

|  |  |
| --- | --- |
| 1  6 | //Angularjs 2.0 example  <ul>    <li \*foreach="#user in users">        {{user.name}}    </li>  </ul> |
|  |  |

For more details you can go on below link.  
<http://www.code-sample.com/2015/03/whats-new-in-angularjs-20.html>  
  
What do you think about **AngularJs 2.0**? I hope you will enjoy and put a valuable comment for the same.

**21. angular 2 hello world example**

Hello everyone, I am going to share an interesting and basic example(angular 2 hello world) using Angular 2.

I also setup the code in plnkr <http://embed.plnkr.co/jI6Nro/>

**Table of contents**

1.       [index.html](http://www.code-sample.com/2016/04/angular-2-hello-world-example.html)

2.       [main.es6.js](http://www.code-sample.com/2016/04/angular-2-hello-world-example.html)

3.       style.css

You can see the all code files on plnkr <http://embed.plnkr.co/jI6Nro/>

The [Angular 2 hello world example](http://www.code-sample.com/2016/04/angular-2-hello-world-example.html) as given below.

//index.html

<!DOCTYPE html>

<html>

<head>

    <link rel="stylesheet" href="style.css" />

    <script src="https://jspm.io/system@0.18.17.js"></script>

    <script src="https://code.angularjs.org/2.0.0-beta.8/angular2-polyfills.js"></script>

    <script src="https://code.angularjs.org/2.0.0-beta.8/Rx.min.js"></script>

    <script src="https://code.angularjs.org/2.0.0-beta.8/angular2.dev.js"></script>

    <script>

        System.config({

            paths: {

                'main.js': 'main.js'

            }

        });

        System.import('main.js');

    </script>

</head>

<body>

    <hello-world-example />

</body>

</html>

//main.es6.js

import {Component} from 'angular2/core';

import {bootstrap} from 'angular2/platform/browser';

@Component({

    selector: 'hello-world-example',

    providers: [myService],

    template: '<b class="color"><br/>{{helloworld}} world!"' +'<br/></b> {{curentdate}} <br/><br/><a href="{{webURL}}">Click for more detail..</a>'

})

class myApp {

    constructor(service: myService) {

        this.helloworld = service.getMesg();

        this.curentdate =service.getDate();

        this.webURL =service.getWebURL();

    }

}

class myService {

    getMesg() {

        return 'Welcome you, in Angular 2! "Hello';

    }

    getDate(){

        return new Date();

    }

    getWebURL(){

        return "http://www.code-sample.com/"

    }

}

//Start aplication from here.

bootstrap(myApp);

**22. Angular2 forms | Angular2 forms validation | Angular2 email Validator**

The Input form is an important part of all the development. In AngularJs 1.x, we manage using the **ng-model** but it’s have some draw back.

In Angular 2, we think over this and trying to do better. The [Angular 2 forms](http://www.code-sample.com/2015/07/angularjs-2-forms-validation.html) modules are easily to use and understand and its does not have any drawbacks.

In this article I am trying to show “**how to use Angular 2 forms**” in the common cases and the detail about it as given below.

Try the [**live example**](http://embed.plnkr.co/bqPtI7oT8KH5gwUObnG6/preview) of the code shown in this page.

[**The Live demo link**:  <http://embed.plnkr.co/bqPtI7oT8KH5gwUObnG6/preview>]  
  
The Examples,

//The HTML code sample

<div>

<userformtemplate></userformtemplate>

</div>

//The Angular 2 with ES5 code sample

**var** appForm = **function** () {

**this**.user = {};

};

**var** userFormTemplate = '<h1> Angular 2 User Form Template </h1> <br/> User Name: <input type="text" class="tb8"> <br> Confirm User Name <input type="text" class="tb8"> <br> <input type="button" value="Submit" class="testbutton">';

appForm.annotations = [

**new** angular.Component({

selector: 'userFormTemplate'

}),

**new** angular.View({

template: userFormTemplate

})];

document.addEventListener("DOMContentLoaded", **function** () {

angular.bootstrap(appForm);

});

//The full live (HTML + Angular 2 with ES5) code sample

<!doctype html>

<html>

<head>

<title>angular 2 Form</title>

<link href="style.css" rel="stylesheet" />

<script type="text/javascript" src="https://code.angularjs.org/2.0.0-alpha.28/angular2.sfx.dev.js"></script>

<script type="text/javascript">

**var** appForm = **function** () {

**this**.user = {};

};

**var** userFormTemplate = '<h1> Angular 2 User Form Template </h1> <br/> User Name: <input type="text" class="tb8"> <br> Confirm User Name <input type="text" class="tb8"> <br> <input type="button" value="Submit" class="testbutton">';

appForm.annotations = [

**new** angular.Component({

selector: 'userFormTemplate'

}),

**new** angular.View({

template: userFormTemplate

})

];

document.addEventListener("DOMContentLoaded", **function** () {

angular.bootstrap(appForm);

});

</script>

</head>

<body>

<userformtemplate></userformtemplate>

</body>

</html>

Try the [**live example**](http://embed.plnkr.co/bqPtI7oT8KH5gwUObnG6/preview) of the code shown in this page.

**23. Angular2 ngif else | Angular 2 ng-if else | ng-if else**

Hello everyone, I am going  to share the code-sample for [angular 2 \*ng-if](http://www.code-sample.com/2016/04/angular-2-ng-if-else-expression.html) expressions. 

**Syntax:**

1.       <div \*ng-if="your-condition">...</div>

2.       <div \*ngif="your-condition">...</div>

3.       <div template="ngIf your-condition">...</div>

4.       <template [ngif]="your-condition">

     <div>...</div>

</template>

Also look,  
  
<h2>Angular 2 if else</h2>

<div \*ng-for="let user of users">

    <div>**{{**user.ID**}}** - **{{**user.Name**}}**</div>

    <div \*ng-if="user.Age > 18"><img src="~/img/user.png"></div>

    <div \*ng-if="user.Age < 19"><img src="~/img/baby.png"></div>

</div>

The example as given below.

//Import root component.

import {Component, View, CORE\_DIRECTIVES} from 'angular2/angular2'

//Component

@Component({

  selector: 'toggle-app',

  bindings: []

})

//View template.

@View({

  template: `

      <div>

           <button (click)="toggle()">Toggle Button</button>

      </div>

      <div class="border">

        <div \*ng-if="isActive">

                 <h1>Hello Angular 2, Toggle Button.</h1>

        </div>

      </div>

     <p>Status(isActive): {{isActive}}</p>

  `,

  directives: [CORE\_DIRECTIVES]

})

export class App {

     isActive: bool = true;

     toggle() {

        this.isActive = !this.isActive;

     }

}

Example (live demo) as given below for using NgIf directive.

**24. Angular2 templateUrl and styleUrls**

**Angular 2 templateUrl :-** The [templateUrl](http://www.code-sample.com/2016/04/angular2-templateurl-and-styleurls.html) is a function which returns HTML template.

**Angular 2 styleUrls:-** The [styleUrls](http://www.code-sample.com/2016/04/angular2-templateurl-and-styleurls.html) is a component which use to write inline styles, style Urls and template inline style.

The example as given below using [templateUrl and styleUrls](http://www.code-sample.com/2016/04/angular2-templateurl-and-styleurls.html).

import {Component} from 'angular2/core';

@Component({

    selector: 'app'

    templateUrl: 'index.html',

    styleUrls: ['main\_style.css']

})

export class App\_Component { }

**25. angular 2 routeparams**

Search Keywords :-

[Angular 2 router link directive,  Angular 2 router-outlet directive,  Angular 2 rout config , Angular 2 Route Params]

**For live example, go to plnkr**[**http://embed.plnkr.co/vjf9Zz/**](http://embed.plnkr.co/vjf9Zz/)

**The Route Params :-**The routeparameter  is used to map given URL's parameters based on the rout URLs and It is an optional parameters for that route.

Syntax :-

             params : {[key: string]: string}

Example,

@RouteConfig([

         {path: '/employ/:id', component: employe, name: 'emp'},

])

**Router-outlet directive :-**Router-outlet directive is used to render the components for specific location of your applications. Both the template and templateUrl render the components where you use this directive.

Syntax :-

 <router-outlet></router-outlet>

**Router-link directive :-**Router-link directive is used to link a specific parts of your applications.

Syntax :-

 <router-link></router-link>

Example**,**

<a [router-link]="['/AboutMe']">About Me</a>

**The Route-Config :-**  The route config  is used to map components to URLs.

Syntax :-

 @RouteConfig([

        {path: '/',        component: Home\_Component, as: 'Home'},

        {path: '/AboutMe', component: AboutMe\_Component, as: 'AboutMe'  }

        {path: '/ContactMe', component: ContactMe\_Component, as: 'ContactMe'  }

    ])

**26. Angular 2 hidden property**

[Angular 2 [hidden]](http://www.code-sample.com/2016/04/angular-2-hidden-property.html) is  a special case binding to [hidden property](http://www.code-sample.com/2016/04/angular-2-hidden-property.html).

It is closest cousin of ng-show and ng-hide .

It is more powerful and use to bind any property of elements. Both the ng-show and ng-hide are used to manage the visibility of elements using ng-hide css class. It is also set the display property "[*display:none*](http://www.code-sample.com/2016/04/angular-2-hidden-property.html)".

All the above features are supported in [Angular 2](http://www.code-sample.com/2016/04/angular-2-hidden-property.html) but added some extra feature like animations etc.

Example for Angular 2

<div [hidden]="!active">

    Hello, This is active!

</div>

Example for Angular 1

<div ng-show="active">

    Hello, This is active!

</div>

**27. The Steps to create Angular 2 routing** as given below.

                     i.      First we need to add Angular 2 routing library

<script src="https://code.angularjs.org/2.0.0-alpha.45/router.dev.js"></script>

                   ii.      After that, we need to Import all routing directive and classes from 'angular2/router'.

import {RouteConfig,  ROUTER\_DIRECTIVES, ROUTER\_PROVIDERS, LocationStrategy, HashLocationStrategy} from 'angular2/router';

                  iii.      After that, we need to map URLs in @RouteConfig

    @RouteConfig([

        {path: '/',        component: Home\_Component, as: 'Home'},

        {path: '/AboutMe', component: AboutMe\_Component, as: 'AboutMe'  }

        {path: '/ContactMe', component: ContactMe\_Component, as: 'ContactMe'  }

    ])

                     i.            After that, use router-outlet where you want to render components.

<router-outlet></router-outlet>

                   ii.            After that, define router-link to show  your components.

<router-link></router-link>

I have setup the code on live plnkr <http://embed.plnkr.co/vjf9Zz/>

The Example as given below for create Angular 2 routing.

**Table of contents**

·     config.js

·      app/app.ts

·       Home.html

·       index.html

·       ContactMe.html

·       AboutMe.html

**//config.js file**

//The typescript used for compilation.

System.config({

    transpiler: 'typescript',

});

**//app/app.ts file**

import {Component, bootstrap, provide} from 'angular2/angular2';

import {RouteConfig,  ROUTER\_DIRECTIVES, ROUTER\_PROVIDERS,

        LocationStrategy, HashLocationStrategy} from 'angular2/router';

    // Home Component

@Component({

    selector: 'home',

    templateUrl: 'Home.html'

})

export class Home\_Component {}

    // About Me Component diective.

@Component({

    selector: 'AboutMe',

    templateUrl: 'AboutMe.html'

})

export class AboutMe\_Component {}

    // Contact Me Component diective.

@Component({

    selector: 'ContactMe',

    templateUrl:'ContactMe.html'

})

export class ContactMe\_Component {}

// Root Component diective.

@Component({

    selector: 'ROUTER\_DIRECTIVES\_EXAMPLE',

    template: `<div><h3><a [router-link]="['/Home']">Home</a>

              <a [router-link]="['/AboutMe']">About Me</a>

              <a [router-link]="['/ContactMe']">Contact Me</a>

              <h3/><hr/></div>

              <router-outlet></router-outlet>

              ` directives: [ROUTER\_DIRECTIVES]

    })

@RouteConfig([

    {path: '/',        component: Home\_Component, as: 'Home'},

    {path: '/AboutMe', component: AboutMe\_Component, as: 'AboutMe'  }

{path: '/ContactMe', component: ContactMe\_Component, as: 'ContactMe'  }

    ])

class Root\_Component{}

bootstrap(Root\_Component, [ROUTER\_PROVIDERS, provide(LocationStrategy, {useClass: HashLocationStrategy})]);

**//Home.html file**

<h2>Welcome to <a href="http://www.code-sample.com" target="\_blank">www.code-sample.com </a></h2>

**//ContactMe.html file**

<h2>Contact Me </h2>

**//index.html file**

<!DOCTYPE html>

<html>

<head>

    <title>UNDERSTAND ROUTING IN ANGULAR 2</title>

    <script src="https://code.angularjs.org/tools/system.js"></script>

    <script src="https://code.angularjs.org/tools/typescript.js"></script>

    <script src="https://code.angularjs.org/2.0.0-alpha.45/angular2.dev.js"></script>

    <script src="https://code.angularjs.org/2.0.0-alpha.45/router.dev.js"></script>

    <script src="config.js"></script>

    <script>

        System.import('app/app.ts');

    </script>

</head>

<body>

    <router\_directives\_example></router\_directives\_example>

</body>

</html>

**28. Angular 2 System is not defined.**

Angular 2 [Uncaught ReferenceError](http://www.code-sample.com/2016/04/uncaught-referenceerror-system-is-not.html): [System is not defined](http://www.code-sample.com/2016/04/uncaught-referenceerror-system-is-not.html).

You can Try to include the System JS file in your HTML header file and resolve this error.

**Include SystemJS Script.**

<script src="https://jspm.io/system@0.18.17.js"></script>

**The Example for detail as given below.**

index.html file

<!DOCTYPE html>

<html>

<head>

    <link rel="stylesheet" href="style.css" />

    <script src="https://jspm.io/system@0.18.17.js"></script>

    <script src="https://code.angularjs.org/2.0.0-alpha.36/angular2.min.js"></script>

    <script>

      System.config({

        paths: {

          'main.js':'main.js'

        }

      });

      System.import('main.js');

    </script>

</head>

<body>

    <app></app>

</body>

</html>

**main.js file**

import {Component, View, bootstrap} from 'angular2/angular2';

@Component({

  selector: 'app',

  bindings: [Service]

})

@View({

  template: '{{greeting}} I am Anil!'

})

class App {

  constructor(service: Service) {

    this.greeting = service.greeting();

    setTimeout(() => this.greeting = 'Hi,', 2000);

  }

}

class Service {

  greeting() {

    return 'Welcome you!';

  }

}

bootstrap(App);

**29. Import css using System import**

[How to import css using System import?](http://www.code-sample.com/2016/04/import-css-using-system-import.html)

Syntax :

[System.import](http://www.code-sample.com/2016/04/import-css-using-system-import.html)('./app/bootstrap/css/boots-trap.css!').then(() => {

    System.import('./app/main-app.css!');

});

**30. Angular 2 async pipe with ngFor loop**

According to [Angular 2 docs](http://www.code-sample.com/2011/11/angularjs-tutorial-step-by-step.html), “*Angular pipes, a way to write display-value transformations that we can declare in our HTML!*”

Angular 2 templates use a special Async pipe to be able to render out Observables.

**Syntax:**

\*ngFor="#obj of asyncObject | async | custom-Pipe: 'prop1' : 'prop2'"

OR

\*ngFor="#obj of (asyncObject | async)? .prop1? .prop2"

For more detail,

<https://angular.io/docs/ts/latest/guide/pipes.html>

<https://egghead.io/lessons/angular-2-rendering-an-observable-with-the-async-pipe>

**31. Angular 2 Features and Benefits**

According to [Angular 2 Docs](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html), “*Angular 2 takes a web component-based approach to building powerful applications for the web. It is used along with* [*TypeScript*](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html) *which provides support for* [*ECMAScript 5*](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html)*, ECMAScript 6, and* [*ECMAScript 7*](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html)*.*”

Angular 2 Features and Benefits

1.       Mobile First

2.       Future Ready

3.       Flexible Development

4.       Speed and Performance

5.       Supports server-side pre-rendering

6.       Simple and Expressive

7.       Comprehensive Routing

8.       Animations

9.       Hierarchical Dependency Injection

10.   Support for Web Components

11.   Internationalizations and localization (i18n) and Accessibility

Building Blocks of Angular 2 Apps

1.       [Module](http://www.code-sample.com/2016/06/introduction-to-angular-2-fundamentals.html)

2.       [Component](http://www.code-sample.com/2016/06/angular-2-components.html)

3.       [Template](http://www.code-sample.com/2016/06/angular-2-template-components.html)

4.       Metadata

5.       Data Binding

6.       Service

7.       [Directive](http://www.code-sample.com/2016/06/angular-2-directives-components.html)

8.       [Dependency Injection](http://www.code-sample.com/2016/04/dependency-injection-in-angular-2.html)

Browser Support

1.       IE,

2.       Android,

3.       Chrome,

4.       Edge,

5.       Firefox and

6.       Safari

**QUEStionS**

1. **7 Best Key Differences - Constructor Vs. ngOnInit [Angular 2]**

**Angular 2 Constructors:-**

1.      The **constructor** is a default method runs when component is being constructed.

2.      The constructor is a typescript **feature** and it is used only for a class **instantiations** and nothing to do with Angular

3.      The constructor called first time before the **ngOnInit**().

**Angular 2 ngOnInit**:-

1.      The **ngOnInit** event is an Angular 2 life-cycle event method that is called after the first ngOnChanges and the ngOnInit method is use to parameters defined with @**Input** otherwise the constructor is **OK**.

2.      The **ngOnInit** is called after the constructor and ngOnInit is called after the first ngOnChanges.

3.      The **ngOnChanges** is called when an input or output binding value changes.

**Example as,**

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

**export** **class** App **implements** OnInit{

**constructor**(){

}

ngOnInit(){

}

}

[When will ngInit be called? How would you make use of onNgInit()?](http://www.code-sample.com/2017/02/angular-2-ngoninit-and-ng-init.html)

**2.When will ngInit be called? How would you make use of ngOnInit()?**

In Angular 1.x, **ngInit** is called when template is re-rendered. In other words “**ng-init**” is called, when I take turns back to a page.

In Angular2, there is no “**ng-init**” but we can create a ways like this using the directive and ngOnInit class. Angular 2 provides life cycle hook **ngOnInit** by default.

The **ngOnInit** is invoked when the component is initialized and invoked only once when the directive is instantiated. It is a best practice to implement these life-cycle interfaces.

According to Angular2 Doc, “The **ngOnInit** is called right after the directive's data-bound properties have been checked for the first time, and before any of its children have been checked. It is invoked only once when the directive is instantiated.”

For example as,

**import** { Directive, Input } from '@angular/core';

@Directive({

selector: '[ngInit]'

})

**class** NgInit {

@Input() ngInit;

ngOnInit() {

**if**(**this**.ngInit) { **this**.ngInit(); }

}

}

In template as following,

<div \*ngIf="Timer.dateTime === currentDateTime">

<div \*ngIf="Timer.checked" [ngInit]="Start"></div>

<div \*ngIf="!Timer.checked" [ngInit]="Stop"></div>

</div>

**3. Angular 2  Component Lifecycle Hooks [Examples Also]**

The common questions ask bye most of Angular 2 lovers,

“**Could anyone tell me about the usage of ngOnInit if we already have a constructor?**” but Angular 2 provides life cycle hook ngOnInit by default.

Angular 2 Components and Directives has multiple life-time hooks where we custom logic can be executed.

**Stayed Informed** - [7 Best Key Differences [Constructor Vs. ngOnInit]](http://www.code-sample.com/2017/02/angular-2-constructor-vs-oninit.html)

**Angular 2 Constructors**:-

The constructor is a default method runs when component is being constructed.

The constructor is a typescript feature and it is used only for a class instantiations and nothing to do with Angular 2.

The constructor called first time before the ngOnInit().

**Example as**,

**import** {Component} **from** 'angular2/core';

**import** {UserService} **from** './userService';

**@Component**({

selector: ‘list-user’,

template: `<ul><li \*ngFor="#user of users">{{user.name}}</li></ul>`

})

**class** **App\_Component** {

users:Array<any>;

constructor(private \_userService: UserService) {

this.users = \_userService.getUsers();

}

}

**Angular 2 ngOnInit and ngOnChanges:-**

The ngOnInit event is an Angular 2 life-cycle event method that is called after the first ngOnChanges and the ngOnInit method is use to parameters defined with @Input otherwise the constructor is OK.

The ngOnInit is called after the constructor and ngOnInit is called after the first ngOnChanges.

The ngOnChanges is called when an input or output binding value changes.

**Examples as**,

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

export **class** **App** implements OnInit{

constructor(){

}

ngOnInit(){

}

}

**Angular 2 ngOnDestroy** :-

The ngDestroy directive is called in a component lifecycle just before the instance of the component is finally destroyed.

**Example as,**

@Directive({

selector: '[destroyDirective]'

})

**export** **class** OnDestroyDirective **implements** OnDestroy {

//Call Constructor and set hello Msg.

**constructor**() {

**this**.helloMsg = window.setInterval(() => alert('Hello, I am Anil'), **2000**);

}

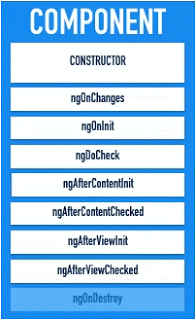
//Destroy to the component

ngOnDestroy() {

window.clearInterval(**this**.helloMsg);

}

}

[](https://3.bp.blogspot.com/-TQlp9HN7Bs0/WJ1rAbOn53I/AAAAAAAAO-Y/SjpwT4L_bKsL0JVREawW-Hih4WkLlDYBQCLcB/s1600/Angular+2+%E2%80%8AComponent+Lifecycle+Hooks.png)

**Angular 2 Complete lifecycle hook interface inventory**:-

1.      ngOnChanges - called when an input binding value changes.

2.      ngOnInit - after the first ngOnChanges.

3.      ngDoCheck - after every run of change detection.

4.      ngAfterContentInit - after component content initialized.

5.      ngAfterContentChecked - after every check of component content.

6.      ngAfterViewInit - after component's view(s) are initialized.

7.      ngAfterViewChecked - after every check of a component's view(s).

8.      ngOnDestroy - just before the component is destroyed.

**Angular 2 Lifecycle Events Log**:-

1.      onChanges

2.      onInit

3.      doCheck

4.      afterContentInit

5.      afterContentChecked

6.      afterViewInit

7.      afterViewChecked

8.      doCheck

9.      afterContentChecked

10. afterViewChecked

11. onChanges

12. doCheck

13. afterContentChecked

14. afterViewChecked

15. onDestroy

**4.Introduction to Angular 2 [A Most Popular JS Framework]**

**Angular 2 is a most popular framework for developing mobile apps**.  It is also for desktop as well mobile applications.  The Angular 2 is development phase you know very well and hopefully you love.

The [**Angular 2**](http://www.code-sample.com/2015/07/angularjs-2-documentation-with-example.html) is focusing on **data-binding**, extensible **HTML** and on application test-ability but it is still in design and prototyping stage.

**Angular framework** helps us to build client applications in **HTML** and **JavaScript**.

**Angular 2** is so **simpler**, **faster**, **modular** and instrumented design.

**Angular 2** targeting to modern browsers and it is developing using **ES6** (**The ES6 is called ECMAScript version 6**). It also support to **ECMAScript** version 5(**ES5**).

You don’t worry about the versions of ECMAScript. The compiler manages to the versioning related problems.

All the **Angular 2 framework** code is already being written in ECMAScript 6.

The set of modern browsers are

1.       Chrome

2.       Firefox

3.       Opera

4.       Safari

5.       IE Version10 and 11.

On mobiles, it is supporting to the list of Chrome on **Android**, **iOS** 6+, **Windows Phone** 8+ and **Fire-Fox** mobile and also trying to support to older versions of **Android**.

The **Angular 2** is using [**Traceur**](http://www.code-sample.com/2015/03/what-is-traceur-compiler.html)compiler to generate the nice **ECMAScript** 5/6 that runs everywhere you want to do.

**Angular 2** team working with [Traceur compiler](http://www.code-sample.com/2015/03/what-is-traceur-compiler.html) team to provide the support to build some extensions. This set of extensions called “**ES 6 +A**”.

**Angular 2 Frequently Q&A :-**

[**What is ECMAScript ES5/ES6?**](http://www.code-sample.com/2015/03/what-is-ecmascript-es6.html)

The **ECMAScript** is a scripting language which is developed by **Ecma International Org.**

Currently **ECMAScript**available in multiple versions that are **ES5 and ES6** and both of versions fully supported to **Chrome**, **Firefox**, **Opera**, **Safari**, and **IE** etc.

[**What is Traceur compiler?**](http://www.code-sample.com/2015/03/what-is-traceur-compiler.html)

The “**Traceur”**is a JavaScript**compiler**. The **Traceur**compiler is very popular now days use to allow use to use the features from the future. This compiler is fully supported to **ES5**, **ES6** and also to **vNext**.

The main goal of **Traceur**compiler is to inform to design of new JavaScript features and wrote the programming code of new efficient and good manners.

[**What is Advantages of Angular 2?**](http://stackoverflow.com/questions/34114593/angular-vs-angular-2/35175128#35175128)

1.       There is many more advantage of Angular 2.

2.       The Angular 2 has better performance.

3.       The Angular 2 has more powerful template system.

4.       The Angular 2 provide simpler APIs, lazy loading and easier to application debugging.

5.       The Angular 2 much more testable.

6.       The Angular 2 provides to nested level components.

7.       The Angular 2 execute run more than two programs at the same time.

The Angular 2 architecture diagram identifies the eight main building blocks as.

1.       [Module](http://www.code-sample.com/2016/04/cannot-find-module-angular2-core.html)

2.       [Component](http://www.code-sample.com/2016/06/angular-2-components.html)

3.       [Template](http://www.code-sample.com/2016/06/angular-2-template-components.html)

4.       [Outpouts](http://www.code-sample.com/2016/06/angular-2-outputs.html)

5.       [Data Binding](http://www.code-sample.com/2016/06/angular-2-template-components.html)

6.       [Directive](http://www.code-sample.com/2016/06/angular-2-directives-components.html)

7.       Service

8.       [Dependency Injection](http://www.code-sample.com/2016/04/dependency-injection-in-angular-2.html)

The Angular 2 framework consists of several libraries, the some of them working as core and some are optional.

**5.What is Typings in Angular 2?**

“Typings is the simple way to manage and install TypeScript definitions.”

OR

“The typings are managed by a type definition package manager called typings”.

1.      Typings allows the TypeScript compiler to use existing classes, properties, and so on…

2.      We can also install typings from a repository using the typings command.

3.      In the third-party libraries, does not ship with their own type definitions. The typings package manager is used to install those type definitions into project.

To define a typings file as,

export declare class User {

    UserName: String;

    constructor(UserName?: String);

}

Two Types of typing,

1.      Static typing

2.      Dynamic typing

**Angular 2 Dynamic vs. Static Typing :-**

Angular 2 applications can be written both in Typescript and in plain old JavaScript. This means we have an important choice to make: whether to go with static typing or choose the dynamically typed path.

**What’s the difference between statically and dynamically typed languages?**

**Static Typing:-**

Static typing requires that all variables and function return values be typed and TypeScript is an examples of statically typed languages.

For example as,

public sumOfTwoNumbers(num1: number, num2: number): number{

    return num1 + num2;

}

This “**sumOfTwoNumbers**()” function takes two numeric (or int) values as arguments and returns a number.

**Dynamically Typing :-**

Dynamically typing does not require explicit type declaration. Python and JavaScript are best examples of dynamically typed languages.

The above example looks like as,

public sumOfTwoNumbers(num1, num2){

    return num1 + num2;

}

After calling this method “**sumOfTwoNumbers**(12, "34");” The output is **1234**. There is "No error", "No warning" and nothing else.

**6. Why type definition (.d.ts) with Typescript in Angular2?**

A TypeScript definition file contains the type information for written in JavaScript code and the “JavaScript” does not contain type information itself, so “TypeScript” cannot retrieve that information. To solve this problem, we will use the type definition with TypeScript.

**7.Can I use Angular 2 with Typings in Angular 2?**

Yes!, “TSD” and “Typings” can play together but it is not recommended and both output in the same directory but it will not conflict.

No!, you do not need “Typings” or “TSD” to use TypeScript.  It is only needed for the typings of libraries not in TypeScript and Angular 2 already has typings in the “node\_modules”.

**8.Are Typescript type definitions Required?**

No!, If the module does it properly using the typings property it will work nicely.

**9. Do we only need type definition files not “node\_modules”?**

A Type definition files provide better experience when using “JS libraries” with auto completion and type checking but are ignored otherwise.

If we install and import TypeScript libraries and they will be compiled to “JS” and it’s included in the “JS” build output.

**10.How to create custom type definitions in Angular 2?  
What is Typing?**

Typings is the simple way to manage and install TypeScript definitions. It uses “typings.json”, which can resolve to the Typings Registry, NPM, HTTP and local files.

**Custom Type Definitions:-**

In Angular 2, when we including 3rd party modules. We also need to include the type definition for this module and if they do not provide one within the modules.

If you want to try to install it with typings,

typings install node –save

If we cannot find the type definition in the registry, we can make an extensive definition in this file for now.

For example as,

**declare** module “myModule” {

**export** **function** toDo(value: **string**): **string**;

}

**11. How to load (.d.ts) file in your Angular 2 project?**

If you are using Angular 2, that time you are using some dependencies like zone. If we adding to “zone” in the apps i.e.

**import** “angular2-zone.js/dist-v2/zone”;

**Noted Point** : - You don't change the extension of this file and Just you add the reference of your custom definition inside this file using the triple slash notation i.e.

/// <reference path="./your-custom-file-typing.d.ts">

**12** [**How to import css using System import?**](http://www.code-sample.com/2016/04/import-css-using-system-import.html)

Syntax :

[System.import](http://www.code-sample.com/2016/04/import-css-using-system-import.html)('./app/bootstrap/css/boots-trap.css!').then(() => {

    System.import('./app/main-app.css!');

});

**12. Load external css style into Angular 2 components**

The [styles](http://www.code-sample.com/2016/04/load-external-css-style-into-angular-2.html) or [styleUrls](http://www.code-sample.com/2016/04/load-external-css-style-into-angular-2.html) should only be used for [css rules](http://www.code-sample.com/2016/04/load-external-css-style-into-angular-2.html) and It is affect the style of the template elements.

This is the best approaches to add styles directly to the components and the view encapsulation is set per component. It is use for some situations.

An example to add external styles to components.

@Component({

    selector: 'app',

    templateUrl: 'app/login.html',

    styleUrls: [

        'app/app.css',

        'app/main.css'

    ],

    encapsulation: ViewEncapsulation.None,

})

export class Component {}

**13. Angular2 cookies | angular2 http cookies | angular2 http get set cookie**

Hello everyone, I am going to share the code sample for Implementation of Angular2 cookies service and the detail as below.

**Table of Contents:-**

Installation :-

To install ng2-cookies library, run the below given code. i.e.

              $npm install ng2-cookies

You can Include angular2-cookie library for the same which has given below.

<script src="~/cookie/angular2-cookie.min.js"></script>

In Angular 2 CookieService, 7 methods are available but 6 methods are also available in Angular 1. Only one method is added in Angular 2 that is removeAll() method.

The CookieService methods as given below.

1.       [get()](http://www.code-sample.com/2016/04/angular-2-cookies-npm-example.html)

2.       getObject()

3.       getAll()

4.       put()

5.       putObject()

6.       remove()

7.       removeAll()

1. **get()** :- This method is returns the value of given cookie key.
2. **getObject()** :- This method is returns the desterilized value of given cookie key.
3. **getAll()** :- This method is returns a key value object with all the cookies.
4. **put()** :- This method is use to set a value for given cookie key.
5. **putObject()** :- This method is use to serializes and set a value for given cookie key.
6. **remove()** :-This method is use to remove given cookie.
7. **removeAll()** :-This method is use to remove all cookies.

Use of Angular 2 cookies, the example in detail as give below.

import {Component} from 'angular2/core';

import {Cookie} from 'angular2-cookie/core';

@Component({

    selector: 'my-cookie-app',

    template: '<div>Cookies in Angular 2</div>',

    providers: [Cookie]

})

export class App\_Component {

    constructor(private \_cookie:Cookie){}

    getCookie(key: string){

        return this.\_cookie.get(key);

    }

    getCookieObject(key: string){

        return this.\_cookie.getObject(key);

    }

}

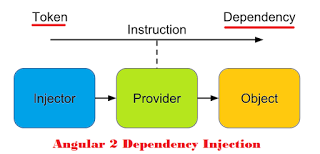
//And other are available methods [put(), putObject(), remove() and removeAll()] wok similar like above methods.

The github example code as,

<https://github.com/salemdar/angular2-cookie>

<https://www.npmjs.com/package/ng2-cookies>

**14. Dependency Injection (DI) in Angular 2**

[](https://2.bp.blogspot.com/-TK9PPhNi4nE/VyGveZ_AL8I/AAAAAAAAOMI/4x0j_TX6-EQLsVugbL0l_qy6UR1OFlPOwCLcB/s1600/Dependency+Injection+in+Angular+2.png)

[Angular 2 Dependency Injection](http://www.code-sample.com/2016/04/dependency-injection-in-angular-2.html) consists of three things.

              1.       Injector

              2.       Provider

              3.       Dependency

**Injector** :- The injector object use to create instances of dependencies.

**Provider** :- A provider is help to injector for create an instance of a dependency. A provider takes a token and maps that to a factory function that creates an object.

**Dependency** :- A dependency is the type of which an object should be created.

For more detail, go to below link

<http://blog.thoughtram.io/angular/2015/05/18/dependency-injection-in-angular-2.html>

<http://www.talkingdotnet.com/difference-between-angular-1-x-and-angular-2/>

**Difference between Angular 1.x and Angular 2**

[**Angular 2**](https://angular.io/) is still in beta (at the time of writing this post), but it has already created a buzz in community. Angular 2 will be a huge learning curve for developers. It is written entirely in [Typescript](http://www.talkingdotnet.com/typescript-interview-questions/) and meets the ES6 specification. And it’s not an update for Angular 1.x. As it’s rewritten and includes breaking changes. So the best way to learn is to compare with Angular 1.x and find out what’s new in Angular 2. In this post, find out difference between Angular 1.x and Angular 2.

**Difference between Angular 1.x and Angular 2**

Angular 1.x was not built with **mobile support** in mind, where Angular 2 is mobile oriented.

Angular 2 provides **more choice for languages**. You can use any of the language from ES5, ES6, TypeScript or Dart to write Angular 2 code. Where, Angular 1.x has ES5, ES6 and Dart. Addition of TypeScript is a great step as TypeScript is awesome way to write JavaScript. Read [TypeScript Interview Questions](http://www.talkingdotnet.com/typescript-interview-questions/) to start with TypeScript.

AngularJS 1.x is easy to setup. All you need to do is to add reference of the library and you are good to go. Where AngularJS 2 is dependent of other libraries and it requires some efforts to set up it. Read [this](http://www.talkingdotnet.com/bind-select-dropdown-list-in-angular-js-2/) post to find out how to configure AngularJS 2.  
**[UPDATE: Angular Team heard this and they introduced**[**Angular CLI to create AngularJS 2 applications**](http://www.talkingdotnet.com/angular-cli-for-angularjs-2/)**.]**

Angular 1.x **controllers and $scope are gone**. We can say that controllers are replaced with “**Components**” in Angular 2. Angular 2 is component based.  
**Angular 1.x Controller**

|  |  |
| --- | --- |
|  | varmyApp = angular     .module("myModule", [])     .controller("productController", function($scope) {       varprods = { name: "Prod1", quantity: 1 };       $scope.products = prods;  }); |

**Angular 2 Components using TypeScript**

|  |  |
| --- | --- |
|  | import{ Component } from 'angular2/core';  @Component({    selector: 'prodsdata',    template: `      <h3>{{prods.name}}</h3> `  })    exportclassProductComponent {    prods = {  name: 'Prod1', quantity: 1 };  } |

Notice, there is a class with export keyword, @Component annotation (that’s also new in Angular 2). The @Component annotation adds the metadata to the class.

Angular 1.x has 2 ways to bootstrap angular. One using ng-app attribute and other via code.

|  |  |
| --- | --- |
|  | <script>     angular.element(document).ready(function() {        angular.bootstrap(document, ['myApp']);     });  </script> |

In Angular 2, say goodbye to ng-app. The only way to bootstrap angular is via code.

|  |  |
| --- | --- |
|  | import{ bootstrap } from 'angular2/platform/browser';  import{ ProductComponent } from './product.component';  bootstrap(ProductComponent); |

The bootstrap function is used and it takes starting component which is also parent component of your angular application.

**Structural directives** syntax is changed. ng-repeat is replaced with \*ngFor.  
Angular 1.x structural directives

|  |  |
| --- | --- |
|  | <ul>     <li ng-repeat="technology in technologies">       {{technology.name}}     </li>  </ul>  <div ng-if="technologies.length">     <h3>You have {{technologies.length}} technologies.</h3>  </div> |

Angular 2 structural directives

|  |  |
| --- | --- |
|  | <ul>    <li \*ngFor="#technology of technologies">      {{technology.name}}    </li>  </ul>  <div \*ngIf="technologies.length">    <h3>You have {{technologies.length}} technologies.</h3>  </div> |

Notice that Asterisk(\*) sign is used as prefix for structural directives, in is replaced with of and camelCase syntax is used.

**UPDATE: In AngularJS 2 version “2.0.0-beta.17”, there is a small change with respect to \*ngFor. Instead of “#” use “let”.** See the sample application [here](http://www.talkingdotnet.com/cascading-dropdown-list-aspnet-core-web-api-angularjs-2/).

In Angular 2, **local variables** are defined using hash(#) prefix (see above code for \*ngFor).

To filter output in our templates in Angular 1.x, we use the pipe character (|) and one or more **filters**. Where in Angular 2 they are called **pipes**. The syntax remains same.

Angular 2 **uses camelCase syntax** for built-in directives. For example, ng-class is now ngClass and ng-model is now ngModel.

One of the major change in Angular 2 is, that it directly uses the valid HTML DOM element properties and events. Due to this, many of the available built-in directives in Angular 1.x are now no longer required. Like, ng-href, ng-src, ng-show and ng-hide. Angular 2 uses href, src and hiddenproperties to get the same output. And same goes with event based directives like ng-click and ng-blur.

|  |  |
| --- | --- |
|  | <button ng-click="doSomething()"> |

And in Angular 2, take the HTML event and wrap it with parentheses.

|  |  |
| --- | --- |
|  | <button (click)="doSomething()"> |

Visit [Bind Select DropDown List in Angular JS 2](http://www.talkingdotnet.com/bind-select-dropdown-list-in-angular-js-2/) to see all these features in action.

In Angular 1.x, ng-bind is used for **one way data binding**, but with Angular 2 it is replaced with [property], where property is valid HTML DOM element property.  
Angular 1.x, one way data binding

|  |  |
| --- | --- |
|  | <input ng-bind="technology.name"></input> |

Angular 2, one way data binding is achieved via wrapping the properties with square brackets.

|  |  |
| --- | --- |
|  | <input [value]="technology.name"></input>  <div [style.color]="color">Some text...</div> |

Remember for events, parentheses is used and for properties, square brackets are used. Read [Bind RadioButton List in Angular 2](http://www.talkingdotnet.com/bind-radiobutton-list-in-angular-2/).

In Angular 1.x, ng-model is used for **two way data binding**, but with Angular 2 it is replaced with [(ngModel)].  
Angular 1.x, two way data binding

|  |  |
| --- | --- |
|  | <input ng-model="technology.name"></input> |

In Angular 2,

|  |  |
| --- | --- |
|  | <input [(ngModel)]="technology.name"></input> |

In Angular 1.x, we can define a [service](https://docs.angularjs.org/guide/services) via 5 different ways.

Factory

Service

Provider

Constant

Values

And in Angular 2, class is the only way to define a service.

|  |  |
| --- | --- |
|  | import{ Injectable } from 'angular2/core';  @Injectable()  exportclassTechnologyService {    getTechnologies() {      return[        newtechnology(1, 'Angular'),        newtechnology(2, 'jQuery',        newtechnology(3, 'Node'),        newtechnology(4, 'Knockout')      ];    }  } |

And once defined, you need to register it with your main component usingprovider. Read [Cascading DropDown List using AngularJS 2](http://www.talkingdotnet.com/cascading-dropdown-select-list-using-angular-js-2/) for working demo of AngularJS 2 service.

|  |  |
| --- | --- |
|  | import{ Component } from 'angular2/core';  import{ TechnologyService } from './character.service';    @Component({    selector: 'my-app',    template: '<technology-list></technology-list>',    providers: [TechnologyService]  })  exportclassAppComponent {} |

One of the advantage of Angular is **Dependency Injection**. With Angular 2 DI is there but now there is a different way to inject dependencies. As everything is class in Angular, so DI is achieve via constructor.

|  |  |
| --- | --- |
|  | varmyApp = angular     .module("myModule", [])     .controller("productController", function($scope, $http) {       varprods = { name: "Prod1", quantity: 1 };       $scope.products = prods;  }); |

In Angular 2,

|  |  |
| --- | --- |
|  | import{ Injectable } from 'angular2/core';    @Injectable()  exportclassTechnologyService {    constructor(private\_http: Http) { }      getTechnologies() {      return[        newtechnology(1, 'Angular'),        newtechnology(2, 'jQuery',        newtechnology(3, 'Node'),        newtechnology(4, 'Knockout')      ];    }  } |

Notice, @Injectable() is added to service class. It is similar to Angluar 1.x$inject used for DI.

In Angular 1.x, we use $routeProvider.when() to configuring routing. Where in Angular 2, @RouteConfig{(...}) is used. ng-view present in Angular 1.x is replaced with <router-outlet>  
In Angular 1.x,

|  |  |
| --- | --- |
|  | varapp = angular          .module("MyModule", ["ngRoute"])          .config(function($routeProvider) {              $routeProvider                .when("/home", { templateUrl: "home.html", controller: "homeController"})                .when("/technology", { templateUrl: "technology.html", controller: "technologyController"})          })         .controller("homeController", function($scope) {              $scope.message = "Home Page";          })         .controller("technologyController", function($scope) {               $scope.technologies = ["ASP.NET", "jQuery", "AngularJS", "JavaScript"];         }) |

In Angular 2,

|  |  |
| --- | --- |
|  | import{ Component } from 'angular2/core';  import{ RouteConfig, ROUTER\_DIRECTIVES, ROUTER\_PROVIDERS } from 'angular2/router';  import{ TechnologyComponent } from './technology/technology.component';  import{ TechnologyService } from './Technology/Technology.service';    @Component({    selector: 'my-app',    templateUrl: 'app/app.component.html',    directives: [ROUTER\_DIRECTIVES],    providers: [      ROUTER\_PROVIDERS,      TechnologyService    ]  })  @RouteConfig([    { path: '/home', name: 'Home', component: HomeComponent, useAsDefault: true},    { path: '/technology', name: 'Technology', component: TechnologyComponent },  ])  exportclassAppComponent { } |

Routing is a separate module that’s why need to import it. And 2 more configurations needs to be to make routing work, one is adding[ROUTER\_DIRECTIVES] as directive and other is to add ROUTER\_DIRECTIVES in providers list. And in HTML page,

|  |  |
| --- | --- |
|  | <ul>    <li><a [routerLink]="['Home']"href="">Home</a></li>    <li><a [routerLink]="['Technology']"href="">Technology</a></li>  </ul> |

ng-href is also replaced by [routerLink]

Angular 2 implements **webstandards** like components and it’s provide better performance than Angular 1.

Read my post [Bind Select DropDown List in Angular JS 2](http://www.talkingdotnet.com/bind-select-dropdown-list-in-angular-js-2/) to see all these new changes in action with sample application.