Smita B Kumar

Angular 7

smitabrijesh@gmail.com

https://www.linkedin.com/in/smita-b-kumar-29385815/

https://twitter.com/brijeshsmita

What is Angular?

- ✓Angular is a new version of the AngularJS framework, developed by Google. It comes with a complete rewrite, and various improvements including optimized builds and faster compile times.
- Angular is a TypeScript-based open-source front-end web application platform led by the Angular Team at Google. Angular documentation states, “Angular is a platform that makes it easy to build applications with the web.
- Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges.”

Features of Angular

- CROSS PLATFORM
  - Progressive Web Apps: Use modern web platform capabilities to deliver app-like experiences. High performance, offline, and zero-step installation.
  - Native: Build native mobile apps with strategies from Cordova, Ionic, or NativeScript.
  - Desktop: Create desktop-installed apps across Mac, Windows, and Linux using the same Angular methods you've learned for the web plus the ability to access native OS APIs.
- SPEED AND PERFORMANCE
  - Code Generation: Angular turns your templates into code that's highly optimized for today's JavaScript virtual machines, giving you all the benefits of hand-written code with the productivity of a framework.
  - Universal: Serve the first view of your application on Node.js®, .NET, PHP, and other servers for near-instant rendering in just HTML and CSS. Also paves the way for sites that optimize for SEO.
  - Code Splitting: Angular apps load quickly with the new Component Router, which delivers automatic code-splitting so users only load code required to render the view they request.

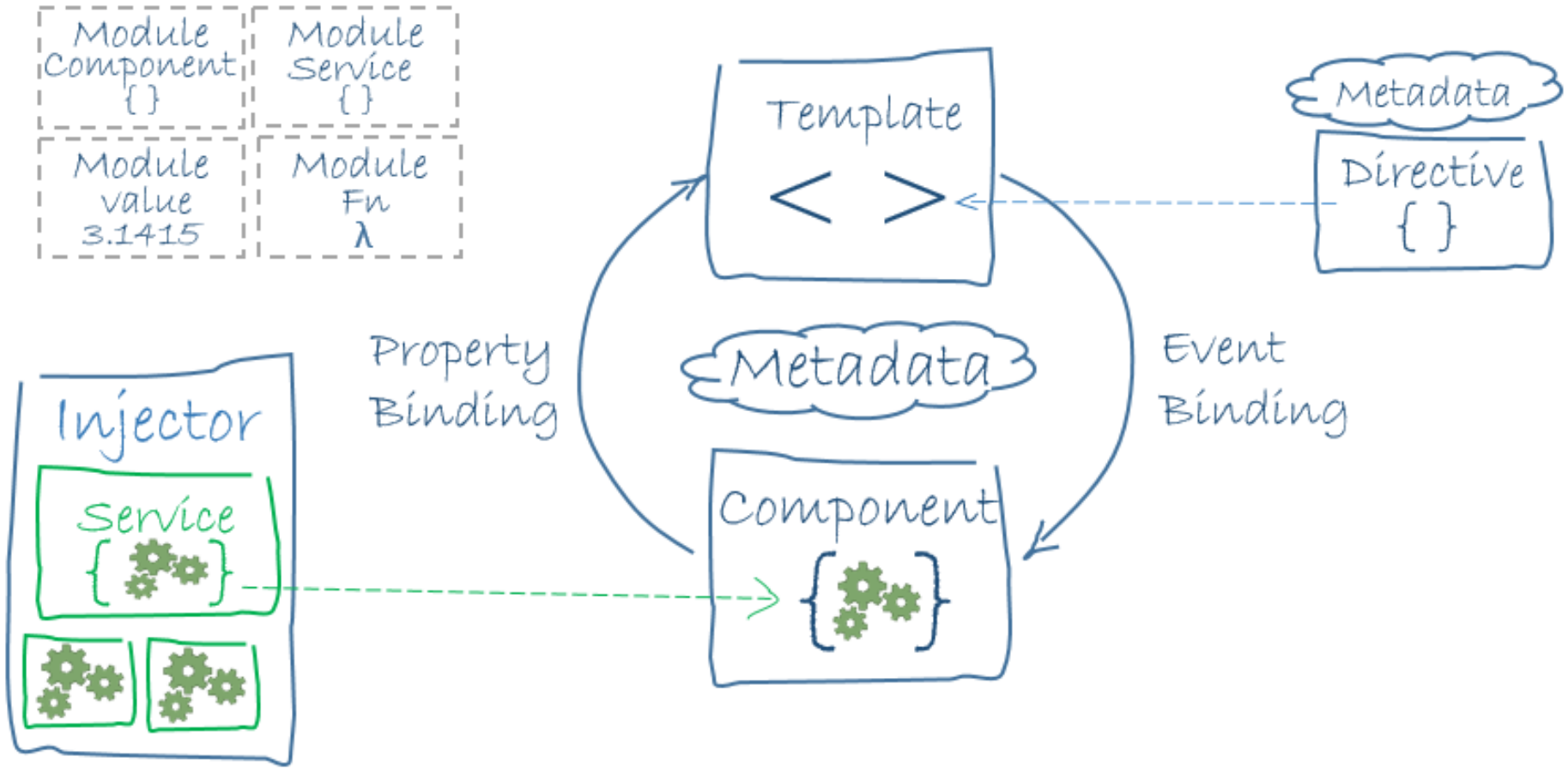
ANGULAR 2

Features of Angular

- PRODUCTIVITY
  - Templates :Quickly create UI views with simple and powerful template syntax.
  - Angular CLI: Command line tools: start building fast, add components and tests, then instantly deploy.
  - IDEs :Get intelligent code completion, instant errors, and other feedback in popular editors and IDEs.
- FULL DEVELOPMENT STORY
  - Testing: With Karma for unit tests, you can know if you've broken things every time you save. And Protractor makes your scenario tests run faster and in a stable manner.
  - Animation: Create high-performance, complex choreographies and animation timelines with very little code through Angular's intuitive API.
  - Accessibility: Create accessible applications with ARIA-enabled components, developer guides, and built-in a11y test infrastructure.

Angular 7

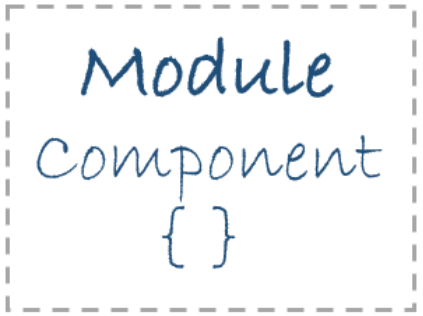
Angular Architecture



Building Blocks of Angular application

- Modules
- Components
- Templates
- Metadata
- Data binding
- Directives
- Services
- Dependency injection
- Pipes

Modules



- Angular apps modularity system is called Angular modules or *ngModules*.
- Every Angular app has at least one module.
- Root module is called *AppModule*.
- Angular module is a class with @NgModule decorator.
- NgModule properties- declarations, exports, imports, providers, bootstrap.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from
'@angular/platform-browser';
@NgModule({
   imports: [BrowserModule],
   providers: [Logger],
   declarations: [AppComponent],
   exports: [AppComponent],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

ANGULAR 2

Bootstrapping

- An NgModule describes how the application parts fit together. Every application has at least one Angular module, the *root* module that you bootstrap to launch the application. By convention, it is usually called AppModule.
- The @NgModule decorator identifies AppModule as an NgModule class. @NgModule takes a metadata object that tells Angular how to compile and launch the application.
  - ***declarations***—this application's lone component.
  - ***imports***—import BrowserModule to have browser specific services such as DOM rendering, sanitization, and location.
  - ***providers***—the service providers.
  - ***bootstrap***—the *root* component that Angular creates and inserts into the index.html host web page.
- The default application created by the Angular CLI only has one component, AppComponent, so it is in both the declarations and the bootstrap arrays.

Components

- Component contributes a View.
- JavaScript class decorated with *@Component*.
- Component contains binding properties and application logic.
- Implements lifecycle methods.

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

@Component({
   selector: '<my-app>',
   templateUrl: 'app/partials/app.html',
   styleUrls:['app/css/app.css']
})
export class AppComponent implements OnInit {

   private count:number;
   @Input()
   private name: string;

   constructor(private service: DataService) { }

   ngOnInit() {
      this.names = this.service.getNames();
   }

   showName(cnt: number) {
      this.count = cnt;
   }
}
```

Templates

- Template is a form of HTML that tells Angular how to render the component.
- Template may contain regular html tags and the following –

Custom directives	-	<item-detail>
Built-in directives	-	*ngFor, *ngIf
Events	-	(click), (dblclick), (focus)
Binding	-	[(name)]
Expression	-	{{ item.name }}

```
<h2>Items List</h2>
<p><i>Pick an item from the list</i></p>
<ul>
    <li *ngFor="let item of items"
        (click)="selectItem(item)">
        {{item.name}}
    </li>
</ul>
<item-detail *ngIf="selectedItem"
             [item]="selectedItem">

</item-detail>
```

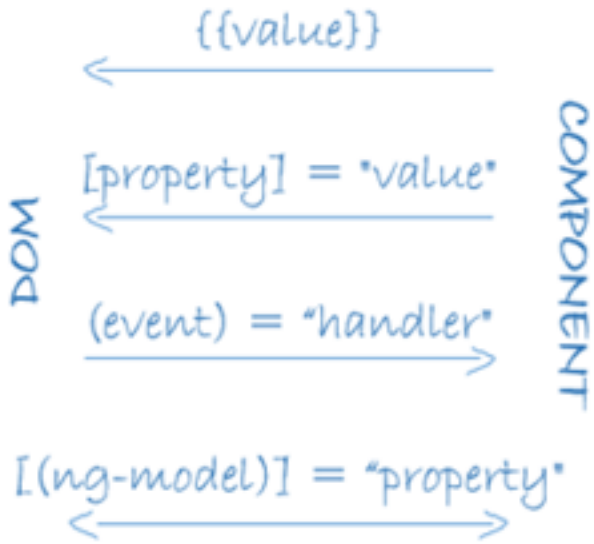
Metadata

- Metadata tells Angular how to process a class.
- In TypeScript, we attach metadata by using a decorator.
- @Component decorator takes a required configuration object to process component and view.
- @Component decorator defines –
  - Selector
  - Template
  - templateUrl
  - Styles
  - styleUrls,
  - Directives,
  - providers etc

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

@Component({
   selector: '<my-app>',
   templateUrl: 'app/partials/app.html',
   styleUrls: ['app/css/app.css']
})
export class AppComponent implements OnInit {
   /* . . . */
}
```

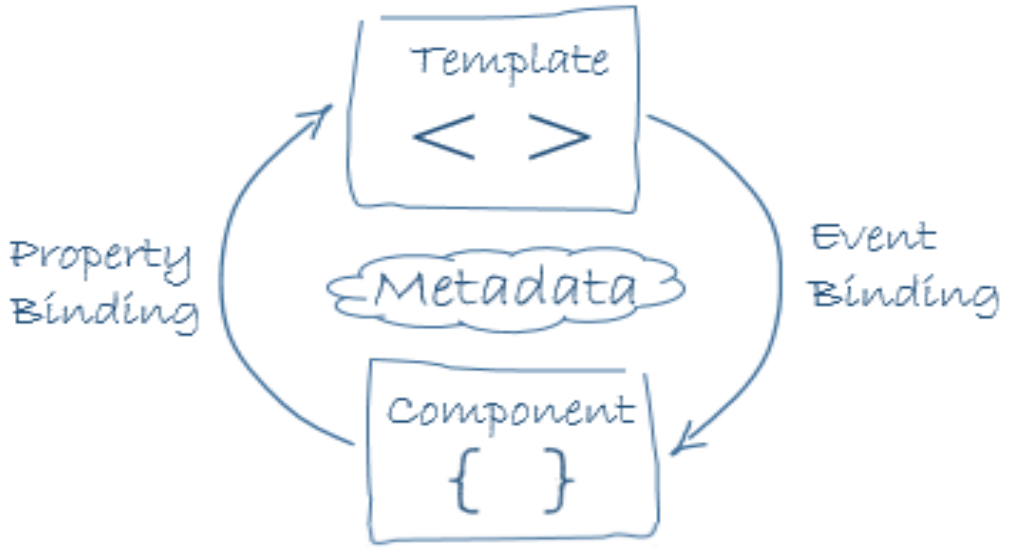
Data binding

- Angular supports data binding.
- Automatic push and pull for interactive UI.
- Binding markup on template HTML tells how to connect template and component.
- Types of binding –
  - {{item.name}} - interpolation
  - [name] - property binding
  - (click) - event binding
  - [(ngModel] – two way binding

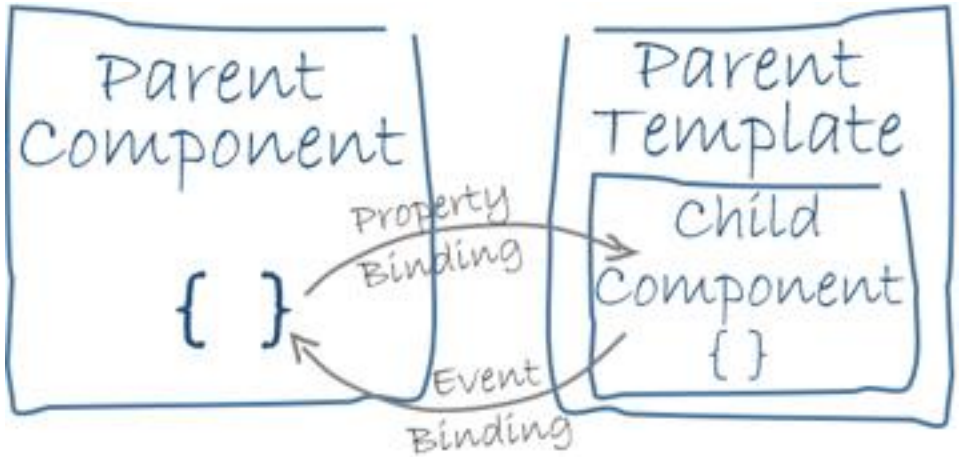



```
<div>
    <H3>{{item.name}}</H3>

    <item-detail [item]="selectedItem"></item-detail>

    <ul>
        <li (click)="selectItem(item)"></li>
    </ul>

    <input [(ngModel)]="item.name">
</div>
```

Two-way data binding



- *Two-way data binding (used mainly in [template-driven forms](#)) combines property and event binding in a single notation.*
- *In two-way binding, a data property value flows to the input box from the component as with property binding. The user's changes also flow back to the component, resetting the property to the latest value, as with event binding.*



Angular processes all data bindings once for each JavaScript event cycle, from the root of the application component tree through all child components.
Data binding plays an important role in communication between a template and its component, and is also important for communication between parent and child components.

Pipes

- Angular pipes let you declare display-value transformations in your template HTML. A class with the @Pipe decorator defines a function that transforms input values to output values for display in a view.
- Angular defines various pipes, such as the date pipe and currency pipe; for a complete list, see the Pipes API list. You can also define new pipes.
- To specify a value transformation in an HTML template, use the pipe operator (|).

  {{interpolated_value | pipe_name}}

  You can chain pipes, sending the output of one pipe function to be transformed by another pipe function. A pipe can also take arguments that control how it performs its transformation. For example, you can pass the desired format to the date pipe.

```
<!-- Default format: output 'Jun 15, 2015'-->
<p>Today is {{today | date}}</p>

<!-- fullDate format: output 'Monday, June 15, 2015'-->
<p>The date is {{today | date:'fullDate'}}</p>

<!-- shortTime format: output '9:43 AM'-->
<p>The time is {{today | date:'shortTime'}}</p>
```

Directives

- Angular templates are rendered according to the instructions of directives.
- A directive is a class with directive metadata.
- @Directive decorator to attach metadata to the class.
- A component is a directive-with-a-template.
- @Component decorator is a @Directive decorator extended with template-oriented features.
- Two kinds of directives – *structural* and *attribute* directives
  - Structural - alter layout by adding, removing, and replacing elements in DOM.

    Eg: *ngFor, *ngIf, <item-detail>
  - Attribute - alter the appearance or behavior of an existing element.

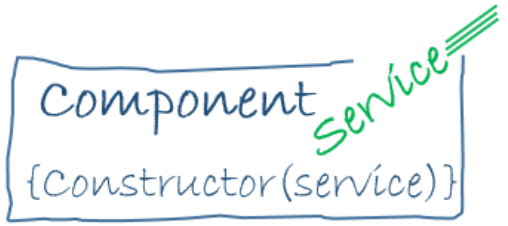
    Eg: ngModel

Services

- Angular services are injectable reusable objects.
- Services can be injected to Components.

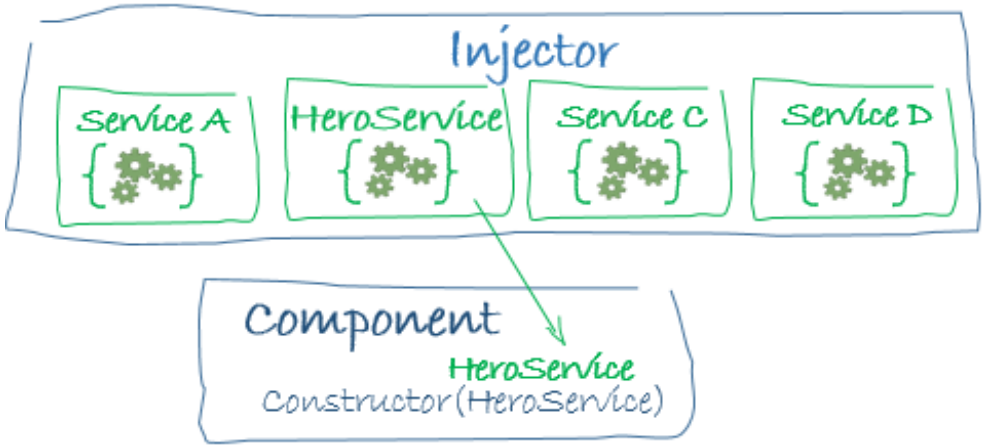
```
export class DataService {
   private items: Item[] = [];

   constructor(
      private backend: BackendService,
      private logger: Logger) { }

   getItems() {
      this.backend.getAll(Item)
         .then((items: Item[]) => {
            this.logger.log(`Fetched ${items.length}
items.`);
            this.items.push(...items); // fill cache
         });
      return this.items;
   }
}
```

Dependency injection



- Is a way to supply a new instance of a class with the fully-formed dependencies it requires.

```
constructor(private service: HeroService) { }
```




```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-
browser';
@NgModule({
   imports: [BrowserModule],
   providers: [
      BackendService,
      HeroService,
      Logger
   ],
   declarations: [AppComponent],
   exports: [AppComponent],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

Register Services to be injected

Tools and techniques

- Responsive programming
  - Lifecycle hooks: Tap into key moments in the lifetime of a component, from its creation to its destruction, by implementing the lifecycle hook interfaces.
  - Observables and event processing: How to use observables with components and services to publish and subscribe to messages of any type, such as user-interaction events and asynchronous operation results.
- Client-server interaction
  - HTTP: Communicate with a server to get data, save data, and invoke server-side actions with an HTTP client.
  - Server-side Rendering: Angular Universal generates static application pages on the server through server-side rendering (SSR). This allows you to run your Angular app on the server in order to improve performance and show the first page quickly on mobile and low-powered devices, and also facilitate web crawlers.
  - Service Workers: Use a service worker to reduce dependency on the network significantly improving the user experience.
- Domain-specific libraries
  - Animations: Use Angular's animation library to animate component behavior without deep knowledge of animation techniques or CSS.
  - Forms: Support complex data entry scenarios with HTML-based validation and dirty checking.

Tools and techniques

- Support for the development cycle
  - Compilation: Angular provides just-in-time (JIT) compilation for the development environment, and ahead-of-time (AOT) compilation for the production environment.
  - Testing platform: Run unit tests on your application parts as they interact with the Angular framework.
  - Internationalization: Make your app available in multiple languages with Angular's internationalization (i18n) tools.
  - Security guidelines: Learn about Angular's built-in protections against common web-app vulnerabilities and attacks such as cross-site scripting attacks.

Tools and techniques

- Setup, build, and deployment configuration
  - CLI Command Reference: The Angular CLI is a command-line tool that you use to create projects, generate application and library code, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.
  - Workspace and File Structure: Understand the structure of Angular workspace and project folders.
  - npm Packages: The Angular Framework, Angular CLI, and components used by Angular applications are packaged as npm packages and distributed via the npm registry. The Angular CLI creates a default package.json file, which specifies a starter set of packages that work well together and jointly support many common application scenarios.
  - TypeScript configuration: TypeScript is the primary language for Angular application development.
  - Browser support: Make your apps compatible across a wide range of browsers.
  - Building and Serving: Learn to define different build and proxy server configurations for your project, such as development, staging, and production.
  - Deployment: Learn techniques for deploying your Angular application to a remote server.

Angular Prerequsites

- Node.JS v4.x.x or later
- NPM v3.x.x or later
- WebStorm 2016.2 (Recommended) / Adobe Brackets / VS Code
- Chrome Browser / Firefox

Project setup

- Create a project folder (eg: ng7-first)
- Create index.html
- Create the following configuration files
  - package.json
  - tsconfig.json
  - typings.json
  - Systemjs.config.js

package.json

```
{
  "name": "Angular-quickstart",
  "version": "1.0.0",
  "scripts": {
    "start": "tsc && concurrently \"npm run tsc:w\" \"npm run lite\" ",
    "lite": "lite-server",
    "postinstall": "typings install",
    "tsc": "tsc",
    "tsc:w": "tsc -w",
    "typings": "typings"
  },
  "license": "ISC",
  "dependencies": {
    "@angular/common": "2.0.0-rc.5",
    "@angular/compiler": "2.0.0-rc.5",
    "@angular/core": "2.0.0-rc.5",
    "@angular/forms": "0.3.0",
    "@angular/http": "2.0.0-rc.5",
    "systemjs": "0.19.27"
    ........
  },
  "devDependencies": {
    "concurrently": "^2.0.0",
    "jquery": "^3.1.0",
    "lite-server": "^2.2.0",
    "typescript": "^1.8.10",
    "typings": "^1.0.4"
  }
}
```

tsconfig.json

- TypeScript compiler configuration.

```
{
  "compilerOptions": {
      "target": "es5",
      "module": "commonjs",
      "moduleResolution": "node",
      "outDir": "app/js",
      "sourceMap": true,
      "emitDecoratorMetadata": true,
      "experimentalDecorators": true,
      "removeComments": false,
      "noImplicitAny": false
  },
  "exclude": [
      "node_modules",
      "typings/index",
      "typings/index.d.ts"
  ]
}
```

systemjs.config.js

```
(function(global) {
    // map tells the System loader where to look for things
    var map = {
        'app':                          'app', // 'dist',
        '@angular':                     'node_modules/@angular',
        'Angular-in-memory-web-api': 'node_modules/Angular-in-memory-web-api',
        'rxjs':                         'node_modules/rxjs'
    };
    // packages tells the System loader how to load when no filename and/or no extension
    var packages = {
        'app':                          { main: 'js/main.js',  defaultExtension: 'js' },
        'rxjs':                         { defaultExtension: 'js' },
        'Angular-in-memory-web-api': { main: 'index.js', defaultExtension: 'js' },
    };
    var ngPackageNames = [
        'common','compiler','core','forms','http','platform-browser','platform-browser-dynamic',
        'router','router-deprecated','upgrade',
    ];

    var config = {
        map: map,
        packages: packages
    };
    System.config(config);
})(this);
```

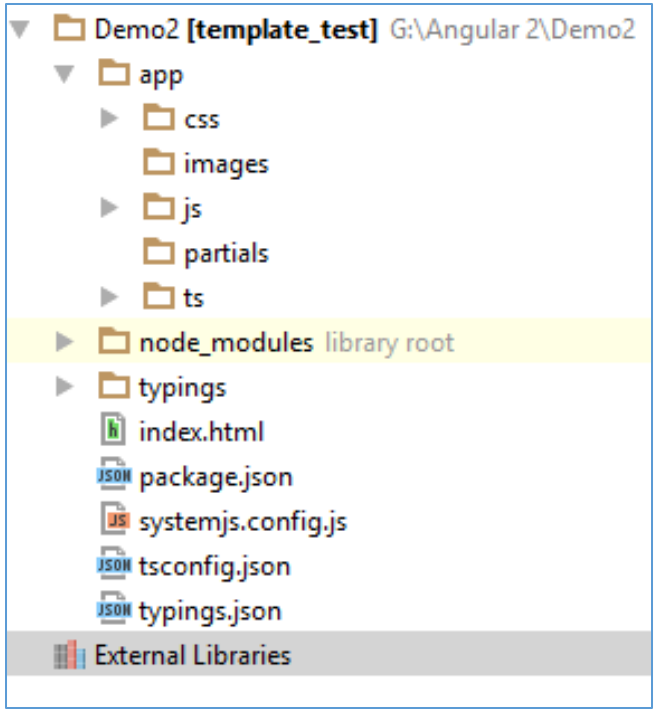
Install packages

- Open command prompt.
- Move to the project folder.
- Execute the npm command to install dependency packages listed in packages.json.

```
C:\ng7-firrst> npm install
```

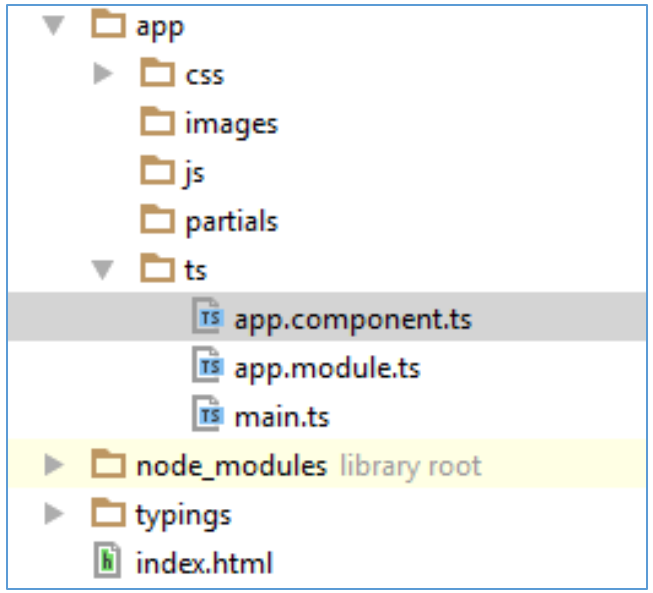
Angular First application

- Create a folder ‘app’ inside the project folder
- Create the following subfolders inside ‘app’ folder.
  - ts- contains TypeScript files
  - js- contains all compiled JavaScript files.
  - css- contains css stylesheets
  - Images- contains images for web site.
  - Partials- contains partial html pages
- Create an index.html in project folder.



TypeScript Files

- Create the following TypeScript files in ‘ts’ folder.
  - app.component.ts
    - Root of the application
    - Conventionally called ‘AppComponent’
  - app.module.ts
    - Compose angular components in to blocks
    - Every application has atleast one module
  - main.ts
    - Bootstrapping angular application



Pipes

- Display only some filtered elements from an array.
- Modify or format the value.
- Use them as a function.
- Multiple pipes can be combined.

```
myValue | myPipe:param1:param2 | mySecondPipe:param1
```

List of Pipes

Filter Name	Angular 1.x	Angular
currency	Supported	Supported
date	Supported	Supported
uppercase	Supported	Supported
json	Supported	Supported
limitTo	Supported	Not
lowercase	Supported	Supported
number	Supported	Not
orderBy	Supported	Not
filter	Supported	Not
async	Not	Supported
decimal	Not	Supported
percent	Not	Supported

Uppercase and lowercase pipes

- Transforms text to uppercase and lowercase

```
expression | lowercase

expression | uppercase
```

Date pipe

- Formats a date value to a string based on the requested format.

expression | date[:format]

- format can be custom format or one of the following predefined formats

  - 'medium': equivalent to 'yMMMdjms'
  - 'short': equivalent to 'yMdjm'
  - 'fullDate': equivalent to 'yMMMMEEEEd'
  - 'longDate': equivalent to 'yMMMMd'
  - 'mediumDate': equivalent to 'yMMMd'
  - 'shortDate': equivalent to 'yMd'
  - 'mediumTime': equivalent to 'jms'
  - 'shortTime': equivalent to 'jm'

Async pipe

- async pipe subscribes to an Observable or Promise and returns the latest value it has emitted.
- When a new value is emitted, the async pipe marks the component to be checked for changes.

```
item | async
```

Decimal (Number) pipe

- Formats a number as local text. i.e. group sizing and separator and other locale-specific configurations are based on the active locale.
- It uses the 'number' keyword to apply filter.

expression | number[:digitInfo]

digitInfo has the following format

{minIntegerDigits}.{minFractionDigits}-{maxFractionDigits}

- minIntegerDigits is the minimum number of integer digits to use. Defaults to 1.
- minFractionDigits is the minimum number of digits after fraction. Defaults to 0.
- maxFractionDigits is the maximum number of digits after fraction. Defaults to 3.

Note: This pipe uses the Internationalization API. Therefore it is only reliable in Chrome and Opera browsers.

Currency pipe

- Formats a number as local currency.

expression | currency[:currencyCode[:symbolDisplay[:digitInfo]]]

- **CurrencyCode** is the ISO 4217 currency code, such as "USD" for the US dollar and "EUR" for the euro.
- **symbolDisplay** is a boolean indicating whether to use the currency symbol (e.g. $) or the currency code (e.g. USD) in the output. The default for this value is **false**.
- digitInfo is for decimal places and number of fraction digits.

{minIntegerDigits}.{minFractionDigits}-{maxFractionDigits}

Note: This pipe uses the Internationalization API. Therefore it is only reliable in Chrome and Opera browsers.

Percent pipe

- Formats a number as local percent.

expression | currency [:digitInfo]

- digitInfo is for decimal places and number of fraction digits.

{minIntegerDigits}.{minFractionDigits}-{maxFractionDigits}

Note: This pipe uses the Internationalization API. Therefore it is only reliable in Chrome and Opera browsers.

HTTP And Observables In Angular

- Using HTTP in Angular we are going to fetch data from the web server, We will make a call to the web server that will in return provides data.
- This HTTP request will hit the web API or service that will in return fetch the data from the database and send it back as an HTTP response. This response that we are getting from the HTTP is nothing but the observable.
- Now the observable needs to be cast to the particular type and the Service will cast this observable data to the array of specific object and return it to the subscribed component. HTTP is just a two-way process, first is to send the request and second is to receive the response.
- The web API/service or the database is only responsible to give the data and their job is done.
- **HTTP response is nothing but the Observable returned by HttpClient.get**
- Components subscribe to the service and receive and operate the data accordingly.

Observables

- Observables provide support for passing messages between publishers and subscribers in your application. Observables offer significant benefits over other techniques for event handling, asynchronous programming, and handling multiple values.
- Observables are declarative—that is, you define a function for publishing values, but it is not executed until a consumer subscribes to it. The subscribed consumer then receives notifications until the function completes, or until they unsubscribe.
- Angular makes use of observables as an interface to handle a variety of common asynchronous operations. For example:
  - The EventEmitter class extends Observable.
  - The HTTP module uses observables to handle AJAX requests and responses.
  - The Router and Forms modules use observables to listen for and respond to user-input events.

AngularJS v/s Angular 2

- The first version of Angular was released in the year of 2010. Some people call this as AngularJS and some people call as Angular 1. But it is officially named as AngularJS.
- The Angular 2 was released in the year of 2016. The important point to note is that Angular 2 is not a simple upgrade of AngularJS. Angular 2 is completely rewritten from scratch and as a result, the ways we write in AngularJS and Angular 2 are completely different.
- AngularJS is completely based on *controllers* and the View communicates using *$scope* whereas Angular 2 is completely a component-based approach. In Angular 2, both, *controller* and *$scope* are completely gone. Angular 2 is completely based on the *component*. Components are the building blocks of an Angular 2 application. The advantage of the Component-based approach is that it facilitates greater code reuse.

AngularJS v/s Angular 2

- As compared to AngularJS, in Angular 2, the communication among *components* is very easy. The *components* can be reused anywhere in the application without much effort.
- In Angular 2, from the *unit testing* standpoint, the use of *components* makes the Angular 2 application more testable with minimal effort.
- From a performance standpoint, Angular 2 is 5 times faster compared to AngularJS.
- AngularJS doesn't support for mobile devices, whereas Angular 2 is designed by keeping in mind to support mobile devices.
- Angular 2 has more language choices (TypeScript, JavaScript, Dart, PureScript and Elm etc.)

Angular 2 v/s Angular 4

- Angular 2and Angular 4 will use the same concept and patterns. Angular 4 simply is the next version of Angular 2.
- If you know Angular 2 already then it is not a big deal to learn Angular 4 because the underline concepts are still the same.
- Angular 4 is an inheritance from Angular 2. Angular 4 is simply is the next version of Angular 2with few changes and enhancements.
- In Angular 4, a lot of improvements made to reduce the size of the AOT (Ahead-of-time) compiler generated code.
- In Angular 2 only TypeScript 1.8 version was supported, whereas, in Angular 4, it supports TypeScript 2.1 and TypeScript 2.2 compatibility, which means now we can use all new features supported in TypeScript 2.1 and TypeScript 2.2 can be used in Angular 4 application.
- The Animation features are separated from *@angular/core* package and moved them to new packages. By this way, if we don't import *animation* packages into your application then the main bundle size will be reduced and gives the performance improvement. As I told you *@angular/core* package and moved them to new packages, so if you are using these packages in your Angular 2 application then at the time of moving from Angular 2 to Angular 4, we need to change the package reference as well.
- In Angular 4, *else* block newly introduced. I mean, along with **ngif*, we can use *else* block as well.

What's new in Angular 5 & 6?

- **Progressive web Application:**Angular CLI has the capability to create own code. Configuration and code on its own are the best features of the angular-CLI. Application logo, push notifications are good in AngularJS. The offline capability and the treatment of push notifications are used through the service workers.
- **Build:**The errors in the build time, not the runtime are a very crucial part in the angular 5. The Angular CLI provides the static analysis of AOT mode. The incremental builds are supported by the AOT mode and this aids for the reduction in the built time. The Google closure compiler can be used to get the desired results in the build.
- **Forms:**The validation will be performed within the specified time. Every time a form control value is changed and it helps for the complex validations to be done perfectly. It is easy to mention the validation accurately with the latest version and change the validation also with the help of change, submit or blur.
- **Reactive forms:**The inputs values change as per the time in the application. This is handled through the model-driven approach. This helps to create multiple controls in a group, Validate from values and implement more advanced forms.

What's new in Angular 5 & 6?

- **Changes in the Angular 5**
  - Injection token is added, error handler has been removed, Ngoutletcontext has been removed, the values enabled, disabled are used for the parameter, the values true, false, legacy- enabled, legacy-disabled are removed, ng-container is added, 18n comments are removed, enablelegacytemplate, enablelegacytemplate is removed, @angular/common/http, angular/http module has been deprecated, reflectiveinjector has been removed, static injector has been added, reflective injector is removed, and staticinjector is added in the angularjs.
- **Tools optimized in the Angular 5**
  - Typescript version 2.4, RxJS 5.5, build tool, AOT, Activation Start and activation End in a router are some of the new tools in the angular 5.
- **Angular 6**
  - Angular 6 the new version of Angular JS supports the typescript version 2.7 and makes the coding part easy. The DOM element in the Angular 6 wraps the Angular component into the custom element which pays way to use the angular component in other projects. **Angular 6** comes with new bug fixing features and thus when installing the latest version it is important to uninstall the existing service worker in the Angular JS. The template element is initially depreciated in the version 4 and now removed in the version 6. The formatting function in angular 6 is improved and the currency value will be rounded to the appropriate digit. The third rendering engine has been introduced in the Angular JS called IVY and it increases the speed.

New in Angular 7?

- **Updated Dependencies:** TypeScript 3.2 support , RxJS 6.3, Node 10
- **The Angular Material CDK**—Now with Virtual Scrolling and Drag and Drop
- **Application Performance-**polyfills.ts file will automatically remove during production
- **Improved Accessibility of Selects -**Improve the accessibility of your application by using a native **select**element inside of a mat-form-field
- **Interactive prompts** -routing support, CSS pre-processor
- **Angular compilation options**- you can define compilerOptions for TypeScript and angularCompilerOptions for Angular in your tsconfig.json files
- **Router**-A new warning has been added if you try to trigger a navigation outside of the Angular zone,
- **Deprecations** - <ng-form>