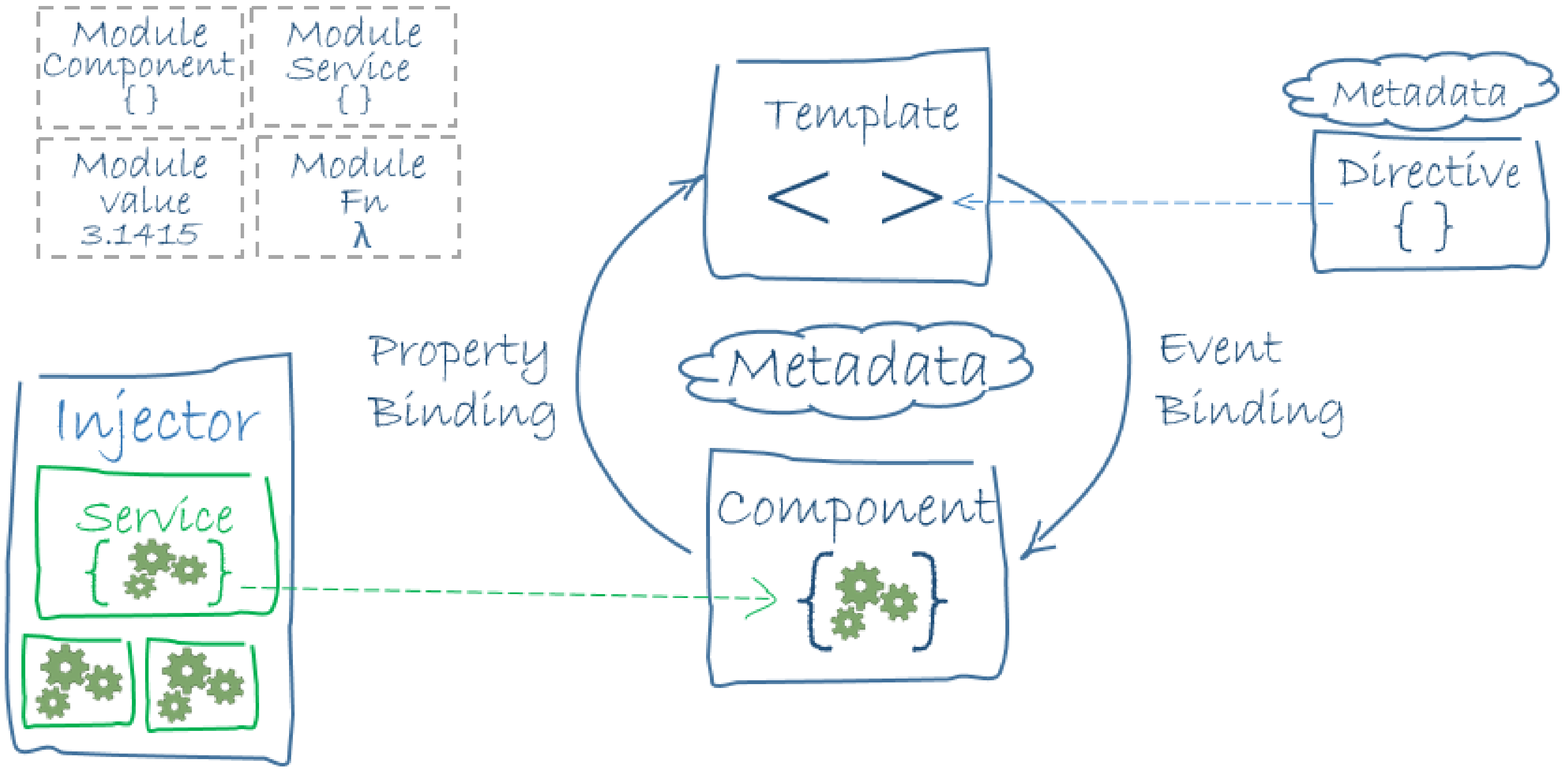
ANGULAR 2

What is Angular 2?

- ✓Angular2 is a framework.
- ✓Can be coded in JS or languages that can be compiled to JS.
- ✓Can be written in JavaScript, Dart or TypeScript.
- ✓Consists core and optional libraries.
- ✓Templates, Components, Pipes, Services and Modules.

ANGULAR 2

Angular2 Architecture



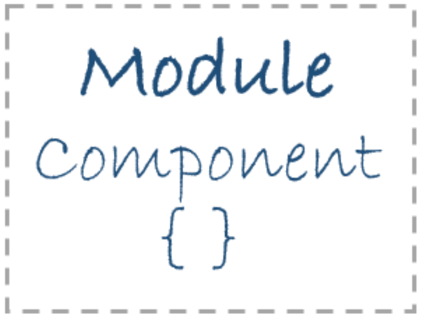
ANGULAR 2

Building Blocks of Angular2 application

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

ANGULAR 2

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 { }
```

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;
   }
}
```

ANGULAR 2

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>
```

ANGULAR 2

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 {
   /* . . . */
}
```

ANGULAR 2

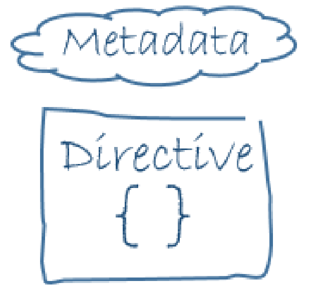
Data binding

- Angular2 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 bidning

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

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

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

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

ANGULAR 2

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

ANGULAR 2

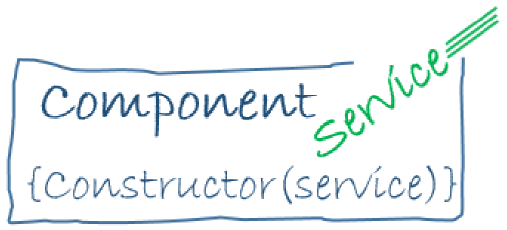
Services

- 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;
   }
}
```

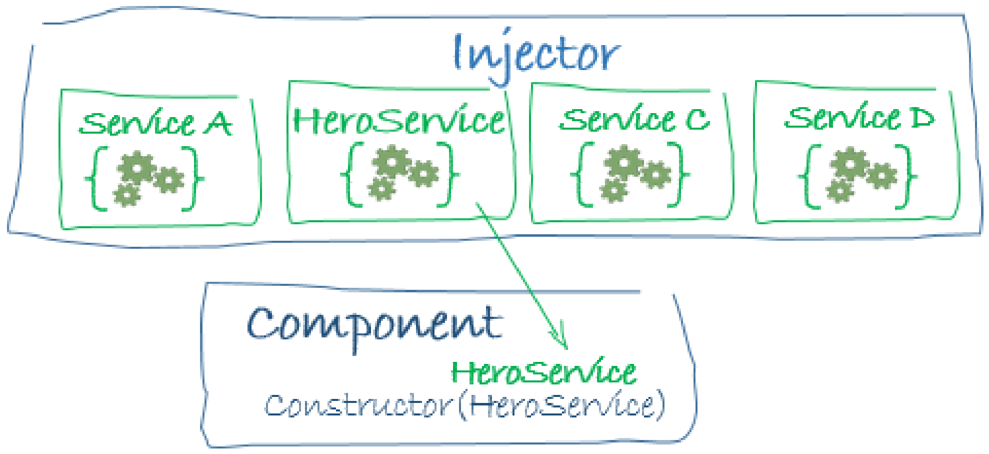
ANGULAR 2

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

ANGULAR 2

Angular2 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

ANGULAR 2

Project setup

- Create a project folder (eg: Angular2-QuickStart)
- Create index.html
- Create the following configuration files
  - package.json
  - tsconfig.json
  - typings.json
  - Systemjs.config.js

ANGULAR 2

package.json

```
{
  "name": "angular2-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"
  }
}
```

ANGULAR 2

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"
  ]
}
```

ANGULAR 2

systemjs.config.js

```
(function(global) {
    // map tells the System loader where to look for things
    var map = {
        'app':                          'app', // 'dist',
        '@angular':                     'node_modules/@angular',
        'angular2-in-memory-web-api': 'node_modules/angular2-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' },
        'angular2-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);
```

ANGULAR 2

Install packages

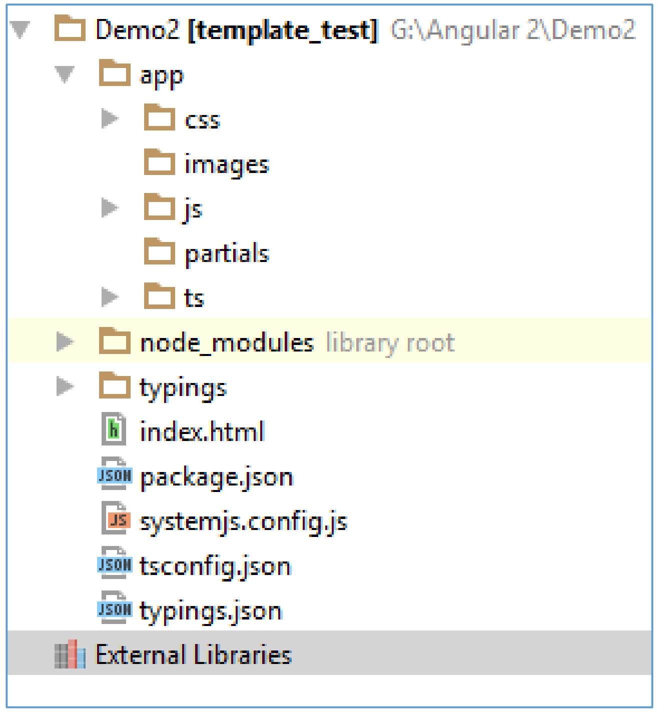
- Open command prompt.
- Move to the project folder.
- Execute the npm command to install dependency packages listed in packages.json.

```
C:\Angular2-quickstart> npm install
```

ANGULAR 2

Angular 2 First application

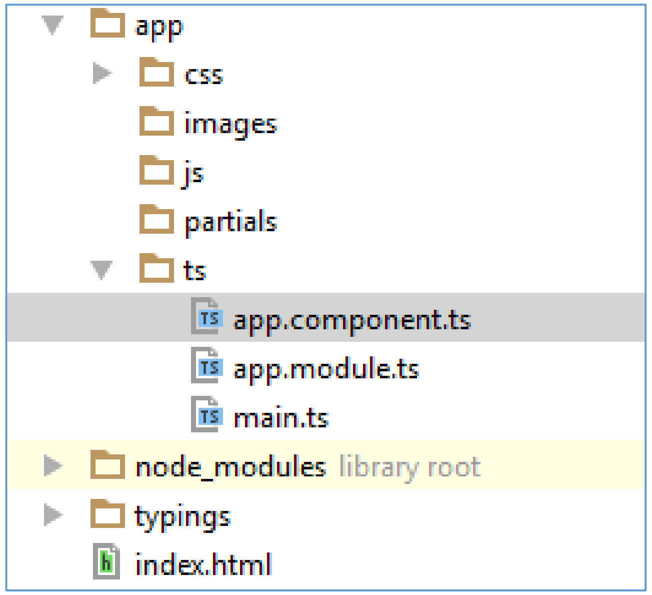
- Create a folder ‘app’ inside the project folder
- Create the following subfolders inside ‘app’ folder.
  - ts- contains TypeScript files
  - js- contains all compiled JavaScript files.
  - css- contains css stylesheets
  - Images- contains images for web site.
  - Partials- contains partial html pages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



```
Demo2 [template_test] G:\Angular 2\Demo2
  app
    css
    images
    js
    partials
    ts
  node_modules library root
  typings
  index.html
  package.json
  systemjs.config.js
  tsconfig.json
  typings.json
External Libraries
```

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



ANGULAR 2

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
```

ANGULAR 2

List of Pipes

Filter Name	Angular 1.x	Angular2
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

ANGULAR 2

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'

ANGULAR 2

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
```

ANGULAR 2

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.

ANGULAR 2

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.

ANGULAR 2

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.

ANGULAR 2