Capgemini

Angular 4

Angular 4 Fundamentals

Angular 4 Introduction

Angular 4 is a JavaScript framework for building web applications and apps in JavaScript, html, and TypeScript, which is a superset of JavaScript.

There are three major releases of Angular. The first version that was released is Angular1, which is also called AngularJS.

Angular1 was followed by Angular2, which came in with a lot of changes when compared to Angular1.

Angular 4 is almost the same as Angular 2. It has a backward compatibility with Angular 2.

Angular 4 Introduction

The structure of Angular is based on the components/services architecture.

AngularJS was based on the model view controller.

Angular 4 released in March 2017 proves to be a major breakthrough and is the latest release from the Angular team after Angular2.

Angular 4 helps us build client applications in HTML and either JavaScript or a language (like Dart or TypeScript) that compiles to JavaScript.

Why Angular

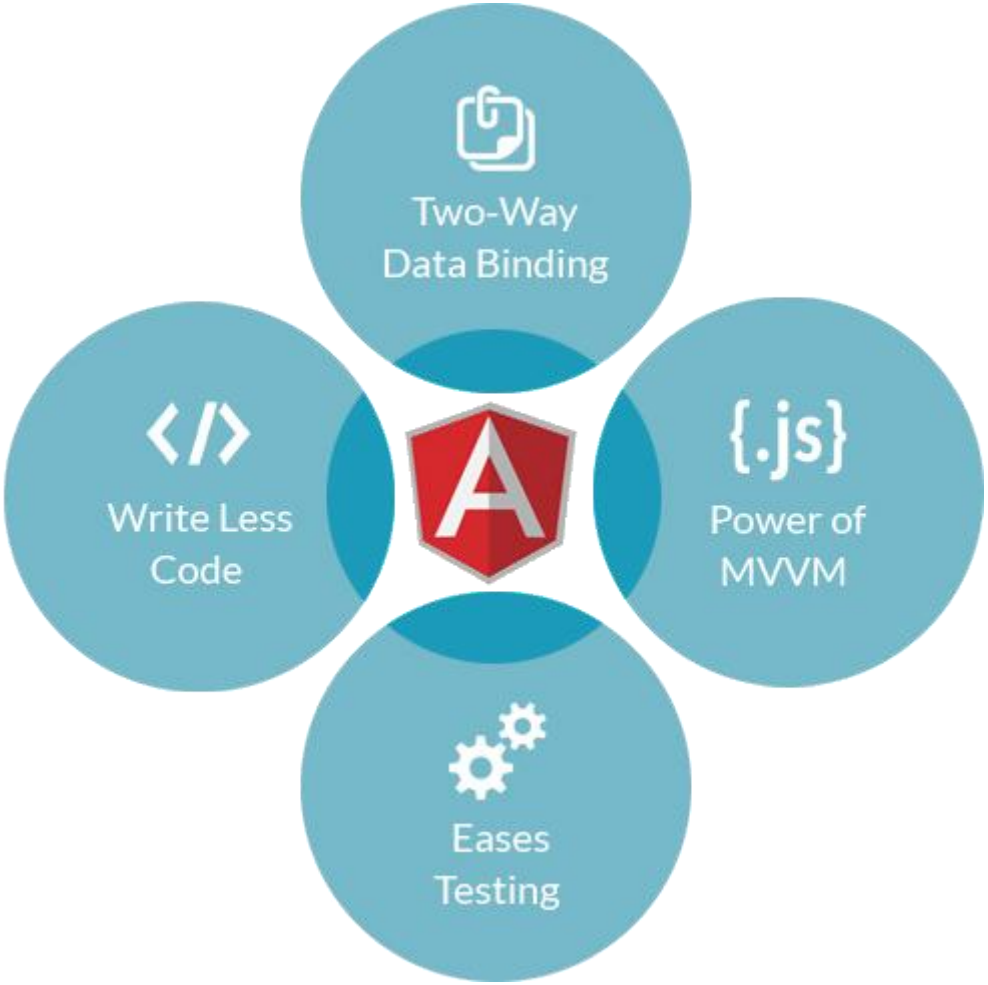
Angular makes HTML more expressive, It powers up HTML with features such as if conditions, for loops and local variables.

Angular has powerful data binding. We can easily display fields from our data model, track changes and process updates from the user.

Angular promotes modularity by design so that the applications become a set of building blocks making it easier to create and reuse contents.

Angular has built-in support for communication with a backend service this makes it easy for Web applications to integrate with the backend service to GET and POST data or execute server side business logic.

Why Angular


Two-Way
Data Binding
Write Less
Code
Power of
MVVM
Eases
Testing

Why Angular 4

Angular 4 was built for speed, It has faster initial loads faster change detection and improved rendering times.

Angular 4 is modern it takes advantage of features provided in the latest JavaScript standards such as classes, modules and decorators.

It leverages web component technologies for building reusable user interface widgets.

It supports both modern and legacy browsers like Chrome, Firefox and Internet Explorer back to IE 9.

It has a simplified API. It has fewer built-in directives to learn simple binding and a lower overall concept count.

It enhances productivity to improve day to day workflows

New features of Angular 4

Angular 4 has been enhanced with the below features. These are additions made to the features of Angular 2.

Ng-if

Angular2 supported only the **if** condition. However, Angular 4 supports the **if else** condition as well. Let us see how it works using the ng-template.

as keyword in for loop

With the help of **as** keyword you can store the value in a for loop

Animation Package

Animation in Angular 4 is available as a separate package and needs to be imported from @angular/animations.

In Angular2, it was available with @**angular/core**. It is still kept the same for its backward compatibility aspect.

New features of Angular 4

Template

Angular 4 uses <ng-template> as the tag instead of <template> which was used in Angular2.

Pipe Title Case

Angular 4 has added a new pipe title case, which changes the first letter of each word into uppercase.

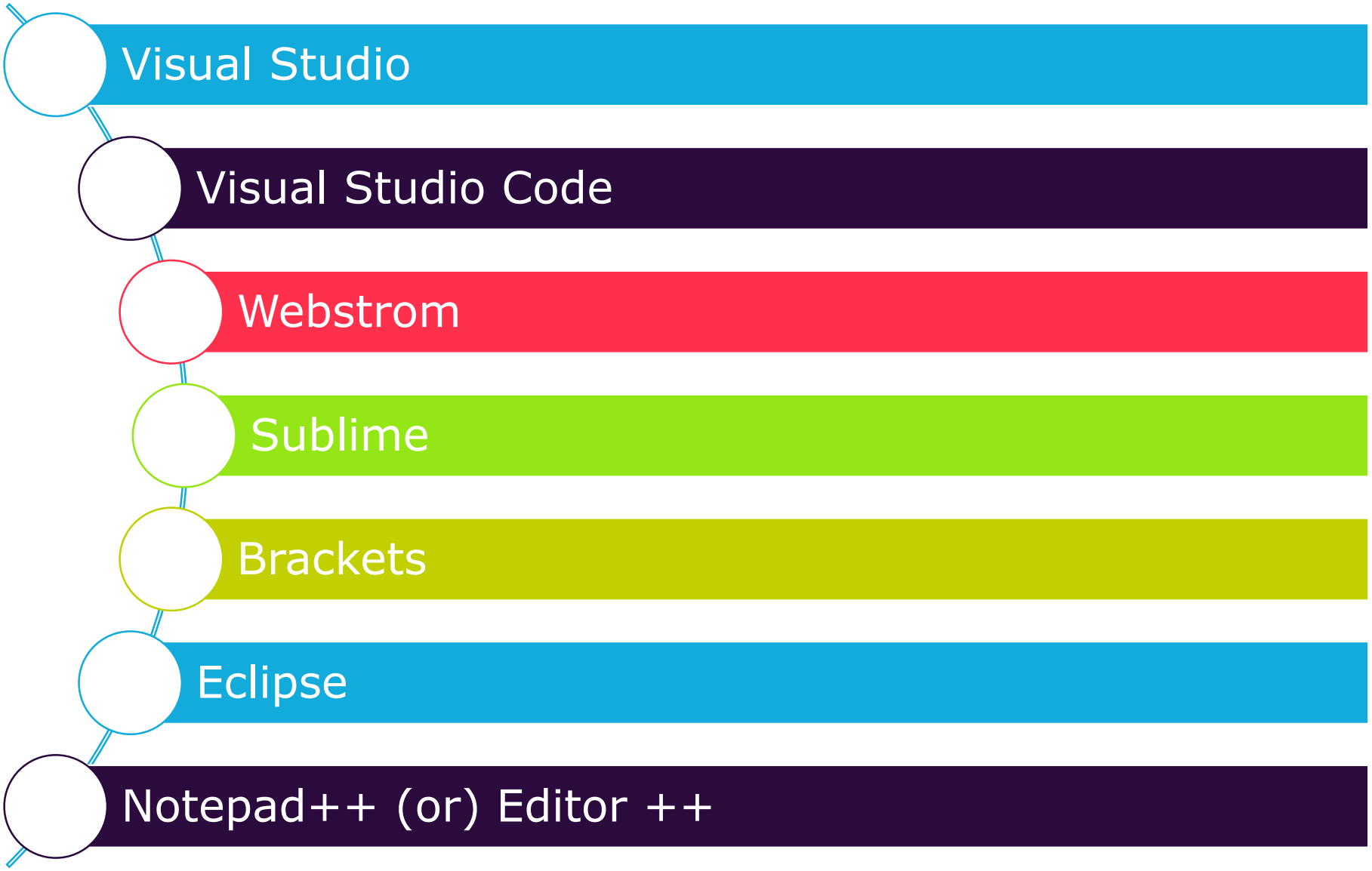
TypeScript 2.2

Angular 4 is updated to a recent version of TypeScript, which is 2.2. This helps improve the speed and gives better type checking in the project.

Smaller and Faster Apps

Angular 4 applications are smaller and faster when compared to Angular2.

Angular 4 Code Editors



Language Choice

ES5

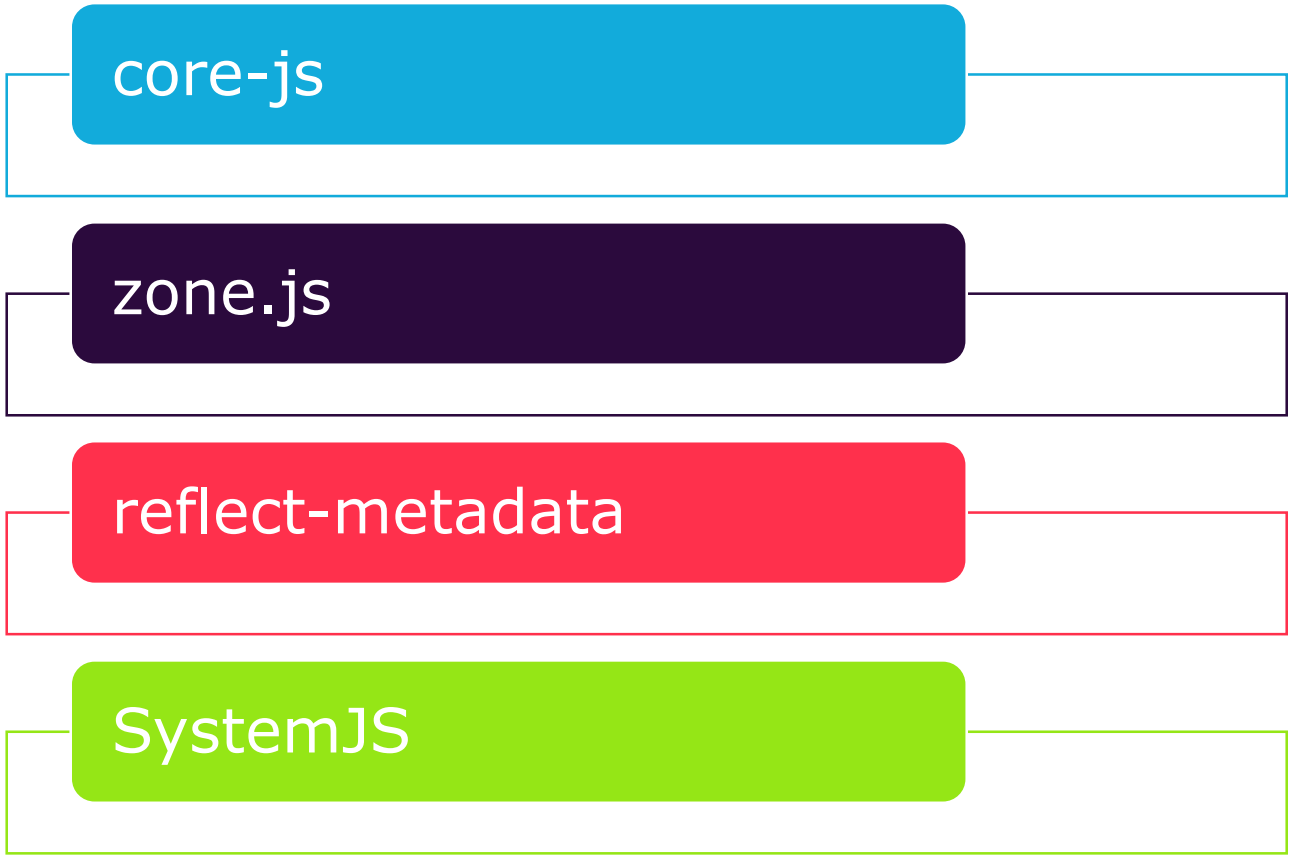
ES6

TypeScript

Dart

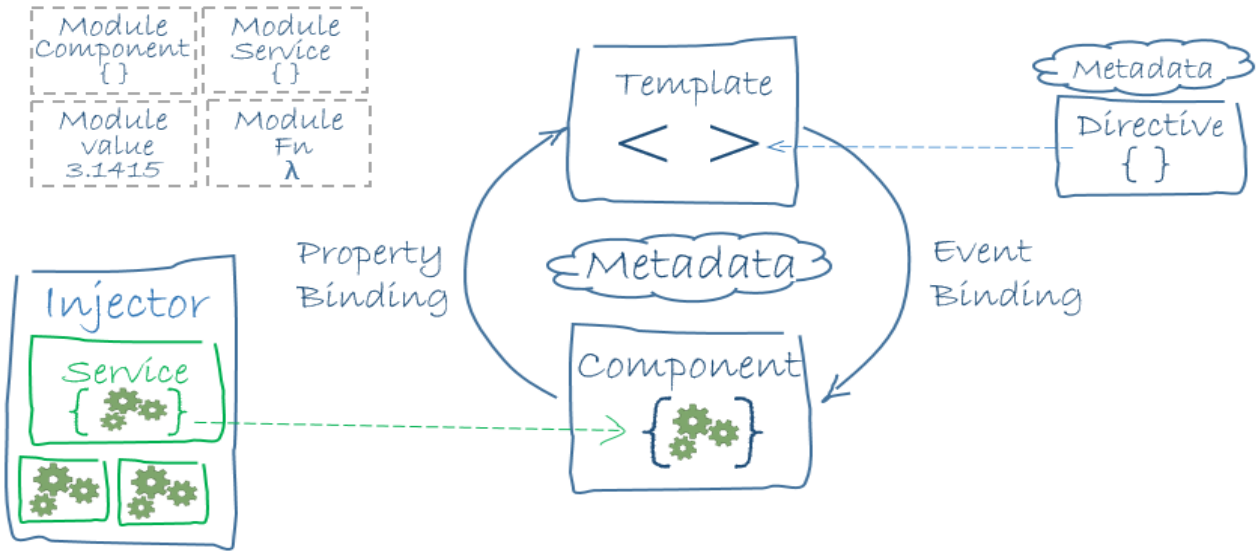
Angular 4 Dependencies

To run Angular 4, we depend on these four libraries:.



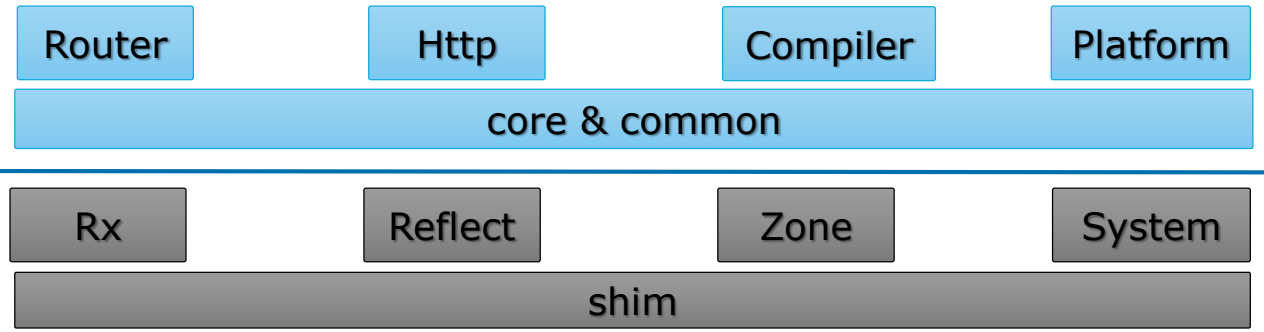
Building Blocks of an Angular 4

Application



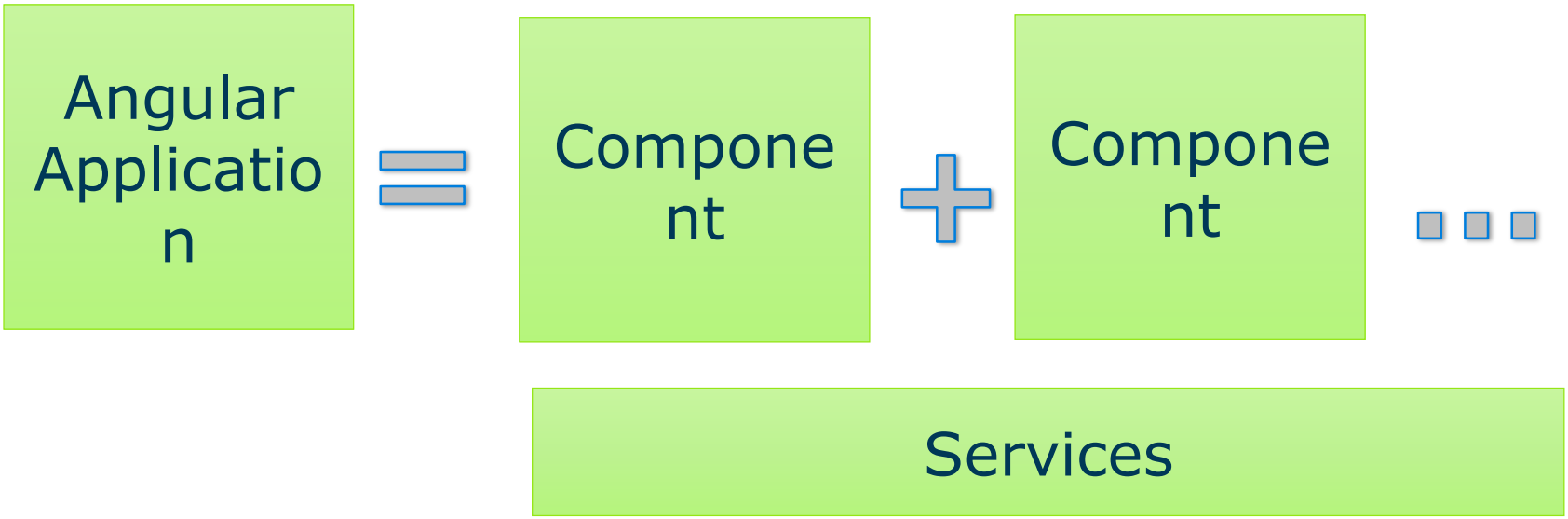
Angular Frameworks

Libraries



Angular 4 Application

Angular 4 application is comprised of a set of components and some services that provide functionality across those components.



Importing Modules

Module loader finds an external function or class using the *import* statement.

imports statement allows us to use exported members from external modules. External modules can be a third party library or our own modules or from angular itself.

If multiple members from the same module is needed, It can be listed in the import list separated by commas.

Angular is a collection of library modules, each library is itself a module made up of several related feature modules.

```
import { NgModule } from '@angular/core'
```

@angular/ core	@angular/ forms	@angular/ http	@angular/ router	**Angular Modules**

Root Module

An Angular module class 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.

- The conventional name for the root module is AppModule .

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-
browser';
import { AppComponent } from './app.component';

@NgModule({
imports: [ BrowserModule ],
declarations: [ AppComponent ],
bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Introduction to Angular Components

Components are the main way to build and specify elements and logic on the page.

Component is comprised of a template and class.

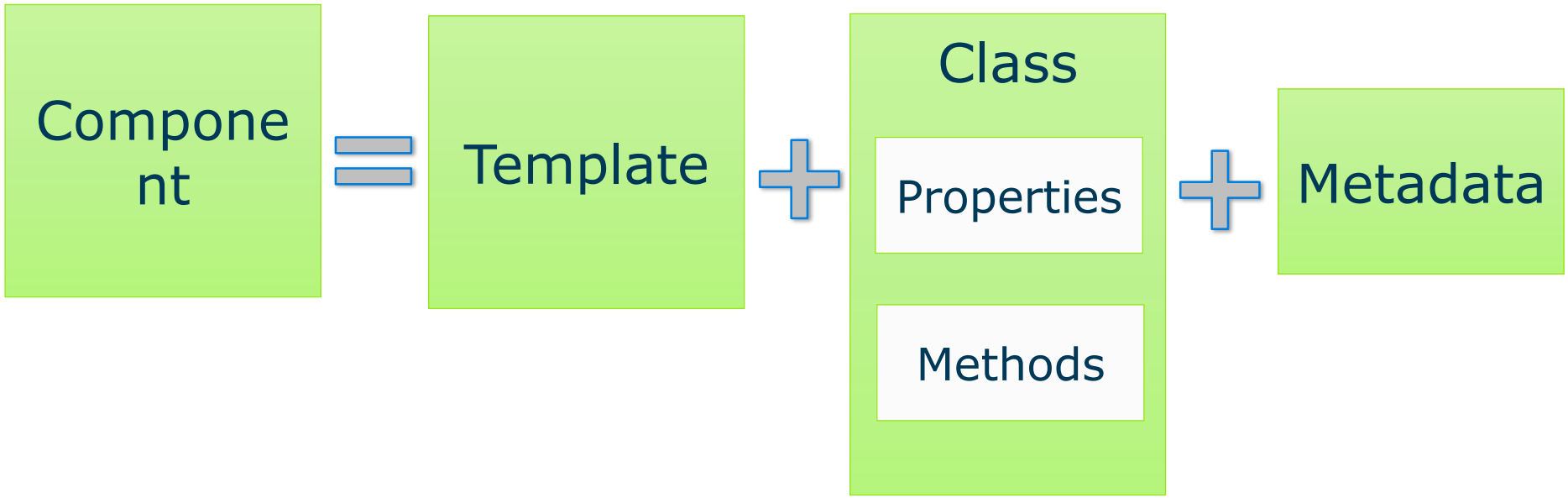
- Template provides HTML(View) for the user interface.
- Class provides the code associated with the view.
- Class contains the properties or data elements to be used in the view and methods to perform actions for the view.

Component also has metadata, which provides additional information about the component

- Meta data that identifies the class as an angular component.

Angular 4 Component

- A component contains application logic that controls a region of the user interface (view)

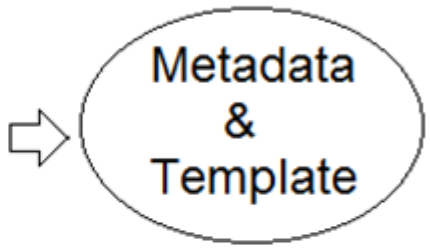


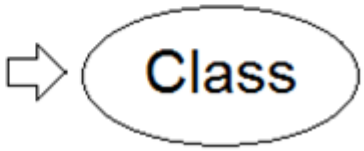
Angular Component Using TypeScript

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

@Component({
  selector: 'my-app',
  template: `<div>
      <h2>{{greet}}</h2>
    </div>`
})

export class AppComponent {
    greet:string = "Hello Angular2!"
}
```





Component Class

Class is a construct that allows to create a type with properties and methods.

As per convention, name the component class with a feature name then append the word *component* as the suffix.

Also by convention root component for an application is called AppComponent.

Class name is used as the component name when the component is referenced in code.

export keyword exports the class; thereby making it available for use by other components of the application.

```
export class AppComponent {
    greet:string = "Hello Angular2!"
}
```

Component Metadata

A class becomes an angular component, when component meta data is given. Metadata tells Angular how to process a class.

Angular needs metadata to understand, how to instantiate the component, construct the view and interact with the component.

Components meta data is defined with the angular *Component* function. In Typescript this function is attached to the class as a decorator.

@Component decorator is used to identify the class as a component. It takes object as a parameter which has many properties like selector, template etc

- selector defines the components directive name which is used to reference the component in HTML.
- Whenever this directive is used in the HTML angular renders this components template

Angular 4 Metadata

The main objective @Component decorators is to add meta-data to the application which tells Angular 2 how to process a class.

When building Angular components we can configure the following options:

- **selector** : defines the name of the HTML tag where the component resides
- **providers**: Used to pass services for the component
- **styles** : Used to style a specific component.
- **template** : This is the portion of our component that holds template. It is an integral part of the component as it allows to tie logic from component directly to a view. It can be either inline, or external template using **templateUrl** to link to an external template.

Demo

Creating-AppComponent

Template

Template are mostly HTML which is used to tell Angular how to render the component.

Template for a component can be created using

- Inline template (Embedded template string)
- Linked template (Template provided in external html file)

Inline template can be defined with ***template*** property using a single or double quotes or with a multiline string by enclosing the HTML in ES 2015 back ticks.

- Back ticks allows to compose multiline string which makes the HTML more readable.

Linked template is used to define the HTML in its own file by providing the URL of the HTML file using ***templateUrl*** property.

- Path provided in templateUrl property is relative to the application root which is usually the location of the index.html

Demo

InlineTemplate

LinkedTemplate

Component Styles

Angular 4 applications are styled with regular CSS. i.e. CSS stylesheets, selectors, rules, and media queries can be directly applied.

Angular 4 has the ability to encapsulate component styles with components enables more modular design than regular stylesheets.

In Angular 4 component, CSS styles can be defined like HTML template in several ways

- As inline style in the template HTML
- Template Link Tags
- By setting ***styles*** or ***styleUrls*** metadata

Demo

ComponentStyles

Summary

Every application must bootstrap at least one component, the root application component.

As a best practice, We need to launch the application by bootstrapping the AppModule in the main.ts file.

The bootstrap array should be used only in root application module i.e. AppModule

NgModule is a way to organize the dependencies for compiler and dependency injection.

NgModule can import other modules as dependencies

Angular uses a library called ***Zone.js*** to automatically detect the things changed and trigger the change detection mechanism

Summary

Every component must be declared in some NgModule and a component can belong to one and only one NgModule

exports key is nothing but the list of public components for NgModule.

Angular2 Application is a tree of components and the top level component is nothing but the application.

Components are Composable.

Template for a component can be created using InlineTemplate and LinkedTemplate using template and templateUrl respectively.

styles and **styleUrls** keys are used in components to work with styles in Angular 2