Install node

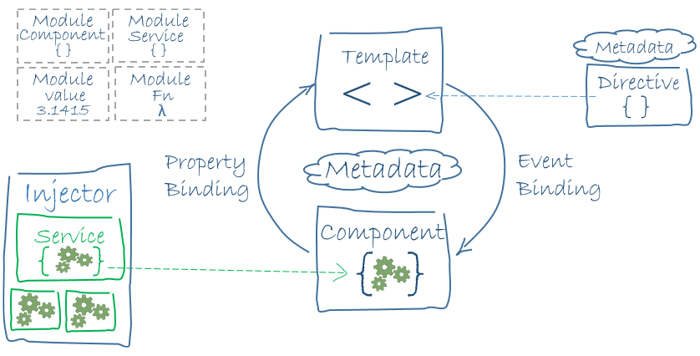
npm install -g @angular/cli

ng new mymeanapp

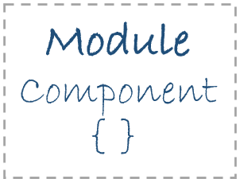
Angular Fundamentals

1. Architecture
2. Template and Data Binding
3. Forms
4. Bootstraping
5. NgModules
6. Dependency Injection
7. Http Client
8. Routing and Navigation
9. Testing

**Architecture:**



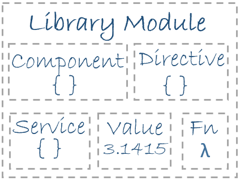
NgModule -Every Angular app has at least one NgModule class, [the *root module*](https://angular.io/guide/bootstrapping), conventionally named AppModule



1. Declarations (the *view classes* that belong to this module. Angular has three kinds of view classes: [components](https://angular.io/guide/architecture#components), [directives](https://angular.io/guide/architecture#directives), and [pipes](https://angular.io/guide/pipes).)
   1. Components
   2. Directives
   3. Pipes
2. Exports
3. Imports
4. Providers
5. Bootstrap - the main application view, called the *root component*, that hosts all other app views. Only the *root module* should set this bootstrap property.

Component classes should be lean. They don't fetch data from the server, validate user input, or log directly to the console. They delegate such tasks to services.

**Angular Libraries -** (Angular ships as a collection of JavaScript modules. You can think of them as library modules.)



ex:

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

**Components**

A *component* controls a patch of screen called a *view*.

**Templates**

You define a component's view with its companion template. A template is a form of HTML that tells Angular how to render the component.

**Metadata**

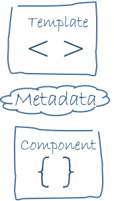


Metadata tells Angular how to process a class.

In fact, HeroListComponent really is *just a class*. It's not a component until you *tell Angular about it*.

To tell Angular that HeroListComponent is a component, attach metadata to the class.

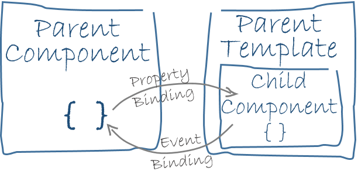
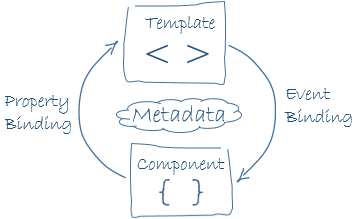
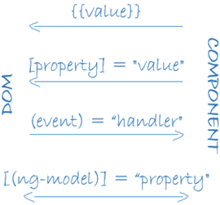
In TypeScript, you attach metadata by using a decorator



Here are a few of the most useful @Component configuration options:

Selector, Template URL and Providers

**Data Binding**



As the diagram shows, there are four forms of data binding syntax. Each form has a direction — to the DOM, from the DOM, or in both directions.

**Directives**



1. Structural Directives : alter layout by adding, removing, and replacing elements in DOM.
   1. ngIf, ngFor
2. Attribute Directive
   1. ngModel, ngSwitch, ngStyle and ngClass

**Services:** *Service* is a broad category encompassing any value, function, or feature that your application needs.



Examples include:

* logging service
* data service
* message bus
* tax calculator
* application configuration

**Dependency Injection :** *Dependency injection* is a way to supply a new instance of a class with the fully-formed dependencies it requires. Most dependencies are services. Angular uses dependency injection to provide new components with the services they need.



1. When Angular creates a component, it first asks an injector for the services that the component requires.
2. An injector maintains a container of service instances that it has previously created
3. In general, add providers to the [root module](https://angular.io/guide/architecture#modules) so that the same instance of a service is available everywhere.

Points to remember about dependency injection:

* Dependency injection is wired into the Angular framework and used everywhere.
* The *injector* is the main mechanism.
  + An injector maintains a *container* of service instances that it created.
  + An injector can create a new service instance from a *provider*.
* A *provider* is a recipe for creating a service.
* Register *providers* with injectors.

**Template and Data Binding**

**Displaying Data:**

* Interpolation with double curly braces to display a component property.
* ngFor to display an array of items.
* A TypeScript class to shape the model data for your component and display properties of that model.
* ngIf to conditionally display a chunk of HTML based on a boolean expression.