Dependency Injection

Jogesh K. Muppala

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

What is Dependency Injection (DI)?

- Software design pattern that implements inversion of control for resolving dependencies
  - Dependency: An object that can be used (a service)
  - Injection: Passing of a dependency to a dependent object so that it can use it. The client does not need to build the object
  - Coined by Martin Fowler in 2004

https://en.wikipedia.org/wiki/Dependency_injection

Dependency

- Three ways for a component to get hold of its dependencies:
  - Create dependency using new operator
  - Look up dependency using a global variable
  - Have dependency passed to it where needed
- Third option is most flexible
  - Hard coding of dependency avoided
  - Testing becomes feasible

Dependency Injection

- DI involves four roles:
  - The service
  - The client
  - The interfaces
  - The injector

Angular and DI

- Separation of business logic and dependency construction
- The dependency is passed to the object consuming it where it is needed
- Angular injector subsystem is responsible for:
  - creating components
  - resolving their dependencies, and
  - providing them to other components

Angular and DI

- DI is extensively used in Angular
  - Components such as services, directives, filters and animations
    - Defined by injectable factory method or constructors
    - Injected with service and value components
  - Controllers can be injected with the components
  - The config and run methods also accept injection of some components

Dependency Annotation in Angular

- Inline array annotation

```
module.controller('MenuController',  ['$scope',  'menuFactory', function($scope,  menuFactory) {


}]);
```

- $inject property annotation

```
var MenuController  = function($scope,  menuFactory) {


};
MenuController.$inject  = ['$scope',   'menuFactory'];
module.controller('MenuController',  MenuController);
```

- Implicit annotation

```
module.controller('MenuController',  function($scope,  menuFactory) {


}]);
```