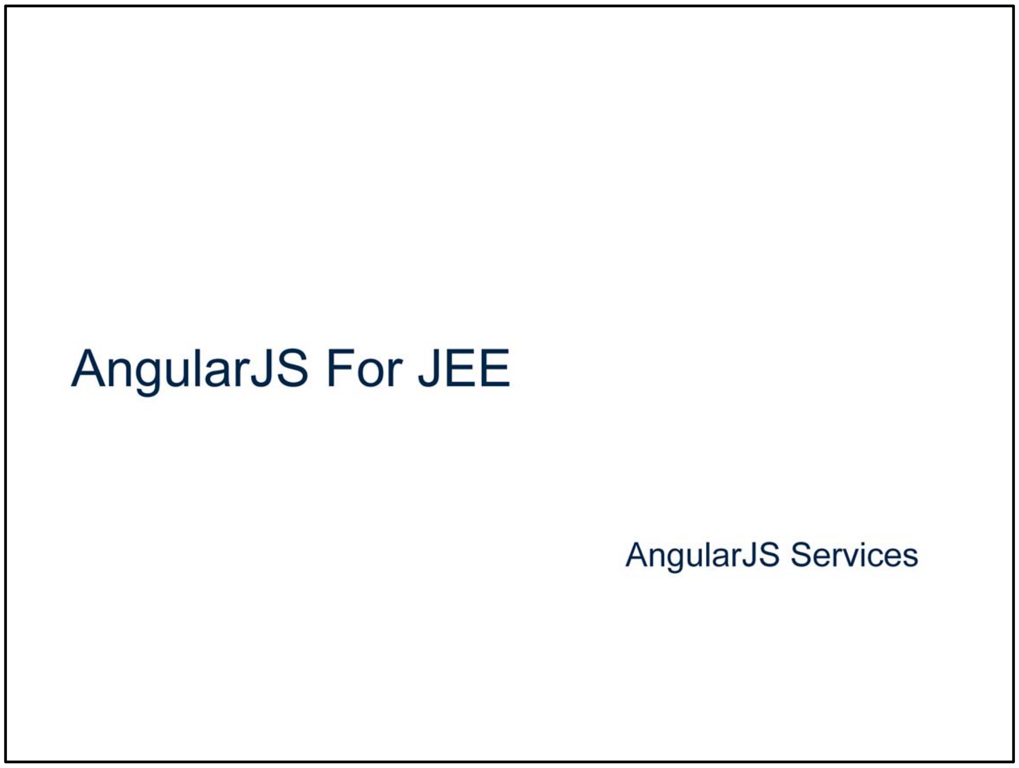
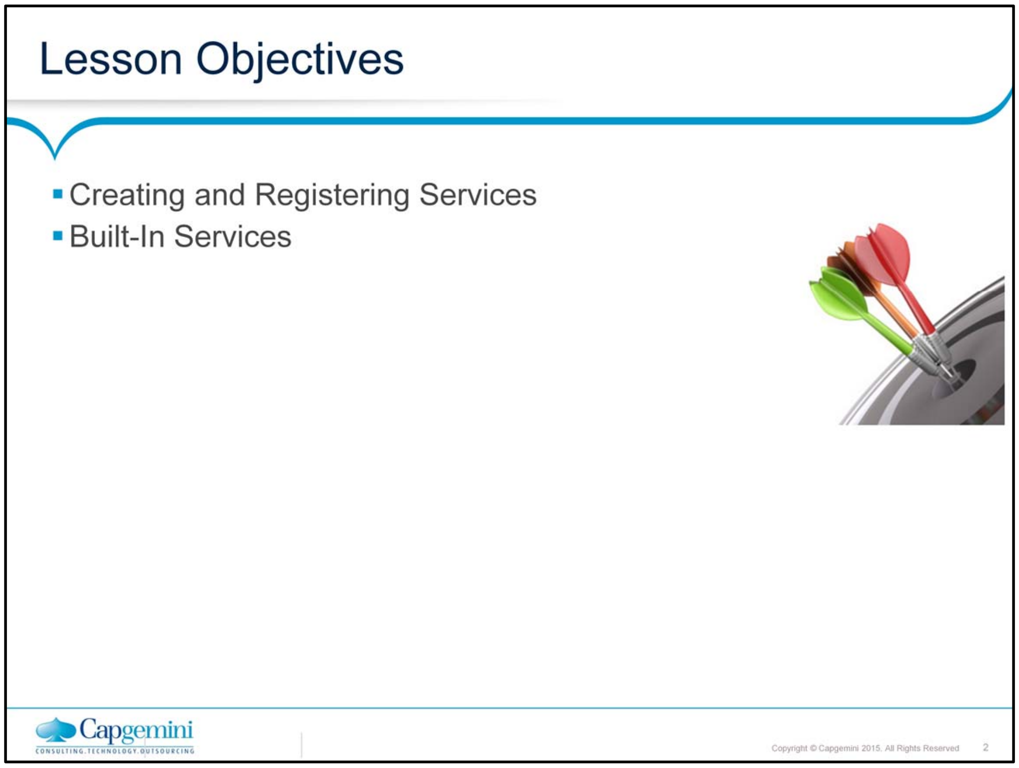


AngularJS For JEE

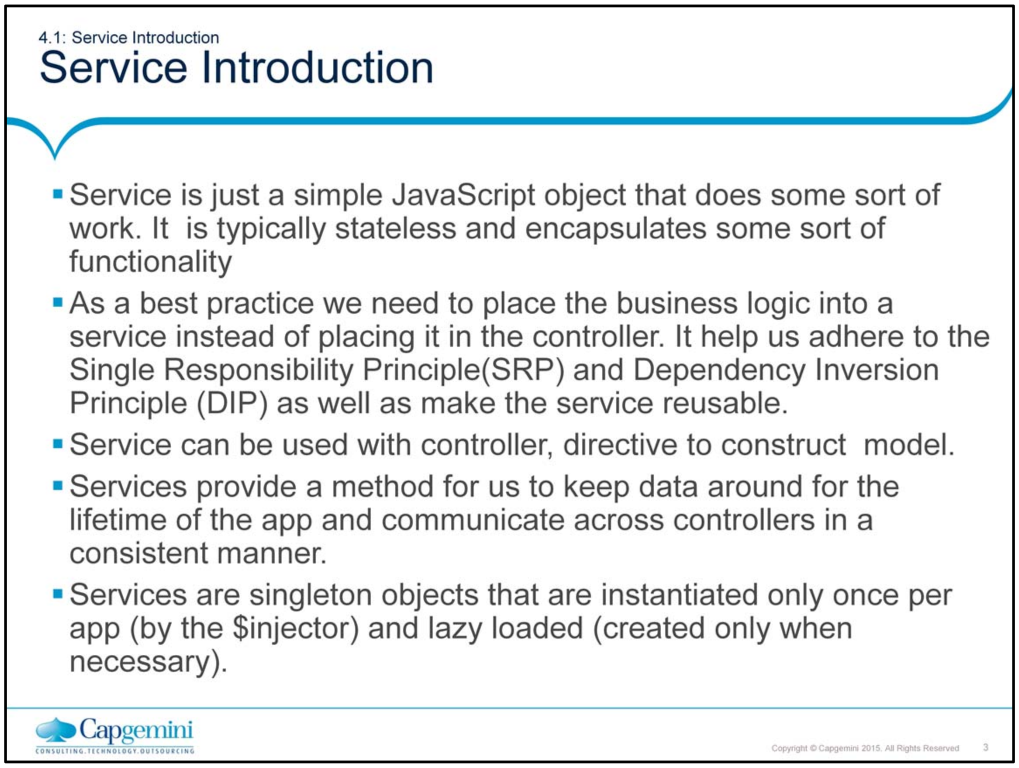
AngularJS Services



Lesson Objectives

- Creating and Registering Services
- Built-In Services

2



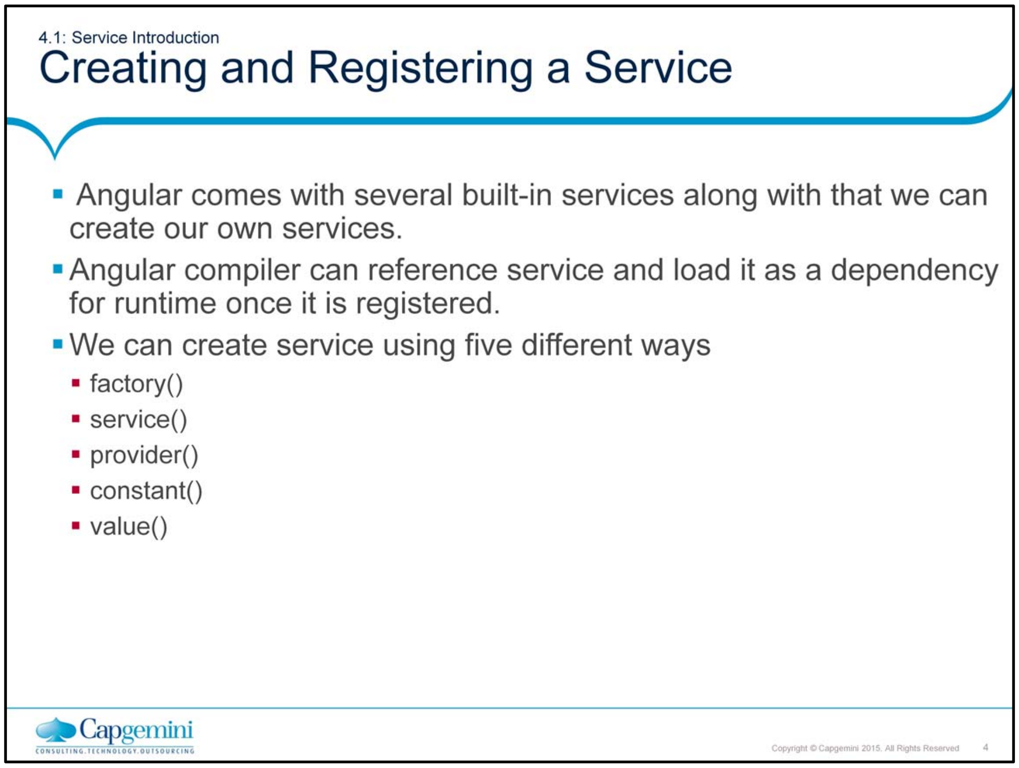
4.1: Service Introduction

Service Introduction

- Service is just a simple JavaScript object that does some sort of work. It is typically stateless and encapsulates some sort of functionality
- As a best practice we need to place the business logic into a service instead of placing it in the controller. It help us adhere to the Single Responsibility Principle(SRP) and Dependency Inversion Principle (DIP) as well as make the service reusable.
- Service can be used with controller, directive to construct model.
- Services provide a method for us to keep data around for the lifetime of the app and communicate across controllers in a consistent manner.
- Services are singleton objects that are instantiated only once per app (by the $injector) and lazy loaded (created only when necessary).

Single Responsibility Principle teaches that every object should have a single responsibility. If we use the example of a controller, it's responsibility is essentially to wire up the scope (which holds your models) to your view; it is essentially the bridge between your models and your views. If your controller is also responsible for making ajax calls to fetch and update data, this is a violation of the SRP principle. Logic like that (and other business logic) should instead be abstracted out into a separate service, then it can be injected into the objects that need to use it. This is where the Dependency Inversion Principle (DIP) comes in.

The DIP states that objects should depend on abstractions, not concretions. In languages like C# and Java, this means your objects depend on Interfaces instead of Classes. In JavaScript you could look at any parameter of any function (constructor function or otherwise) as an abstraction, since you can pass in any object for that parameter so long as it has the members on it that are used within that method. But the key here is the ability to use dependency injection — the ability to inject into other objects. This means that all of your controllers, directives, filters and other services should take in any external dependencies as parameters during construction. This allows you to have loosely coupled code and has the added benefit of making unit testing much easier.

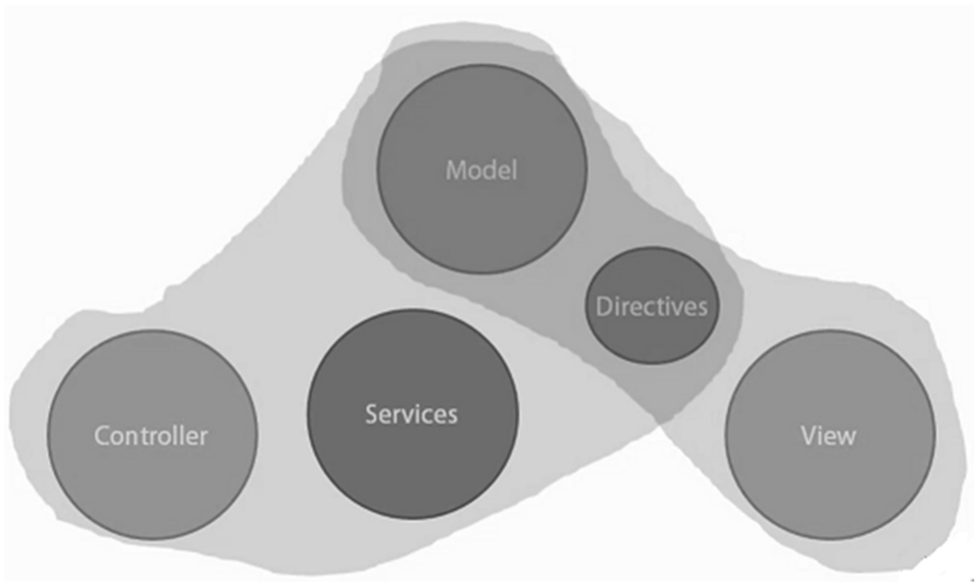


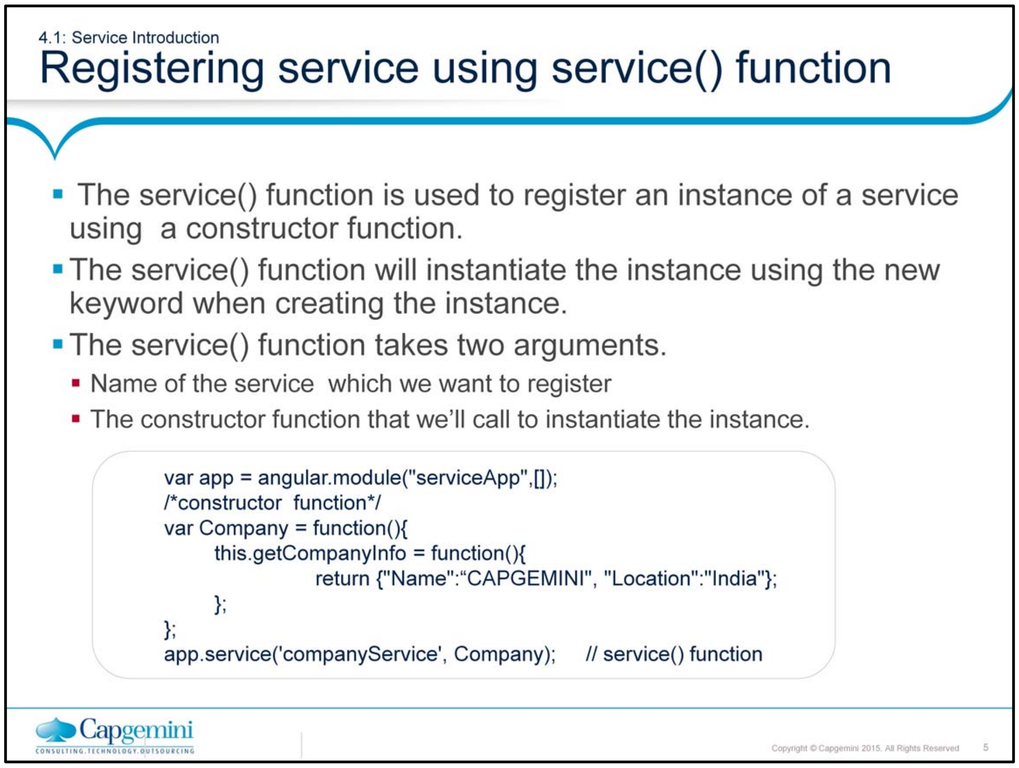
4.1: Service Introduction

Creating and Registering a Service

- Angular comes with several built-in services along with that we can create our own services.
- Angular compiler can reference service and load it as a dependency for runtime once it is registered.
- We can create service using five different ways
  - factory()
  - service()
  - provider()
  - constant()
  - value()

We are discussing here two ways 1. factory 2 Service





4.1: Service Introduction

Registering service using service() function

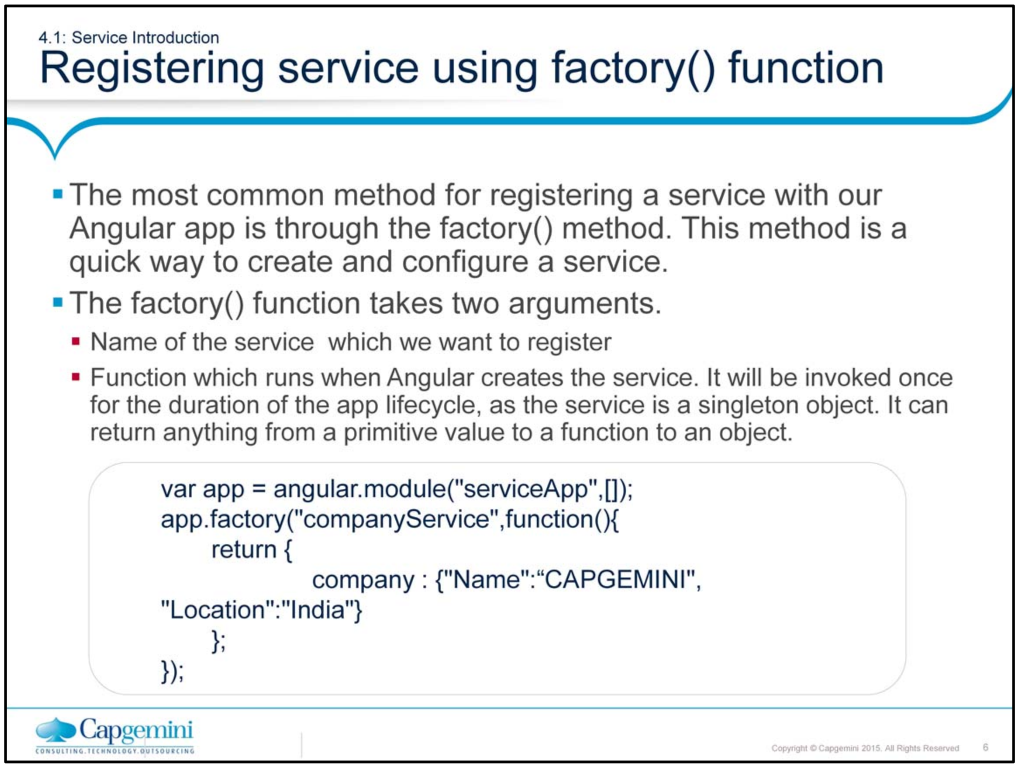
- The service() function is used to register an instance of a service using a constructor function.
- The service() function will instantiate the instance using the new keyword when creating the instance.
- The service() function takes two arguments.
  - Name of the service which we want to register
  - The constructor function that we'll call to instantiate the instance.

```
var app = angular.module("serviceApp",[]);
/*constructor  function*/
var Company = function(){
        this.getCompanyInfo = function(){
                        return {"Name":“CAPGEMINI", "Location":"India"};
        };
};
app.service('companyService', Company);     // service() function
```

Copyright © Capgemini 2015. All Rights Reserved 5

We can define a service like this:
app.service('MyService', function () {
this.sayHello = function () {
console.log('hello'); }; });

.service() is a method on our module that takes a name and a function that defines the service. Once defined, we can inject and use that particular service in other components, like controllers, directives and filters, like this:
app.controller('AppController', function (MyService) { MyService.sayHello(); // logs 'hello' });



4.1: Service Introduction

Registering service using factory() function

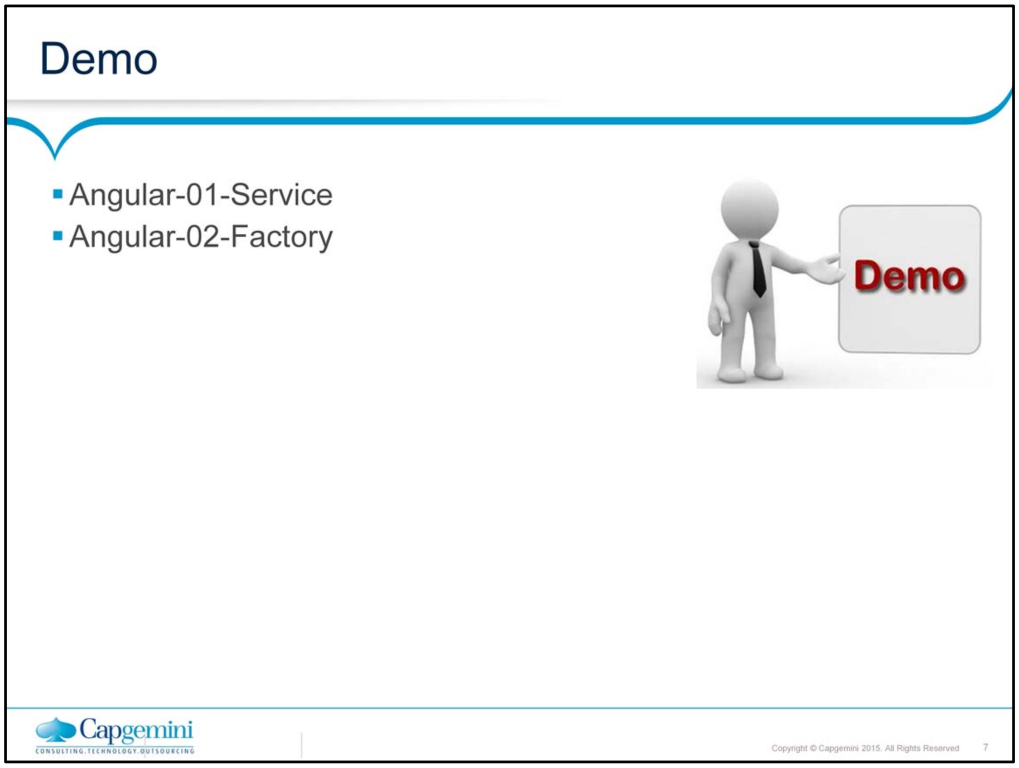
- The most common method for registering a service with our Angular app is through the factory() method. This method is a quick way to create and configure a service.
- The factory() function takes two arguments.
  - Name of the service which we want to register
  - Function which runs when Angular creates the service. It will be invoked once for the duration of the app lifecycle, as the service is a singleton object. It can return anything from a primitive value to a function to an object.

```
var app = angular.module("serviceApp",[]);
app.factory("companyService",function(){
      return {
                  company : {"Name":"CAPGEMINI",
"Location":"India"}
      };
});
```

6

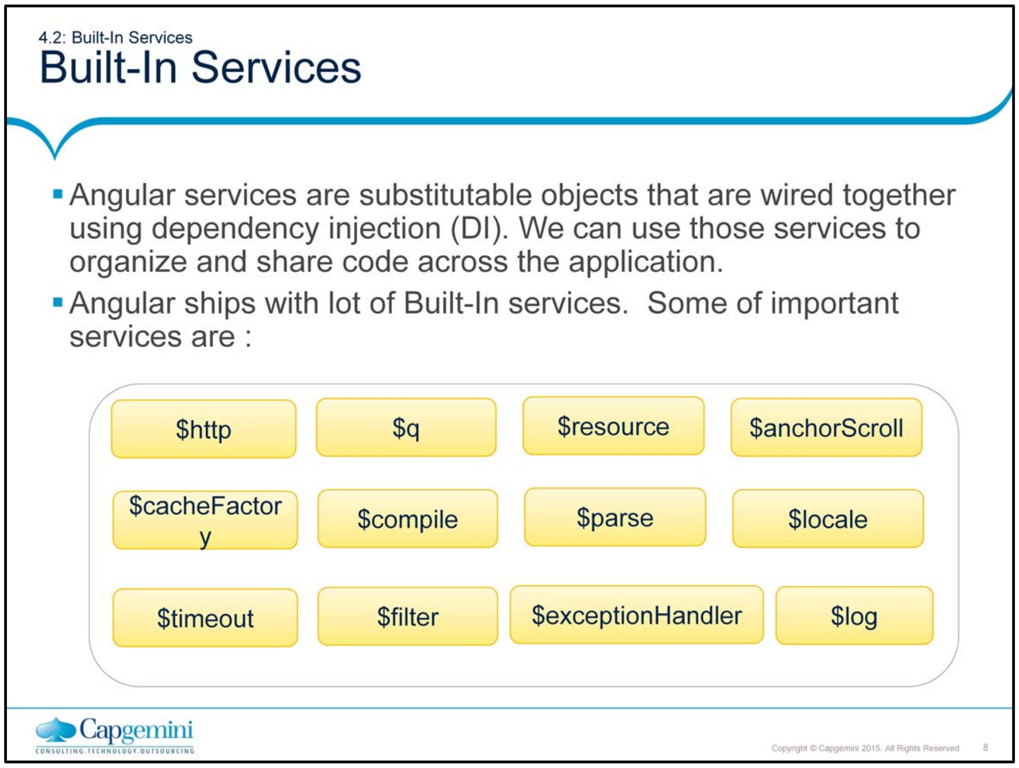
```
app.factory('MyService', function () {
return {
sayHello: function ()
{ console.log('hello'); }; } });
```

.factory() is a method on our module and it also takes a name and a function, that defines the factory. We can inject and use that thing exactly the same way we did with the service.
So What’s the difference .Instead of working with this in the factory, we’re returning an object literal. A service is a constructor function whereas a factory is not. However, a factory function is really just a function that gets called, which is why we have to return an object explicitly.



Demo

- Angular-01-Service
- Angular-02-Factory



4.2: Built-In Services

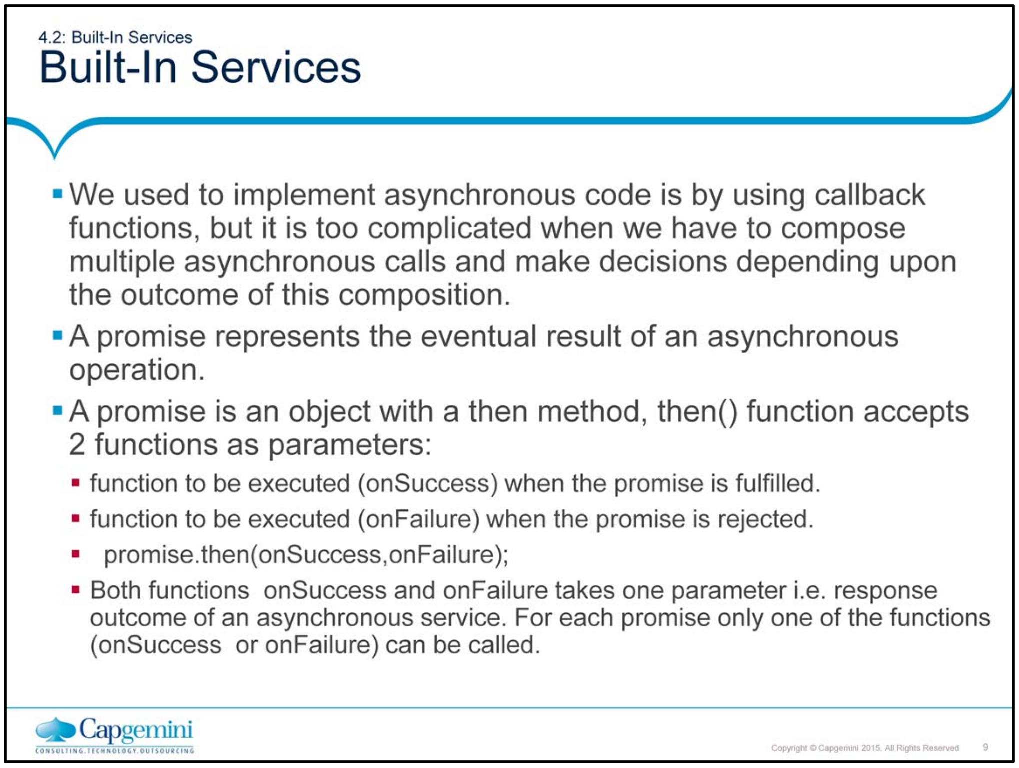
Built-In Services

- Angular services are substitutable objects that are wired together using dependency injection (DI). We can use those services to organize and share code across the application.
- Angular ships with lot of Built-In services. Some of important services are :

$http
$q
$resource
$anchorScroll
$cacheFactory
$compile
$parse
$locale
$timeout
$filter
$exceptionHandler
$log

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

Angular services are injectable object injected using dependency injection (DI) mechanism of AngularJS. The AngularJS services can be used to organize and share code across your app.
Angular services are:
Lazily instantiated - Angular only instantiates a service when an application component depends on it.
Singletons - Each component dependent on a service gets a reference to the single instance generated by the service factory.
$http: The $http service is a core Angular service that facilitates communication with the remote HTTP servers via the browser's XMLHttpRequest object or via JSONP.
$q:A service that helps you run functions asynchronously, and use their return values (or exceptions) when they are done processing.
$anchorScroll: It is used to set checkbox checked.
$cacheFactory: It evaluates specified expression when the user changes the input.
$compile: Compiles an HTML string or DOM into a template and produces a template function, which can then be used to link scope and the template together.
$parse: It is used to convert AngularJS expression into a function.
**$locale:**$locale service provides localization rules for various Angular components.
$timeout: Angular's wrapper for window.setTimeout. The fn function is wrapped into a try/catch block and delegates any exceptions to $exceptionHandler service.
$exceptionHandler: It is used to handle an uncaught exception in angular expressions by overriding the default exception handler behaviour.
$filter: Filters are used for formatting data displayed to the user.
$log: It is used to write error, info, warning and debugging into the browser's console.
We are discussing here two ways 1. $q 2 $http



4.2: Built-In Services

Built-In Services

- We used to implement asynchronous code is by using callback functions, but it is too complicated when we have to compose multiple asynchronous calls and make decisions depending upon the outcome of this composition.
- A promise represents the eventual result of an asynchronous operation.
- A promise is an object with a then method, then() function accepts 2 functions as parameters:
  - function to be executed (onSuccess) when the promise is fulfilled.
  - function to be executed (onFailure) when the promise is rejected.
  - promise.then(onSuccess,onFailure);
  - Both functions onSuccess and onFailure takes one parameter i.e. response outcome of an asynchronous service. For each promise only one of the functions (onSuccess or onFailure) can be called.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

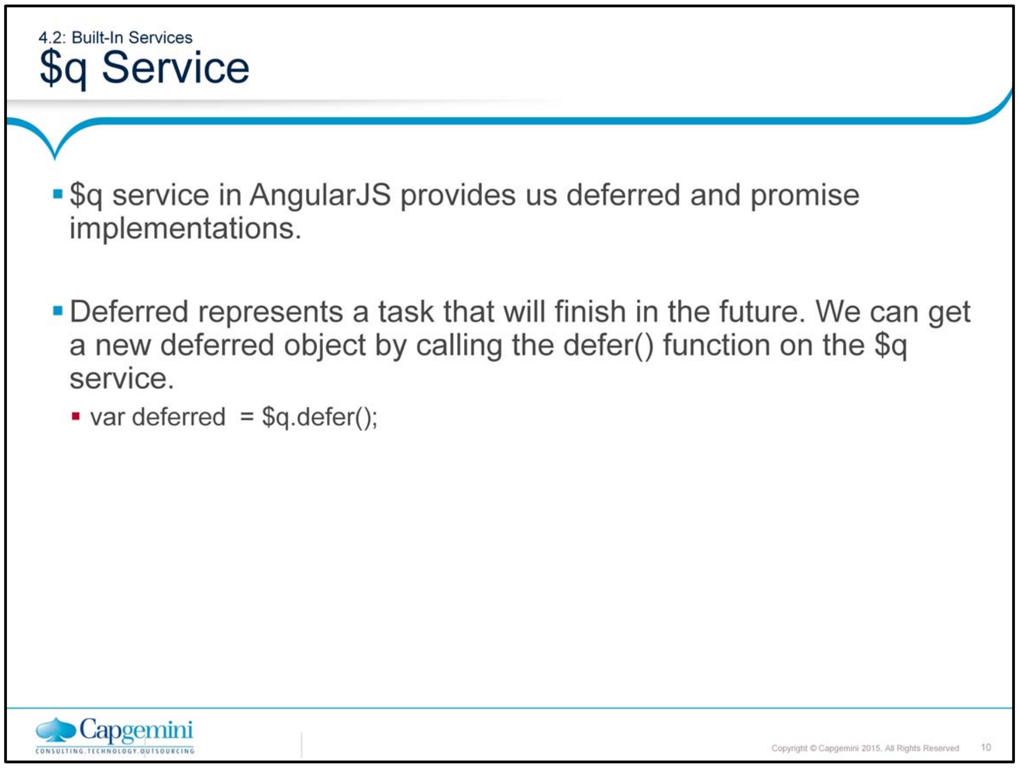
A promise in the Javascript and AngularJS world is an assurance that we will get a result from an action at some point in the future, two possible results of a promise:

1. A promise is said to be **fulfilled** when we get a result from that action
2. A promise is said to be **rejected** when we don’t get a response

A promise represents the eventual result of an operation. You can use a promise to specify what to do when an operation eventually succeeds or fails.

```
var promise = $http.get("/api/my/name");
promise.success(function(name) {
console.log("Your name is: " + name); });
promise.error(function(response, status)
{ console.log("The request failed with response " + response + " and status code " + status); });
```

Promises are not actually complicated, they're objects that contain a reference to functions to call when something fails or succeeds. AngularJS actually wires up a promise for an HTTP request


4.2: Built-In Services

$q Service

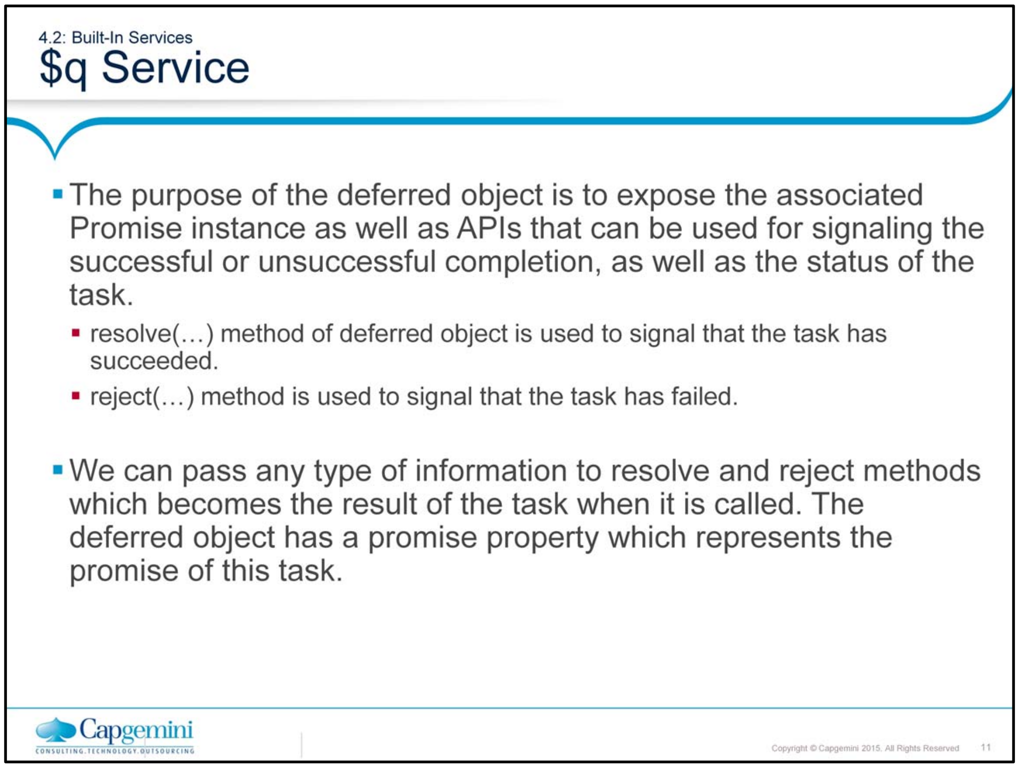
- $q service in AngularJS provides us deferred and promise implementations.
- Deferred represents a task that will finish in the future. We can get a new deferred object by calling the defer() function on the $q service.
  - var deferred = $q.defer();

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 10

The $q service itself is also a function that allows you to quickly convert a callback based asynchronous function into a promise based solution.
This method accomplishes the same thing as manually creating the deferred object-- which way you choose is up to preference and whether you want to notify() the calling code.

```
function getData($timeout, $q)
{ return function() {
// simulated async function
 return $q(function(resolve, reject)
{ $timeout(function()
{ if(Math.round(Math.random())) { resolve('data received!') }
else { reject('oh no an error! try again') } }, 2000) }) } }
```



4.2: Built-In Services

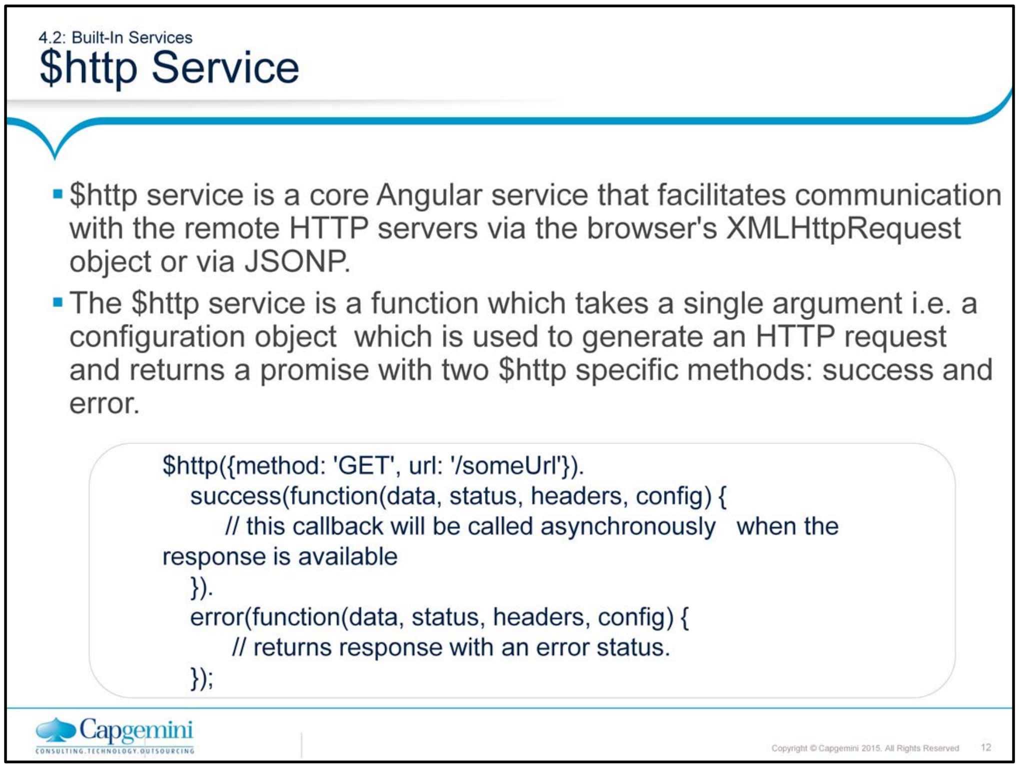
$q Service

- The purpose of the deferred object is to expose the associated Promise instance as well as APIs that can be used for signaling the successful or unsuccessful completion, as well as the status of the task.
  - resolve(…) method of deferred object is used to signal that the task has succeeded.
  - reject(…) method is used to signal that the task has failed.
- We can pass any type of information to resolve and reject methods which becomes the result of the task when it is called. The deferred object has a promise property which represents the promise of this task.

11

The $q service itself is also a function that allows you to quickly convert a callback based asynchronous function into a promise based solution.
This method accomplishes the same thing as manually creating the deferred object-- which way you choose is up to preference and whether you want to notify() the calling code.

```
function getData($timeout, $q)
{ return function() {
// simulated async function
 return $q(function(resolve, reject)
{ $timeout(function()
{ if(Math.round(Math.random())) { resolve('data received!') }
else { reject('oh no an error! try again') } }, 2000) }) } }
```



4.2: Built-In Services

$http Service

- $http service is a core Angular service that facilitates communication with the remote HTTP servers via the browser's XMLHttpRequest object or via JSONP.
- The $http service is a function which takes a single argument i.e. a configuration object which is used to generate an HTTP request and returns a promise with two $http specific methods: success and error.

```
$http({method: 'GET', url: '/someUrl'}).
   success(function(data, status, headers, config) {
       // this callback will be called asynchronously   when the
response is available
   }).
   error(function(data, status, headers, config) {
       // returns response with an error status.
   });
```

12

$http gives shortcut following methods by just passing the URL which returns promise.

- $http.get(url, config)
- $http.post(url, data, config)
- $http.put(url, data, config)
- $http.delete(url, config)
- $http.head(url, config)

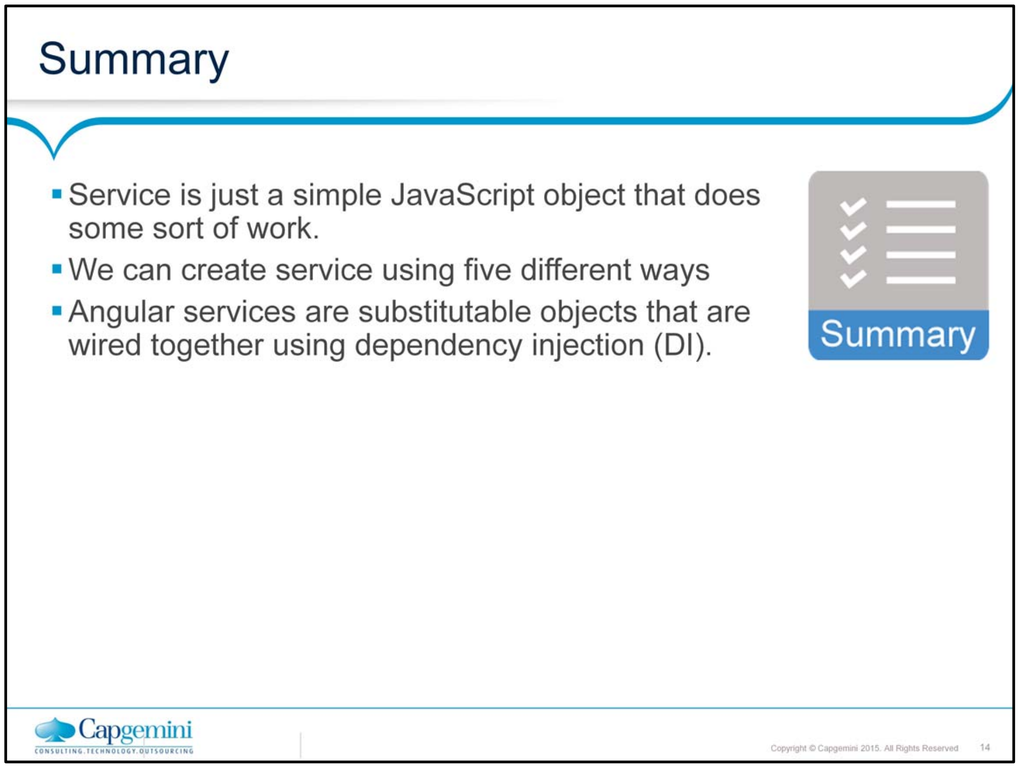
var responsePromise = $http.get("url");

The config parameter passed to the different $http functions controls the HTTP request sent to the server. The config parameter is a JavaScript object which can contain the following properties:

- method
- url
- params
- headers
- timeout
- cache
- transformRequest
- transformResponse

Demo

- Lesson04-Http

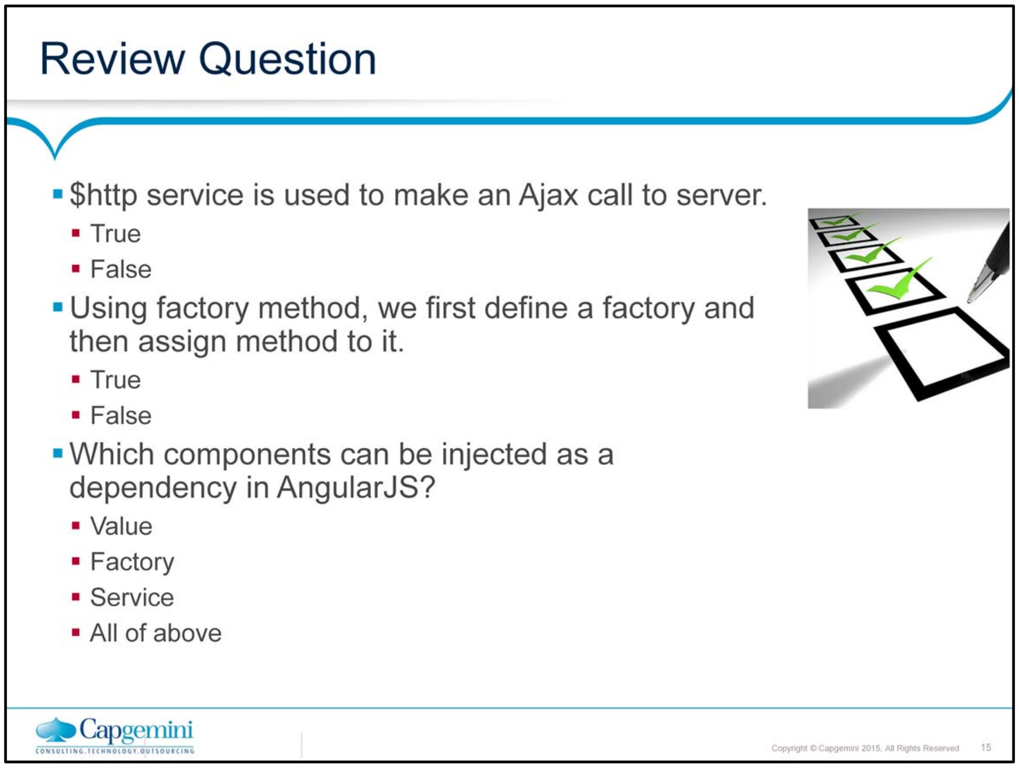


Summary

- Service is just a simple JavaScript object that does some sort of work.
- We can create service using five different ways
- Angular services are substitutable objects that are wired together using dependency injection (DI).

14

Add the notes here.



Review Question

- $http service is used to make an Ajax call to server.
  - True
  - False
- Using factory method, we first define a factory and then assign method to it.
  - True
  - False
- Which components can be injected as a dependency in AngularJS?
  - Value
  - Factory
  - Service
  - All of above