

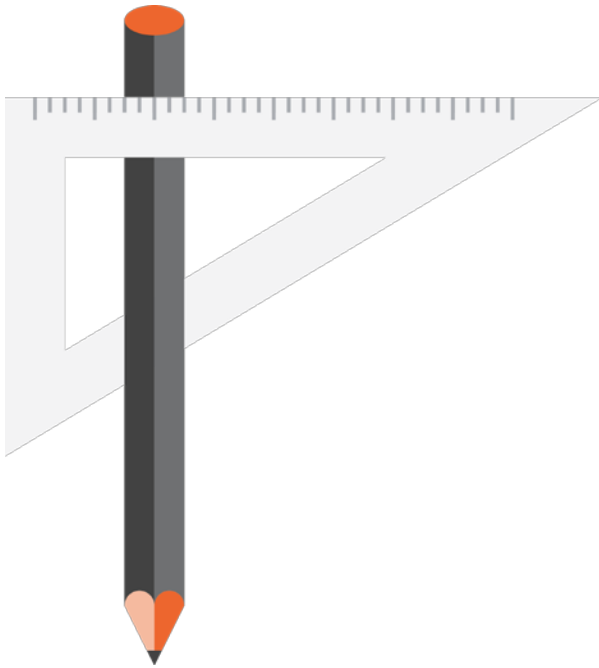
# Creating Services



Brice Wilson

@brice\_wilson | [www.BriceWilson.net](http://www.BriceWilson.net)

# Overview



Five different ways to create services!!!

Providers versus services

Dependency annotations

# 5 Service Flavors

provider()

factory()

service()

value()

constant()

\$provide

\$provide


provider

service

```
$provide.provider('books', function () {  
  this.$get = function () {  
    var appName = 'Book Logger';  
    return {  
      appName: appName  
    };  
  }  
});
```

## Using \$provide.provider()


- Call the “provider” function on the \$provide service
- Provider must define a “\$get” function
- Service is the object returned from the \$get function
- Configurable via the underlying provider




```
function factory(name, factoryFn, enforce) {  
  return provider(name, {  
    $get: enforce !== false ? enforceReturnValue(name, factoryFn) : factoryFn  
  });  
}
```

## Using \$provide.factory()

- Simpler version of provider when additional configuration is unnecessary
- Registers a service factory function that will return a service instance



```
function service(name, constructor) {  
  return factory(name, ['$injector', function($injector) {  
    return $injector.instantiate(constructor);  
  }]);  
}
```



## Using \$provide.service()

- Calls factory function which calls provider function
- Treats function it is passed as a constructor
- Executes constructor function with “new” operator

# Value and Constant Services

## Value Services

- Shorthand for factory with no parameters
- Cannot be injected into a module configuration function
- Can be overridden by an AngularJS decorator

## Constant Services

- Simply registers service with injector, no factory/provider calls
- Can be injected into a module configuration function
- Cannot be overridden by an AngularJS decorator

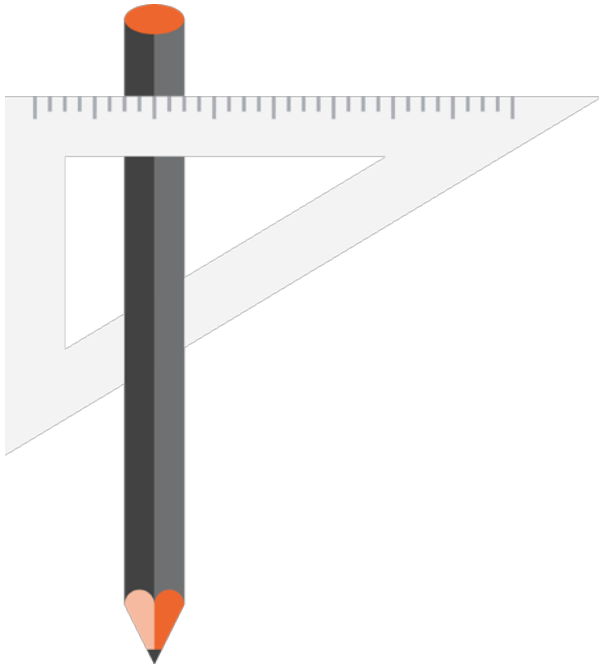


# Dependency Annotations

- Inform injector what services to inject
- Use to support minimization
- Three techniques available
  - Implicitly from function parameter names
  - Using `$inject` property annotation
  - Using inline array annotation



# Summary



Reasons for different creation methods

How and why to use a particular API

Minimization-safe dependency injection