



Shaping Up with Angular JS

Level 5: Dependencies and Services



Starting to get a bit cluttered?

```
(function(){
  var app = angular.module('store', []);
  app.controller('StoreController', function(){ . . . });
  app.directive('productTitle', function(){ . . . });
                                                          Can we refactor
these out?
  app.directive('productGallery', function(){ . . . });
  app.directive('productPanels', function(){ . . . });
})();
```



Extract into a new file ... products.js



Make a new Module

```
(function(){
  var app = angular.module('store', []);
  app.controller('StoreController', function(){
       . . .
})();
Define a new module just
  for Product stuff...

Module Name

app.js
```

Different closure means different app variable.

```
(function(){
   var app = angular.module('store-products', [ ]);
   app.directive('productTitle', function(){ . . . });
   app.directive('productGallery', function(){ . . . });
   app.directive('productPanels', function(){ . . . });
})();
   products.js
```



Add it to the dependencies ...

store depends on store-products

```
(function(){
  var app = angular.module('store-products', [ ]);
  app.directive('productTitle', function(){ . . . });
  app.directive('productGallery', function(){ . . . });
  app.directive('productPanels', function(){ . . . });
})();
  products.js
```

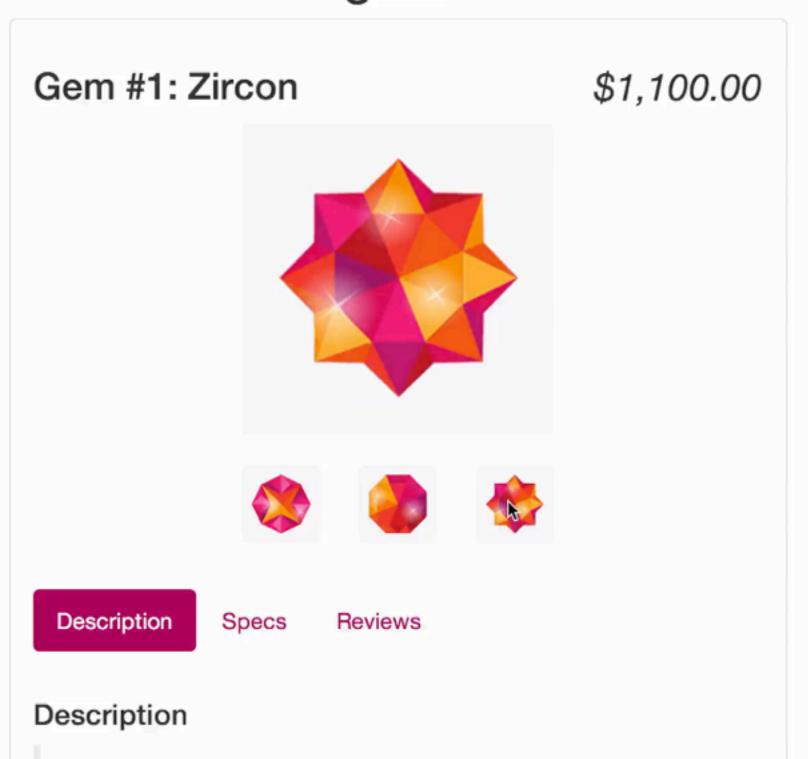


We'll also need to include the file

```
<!DOCTYPE html>
<html ng-app="store">
  <head> . . </head>
  <body ng-controller="StoreController as store">
   <script src="angular.js"></script>
   <script src="app.js"></script>
    <script src="products.js"></script>
 </body>
</html>
                                                              index.html
```

Flatlander Crafted Gems

- an Angular store -





How should I organize my application Modules?

Best to split Modules around functionality:

- app.js top-level module attached via ng-app
- products.js all the functionality for products and only products



Challenges



Does this feel strange?

```
(function(){
  var app = angular.module('store', [ 'store-products' ]);
  app.controller('StoreController', function(){
    this.products = [
      { name: '. . .', price: 1.99, . . . },
      { name: '. . .', price: 1.99, . . . },
      { name: '. . .', price: 1.99, . . . },
                                        What is all this data
                                             doing here?
 });
})();
                                                                  app.js
```

Where can we put it? Shouldn't we fetch this from an API?



How do we get that data?

http://api.example.com/products.json



We need a Service!

Services give your Controller additional functionality, like ...

- Fetching JSON data from a web service with \$http
- Logging messages to the JavaScript console with \$log
- Filtering an array with \$filter

All built-in Services start with a \$ sign ...



Introducing the \$http Service!

The \$http Service is how we make an async request to a server ...

By using \$http as a function with an options object:

```
$http({ method: 'GET', url: '/products.json' });
```

Or using one of the shortcut methods:

```
$http.get('/products.json', { apiKey: 'myApiKey' });
```

- Both return a Promise object with .success() and .error()
- If we tell \$http to fetch JSON, the result will be automatically decoded into JavaScript objects and arrays

```
So how do we use it?
```



How does a Controller use a Service like \$http?

Use this funky array syntax:

```
app.controller('SomeController', ['$http', function($http){
}]);

Service name

Service name as an argument

app.controller('SomeController', ['$http', '$log', function($http, $log){
}]);

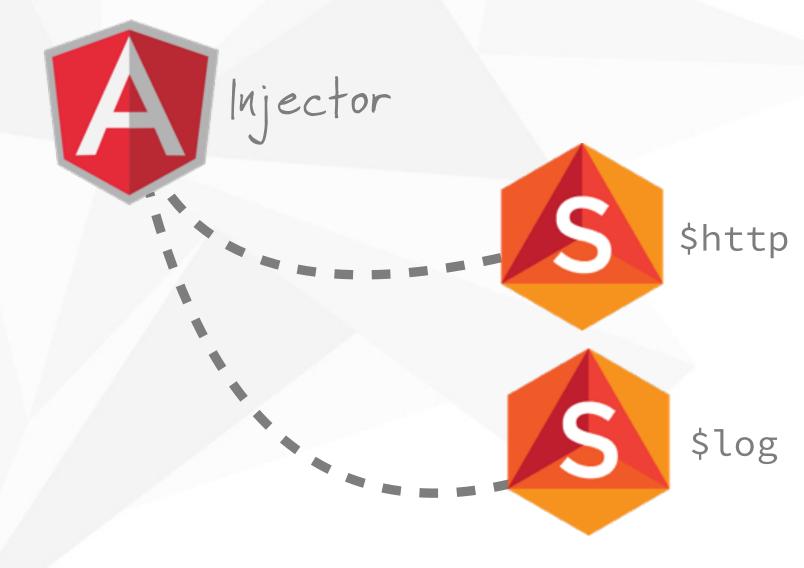
If you needed more than one
```



When Angular is Loaded Services are Registered

```
app.controller('SomeController', [ '$http', '$log', function($http, $log){
} ]);
```

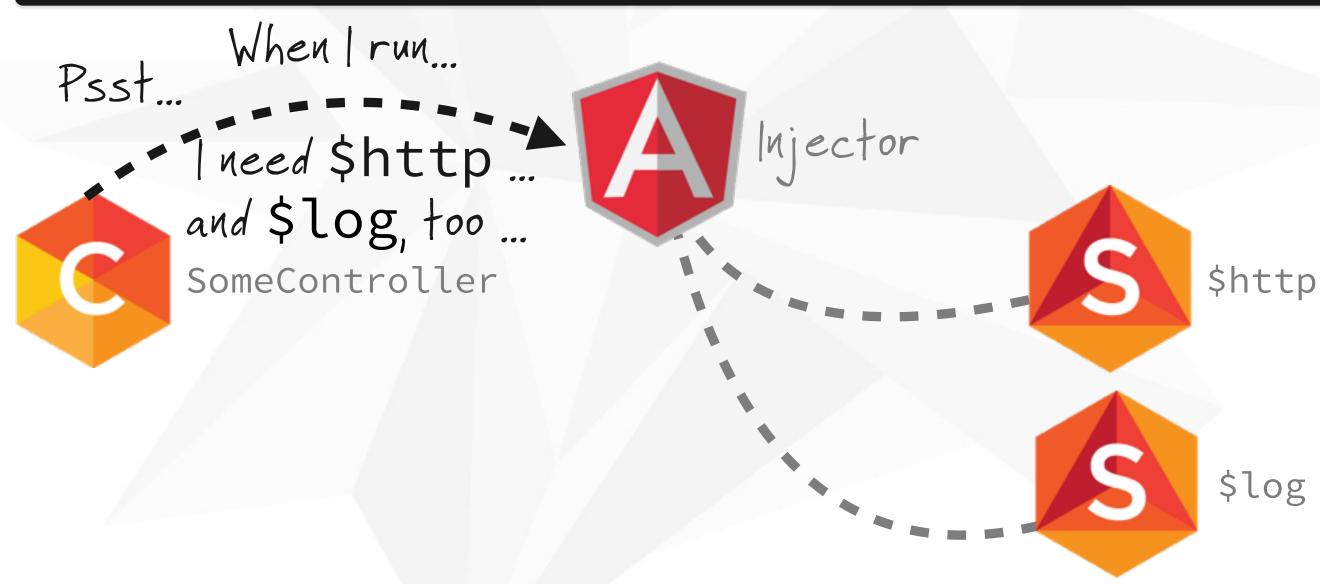






A Controller is Initialized

```
app.controller('SomeController', [ '$http', '$log', function($http, $log){
} ]);
```





Then When the Controller runs ...

```
app.controller('SomeController', [ '$http', '$log', function($http, $log){
           } ]);
Dependency Injection!
                                                Injector
                          Here ya go!
                   SomeController
                                                                       $http
                                                                        $log
```



So where were we?

```
(function(){
  var app = angular.module('store', [ 'store-products' ]);
  app.controller('StoreController', function(){
    this.products = ???;
 });
})();
```



Time for your injection!

```
(function(){
  var app = angular.module('store', [ 'store-products' ]);
  app.controller('StoreController', [ '$http',function($http){
    this.products = ???;
 });
                                                       ...so $http
})();
            StoreController needs
                                                    gets injected as an argument!
                the $http Service...
                                                                  app.js
```

Now what?



Lets use our Service!

```
(function(){
 var app = angular.module('store', [ 'store-products' ]);
  app.controller('StoreController', [ '$http',function($http){
   this.products = ???;
   $http.get('/products.json')
 });
                           Fetch the contents of
})();
                            products.json...
```



Our Service will Return Data

```
(function(){
  var app = angular.module('store', [ 'store-products' ]);
  app.controller('StoreController', [ '$http',function($http){
    this.products = ???;
    $http.get('/products.json').success(function(data){
     ??? = data;
   });
                                     $http returns a Promise, so
 });
            What do we assign data to, though...?
})();
                                    success() gets the data...
                                                                  app.js
```



Storing the Data for use in our Page

```
(function(){
 var app = angular.module('store', [ 'store-products' ]);
 app.controller('StoreController', [ '$http',function($http){
    var store = this;
   $http.get('/products.json').success(function(data){
     store.products = data;
   });
 } );
                            -- and now we have somewhere
})();
                                            to put our data!
 We need to store what this is ...
                                                               app.js
```

But the page might look funny until the data loads.

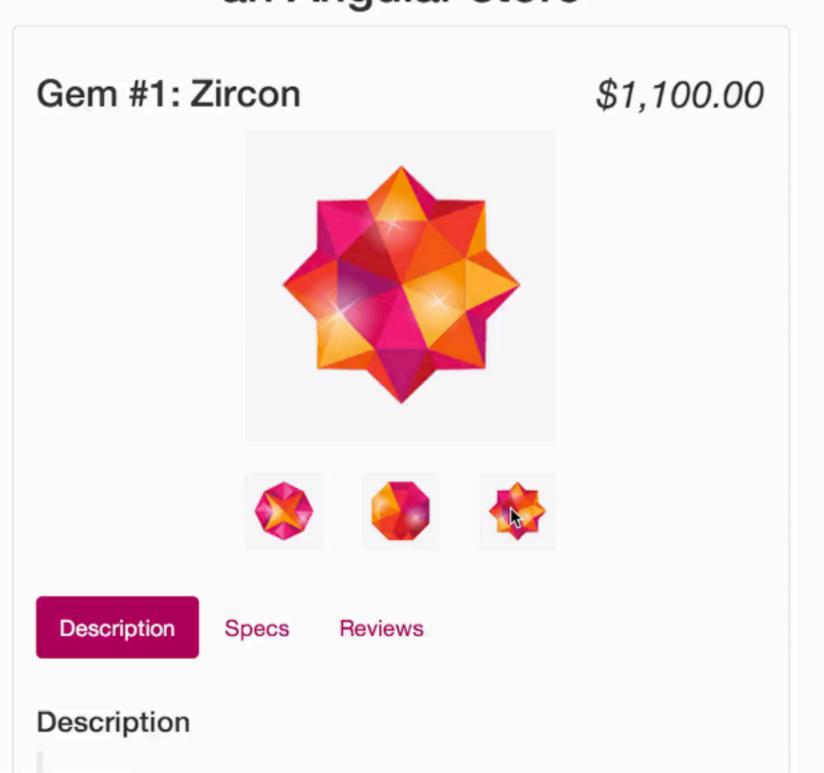


Initialize Products to be a Blank Array

```
(function(){
  var app = angular.module('store', [ 'store-products' ]);
  app.controller('StoreController', [ '$http',function($http){
    var store = this;
   store.products = [];
    $http.get('/products.json').success(function(data){
      store.products = data;
                             We need to initialize products to an empty
   });
                             array, since the page will render before our
 });
                                data returns from our get request.
})();
```

Flatlander Crafted Gems

- an Angular store -





Additional \$http functionality

```
In addition to get() requests, $http can post(), put(), delete()...
 $http.post('/path/to/resource.json', { param: 'value' });
 $http.delete('/path/to/resource.json');
...or any other HTTP method by using a config object:
 $http({ method: 'OPTIONS', url: '/path/to/resource.json' });
 $http({ method: 'PATCH', url: '/path/to/resource.json' });
 $http({ method: 'TRACE', url: '/path/to/resource.json' });
```



Challenges

