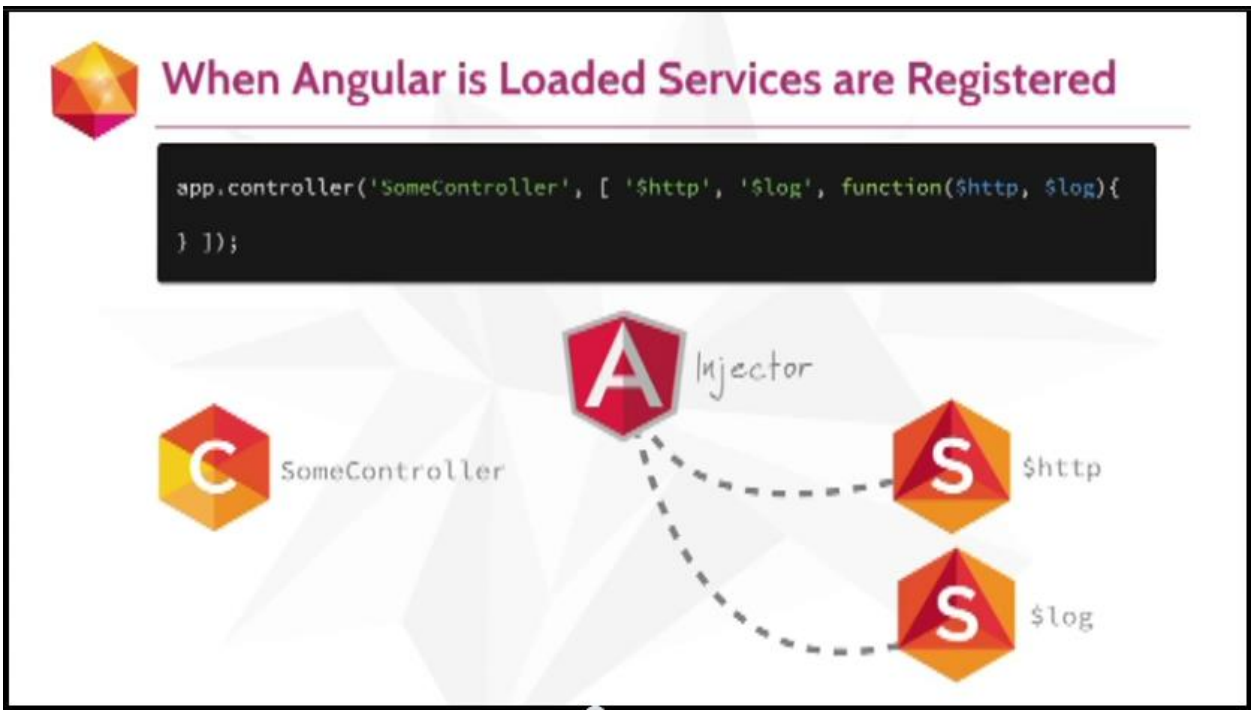
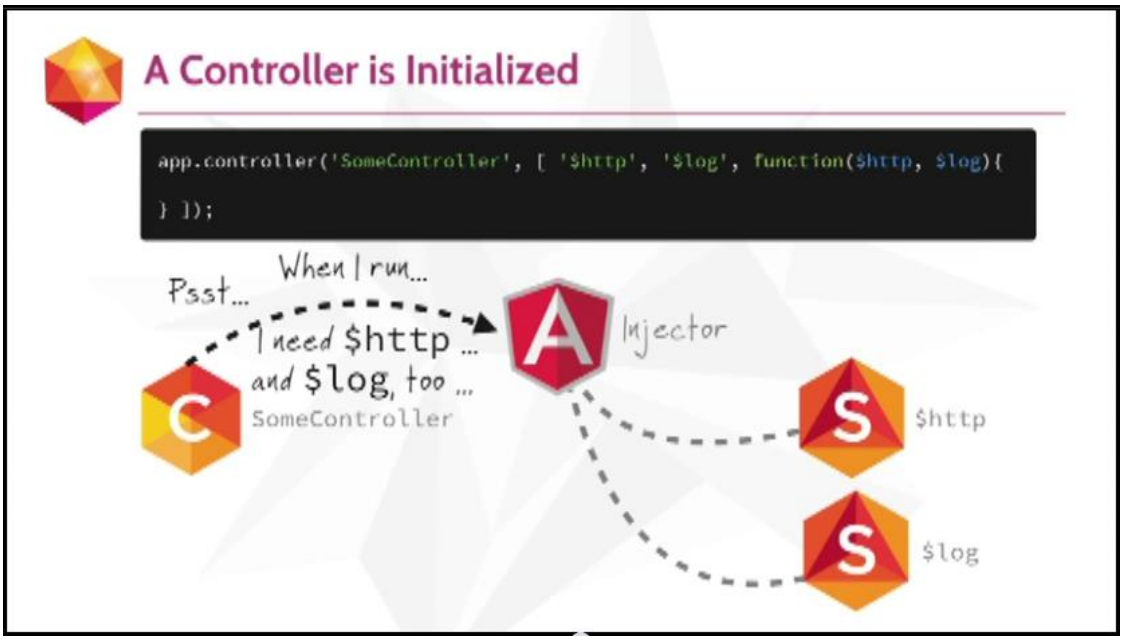
When Angular loads, it creates an Injector:

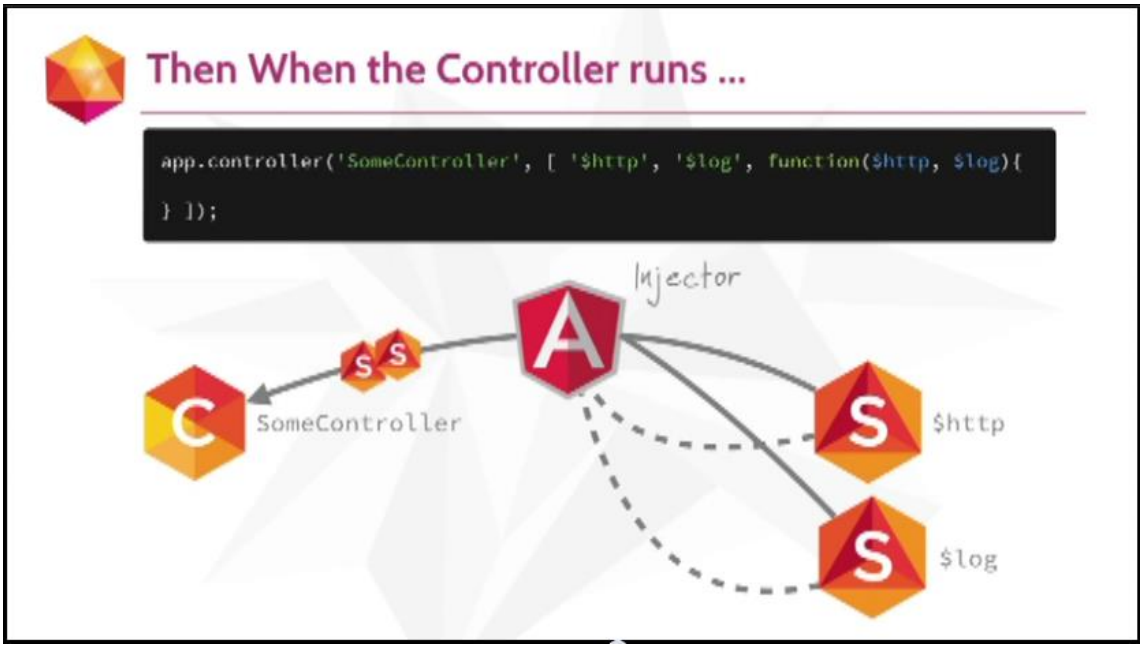
When Angular loads, it creates an Injector. When the built-in services load, they register with the Injector, as being availabe libraries:



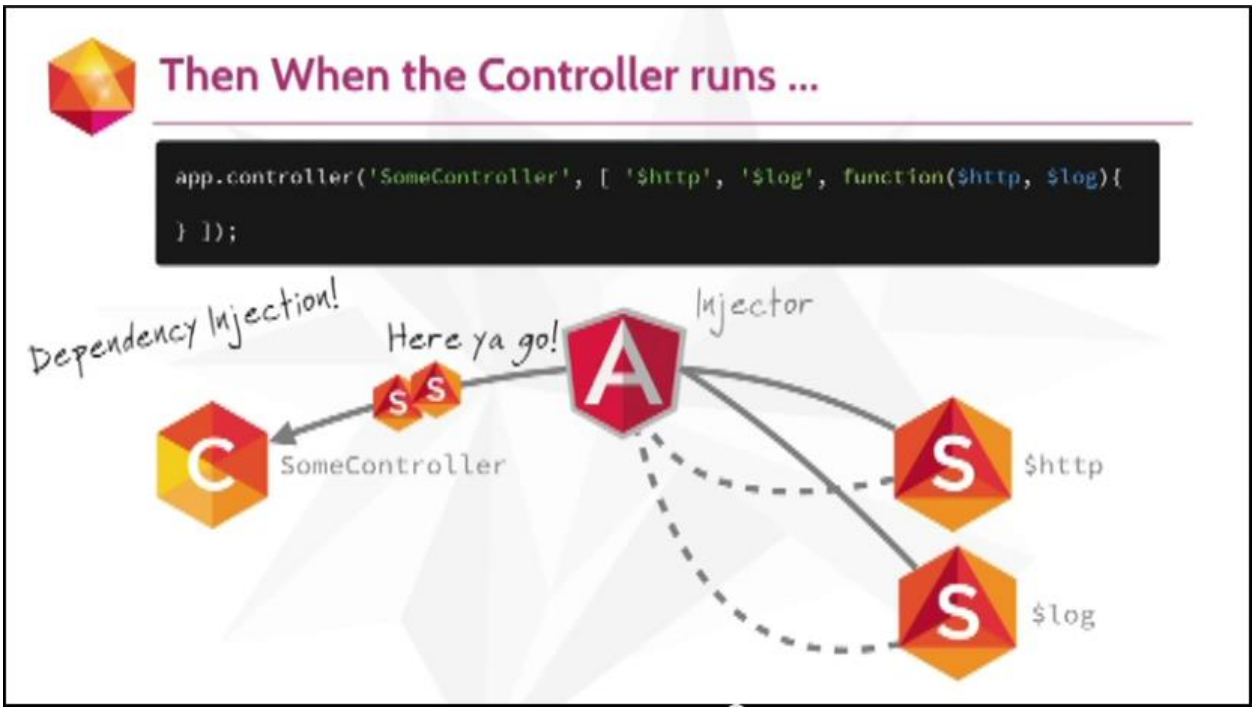
When Angular loads, it creates an Injector. When the built-in services load, they register with the Injector, as being availabe libraries. Whe the application loads, it registers the controller with the injector, telling it when it gets executed it is going to need the $http and $log services:

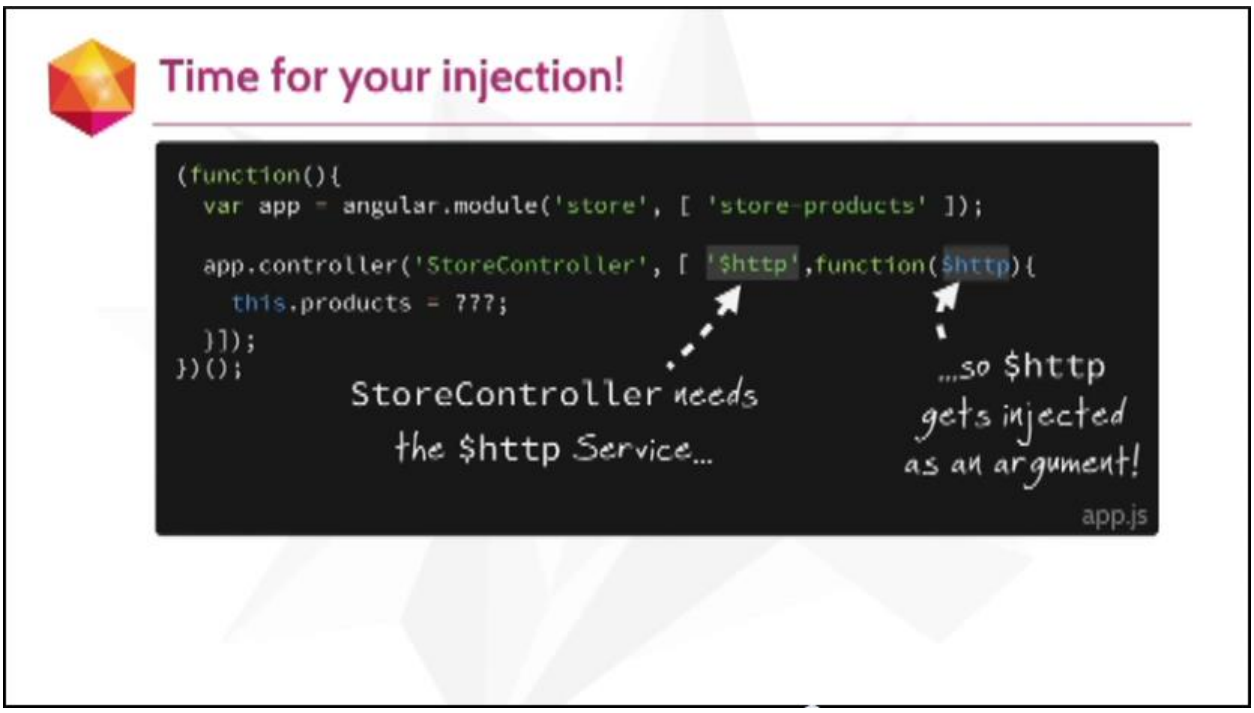


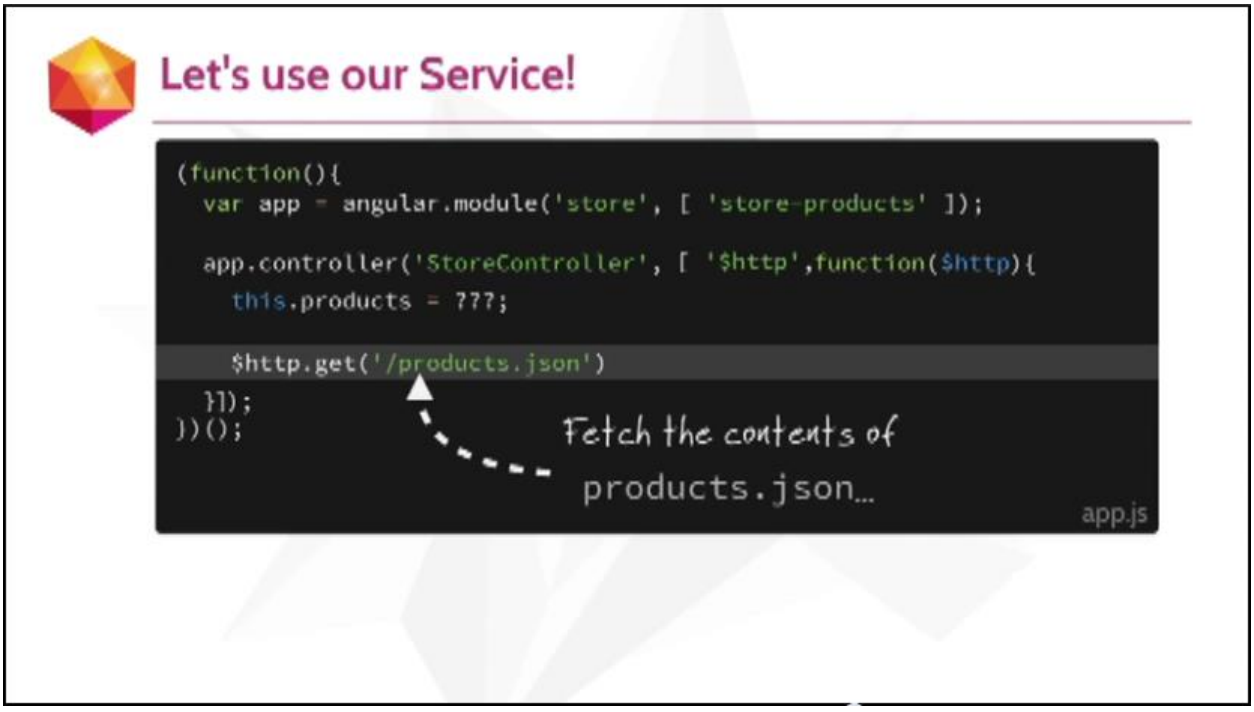
When Angular loads, it creates an Injector. When the built-in services load, they register with the Injector, as being availabe libraries. Whe the application loads, it registers the controller with the injector, telling it when it gets executed it is going to need the $http and $log services. Injector says cool. When a page loads, and when the contoller gets used, the Injector grabs the services that a controller needs and passes them in as arguments:



When Angular loads, it creates an Injector. When the built-in services load, they register with the Injector, as being availabe libraries. Whe the application loads, it registers the controller with the injector, telling it when it gets executed it is going to need the $http and $log services. Injector says cool. When a page loads, and when the contoller gets used, the Injector grabs the services that a controller needs and passes them in as arguments. That is called Dependency Injection! Because it is injecting the dependencies (the services) and to the controller as arguments:




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


Let's 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...
app.js

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;
    });
  }]);
})();
```

app.js

We need to initialize products to an empty array, since the page will render before our data returns from our get request.

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' });
```