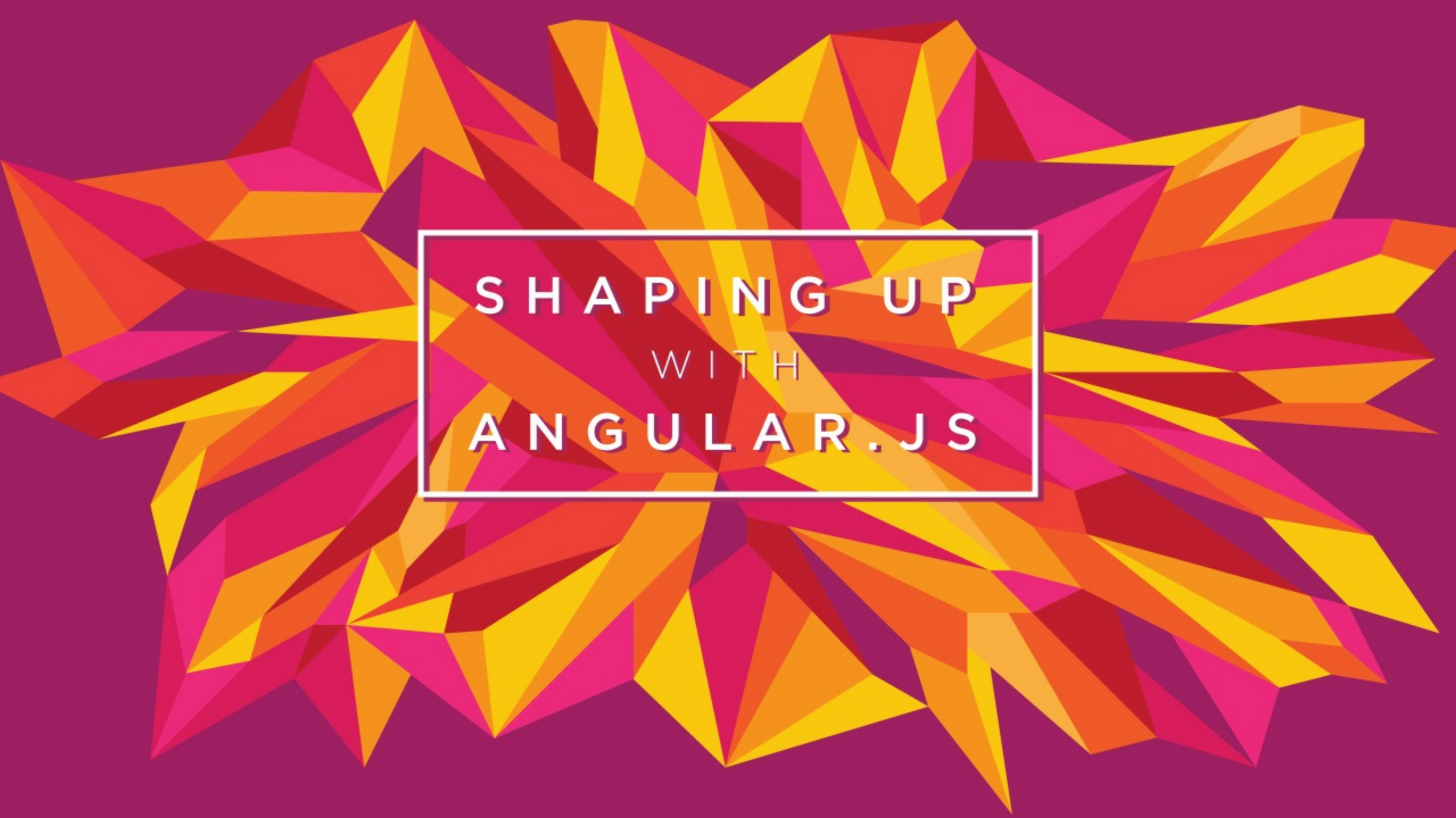

SHAPING UP
WITH
ANGULAR.JS

Shaping Up with Angular JS

Level 5: Dependencies and Services

SHAPING UP
WITH
ANGULAR.JS

Starting to get a bit cluttered?

```
(function(){
  var app = angular.module('store', []);

  app.controller('StoreController', function(){ . . . });

  app.directive('productTitle', function(){ . . . });

  app.directive('productGallery', function(){ . . . });

  app.directive('productPanels', function(){ . . . });

  . . .
})();
```

app.js

Can we refactor these out?

SHAPING UP WITH ANGULAR.JS

Extract into a new file ... products.js

```
(function(){
  var app = angular.module('store', []);

  app.controller('StoreController', function(){ . . . });
  . . .
})();
```

app.js

```
app.directive('productTitle', function(){ . . . });

app.directive('productGallery', function(){ . . . });

app.directive('productPanels', function(){ . . . });
```

products.js

Make a *new* Module

```
(function(){
  var app = angular.module('store', [ ]);

  app.controller('StoreController', function(){ . . . });

  . . .
})();
```

app.js

Define a new module just for Product stuff...

Module Name

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', ['store-products']);

  app.controller('StoreController', function(){ . . . });
  . . .
})();
```

Module Name

app.js

```
(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

SHAPING UP WITH ANGULAR.JS

Flatlander Crafted Gems

– an Angular store –

Gem #1: Zircon *$1,100.00*



Description Specs Reviews

Description

> Zircon is our most coveted and sought after gem

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

SHAPING UP WITH ANGULAR.JS

Challenges

SHAPING UP
WITH
ANGULAR.JS

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, . . . },
      . . .
    ];
    . . .
  });
})();
```

app.js

What is all this data doing here?

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

SHAPING UP WITH ANGULAR.JS

How do we get that data?

```
(function(){
  var app = angular.module('store', [ 'store-products' ]);

  app.controller('StoreController', function(){
    this.products = ???;
    . . .
  });
})();
```

How do we fetch our products from an API?

app.js

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

```
[
  { name: '. . .', price: 1.99, . . . },
  { name: '. . .', price: 1.99, . . . },
  { name: '. . .', price: 1.99, . . . },
  . . .
]
```

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 ...

SHAPING UP WITH ANGULAR.JS

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?

SHAPING UP WITH ANGULAR.JS

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

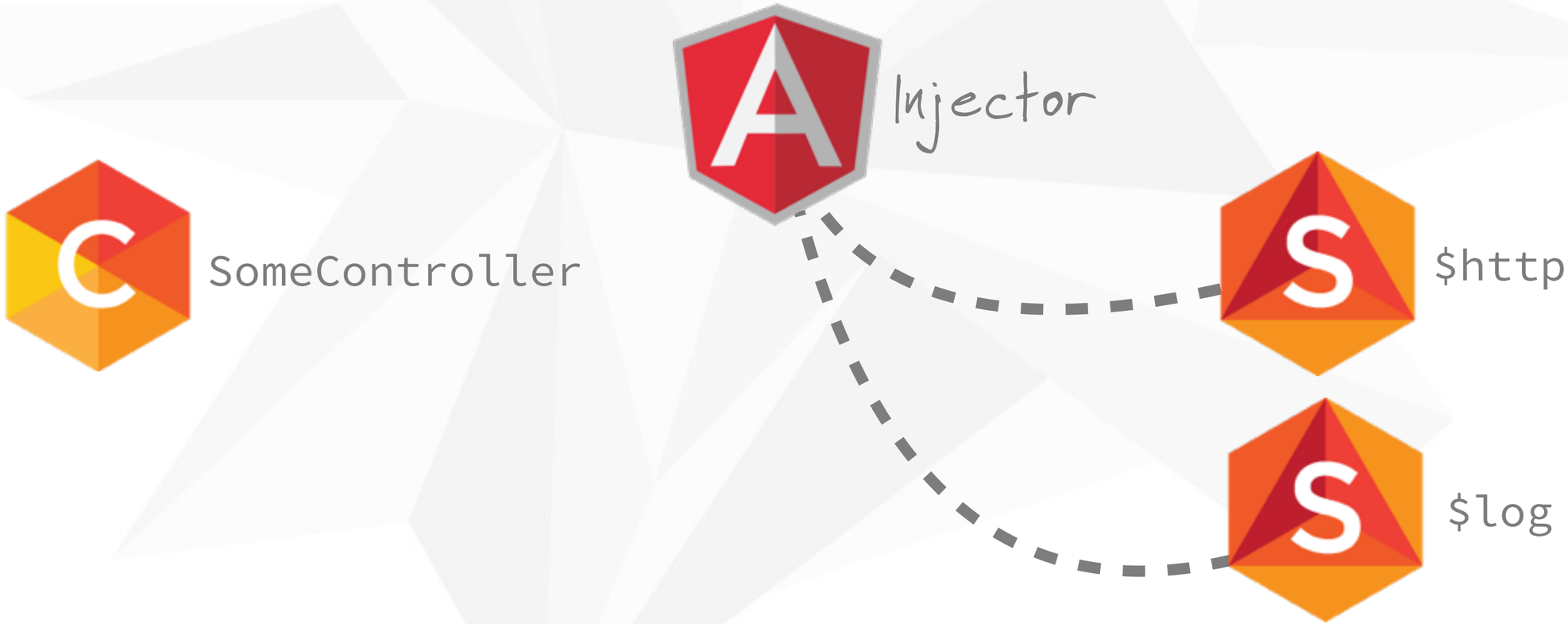
Dependency Injection!

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

} ]);
```

If you needed more than one

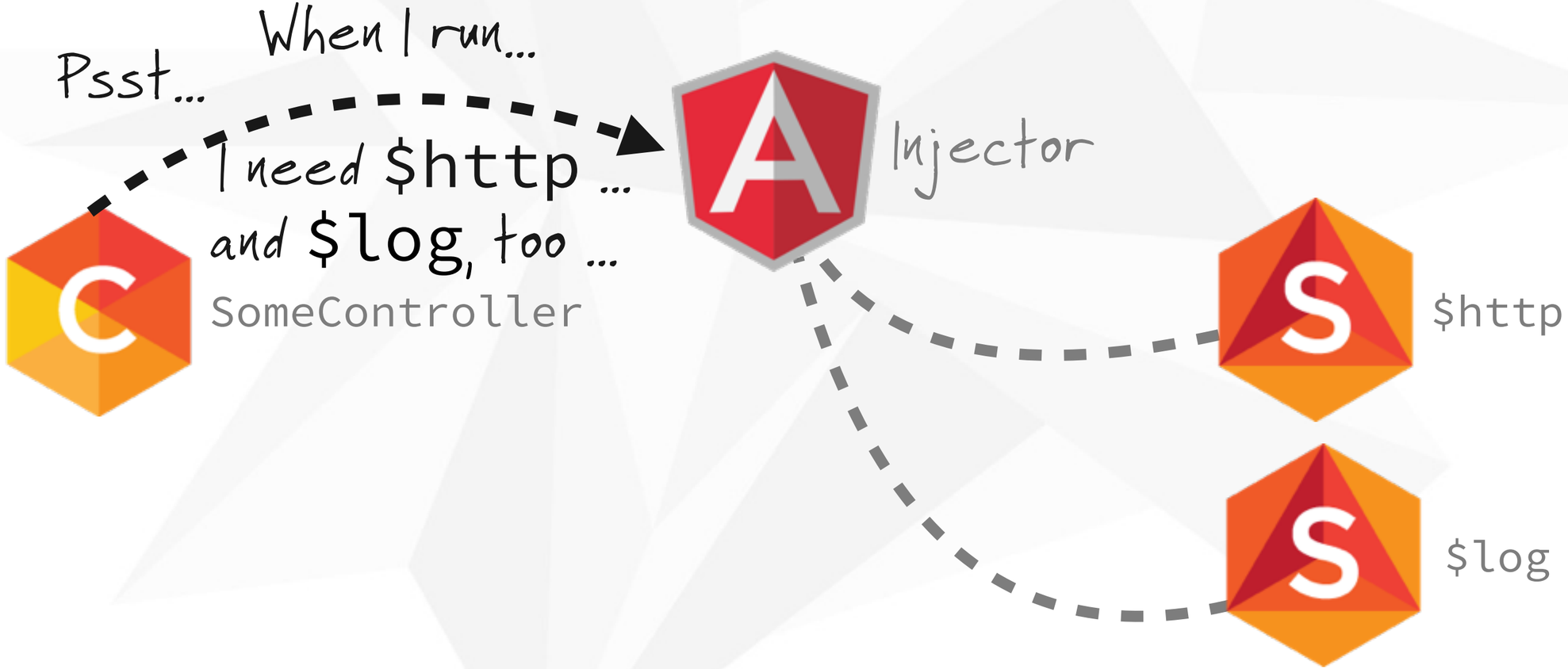
When Angular is Loaded Services are Registered

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

} ]);
```



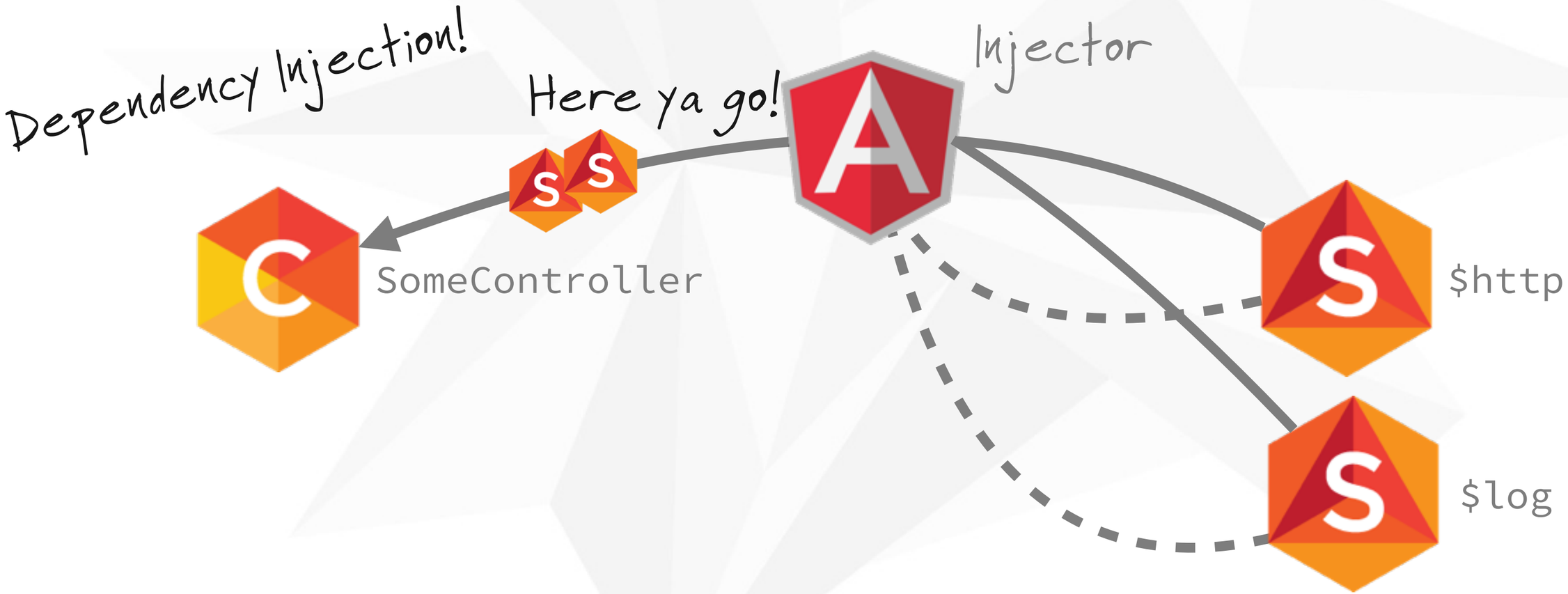
A Controller is Initialized

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

} ]);
```



Then When the Controller runs ...

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

} ]);
```



So where were we?

```
(function(){
  var app = angular.module('store', [ 'store-products' ]);

  app.controller('StoreController', function(){
    this.products = ???;
  } );
})();
```

app.js

SHAPING UP WITH ANGULAR.JS

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

Now what?

SHAPING UP WITH ANGULAR.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

SHAPING UP WITH ANGULAR.JS

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

What do we assign data to, though...?

$http returns a Promise, so success() gets the data...

app.js

SHAPING UP WITH ANGULAR.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;
    });
  }]);
})();
```

app.js

We need to store what this is ...

... and now we have somewhere to put our data!

But the page might look funny until the data loads.

SHAPING UP WITH ANGULAR.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;
    });
  }]);
})();
```

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

app.js

SHAPING UP WITH ANGULAR.JS

Flatlander Crafted Gems

– an Angular store –

Gem #1: Zircon *$1,100.00*

Description Specs Reviews

Description

Zircon is our most coveted and sought after gem

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:

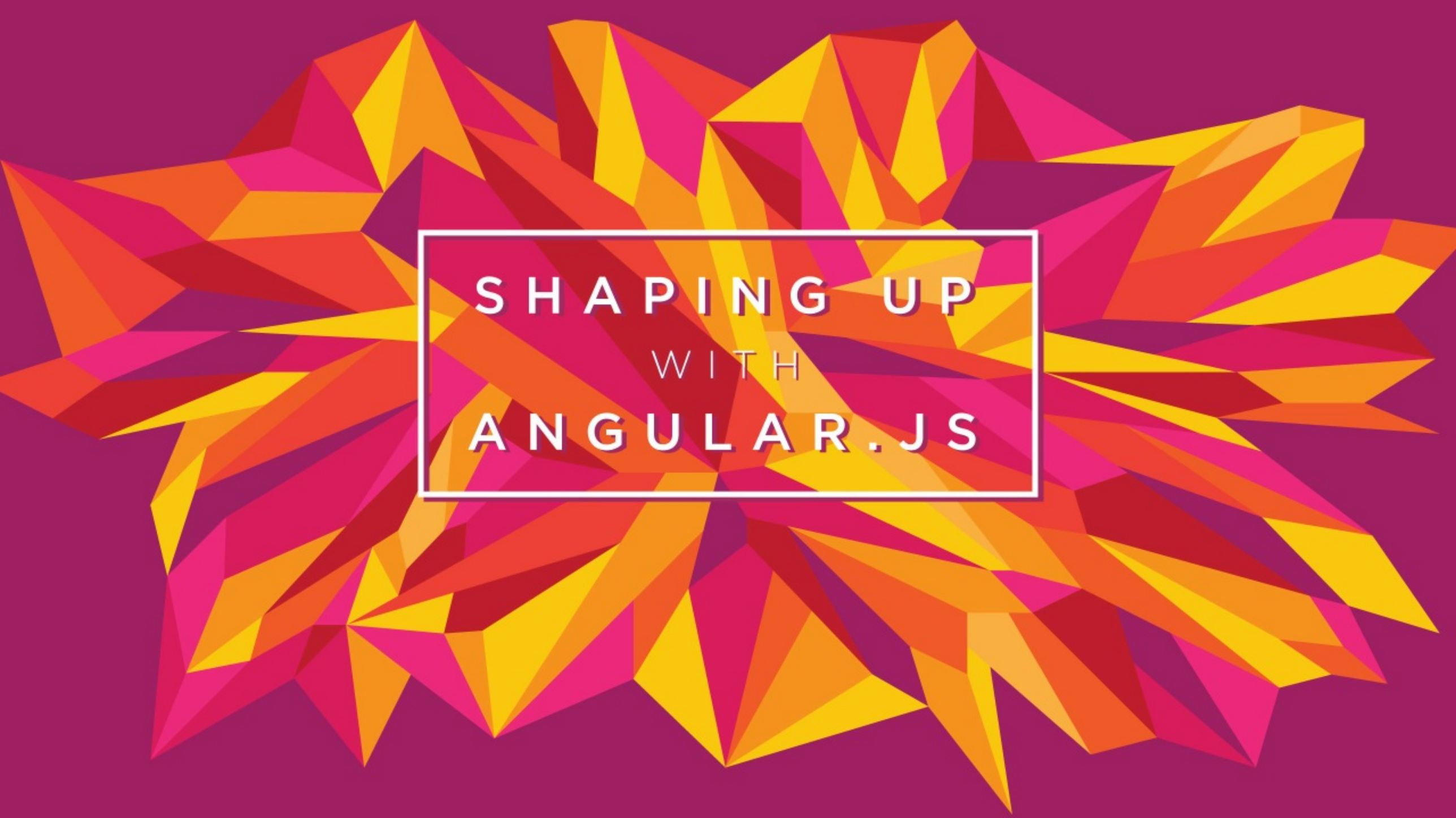
```
$http({ method: 'OPTIONS', url: '/path/to/resource.json' });
```

```
$http({ method: 'PATCH', url: '/path/to/resource.json' });
```

```
$http({ method: 'TRACE', url: '/path/to/resource.json' });
```

Challenges

SHAPING UP
WITH
ANGULAR.JS


SHAPING UP
WITH
ANGULAR.JS