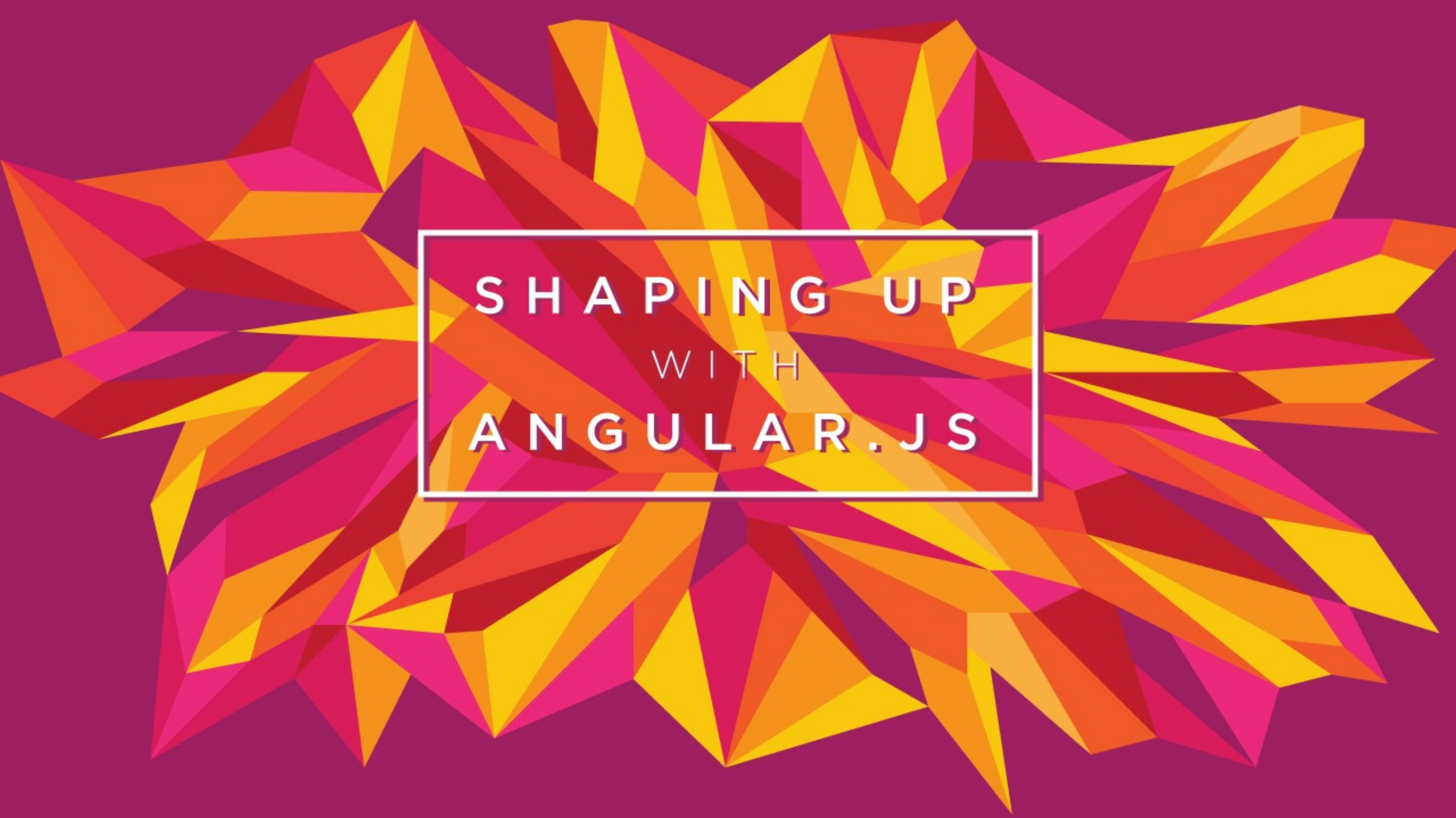

SHAPING UP
WITH
ANGULAR.JS

Shaping Up with Angular

Level 1: Getting Started

SHAPING UP
WITH
ANGULAR.JS

What you need to know

Must know

HTML & CSS
JavaScript

Nice to know

Automated Testing
BDD - Behavior Driven Development
TDD - Test Driven Development
etc

Not so important

jQuery
Ruby on Rails
Python, PHP, etc
Databases

Why  Angular?

If you're using JavaScript to create a dynamic website, Angular is a good choice.

- Angular helps you organize your JavaScript
- Angular helps create responsive (as in fast) websites.
- Angular plays well with jQuery
- Angular is easy to test

Traditional Page-Refresh

Web Server

Web Browser

URL Request to server

Response with Webpage & Assets

HTML JavaScript

Browser loads up entire webpage.

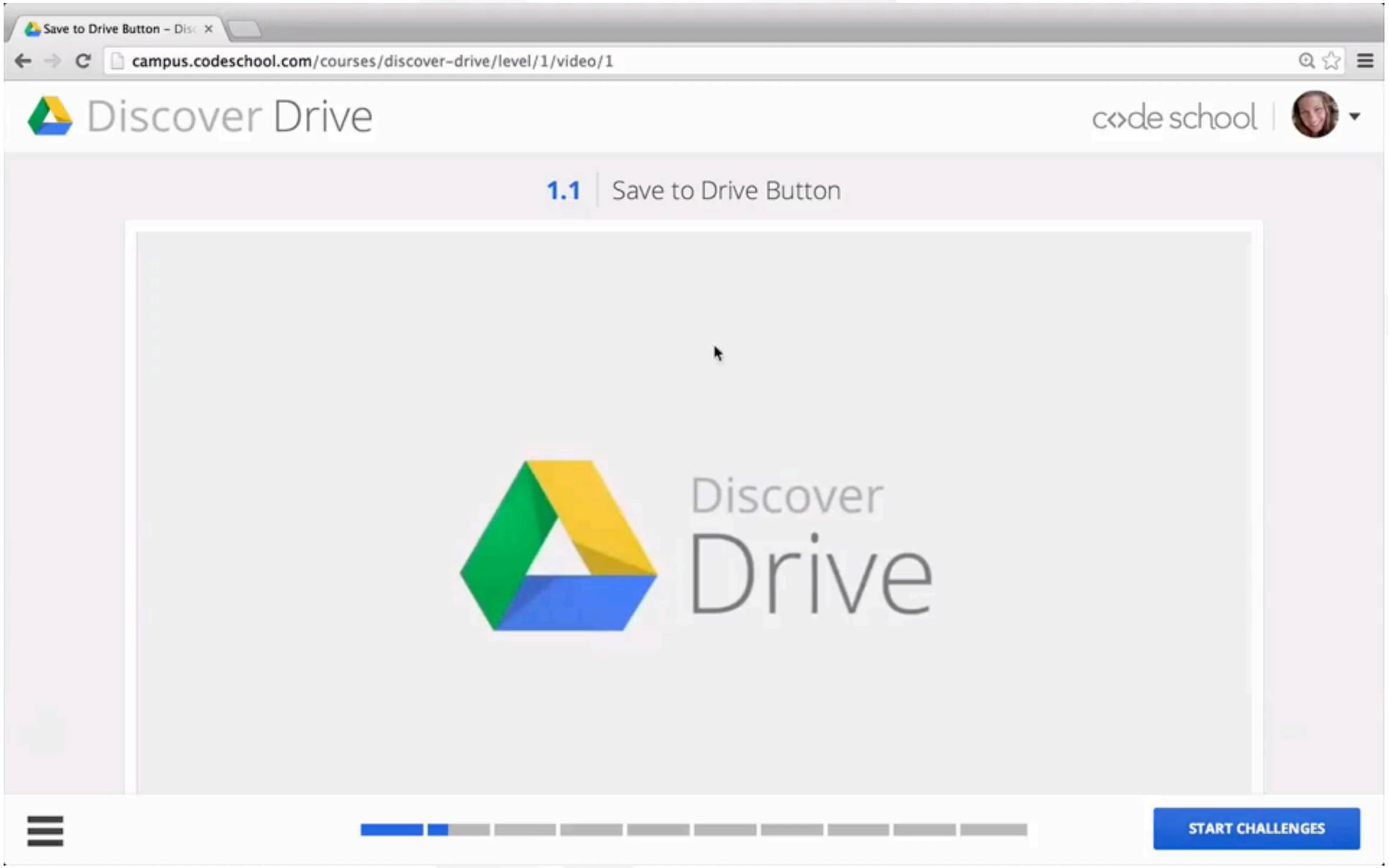
User clicks on link, new Request

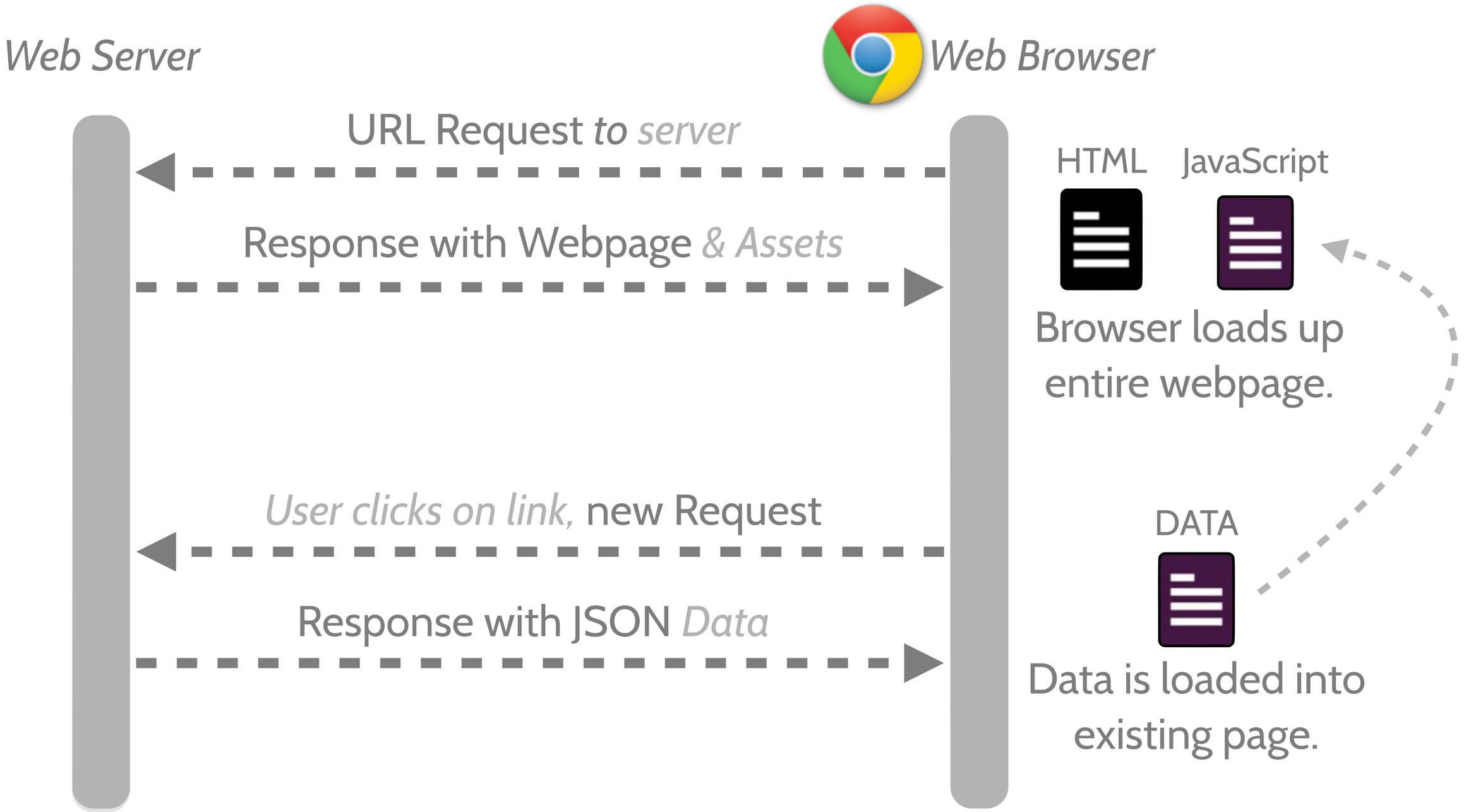
Response with Webpage & Assets

HTML JavaScript

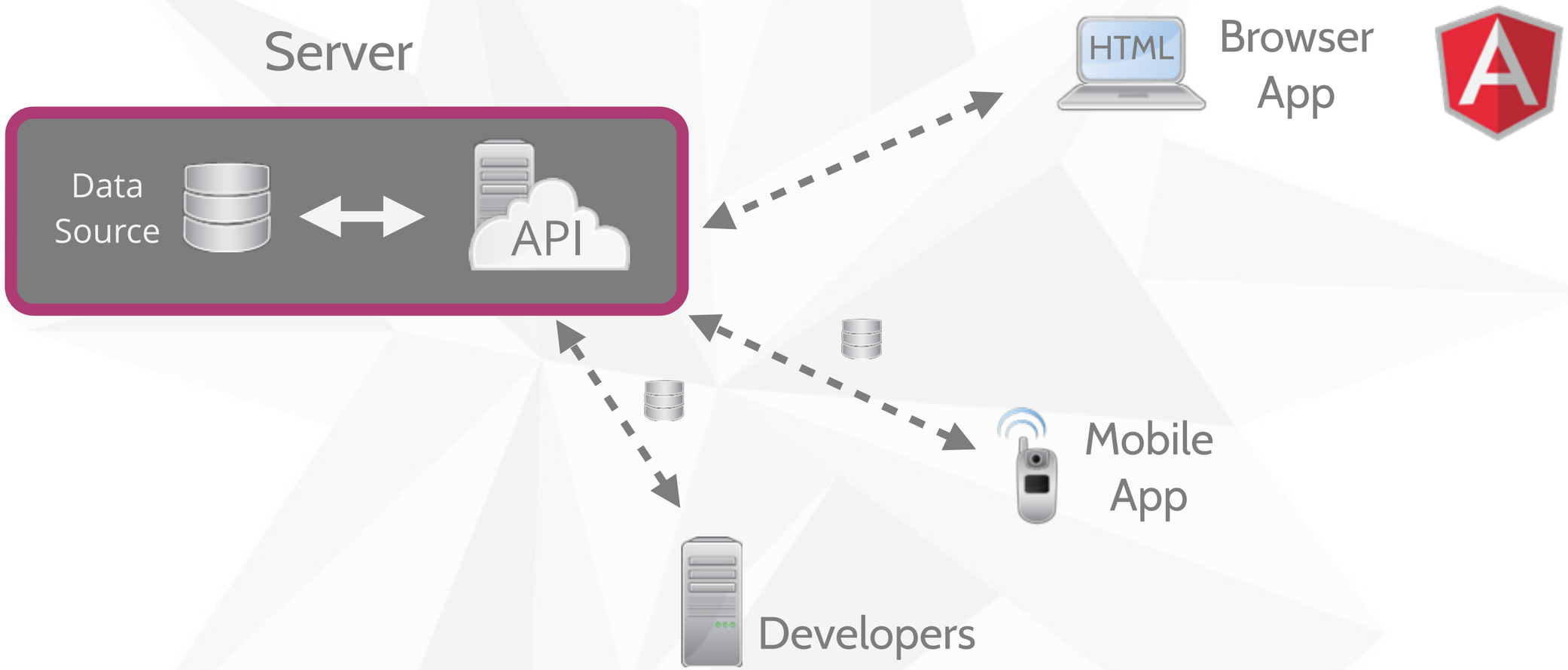
Browser loads up entire webpage.

A “responsive” website using Angular




Web Server
Web Browser
URL Request to server
Response with Webpage & Assets
HTML
JavaScript
Browser loads up entire webpage.
User clicks on link, new Request
DATA
Response with JSON Data
Data is loaded into existing page.

Modern API-Driven Application



What is Angular JS?

A client-side JavaScript Framework for adding interactivity to HTML.

How do we tell our **HTML** when to trigger our **JavaScript**?

```
<!DOCTYPE html>
<html>
  <body>
  . . .
  </body>
</html>
```

index.html

```
function Store(){
  alert('Welcome, Gregg!');
}
```

app.js

SHAPING UP WITH ANGULAR.JS

Directives

A **Directive** is a marker on a **HTML tag** that tells Angular to run or reference some **JavaScript code**.

```
<!DOCTYPE html>
<html>
  <body ng-controller="StoreController">
  . . .
  </body>
</html>
```

index.html

```
function StoreController(){
  alert('Welcome, Gregg!');
}
```

Name of function to call

app.js

The page at https://www.codeschool.com says:

Welcome, Gregg!

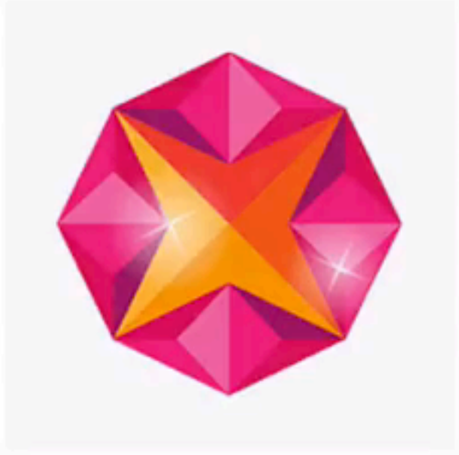
OK

SHAPING UP WITH ANGULAR.JS

Flatlander Crafted Gems

– an Angular store –

Gem #1: Zircon *$1,100.00*



Description Specs Reviews

Description

n is our most coveted and sought after gem

Downloading the libraries

Download AngularJS *http://angularjs.org/*

We'll need **angular.min.js**

Download Twitter Bootstrap *http://getbootstrap.com/*

We'll need **bootstrap.min.css**

SHAPING UP
WITH
ANGULAR.JS

Getting Started

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="bootstrap.min.css" />
  </head>
  <body>
    <script type="text/javascript" src="angular.min.js"></script>
  </body>
</html>
```

Twitter Bootstrap

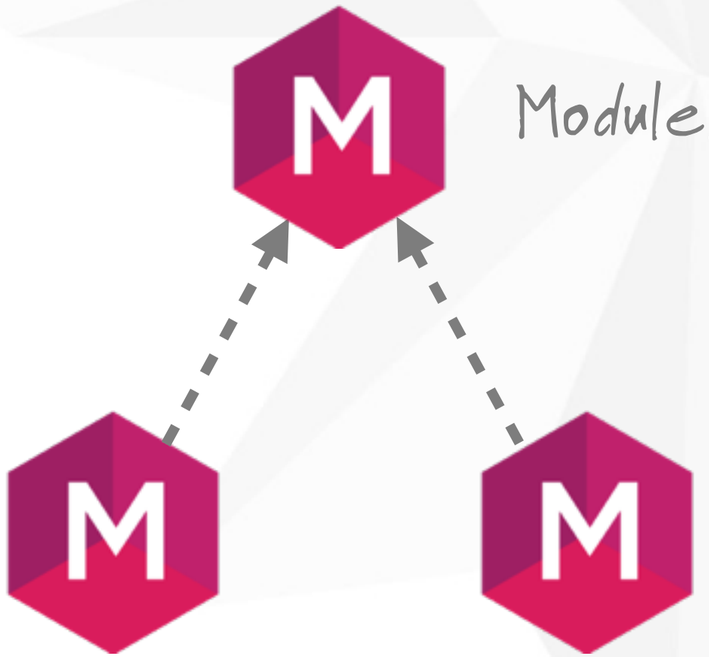
AngularJS

index.html

SHAPING UP WITH ANGULAR.JS

Modules

- Where we write pieces of our Angular application.
- Makes our code more maintainable, testable, and readable.
- Where we define dependencies for our app.



SHAPING UP WITH ANGULAR.JS

Creating Our First Module

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

AngularJS

Application Name

Dependencies

Other libraries we might need.
We have none... for now...

SHAPING UP WITH ANGULAR.JS

Including Our Module

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="bootstrap.min.css" />
  </head>
  <body>
    <script type="text/javascript" src="angular.min.js"></script>
    <script type="text/javascript" src="app.js"></script>
  </body>
</html>
```

index.html

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

app.js

SHAPING UP WITH ANGULAR.JS

Including Our Module

```
<!DOCTYPE html>
<html ng-app="store">
  <head>
    <link rel="stylesheet" type="text/css" href="bootstrap.min.css" />
  </head>
  <body>
    <script type="text/javascript" src="angular.min.js"></script>
    <script type="text/javascript" src="app.js"></script>
  </body>
</html>
```

Run this module when the document loads

index.html

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

app.js

SHAPING UP WITH ANGULAR.JS

Expressions

Allow you to insert dynamic values into your HTML.

Numerical Operations

```
<p>
  I am {{4 + 6}}
</p>
```

evaluates to

```
<p>
  I am 10
</p>
```

String Operations

```
<p>
  {{"hello" + " you"}}
</p>
```

evaluates to

```
<p>
  hello you
</p>
```

+ More Operations:
  http://docs.angularjs.org/guide/expression

SHAPING UP WITH ANGULAR.JS

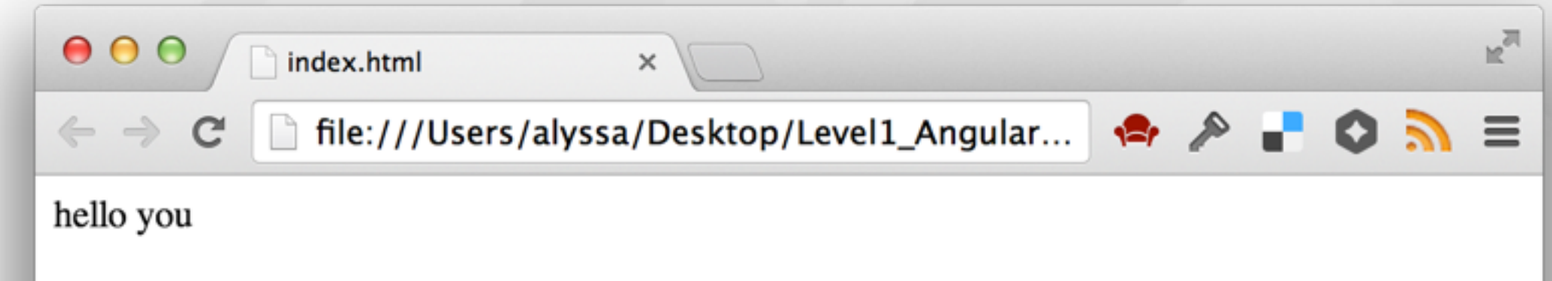
Including Our Module

```
<!DOCTYPE html>
<html ng-app="store">
  <head>
    <link rel="stylesheet" type="text/css" href="bootstrap.min.css" />
  </head>
  <body>
    <script type="text/javascript" src="angular.min.js"></script>
    <script type="text/javascript" src="app.js"></script>
    <p>{{"hello" + " you"}}</p>
  </body>
</html>
```

index.html

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

app.js



SHAPING UP WITH ANGULAR.JS

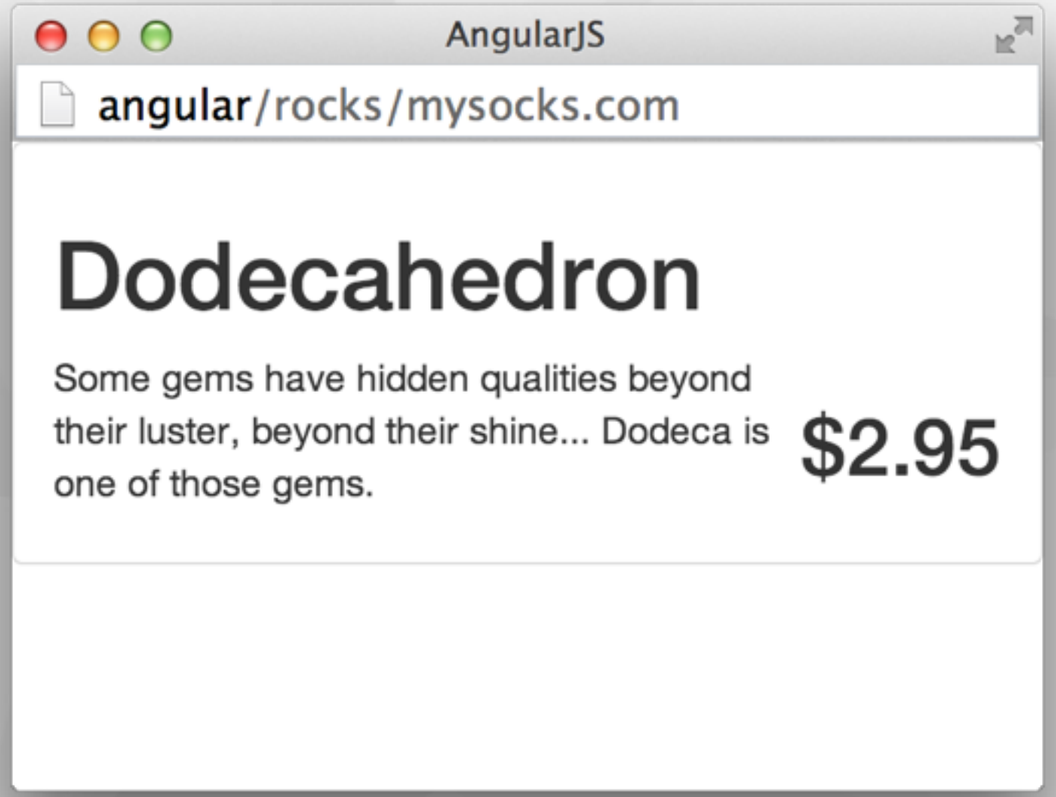
Challenges

SHAPING UP
WITH
ANGULAR.JS

Working With Data

```
var gem = {
  name: 'Dodecahedron',
  price: 2.95,
  description: '. . .',
}
```

...just a simple object we want to print to the page.



SHAPING UP WITH ANGULAR.JS

Controllers

Controllers are where we define our app's behavior by defining functions and values.

Wrapping your Javascript in a closure is a good habit!

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

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

  });
})();
```

app.js

```
var gem = {
  name: 'Dodecahedron',
  price: 2.95,
  description: '. . .',
}
```

Notice that controller is attached to (inside) our app.

SHAPING UP WITH ANGULAR.JS

Storing Data Inside the Controller

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

  app.controller('StoreController', function(){
    this.product = gem;
  });

  var gem = {
    name: 'Dodecahedron',
    price: 2.95,
    description: '. . .',
  }
})();
```

app.js

Now how do we print out this data inside our webpage?

SHAPING UP WITH ANGULAR.JS

Our Current HTML

```
<!DOCTYPE html>
<html ng-app="store">
  <head>
    <link rel="stylesheet" type="text/css" href="bootstrap.min.css" />
  </head>
  <body>
    <div>
      <h1> Product Name </h1>
      <h2> $Product Price </h2>
      <p> Product Description </p>
    </div>
    <script type="text/javascript" src="angular.min.js"></script>
    <script type="text/javascript" src="app.js"></script>
  </body>
</html>
```

Let's load our data into this part of the page.

index.html

Attaching the Controller

```
<body>
  <div>
    <h1> Product Name </h1>
    <h2> $Product Price </h2>
    <p> Product Description </p>
  </div>
  <script type="text/javascript" src="angular.min.js"></script>
  <script type="text/javascript" src="app.js"></script>
</body>
```

index.html

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

  app.controller('StoreController', function(){
    this.product = gem;
  });
  . . .
})();
```

app.js

SHAPING UP WITH ANGULAR.JS

Attaching the Controller

Directive Controller name Alias

```
<body>
  <div ng-controller="StoreController as store">
    <h1> Product Name </h1>
    <h2> $Product Price </h2>
    <p> Product Description </p>
  </div>
  <script type="text/javascript" src="angular.min.js"></script>
  <script type="text/javascript" src="app.js"></script>
</body>
```

index.html

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

  app.controller('StoreController', function(){
    this.product = gem;
  });
  . . .
})();
```

app.js

SHAPING UP WITH ANGULAR.JS

Displaying Our First Product

```
<body>
  <div ng-controller="StoreController as store">
    <h1> {{store.product.name}} </h1>
    <h2> ${{store.product.price}} </h2>
    <p> {{store.product.description}} </p>
  </div>
  <script type="text/javascript" src="angular.min.js"></script>
  <script type="text/javascript" src="app.js"></script>
</body>
```

index.html

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

  app.controller('StoreController', function(){
    this.product = gem;
  });
  . . .
})();
```

Understanding Scope

```
<body>
  <div ng-controller="StoreController as store">
     <h1> {{store.product.name}} </h1>
     <h2> ${{store.product.price}} </h2>
     <p> {{store.product.description}} </p>
  </div>
  {{store.product.name}}
  <script type="text/javascript" src="angular.min.js"></script>
  <script type="text/javascript" src="app.js"></script>
</body>
```

The scope of the Controller is only inside here...

Would never print a value!

SHAPING UP WITH ANGULAR.JS

Challenges

SHAPING UP
WITH
ANGULAR.JS

Adding A Button

```
<body ng-controller="StoreController as store">
  <div>
    <h1> {{store.product.name}} </h1>
    <h2> ${{store.product.price}} </h2>
    <p> {{store.product.description}} </p>

  </div>
  <script type="text/javascript" src="angular.min.js"></script>
  <script type="text/javascript" src="app.js"></script>
</body>
```

index.html

```
var gem = {
  name: 'Dodecahedron',
  price: 2.95,
  description: '. . .',
}
```

SHAPING UP WITH ANGULAR.JS

Adding A Button

```
<body ng-controller="StoreController as store">
  <div>
    <h1> {{store.product.name}} </h1>
    <h2> ${{store.product.price}} </h2>
    <p> {{store.product.description}} </p>
    <button> Add to Cart </button>
  </div>
  <script type="text/javascript" src="angular.min.js"></script>
  <script type="text/javascript" src="app.js"></script>
</body>
```

index.html

Directives to the rescue!

How can we only show this button...

```
var gem = {
  name: 'Dodecahedron',
  price: 2.95,
  description: '. . .',
  canPurchase: false
}
```

...when this is true?

SHAPING UP WITH ANGULAR.JS

NgShow Directive

```
<body ng-controller="StoreController as store">
  <div>
    <h1> {{store.product.name}} </h1>
    <h2> ${{store.product.price}} </h2>
    <p> {{store.product.description}} </p>
    <button ng-show="store.product.canPurchase"> Add to Cart </button>
  </div>
  <script type="text/javascript" src="angular.min.js"></script>
  <script type="text/javascript" src="app.js"></script>
</body>
```

index.html

```
var gem = {
  name: 'Dodecahedron',
  price: 2.95,
  description: '. . .',
  canPurchase: false
}
```

Will only show the element if the value of the Expression is **true**.

SHAPING UP WITH ANGULAR.JS

```
10    <script data-require="angular.js@1.2.x" src="http://code.angularjs.org/1.2.15/angul
11    <script src="app.js"></script>
12  </head>
13
14
15  <body ng-controller="StoreController as store">
16
17    <!--  Products Container  -->
18    <div class="list-group">
19      <!--  Product Container  -->
20      <div class="list-group-item">
21        <h1>{{store.product.name}}</h1>
22        <h2>${{store.product.price}}</h2>
23        <p>{{store.product.description}}</p>
24        <button ng-show="store.product.canPurchase">Add to Cart</button>
25      </div>
26    </div>
27  </body>
28
29  </html>
```

their shine... Dodeca i

index.html

NgHide Directive

```
<body ng-controller="StoreController as store">
  <div>
    <h1> {{store.product.name}} </h1>
    <h2> ${{store.product.price}} </h2>
    <p> {{store.product.description}} </p>
    <button ng-show="store.product.canPurchase"> Add to Cart </button>
  </div>
  <script type="text/javascript" src="angular.min.js"></script>
  <script type="text/javascript" src="app.js"></script>
</body>
```

index.html

```
var gem = {
  name: 'Dodecahedron',
  price: 2.95,
  description: '. . .',
  canPurchase: true,
  soldOut: true
}
```

If the product is sold out we want to hide it.

SHAPING UP WITH ANGULAR.JS

NgHide Directive

```
<body ng-controller="StoreController as store">
  <div ng-show="!store.product.soldOut">
    <h1> {{store.product.name}} </h1>
    <h2> ${{store.product.price}} </h2>
    <p> {{store.product.description}} </p>
    <button ng-show="store.product.canPurchase"> Add to Cart </button>
  </div>
  <script type="text/javascript" src="angular.min.js"></script>
  <script type="text/javascript" src="app.js"></script>
</body>
```

index.html

This is awkward and a good example to use ng-hide!

```
var gem = {
  name: 'Dodecahedron',
  price: 2.95,
  description: '. . .',
  canPurchase: true,
  soldOut: true,
}
```

If the product is sold out we want to hide it.

SHAPING UP WITH ANGULAR.JS

NgHide Directive

```
<body ng-controller="StoreController as store">
  <div ng-hide="store.product.soldOut">
    <h1> {{store.product.name}} </h1>
    <h2> ${{store.product.price}} </h2>
    <p> {{store.product.description}} </p>
    <button ng-show="store.product.canPurchase"> Add to Cart </button>
  </div>
  <script type="text/javascript" src="angular.min.js"></script>
  <script type="text/javascript" src="app.js"></script>
</body>
```

index.html

Much better!

```
var gem = {
  name: 'Dodecahedron',
  price: 2.95,
  description: '. . .',
  canPurchase: true,
  soldOut: true,
}
```

If the product is sold out we want to hide it.

SHAPING UP WITH ANGULAR.JS

Multiple Products

```
app.controller('StoreController', function(){
  this.product = gem;
});


var gem = {
  name: "Dodecahedron",
  price: 2.95,
  description: ". . .",
  canPurchase: true,
}
```

app.js

SHAPING UP WITH ANGULAR.JS

Multiple Products

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

var gems = [
  {
    name: "Dodecahedron",
    price: 2.95,
    description: ". . .",
    canPurchase: true,
  },
  {
    name: "Pentagonal Gem",
    price: 5.95,
    description: ". . .",
    canPurchase: false,
  }…
];
```

So we have multiple products...

Now we have an array...

Maybe a Directive?

How might we display all these products in our template?

app.js

Working with An Array

```
<body ng-controller="StoreController as store">
  <div>
    <h1> {{store.products[0].name}} </h1>
    <h2> ${{store.products[0].price}} </h2>
    <p> {{store.products[0].description}} </p>
    <button ng-show="store.products[0].canPurchase">
      Add to Cart</button>
  </div>
  . . .
</body>
```

index.html

Working with An Array

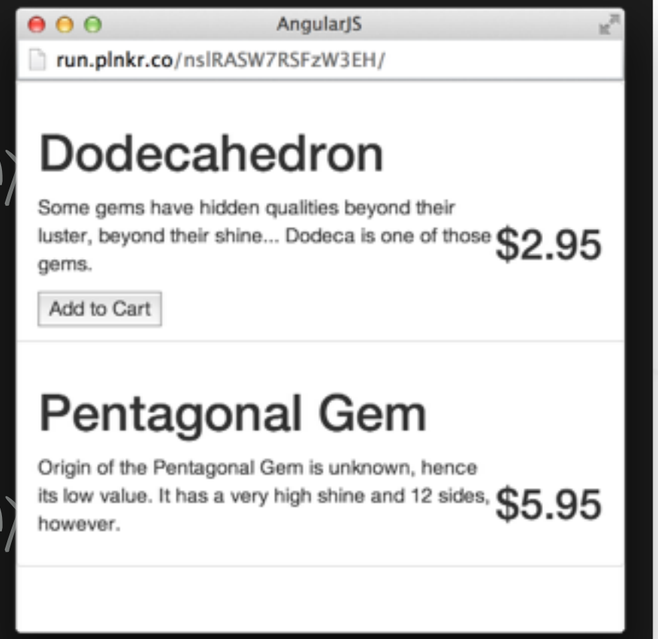
```
<body ng-controller="StoreController as store">
  <div>
    <h1> {{store.products[0].name}} </h1>
    <h2> ${{store.products[0].price}} </h2>
    <p> {{store.products[0].description}} </p>
    <button ng-show="store.products[0].canPurchase">
      Add to Cart</button>
  </div>
  . . .
</body>
```

Displaying the first product is easy enough...

index.html

Working with An Array

```
<body ng-controller="StoreController as store">
  <div>
    <h1> {{store.products[0].name}} </h1>
    <h2> ${{store.products[0].price}} </h2>
    <p> {{store.products[0].description}} </p>
    <button ng-show="store.products[0].canPurchase">
      Add to Cart</button>
  </div>
  <div>
    <h1> {{store.products[1].name}} </h1>
    <h2> ${{store.products[1].price}} </h2>
    <p> {{store.products[1].description}} </p>
    <button ng-show="store.products[1].canPurchase">
      Add to Cart</button>
  </div>
  . . .
</body>
```



Why

Why

That works...

Why... You get it.

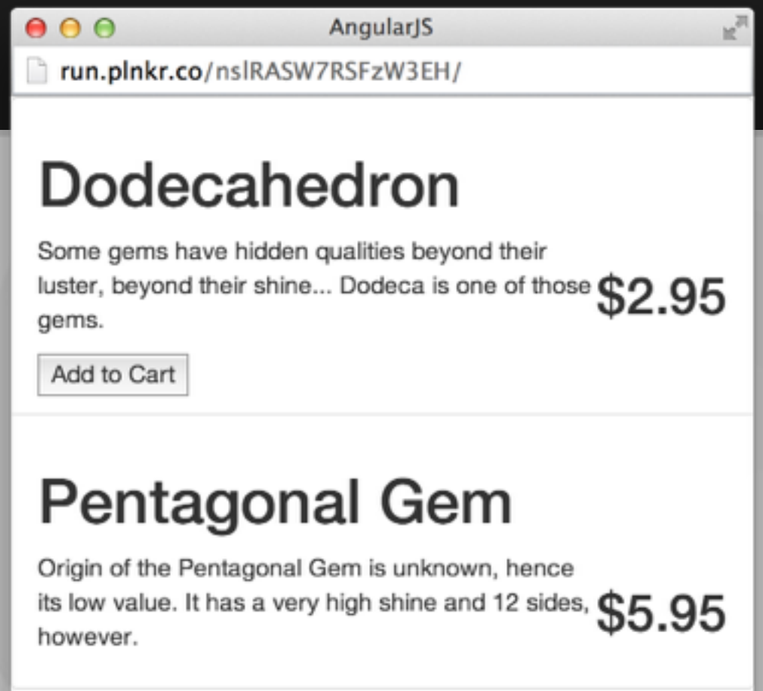
index.html

Working with An Array

```
<body ng-controller="StoreController as store">
  <div ng-repeat="product in store.products">
    <h1> {{product.name}} </h1>
    <h2> ${{product.price}} </h2>
    <p> {{product.description}} </p>
    <button ng-show="product.canPurchase">
      Add to Cart</button>
  </div>
  . . .
</body>
```

Repeat this section for each product.

index.html



What We Have Learned So Far

Directives – HTML annotations that trigger Javascript behaviors

Modules – Where our application components live

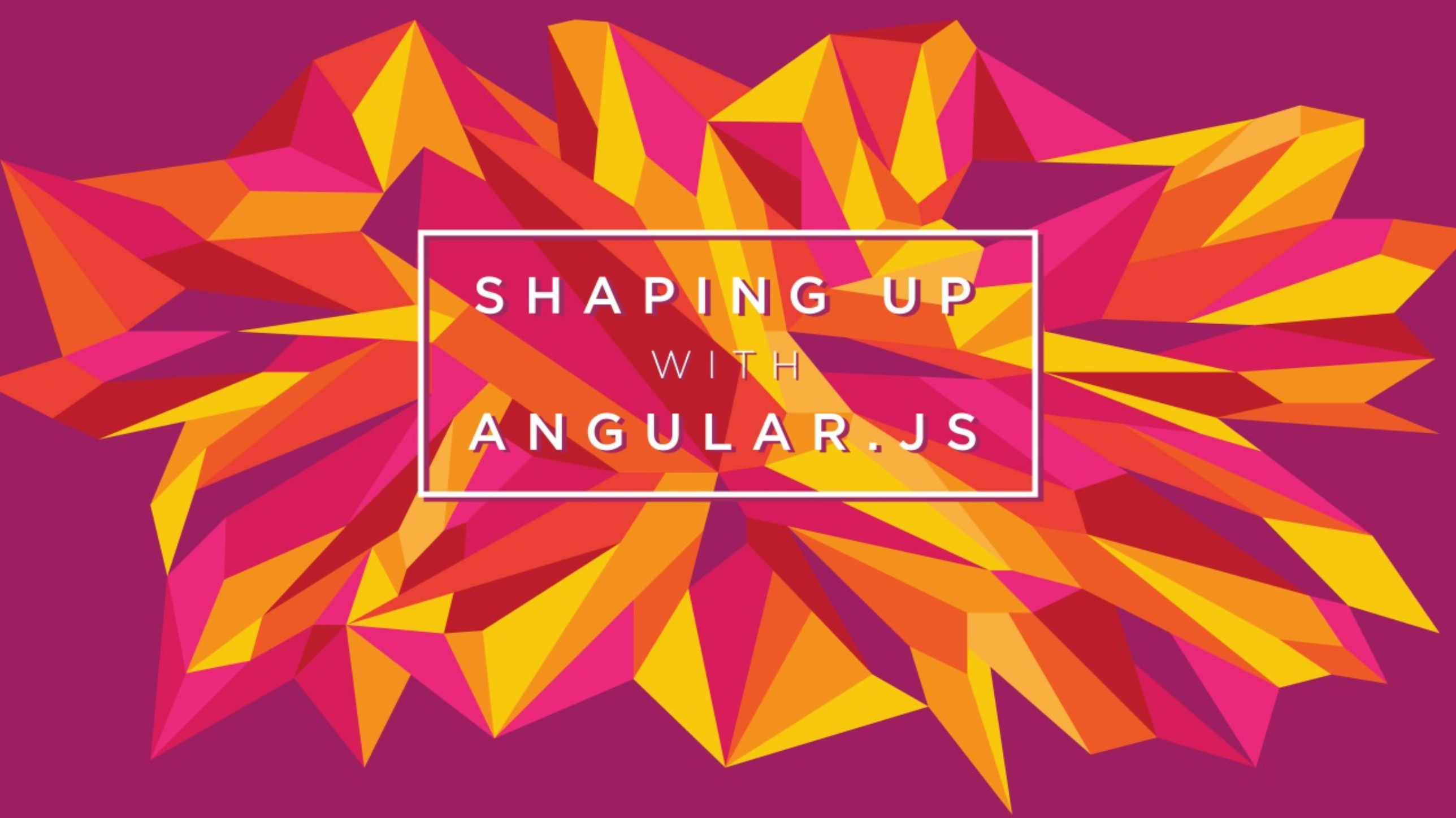
Controllers – Where we add application behavior

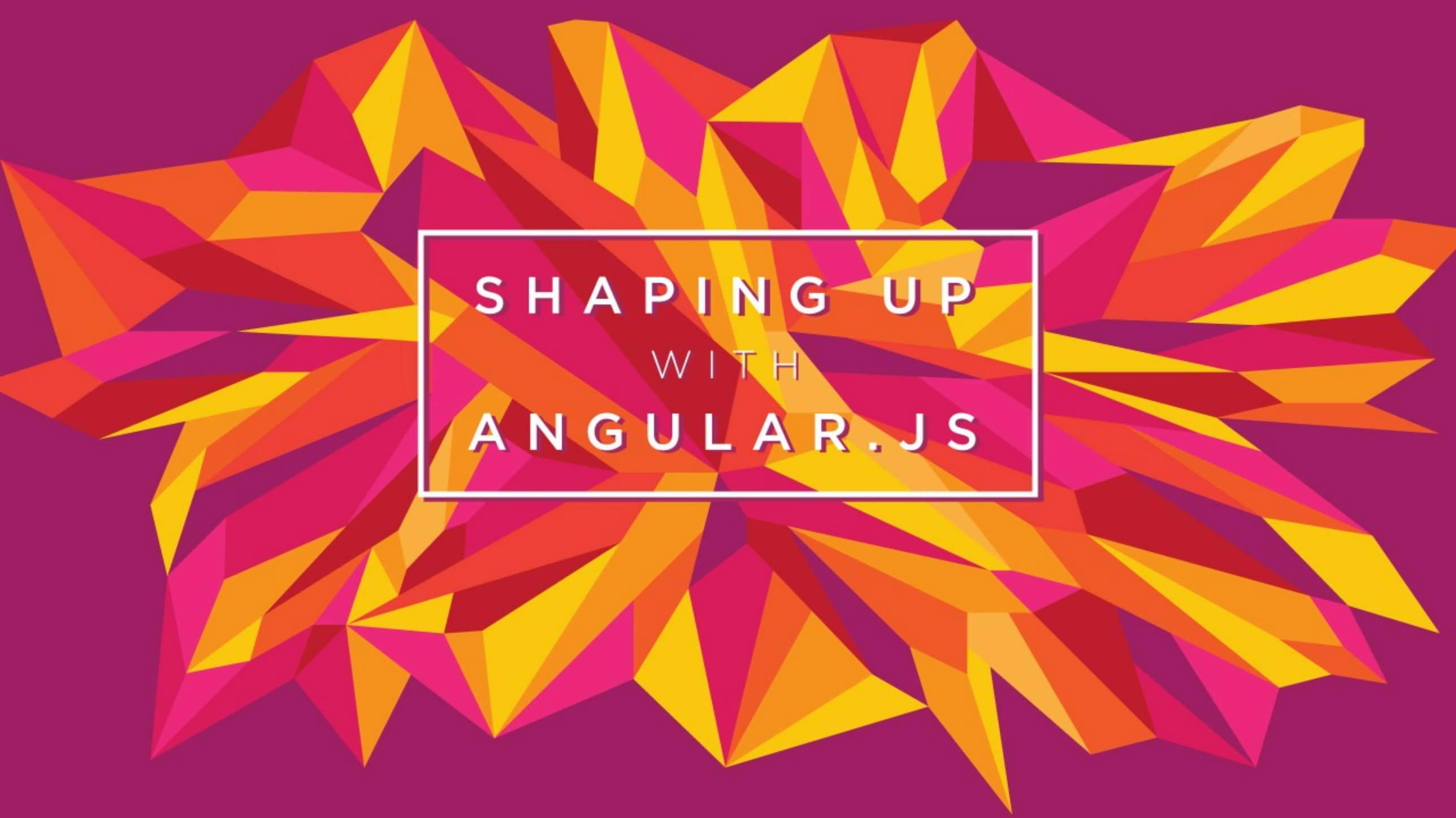
Expressions – How values get displayed within the page

SHAPING UP
WITH
ANGULAR.JS

Challenges

SHAPING UP
WITH
ANGULAR.JS


SHAPING UP
WITH
ANGULAR.JS


SHAPING UP
WITH
ANGULAR.JS

Shaping Up with Angular.JS

Level 2: Filters, Directives, and Cleaner Code

SHAPING UP
WITH
ANGULAR.JS

Directives We Know & Love

ng-app – attach the Application Module to the page

```
<html ng-app="store">
```

ng-controller – attach a Controller function to the page

```
<body ng-controller="StoreController as store">
```

ng-show / ng-hide – display a section based on an Expression

```
<h1 ng-show="name"> Hello, {{name}}! </h1>
```

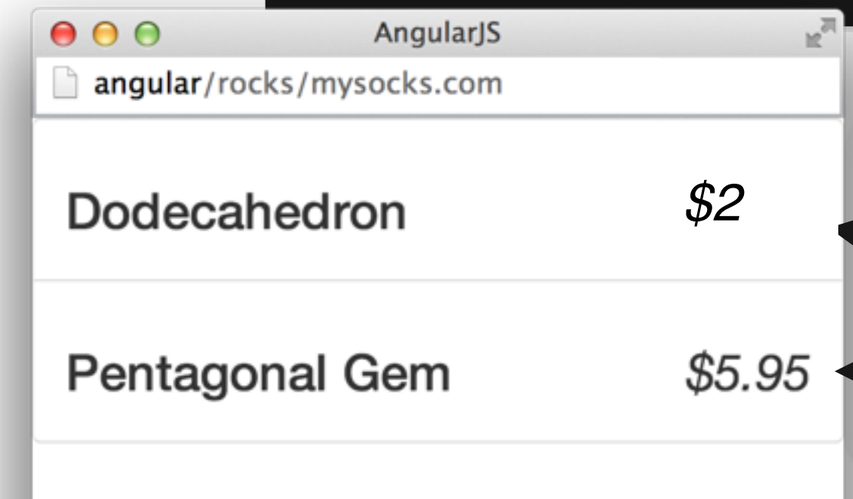
ng-repeat – repeat a section for each item in an Array

```
<li ng-repeat="product in store.products"> {{product.name}} </li>
```

Our Current Code

```
<body ng-controller="StoreController as store">
  <ul class="list-group">
    <li class="list-group-item" ng-repeat="product in store.products">
      <h3>
        {{product.name}}
        <em class="pull-right">${{product.price}}</em>
      </h3>
    </li>
  </ul>
</body>
```

index.html

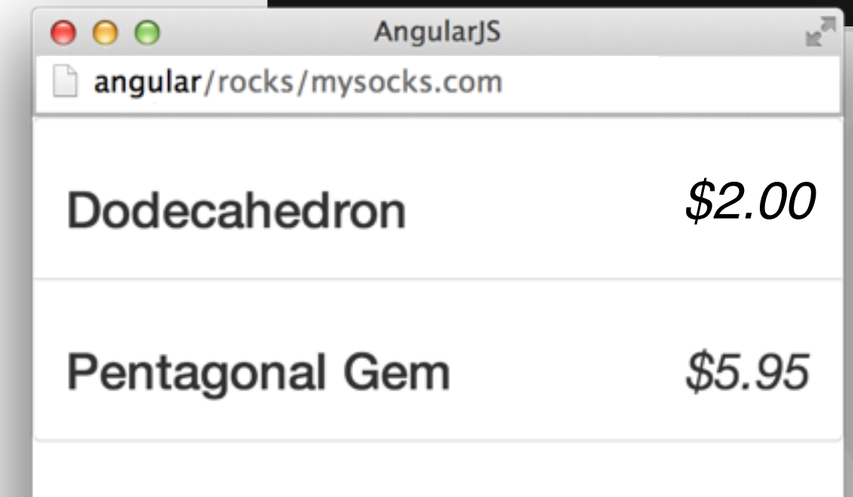


There's a better way to print out prices.

Our First Filter

```
<body ng-controller="StoreController as store">
  <ul class="list-group">
    <li class="list-group-item" ng-repeat="product in store.products">
      <h3>
        {{product.name}}
        <em class="pull-right">{{product.price | currency }}</em>
      </h3>
    </li>
  </ul>
</body>
```

index.html



Format this into currency

Pipe - "send the output into"

Notice it gives the dollar sign (localized)

Specifies number of decimals

Formatting with Filters

*Our Recipe `{{ data* | filter:options* }}`

date

```
{{'1388123412323' | date:'MM/dd/yyyy @ h:mma'}}
```

12/27/2013 @ 12:50AM

uppercase & lowercase

```
{{'octagon gem' | uppercase}}
```

OCTAGON GEM

limitTo

```
{{'My Description' | limitTo:8}}
```

My Descr

```
<li ng-repeat="product in store.products | limitTo:3">
```

orderBy

Will list products by descending price.

```
<li ng-repeat="product in store.products | orderBy:'-price'">
```

Without the - products would list in ascending order.

Adding an Image Array to our Product Array

```
var gems = [
   { name: 'Dodecahedron Gem',
     price: 2.95,
     description: '. . .',
     images: [
       {
         full: 'dodecahedron-01-full.jpg',
         thumb: 'dodecahedron-01-thumb.jpg'
       },
       {
         full: "dodecahedron-02-full.jpg",
         . . .
```

Our New Array

Image Object

app.js

To display the first image in a product: `{{product.images[0].full}}`

Using ng-src for Images

Using Angular Expressions inside a **src** attribute causes an error!

```
<img src="{{product.images[0].full}}"/>
```

...the browser tries to load the image *before* the Expression evaluates.

```
<body ng-controller="StoreController as store">
  <ul class="list-group">
    <li class="list-group-item" ng-repeat="product in store.products">
      <h3>
        {{product.name}}
        <em class="pull-right">{{product.price | currency}}</em>
        <img ng-src="{{product.images[0].full}}"/>
      </h3>
    </li>
  </ul>
</body>
```

NG-SOURCE to the rescue!

index.html

Our Products with Images

AngularJS

angular/rocks/mysocks.com

Flatlander Crafted Gems

– an Angular store –

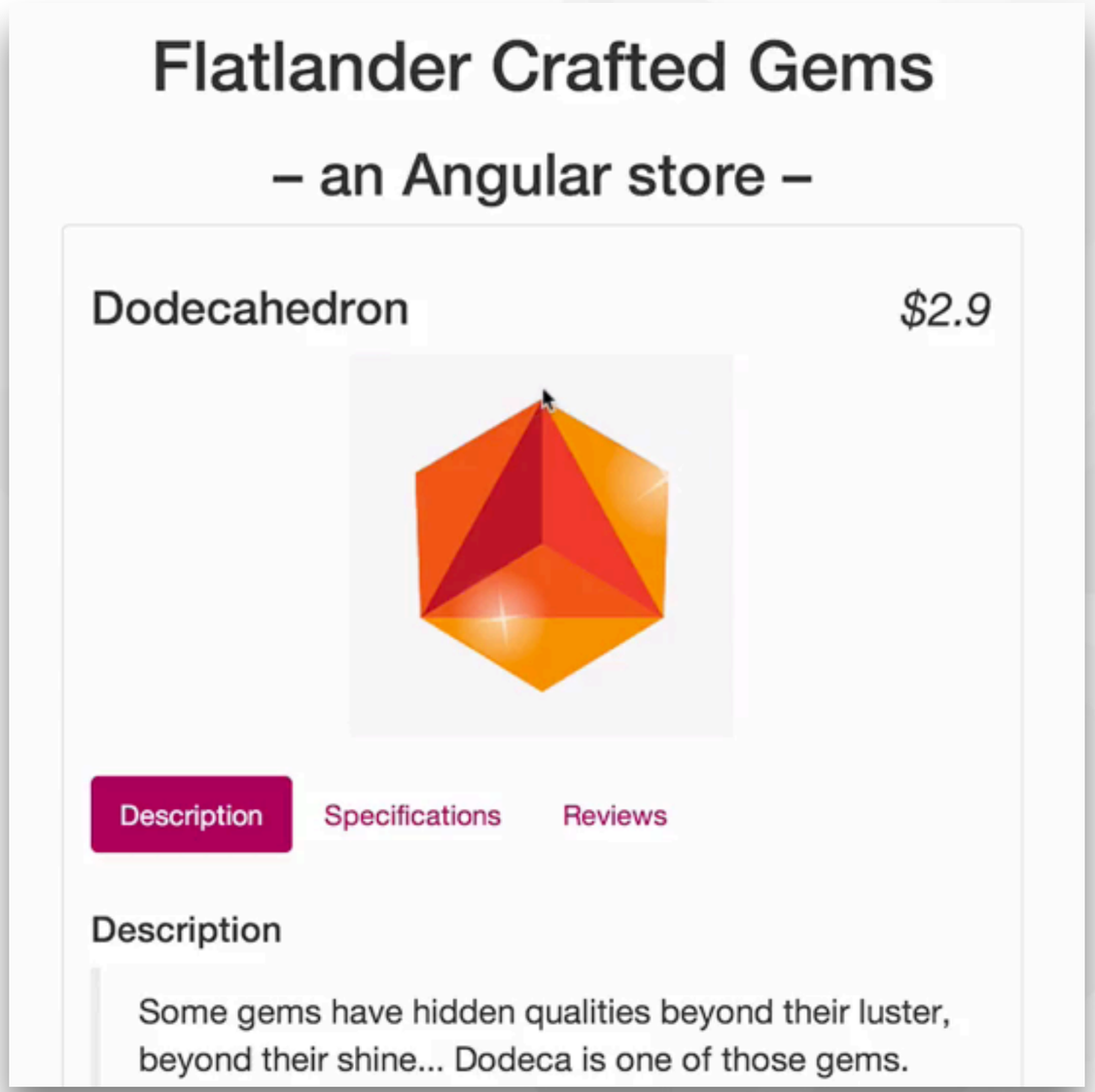
Pentagonal Gem $5.95

Dodecahedron $2.95

Challenges

SHAPING UP
WITH
ANGULAR.JS

More With Directives

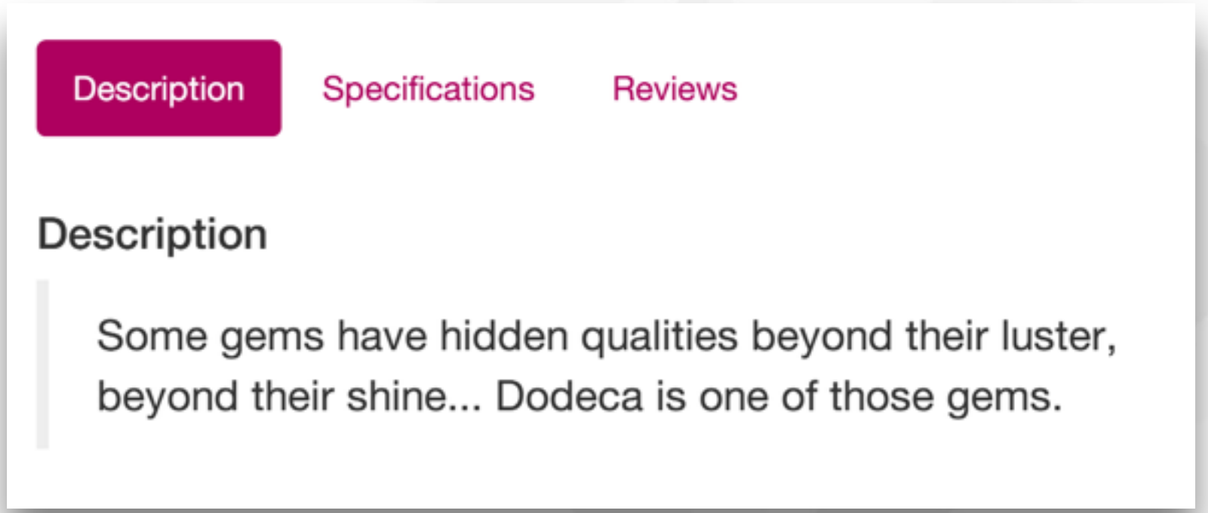


How can I make my application more interactive?

A Simple Set of Tabs

```
<section>
  <ul class="nav nav-pills”>
    <li> <a href>Description</a> </li>
    <li> <a href>Specifications</a> </li>
    <li> <a href>Reviews</a> </li>
  </ul>
</section>
```

index.html



SHAPING UP WITH ANGULAR.JS

Introducing a new Directive!

```
<section>
  <ul class="nav nav-pills”>
    <li> <a href ng-click="tab = 1">Description</a> </li>
    <li> <a href ng-click="tab = 2">Specifications</a> </li>
    <li> <a href ng-click="tab = 3">Reviews</a> </li>
  </ul>
  {{tab}}
</section>
```

index.html

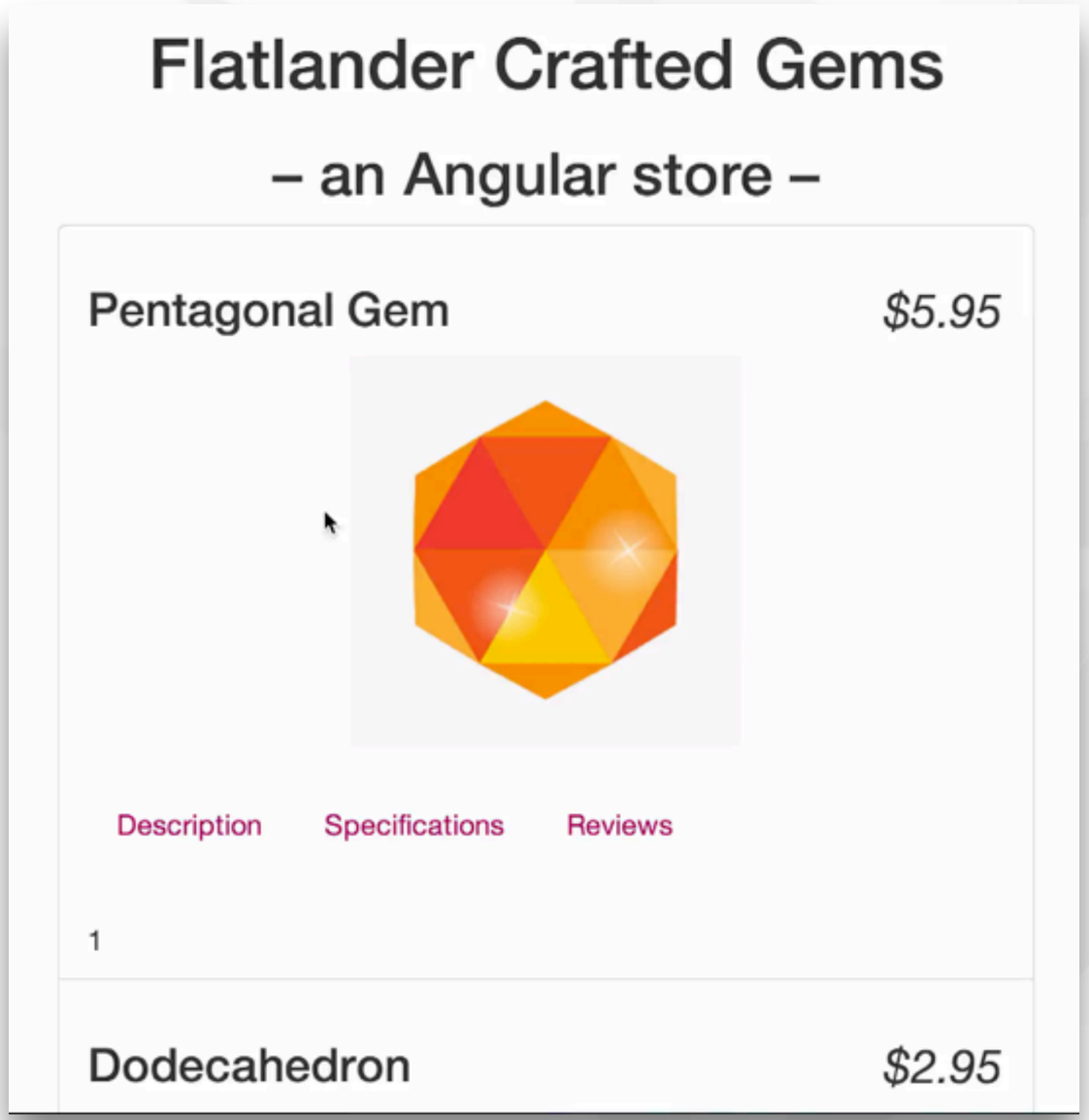
Assigning a value to **tab**.

For now just print this value to the screen.

SHAPING UP WITH ANGULAR.JS

angular/rocks/mysocks.com

Introducing a new Directive!



Whoa, it's dynamic and stuff...

When `ng-click` changes the value of `tab` ...

... the `{{tab}}` expression automatically gets updated!

Expressions define a 2-way Data Binding ...

this means Expressions are re-evaluated when a property changes.

Let's add the tab content panels

```
  ▲
  '---------- tabs are up here...
. . .
<div class="panel">
  <h4>Description</h4>
  <p>{{product.description}}</p>
</div>
<div class="panel">
  <h4>Specifications</h4>
  <blockquote>None yet</blockquote>
</div>
<div class="panel">
  <h4>Reviews</h4>
  <blockquote>None yet</blockquote>
</div>
```

How do we make the tabs trigger the panel to show?

Let's add the tab content panels

```
<div class="panel" ng-show="tab === 1">
  <h4>Description</h4>
  <p>{{product.description}}</p>
</div>
<div class="panel" ng-show="tab === 2">
  <h4>Specifications</h4>
  <blockquote>None yet</blockquote>
</div>
<div class="panel" ng-show="tab === 3">
  <h4>Reviews</h4>
  <blockquote>None yet</blockquote>
</div>
```

show the panel if tab is the right number

Now when a tab is selected it will show the appropriate panel!

Flatlander Crafted Gems

– an Angular store –

Pentagonal Gem $5.95

Description Specifications Reviews

Dodecahedron $2.95

But how do we set an initial value for a tab?

Setting the Initial Value

`ng-init` allows us to evaluate an expression in the current scope.

```
<section ng-init="tab = 1">
  <ul class="nav nav-pills">
    <li> <a href ng-click="tab = 1">Description</a> </li>
    <li> <a href ng-click="tab = 2">Specifications</a> </li>
    <li> <a href ng-click="tab = 3">Reviews</a> </li>
  </ul>
  . . .
```

index.html

Now with the Initial Tab

Flatlander Crafted Gems

– an Angular store –

Pentagonal Gem *$5.95*

Description Specifications Reviews

Description

> Origin of the Pentagonal Gem is unknown, hence its low value. It has a very high shine and 12 sides,

Setting the Active Class

We need to set the `class` to `active` if current `tab` is selected ...

```
<section ng-init="tab = 1">
  <ul class="nav nav-pills”>
    <li> <a href  ng-click="tab = 1">Description</a> </li>
    <li> <a href  ng-click="tab = 2">Specifications</a> </li>
    <li> <a href  ng-click="tab = 3">Reviews</a></li>
  </ul>
```

index.html

SHAPING UP WITH ANGULAR.JS

The ng-class directive

```
<section ng-init="tab = 1">
  <ul class="nav nav-pills”>
    <li ng-class="{ active:tab === 1 }">
      <a href ng-click="tab = 1">Description</a>
    </li>
    <li ng-class="{ active:tab === 2 }">
      <a href ng-click="tab = 2">Specifications</a>
    </li>
    <li ng-class="{ active:tab === 3 }">
      <a href ng-click="tab = 3">Reviews</a>
    </li>
  </ul>
  . . .
```

Expression to evaluate
If true, set class to "active",
otherwise nothing.

index.html

Name of the class to set.

SHAPING UP WITH ANGULAR.JS

Flatlander Crafted Gems

– an Angular store –

Pentagonal Gem $5.95

Description | Specifications | Reviews

Description

> Origin of the Pentagonal Gem is unknown, hence its low value. It has a very high shine and 12 sides,

Feels dirty, doesn't it?

All our application's logic is inside our HTML.

```
<section ng-init="tab = 1">
  <ul class="nav nav-pills">
    <li ng-class="{ active:tab === 1}">
      <a href ng-click="tab = 1">Description</a>
    </li>
    <li ng-class="{ active:tab === 2}">
      <a href ng-click="tab = 2">Specifications</a>
    </li>
    <li ng-class="{ active:tab === 3 }">
      <a href ng-click="tab = 3">Reviews</a>
    </li>
  </ul>
  <div class="panel" ng-show="tab === 1">
    <h4>Description </h4>
    <p>{{product.description}}</p>
  </div>
…
```

How might we pull this logic into a Controller?

index.html

Creating our Panel Controller

```
<section ng-init="tab = 1" ng-controller="PanelController as panel">
  <ul class="nav nav-pills">
    <li ng-class="{ active:tab === 1 }">
      <a href ng-click="tab = 1">Description</a>
    </li>
    <li ng-class="{ active:tab === 2 }">
      <a href ng-click="tab = 2">Specifications</a>
    </li>
    <li ng-class="{ active:tab === 3 }">
      <a href ng-click="tab = 3">Reviews</a>
    </li>
  </ul>
  <div class="panel" ng-show="tab === 1">
    <h4>Description </h4>
    <p>{{product.description}}</p>
  </div>
…
```

```
app.controller("PanelController", function(){
});
```

app.js

Moving our tab initializer

```
<section ng-controller="PanelController as panel">
  <ul class="nav nav-pills">
    <li ng-class="{ active:tab === 1 }">
      <a href ng-click="tab = 1">Description</a>
    </li>
    <li ng-class="{ active:tab === 2 }">
      <a href ng-click="tab = 2">Specifications</a>
    </li>
    <li ng-class="{ active:tab === 3 }">
      <a href ng-click="tab = 3">Reviews</a>
    </li>
  </ul>
  <div class="panel" ng-show="tab === 1">
    <h4>Description </h4>
    <p>{{product.description}}</p>
  </div>
…
```

```
app.controller("PanelController", function(){
  this.tab = 1;
});
```

app.js

Creating our selectTab function

```
<section ng-controller="PanelController as panel">
  <ul class="nav nav-pills">
    <li ng-class="{ active:tab === 1 }">
      <a href ng-click="panel.selectTab(1)">Description</a>
    </li>
    <li ng-class="{ active:tab === 2 }">
      <a href ng-click="panel.selectTab(2)">Specifications</a>
    </li>
    <li ng-class="{ active:tab === 3 }">
      <a href ng-click="panel.selectTab(3)" >Reviews</a>
    </li>
  </ul>
  <div class="panel" ng-show="tab === 1">
    <h4>Description </h4>
    <p>{{product.description}}</p>
  </div>
…
```

```
app.controller("PanelController", function(){
  this.tab = 1;

  this.selectTab = function(setTab) {
    this.tab = setTab;
  };
});
```

app.js

Creating our isSelected function

```
<section ng-controller="PanelController as panel">
  <ul class="nav nav-pills">
    <li ng-class="{ active: panel.isSelected(1) }">
      <a href ng-click="panel.selectTab(1)" >Description</a>
    </li>
    <li ng-class="{ active: panel.isSelected(2)}">
      <a href ng-click="panel.selectTab(2)">Specifications</a>
    </li>
    <li ng-class="{ active: panel.isSelected(3)}">
      <a href ng-click="panel.selectTab(3)">Reviews</a>
    </li>
  </ul>
  <div class="panel" ng-show="panel.isSelected(1)">
    <h4>Description </h4>
    <p>{{product.description}}</p>
  </div>
…
```

```
app.controller("PanelController", function(){
  this.tab = 1;

  this.selectTab = function(setTab) {
    this.tab = setTab;
  };
  this.isSelected = function(checkTab){
    return this.tab === checkTab;
  };
});
```

app.js

Flatlander Crafted Gems

– an Angular store –

Pentagonal Gem *$5.95*

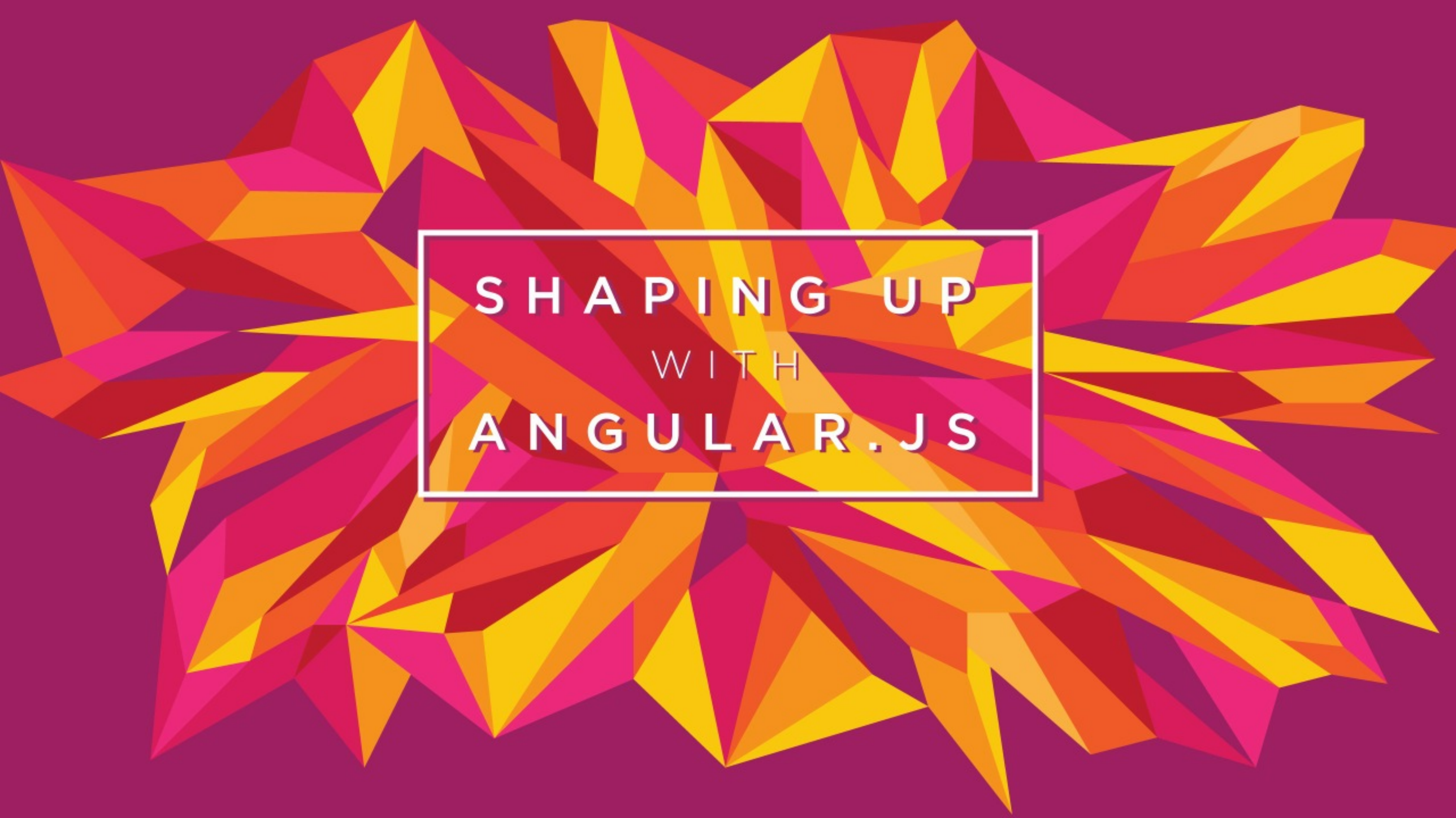
Description | Specifications | Reviews

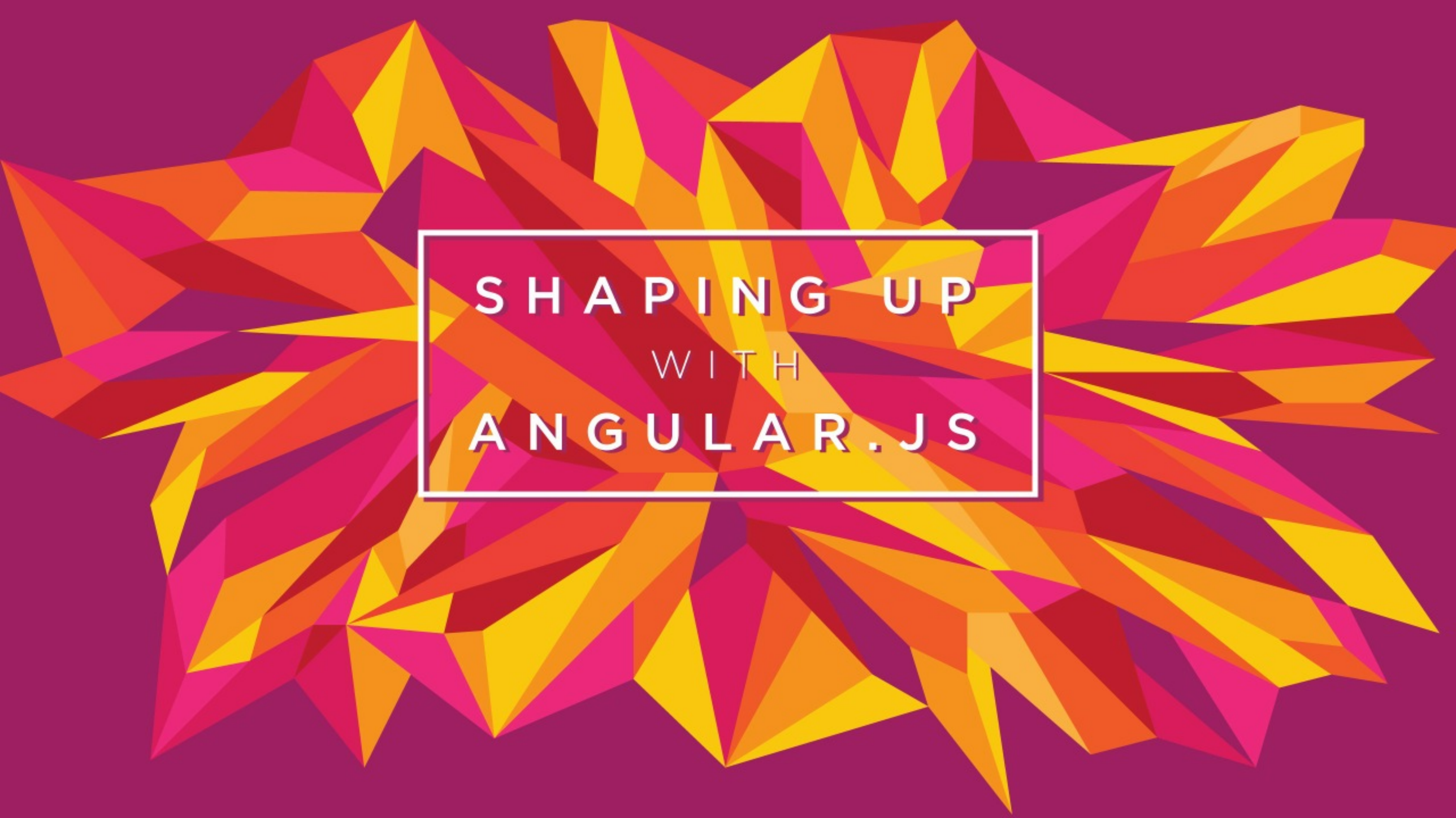
Description

> Origin of the Pentagonal Gem is unknown, hence its low value. It has a very high shine and 12 sides,

Challenges

SHAPING UP
WITH
ANGULAR.JS


SHAPING UP
WITH
ANGULAR.JS

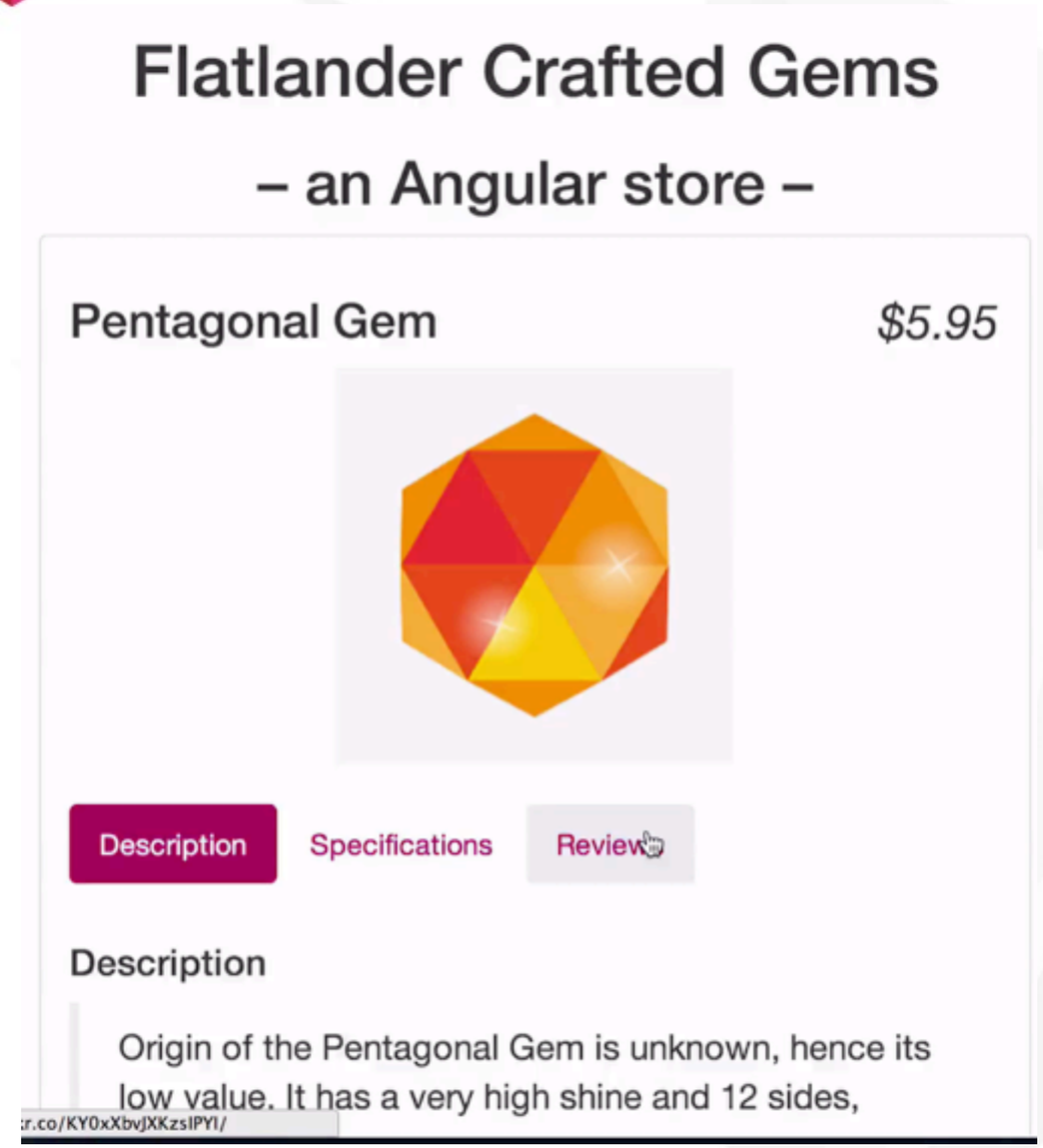

SHAPING UP
WITH
ANGULAR.JS

Shaping Up with Angular

Level 3: Forms, Models, and Validations

SHAPING UP
WITH
ANGULAR.JS

More Directives: Forms & Models



How can I let my users add content?

Adding reviews to our products

```
app.controller("StoreController", function(){
  this.products = [
    {
      name: 'Awesome Multi-touch Keyboard',
      price: 250.00,
      description: "...",
      images: [...],
      reviews: [
        {
          stars: 5,
          body: "I love this product!",
          author: "joe@thomas.com"
        },
        {
          stars: 1,
          body: "This product sucks",
          author: "tim@hater.com"
        }
        . . .
```

app.js

Looping Over Reviews in our Tab

```
<li class="list-group-item" ng-repeat="product in store.products">
  . . .
  <div class="panel" ng-show="panel.isSelected(3)">
    <h4> Reviews </h4>

    <blockquote ng-repeat="review in product.reviews">
      <b>Stars: {{review.stars}}</b>
      {{review.body}}
      <cite>by: {{review.author}}</cite>
    </blockquote>
  </div>
```

index.html

SHAPING UP WITH ANGULAR.JS

We're displaying Reviews!

Reviews

> **3 Stars** I think this gem was just OK, could honestly use more shine, IMO.
> —JimmyDean@sausage.com

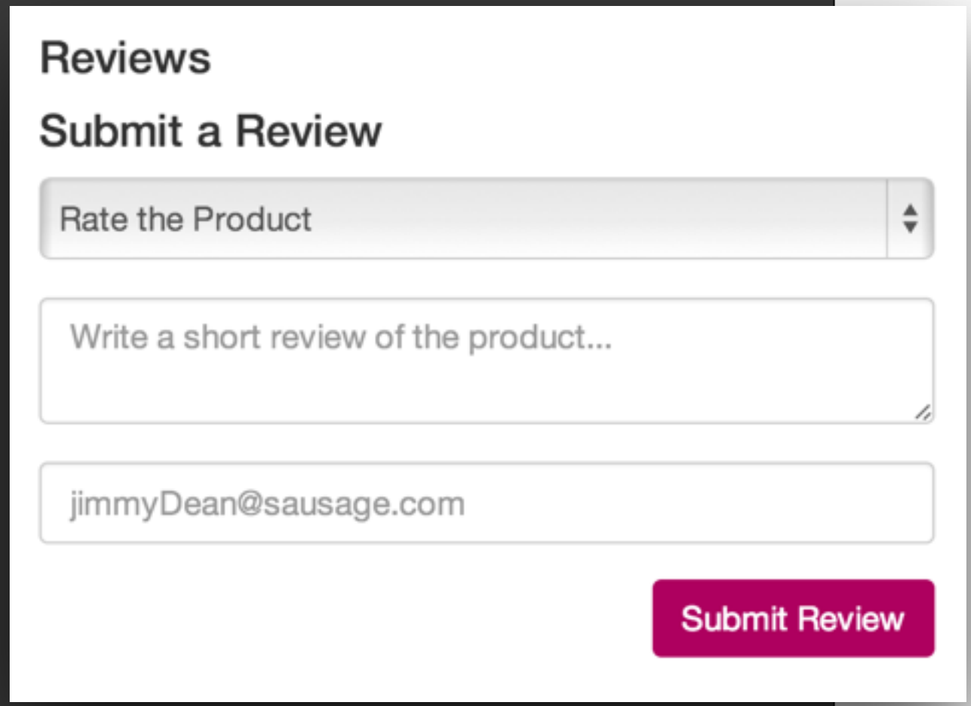
> **4 Stars** Any gem with 12 faces is for me!
> —gemsRock@alyssaNicoll.com

Nothing new here, but how do we start to implement forms?

Writing out our Review Form

```
<h4> Reviews </h4>

<blockquote ng-repeat="review in product.reviews">...</blockquote>
```

```
<form name="reviewForm">
  <select>
    <option value="1">1 star</option>
    <option value="2">2 stars</option>
    . . .
  </select>
  <textarea></textarea>
  <label>by:</label>
  <input type="email" />
  <input type="submit" value="Submit" />
</form>
```



index.html

With Live Preview

```
<form name="reviewForm">
  <blockquote>
    <b>Stars: {{review.stars}}</b>
    {{review.body}}
    <cite>by: {{review.author}}</cite>
  </blockquote>
  <select>
    <option value="1">1 star</option>
    <option value="2">2 stars</option>
    . . .
  </select>
  <textarea></textarea>
  <label>by:</label>
  <input type="email" />
  <input type="submit" value="Submit" />
</form>
```

How do we bind this review object to the form?

index.html

Introducing ng-model

```
<form name="reviewForm">
  <blockquote>
    <b>Stars: {{review.stars}}</b>
    {{review.body}}
    <cite>by: {{review.author}}</cite>
  </blockquote>
  <select ng-model="review.stars">
    <option value="1">1 star</option>
    <option value="2">2 stars</option>
    . . .
  </select>
  <textarea ng-model="review.body"></textarea>
  <label>by:</label>
  <input ng-model="review.author" type="email" />
  <input type="submit" value="Submit" />
</form>
```

ng-model binds the form element value to the property

index.html

Live Preview In Action

5 Stars This gem is SO

—

Submit a Review

5

This gem is SO

jimmyDean@sausage.com

Submit Review

But how do we actually add the new review?

Two More Binding Examples

With a Checkbox

```
<input ng-model="review.terms" type="checkbox" /> I agree to the terms
```

Sets value to true or false

With Radio Buttons

```
What color would you like?

<input ng-model="review.color" type="radio" value="red" /> Red
<input ng-model="review.color" type="radio" value="blue" /> Blue
<input ng-model="review.color" type="radio" value="green" /> Green
```

Sets the proper value based on which is selected

Challenges

SHAPING UP
WITH
ANGULAR.JS

We need to Initialize the Review

```
<form name="reviewForm">
  <blockquote>
  <b>Stars: {{review.stars}}</b>
  {{review.body}}
  <cite>by: {{review.author}}</cite>
  </blockquote>

  <select ng-model="review.stars">
    <option value="1">1 star</option>
    <option value="2">2 stars</option>
    . . .
  </select>
  <textarea ng-model="review.body"></textarea>
  <label>by:</label>
  <input ng-model="review.author" type="email" />
  <input type="submit" value="Submit" />
</form>
```

index.html

We could do **ng-init**, but we're better off creating a controller.

Creating the Review Controller

```
app.controller("ReviewController", function(){
  this.review = {};
});
```

app.js

```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl">
  <blockquote>
  <b>Stars: {{review.stars}}</b>
  {{review.body}}
  <cite>by: {{review.author}}</cite>
  </blockquote>

<select ng-model="review.stars">
  <option value="1">1 star</option>
  <option value="2">2 stars</option>
  . . .
</select>
<textarea ng-model="review.body"></textarea>
```

Now we need to update all the Expressions to use this controller alias.

index.html

Using the reviewCtrl.review

```
app.controller("ReviewController", function(){
  this.review = {};
});
```

app.js

```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl">
  <blockquote>
  <b>Stars: {{reviewCtrl.review.stars}}</b>
  {{reviewCtrl.review.body}}
  <cite>by: {{reviewCtrl.review.author}}</cite>
  </blockquote>

<select ng-model="reviewCtrl.review.stars">
  <option value="1">1 star</option>
  <option value="2">2 stars</option>
  . . .
</select>
<textarea ng-model="reviewCtrl.review.body"></textarea>
```

index.html

Using ng-submit to make the Form Work

```
app.controller("ReviewController", function(){
  this.review = {};
});
```

app.js

`ng-submit` allows us to call a function when the form is submitted.

```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl"
                        ng-submit="reviewCtrl.addReview(product)">
  <blockquote>
  <b>Stars: {{reviewCtrl.review.stars}}</b>
  {{reviewCtrl.review.body}}
  <cite>by: {{reviewCtrl.review.author}}</cite>
  </blockquote>
```

We need to define this function.

index.html

Using ng-submit to make the Form Work

```
app.controller("ReviewController", function(){
  this.review = {};

  this.addReview = function(product) {
    product.reviews.push(this.review);
  };
});
```

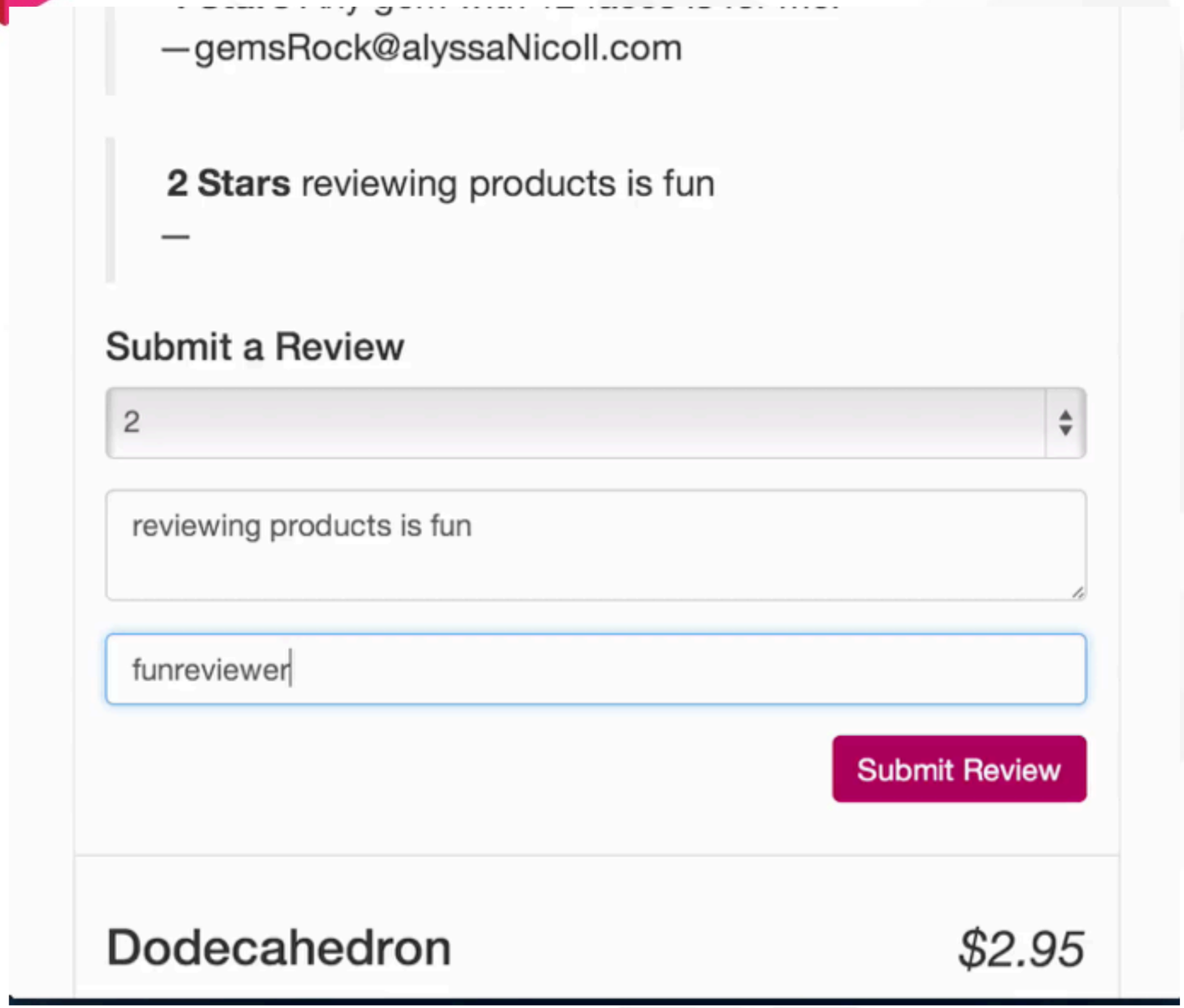
Push **review** onto this product's **reviews** array.

app.js

```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl"
                        ng-submit="reviewCtrl.addReview(product)"
  <blockquote>
  <b>Stars: {{reviewCtrl.review.stars}}</b>
  {{reviewCtrl.review.body}}
  <cite>by: {{reviewCtrl.review.author}}</cite>
  </blockquote>
```

index.html

Now with Reviews!



Review gets added, but the form still has all previous values!

Resetting the Form on Submit

```
app.controller("ReviewController", function(){
  this.review = {};

  this.addReview = function(product) {
    product.reviews.push(this.review);
    this.review = {};
  };
});
```

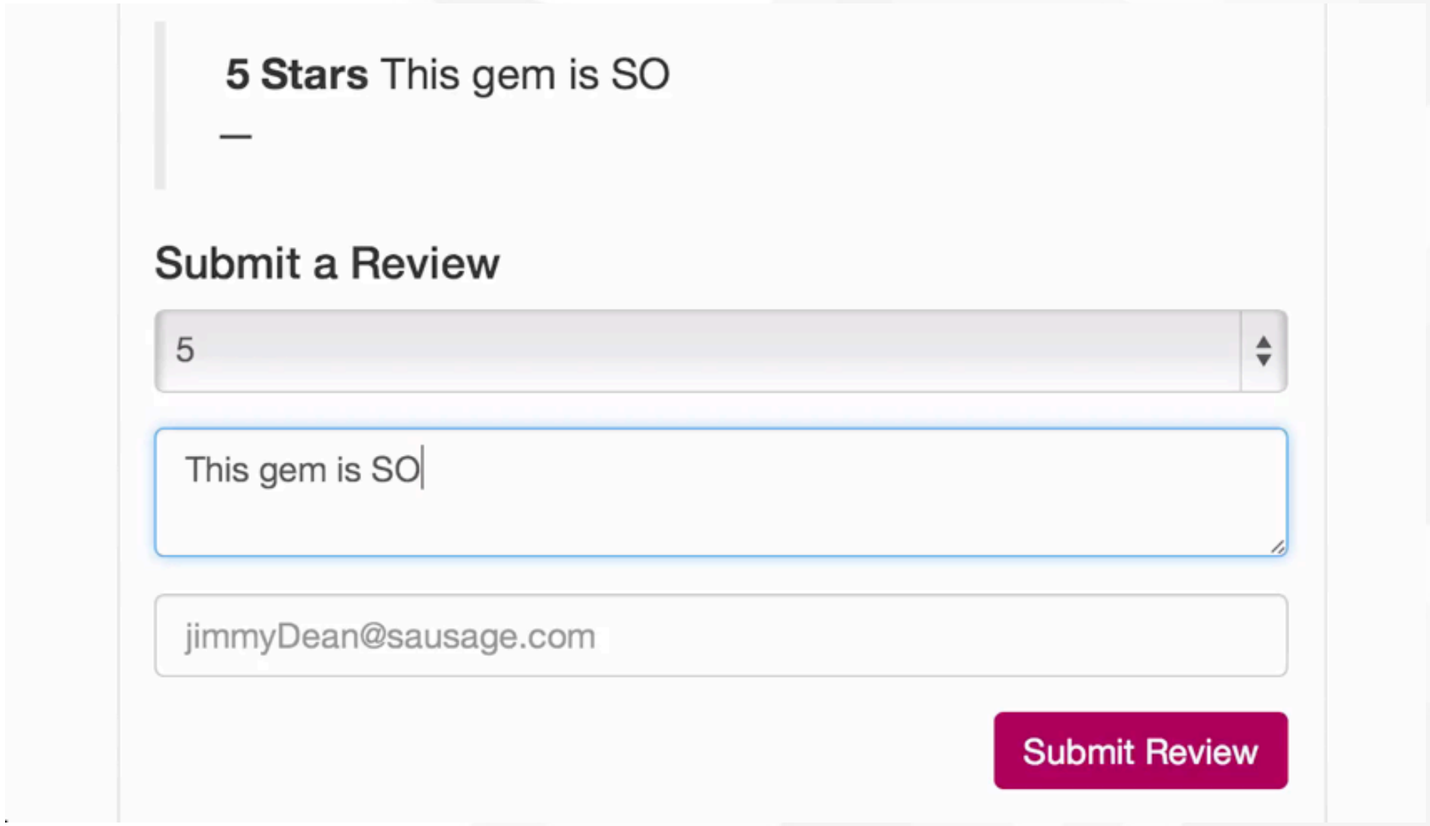
Clear out the review, so the form will reset.

app.js

```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl"
                        ng-submit="reviewCtrl.addReview(product)">
  <blockquote>
  <b>Stars: {{reviewCtrl.review.stars}}</b>
  {{reviewCtrl.review.body}}
  <cite>by: {{reviewCtrl.review.author}}</cite>
</blockquote>
```

index.html

This Time the Form Resets



However, if we refresh, the reviews get reset!

We're not saving the reviews anywhere yet...

Challenges

SHAPING UP
WITH
ANGULAR.JS

What about validations?

Turns out Angular has some great client side validations we can use with our directives.

Our Old Form Code

```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl"
                        ng-submit="reviewCtrl.addReview(product)">
  <select ng-model="reviewCtrl.review.stars">
    <option value="1">1 star</option>
    ...
  </select>

  <textarea name="body" ng-model="reviewCtrl.review.body"></textarea>
  <label>by:</label>
  <input name="author" ng-model="reviewCtrl.review.author" type="email" />

  <input type="submit" value="Submit" />
</form>
```

index.html

Now with validation

Turn Off Default HTML Validation

```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl"
                        ng-submit="reviewCtrl.addReview(product)" novalidate>
  <select ng-model="reviewCtrl.review.stars" required>
    <option value="1">1 star</option>
    ...
  </select>

  <textarea name="body" ng-model="reviewCtrl.review.body" required></textarea>
  <label>by:</label>
  <input name="author" ng-model="reviewCtrl.review.author" type="email" required/>

  <div> reviewForm is {{reviewForm.$valid}} </div>
  <input type="submit" value="Submit" />
</form>
```

Mark Required Fields

Print Forms Validity

index.html

With validations

5 Stars

—alyssa@codeschool.com

Submit a Review

5

Write a short review of the product...

alyssa@codeschool.com

Submit Review

reviewForm is false

We don't want the form to submit when it's invalid.

Preventing the Submit

```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl"
                        ng-submit="reviewCtrl.addReview(product)" novalidate>
```

index.html

We only want this method to be called if **reviewForm.$valid** is true.

SHAPING UP WITH ANGULAR.JS

Preventing the Submit

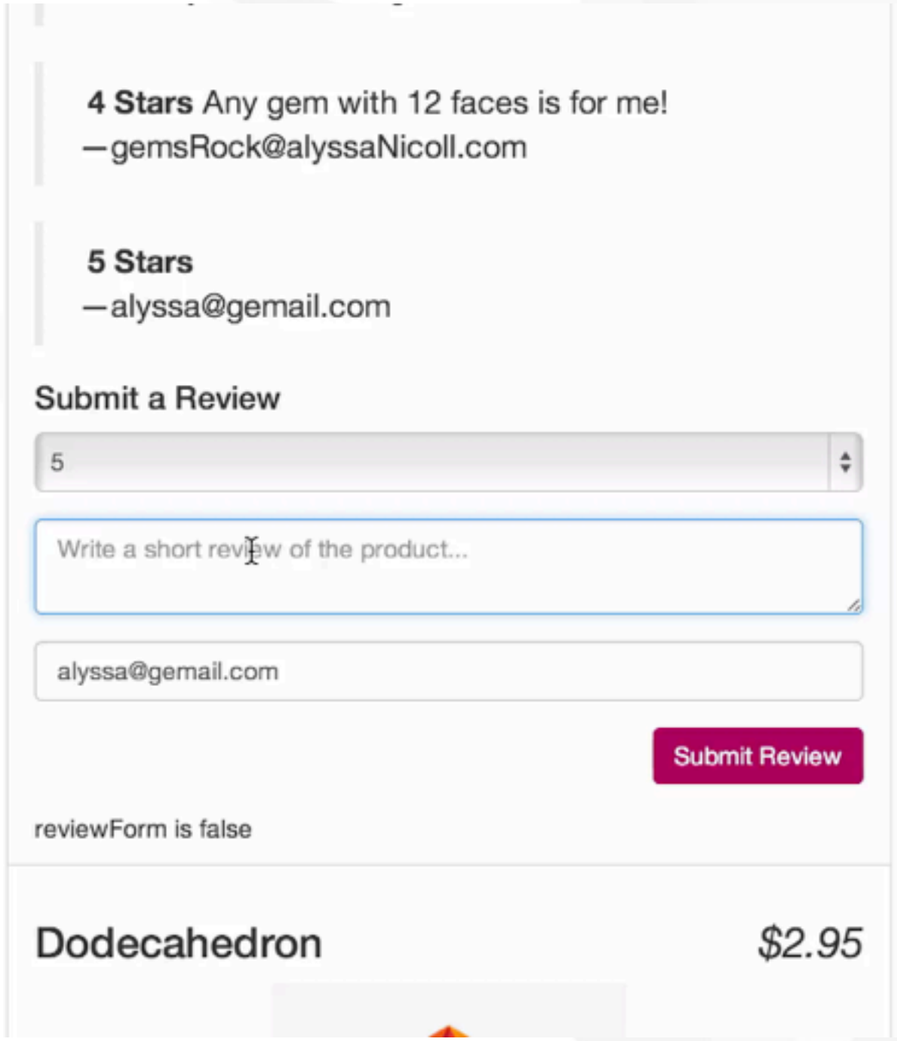
```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl"
    ng-submit="reviewForm.$valid && reviewCtrl.addReview(product)" novalidate>
```

index.html

If valid is **false**, then **addReview** is never called.

SHAPING UP WITH ANGULAR.JS

Doesn't Submit an Invalid Form



How might we give a hint to the user why their form is invalid?

The Input Classes

index.html

```
<input name="author" ng-model="reviewCtrl.review.author" type="email" required />
```

Source before typing email

```
<input name="author" . . . class="ng-pristine ng-invalid">
```

Source after typing, with invalid email

```
<input name="author". . . class="ng-dirty ng-invalid">
```

Source after typing, with valid email

```
<input name="author" . . . class="ng-dirty ng-valid">
```

So, let's highlight the form field using classes after we start typing, `ng-dirty` showing if a field is valid or invalid. `ng-valid` `ng-invalid`

The classes

```
<input name="author" ng-model="reviewCtrl.review.author" type="email" required />
```

index.html

```
.ng-invalid.ng-dirty {
  border-color: #FA787E;
}

.ng-valid.ng-dirty {
  border-color: #78FA89;
}
```

Red border for invalid

Green border for valid

style.css

SHAPING UP WITH ANGULAR.JS

Now with red and green borders!

4 Stars Any gem with 12 faces is for me!
—gemsRock@alyssaNicoll.com

Submit a Review

Rate the Product

Write a short review of the product...

jimmyDean@sausage.com

Submit Review

reviewForm is false

Dodecahedron $2.95

HTML5-based type validations

Web forms usually have rules around valid input:

- Angular JS has built-in validations for common input types:

```
<input type="email" name="email">
```

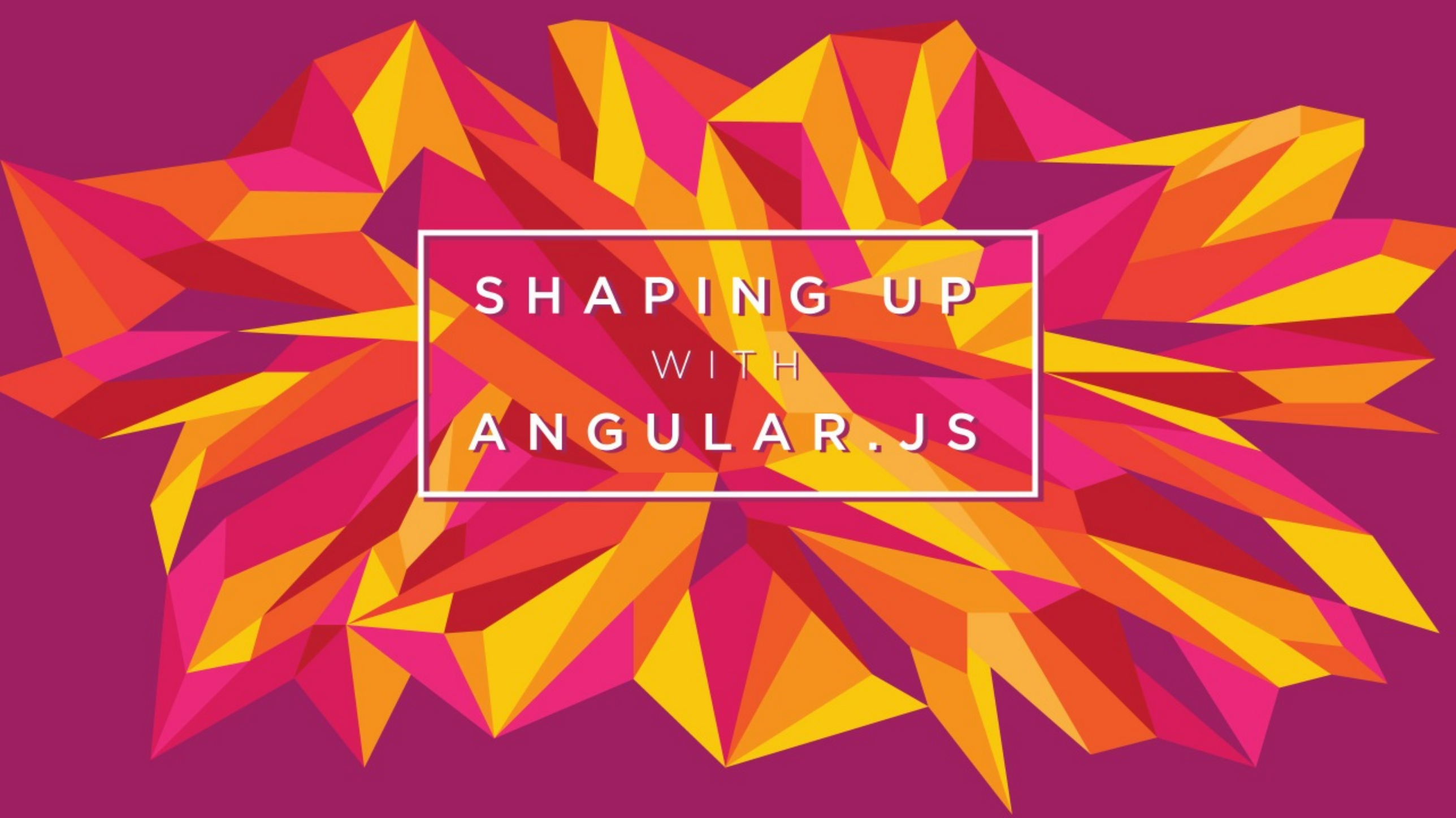
```
<input type="url" name="homepage">
```

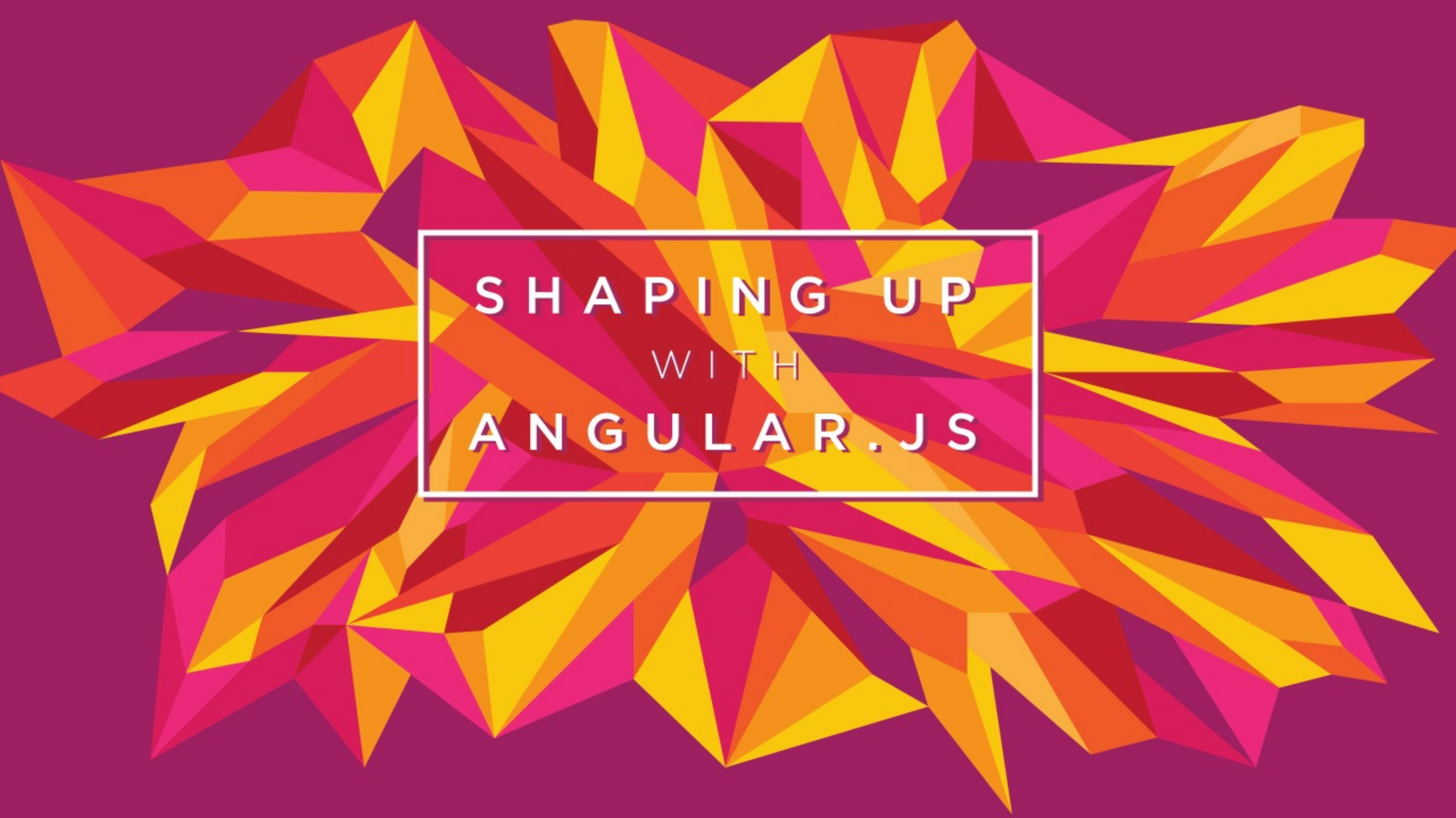
```
<input type="number" name="quantity">
```

Can also define min & max with numbers

```
min=1 max=10
```

SHAPING UP WITH ANGULAR.JS


SHAPING UP
WITH
ANGULAR.JS


SHAPING UP
WITH
ANGULAR.JS

Shaping Up with Angular JS

Level 4: Creating a Directive with an Associated Controller

SHAPING UP
WITH
ANGULAR.JS

Decluttering our Code

```
<ul class="list-group">
  <li class="list-group-item" ng-repeat="product in store.products">
    <h3>
      {{product.name}}
      <em class="pull-right">${{product.price}}</em>
    </h3>
    <section ng-controller="PanelController as panel">
    . . .
```

index.html

We're going to have multiple pages that want to reuse this HTML snippet.

How do we eliminate this duplication?

SHAPING UP WITH ANGULAR.JS

Using ng-include for Templates

```
<ul class="list-group">
  <li class="list-group-item" ng-repeat="product in store.products">
    <h3 ng-include="'product-title.html'">
    </h3>
    <section ng-controller="PanelController as panel">
```

name of file to include

index.html

`ng-include` is expecting a variable with the name of the file to include.

To pass the name directly as a string, use single quotes (`'...'`)

```
 {{product.name}}
 <em class="pull-right">${{product.price}}</em>
```

product-title.html

```
<h3 ng-include="'product-title.html'" class="ng-scope">
  <span class="ng-scope ng-binding">Awesome Multi-touch Keyboard</span>
  <em class="pull-right ng-scope ng-binding">$250.00</em>
</h3>
```

generated html

Flatlander Crafted Gems

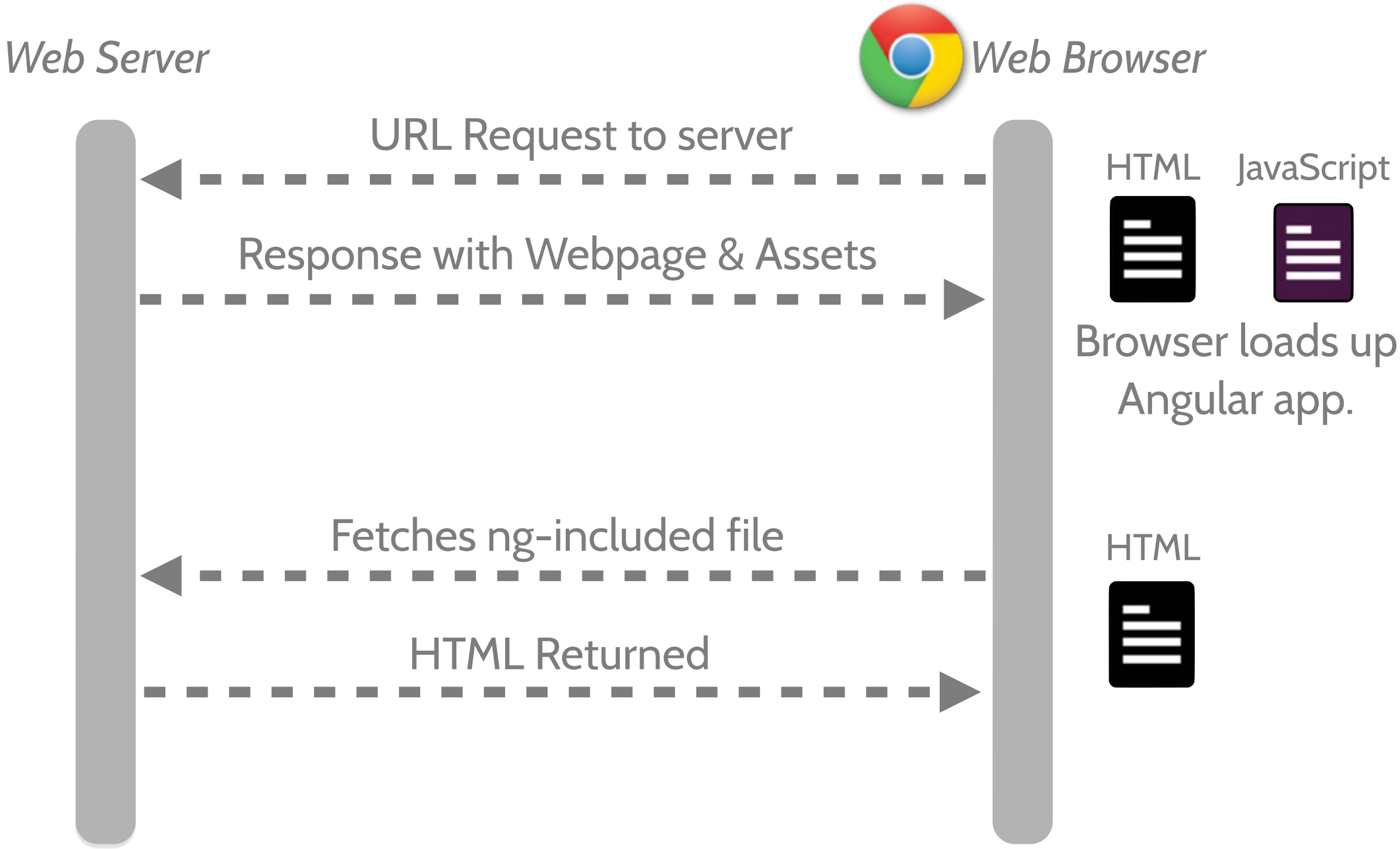
– an Angular store –

Pentagonal Gem $5.95

Description Specifications Reviews

Description

> Origin of the Pentagonal Gem is unknown, hence its low value. It has a very high shine and 12 sides,


Web Server
Web Browser
URL Request to server
Response with Webpage & Assets
HTML
JavaScript
Browser loads up Angular app.
Fetches ng-included file
HTML
HTML Returned

Creating our First Custom Directive

Using ng-include...

```
<h3 ng-include="'product-title.html'"></h3>
```

index.html

Custom Directive

```
<product-title></product-title>
```

index.html

Our old code and our custom Directive will do the same thing...
with some additional code.

Why write a Directive?

SHAPING UP WITH ANGULAR.JS

Why Directives?

Directives allow you to write HTML that expresses the behavior of your application.

```
<aside class="col-sm-3">
  <book-cover></book-cover>
  <h4><book-rating></book-rating></h4>
</aside>

<div class="col-sm-9">
  <h3><book-title></book-title></h3>

  <book-authors></book-authors>

  <book-review-text></book-review-text>

  <book-genres></book-genres>
</div>
```

Can you tell what this does?

SHAPING UP WITH ANGULAR.JS

Writing Custom Directives

Template-expanding Directives are the simplest:

- define a custom tag or attribute that is expanded or replaced
- can include Controller logic, if needed

Directives can also be used for:

- Expressing complex UI
- Calling events and registering event handlers
- Reusing common components

How to Build Custom Directives

```
<product-title></product-title>
```

index.html

```
app.directive('productTitle', function(){
  return {
    A configuration object defining how your directive will work
  };
});
```

app.js

How to Build Custom Directives

```
<product-title></product-title>
```

index.html

dash in HTML translates to ... **camelCase** in JavaScript

```
app.directive('productTitle', function(){
  return {
    restrict: 'E',
    templateUrl: 'product-title.html'
  };
});
```

Type of Directive (E for Element)

Url of a template

generates into

app.js

```
<h3>
  {{product.name}}
  <em class="pull-right">$250.00</em>
</h3>
```

index.html

Flatlander Crafted Gems

– an Angular store –

Pentagonal Gem *$5.95*

Description Specifications Reviews

Description

> Origin of the Pentagonal Gem is unknown, hence its low value. It has a very high shine and 12 sides,

Attribute vs Element Directives

Element Directive

```
<product-title></product-title>
```

index.html

Notice we're not using a self-closing tag... `<product-title/>`

...some browsers don't like self-closing tags.

Attribute Directive

```
<h3 product-title></h3>
```

index.html

Use Element Directives for UI widgets and Attribute Directives for mixin behaviors... like a tooltip.

SHAPING UP WITH ANGULAR.JS

Defining an Attribute Directive

```
<h3 product-title></h3>
```

index.html

```
app.directive('productTitle', function(){
  return {
    restrict: 'A',
    templateUrl: 'product-title.html'
  };
});
```

Type of Directive (A for Attribute)

generates into

app.js

```
<h3>
  {{product.name}}
  <em class="pull-right">$250.00</em>
</h3>
```

index.html

Though normally attributes would be for mixin behaviors ...

Directives allow you to write better HTML

When you think of a dynamic web application, do you think you'll be able to understand the functionality just by looking at the HTML?

No, right?

When you're writing an Angular JS application, you should be able to understand the behavior and intent from just the HTML.

And you're likely using custom directives to write expressive HTML.

Shaping Up with Angular JS

Creating Our Own Directives

SHAPING UP
WITH
ANGULAR.JS

Reviewing our Directive

Template-Expanding Directives

```
<h3>
  {{product.name}}
  <em class="pull-right">${{product.price}}</em>
</h3>
```

index.html

An Attribute Directive

```
<h3 product-title></h3>
```

An Element Directive

```
<h3> <product-title></product-title> </h3>
```

What if we *need* a Controller?

```
<section ng-controller="PanelController as panels">
 <ul class="nav nav-pills"> . . . </ul>
 <div class="panel" ng-show="panels.isSelected(1)"> . . . </div>
 <div class="panel" ng-show="panels.isSelected(2)"> . . . </div>
 <div class="panel" ng-show="panels.isSelected(3)"> . . . </div>
</section>
```

index.html

Directive?

SHAPING UP WITH ANGULAR.JS

First, extract the template...

```
<h3> <product-title> </h3>
<product-panels ng-controller="PanelController as panels">
  . . .
</product-panels>
```

index.html

```
<section>
<ul class="nav nav-pills"> . . . </ul>
<div class="panel" ng-show="panels.isSelected(1)"> . . . </div>
<div class="panel" ng-show="panels.isSelected(2)"> . . . </div>
<div class="panel" ng-show="panels.isSelected(3)"> . . . </div>
</section>
```

product-panels.html

SHAPING UP WITH ANGULAR.JS

Now write the Directive ...

```
<product-panels ng-controller="PanelController as panels">
  . . .
</product-panels>
```

index.html

```
app.directive('productPanels', function(){
  return {
    restrict: 'E',
    templateUrl: 'product-panels.html'
  };
});
```

app.js

What about the Controller?

```
<product-panels ng-controller="PanelController as panels">
  . . .
</product-panels>
```

index.html

```
app.directive('productPanels', function(){
  return {
    restrict: 'E',
    templateUrl: 'product-panels.html'
  };
});
app.controller('PanelController', function(){
  . . .
});
```

app.js

First we need to move the functionality inside the directive

Moving the Controller Inside

```
<product-panels  ng-controller="PanelController as panels" >
  . . .
</product-panels>
```

index.html

Next, move the alias inside

```
app.directive('productPanels', function(){
  return {
    restrict: 'E',
    templateUrl: 'product-panels.html',
    controller:function(){
      . . .
    }
  };
});
```

app.js

Need to Specify the Alias

```
<product-panels>
  . . .
</product-panels>
```

index.html

```
app.directive('productPanels', function(){
  return {
    restrict: 'E',
    templateUrl: 'product-panels.html',
    controller:function(){
      . . .
    },
    controllerAs: 'panels'
  };
});
```

Now it works, using **panels** as our Controller Alias.

app.js

Flatlander Crafted Gems

– an Angular store –

Pentagonal Gem $5.95

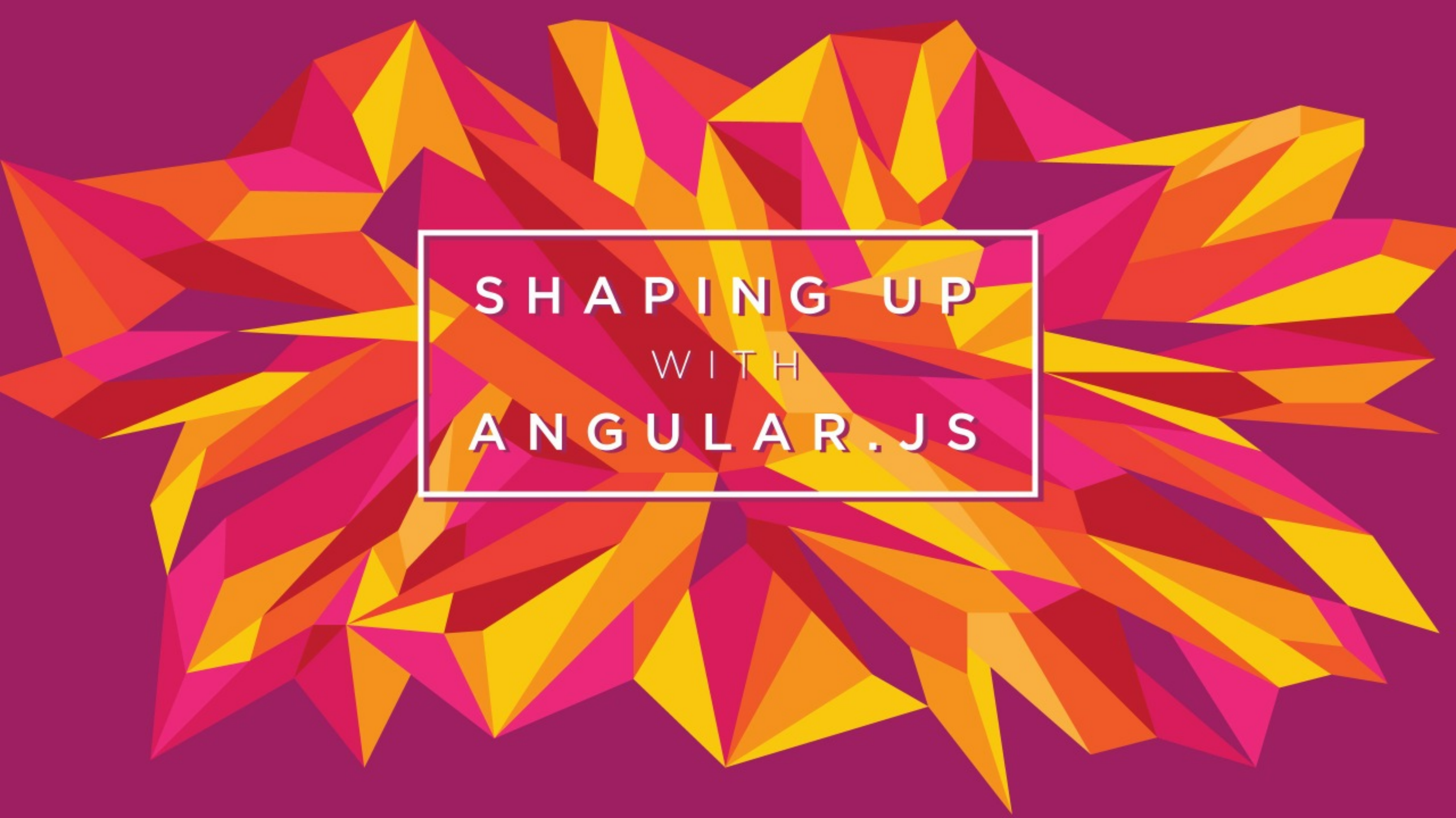
Description | Specifications | Reviews

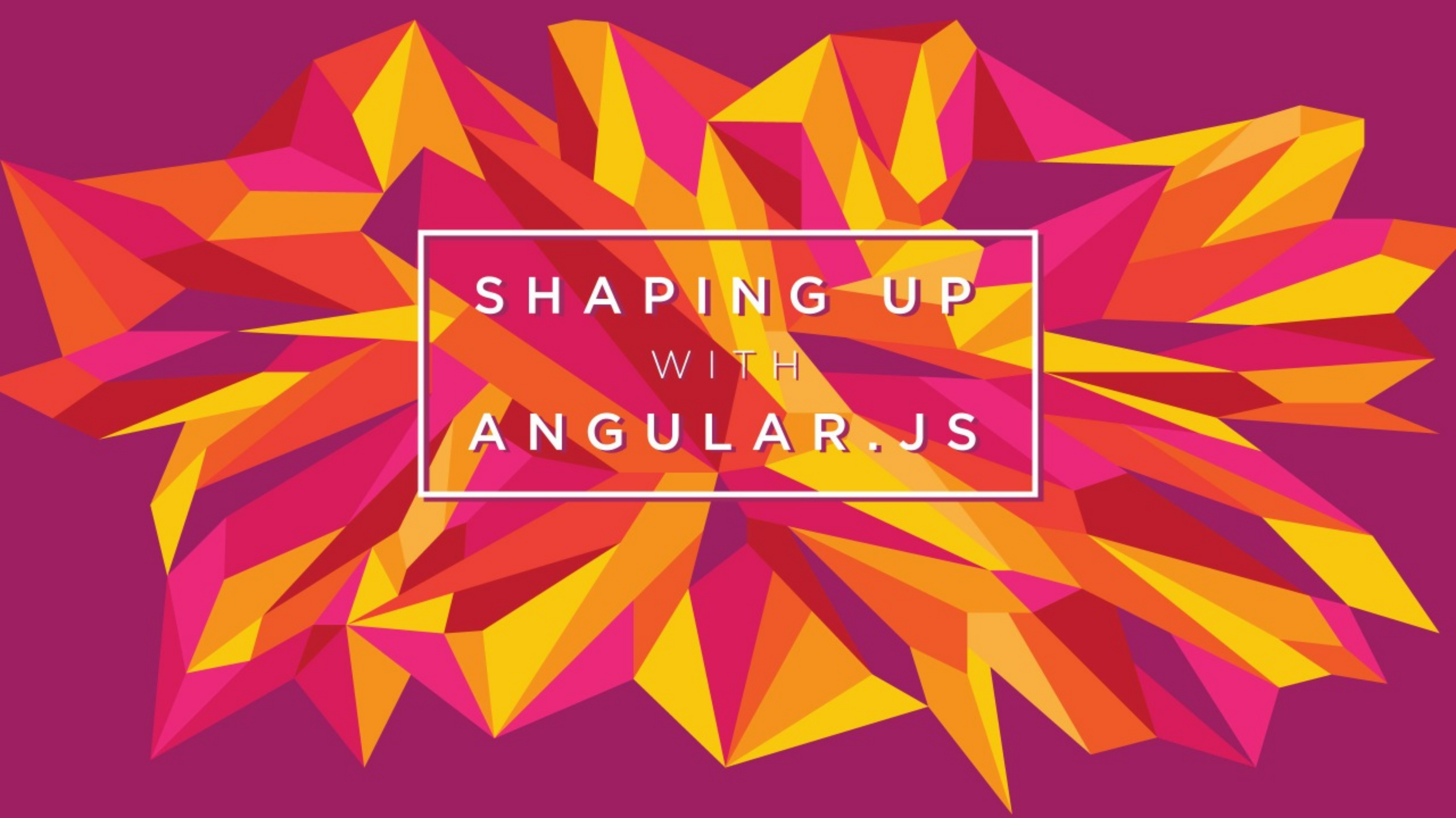
Description

> Origin of the Pentagonal Gem is unknown, hence its low value. It has a very high shine and 12 sides,

Challenges

SHAPING UP
WITH
ANGULAR.JS


SHAPING UP
WITH
ANGULAR.JS


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

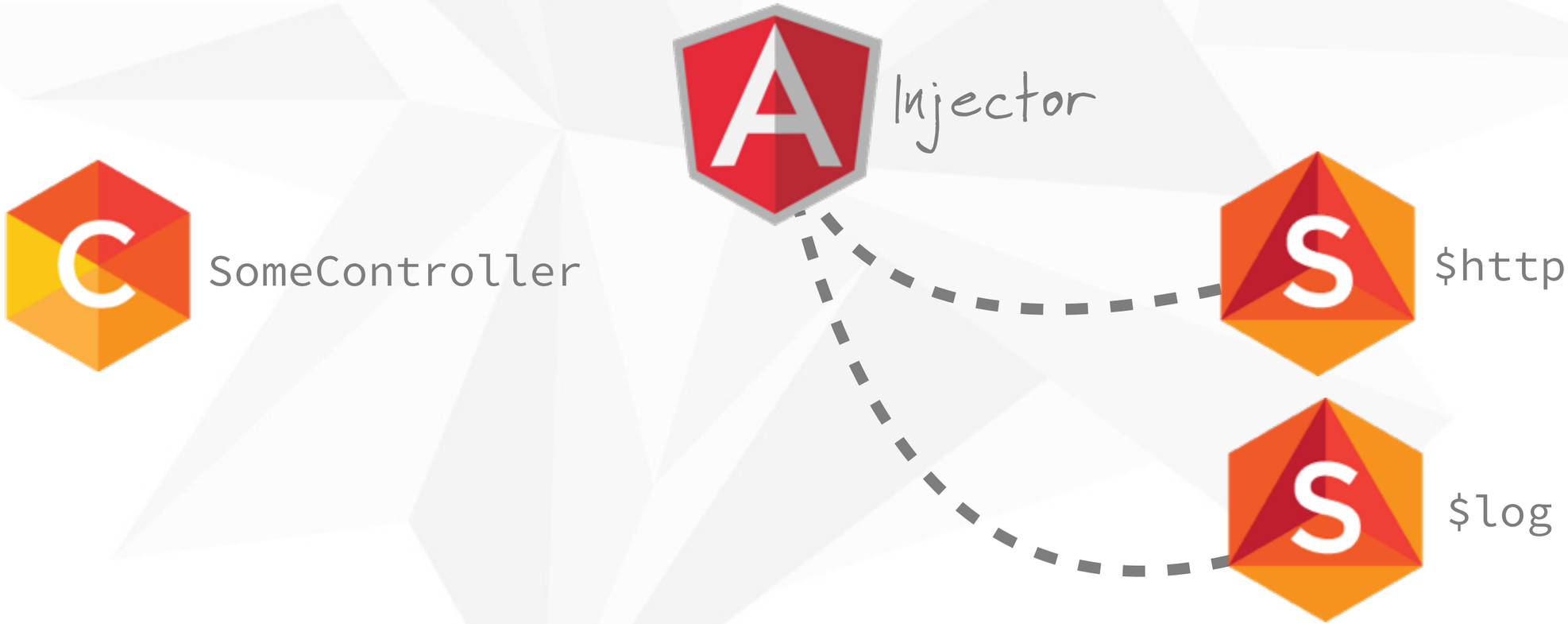
Dependency Injection!

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

} ]);
```

If you needed more than one

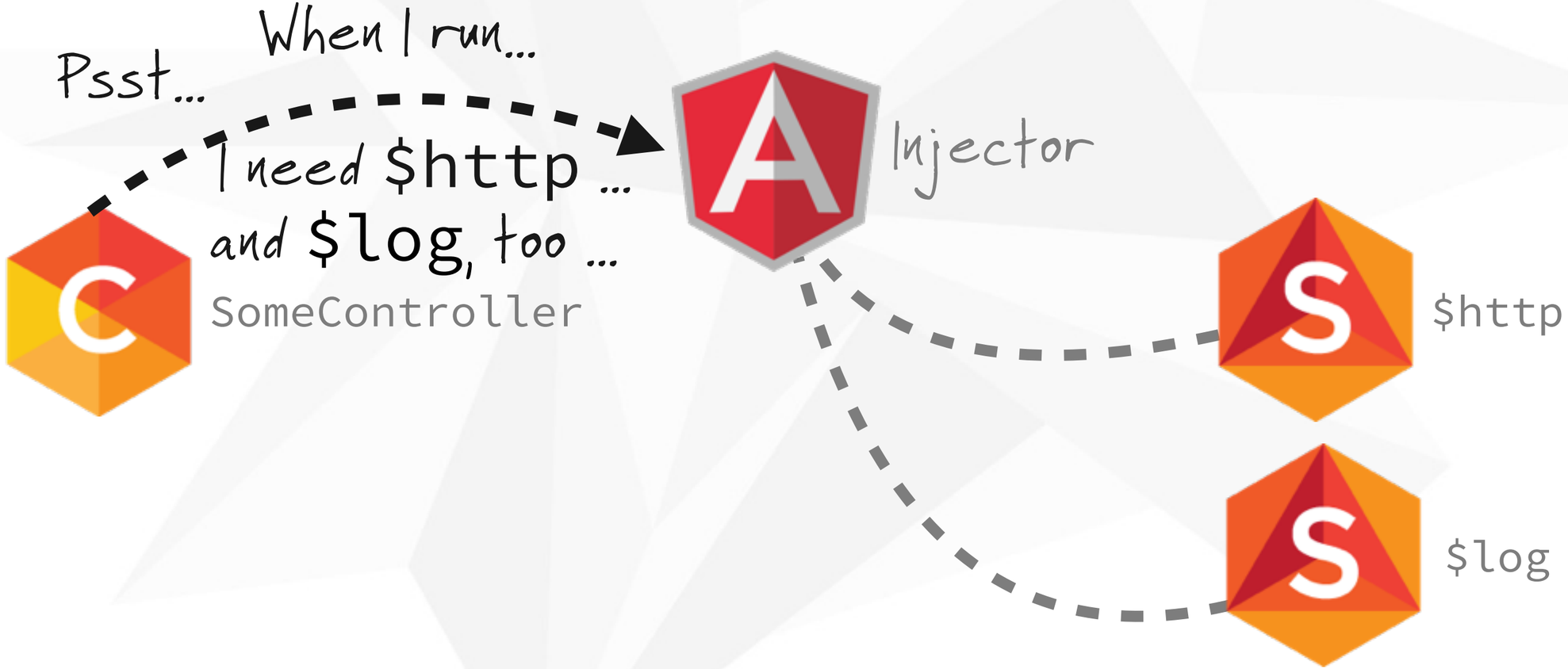
When Angular is Loaded Services are Registered

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

} ]);
```



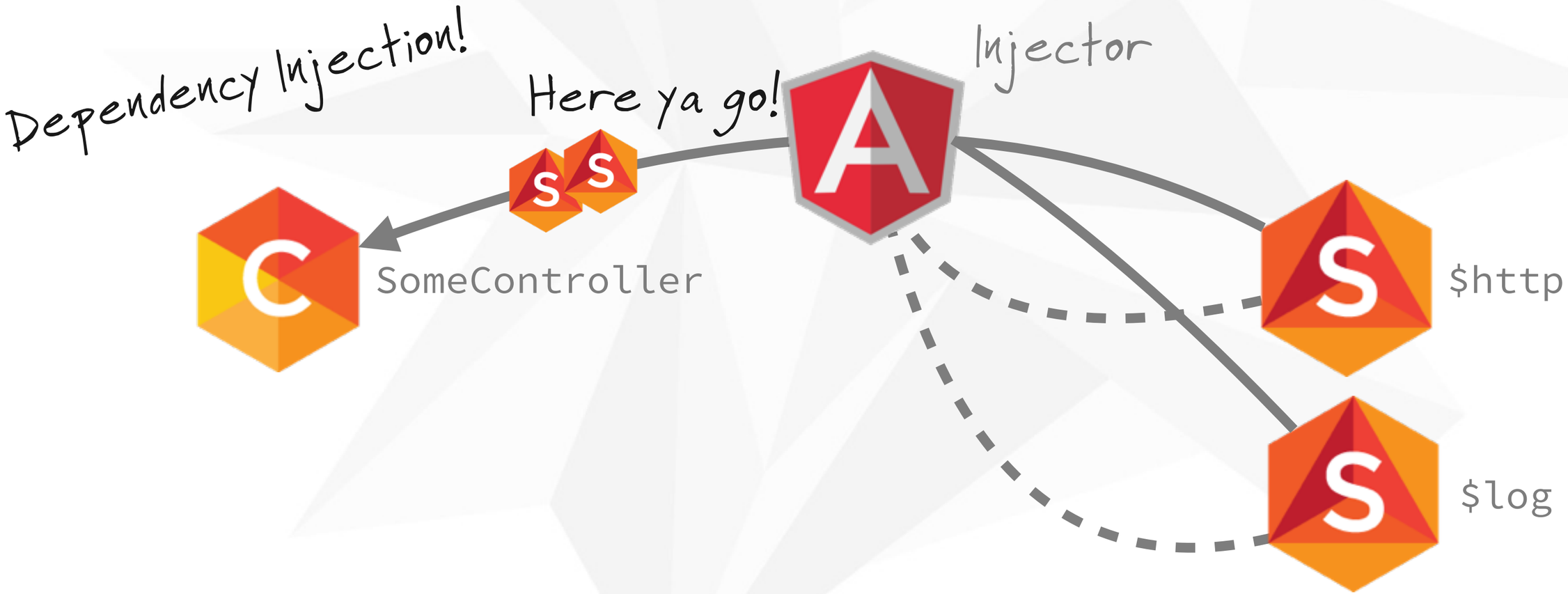
A Controller is Initialized

```
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:

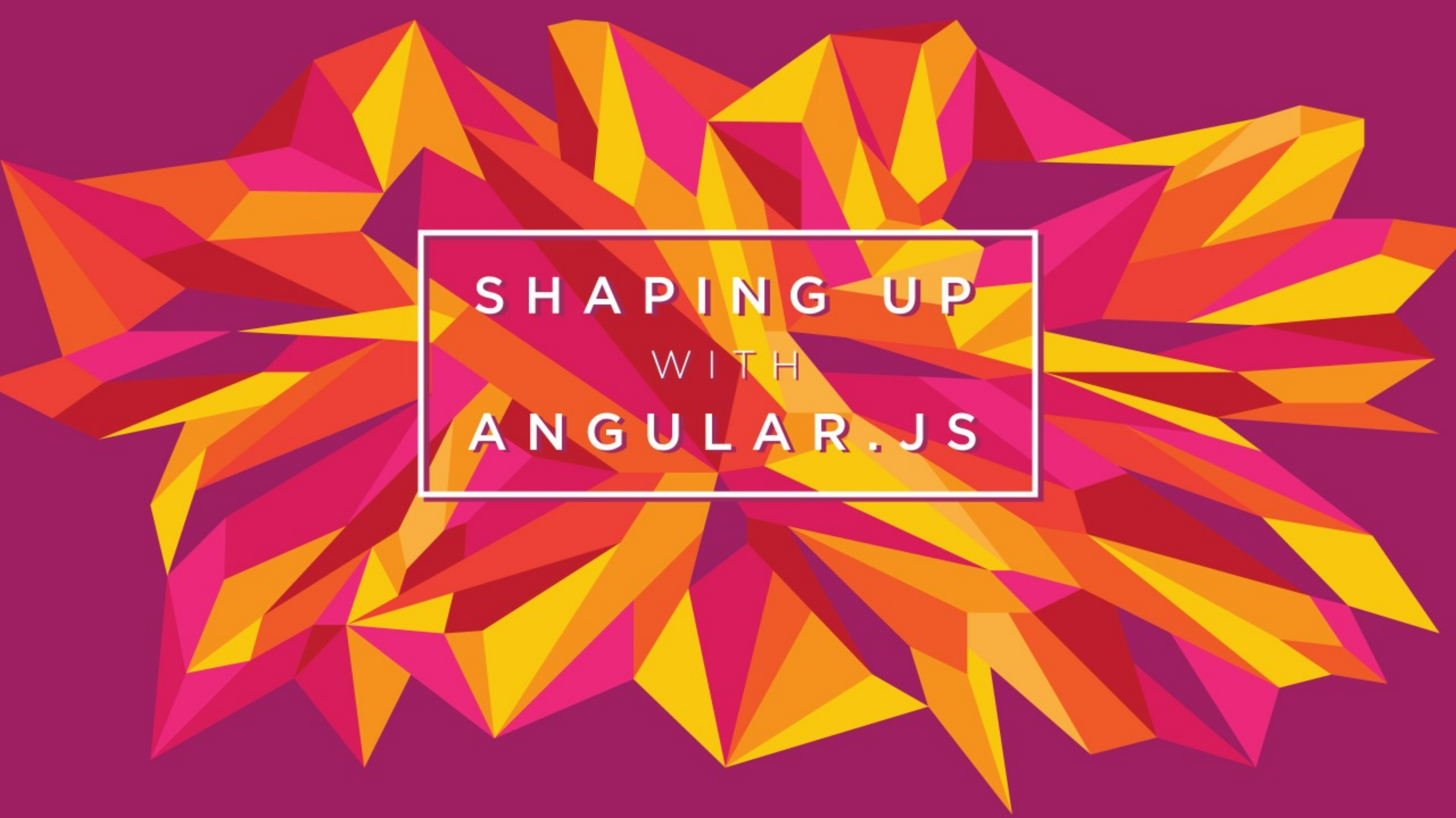
```
$http({ method: 'OPTIONS', url: '/path/to/resource.json' });
```

```
$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