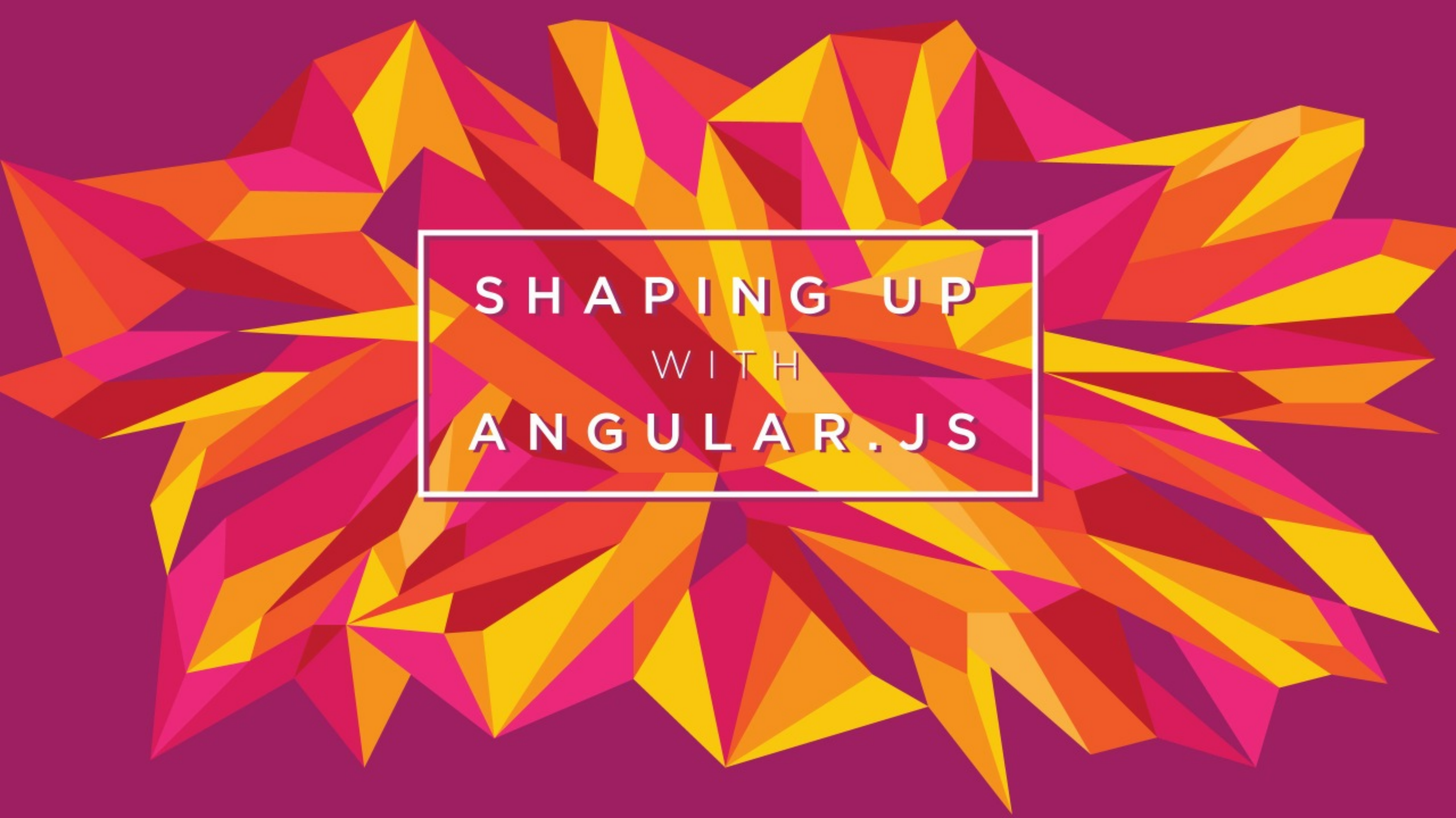

SHAPING UP
WITH
ANGULAR.JS

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

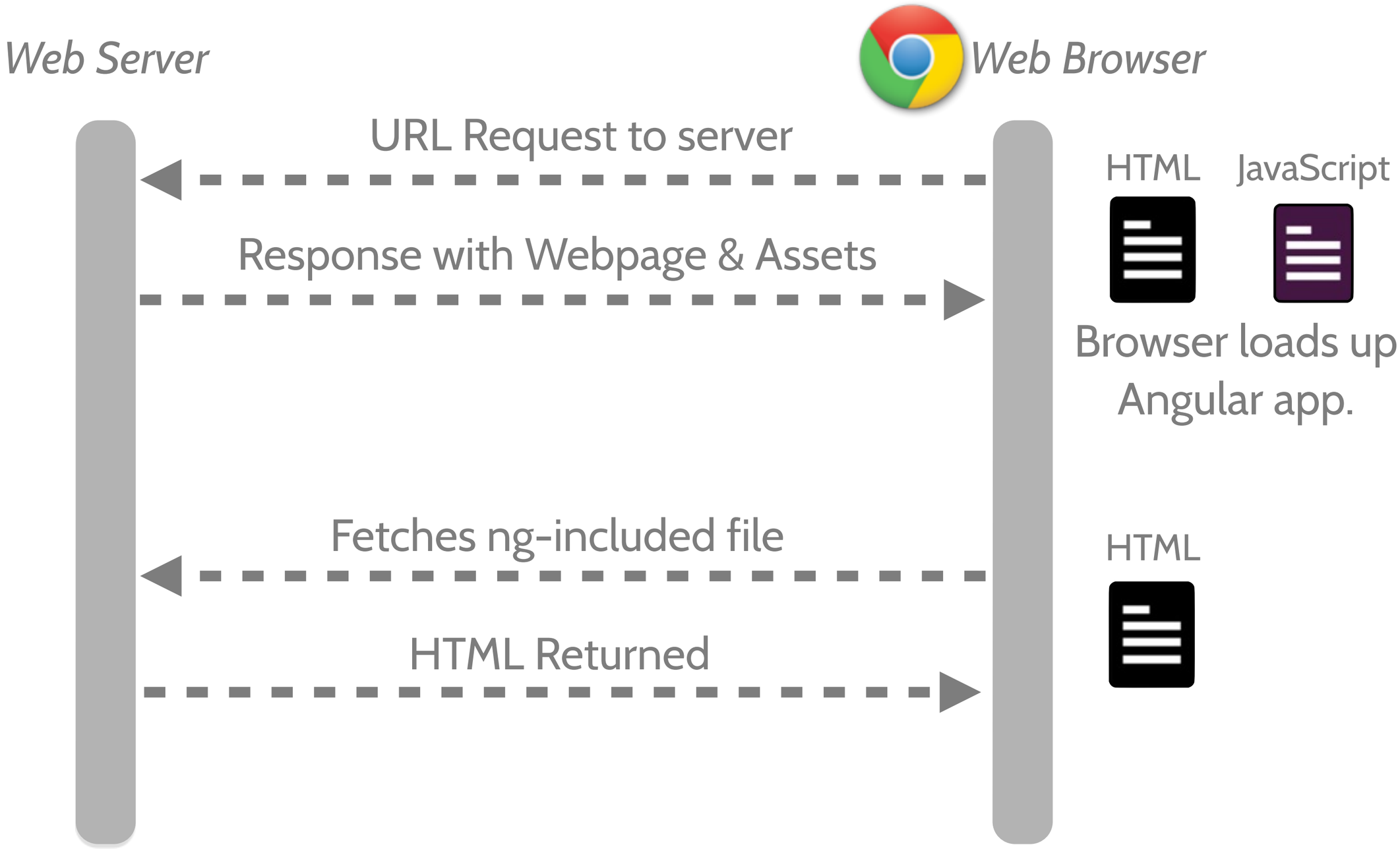
– an Angular store –

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

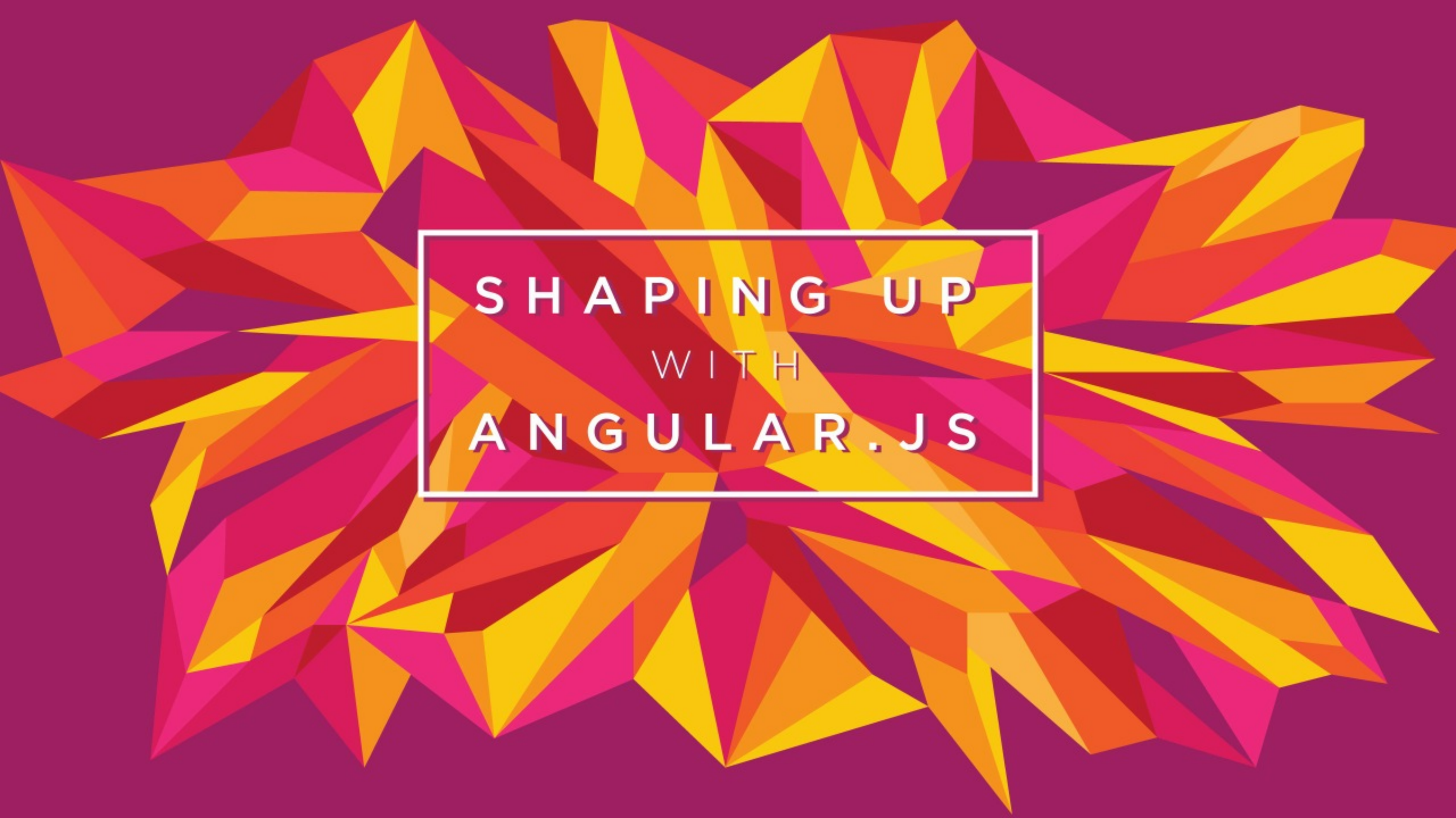
Description | Specifications | Reviews

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