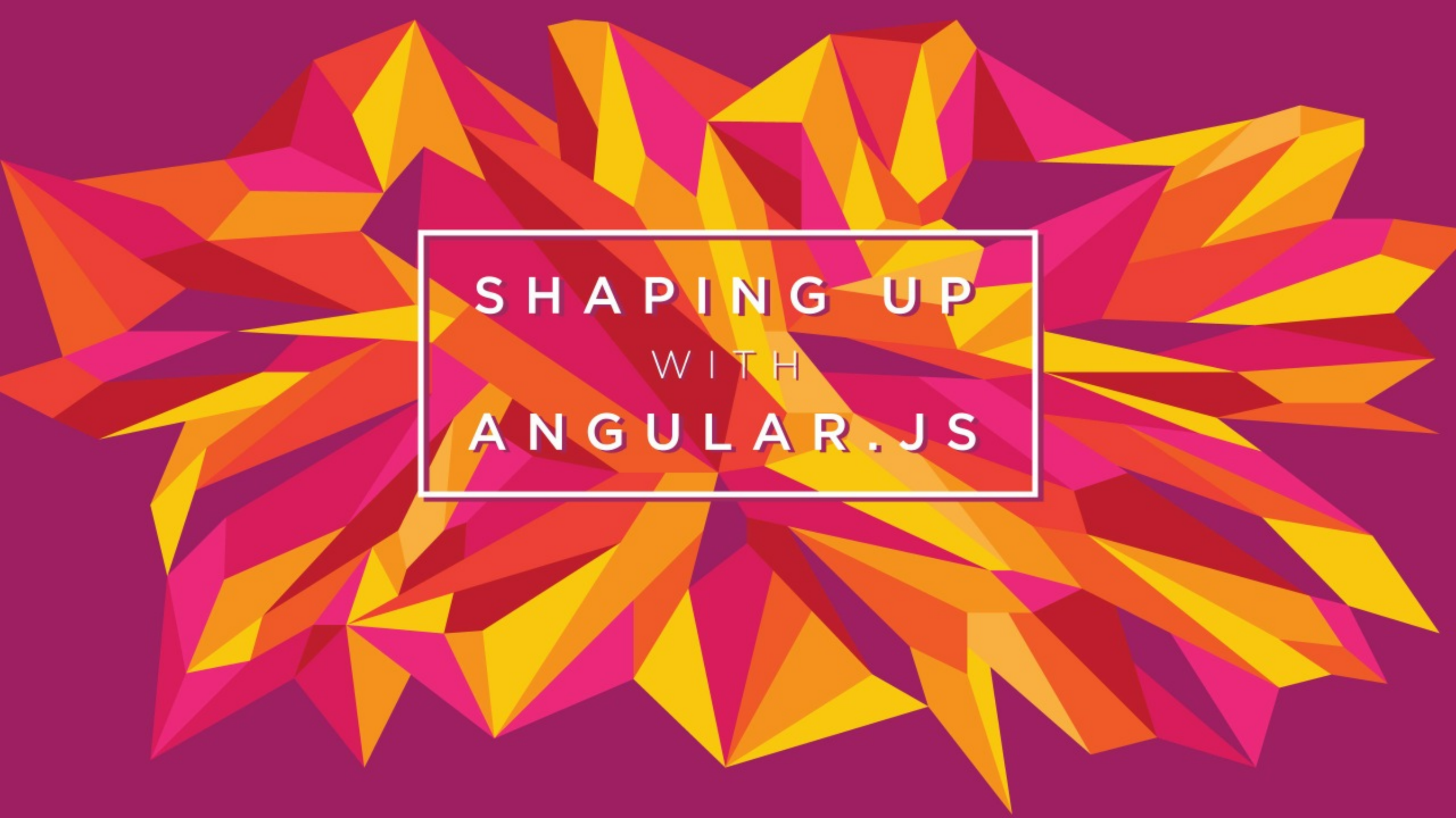
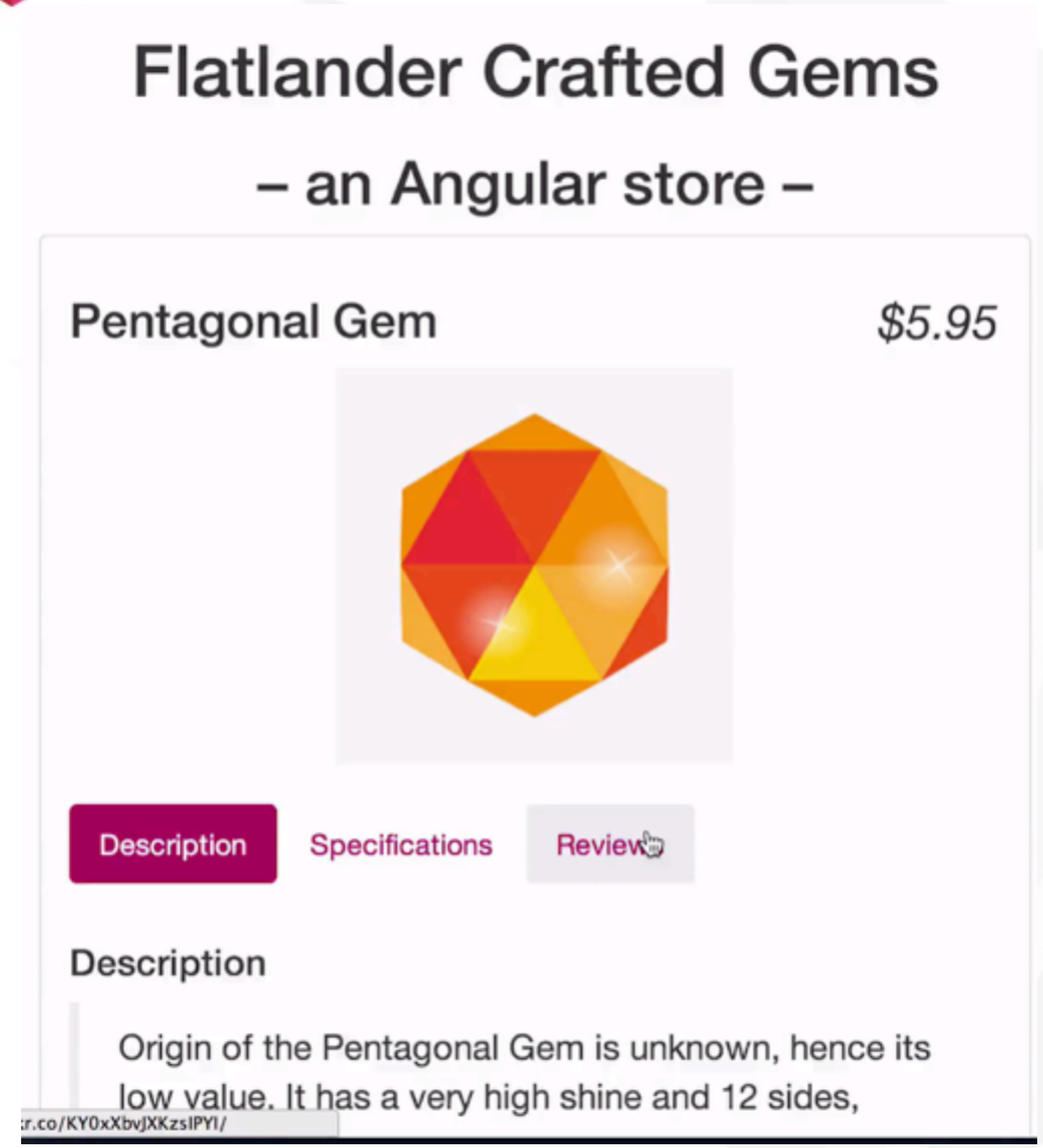

SHAPING UP
WITH
ANGULAR.JS

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

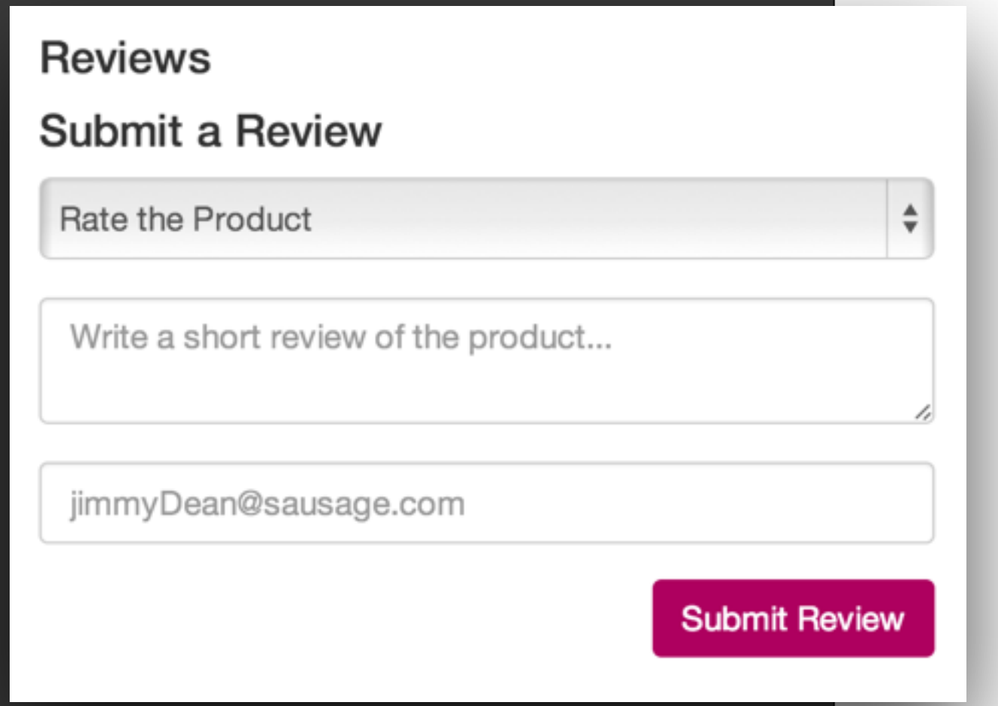
Nothing new here, but how do we start to implement forms?

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!

—gemsRock@alyssaNicoll.com

2 Stars reviewing products is fun
—

Submit a Review

2

reviewing products is fun

funreviewer

Submit Review

Dodecahedron $2.95

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 = {};
  };
});
```

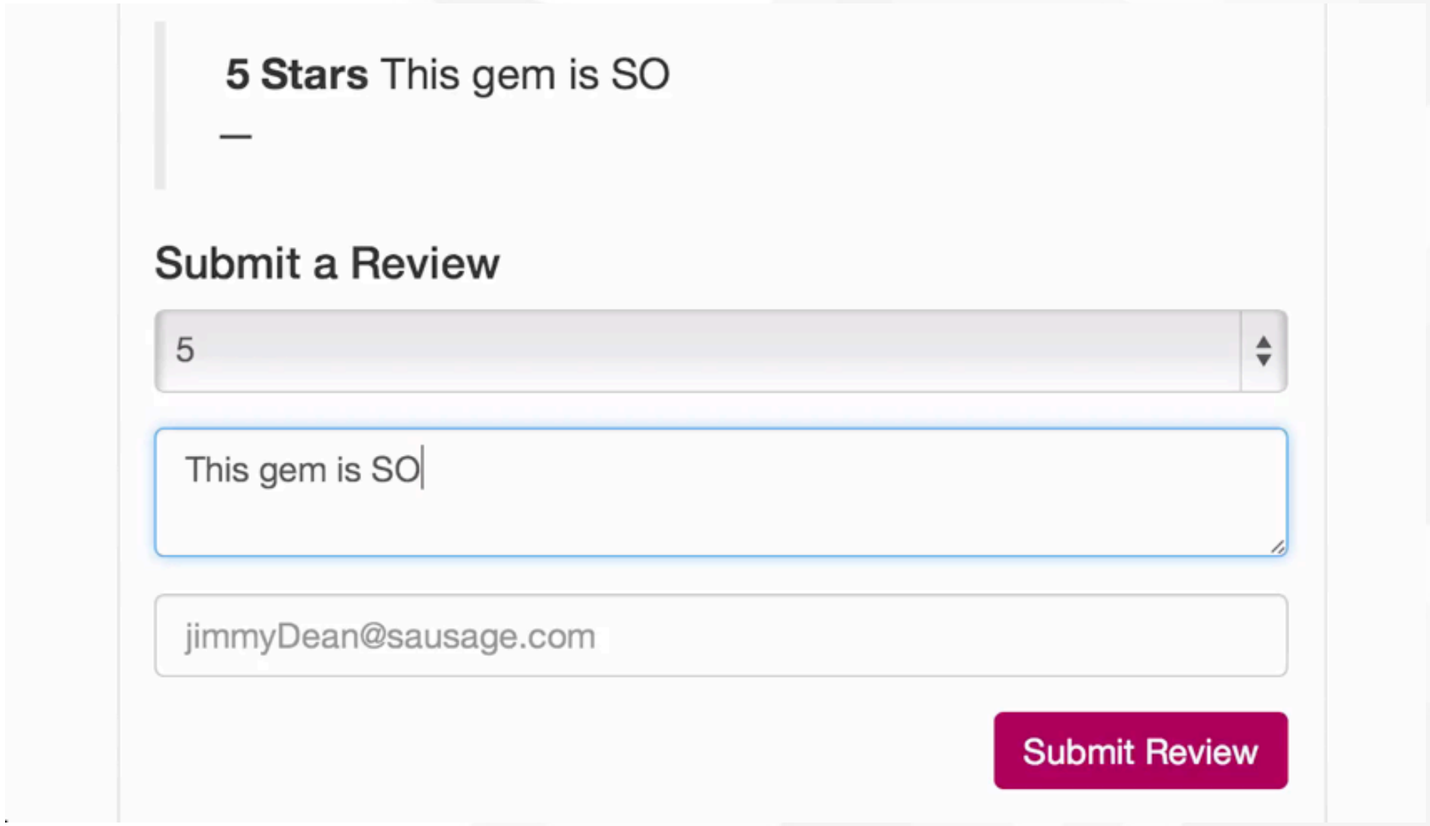
Clear out the review, so the form will reset.

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

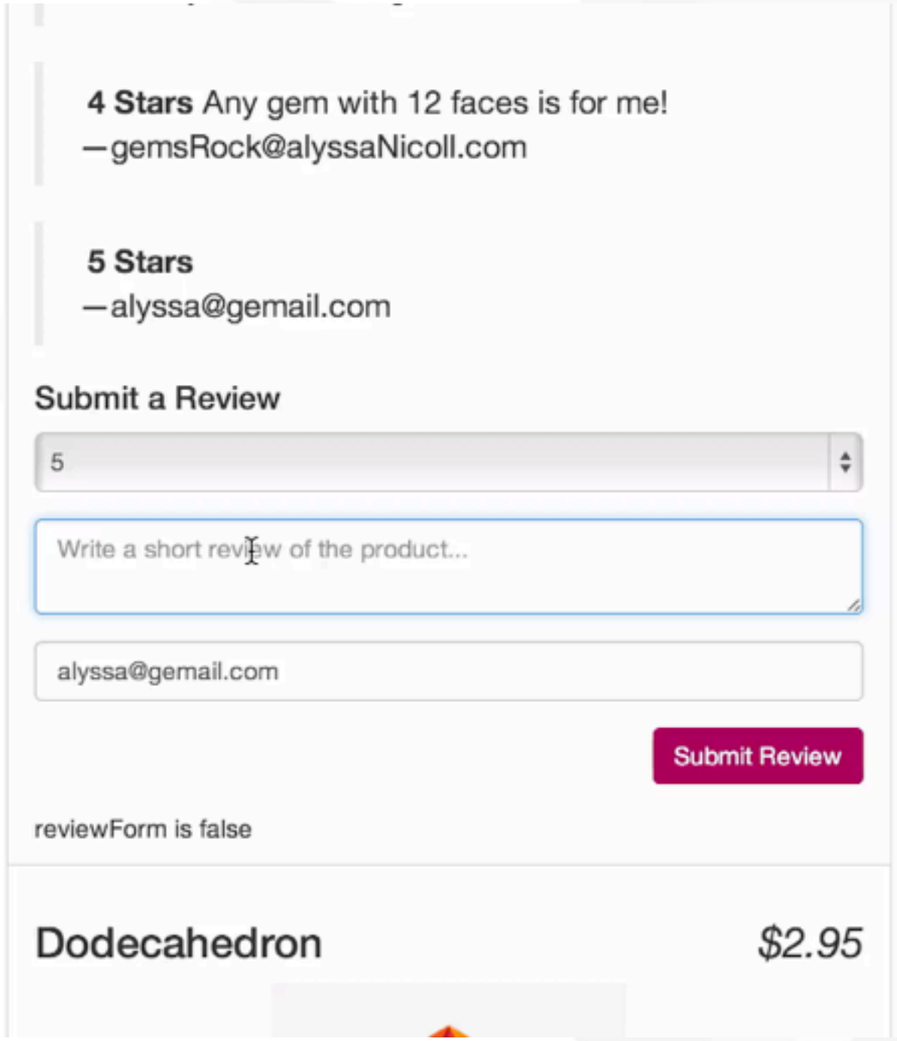
```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl"
    ng-submit="reviewForm.$valid && reviewCtrl.addReview(product)" novalidate>
```

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">
```

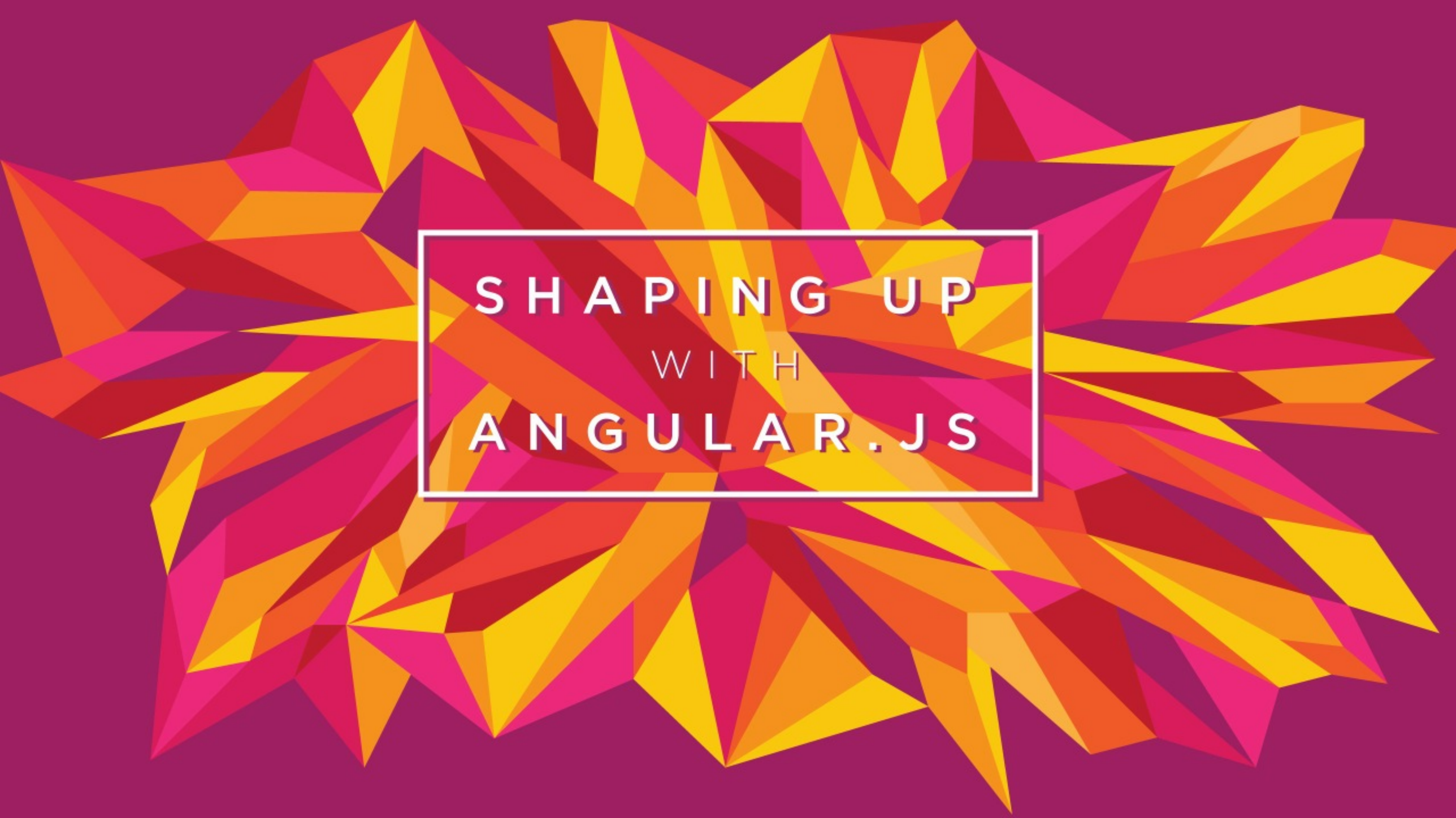
```
<input type="url" name="homepage">
```

```
<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