Startup Engineering

AngularJS 101 and Firebase

Sung Kim

So far

- AWS (EC2)
- Basic Linux Command
- HTML/CSS/Bootstrap
- Git/github
- AngularJS
- Android

Project Mission

Full stack

Project todo

- Proposal presentation on Jan 25, 3PM
  - Why we should use your web/mobile app?
  - Key requirements (features) + wireframe?
- Select and form your own team (3~4 people)
- Find something your team is interested in
- Use trello & github issue trackers from the beginning (brainstorming).

AngularJS Demo written by Sung

- http://q.comp3111.xyz
- http://kbill.org
- http://kproperty.xyz
- http://racephotos.org

AngularJS

O'REILLY®

Brad Green & Shyam Seshadri

AngularJS Features

1. Client template
2. Date binding
3. Dependency Injection
4. Directives

1. Client-side template

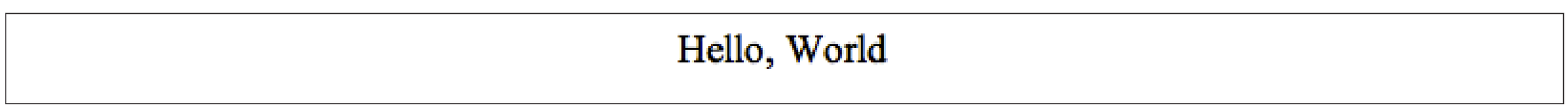
For it, we'll create our template in *hello.html*:

```
<html ng-app>
<head>
  <script src="angular.js"></script>
  <script src="controllers.js"></script>
</head>
<body>
  <div ng-controller='HelloController'>
    <p>{{greeting.text}}, World</p>
  </div>
</body>
</html>
```

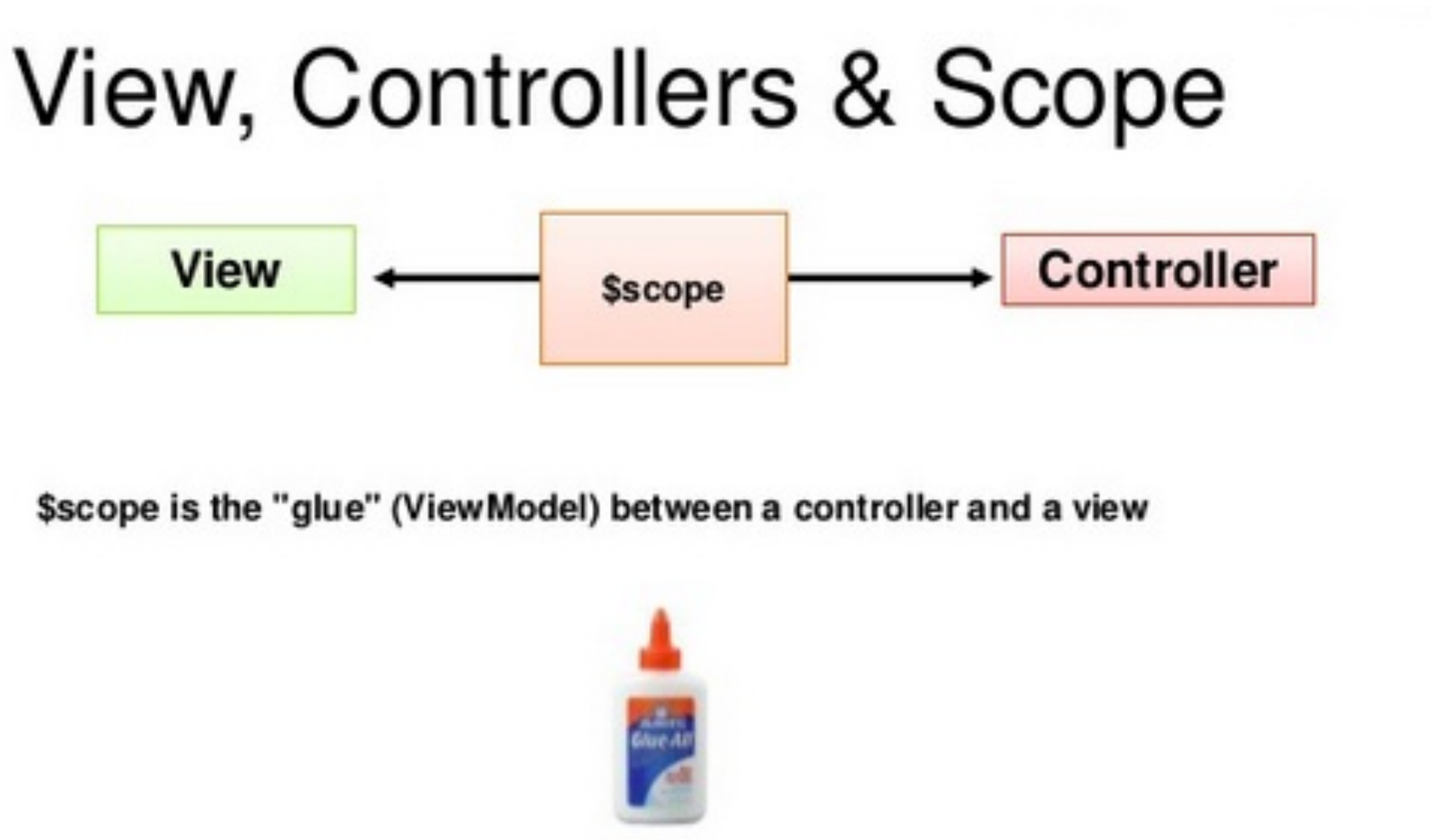
And our logic in *controllers.js*:

```
function HelloController($scope) {
  $scope.greeting = { text: 'Hello' };
}
```

Loading *hello.html* into any browser will then produce what we see in Figure 1-1:



Scope in AngularJS



http://www.c-sharpcorner.com/UploadFile/dev4634/scope-in-angular-js-for-begineers/

1. Client-side template

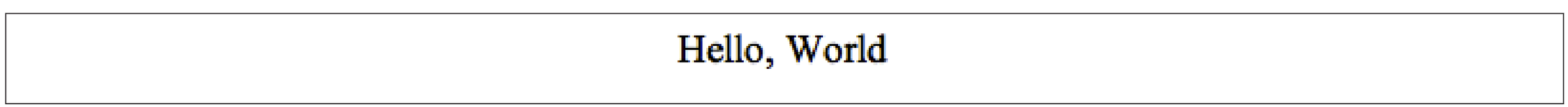
For it, we'll create our template in *hello.html*:

```
<html ng-app>
<head>
  <script src="angular.js"></script>
  <script src="controllers.js"></script>
</head>
<body>
  <div ng-controller='HelloController'>
    <p>{{greeting.text}}, World</p>
  </div>
</body>
</html>
```

And our logic in *controllers.js*:

```
function HelloController($scope) {
  $scope.greeting = { text: 'Hello' };
}
```

Loading *hello.html* into any browser will then produce what we see in Figure 1-1:



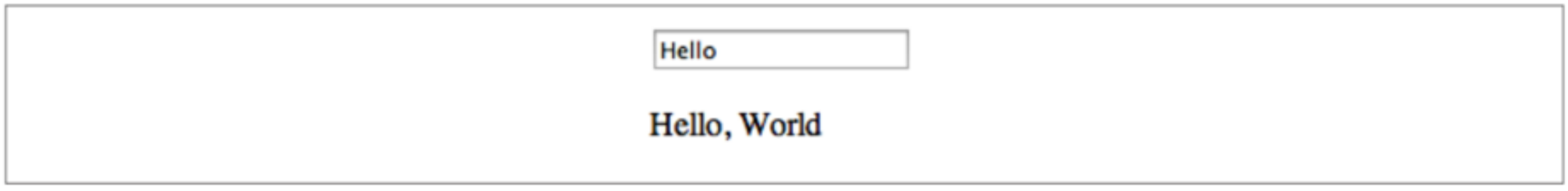
2. Data binding

Here's the new template:

```
<html ng-app>
<head>
  <script src="angular.js"></script>
  <script src="controllers.js"></script>
</head>
<body>
  <div ng-controller='HelloController'>
    <input ng-model='greeting.text'>
    <p>{{greeting.text}}, World</p>
  </div>
</body>
</html>
```

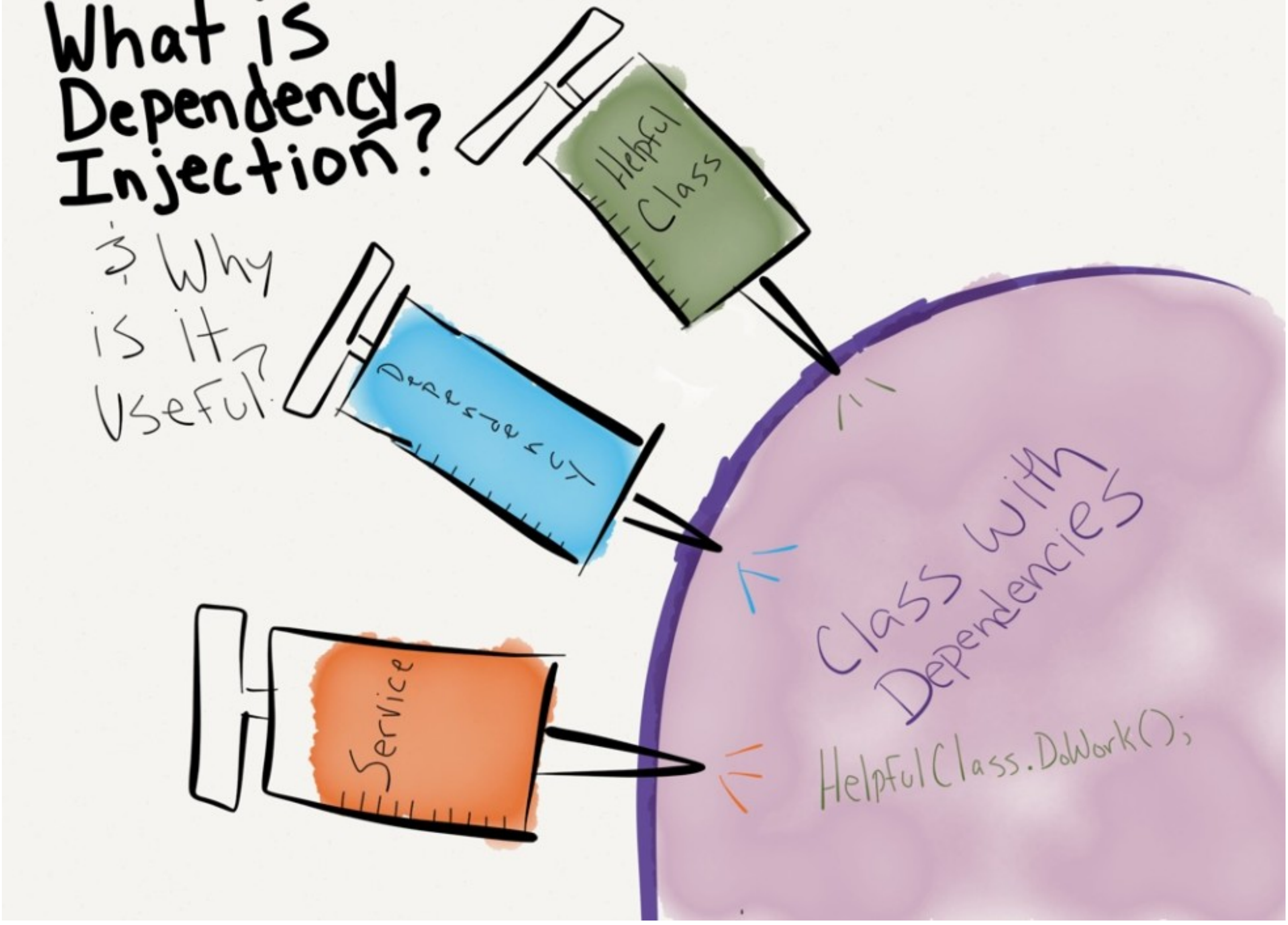
The controller, `HelloController`, can stay exactly the same.

Loading it in a browser, we'd see the screen captured in Figure 1-2.



What is Dependency Injection?

Why is it useful?



3. Dependency Injection

“In software engineering, dependency injection is a software design pattern that implements inversion of control for resolving dependencies. A dependency is an object that can be used (a service). An injection is the passing of a dependency to a dependent object (a client) that would use it.” Wikipedia, the free encyclopedia

841

Basically, instead of having your objects creating a dependency or asking a factory object to make one for them, you pass the needed dependencies in to the constructor or via property setters, and you make it somebody else's problem (an object further up the dependency graph, or a dependency injector that builds the dependency graph). A dependency as I'm using it here is any other object the current object needs to hold a reference to.

One of the major advantages of dependency injection is that it can make testing lots easier. Suppose you have an object which in its constructor does something like:

```
public SomeClass() {
    myObject = Factory.getObject();
}
```

This can be troublesome when all you want to do is run some unit tests on SomeClass, especially if myObject is something that does complex disk or network access. So now you're looking at mocking myObject but also somehow intercepting the factory call. Hard. Instead, pass the object in as an argument to the constructor. Now you've moved the problem elsewhere, but testing can become lots easier. Just make a dummy myObject and pass that in. The constructor would now look a bit like:

Dependency Injection

```
public SomeClass (MyClass myObject) {
    this.myObject = myObject;
}
```

Most people can probably work out the other problems that might arise when not using dependency injection while testing (like classes that do too much work in their constructors etc.) Most of this is stuff I picked up on the Google Testing Blog, to be perfectly honest...

http://stackoverflow.com/questions/130794/what-is-dependency-injection

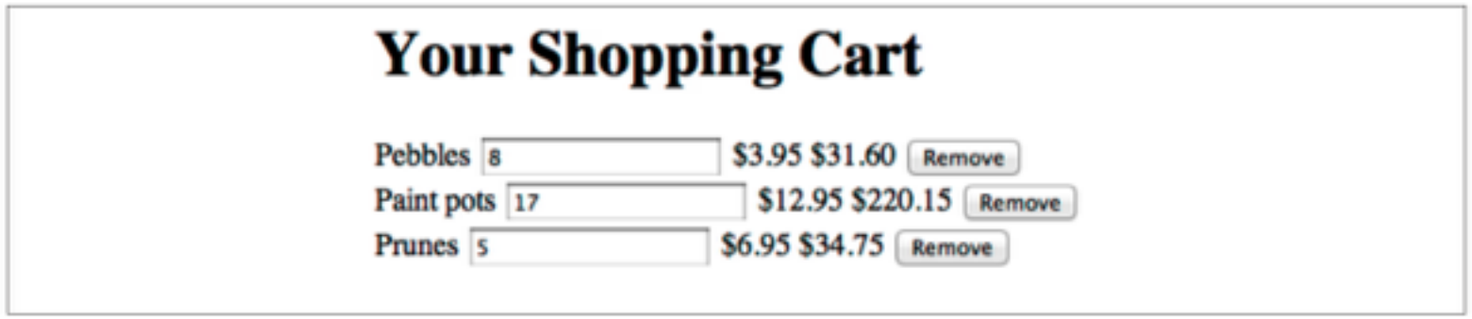
Dependency Injection in AngularJS

```
function HelloController($scope, $location) {
  $scope.greeting = { text: 'Hello' };
  // use $location for something good here...
}
```

Directives

- ng-app
- ng-controller
- ng-repeat
- ng-click
- ng-model
- ng-show

Example




```
<html ng-app='myApp'>
<head>
  <title>Your Shopping Cart</title>
</head>
<body ng-controller='CartController'>
  <h1>Your Order</h1>
  <div ng-repeat='item in items'>
    <span>{{item.title}}</span>
    <input ng-model='item.quantity'>
    <span>{{item.price | currency}}</span>
    <span>{{item.price * item.quantity | currency}}</span>
    <button ng-click="remove($index)">Remove</button>
  </div>
  <script src="angular.js"></script>
  <script>
    function CartController($scope) {
      $scope.items = [
        {title: 'Paint pots', quantity: 8, price: 3.95},
        {title: 'Polka dots', quantity: 17, price: 12.95},
        {title: 'Pebbles', quantity: 5, price: 6.95}
      ];

      $scope.remove = function(index) {
        $scope.items.splice(index, 1);
      }
    }
  </script>
  </body>
</html>
```

Anatomy of AngularJS Apps

Invoking Angular

Any application must do two things to start Angular:

1. Load the *angular.js* library
2. Tell Angular which part of the DOM it should manage with the `ng-app` directive

Loading the Script

Loading the library is straightforward and follows the same rules as any other JavaScript library. You can load the script from Google's content delivery network (CDN), like so:

```
<script
    src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.4/angular.min.js">
</script>
```

Declaring Angular's Boundaries with ng-app

The `ng-app` directive lets you tell Angular which part of your page it should expect to manage. If you're building an all-Angular application, you should include `ng-app` as part of the `<html>` tag, like so:

```
<html ng-app>
  …
</html>
```

This tells Angular to manage all DOM elements in the page.

If you've got an existing app where some other technology expects to manage the DOM, such as Java or Rails, you can tell Angular to manage only a part of the page by placing it on some element like a `<div>` within the page.

```
<html>
  …
  <div ng-app>
    …
  </div>
  …
</html>
```

Model View Controller

- A model containing data that represents the current state of your application.
- Views that display this data.
- Controllers that manage the relationship between your model and your views.

```
var someText = 'You have started your journey.';
```

```
<p>{{someText}}</p>
```

The controllers are classes or types you write to tell Angular which objects or primitives make up your model by assigning them to the `$scope` object passed into your controller:

```
function TextController($scope) {
  $scope.someText = someText;
}
```

```
<html ng-app>
<body ng-controller="TextController">
  <p>{{someText}}</p>

  <script
      src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.1/angular.min.js">
  </script>

  <script>
    function TextController($scope) {
      $scope.someText = 'You have started your journey.';
    }
  </script>
</body>
</html>
```

Module

```
<html ng-app='myApp'>
<body ng-controller='TextController'>
  <p>{{someText.message}}</p>

<script
    src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.1/angular.min.js">
</script>
```

```
<script>
  var myAppModule = angular.module('myApp', []);

  myAppModule.controller('TextController',
      function($scope) {
    var someText = {};
    someText.message = 'You have started your journey.';
    $scope.someText = someText;
  });
</script>
</body>
</html>
```

Data binding

```
<form ng-controller="SomeController">
  <input type="checkbox" ng-model="youCheckedIt">
</form>
```

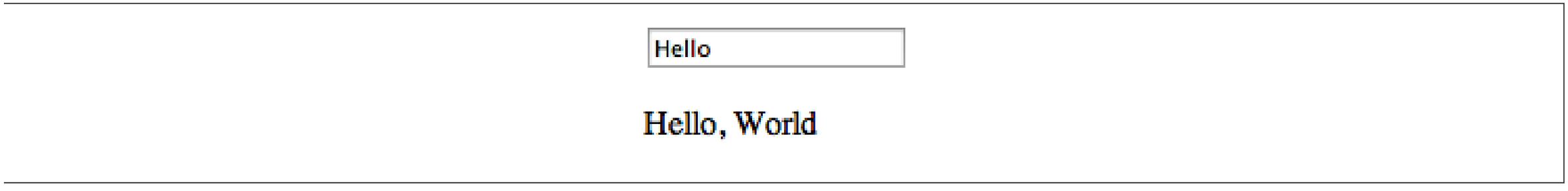
This means that:

1. When the user checks the box, a property called `youCheckedIt` on the `SomeController`'s `$scope` will become true. Unchecking the box makes `youCheckedIt` false.
2. If you set `$scope.youCheckedIt` to true in `SomeController`, the box becomes checked in the UI. Setting it to false unchecks the box.

```
<html ng-app>
<head>
  <script src="angular.js"></script>
  <script src="controllers.js"></script>
</head>
<body>
  <div ng-controller='HelloController'>
    <input ng-model='greeting.text'>
    <p>{{greeting.text}}, World</p>
  </div>
</body>
</html>
```

The controller, `HelloController`, can stay exactly the same.

Loading it in a browser, we'd see the screen captured in Figure 1-2.



ng-change

```
<form ng-controller="StartUpController">
  Starting: <input
                    ng-model="funding.startingEstimate">
  Recommendation: {{funding.needed}}
</form>
```

```
function StartUpController($scope) {
  $scope.funding = { startingEstimate: 0 };

  $scope.computeNeeded = function() {
    $scope.needed = $scope.startingEstimate * 10;
  };
}
```

watch?

```
<form ng-controller="StartUpController">
  Starting: <input ng-change="computeNeeded()"
                   ng-model="funding.startingEstimate">
  Recommendation: {{funding.needed}}
</form>
```

```
function StartUpController($scope) {
  $scope.funding = { startingEstimate: 0 };

  $scope.computeNeeded = function() {
    $scope.needed = $scope.startingEstimate * 10;
  };
}
```

watch?

```
<form ng-controller="StartUpController">
  Starting: <input ng-change="computeNeeded()"
                   ng-model="funding.startingEstimate">
  Recommendation: {{funding.needed}}
</form>
```

```
function StartUpController($scope) {
  $scope.funding = { startingEstimate: 0 };

  computeNeeded = function() {
    $scope.funding.needed = $scope.funding.startingEstimate * 10;
  };

  $scope.$watch('funding.startingEstimate', computeNeeded);
}
```

ng-submit

```
<form ng-submit="requestFunding()" ng-controller="StartUpController">
  Starting: <input ng-change="computeNeeded()" ng-model="startingEstimate">
  Recommendation: {{needed}}
  <button>Fund my startup!</button>
</form>

function StartUpController($scope) {
  $scope.computeNeeded = function() {
    $scope.needed = $scope.startingEstimate * 10;
  };

  $scope.requestFunding = function() {
    window.alert("Sorry, please get more customers first.");
  };
}
```

ng-click

```
<div ng-click="doSomething()">...</div>
```

```
<form ng-submit="requestFunding()" ng-controller="StartUpController">
  Starting: <input ng-change="computeNeeded()" ng-model="startingEstimate"
  Recommendation: {{needed}}
  <button>Fund my startup!</button>
  <button ng-click="reset()">Reset</button>
</form>

function StartUpController($scope) {
  $scope.computeNeeded = function() {
    $scope.needed = $scope.startingEstimate * 10;
  };

  $scope.requestFunding = function() {
    window.alert("Sorry, please get more customers first.");
  };
```

List, Tables, and Other repeated Elements

```
var students = [{name:'Mary Contrary', id:'1'},
                {name:'Jack Sprat', id:'2'},
                {name:'Jill Hill', id:'3'}];

function StudentListController($scope) {
  $scope.students = students;
}
```

To display this list of students, we can do something like the following:

```
<ul ng-controller=''>
  <li ng-repeat='student in students'>
    <a href='/student/view/{{student.id}}'>{{student.name}}</a>
  </li>
</ul>
```

Automatic update (data-binding), of course!

```
var students = [{name:'Mary Contrary', id:'1'},
                {name:'Jack Sprat', id:'2'},
                {name:'Jill Hill', id:'3'}];

function StudentListController($scope) {
  $scope.students = students;

  $scope.insertTom = function () {
    $scope.students.splice(1, 0, {name:'Tom Thumb', id:'4'});
  };
}
```

and adding a button to invoke it in the template:

```
<ul ng-controller=''>
  <li ng-repeat='student in students'>
    <a href='/student/view/{{student.id}}'>{{student.name}}</a>
  </li>
</ul>
<button ng-click="insertTom()">Insert</button>
```

we now see:

- Mary Contrary
- Tom Thumb
- Jack Sprat
- Jill Hill

ng-repeat comes with $index

```
<table ng-controller='AlbumController'>
  <tr ng-repeat='track in album'>
    <td>{{$index + 1}}</td>
    <td>{{track.name}}</td>
    <td>{{track.duration}}</td>
  </tr>
</table>
```

Hiding and Showing
ng-hide & ng-show

```
<div ng-controller='DeathrayMenuController'>
  <button ng-click='toggleMenu()'>Toggle Menu</button>
  <ul ng-show='menuState.show'>
    <li ng-click='stun()'>Stun</li>
    <li ng-click='disintegrate()'>Disintegrate</li>
    <li ng-click='erase()'>Erase from history</li>
  </ul>
<div/>

function DeathrayMenuController($scope) {
  $scope.menuState.show = false;

  $scope.toggleMenu = function() {
    $scope.menuState.show = !$scope.menuState.show;
  };

  // death ray functions left as exercise to reader
}
```

CSS Classes and Styles

Given this CSS:

```
.menu-disabled-true {
  color: gray;
}
```

you could show the `stun` function of your `DeathRay` as disabled with this template:

```
<div ng-controller='DeathrayMenuController'>
  <ul>
    <li class='menu-disabled-{{isDisabled}}' ng-click='stun()'>Stun</li>
    ...
  </ul>
<div/>
```

where you'd set the `isDisabled` property via your controller as appropriate:

```
function DeathrayMenuController($scope) {
  $scope.isDisabled = false;

  $scope.stun = function() {
    // stun the target, then disable menu to allow regeneration
    $scope.isDisabled = 'true';
  };
}
```

```
.error {
  background-color: red;
}

.warning {
  background-color: yellow;
}
```

```
<div ng-controller='HeaderController'>
  ...
  <div ng-class='{error: isError, warning: isWarning}'>{{messageText}}</div>
  …
  <button ng-click='showError()'>Simulate Error</button>
  <button ng-click='showWarning()'>Simulate Warning</button>
</div>
```

```
function HeaderController($scope) {
  $scope.isError = false;
  $scope.isWarning = false;

  $scope.showError = function() {
    $scope.messageText = 'This is an error!';
    $scope.isError = true;
    $scope.isWarning = false;
  };

  $scope.showWarning = function() {
    $scope.messageText = 'Just a warning.  Please carry on.';
    $scope.isWarning = true;
    $scope.isError = false;
  };
}
```

src and href attributes

When data binding to an `<img>` or `<a>` tag, the obvious path of using {{ }} in the src or href attributes won't work well. Because browsers are aggressive about loading images parallel to other content, Angular doesn't get a chance to intercept data binding requests. While the obvious syntax for an `<img>` might be:

```
<img src="/images/cats/{{favoriteCat}}">
```

Instead, you should use the `ng-src` attribute and write your template as:

```
<img ng-src="/images/cats/{{favoriteCat}}">
```

Similarly, for the `<a>` tag, you should use `ng-href`:

```
<a ng-href="/shop/category={{numberOfBalloons}}">some text</a>
```

Expressions (but bring them in to JS)

simple math (+, -, /, *, %), make comparisons (==, !=, >, <, >=, ⇐), perform boolean logic (&&, ||, !) and bitwise operations (\^, &, |). You can call functions you expose on `$scope` in your controller and you can reference arrays and object notation ([ ], { }, .).

All of these are valid examples of expressions:

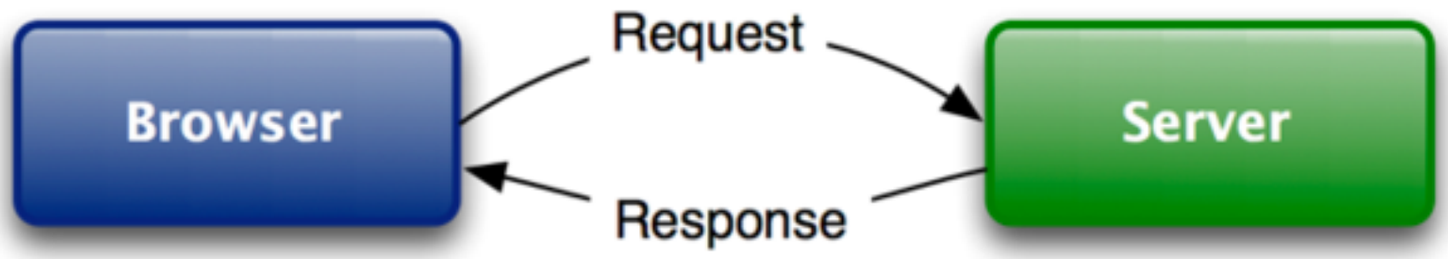
```
<div ng-controller='SomeController'>
  <div>{{recompute() / 10}}</div>
  <ul ng-repeat='thing in things'>
    <li ng-class='{highlight: $index % 4 >= threshold($index)}'>
      {{otherFunction($index)}}
    </li>
  </ul>
</div>
```

Talking to Servers

Okay, enough messing around. Real apps generally talk to real servers. Mobile apps and the emerging Chrome desktop apps may be exceptions, but for everything else, whether you want persistence in the cloud or real-time interactions with other users, you probably want your app to talk to a server.

For this, Angular provides a service called `$http`. It has an extensive list of abstractions that make it easier to talk to servers. It supports vanilla HTTP, JSONP, and CORS.

Understanding HTTP Basics

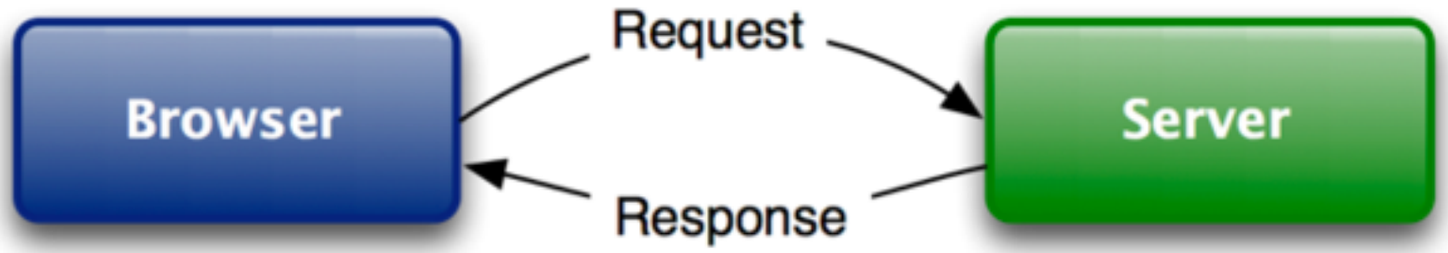



```
1  GET /home?pageId=c5789534 HTTP/1.1
2  Host: www.buildvsbreak.com
3  User-Agent: Mozilla/5.0
4  Accept: text/html,application/xhtml+xml,application/xml;
5  Accept-Language: en;q=0.5
6  Accept-Encoding: gzip, deflate
7  DNT: 1
8  Connection: keep-alive
```

```
 1  HTTP/1.1 200 OK
 2  Content-Type: text/html
 3  Date: Mon 15 Jul 2013 20:48:49 GMT
 4  Server: Apache/2.2.22 (Ubuntu)
 5  X-Powered-By: PHP/5.3.10-1 ubuntu3.2
 6  Content-Length: 2838
 7
 8  <!DOCTYPE html>
 9  <html xmlns="http://www.w3.org/1999/xhtml">
10  <head>
11  <META http-equiv="Content-Type" content="text/html;
12  <title>Build vs Break Technical Training</title>
13  <meta name="keywords" content="Programming Security
```

http://learn.onemonth.com/understanding-http-basics

Understanding HTTP Basics




```
1  POST /signup HTTP/1.1
2  Host: www.buildvsbreak.com
3  User-Agent: Mozilla/5.0
4  Accept: */*
5  Accept-Language: en-US,en;q=0.5
6  Content-Type: application/x-www-form-urlencoded;
7  Referer: http://www.buildvsbreak.com/sign-up/
8  Content-Length: 114
9  Cookie: ubpv=a%2C73fa268a-e729-11e2-a9c8-12313e02a4f0;
10
11 name=Jon&email=jrose400%40gmail.com&company=test
```

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html
3  Date: Mon 15 Jul 2013 20:48:49 GMT
4  Server: Apache/2.2.22 (Ubuntu)
5  X-Powered-By: PHP/5.3.10-1 ubuntu3.2
6  Content-Length: 2838
7
8  <!DOCTYPE html>
9  <html xmlns="http://www.w3.org/1999/xhtml">
10 <head>
11 <META http-equiv="Content-Type" content="text/html;
12 <title>Build vs Break Technical Training</title>
13 <meta name="keywords" content="Programming Security
```

http://learn.onemonth.com/understanding-http-basics

AN SEO'S GUIDE TO HTTP STATUS CODES

Every web page you visit returns a status code, to give the browser additional information and instructions. Search bots see these codes, and some of them can impact SEO. Here are a few of the big ones:

CAST OF CHARACTERS

HTTP STATUS CODES

200 OK/Success — Everyone arrives at Page A. There is much rejoicing!

301 Permanent* — Everyone is redirected to the new location, Page B.

302 Temporary* — Visitors and bots are redirected. Juice is left behind.

404 Not Found — Original page is gone. Visitors may see a 404 page.

500 Server Error — No page is returned. Everyone is lost and confused :(

503 Unavailable — Asks everyone to come back later. A 404 alternative.

* Technically, code 301 is "Moved Permanently" and 302 is "Found", but SEOs refer to them as "Permanent Redirect" and "Temporary Redirect".

THE CANONICAL TAG

REL Canonical — Alternative to 301-redirects. Visitors still see Page A.

Old school

```
var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {
 if (xmlhttp.readystate == 4 && xmlhttp.status == 200) {
   var response = xmlhttp.responseText;
 } else if (xmlhttp.status == 400) { // or really anything in the 4 series
   // Handle error gracefully
 }
};

// Setup connection
xmlhttp.open(“GET”, “http://myserver/api”, true);

// Make the request
xmlhttp.send();
```

Communicating over $http

```
$http.get('api/user', {params: {id: '5'}
}).success(function(data, status, headers, config) {
 // Do something successful.
}).error(function(data, status, headers, config) {
 // Handle the error
});
```

```
var postData = {text: 'long blob of text'};
  // The next line gets appended to the URL as params
  // so it would become a post request to /api/user?id=5
  var config = {params: {id: '5'}};
  $http.post('api/user', postData, config
).success(function(data, status, headers, config) {
  // Do something successful
}).error(function(data, status, headers, config) {
  // Handle the error
});
```

More info?

https://angularjs.org

ANGULARJS | Home | Learn | Develop | Discuss | Search

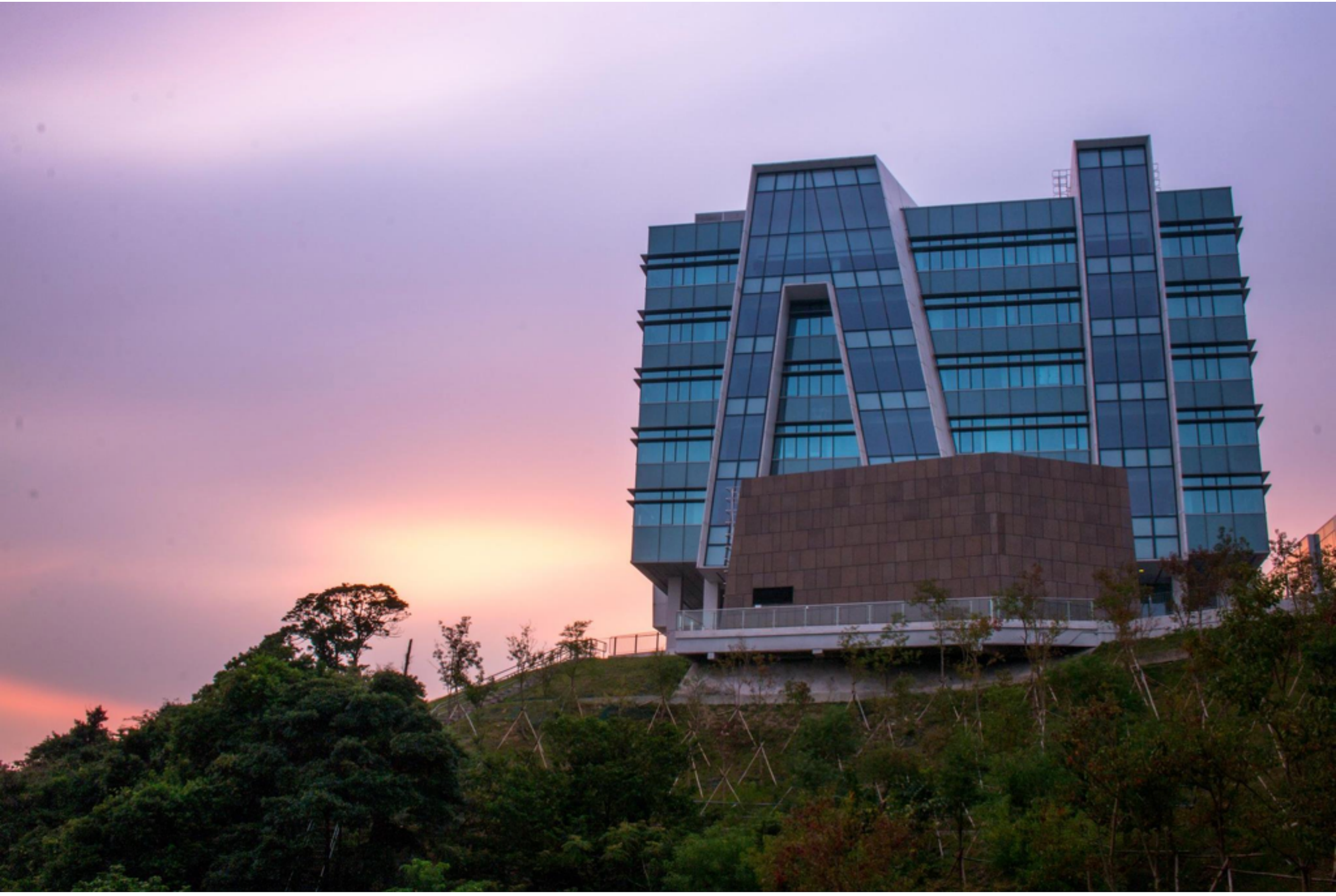
ANGULARJS by Google

HTML enhanced for web apps!

View on GitHub | Download (1.4.5 / 1.2.28) | Design Docs & Notes

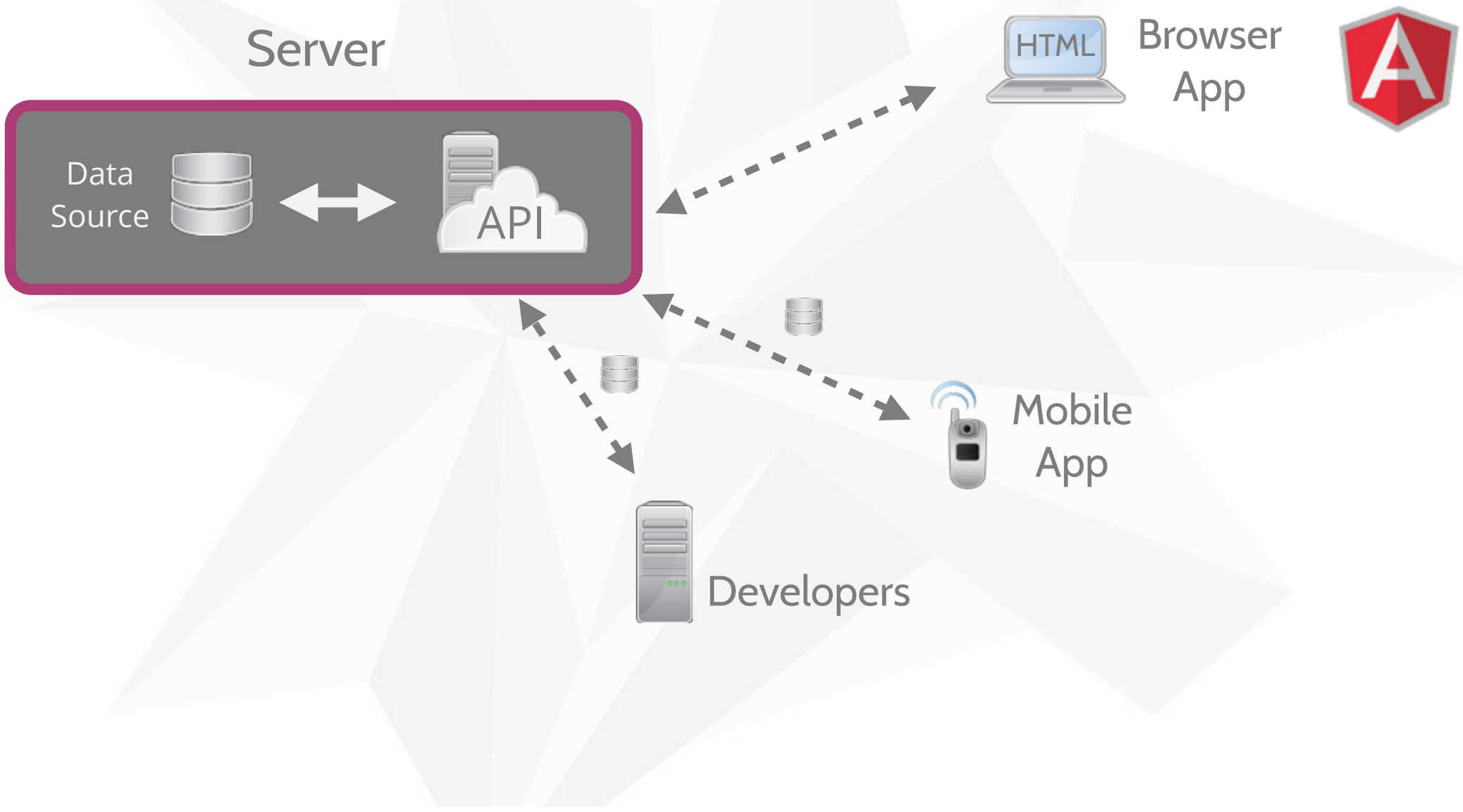
Follow +AngularJS on | Follow @angularjs 79.3K followers | Tweet 4,808

Learn Angular in your browser for free!




Targets
HKUST LIBRARY
WATERLOO

Modern API-Driven Application



https://www.codeschool.com/courses/shaping-up-with-angular-js

Firebase

- Lightning-fast data synchronization
- No backend server
- Built-in authentication
- Free hosting
- Magical data bindings

https://www.firebase.com/docs/web/libraries/angular/guide/intro-to-angularfire.html

REALTIME DATABASE

Store & sync data with our NoSQL cloud database. Data is stored as JSON, synced to all connected clients in realtime, and available when your app goes offline.

VERSION 1.1.2 – CHANGELOG

AngularFire

Download v1.1.2

Contribute on GitHub

Quickstart

5 min introduction

Guide

Step by Step Guide

API

Full List of API Methods

EXAMPLES

Code examples that showcase AngularFire.

- TodoMVC
- Tic-Tac-Tic-Tac-Toe
- Firereader
- Firepoker

RECIPES

Bite size code snippets that help speed up development.

- Date Object To A Firebase Timestamp Using $Extend
- Filter A $FirebaseArray

https://www.firebase.com/docs/web/libraries/angular/

1

Create an account

The first thing we need to do is sign up for a free Firebase account. A brand new Firebase app will automatically be created with its own unique URL ending in `firebaseio.com`. We'll use this URL to authenticate users and store and sync data with AngularFire.

https://www.firebase.com/docs/web/libraries/angular/quickstart.html

2 Add Script Dependencies

In order to use AngularFire in a project, include the following script tags:

```
<!-- AngularJS -->
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.15/angular.min.js"></script>

<!-- Firebase -->
<script src="https://cdn.firebase.com/js/client/2.2.4/firebase.js"></script>

<!-- AngularFire -->
<script src="https://cdn.firebase.com/libs/angularfire/1.1.3/angularfire.min.js"></script>
```

NPM, BOWER, AND YEOMAN SUPPORT

Firebase and AngularFire are available via npm and Bower as `firebase` and `angularfire`, respectively. A Yeoman generator is also available.

https://www.firebase.com/docs/web/libraries/angular/quickstart.html

4

Add Three-Way, Object Bindings

Angular is known for its two-way data binding between JavaScript models and the DOM, and Firebase has a lightning-fast, realtime database. For synchronizing simple key / value pairs, AngularFire can be used to *glue* the two together, creating a "three-way data binding" which automatically synchronizes any changes to your DOM, your JavaScript, and the Firebase database.

To set up this three-way data binding, we use the `$firebaseObject` service introduced above to create a synchronized object, and then call `$bindTo()`, which binds it to a `$scope` variable.

APP.JS | INDEX.HTML

```
1.  var app = angular.module("sampleApp", ["firebase"]);
2.
3.  app.controller("SampleCtrl", function($scope, $firebaseObject) {
4.    var ref = new Firebase("https://<YOUR-FIREBASE-APP>.firebaseio.com/data");
5.
6.    // download the data into a local object
7.    var syncObject = $firebaseObject(ref);
8.
9.    // synchronize the object with a three-way data binding
10.   // click on `index.html` above to see it used in the DOM!
11.   syncObject.$bindTo($scope, "data");
12. });
```

https://www.firebase.com/docs/web/libraries/angular/quickstart.html

5

Synchronize Collections as Arrays

Three-way data bindings are amazing for simple key / value data. However, there are many times when an array would be more practical, such as when managing a collection of messages. This is done using the `$firebaseArray` service.

We synchronize a list of messages into a **read-only** array by using the `$firebaseArray` service and then assigning the array to `$scope`:

APP.JS | INDEX.HTML

Copy

```
1. var app = angular.module("sampleApp", ["firebase"]);
2.
3. app.controller("SampleCtrl", function($scope, $firebaseArray) {
4.   var ref = new Firebase("https://<YOUR-FIREBASE-APP>.firebaseio.com/messages");
5.
6.   // create a synchronized array
7.   // click on `index.html` above to see it used in the DOM!
8.   $scope.messages = $firebaseArray(ref);
9. });
```

https://www.firebase.com/docs/web/libraries/angular/quickstart.html

5

Synchronize Collections as Arrays

Because the array is synchronized with server data and being modified concurrently by the client, it is possible to lose track of the fluid array indices and corrupt the data by manipulating the wrong records. Therefore, the placement of items in the list should never be modified directly by using array methods like `push()` or `splice()`.

Instead, AngularFire provides a set of methods compatible with manipulating synchronized arrays: `$add()`, `$save()`, and `$remove()`.

APP.JS | INDEX.HTML

Copy

```
var app = angular.module("sampleApp", ["firebase"]);

app.controller("SampleCtrl", function($scope, $firebaseArray) {
  var ref = new Firebase("https://<YOUR-FIREBASE-APP>.firebaseio.com/messages");

  // create a synchronized array
  $scope.messages = $firebaseArray(ref);

  // add new items to the array
  // the message is automatically added to our Firebase database!
  $scope.addMessage = function() {
    $scope.messages.$add({
      text: $scope.newMessageText
    });
  };

  // click on `index.html` above to see $remove() and $save() in action
});
```

https://www.firebase.com/docs/web/libraries/angular/quickstart.html

6

Add Authentication

Firebase provides a hosted authentication service which offers a completely client-side solution to account management and authentication. It supports anonymous authentication, email / password login, and login via several OAuth providers, including Facebook, GitHub, Google, and Twitter.

AngularFire provides a service named `$firebaseAuth` which wraps the authentication methods provided by the Firebase client library. It can be injected into any controller, service, or factory.

```
app.controller("SampleCtrl", function($scope, $firebaseAuth) {
  var ref = new Firebase("https://<YOUR-FIREBASE-APP>.firebaseio.com");

  // create an instance of the authentication service
  var auth = $firebaseAuth(ref);

  // login with Facebook
  auth.$authWithOAuthPopup("facebook").then(function(authData) {
    console.log("Logged in as:", authData.uid);
  }).catch(function(error) {
    console.log("Authentication failed:", error);
  });
});
```

https://www.firebase.com/docs/web/libraries/angular/quickstart.html

```
// define our app and dependencies (remember to include firebase!)
var app = angular.module("sampleApp", ["firebase"]);

// this factory returns a synchronized array of chat messages
app.factory("chatMessages", ["$firebaseArray",
  function($firebaseArray) {
    // create a reference to the database location where we will store our data
    var randomRoomId = Math.round(Math.random() * 100000000);
    var ref = new Firebase("https://docs-sandbox.firebaseio.com/af/intro/demo/" + randomRoomId);

    // this uses AngularFire to create the synchronized array
    return $firebaseArray(ref);
  }
]);

app.controller("ChatCtrl", ["$scope", "chatMessages",
  // we pass our new chatMessages factory into the controller
  function($scope, chatMessages) {
    $scope.user = "Guest " + Math.round(Math.random() * 100);

    // we add chatMessages array to the scope to be used in our ng-repeat
    $scope.messages = chatMessages;

    // a method to create new messages; called by ng-submit
    $scope.addMessage = function() {
      // calling $add on a synchronized array is like Array.push(),
      // except that it saves the changes to our database!
      $scope.messages.$add({
        from: $scope.user,
        content: $scope.message
      });

      // reset the message input
      $scope.message = "";
    };

    // if the messages are empty, add something for fun!
    $scope.messages.$loaded(function() {
      if ($scope.messages.length === 0) {
        $scope.messages.$add({
          from: "Firebase Docs",
          content: "Hello world!"
        });
      }
    });
  }
]);
```

Web chatting with Firebase

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.15/angular.min.js"></script>
<script src="https://cdn.firebase.com/js/client/2.2.4/firebase.js"></script>
<script src="https://cdn.firebase.com/libs/angularfire/1.1.2/angularfire.min.js"></script>

<div ng-app="sampleApp" ng-controller="ChatCtrl">
  <ul class="chatbox">
    <li ng-repeat="message in messages">{{ message.from }}: {{ message.content }}</li>
  </ul>
  <form ng-submit="addMessage()">
    <input type="text" ng-model="message">
    <input type="submit" value="Add Message">
  </form>
</div>
```

- Firebase Docs: Hello world!
- Guest 100: Hello COMP3111!

Add Message

Bookmark demo

Running local server

- Install node and npm
  - Windos: http://blog.teamtreehouse.com/install-node-js-npm-windows
  - Mac: http://blog.teamtreehouse.com/install-node-js-npm-mac
  - Linux: http://blog.teamtreehouse.com/install-node-js-npm-linux
- npm start (with this package.json)

```
{
  "private": true,

"scripts": {
    "prestart": "npm install",
    "start": "http-server -a 0.0.0.0 -p 8000"
  }
}

```

Facebook web app setting

Facebook Authentication

Configuring Your Application

To get started with Facebook authentication, you need to first create a new Facebook application. Click the **Add a New App** button in the top right of that page and select **Website** as your platform. Then choose an App ID and click **Create New Facebook App ID**. Select your app's category and click **Create App ID**. In the top right, click **Skip Quick Start**.

In your Facebook app configuration, click on the **Settings** tab on the left-hand navigation menu. Then go to the **Advanced** tab at the top and scroll down to the **Security** section. At the bottom of that section, add `https://auth.firebase.com/v2/<YOUR-FIREBASE-APP>/auth/facebook/callback` to your **Valid OAuth redirect URIs** and click **Save Changes** at the bottom of the page.

Next, you'll need to get your app credentials from Facebook. Click on the **Basic** tab at the top of the page. You should still be within the **Settings** tab. Towards the top of this page, you will see your **App ID** and **App Secret**. Your **App ID** will be displayed in plain text and you can view your **App Secret** by clicking on the **Show** button and typing in your Facebook password. Copy these Facebook application credentials (**App ID** and **Secret**) in the **Login & Auth** section in your App Dashboard.

https://www.firebase.com/docs/web/guide/login/facebook.html

URL

https://developers.facebook.com/apps

facebook for developers

Products Docs Tools & Support News

My Apps

Add

Search apps by title

+ Add a New App

carmel.cse.ust.hk.
App ID: 591819294280029

Class Questions
App ID: 821159334657979

Sitelook
App ID: 1058367534184879

Developers

Add a New App

Select a platform to get started

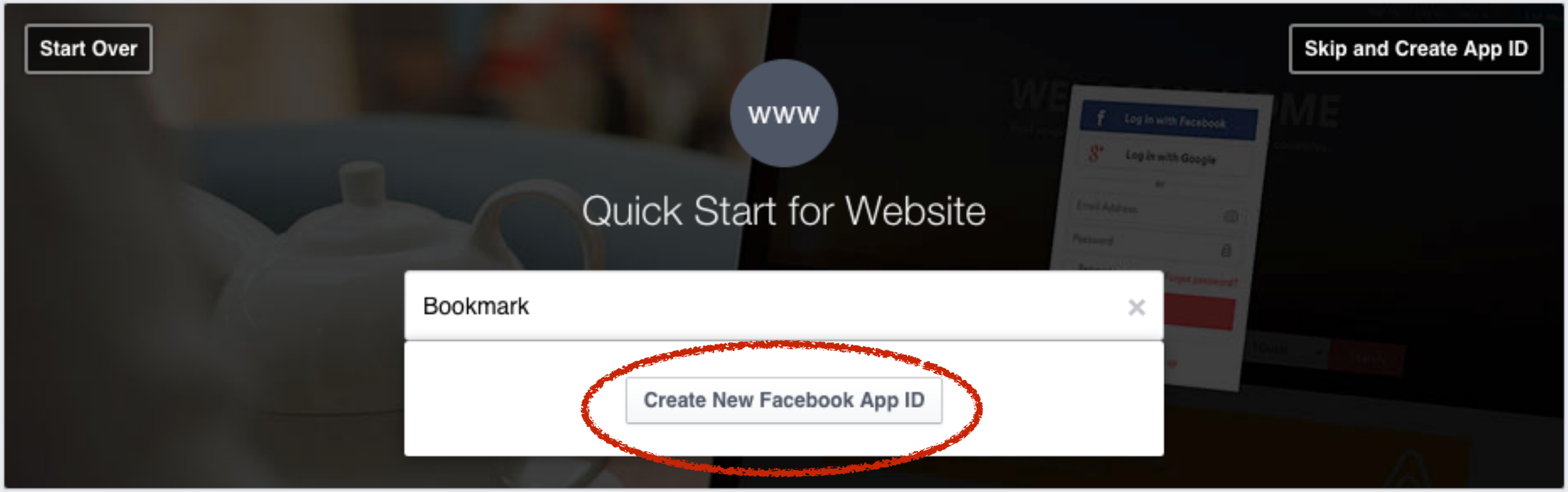
iOS

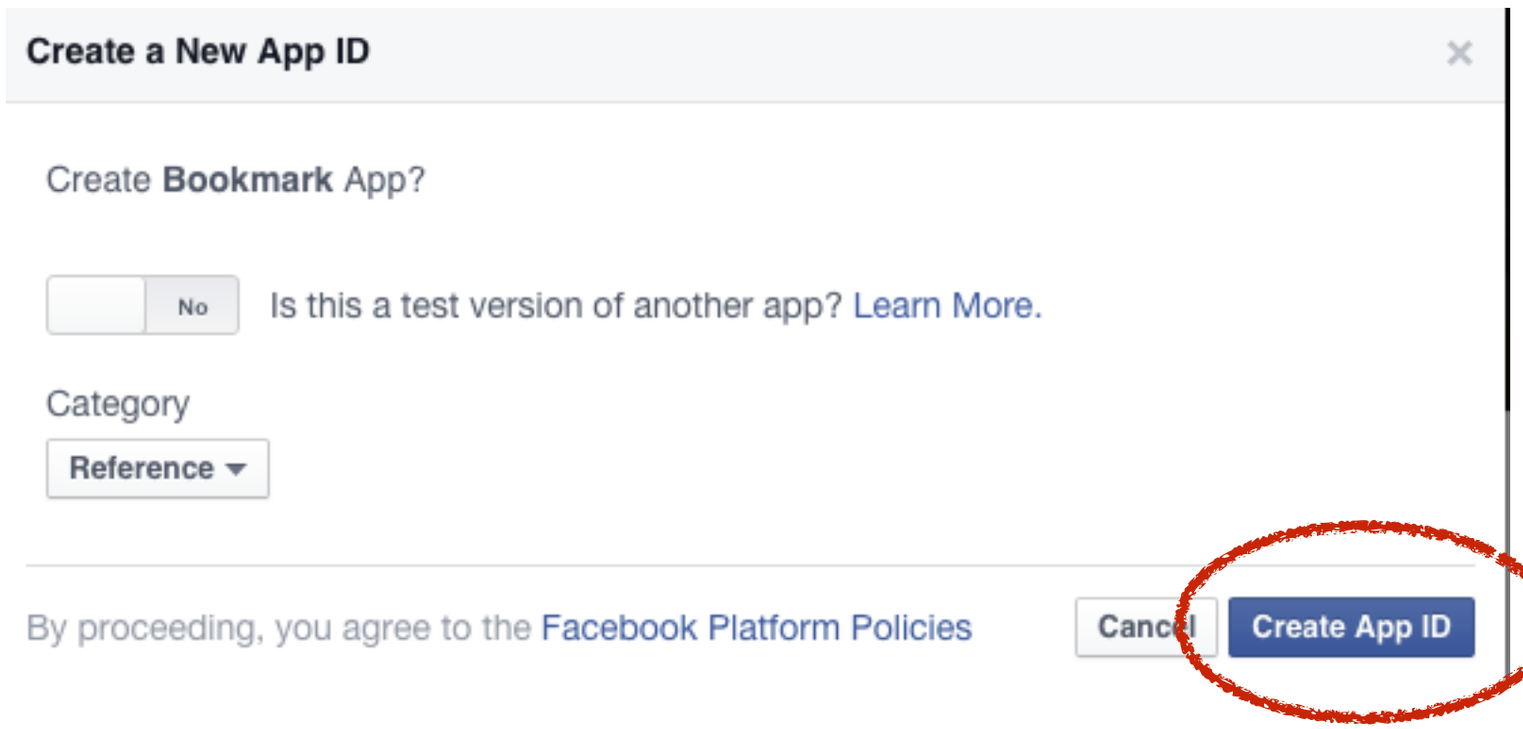
Android

Facebook Canvas

Website

If you're developing on another platform or would like to setup Audience Network please use the advanced setup.


Start Over
Skip and Create App ID
www
Quick Start for Website
Bookmark
Create New Facebook App ID


Create a New App ID
Create Bookmark App?
No
Is this a test version of another app? Learn More.
Category
Reference
By proceeding, you agree to the Facebook Platform Policies
Cancel
Create App ID

Setup SDK — App Configuration — Test — Finished

Next Steps

Congratulations! You have added the Facebook SDK to your project. You are now in the next stage in integrating your app with Facebook. What do you want to do next? Skip to Developer Dashboard or Documentation

Share
Add the share dialog to let people share your content with their friends.

Login
Add Facebook Login to let people quickly and easily login to your app.

Social Plugins
See what your Facebook friends have liked, shared, or commented on across the Web.

Ads
Grow your app with Mobile App Ads.

Dashboard

Bookmark

This app is in development mode and can only be used by app admins, developers and testers [?]

App ID: 169497970081387

API Version [?]: v2.5

App Secret: •••••••• Show

Get Started with the Facebook SDK

Use our quick start guides to set up the Facebook SDK for your iOS or Android app, Canvas game or website.

Choose a Platform

Secure Your App Settings

Use our app security checkup tool to see how secure your app is and identify potential vulnerabilities.

Try It Now

Facebook Analytics for Apps

Get Analytics and Trends

Use Facebook Analytics for Apps to understand how people are using your iOS, Android or Canvas app.

Try It Now

Facebook Login

Bookmark
Dashboard
Settings
Status & Review
App Details
Roles
Open Graph
Alerts
Localize
Canvas Payments
Audience Network
Test Apps
Webhooks
Analytics

Basic | Advanced | Migrations

App ID
169497970081387

App Secret
••••••••
Show

Display Name
Bookmark

Namespace

App Domains

Contact Email
Used for important communication about your app

Website
Quick Start

Site URL
http://www.cse.ust.hk/

+ Add Platform

Delete App

Discard
Save Changes

Yes

Client OAuth Login
Enables the standard OAuth client token flow. Secure your application and prevent abuse by locking down which token redirect URIs are allowed with the options below. Disable globally if not used. [?]

Yes

Web OAuth Login
Enables web based OAuth client login for building custom login flows. [?]

No

Force Web OAuth Reauthentication
When on, prompts people to enter their Facebook password in order to log in on the web. [?]

No

Embedded Browser OAuth Login
Enables browser control redirect uri for OAuth client login. [?]

Valid OAuth redirect URIs

https://auth.firebase.com/v2/sbookmark/auth/facebook/callback ×

No

Login from Devices
Enables the OAuth client login flow for devices like a smart TV [?]

Business Manager

Associate your app with a business to use the Business Mapping API and access direct developer support.

Get Started

Advertising Accounts

Ads API

Authorized Ad Account IDs

Analytics for Apps

Yes

Collect the Apple Advertising Identifier (IDFA) with App Events
This lets you collect the IDFA through App Events. Developers must ensure they are in compliance with the iOS developer program license agreement policies governing the use of this identifier. Note: if you are running or have run Facebook mobile app ads in the previous 28 days, the IDFA will automatically be collected through the Facebook SDK, even if this setting is set to 'no'.

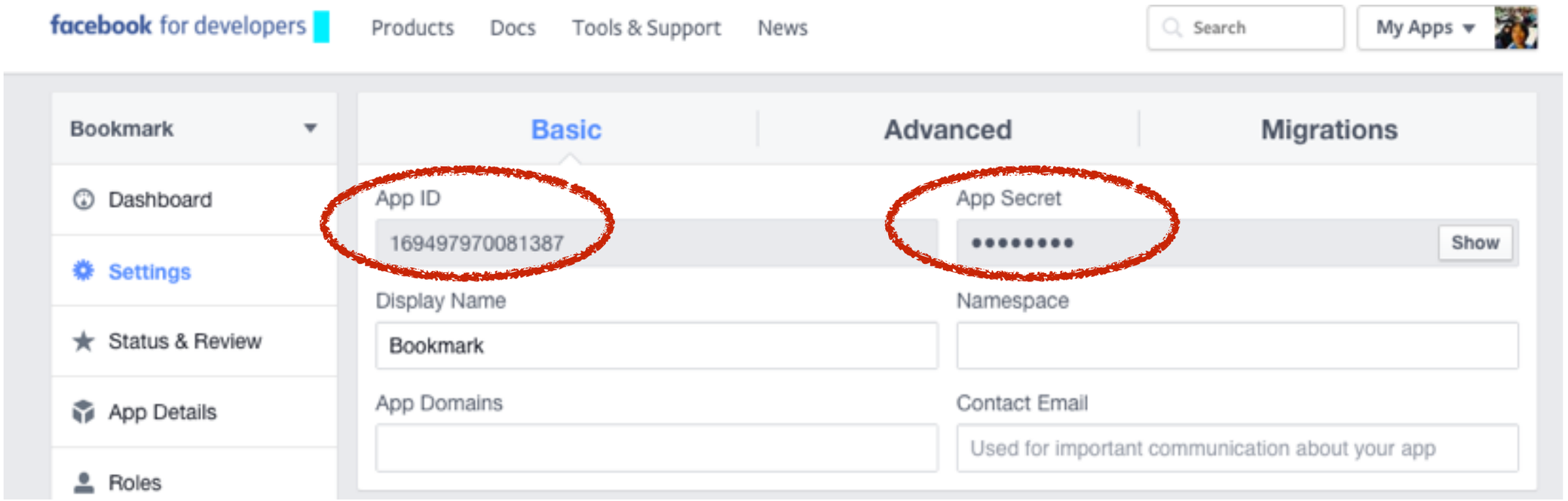
No

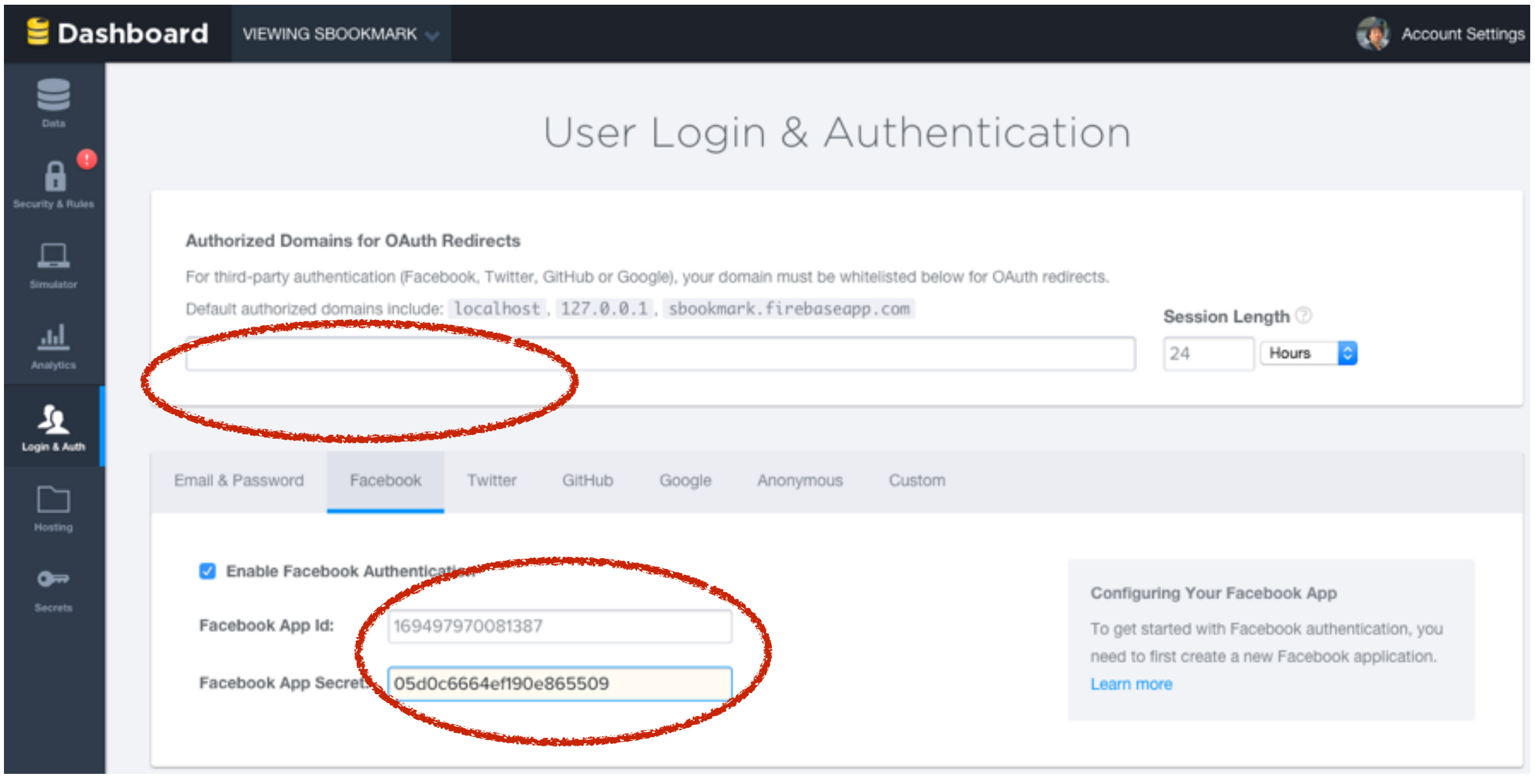
Allow All Analytics Users to See Sensitive Info
Turning this on allows anyone with access to this Analytics account to see all data for this app, including sensitive info like purchases, revenue and more.

Delete App

Discard

Save Changes


facebook for developers
Products
Docs
Tools & Support
News
Search
My Apps
Bookmark
Dashboard
Settings
Status & Review
App Details
Roles
Basic
Advanced
Migrations
App ID
169497970081387
App Secret
••••••••
Show
Display Name
Bookmark
Namespace
App Domains
Contact Email
Used for important communication about your app

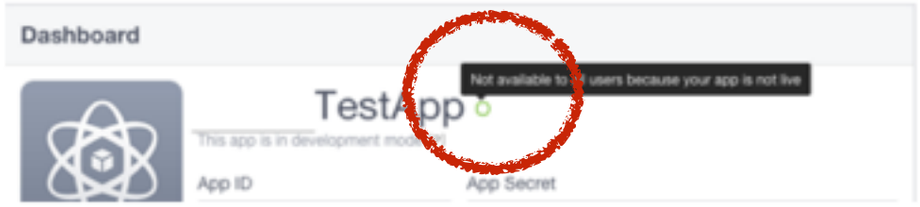

Dashboard
VIEWING SBOOKMARK
Account Settings
Data
Security & Rules
Simulator
Analytics
Login & Auth
Hosting
Secrets
User Login & Authentication
Authorized Domains for OAuth Redirects
For third-party authentication (Facebook, Twitter, GitHub or Google), your domain must be whitelisted below for OAuth redirects.
Default authorized domains include: localhost , 127.0.0.1 , sbookmark.firebaseapp.com
Session Length
24
Hours
Email & Password
Facebook
Twitter
GitHub
Google
Anonymous
Custom
Enable Facebook Authentication
Facebook App Id:
169497970081387
Facebook App Secret:
05d0c6664ef190e865509
Configuring Your Facebook App
To get started with Facebook authentication, you need to first create a new Facebook application.
Learn more

397

1. Go to https://developers.facebook.com/

2. Click on the Apps menu on the top bar.

3. Select the respective app from the drop down.



The circle next to your app name is not fully green. When you hover mouse on it, you'll see a popup saying, "Not available to all users because your app is not live."

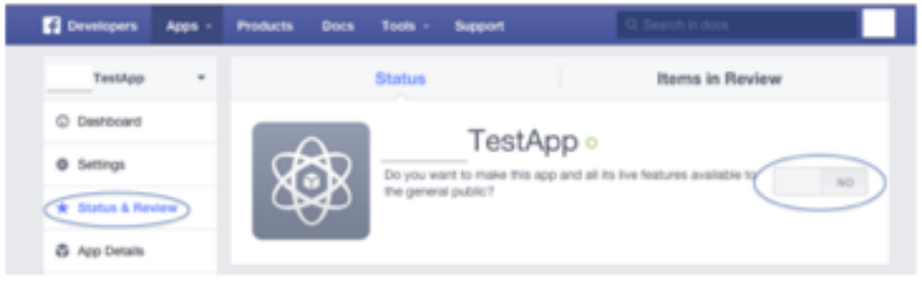
So next, you've to make it publicly available.

4. Click on setting at left panel. [see the screenshot below]

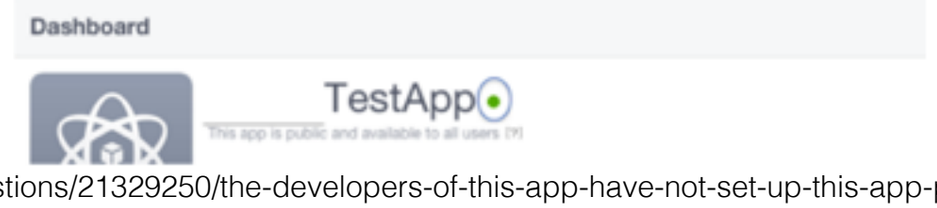
5. In Basic tab add your "Contact Email" (a valid email address - I've added the one which I'm using with developers.facebook.com) and make "Save changes".

6. Next click "Status & Review" at left panel. [see the screenshot below]

7. Look for this, **Do you want to make this app and all its live features available to the general public?** and Turn ON the switch next to this.



8. That's it! - App is now publicly available. See the fully green circle next to the app name.



SECURITY & RULES GUIDE

Securing Your Data

Firebase provides a flexible, expression-based rules language with JavaScript-like syntax to easily define how your data should be structured and when your data can be read from and written to. Combined with our login service which allows for easy authentication, you can define who has access to what data and keep all of your user's personal information secure. The rules live on the Firebase servers and are automatically enforced at all times.

https://www.firebase.com/docs/security/guide/

FIREBASE RULES

```
{
    "rules": {
        ".read": true,
        ".write": true
    }
}
```

FIREBASE RULES

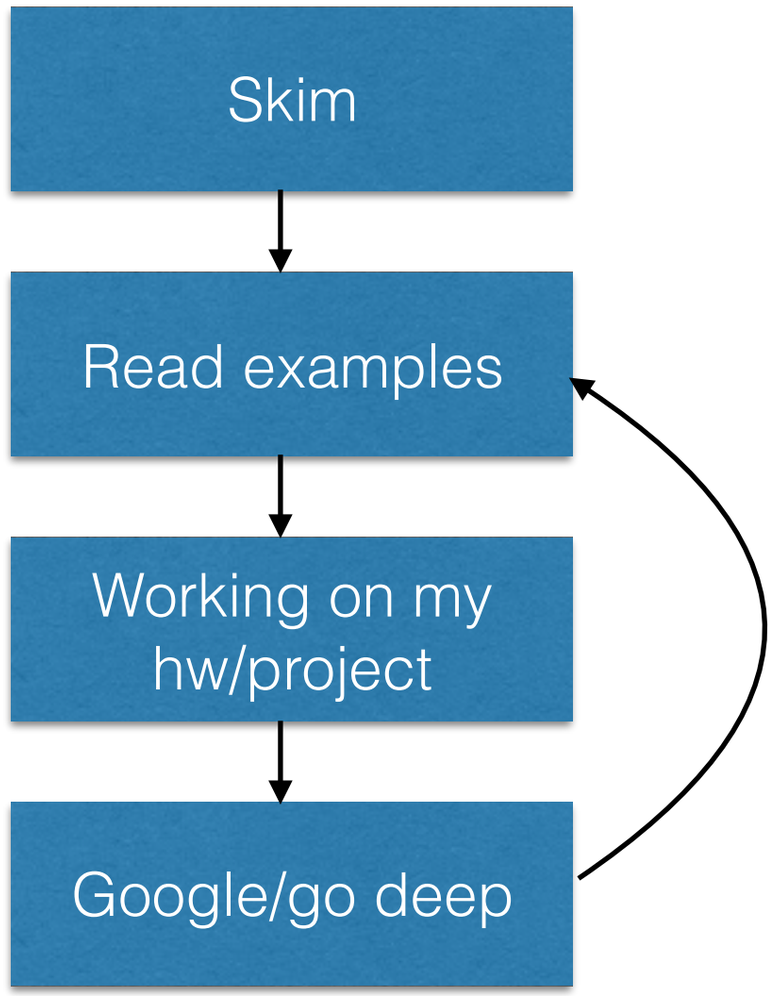
```
{
  "rules": {
    "@demo_": {
        ".read": true,
        ".write": "auth !== null && auth.uid === 'hunkim'"
       // ".write": true
    },

    "$uid": {
        // grants write access to the owner of this user account
       // whose uid must exactly match the key ($uid)
        ".write": "auth !== null && auth.uid === $uid",
        ".read": "auth !== null && auth.uid === $uid"
    }
  }
}
```

FIREBASE RULES

```
{
    "rules": {
        ".read": false,
        ".write": false,
        "sales": {
            "$fid": {
              ".write": false,
             // grants write access to the owner of this user account
             // whose uid must exactly match the key ($uid)
              ".read": "auth !== null && auth.uid === $fid"
            }
        },

        "posts": {
            "$fid": {
            // grants write access to the owner of this user account
            // whose uid must exactly match the key ($uid)
            ".write": "auth !== null && auth.uid === $fid",
            ".read": "auth !== null && auth.uid === $fid"
          }
        }
    }
}
```

More bookmark demo

Steps of learning new stuff



Lab

- Do 5 min AngularFire tutorial: https://www.firebase.com/tutorial/#tutorial/angular/0
- Fork https://github.com/hunkim/bookmark101
- Design bookmark html
- Try basic app.js (basic bookmark features + FB login)
- Validate URL (http://esmash.xyz VS aa:test//url)
- make/work on your directory in gitworkspace and send a pull request

HW

- Individual Bookmark: each FB user should see only their bookmark.
- Get Title of the URL automatically.
- Get homepage thumbnail for each URL
- Post it s3 or EC2 with a nice (free)domain name
- Create an issue with your URL at https://github.com/hunkim/bookmark101

Hello Jindae Kim! You have 2 Bookmarks.

Logout

Logout

www.naver.com

Add URL

Daum – 모으다 잇다 흔들다

www.daum.net

Delete Bookmark

NAVER

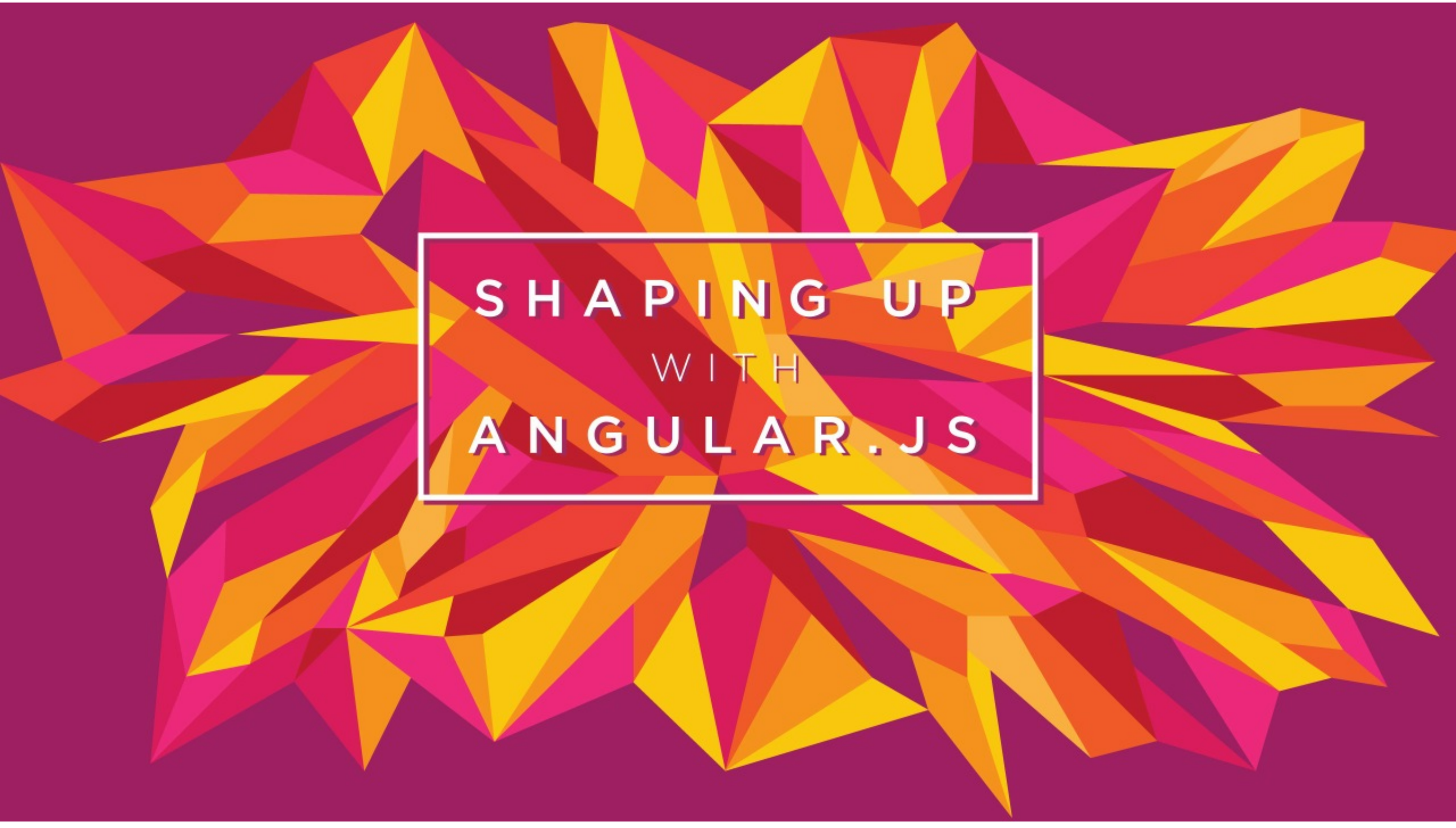
www.naver.com

Delete Bookmark

HW - Challenges (optional)

- Reuse created thumbnails/use S3
- Make it beautiful: hint bootstrap
- Also make it beautiful for mobile (m.*) with swipe/drag
- Sort url by title/url
- +a: make it really useful URL service

Appendix



https://www.codeschool.com/courses/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.

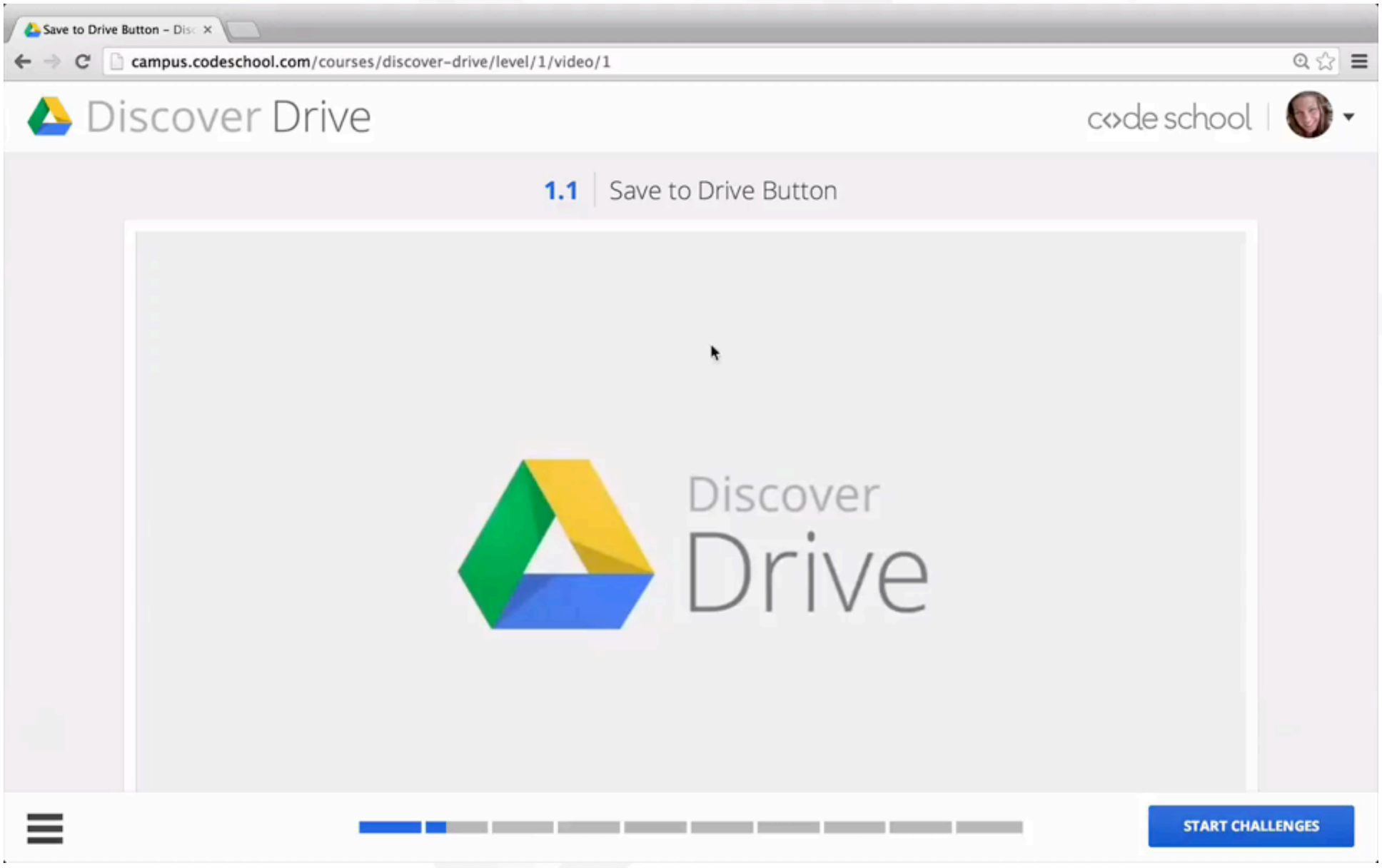
User clicks on link, new Request

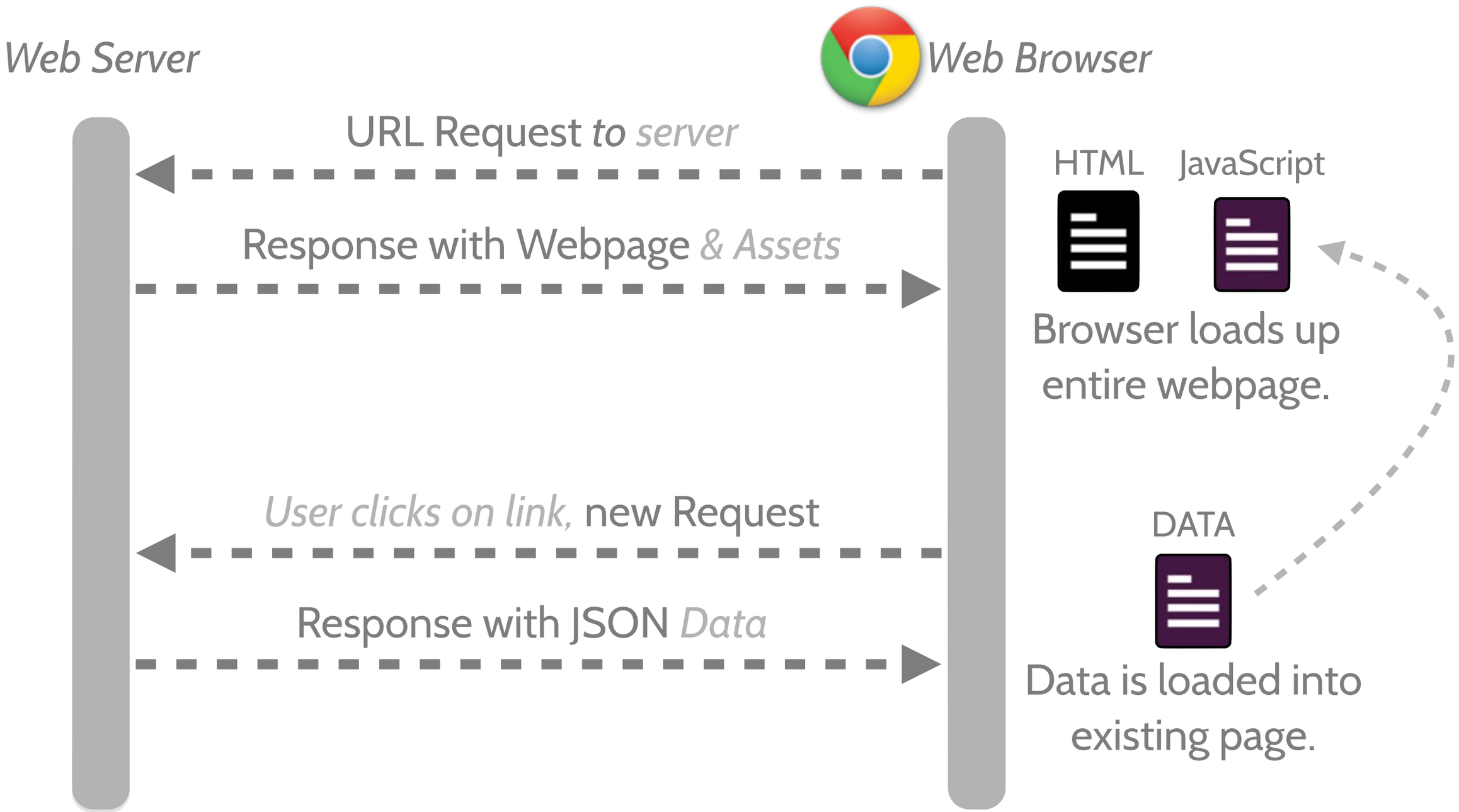
Response with Webpage & Assets

HTML JavaScript

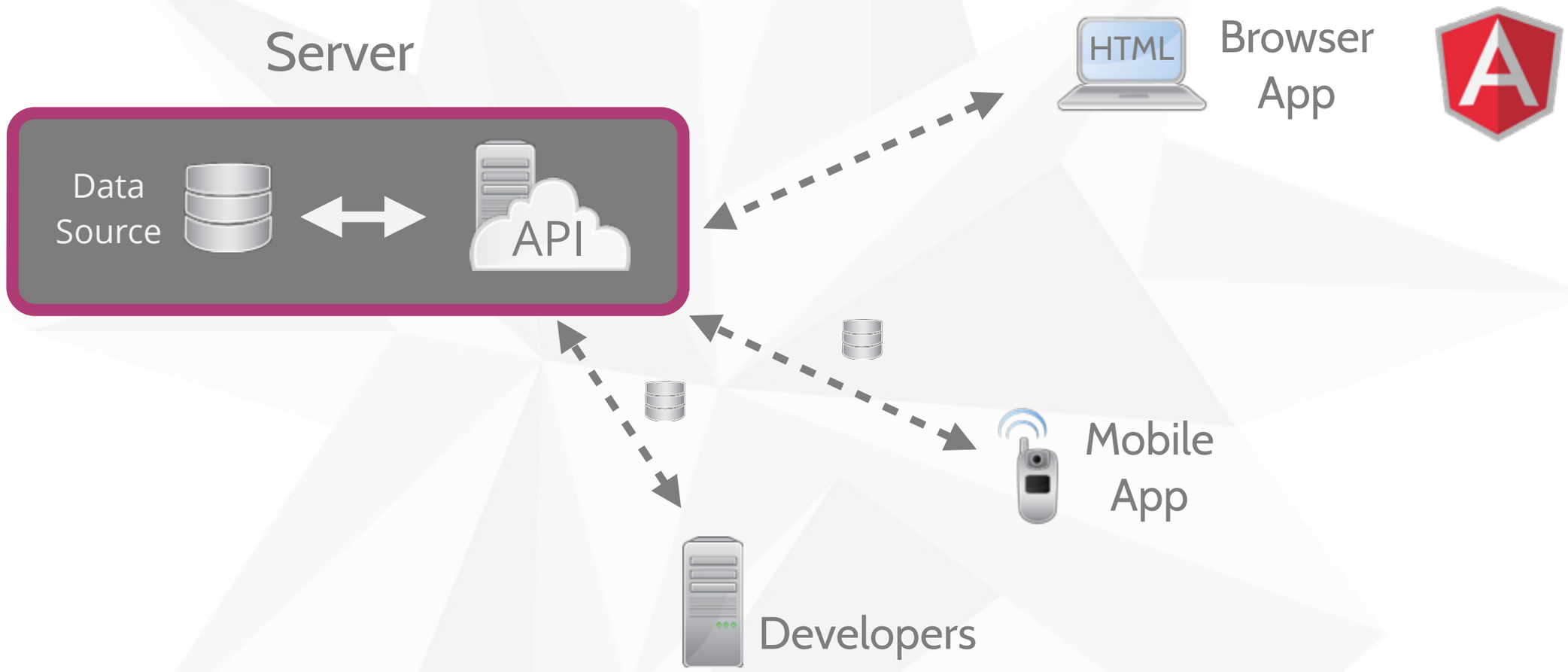
Browser loads up entire webpage.

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
Response with JSON Data
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**.

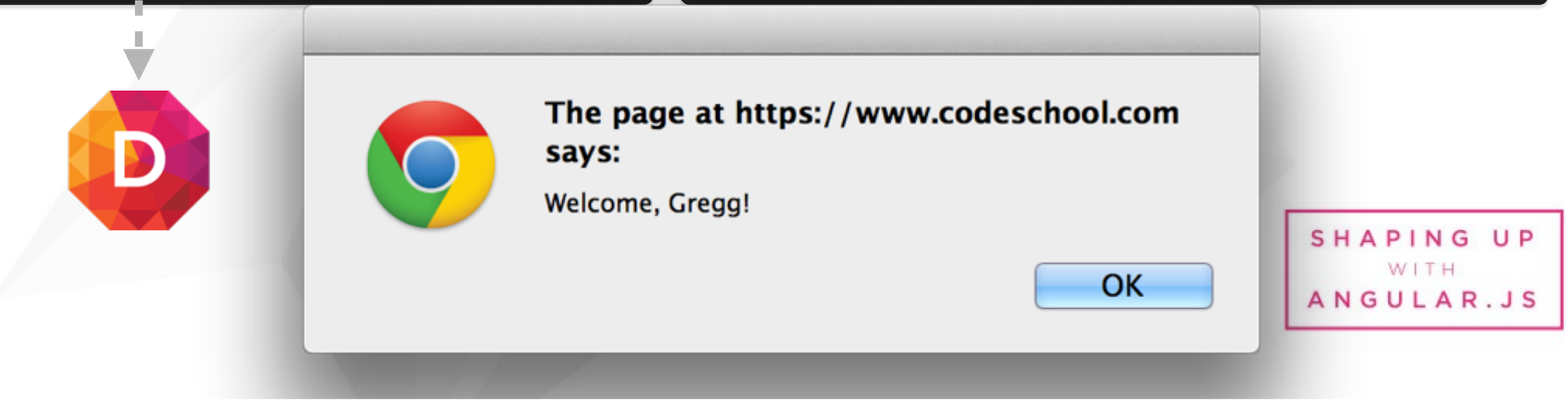
```
<!DOCTYPE html>
<html>
  <body ng-controller="StoreController">
  . . .
  </body>
</html>
```

index.html

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

Name of function to call

app.js



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**

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

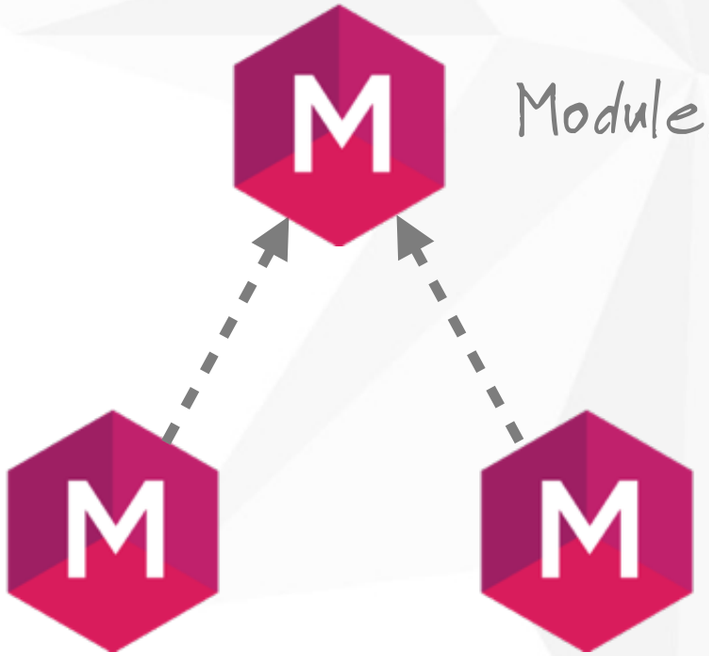
AngularJS

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.



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

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

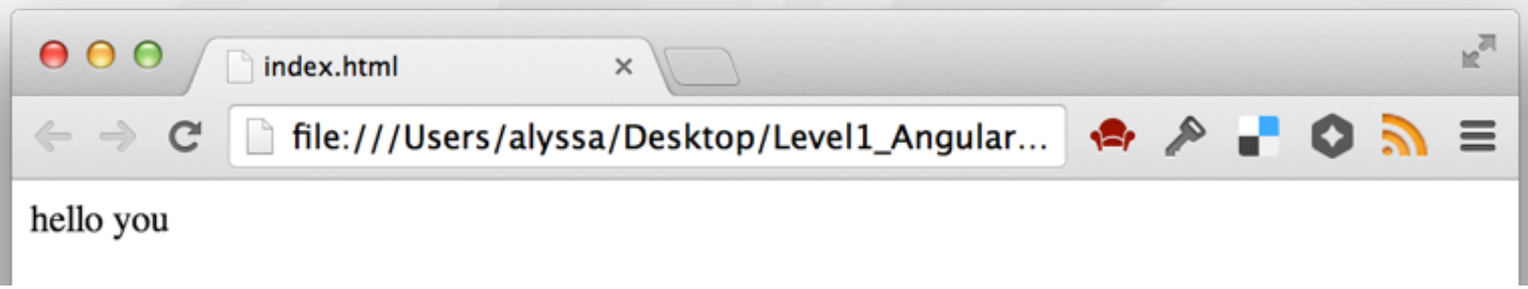
Including Our Module

```
<!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

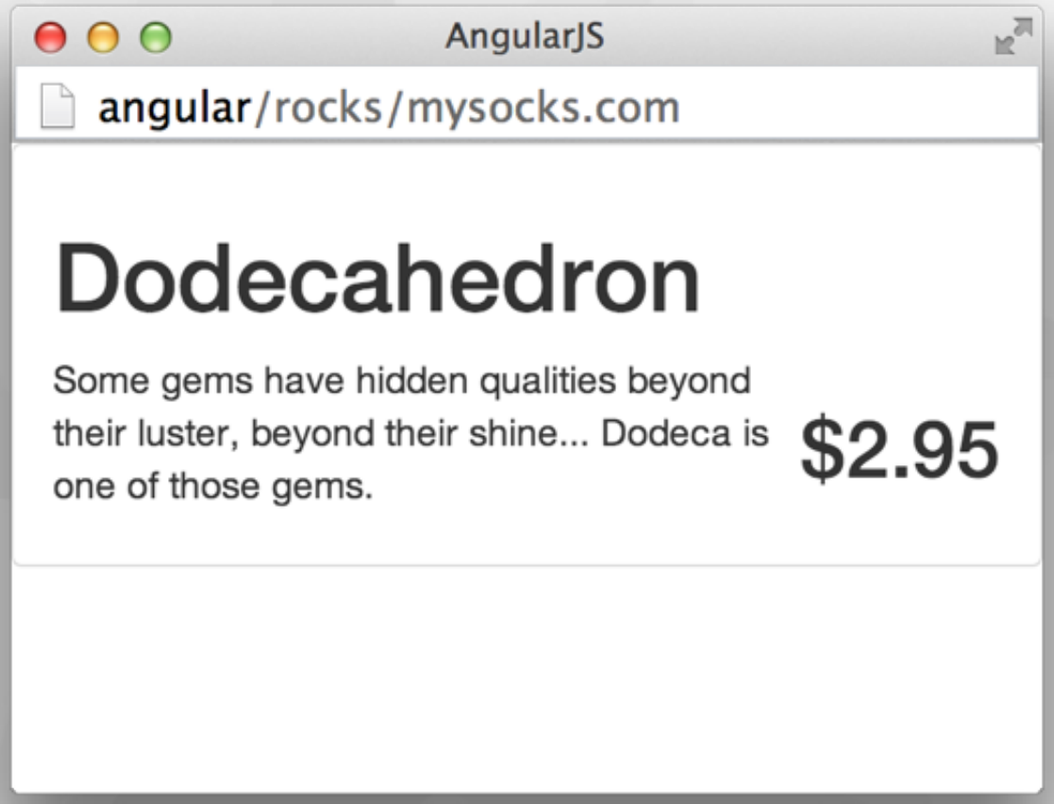


SHAPING UP WITH ANGULAR.JS

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.

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?

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

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

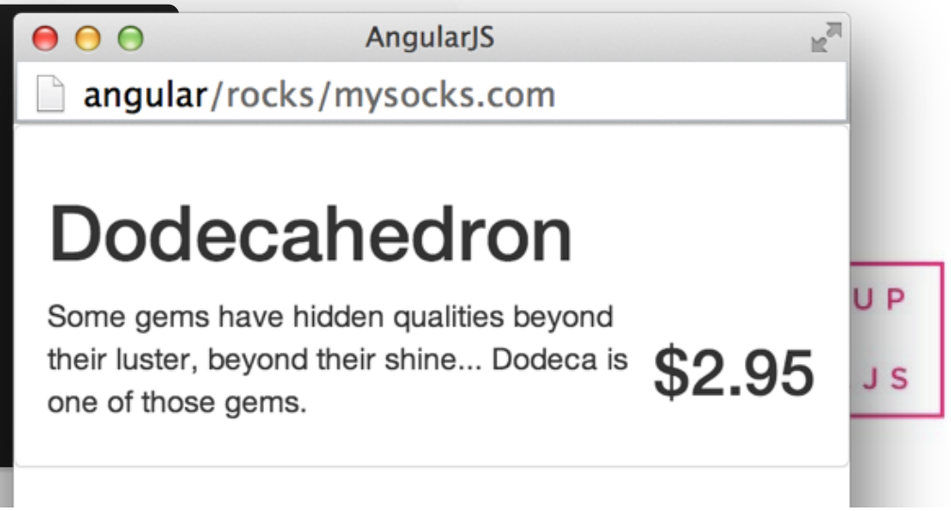
SHAPING UP WITH ANGULAR.JS

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

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/angu
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.

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;
});
```

So we have multiple products...

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

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

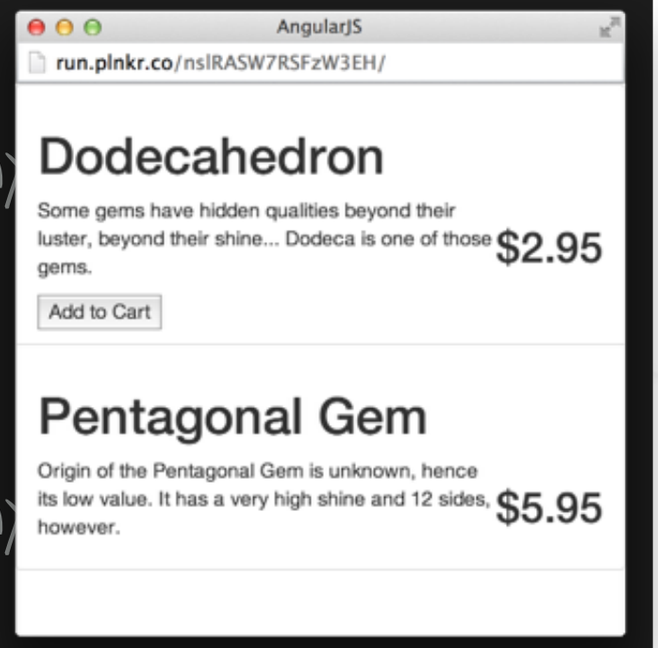
```
<body ng-controller="StoreController as store">
  <div>
    <h1> {{store.products[0].name}} </h1>
    <h2> ${{store.products[0].price}} </h2>
    <p> {{store.products[0].description}} </p>
    <button ng-show="store.products[0].canPurchase">
      Add to Cart</button>
  </div>
  <div>
    <h1> {{store.products[1].name}}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

That works...

Why

Why

Why... You get it.



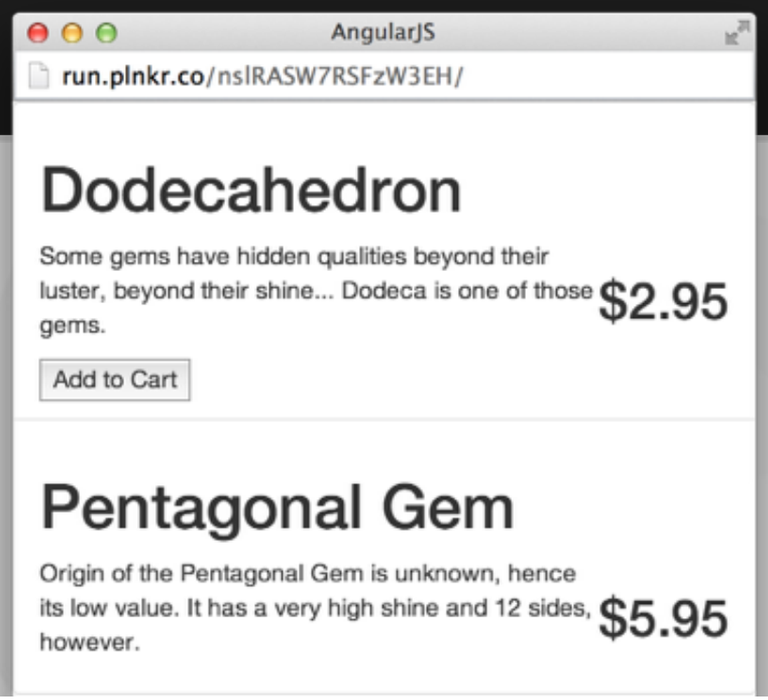
index.html

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

Shaping Up with Angular.JS

Level 2: Filters, Directives, and Cleaner Code

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>
```

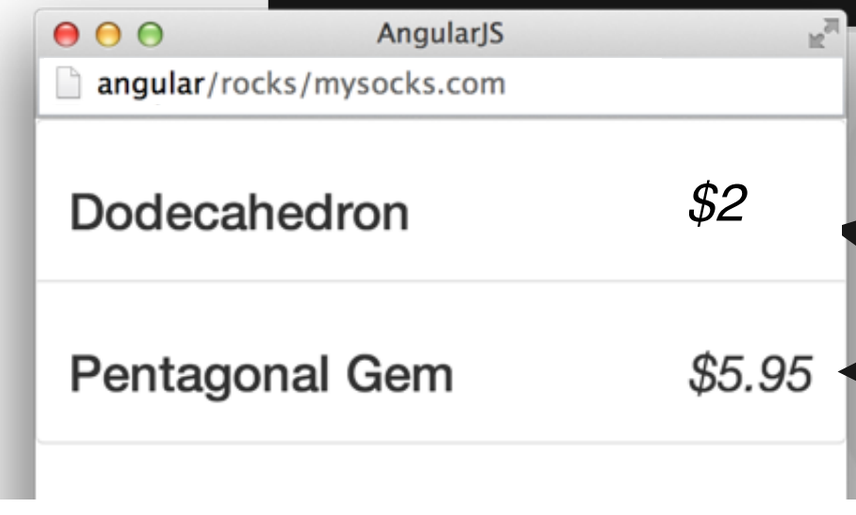
ng-repeat – repeat a section for each item in an Array

```
<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

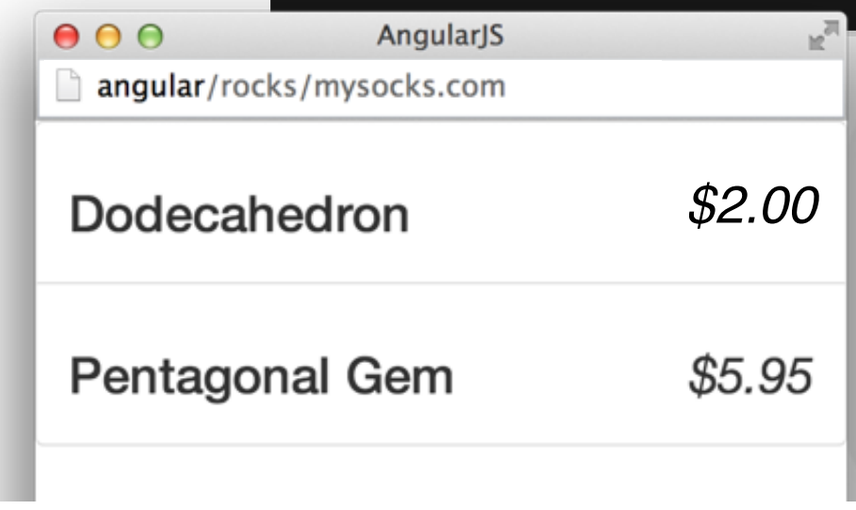


There's a better way to print out prices.

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}}"/>
```

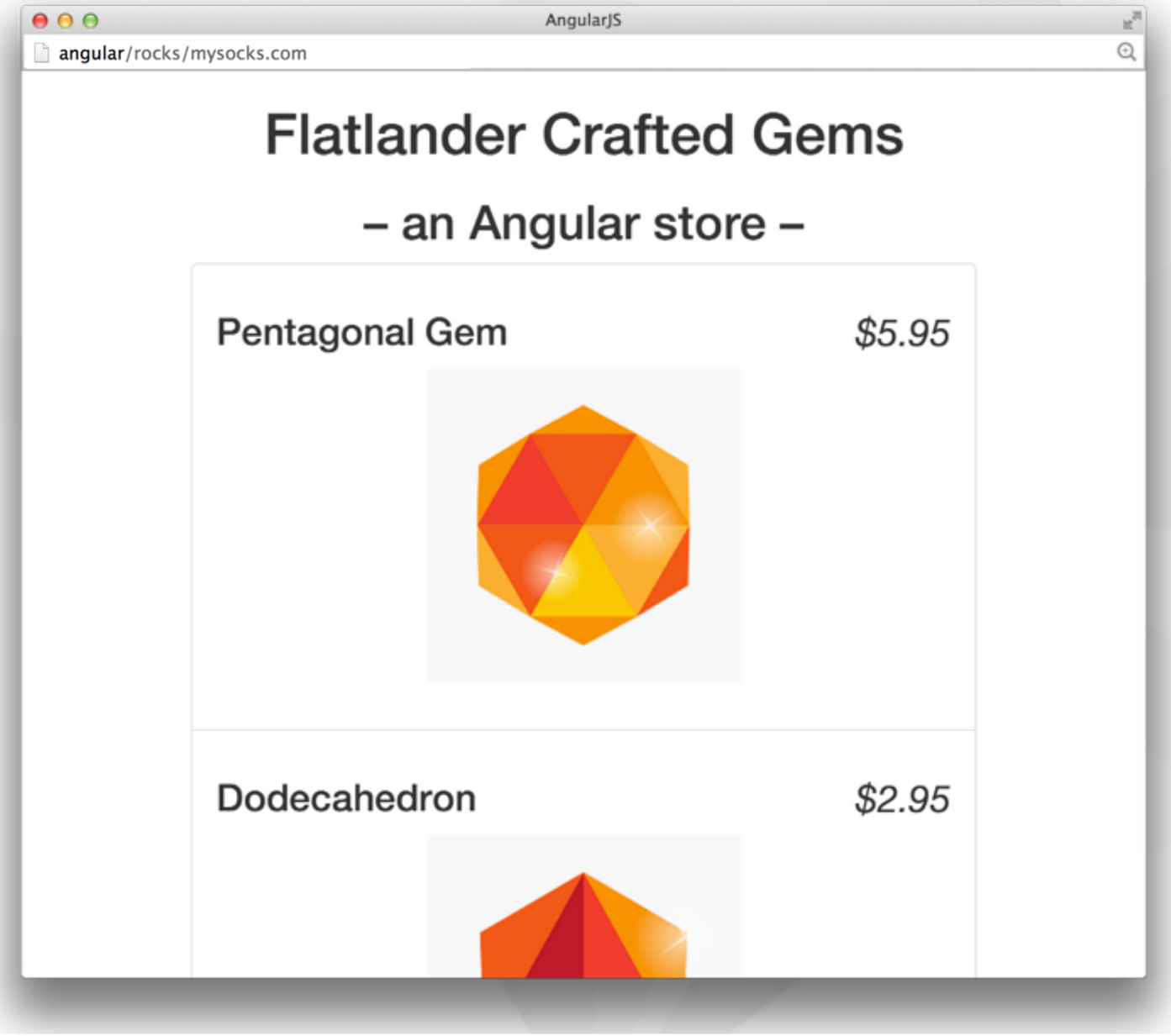
...the browser tries to load the image *before* the Expression evaluates.

```
<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

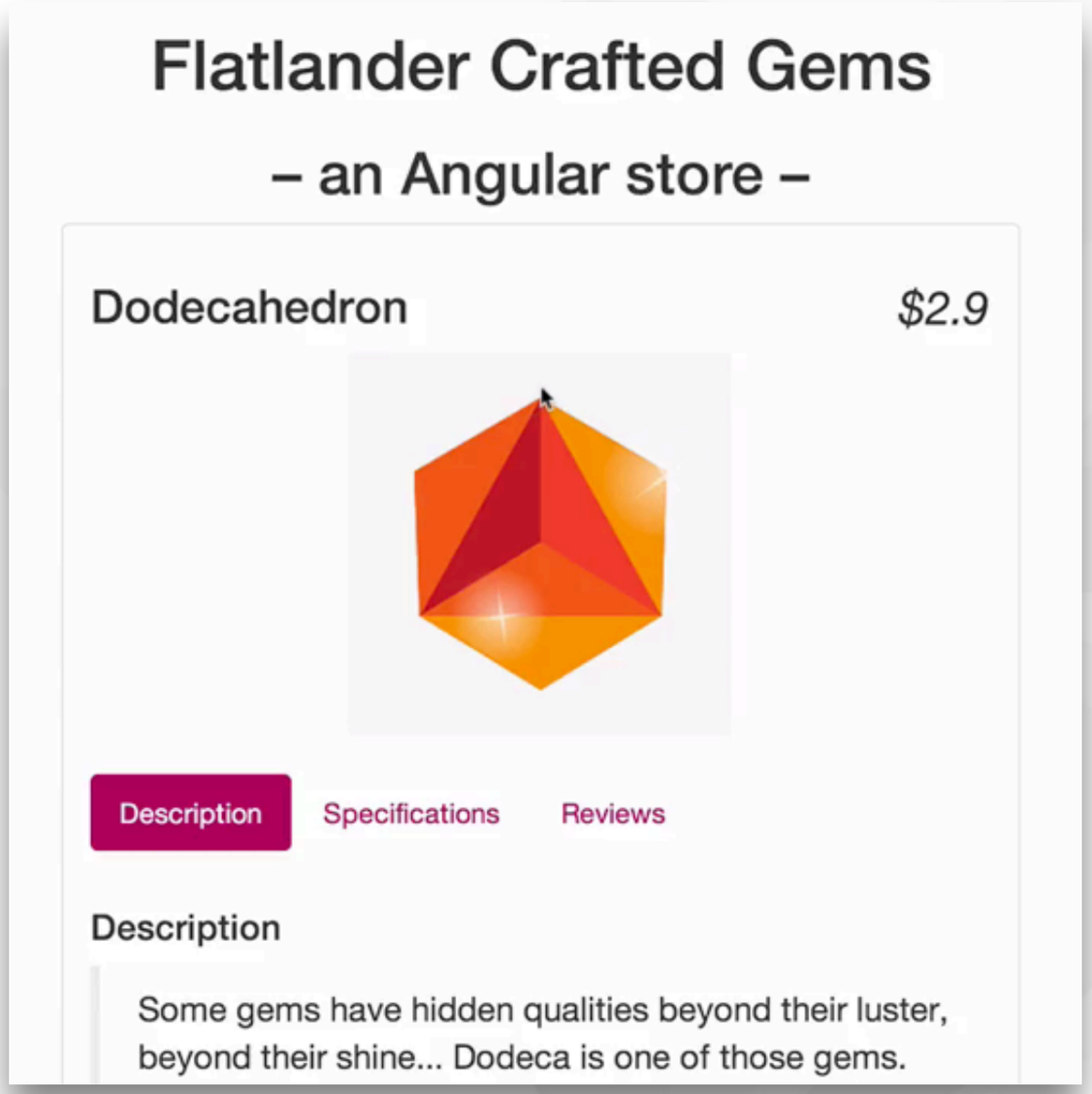


Challenges

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

Description Specifications Reviews

Description

> Some gems have hidden qualities beyond their luster, beyond their shine... Dodeca is one of those gems.

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

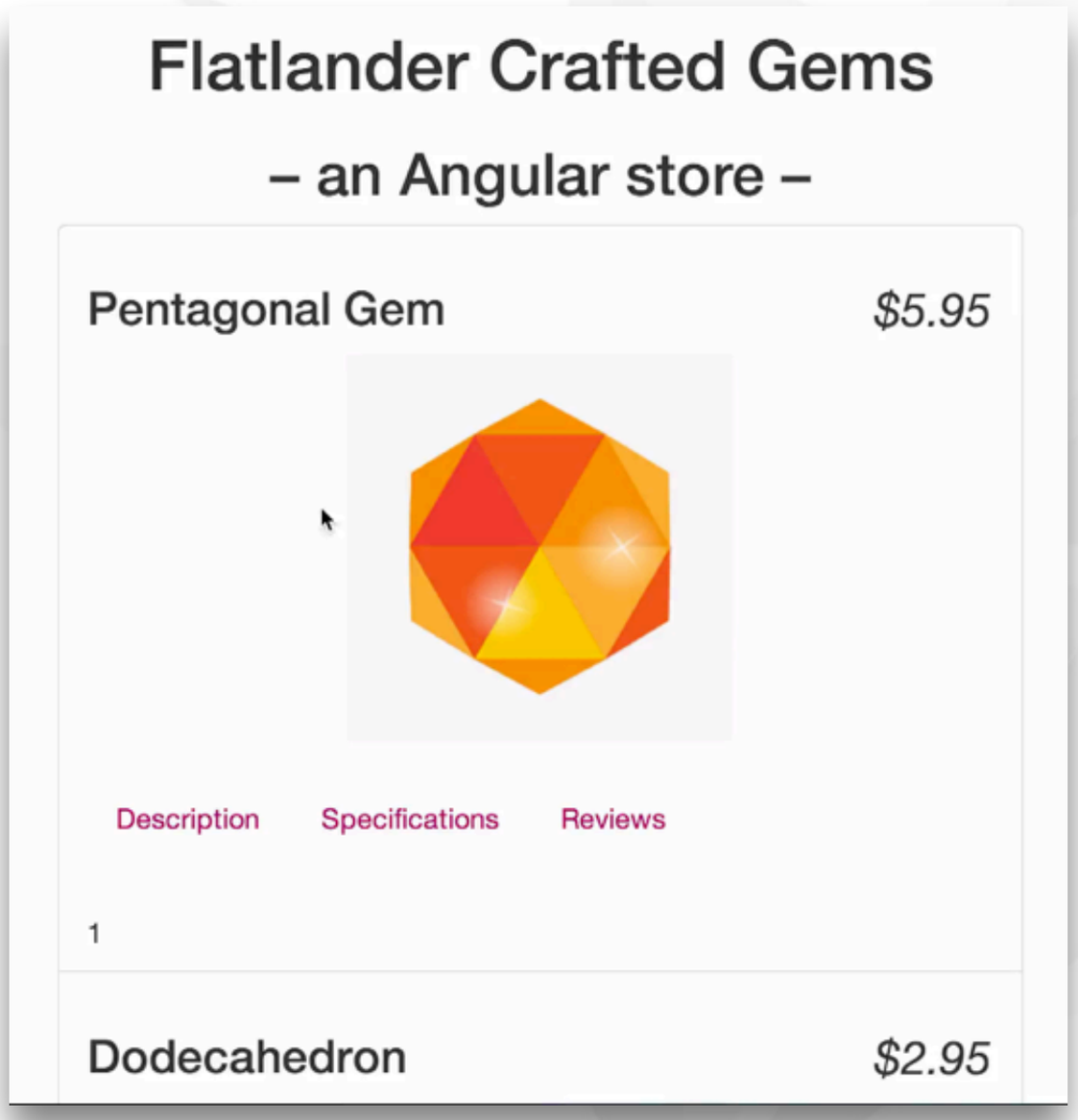
Assigning a value to **tab**.

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>
```

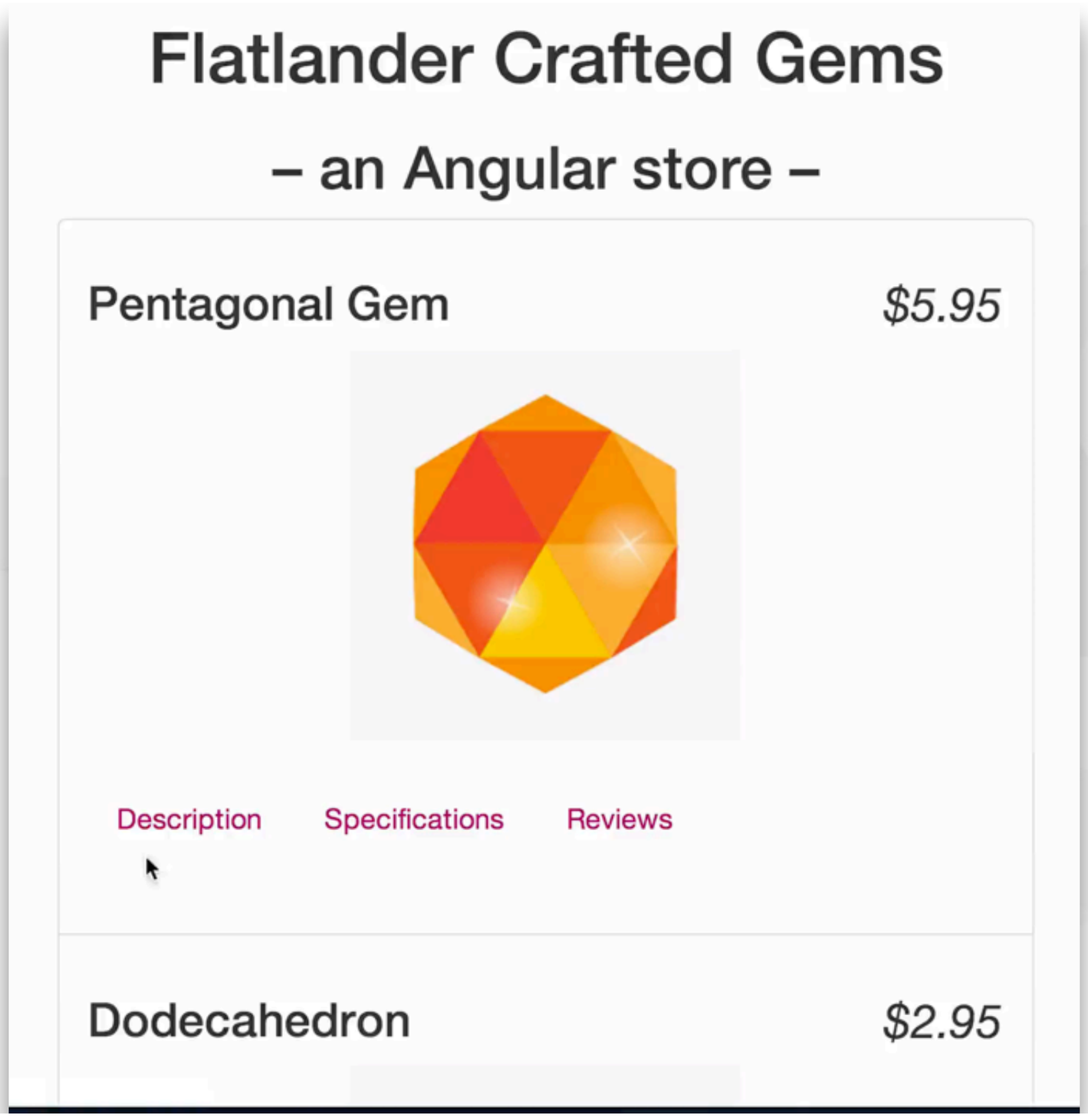
How do we make the tabs trigger the panel to show?

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!



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

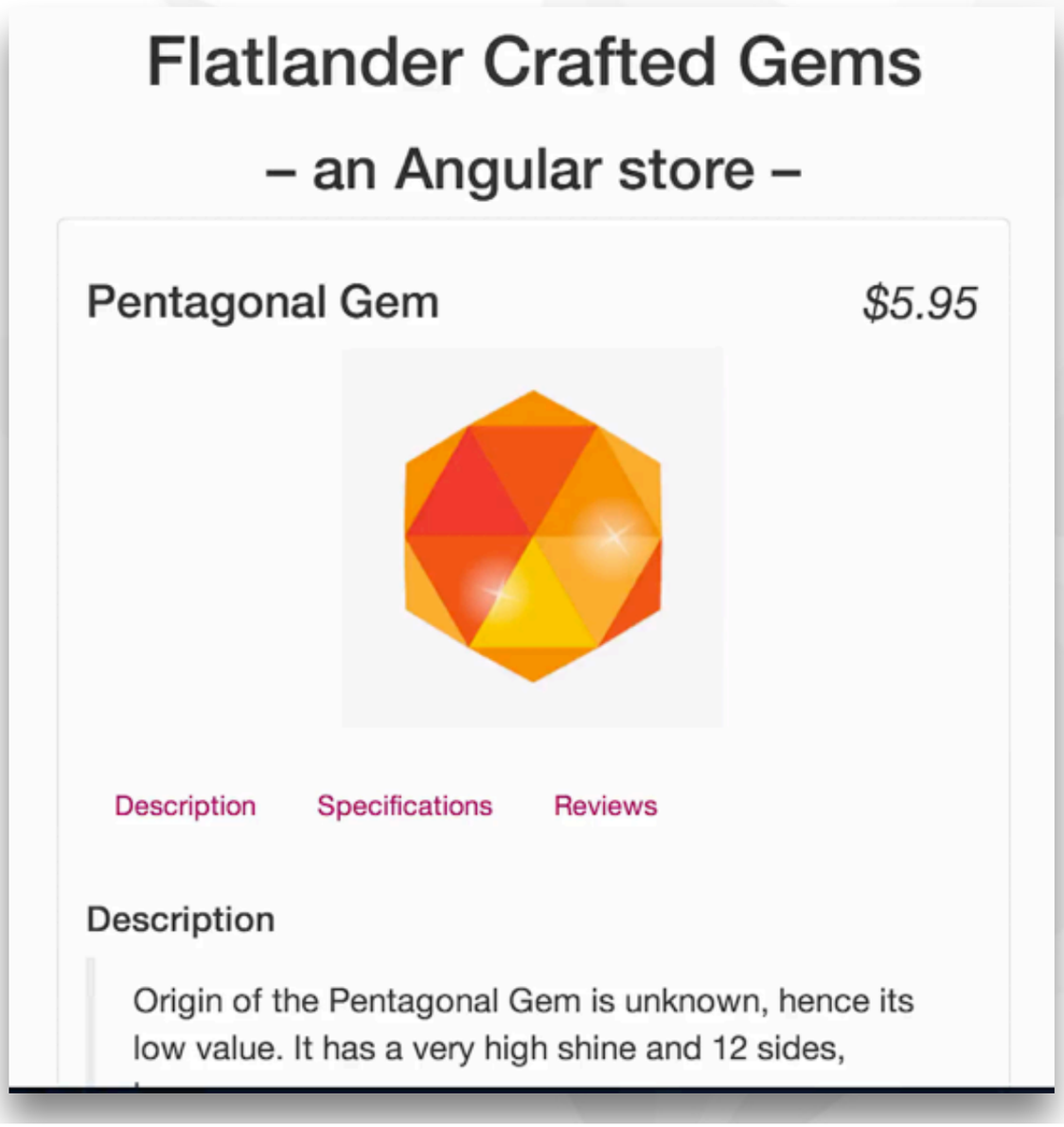
Setting the Initial Value

`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



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

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>
  . . .
```

index.html

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

Name of the class to set.

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>
…
```

index.html

How might we pull this logic into a Controller?

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

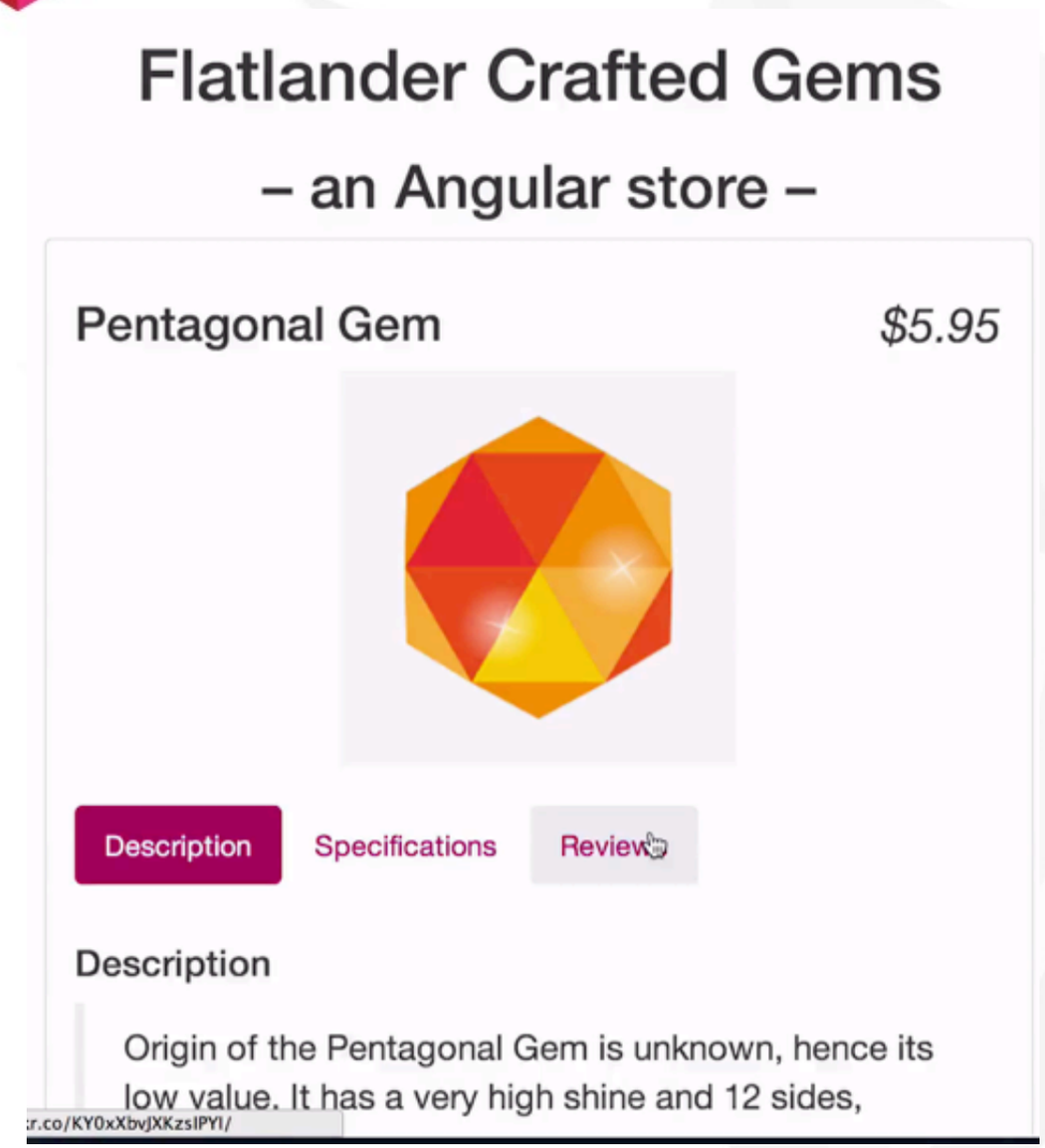
> Origin of the Pentagonal Gem is unknown, hence its low value. It has a very high shine and 12 sides,

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

We're displaying Reviews!

Reviews

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

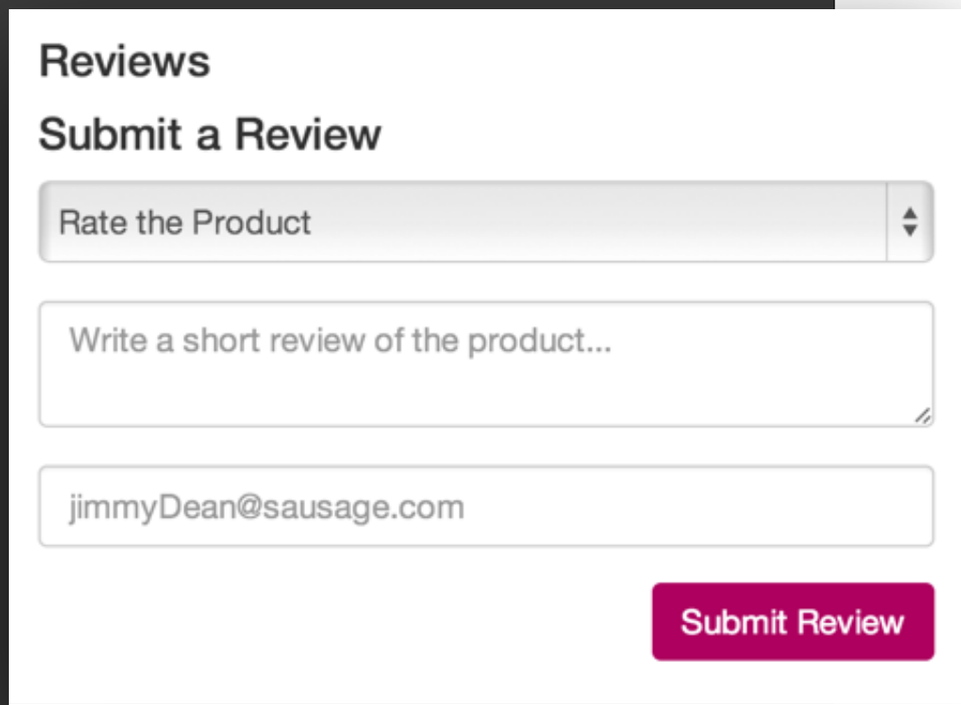
> **4 Stars** Any gem with 12 faces is for me!
> —gemsRock@alyssaNicoll.com

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>
```

index.html

How do we bind this review object to the form?

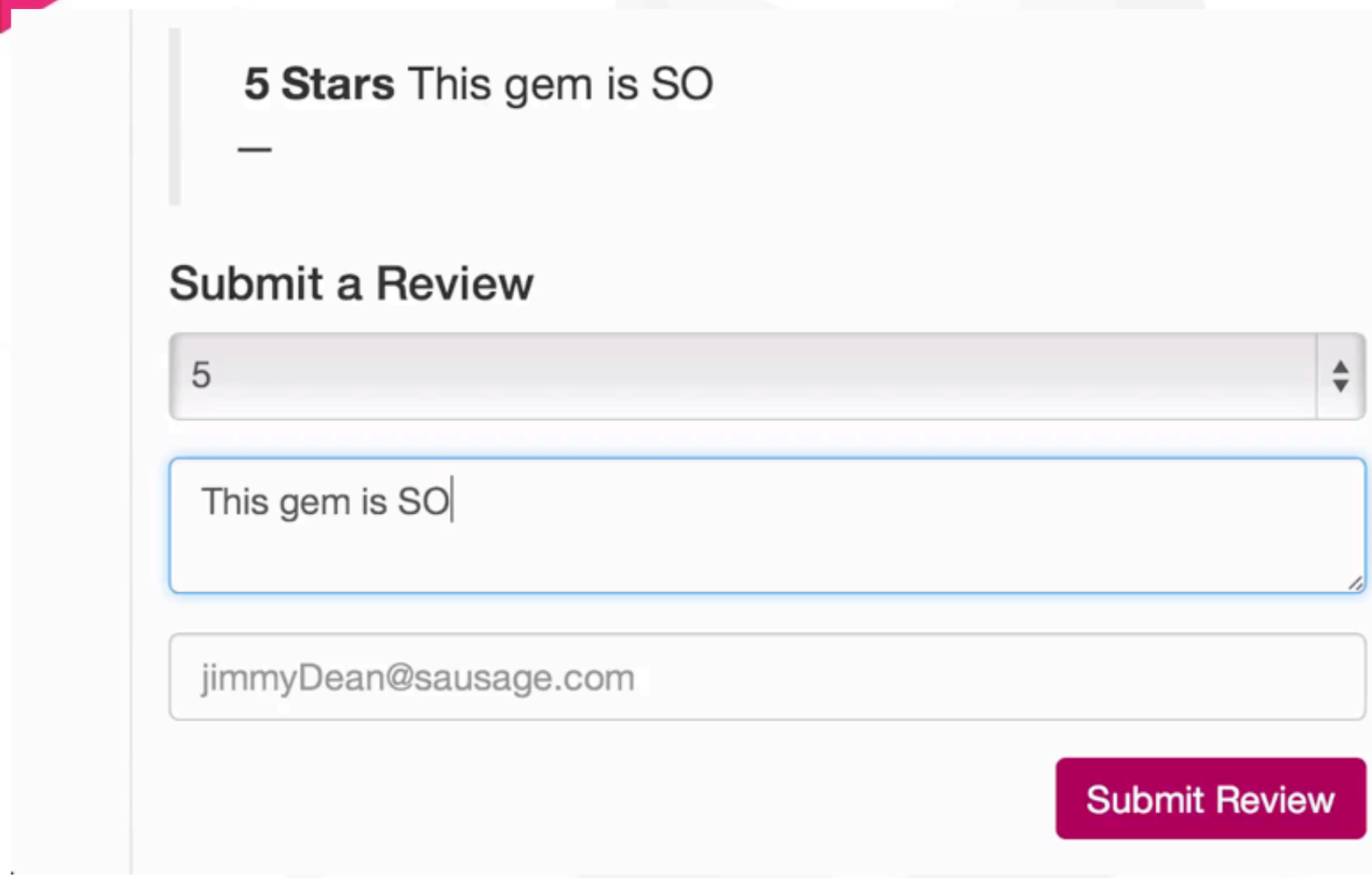
Introducing ng-model

```
<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>
```

index.html

ng-model binds the form element value to the property

Live Preview In Action



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);
  };
});
```

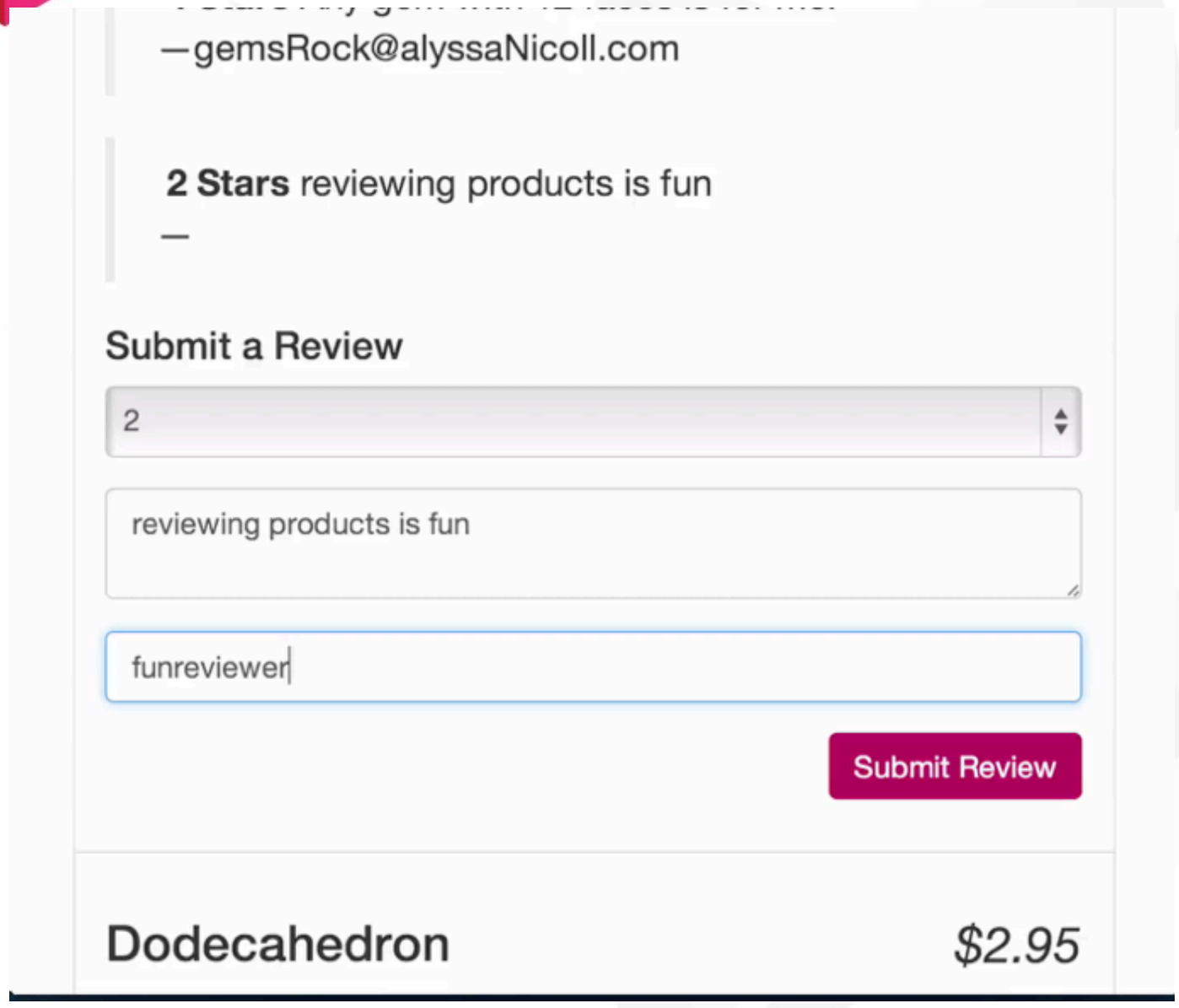
Push **review** onto this product's **reviews** array.

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

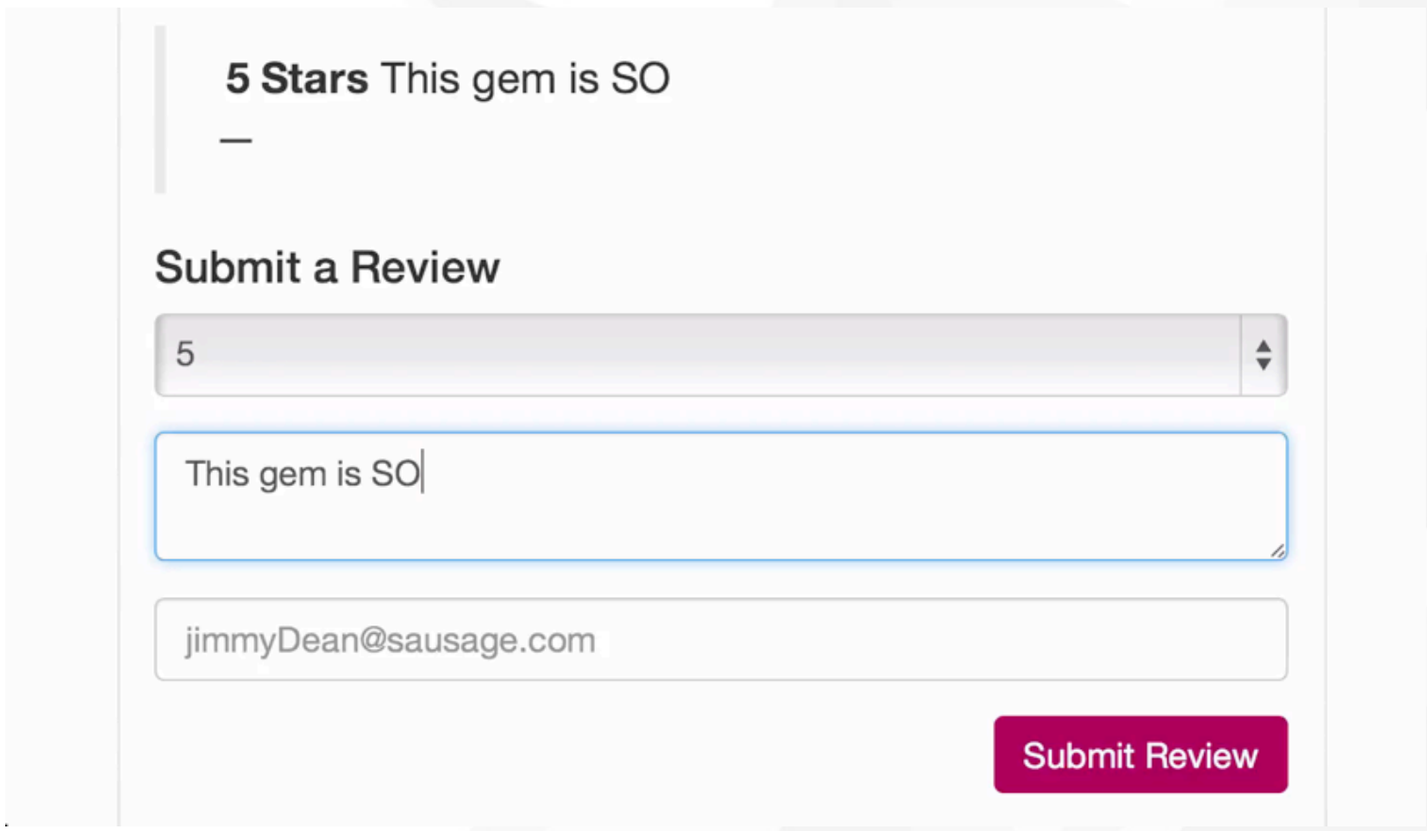
Clear out the review, so the form will reset.

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

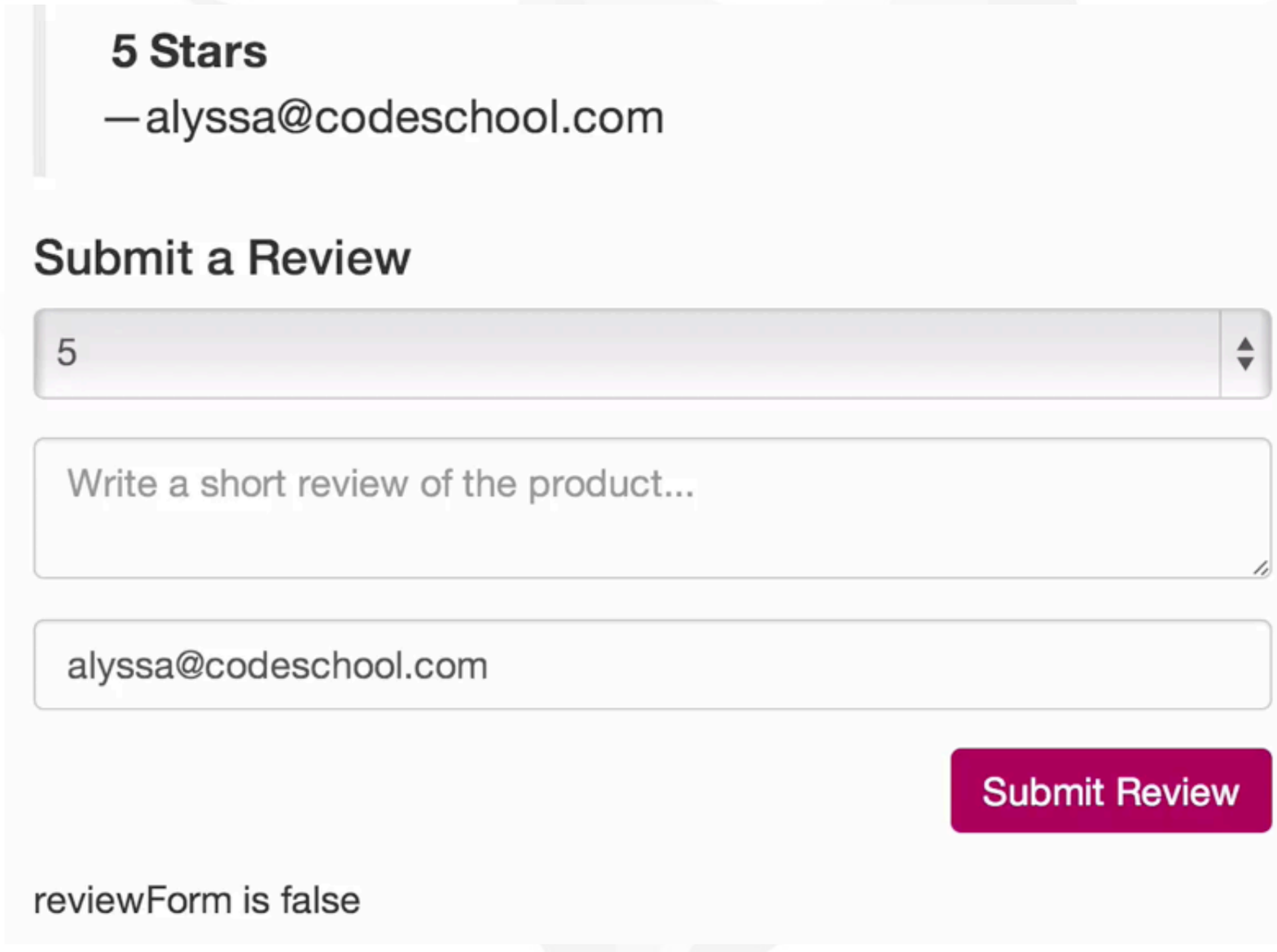
```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl"
                        ng-submit="reviewCtrl.addReview(product)" novalidate>
  <select ng-model="reviewCtrl.review.stars" required>
    <option value="1">1 star</option>
    ...
  </select>

  <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



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

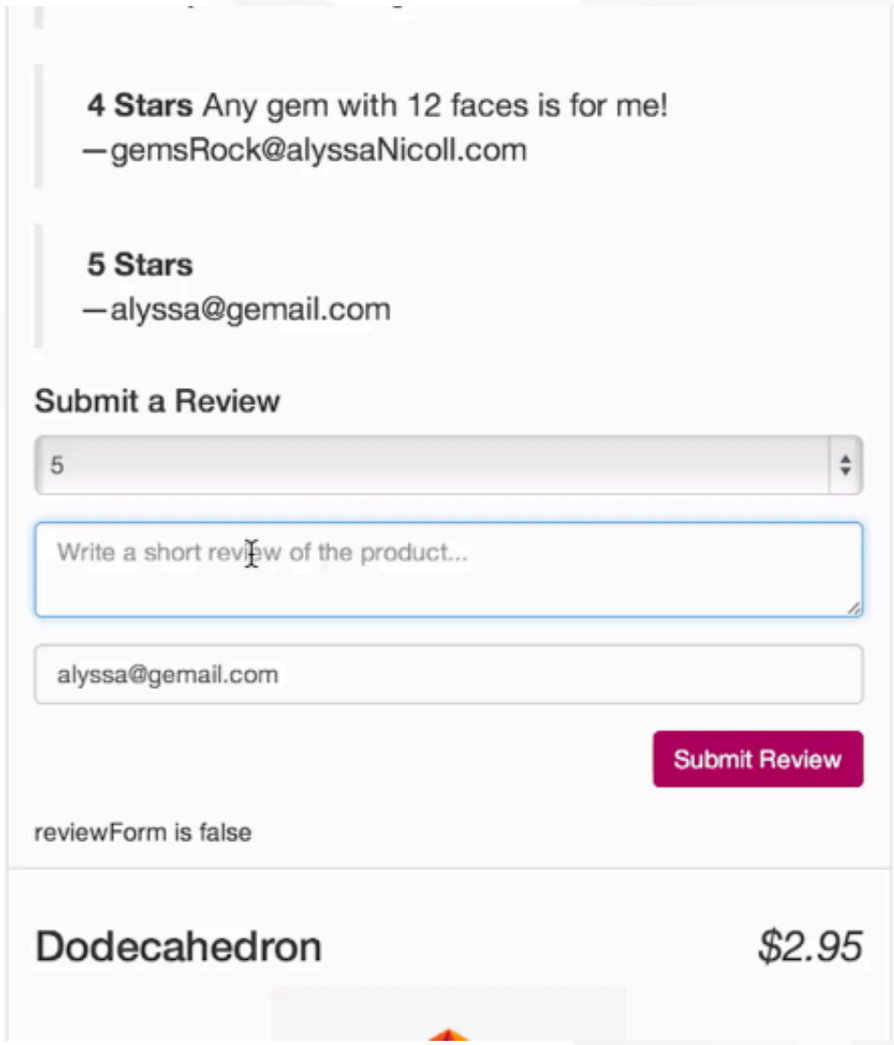
```
<form name="reviewForm" ng-controller="ReviewController as reviewCtrl"
    ng-submit="reviewForm.$valid && reviewCtrl.addReview(product)" novalidate>
```

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;
}
```

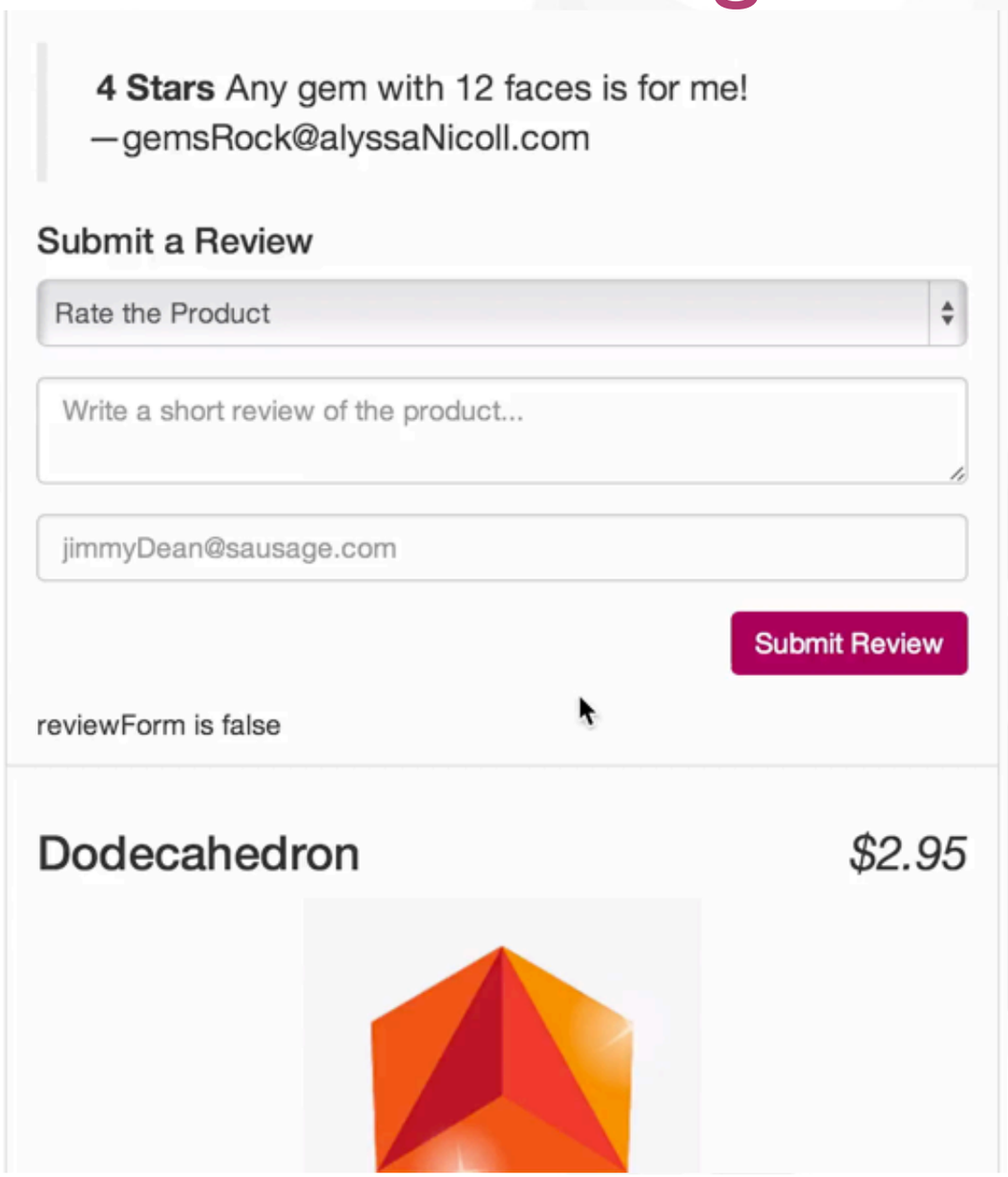
Red border for invalid

Green border for valid

style.css

SHAPING UP WITH ANGULAR.JS

Now with red and green borders!



HTML5-based type validations

Web forms usually have rules around valid input:

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

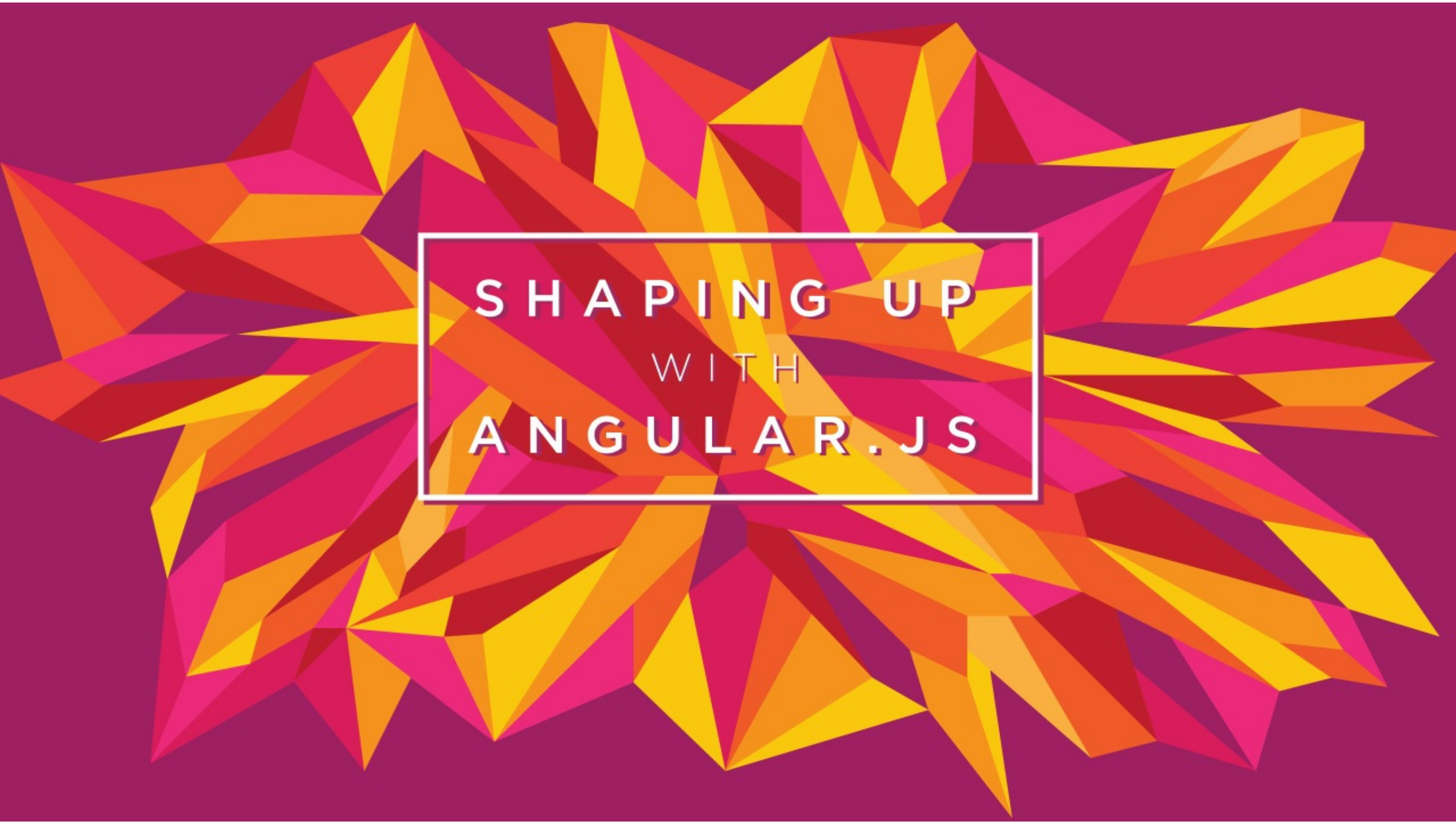
```
<input type="email" name="email">
```

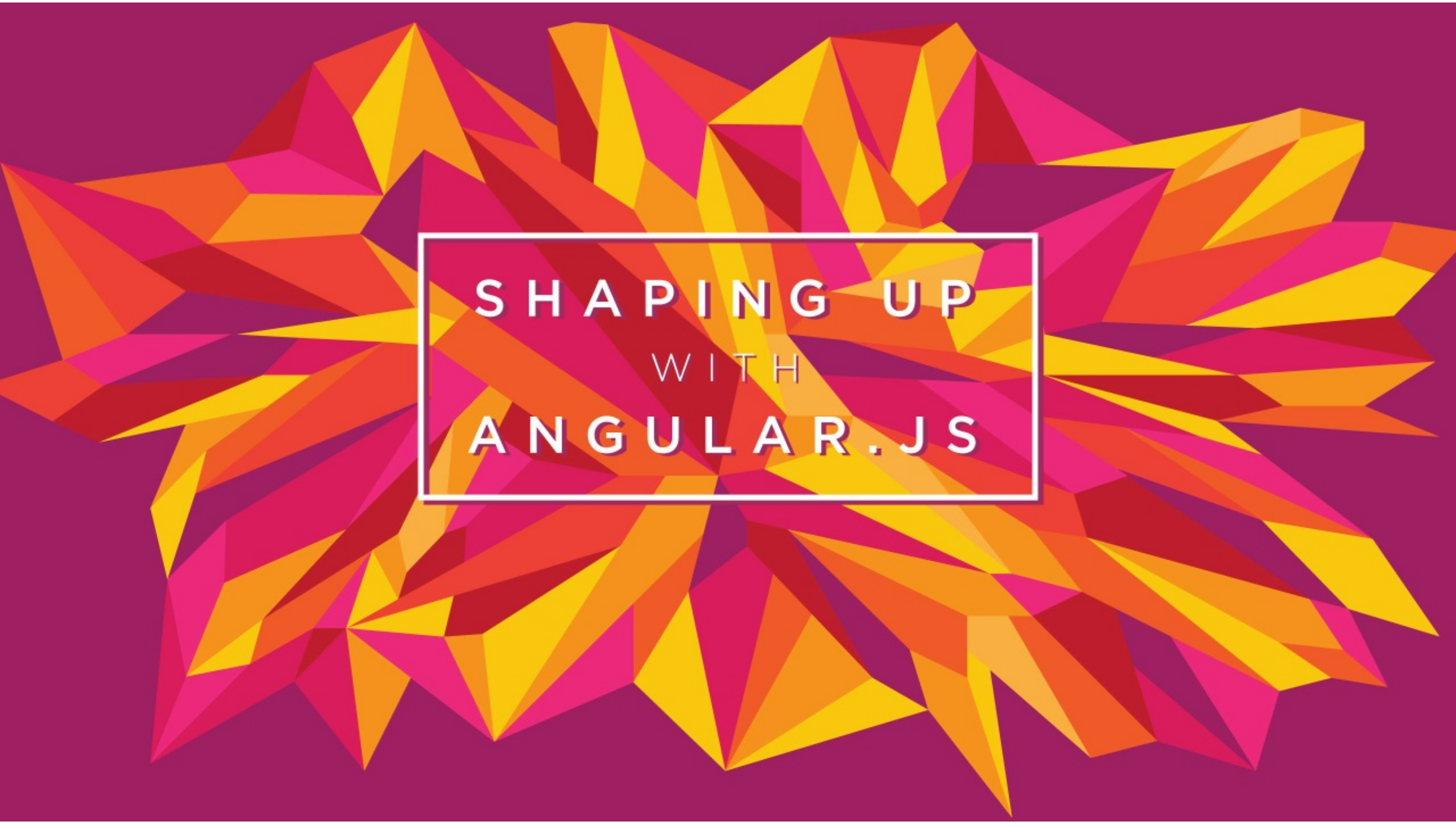
```
<input type="url" name="homepage">
```

```
<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

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?

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?

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?

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.

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>
```

Directive?

index.html

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

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*

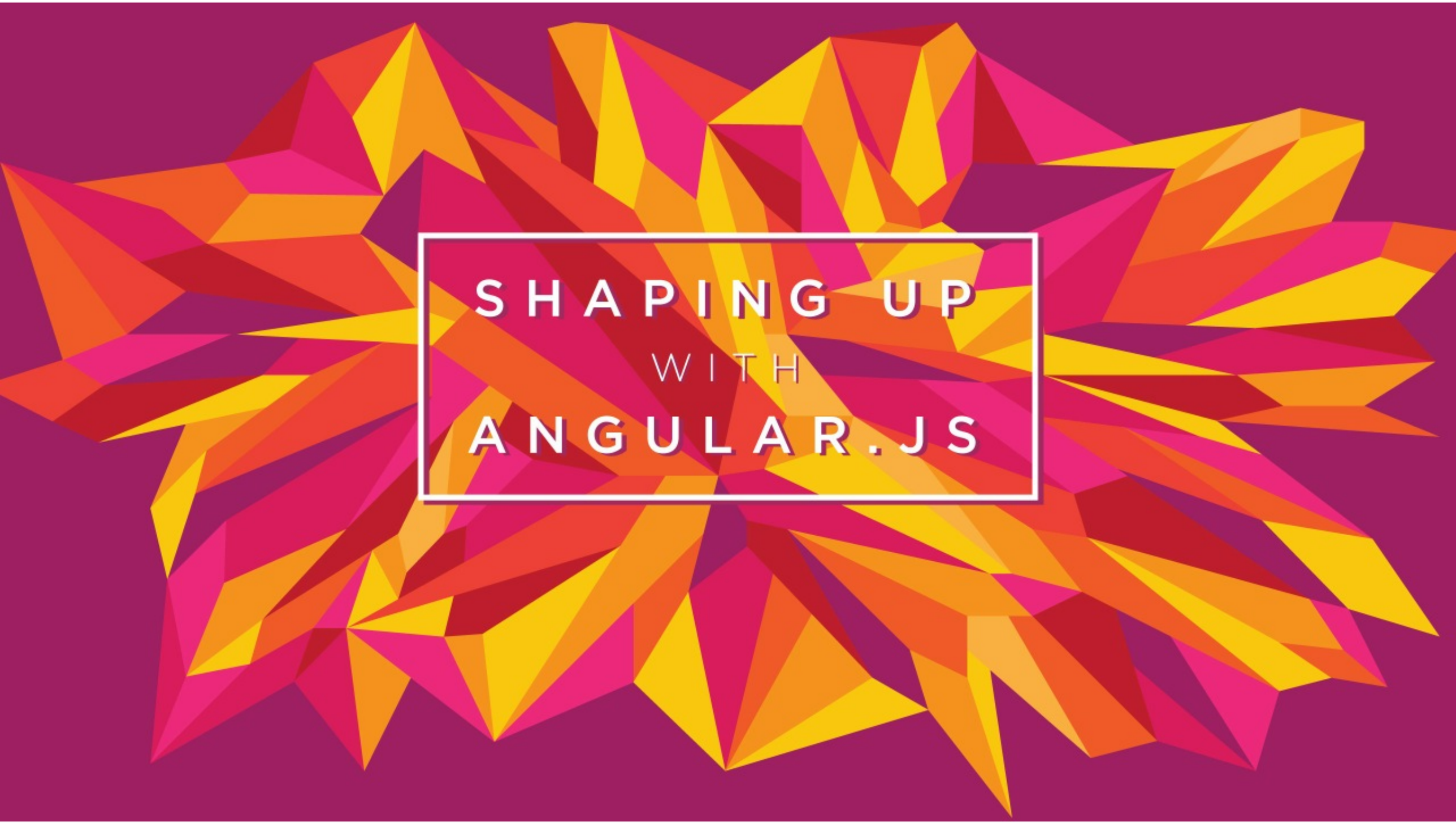
Description Specifications Reviews

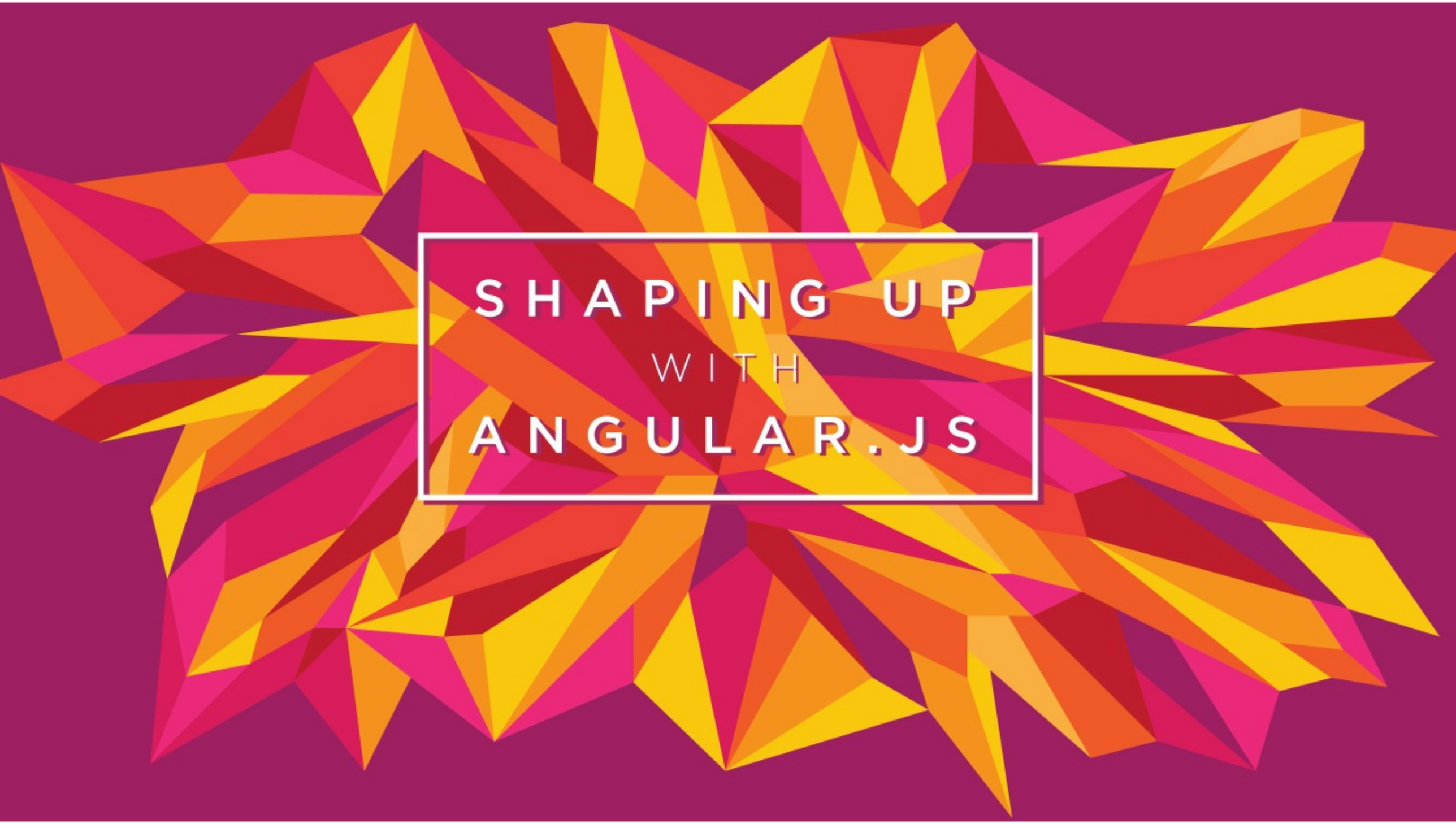
Description

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(){ . . . });

  . . .
})();
```

app.js

Module Name

```
(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

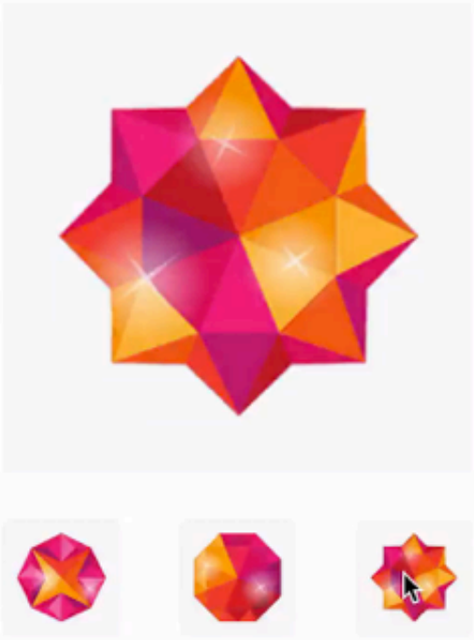
```
<!DOCTYPE html>
<html ng-app="store">
  <head> . . . </head>
  <body ng-controller="StoreController as store">
    . . .
    <script src="angular.js"></script>
    <script src="app.js"></script>
    <script src="products.js"></script>
  </body>
</html>
```

index.html

Flatlander Crafted Gems

– an Angular store –

Gem #1: Zircon *$1,100.00*



Description Specs Reviews

Description

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

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?

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?

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

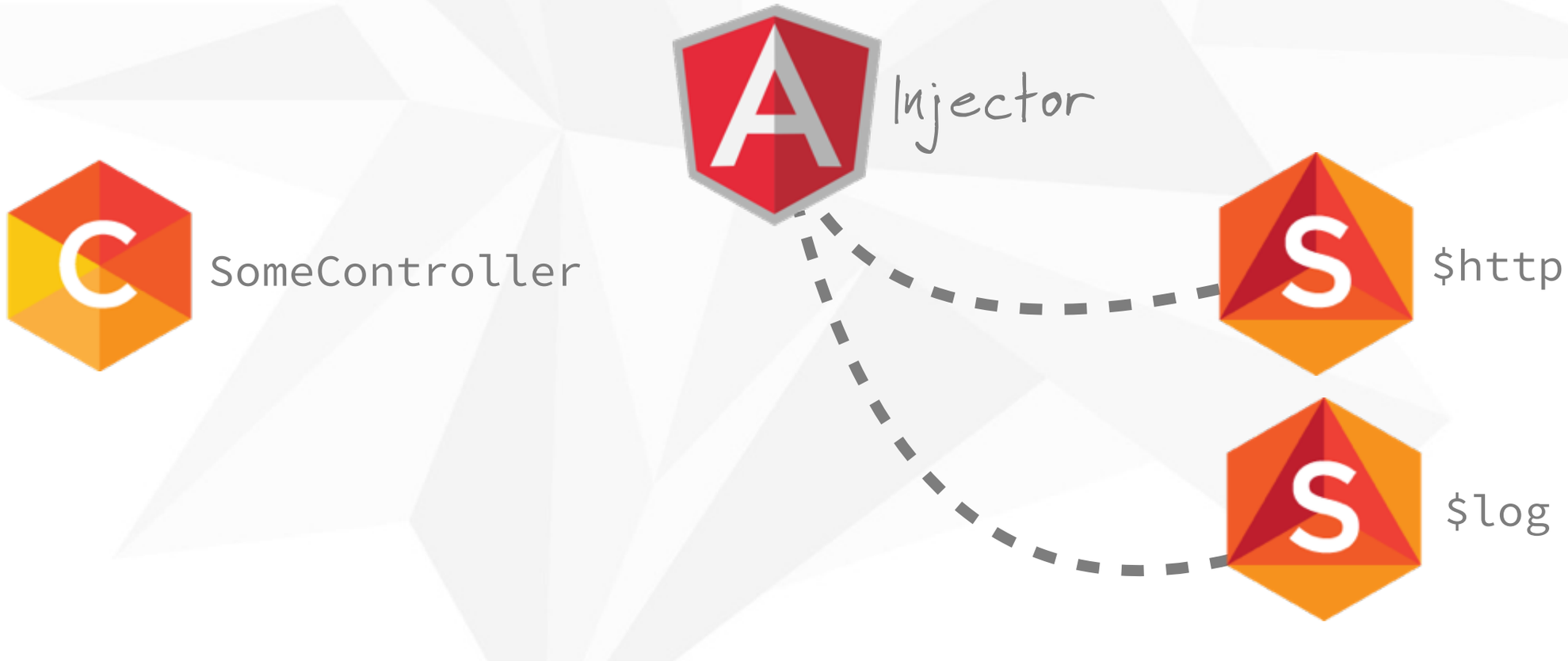
Dependency Injection!

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

} ]);
```

If you needed more than one

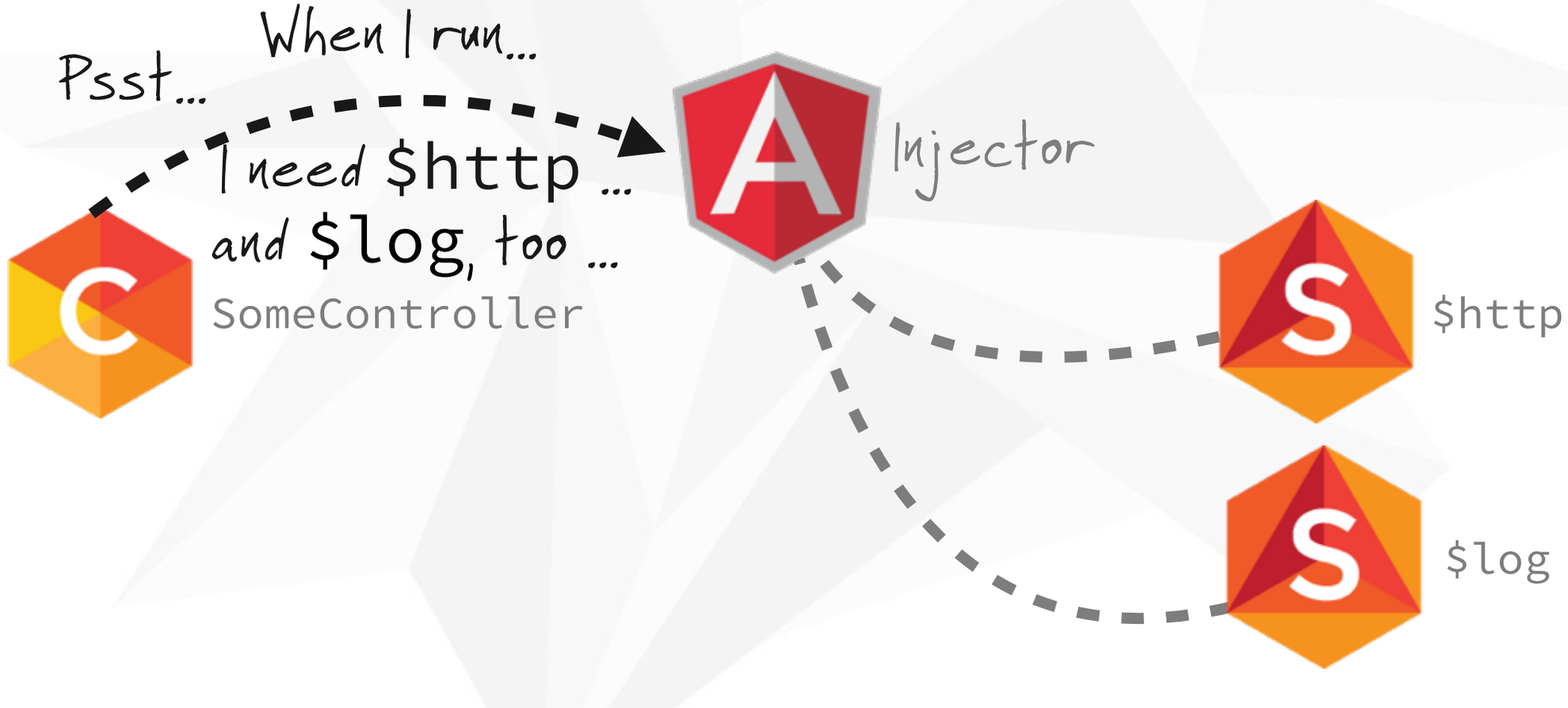
When Angular is Loaded Services are Registered

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

} ]);
```



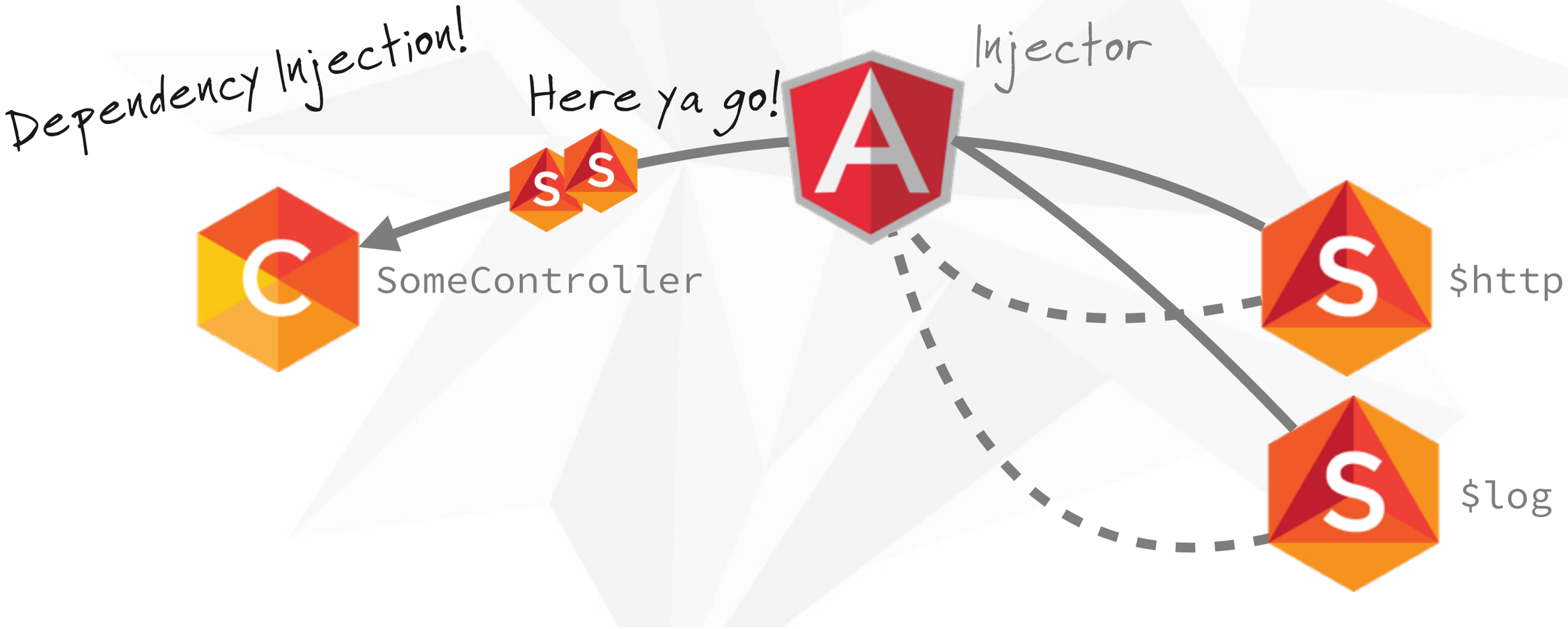
A Controller is Initialized

```
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

Time for your injection!

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

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

app.js

StoreController needs the $http Service...

...so $http gets injected as an argument!

Now what?

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

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

We need to store what this is ...

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

app.js

But the page might look funny until the data loads.

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

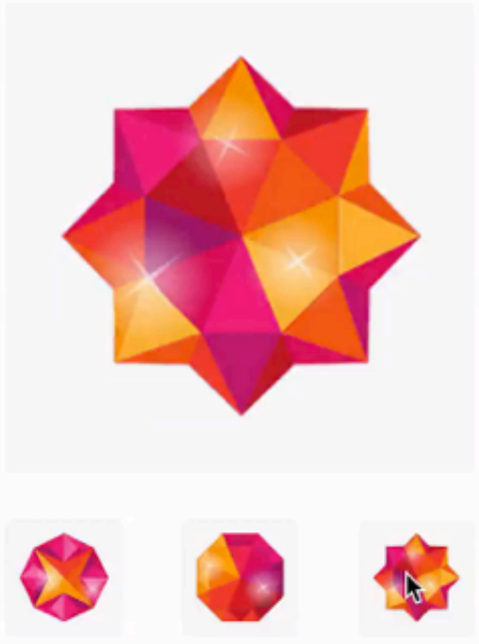
We need to initialize products to an empty array, since the page will render before our data returns from our get request.

app.js

Flatlander Crafted Gems

– an Angular store –

Gem #1: Zircon *$1,100.00*



Description Specs Reviews

Description

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:

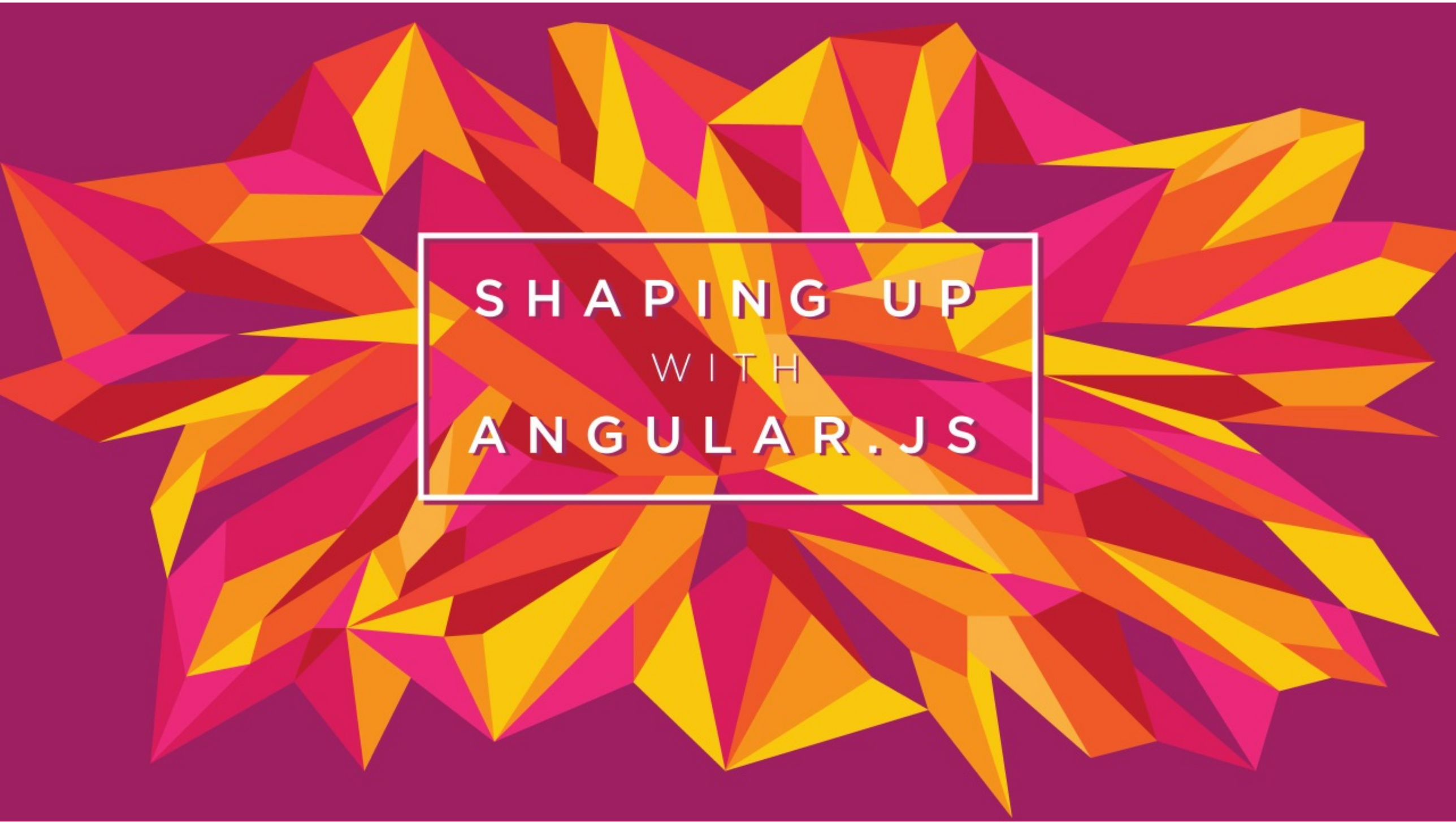
```
$http({ method: 'OPTIONS', url: '/path/to/resource.json' });
```

```
$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