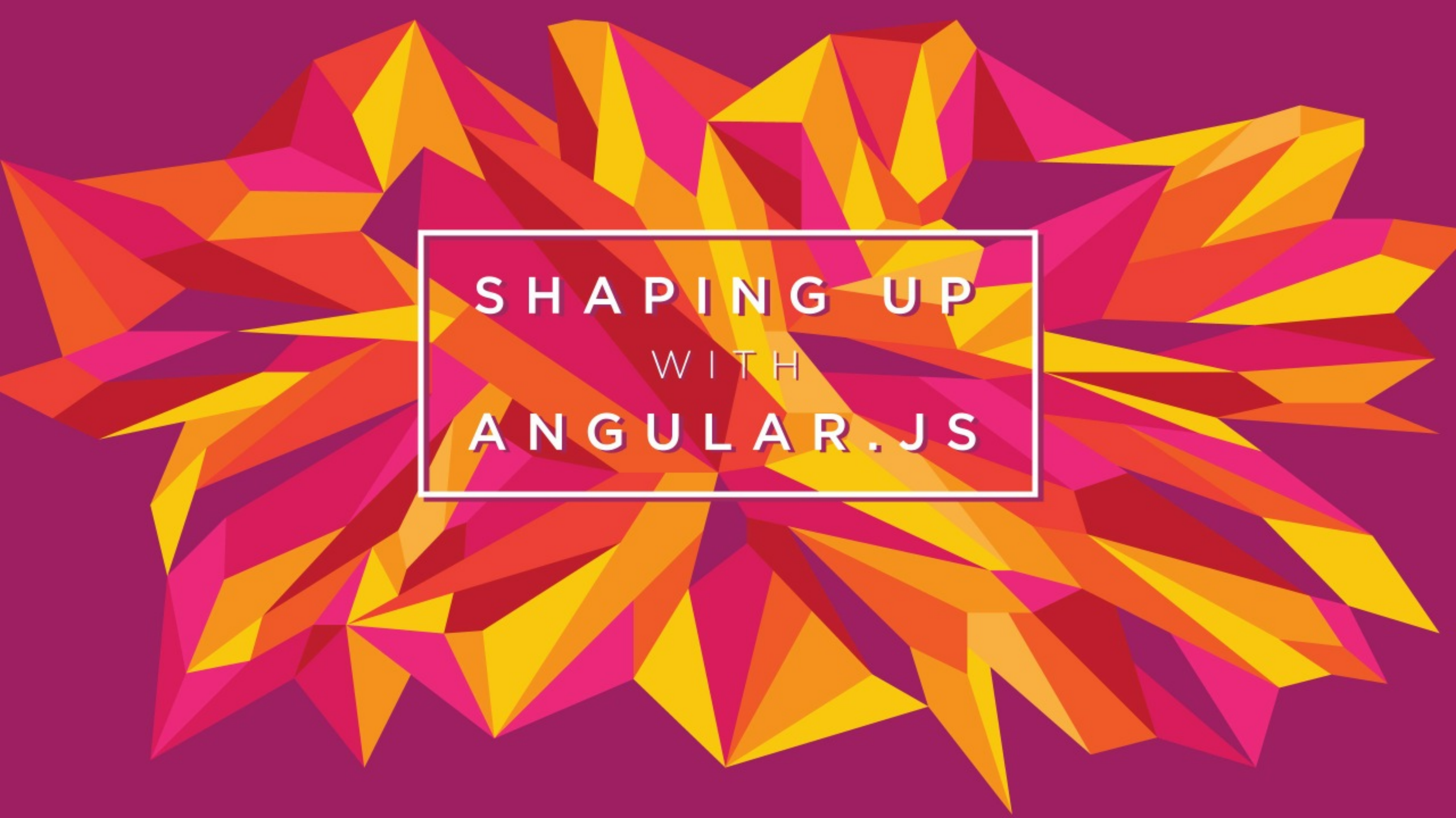

SHAPING UP
WITH
ANGULAR.JS

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

SHAPING UP
WITH
ANGULAR.JS

Traditional Page-Refresh

Web Server

Web Browser

URL Request to server

Response with Webpage & Assets

HTML JavaScript

Browser loads up entire webpage.

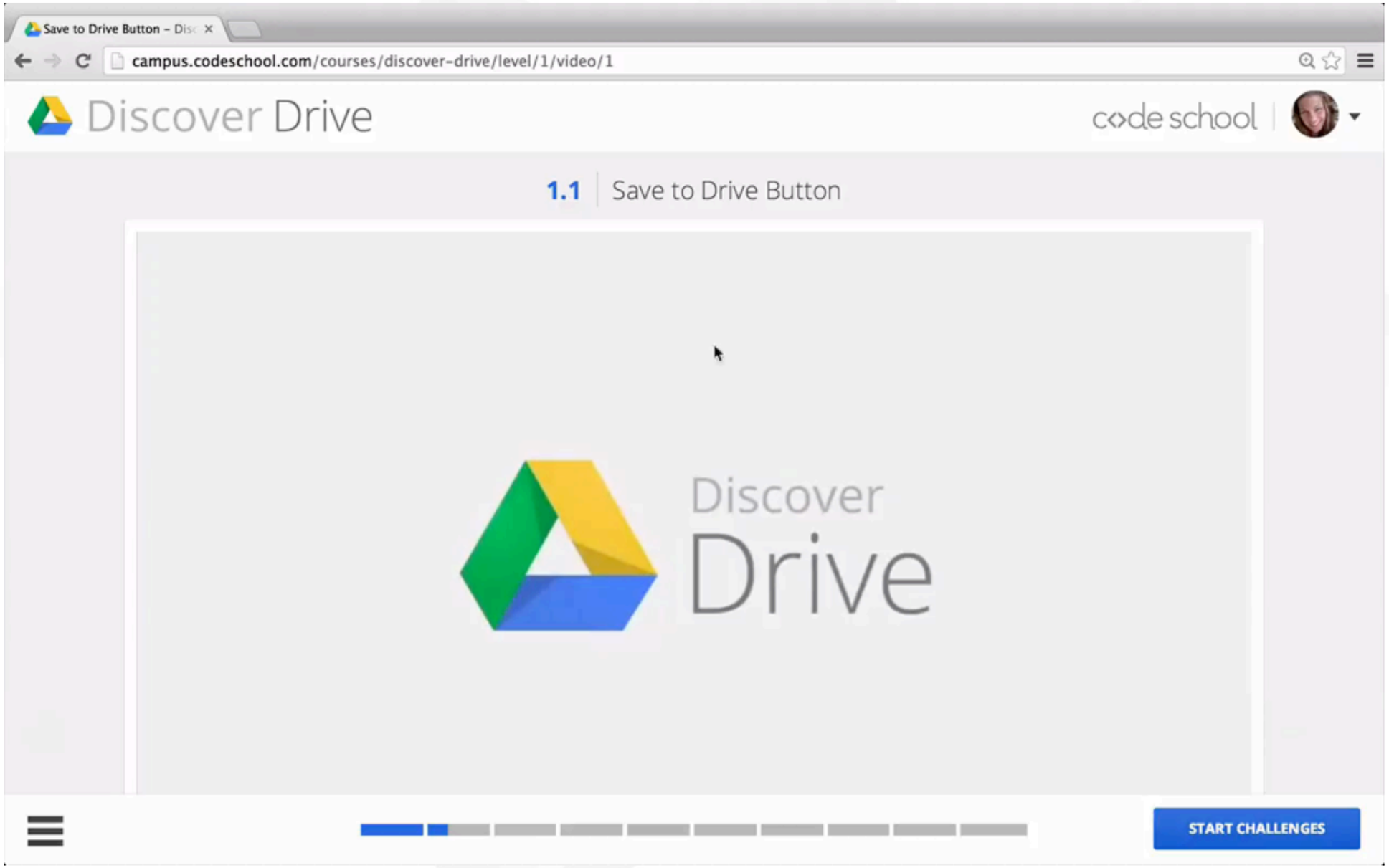
User clicks on link, new Request

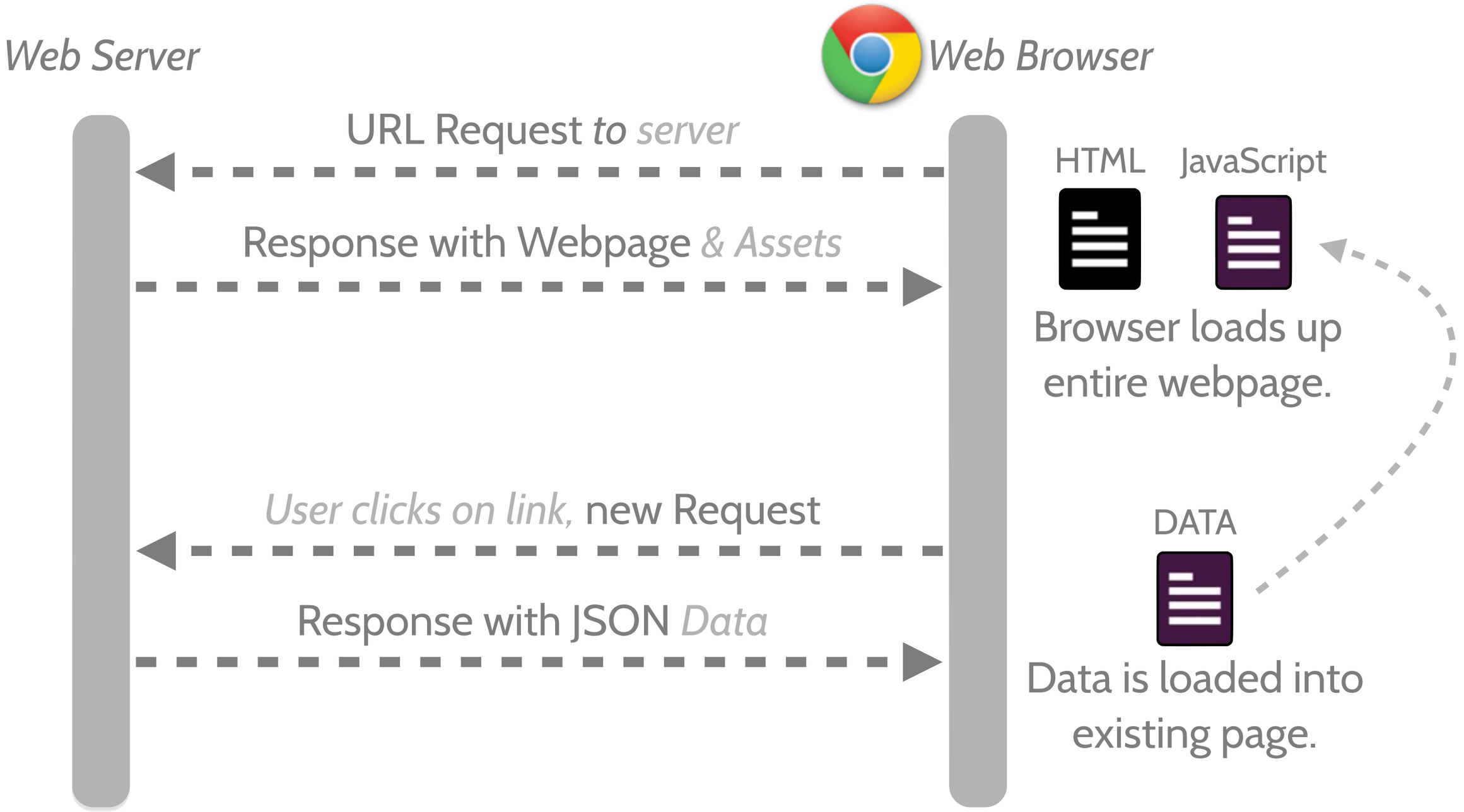
Response with Webpage & Assets

HTML JavaScript

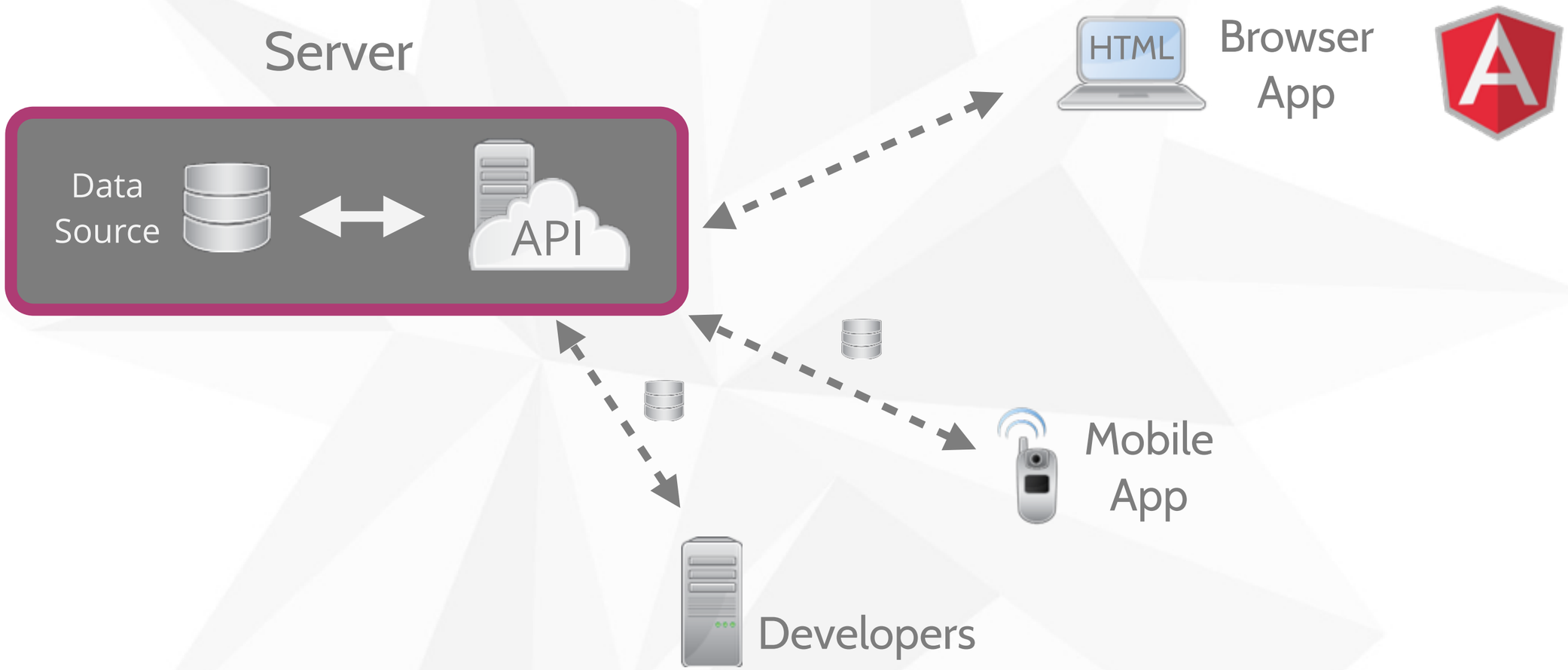
Browser loads up entire webpage.

A “responsive” website using Angular




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

Modern API-Driven Application



What is Angular JS?

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

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

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

index.html

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

app.js

SHAPING UP WITH ANGULAR.JS

Directives

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

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

index.html

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

Name of function to call

app.js

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

Welcome, Gregg!

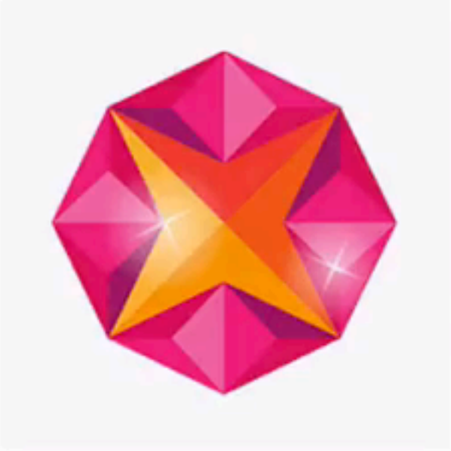
OK

SHAPING UP WITH ANGULAR.JS

Flatlander Crafted Gems

– an Angular store –

Gem #1: Zircon *$1,100.00*



Description Specs Reviews

Description

Downloading the libraries

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

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

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

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

SHAPING UP WITH ANGULAR.JS

Getting Started

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

Twitter Bootstrap

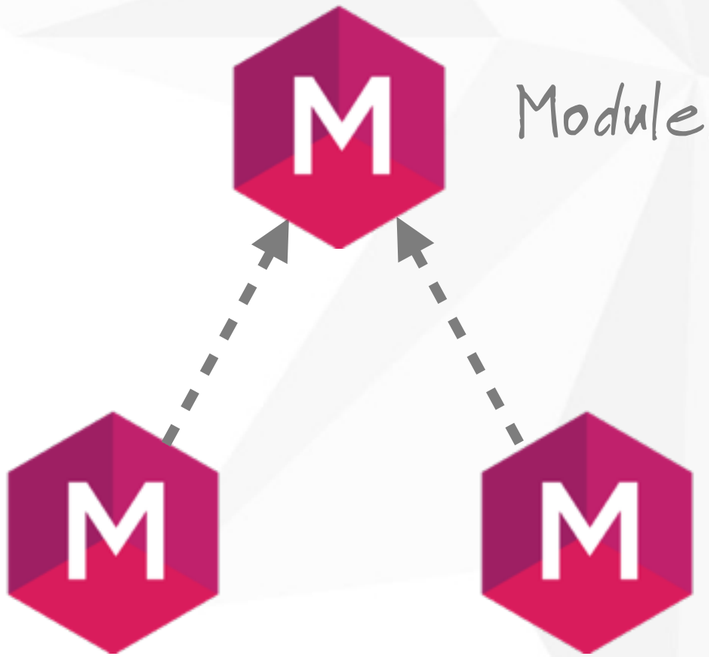
AngularJS

index.html

SHAPING UP WITH ANGULAR.JS

Modules

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



Modules can use other Modules...

SHAPING UP WITH ANGULAR.JS

Creating Our First Module

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

AngularJS

Application Name

Dependencies

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

SHAPING UP WITH ANGULAR.JS

Including Our Module

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

index.html

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

app.js

SHAPING UP WITH ANGULAR.JS

Including Our Module

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

Run this module when the document loads

index.html

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

app.js

SHAPING UP WITH ANGULAR.JS

Expressions

Allow you to insert dynamic values into your HTML.

Numerical Operations

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

evaluates to

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

String Operations

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

evaluates to

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

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

SHAPING UP WITH ANGULAR.JS

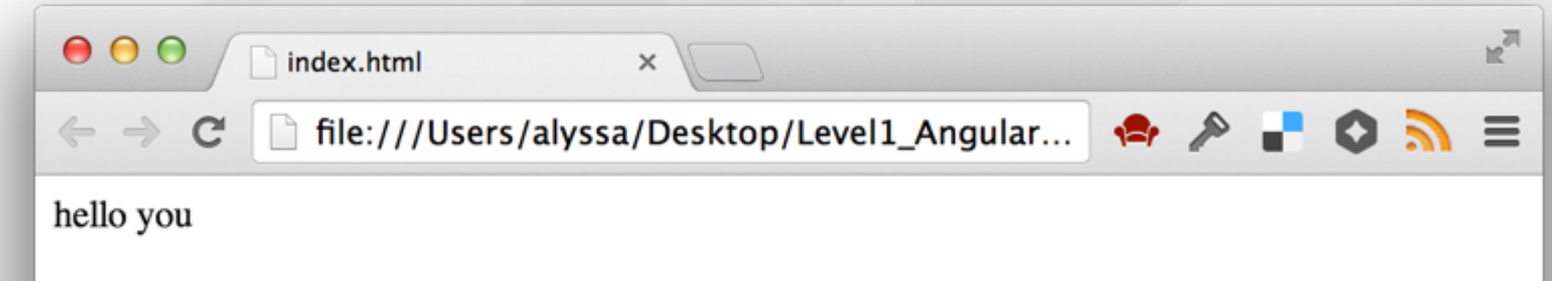
Including Our Module

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

index.html

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

app.js



SHAPING UP WITH ANGULAR.JS

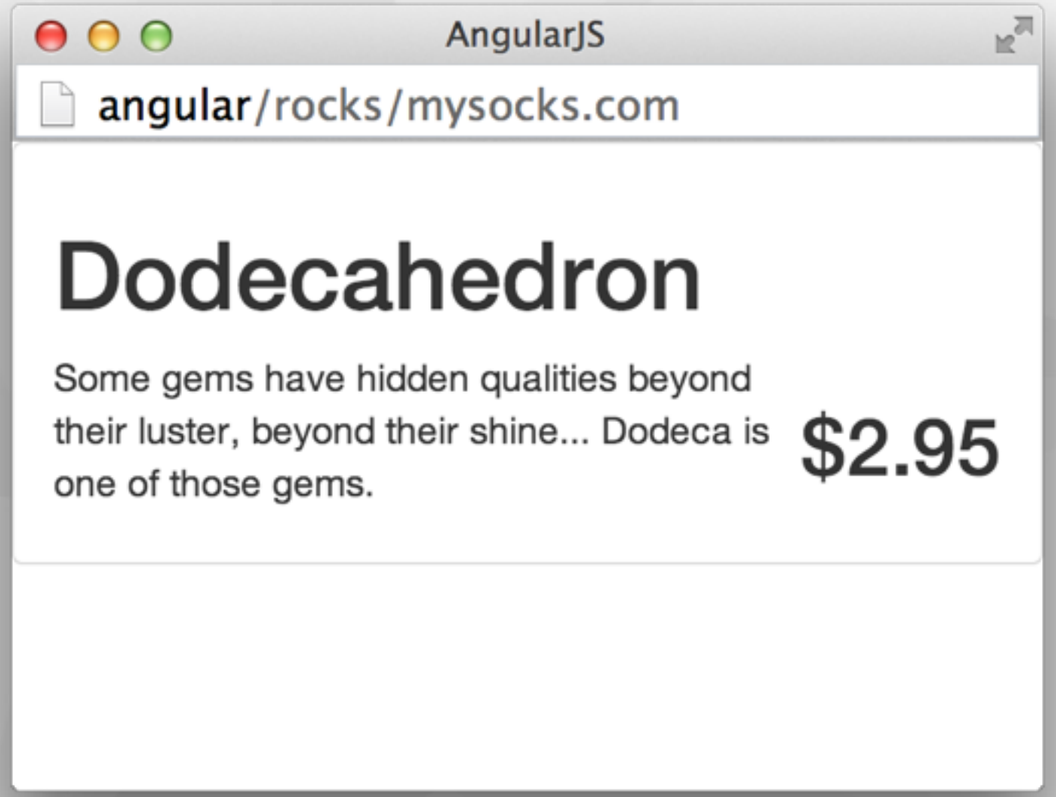
Challenges

SHAPING UP
WITH
ANGULAR.JS

Working With Data

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

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



Controllers

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

Wrapping your Javascript in a closure is a good habit!

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

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

  });
})();
```

app.js

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

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

SHAPING UP WITH ANGULAR.JS

Storing Data Inside the Controller

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

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

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

app.js

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

SHAPING UP WITH ANGULAR.JS

Our Current HTML

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

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

index.html

Attaching the Controller

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

index.html

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

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

app.js

SHAPING UP WITH ANGULAR.JS

Attaching the Controller

Directive Controller name Alias

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

index.html

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

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

app.js

SHAPING UP WITH ANGULAR.JS

Displaying Our First Product

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

index.html

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

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

Understanding Scope

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

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

Would never print a value!

SHAPING UP WITH ANGULAR.JS

Challenges

SHAPING UP
WITH
ANGULAR.JS

Adding A Button

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

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

index.html

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

SHAPING UP WITH ANGULAR.JS

Adding A Button

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

index.html

Directives to the rescue!

How can we only show this button...

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

...when this is true?

SHAPING UP WITH ANGULAR.JS

NgShow Directive

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

index.html

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

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

SHAPING UP WITH ANGULAR.JS

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

their shine... Dodeca i

index.html

NgHide Directive

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

index.html

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

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

SHAPING UP WITH ANGULAR.JS

NgHide Directive

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

index.html

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

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

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

SHAPING UP WITH ANGULAR.JS

NgHide Directive

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

index.html

Much better!

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

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

SHAPING UP WITH ANGULAR.JS

Multiple Products

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


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

app.js

SHAPING UP WITH ANGULAR.JS

Multiple Products

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

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

So we have multiple products...

Now we have an array...

Maybe a Directive?

How might we display all these products in our template?

app.js

Working with An Array

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

index.html

Working with An Array

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

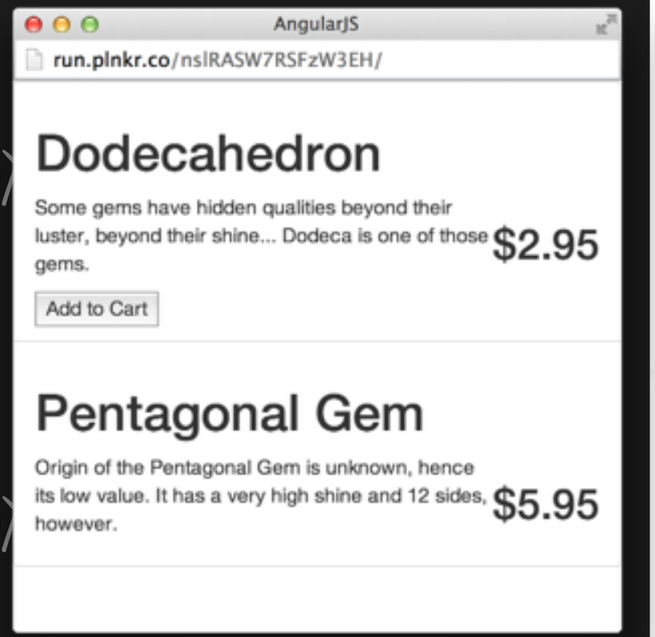
Displaying the first product is easy enough...

index.html

Working with An Array

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

That works...



Why

Why

Why... You get it.

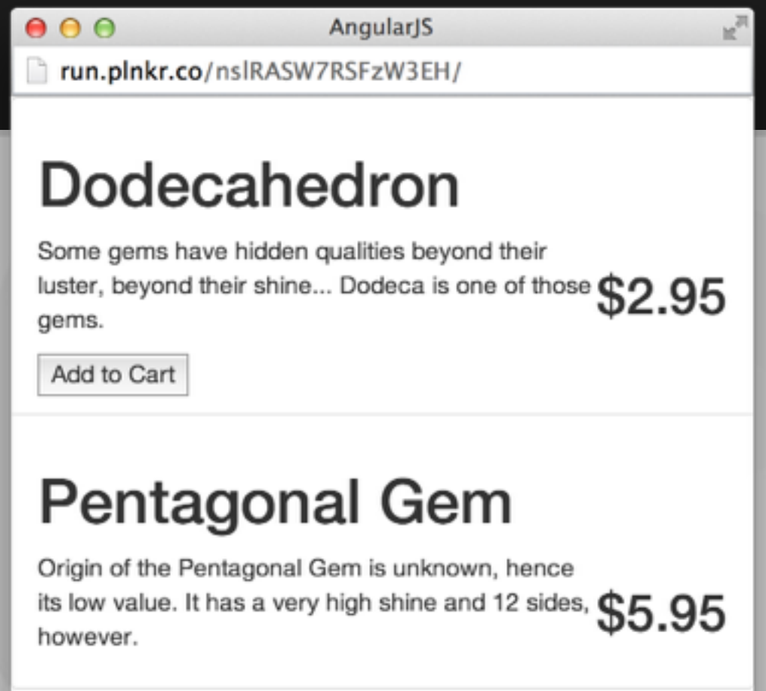
index.html

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

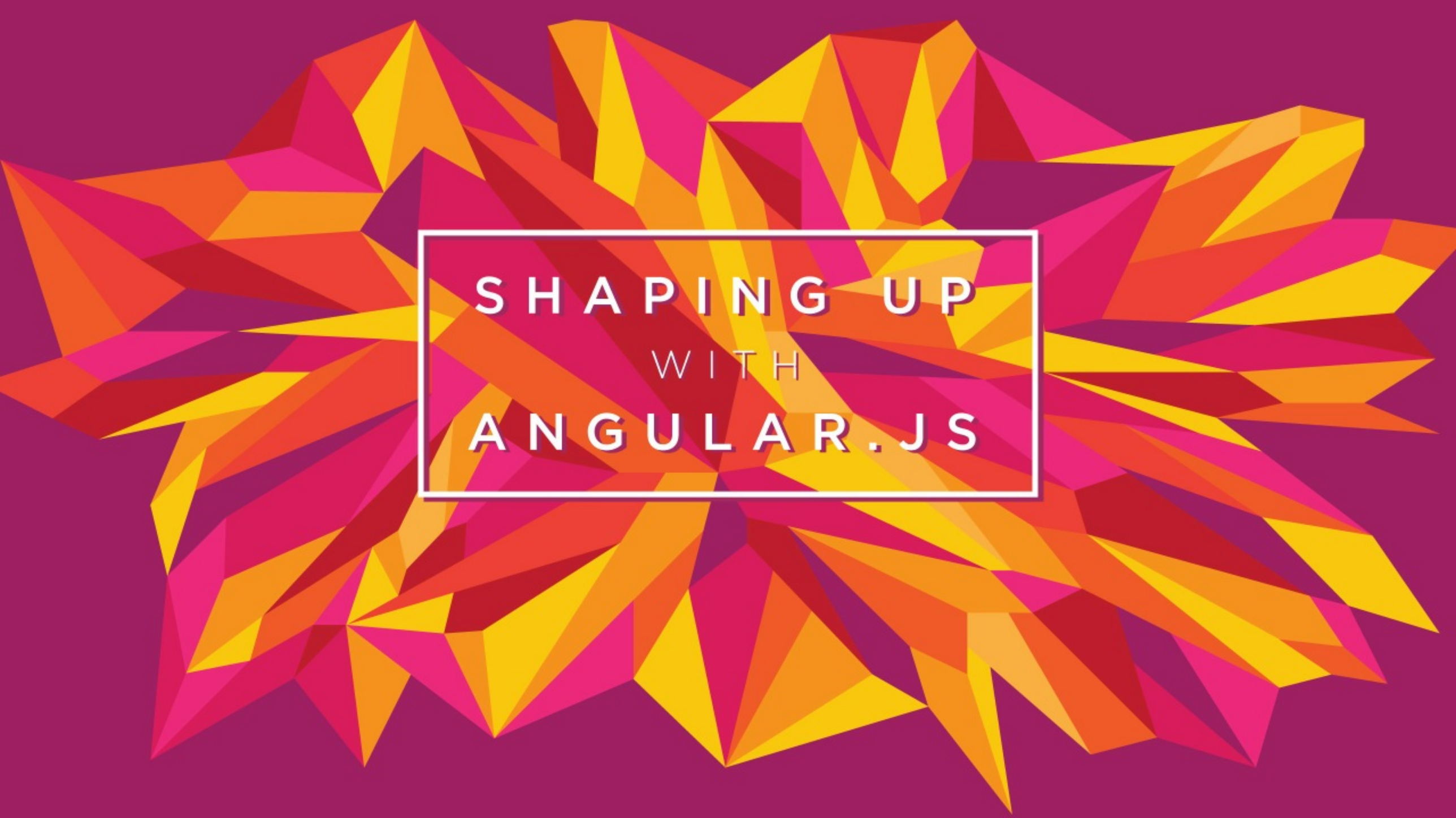
Controllers – Where we add application behavior

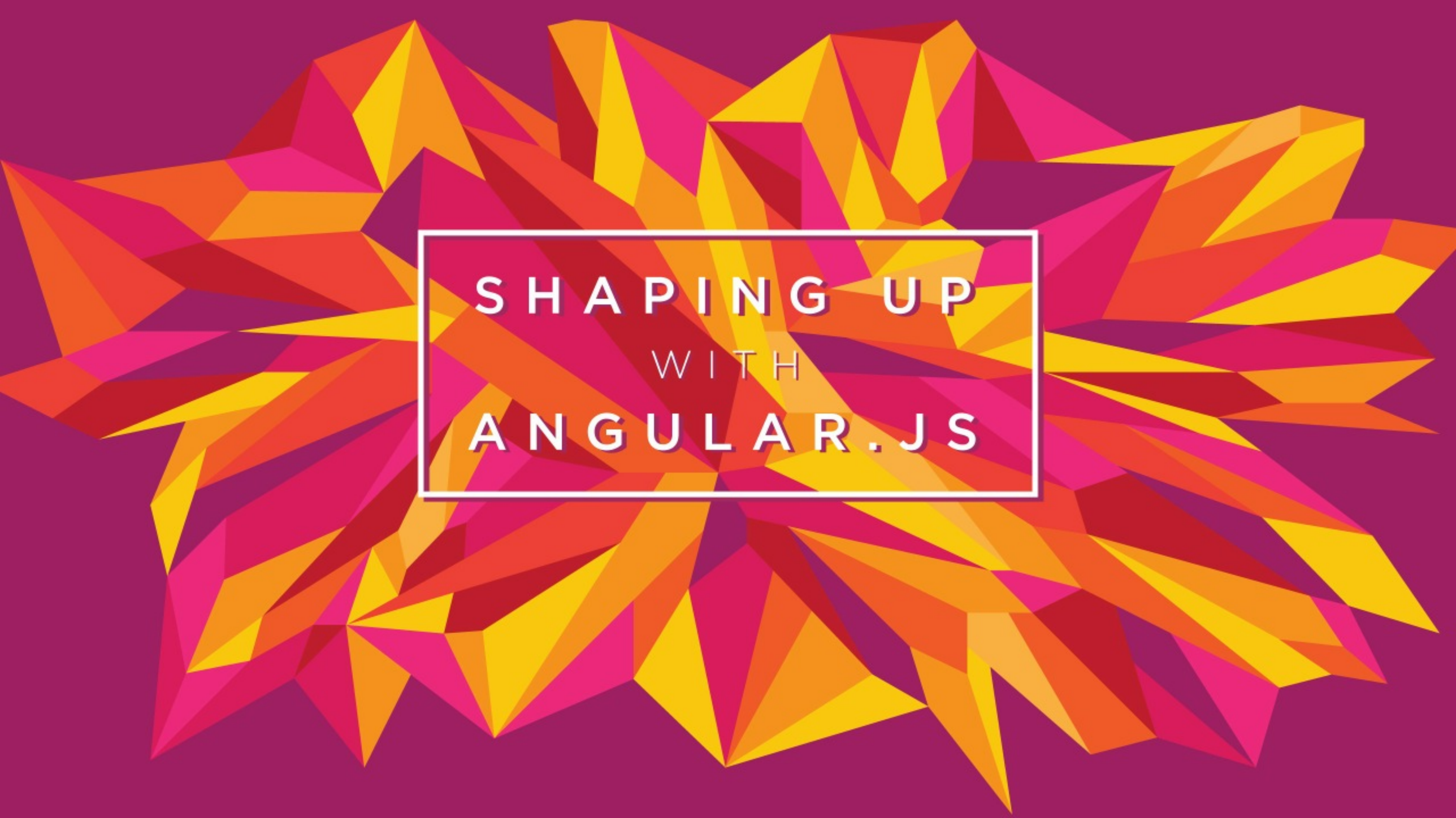
Expressions – How values get displayed within the page

SHAPING UP
WITH
ANGULAR.JS

Challenges

SHAPING UP
WITH
ANGULAR.JS


SHAPING UP
WITH
ANGULAR.JS


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

SHAPING UP
WITH
ANGULAR.JS

Traditional Page-Refresh

Web Server

Web Browser

URL Request to server

Response with Webpage & Assets

HTML JavaScript

Browser loads up entire webpage.

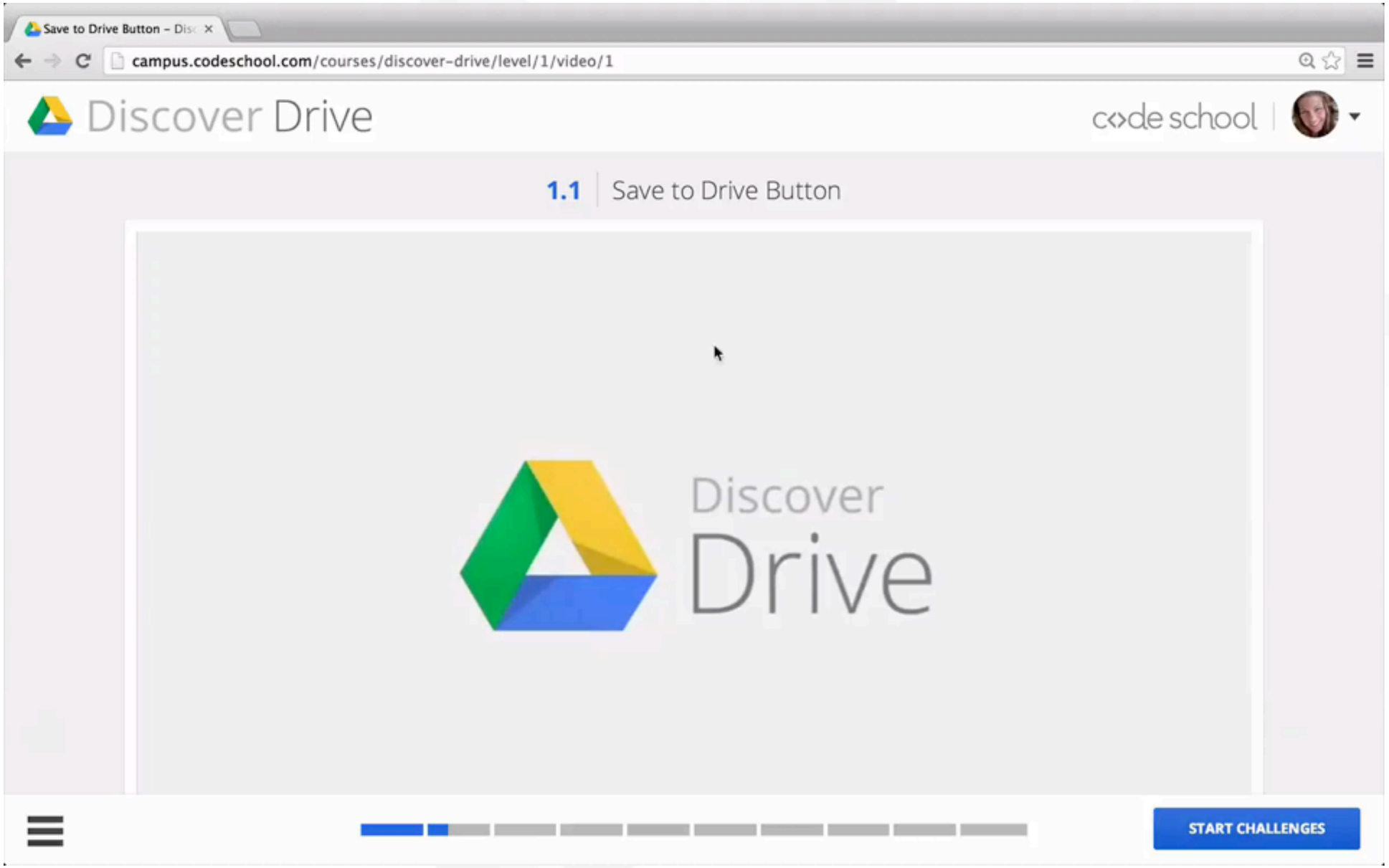
User clicks on link, new Request

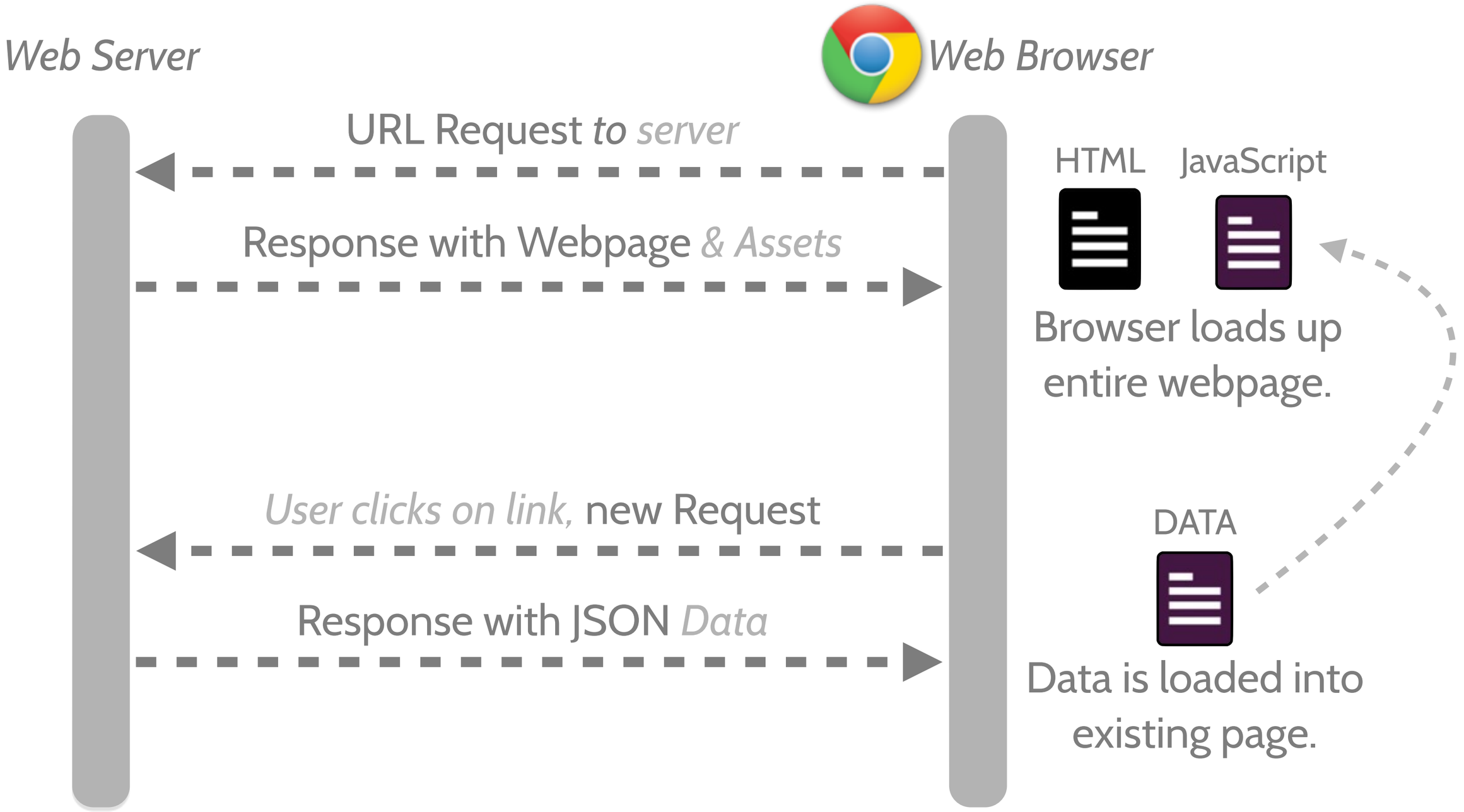
Response with Webpage & Assets

HTML JavaScript

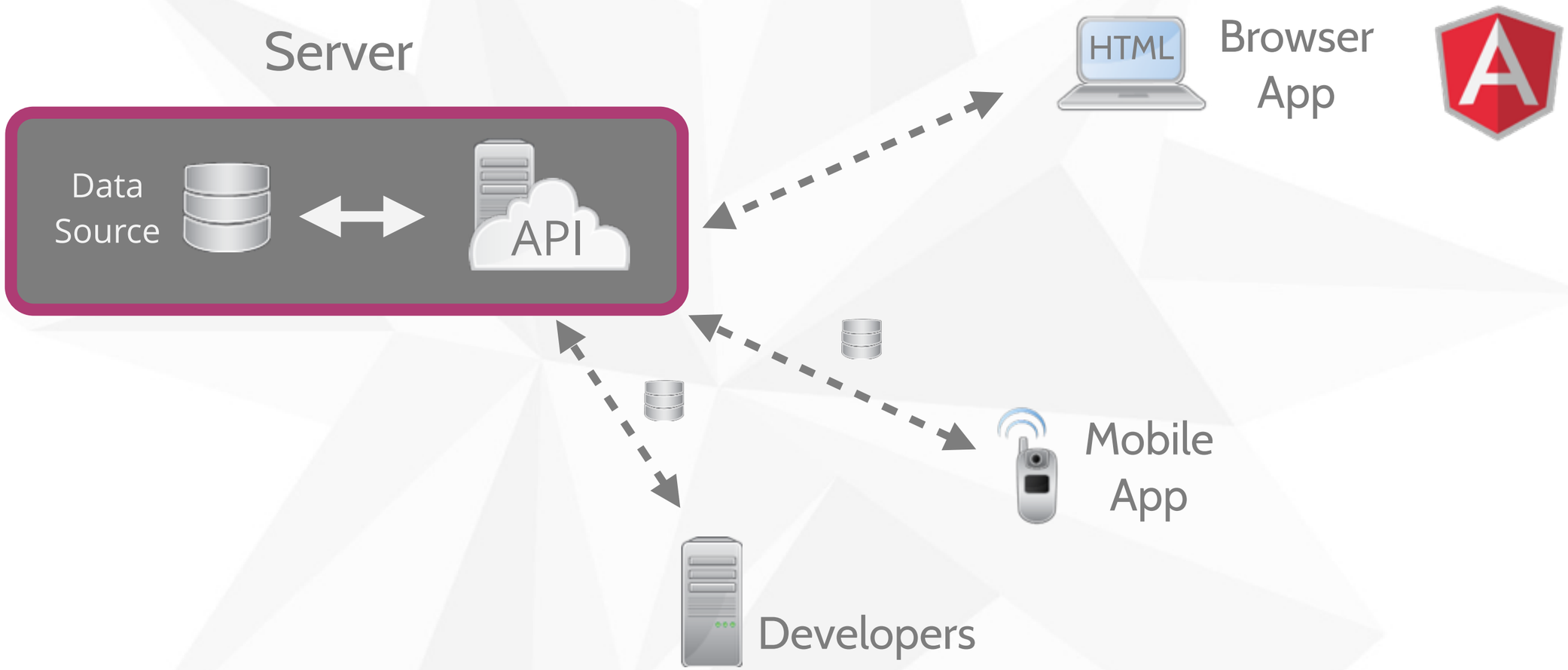
Browser loads up entire webpage.

A “responsive” website using Angular




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

Modern API-Driven Application



What is Angular JS?

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

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

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

index.html

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

app.js

SHAPING UP WITH ANGULAR.JS

Directives

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

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

index.html

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

Name of function to call

app.js

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

Welcome, Gregg!

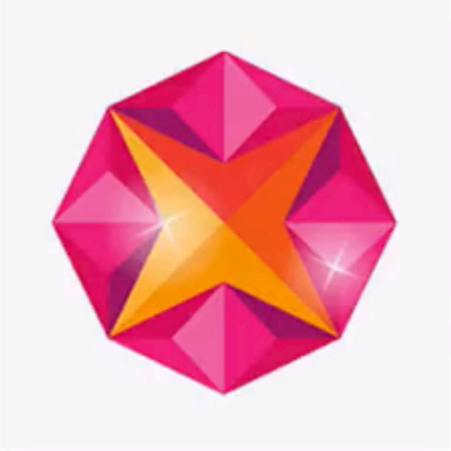
OK

SHAPING UP WITH ANGULAR.JS

Flatlander Crafted Gems

– an Angular store –

Gem #1: Zircon *$1,100.00*



Description Specs Reviews

Description

Downloading the libraries

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

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

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

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

SHAPING UP WITH ANGULAR.JS

Getting Started

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

Twitter Bootstrap

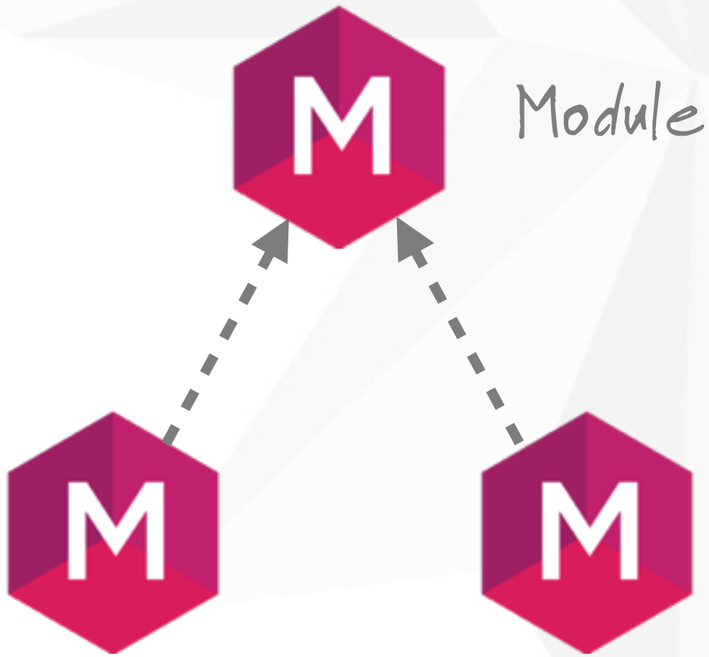
AngularJS

index.html

SHAPING UP WITH ANGULAR.JS

Modules

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



Modules can use other Modules...

SHAPING UP WITH ANGULAR.JS

Creating Our First Module

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

AngularJS

Application Name

Dependencies

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

SHAPING UP WITH ANGULAR.JS

Including Our Module

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

index.html

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

app.js

SHAPING UP WITH ANGULAR.JS

Including Our Module

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

Run this module when the document loads

index.html

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

app.js

SHAPING UP WITH ANGULAR.JS

Expressions

Allow you to insert dynamic values into your HTML.

Numerical Operations

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

evaluates to

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

String Operations

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

evaluates to

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

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

SHAPING UP WITH ANGULAR.JS

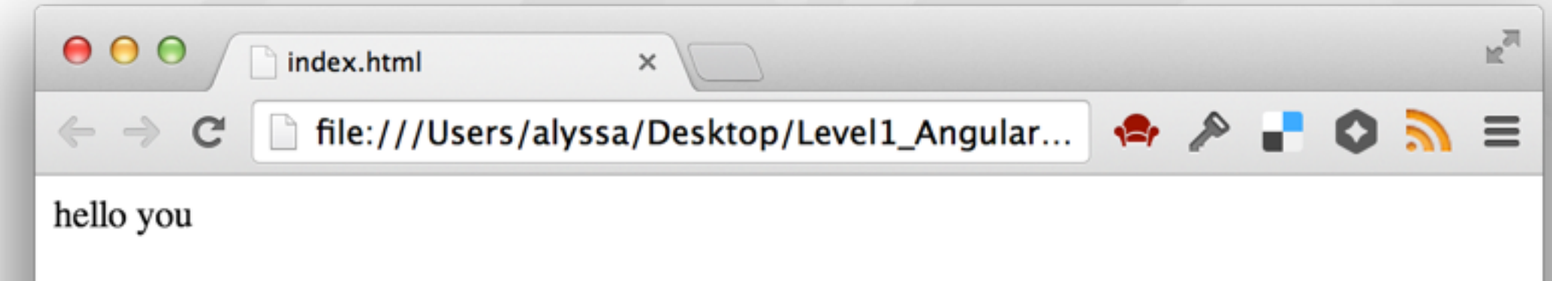
Including Our Module

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

index.html

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

app.js



SHAPING UP WITH ANGULAR.JS

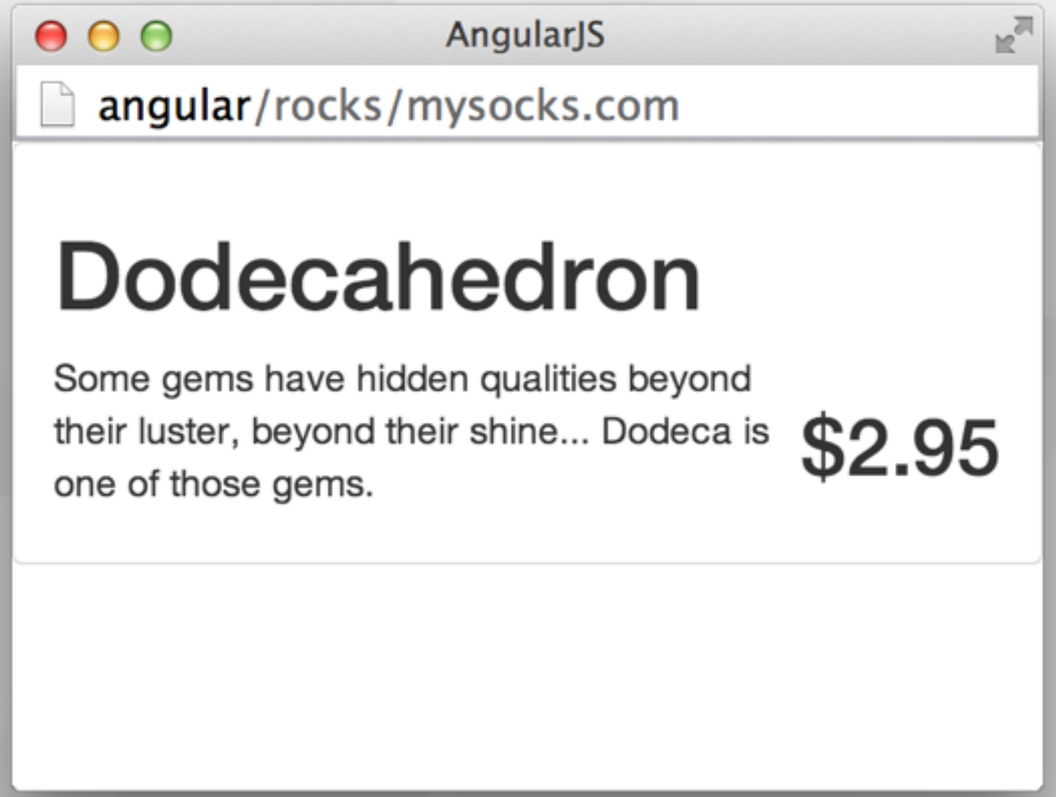
Challenges

SHAPING UP
WITH
ANGULAR.JS

Working With Data

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

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



Controllers

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

Wrapping your Javascript in a closure is a good habit!

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

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

  });
})();
```

app.js

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

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

SHAPING UP WITH ANGULAR.JS

Storing Data Inside the Controller

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

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

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

app.js

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

SHAPING UP WITH ANGULAR.JS

Our Current HTML

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

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

index.html

Attaching the Controller

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

index.html

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

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

app.js

SHAPING UP WITH ANGULAR.JS

Attaching the Controller

Directive Controller name Alias

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

index.html

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

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

app.js

SHAPING UP WITH ANGULAR.JS

Displaying Our First Product

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

index.html

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

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

Understanding Scope

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

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

Would never print a value!

SHAPING UP WITH ANGULAR.JS

Challenges

SHAPING UP
WITH
ANGULAR.JS

Adding A Button

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

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

index.html

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

SHAPING UP WITH ANGULAR.JS

Adding A Button

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

index.html

Directives to the rescue!

How can we only show this button...

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

...when this is true?

SHAPING UP WITH ANGULAR.JS

NgShow Directive

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

index.html

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

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

SHAPING UP WITH ANGULAR.JS

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

their shine... Dodeca i

index.html

NgHide Directive

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

index.html

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

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

SHAPING UP WITH ANGULAR.JS

NgHide Directive

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

index.html

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

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

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

SHAPING UP WITH ANGULAR.JS

NgHide Directive

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

index.html

Much better!

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

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

SHAPING UP WITH ANGULAR.JS

Multiple Products

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


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

app.js

SHAPING UP WITH ANGULAR.JS

Multiple Products

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

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

So we have multiple products...

Now we have an array...

Maybe a Directive?

How might we display all these products in our template?

app.js

Working with An Array

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

index.html

Working with An Array

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

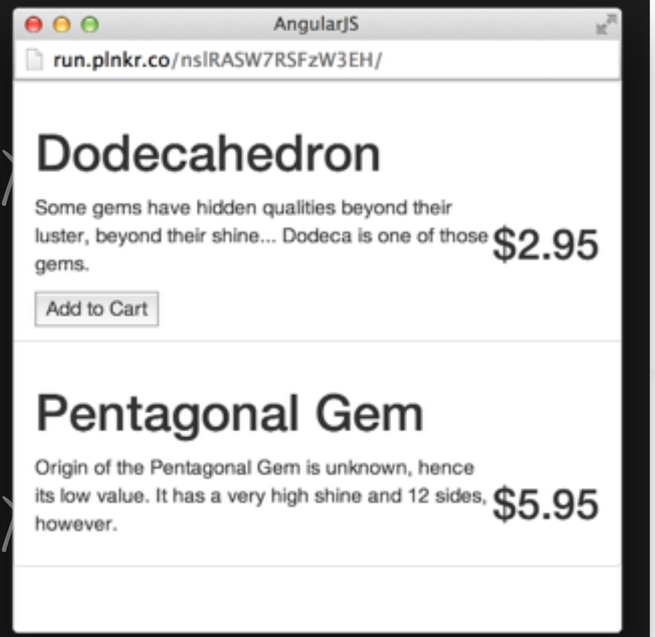
Displaying the first product is easy enough...

index.html

Working with An Array

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

That works...



Why

Why

Why... You get it.

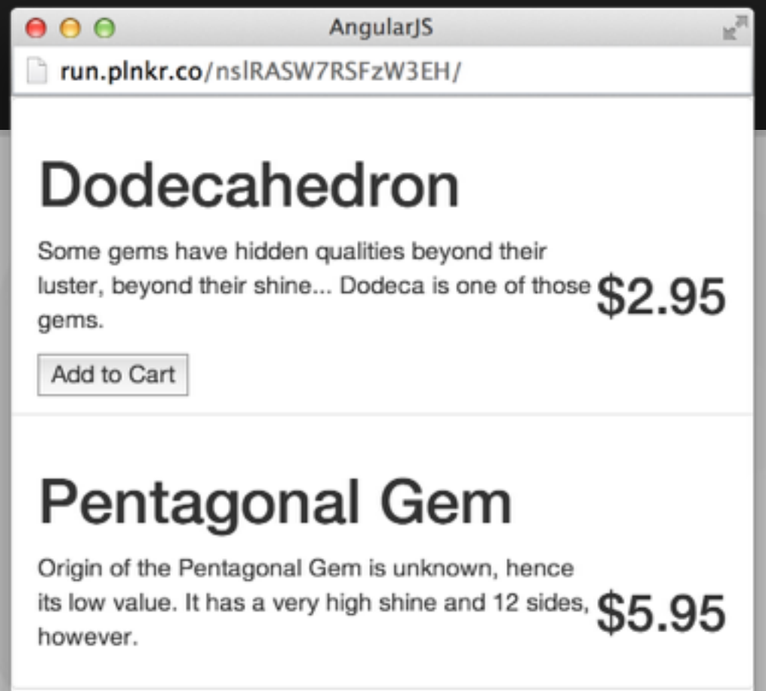
index.html

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

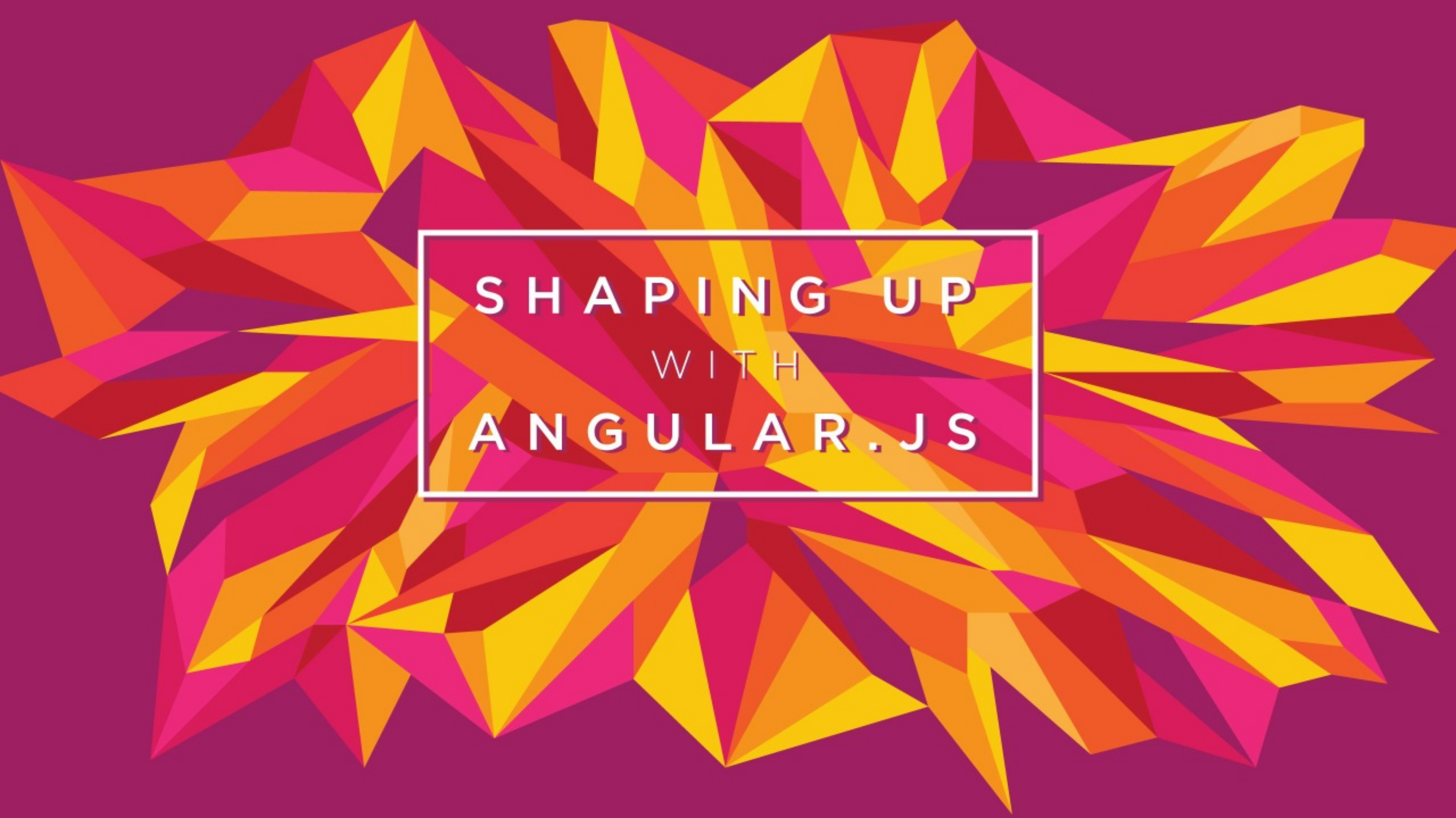
Controllers – Where we add application behavior

Expressions – How values get displayed within the page

SHAPING UP
WITH
ANGULAR.JS

Challenges

SHAPING UP
WITH
ANGULAR.JS


SHAPING UP
WITH
ANGULAR.JS