

# AngularJS

2015/10/05/1회 차

정광윤

# Overview



- [www.angularjs.org](http://www.angularjs.org)
- Stable version: 1.4.7
- 탄생
  - 2009년 구글 직원, Misko Hevery, Adam Abrons 공동 개발
  - 2013년 9월 1.0.8 버전 Release
- MIT license open source
  - <https://github.com/angular/angular.js>
- Web Application Framework
- SPA(Single Page Application) 형태의 웹 애플리케이션을 빠르게 개발할 수 있도록 도와준다.
- referenced by codeschool, w3school

# 맛보기

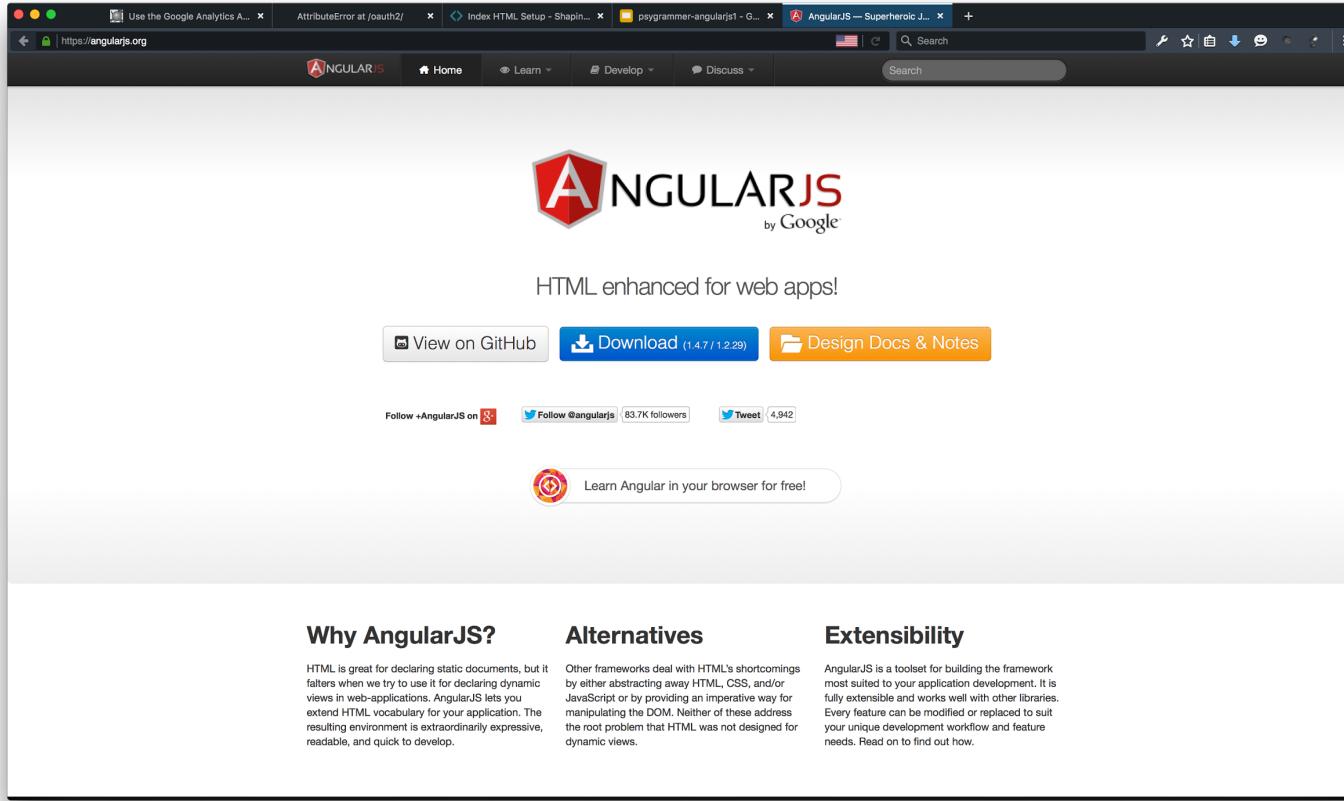
## AngularJS Example

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
<body>

<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>

</body>
</html>
```

Try it Yourself »

A screenshot of a web browser showing the AngularJS homepage. The browser has several tabs open, including "Use the Google Analytics A...", "AttributeError at /oauth2/", "Index HTML Setup - Shapin...", "pysgammer-angularjs1 - G...", and "AngularJS — Superheroic J...". The main content area shows the AngularJS logo, the tagline "HTML enhanced for web apps!", download links for GitHub, npm, and Design Docs & Notes, social media links for Google+ and Twitter, and a button to "Learn Angular in your browser for free!".

The AngularJS homepage features the following content:

- AngularJS by Google**
- HTML enhanced for web apps!**
- [View on GitHub](#)
- [Download \(1.4.7 / 1.2.29\)](#)
- [Design Docs & Notes](#)
- [Follow +AngularJS on Google+](#)
- [Follow @angularjs on Twitter](#) (83.7K followers)
- [Tweet](#) (4.942)
- [Learn Angular in your browser for free!](#)
- Why AngularJS?**

HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web-applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.
- Alternatives**

Other frameworks deal with HTML's shortcomings by either abstracting away HTML, CSS, and/or JavaScript or by providing an imperative way for manipulating the DOM. Neither of these address the root problem that HTML was not designed for dynamic views.
- Extensibility**

AngularJS is a toolkit for building the framework most suited to your application development. It is fully extensible and works well with other libraries. Every feature can be modified or replaced to suit your unique development workflow and feature needs. Read on to find out how.

Learn Angular in your browser for free! - codeschool



# Shaping Up with Angular

Level 1: Getting Started

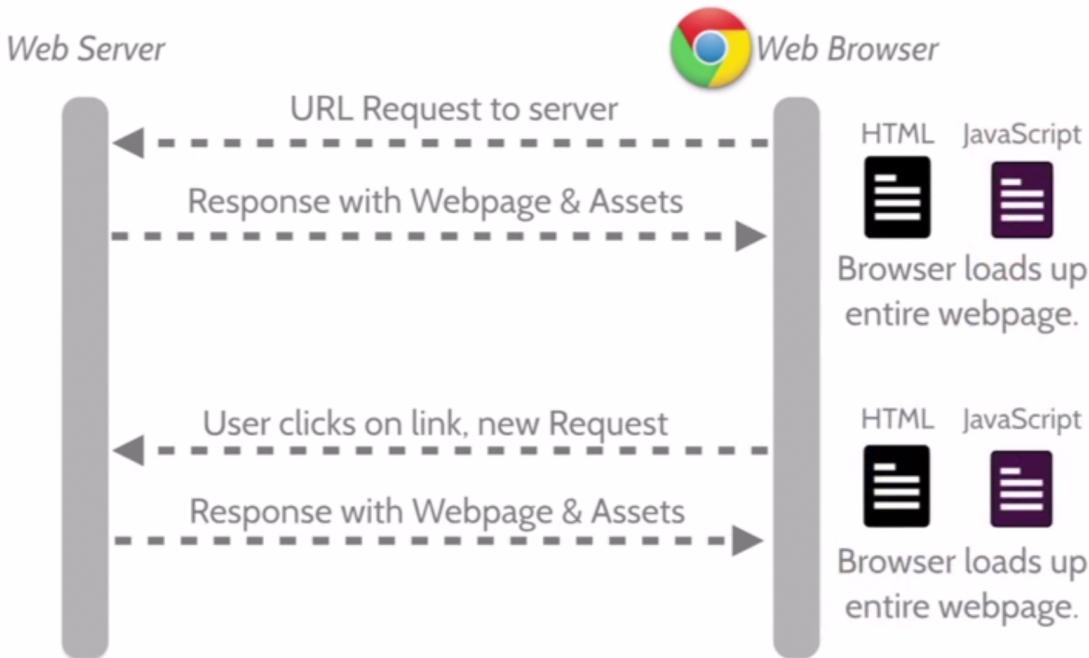
SHAPING UP  
WITH  
ANGULAR.JS

# What is AngularJS?

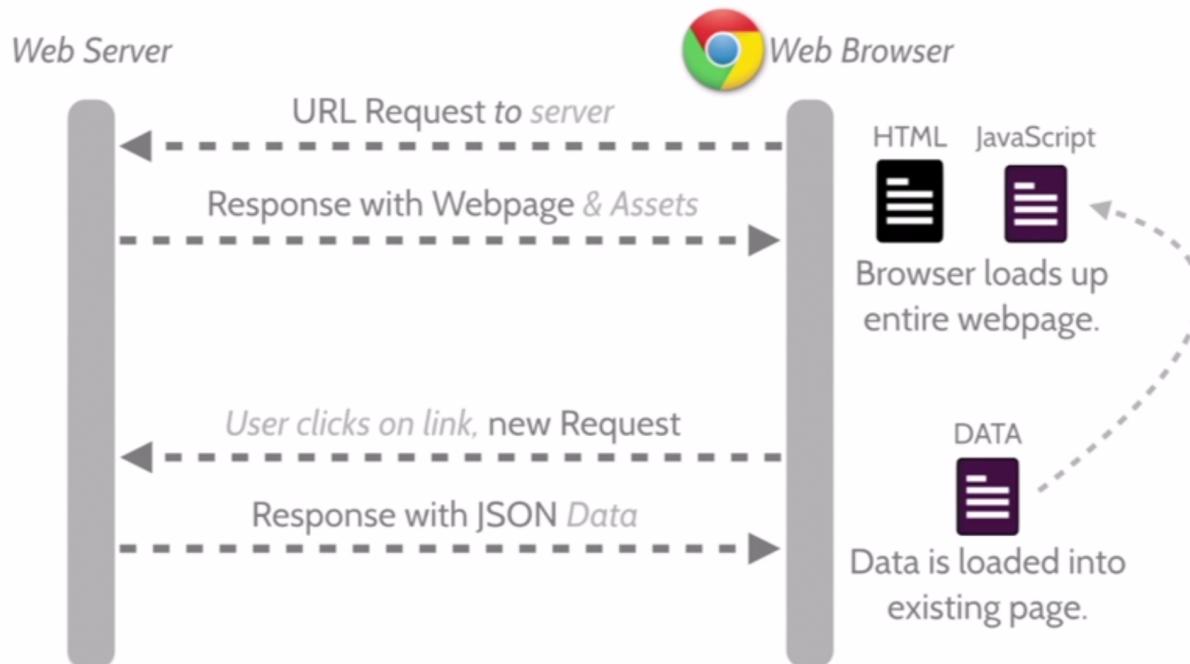
If you're using Javascript to create a dynamic website, Angular is good choice.

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

# Traditional Page-Refresh



# A Responsive website using Angular





# 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

**Directives**  
Modules  
Expressions  
Controllers

# Directives

- ‘**ng-**’ prefix adds directives.
- in HTML tag
- The **ng-app** directive initializes an AngularJS application.
- The **ng-init** directive initializes application data.
- The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.
- The **ng-bind** directive binds application data to the HTML view.
- The **ng-repeat** directive repeats an HTML element.



## 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



# 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

Directives  
**Modules**  
Expressions  
Controllers

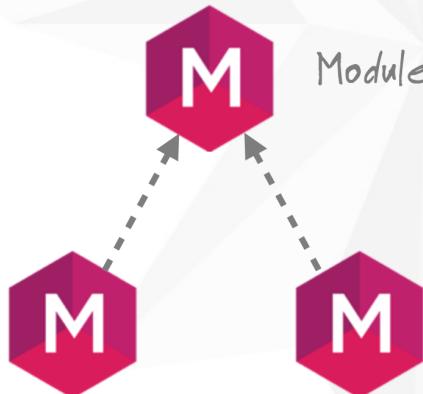
# Modules

- An AngularJS module defines an application.
- The module is a container for the different parts of an application.
- The module is a container for the application controllers.
- Controllers always belong to a module.
  
- It is common in AngularJS applications to put the module and the controllers in JavaScript files.
- In this example, "myApp.js" contains an application module definition, while "myCtrl.js" contains the controller.



## 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...



# Creating Our First Module

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



AngularJS

Application Name

Dependencies

Name

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



# 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"> ←----- Run this module
  <head> when the document
    <link rel="stylesheet" type="text/css" href="bootstrap.min.css" /> loads
  </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

Directives  
Modules  
**Expressions**  
Controllers

# Expressions

- AngularJS expressions are written inside double braces: `{{ expression }}`.
- AngularJS expressions binds data to HTML the same way as the `ng-bind` directive.
- **AngularJS expressions** are much like **JavaScript expressions**: They can contain literals, operators, and variables.
  - Example `{{ 5 + 5 }}` or `{{ firstName + " " + lastName }}`



# 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



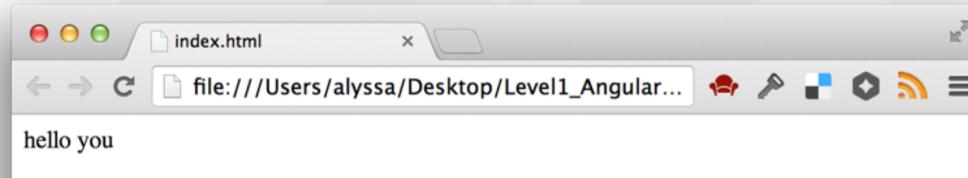
# Including Our Module

```
<!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

Directives  
Modules  
Expressions  
**Controllers**

# Controllers

- AngularJS controllers **control the data** of AngularJS applications.
- AngularJS controllers are regular **JavaScript Objects**.
  - {‘key’: ‘value’, ....}
  - like dictionaries in python



## 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?



# 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



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



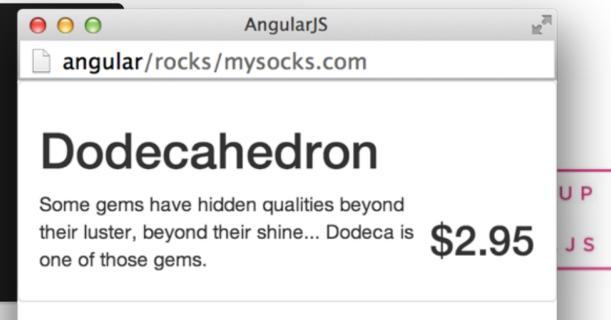
# Displaying Our First Product

```
<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;
  });
  . . .
})();
```





# 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> ←----- Directives to
  </div>          the rescue!
  <script type="text/javascript" src="angular.min.js"></script>
  <script type="text/javascript" src="app.js"></script>
</body>
```

index.html

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



# 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, <----- we want to hide it.
}
```

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

SHAPING UP  
WITH  
ANGULAR.JS



# Multiple Products

```
app.controller('StoreController', function(){
  this.products = gems;
}); So we have multiple products...
var gems = [ <----- Now we have an array...
{
  name: "Dodecahedron",
  price: 2.95,
  description: "...",
  canPurchase: true,
},
{
  name: "Pentagonal Gem",
  price: 5.95,
  description: "...",
  canPurchase: false,
}...
];
```

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 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

Dodecahedron

Some gems have hidden qualities beyond their luster, beyond their shine... Dodeca is one of those **\$2.95** gems.

Add to Cart

Pentagonal Gem

Origin of the Pentagonal Gem is unknown, hence its low value. It has a very high shine and 12 sides, **\$5.95** however.



## What We Have Learned So Far

---



**D** Directives – HTML annotations that trigger Javascript behaviors



**M** Modules – Where our application components live



**C** Controllers – Where we add application behavior



**E** Expressions – How values get displayed within the page

# Example: todo

## Todo

1 of 1 remaining [ [archive](#) ]

- build an angular app

add new todo here

add

# Example1: todo

## Todo

1 of 1 remaining [ [archive](#) ]

- build an angular app

add new todo here

add



# Shaping Up with Angular.JS

Level 2: Filters, Directives, and Cleaner Code

SHAPING UP  
WITH  
ANGULAR.JS



# Shaping Up with Angular

Level 3: Forms, Models, and Validations

SHAPING UP  
WITH  
ANGULAR.JS



# Shaping Up with Angular JS

Level 4: Creating a Directive with an  
Associated Controller

SHAPING UP  
WITH  
ANGULAR.JS



# Shaping Up with Angular JS

Creating Our Own Directives

SHAPING UP  
WITH  
ANGULAR.JS



# Shaping Up with Angular JS

Level 5: Dependencies and Services

SHAPING UP  
WITH  
ANGULAR.JS