AngularJS

2015/10/05/1회 차

정광윤

Overview

- 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 »

Learn Angular in your browser for free! - codeschool

Shaping Up with Angular

Level 1: Getting Started

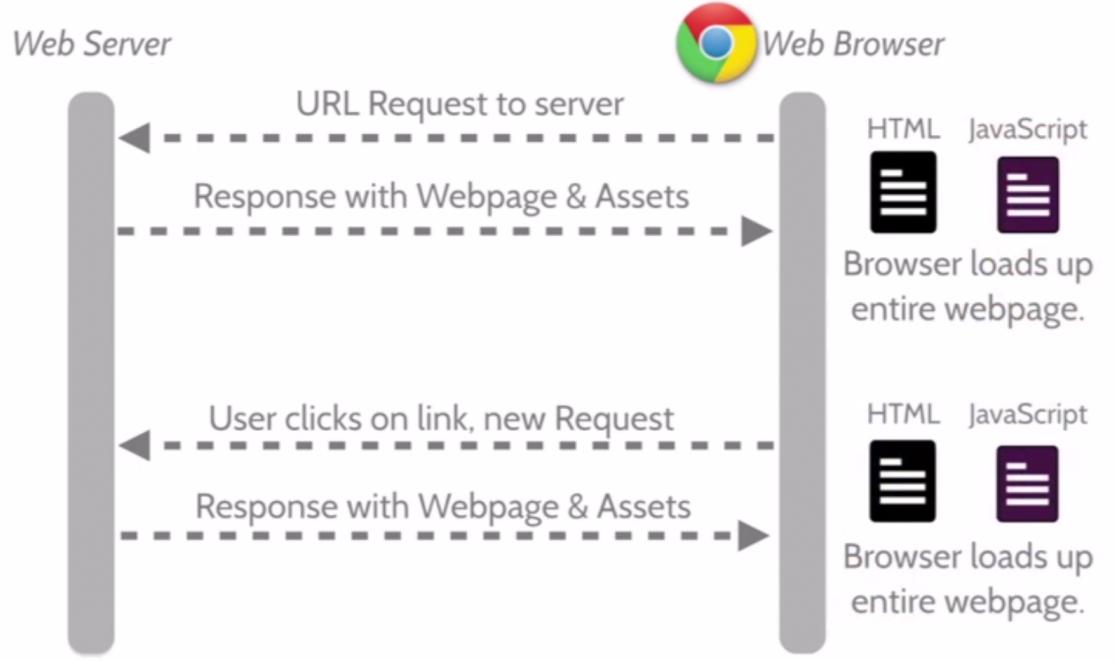
SHAPING UP
WITH
ANGULAR.JS

What is AngularJS?

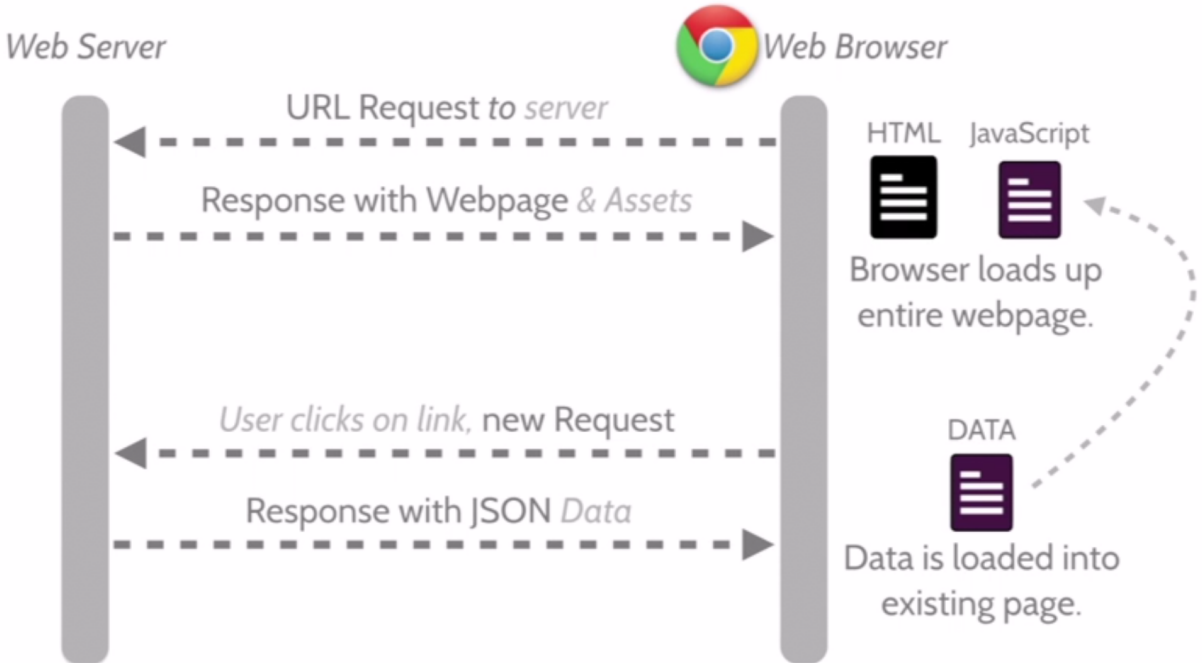
If you’re using Javascript to create a dynamic website, Angular is good choice.

- 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

SHAPING UP WITH ANGULAR.JS

Directives

Modules

Expressions

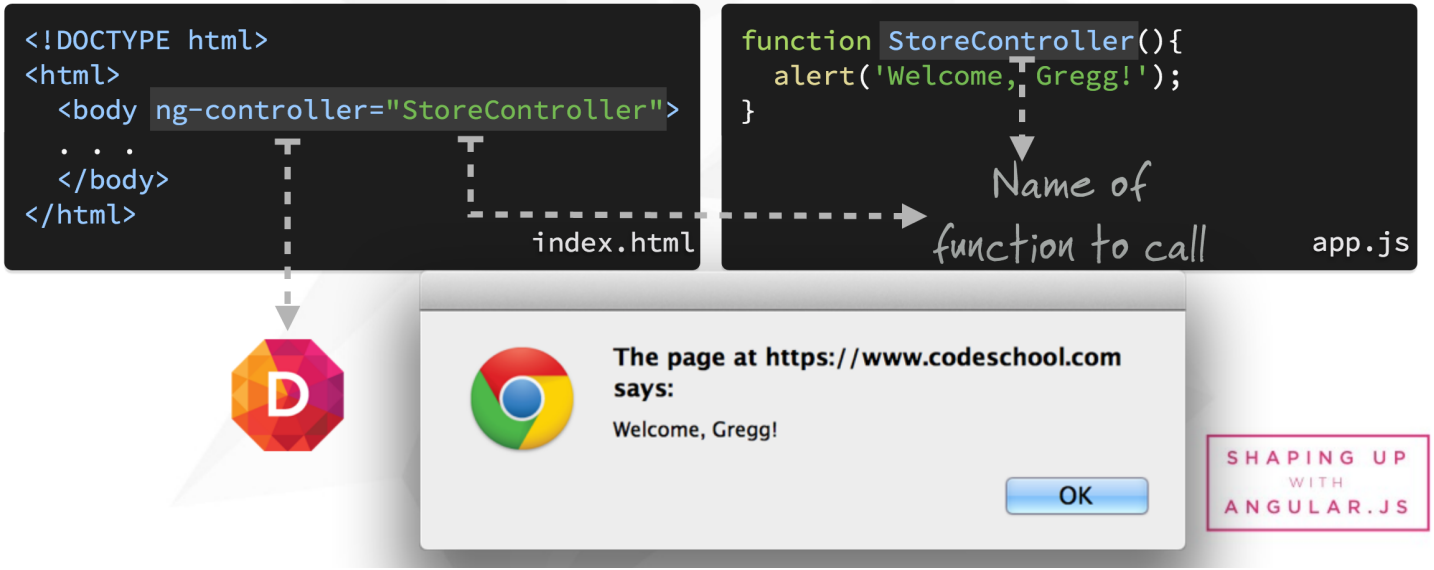
Controllers

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**.



run.plnkr.co/XEnzEl4Zvkfk1vGg/

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

SHAPING UP WITH ANGULAR.JS

Directives

Modules

Expressions

Controllers

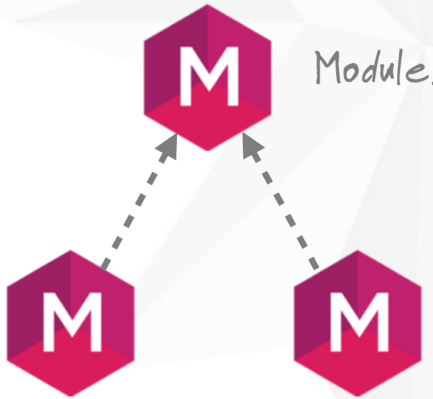
Modules

- 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.



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

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

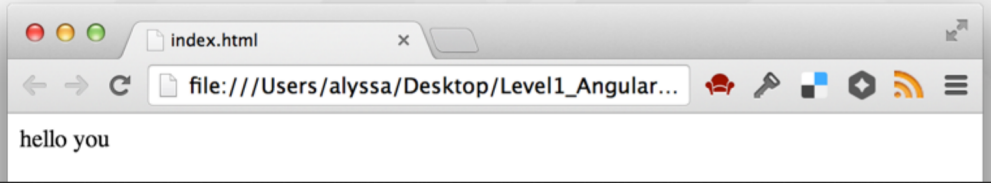
Including Our Module

```
<!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?

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 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>
  </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

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.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>
```

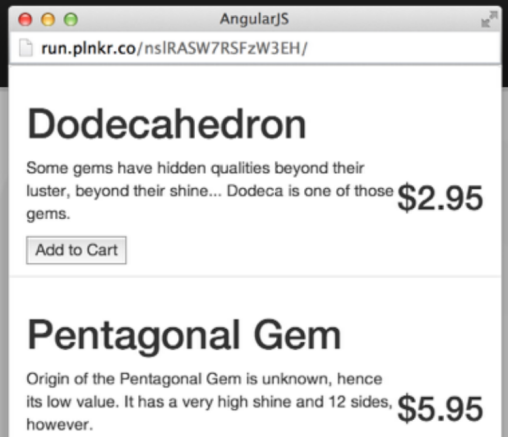
index.html

Working with An Array

```
<body ng-controller="StoreController as store">
  <div 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

Example: todo

Example1: todo

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