CS454 Node.js & Angular.js

Cydney Auman
CSULA

Introduction to Angular - Week 4

What is Angular.js?

AngularJS is a JavaScript MVW frontend framework for creating web applications.

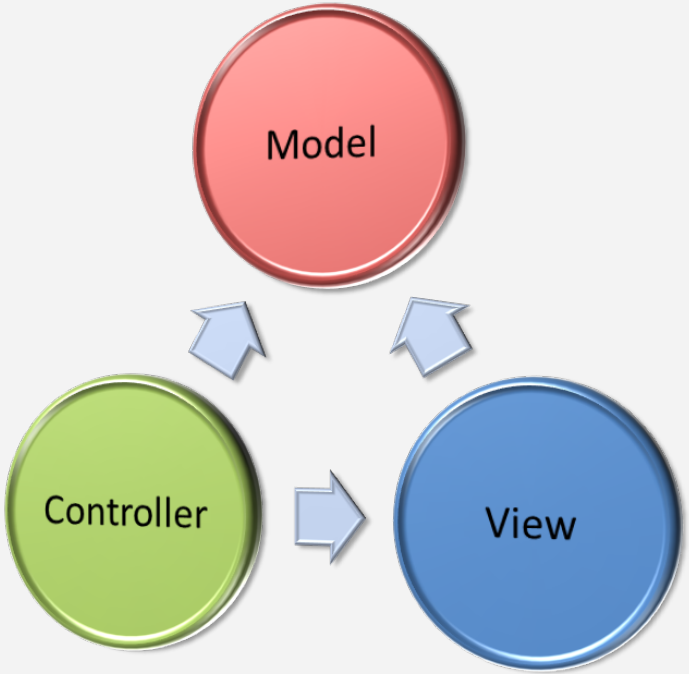
Angular lets you extend HTML attributes with Directives and binds data to HTML.

What IS MVW?

Model

View

Whatever?



Key Angular Concepts

- Directives
- Two Way Data Binding
- Dependency Injection

Directives

Markers on a DOM element (such as an attribute, element name, CSS class) that tell AngularJS's HTML compiler to attach a specified behavior to that DOM element.

In other words - Directives teach HTML new tricks.

Directives

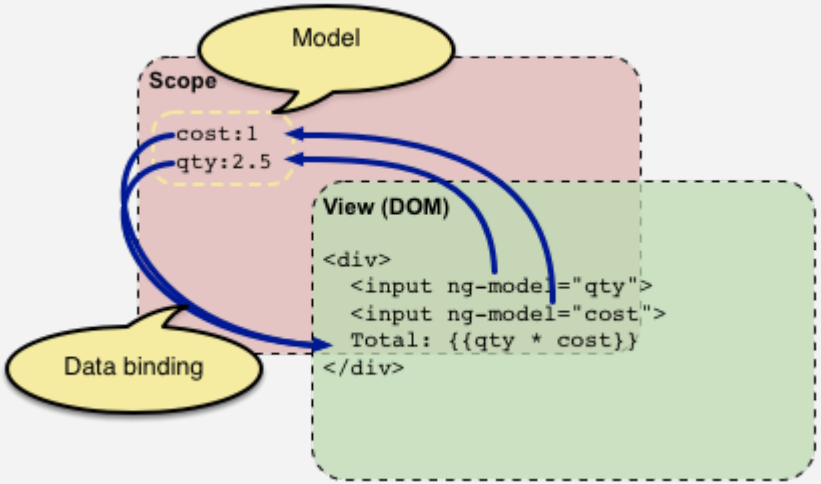
```
<!DOCTYPE html>
<html ng-app>
<head>
  <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.9/angular.min.js"
></script>
</head>
<body>

  <div class="container">
    <input type="text" ng-model="data" />
    <h1>{{data}}</h1>
  </div>

</body>
</html>
```

Two Way Data Binding

Whenever the input values change, the value of the expressions are automatically recalculated and the DOM is updated with their values. The concept behind this is "two-way data binding".



View, Controllers, $scope

View

What the view sees - the HTML side - which is where we use Directives, Filters and Data Binding.

Controller

The business logic behind the views. The controllers are regular JavaScript Objects. The ng-controller directive defines the application controller.

$scope

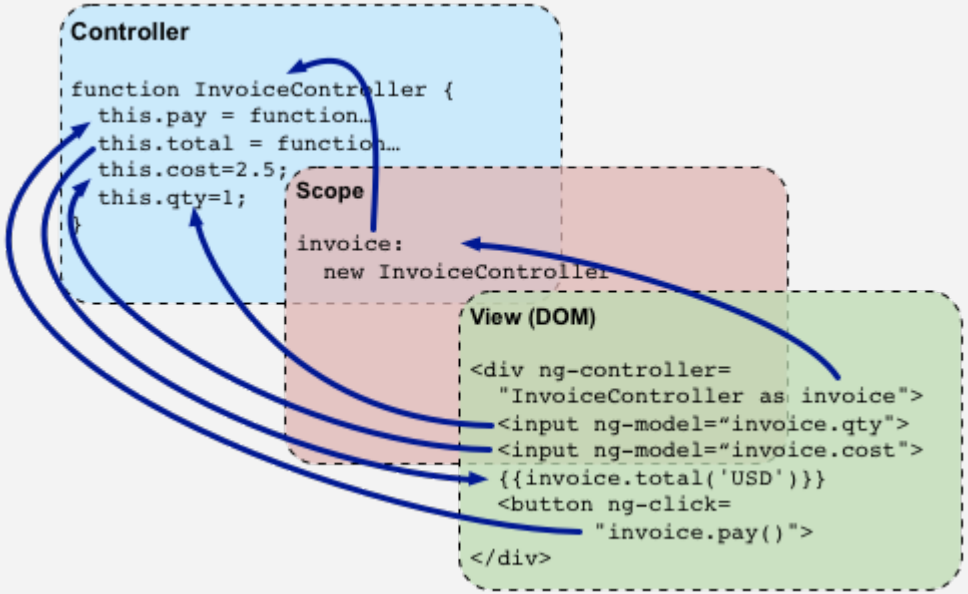
It is context where the model is stored so that controllers, directives and expressions can access it.

Expressions {{ }}

Access variables and functions from the $scope.

Controllers

Controls what data gets bound into the View. If the view passes up data to the controller it will then handle passing data off to a service which then updates a back-end data store.



Controllers

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

var cars = [
  { make: 'Toyota', model: '4 Runner', type: 'Gas', year: '2013' },
  { make: 'Tesla', model: 'Model S', type: 'Electric', year: '2014' },
  { make: 'Subaru', model: 'WRX', type: 'Gas', year: '2014' }
  ];

app.controller("carController", function($scope) {
  $scope.cars = cars;
});
```

Controllers

```
<!DOCTYPE html>
<html ng-app="demoApp">
<head>
  <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.9/angular.min.js">
</script>
  <script type="text/javascript" src="core.js"></script>
</head>
<body class="container">
  <div ng-controller="carController">
      <ul ng-repeat="car in cars">
        <li>{{ car.year }} {{ car.make }} {{ car.model }} {{car.type}}</li>
      </ul>
  </div>
</body>
</html>
```

Factories/Services/Providers

Another feature of AngularJS is the ability to encapsulate data functionality into factory, services, provider or little value providers

So for instance we have to go and fetch data and then we need data in multiple controllers - we wouldn’t want to copy the same function into each controller.

It just wouldn’t make sense and there’d be a lot of duplication there. Instead what we do is move that code to a factory, service or provider.

Factories/Services/Providers

The difference between the three is just the way in which they create the object that goes and gets the data.

- With the factory you actually create an object inside of the factory and return it.
- With the service you just have a standard function that uses the this keyword to define function.
- With the provider there’s a $get you define and it can be used to get the object that returns the data.

Dependency Injection

Dependency Injection is a software design pattern in which an object is given its dependencies, rather than the object creating them itself.

It is about removing the hard-coded dependencies and making it possible to change them whenever needed.