ANGULARJS (level 100)

HTML enhanced for web apps!

Vihang Shah

Associate Project Manager @ Tatvasoft

http://vihang-shah.blogspot.com/

https://www.facebook.com/vihangshah.it

What is ANGULAR?

- It’s not a JavaScript library (As they say). There are no functions which we can directly call and use.
- It is not a DOM manipulation library like jQuery. But it uses subset of jQuery for DOM manipulation (called jqLite).
- Focus more on HTML side of web apps.
- For MVC/MVVM design pattern
- AngularJS is a Javascript MV* framework created by Google to build properly architectured and maintainable web applications.
- AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly.

Conceptual Overview

Concept	Description
Template	HTML with additional markup
Directives	extend HTML with custom attributes and elements
Model	the data shown to the user in the view and with which the user interacts
Scope	context where the model is stored so that controllers, directives and expressions can access it
Expressions	access variables and functions from the scope
Compiler	parses the template and instantiates directives and expressions
Filter	formats the value of an expression for display to the user
View	what the user sees (the DOM)
Data Binding	sync data between the model and the view
Controller	the business logic behind views
Dependency Injection	Creates and wires objects and functions
Injector	dependency injection container
Module	a container for the different parts of an app including controllers, services, filters, directives which configures the Injector
Service	reusable business logic independent of views

Declaration

Angular JS sample declaration

```
<!doctype html>
<html ng-app>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>
  </head>
  <body>
    <div>
      <label>Name:</label>
      <input type="text" ng-model="yourName" placeholder="Enter a name here">
      <hr>
      <h1>Hello {{yourName}}!</h1>
    </div>
  </body>
</html>
```

Why ANGULARJS?

- Defines numerous ways to organize web application at client side.
- Enhances HTML by attaching directives, custom tags, attributes, expressions, templates within HTML.
- Encourage TDD
- Encourage MVC/MVVM design pattern
- Code Reuse
- Good for Single Page Apps (SPA)
- Cool Features -> Next Slide

Key Features of ANGULARJS

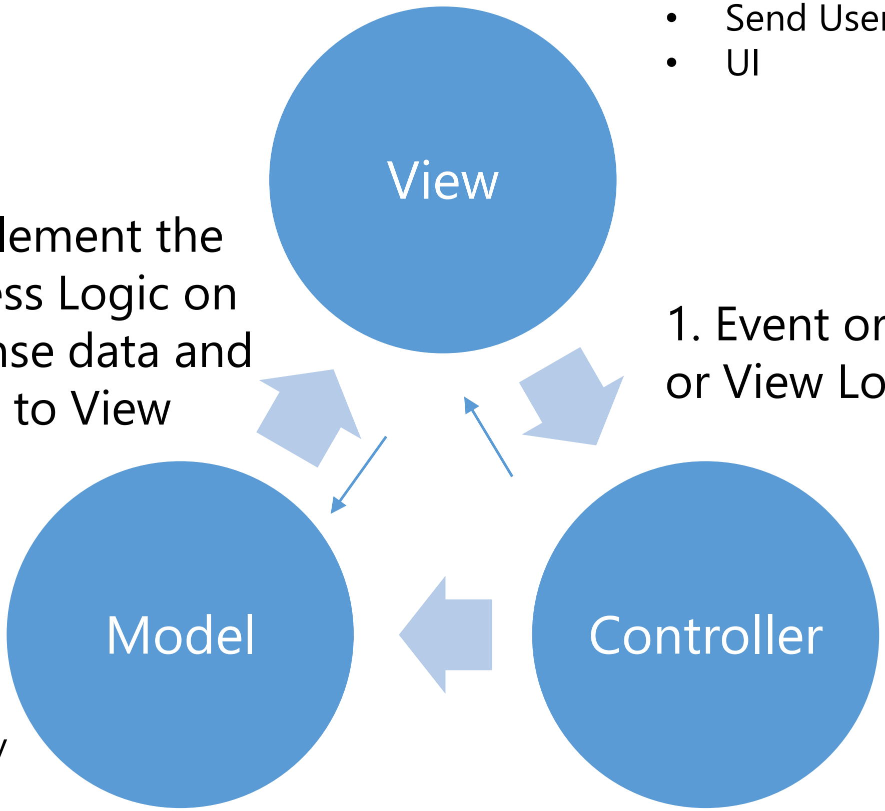
- Declarative HTML approach
- Easy Data Binding : Two way Data Binding
- Reusable Components
- MVC/MVVM Design Pattern
- Dependency Injection
- End to end Integration Testing / Unit Testing
- Routing
- Dynamic Templating

- Modules
- Services
- Expressions
- Filters
- Directives
- Form Validation
- $scope, $http, $routeProvider…

MVC : Model View Controller

View :

- Renders the Model data
- Send User actions/events to controller
- UI



1. Event or User Action or View Load

2. Maps to particular Model after fetching the data

3. Implement the Business Logic on response data and Bind it to View

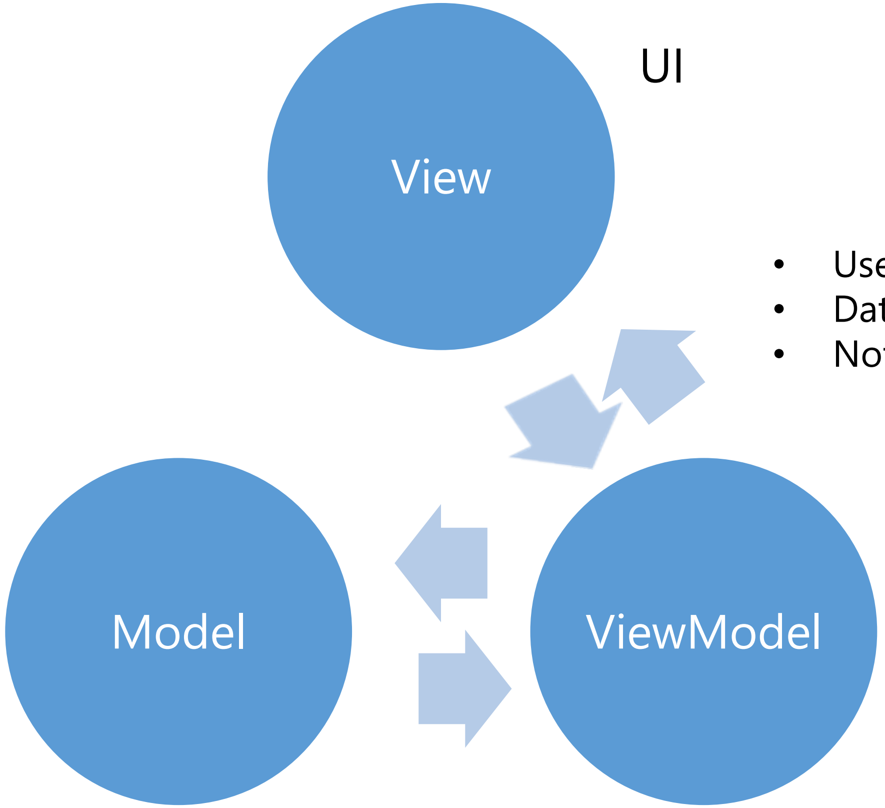
Model:

- Business Logic
- Notify view changes
- Application Functionality
- Data in general

Controller:

- Define Application Behavior
- Maps user actions to Model
- Select view for response

MVVM: Model View ViewModel



UI

- User actions, commands
- Data binding
- Notifications

Business Logic and Data

Presentation Logic

- Data Access
- Update ViewModel about change

ng-app

Use this directive to auto-bootstrap an application.

Only one ng-app directive can be used per HTML document

<html ng-app>

HTML Compiler

Angular's HTML compiler allows the developer to teach the browser new HTML syntax. The compiler allows you to attach behavior to any HTML element or attribute and even create new HTML elements or attributes with custom behavior. Angular calls these behavior extensions directives.

Compiler is an angular service which traverses the DOM looking for attributes. The compilation process happens in two phases.

Compile: traverse the DOM and collect all of the directives. The result is a linking function.

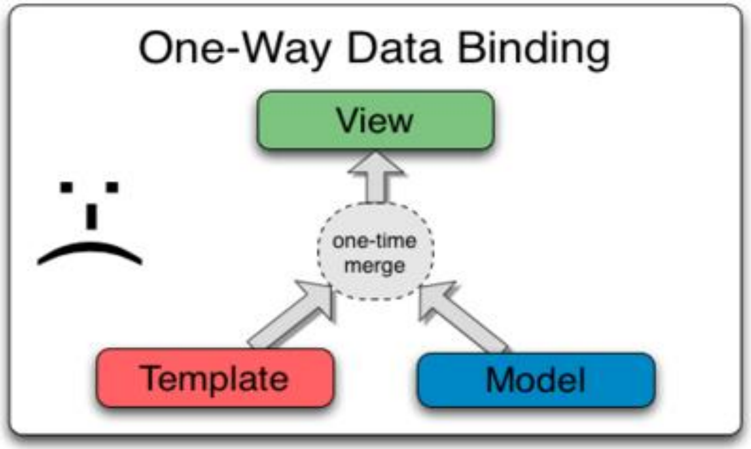
Link: combine the directives with a scope and produce a live view. Any changes in the scope model are reflected in the view, and any user interactions with the view are reflected in the scope model. This makes the scope model the single source of truth.

http://docs.angularjs.org/guide/compiler

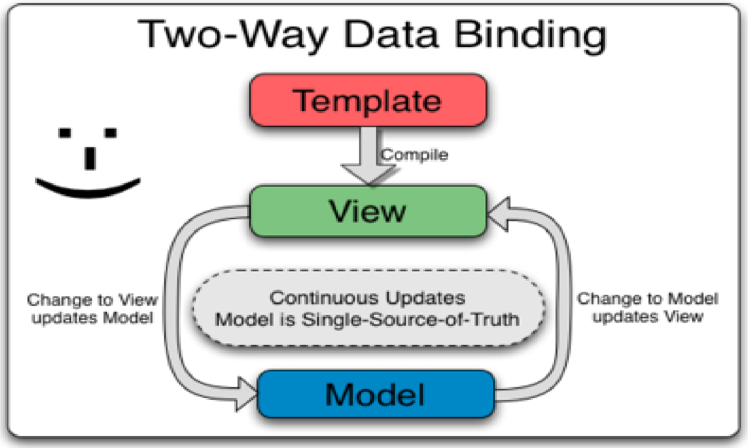
Binding

Data-binding in Angular apps is the automatic synchronization of data between the model and view components. The way that Angular implements data-binding lets you treat the model as the single-source-of-truth in your application. The view is a projection of the model at all times. When the model changes, the view reflects the change, and vice versa.

Data Binding in Classical Template Systems



Data Binding in Angular Templates


Two-Way Data Binding
Template
Compile
View
Change to View updates Model
Continuous Updates
Model is Single-Source-of-Truth
Change to Model updates View
Model

Controllers

AngularJS applications are controlled by controllers. The ng-controller directive defines the application controller. A Controller is defined by a JavaScript constructor function that is used to augment the Angular Scope.

```
<div ng-app="" ng-controller="personController">

        First Name: <input type="text" ng-model="person.firstName"><br />

        Last Name: <input type="text" ng-model="person.lastName"><br /><br />

        Full Name: {{person.firstName + " " + person.lastName}}

    </div>

    <script>function personController($scope) { $scope.person = { firstName: "John", lastName: "Doe" };
}</script>
```

Controllers – Do's and Don'ts

- Use controllers to:
  - Set up the initial state of the $scope object.
  - Add behavior to the $scope object.
- Do not use controllers to:
  - Manipulate DOM — Controllers should contain only business logic. Putting any presentation logic into Controllers significantly affects its testability. Angular has data binding for most cases and directives to encapsulate manual DOM manipulation.
  - Format input — Use angular form controls instead.
  - Filter output — Use angular filters instead.
  - Share code or state across controllers — Use angular services instead.
  - Manage the life-cycle of other components (for example, to create service instances).

Services

- Angular services are substitutable objects that are wired together using dependency injection (DI). You can use services to organize and share code across your app.

  Angular services are:

  - Lazily instantiated – Angular only instantiates a service when an application component depends on it.
  - Singletons – Each component dependent on a service gets a reference to the single instance generated by the service factory.

- Angular offers several useful services (like $http), but for most applications you'll also want to create your own.

- Application developers are free to define their **own services** by registering the service's name and service factory function, with an Angular module.

Services

Many general purpose services provided by AngularJS

- $http

  Used for XMLHttpRequest handling
- $location

  Provide information about the current URL
- $q

  A promise/deferred module for asynchronous requests
- $routeProvider

  Configure routes in an SPA
- $log

  Logging service
- Many more

DEMO

Expressions

- Angular expressions are JavaScript-like code snippets that are mainly placed in interpolation bindings such as `<span title="{{ attrBinding }}">{{ textBinding }}</span>`,

- but also used directly in directive attributes such as `ng-click="functionExpression()"`

- For example, these are valid expressions in Angular:

  - 1+2
  - a+b
  - user.name
  - items[index]

Forms & Validations

Form and controls provide validation services, so that the user can be notified of invalid input. This provides a better user experience, because the user gets instant feedback on how to correct the error.

```
<input type="text" ng-model="user.name" name="uName" required />

<button ng-click="update(user)" ng-disabled="form.$invalid || isunchanged(user)">SAVE</button>
```

Scope

- Scope is an object that refers to the application model.
- It is an execution context for expressions.
- Scopes are arranged in hierarchical structure which mimic the DOM structure of the application.
- Scopes can watch expressions and propagate events.
- Actually the ViewModel of MVVM.
- $scope

```
angular.module('app', [])
     .controller('TestCtrl', function ($scope) {
         console.log($scope)
     })
     .directive('testDirective', function () {
         return {
             link: function (scope) {
                 console.log(scope)
             }
         }
     });
```

DEMO

What is MEAN?

- The acronym "MEAN" stands for "MongoDB Express.js AngularJS Node.js“
- Represents a group of technologies which are known to synergize well together.
- The major benefit of the MEAN stack is that it's extremely quick to prototype with.
  - Node.js allows you to use Javascript on the backend as well as the frontend which can save you from having to learn a separate language.
  - In addition, the NoSQL nature of MongoDB allows you to quickly change and alter the data layer without having to worry about migrations, which is a very valuable attribute when you're trying to build a product without clear specifications.
- These technologies have a lot of community support behind them so finding answers to questions or hiring help is going to be much easier using these technologies.

Resources

Documentation

- AngularJS Developer Guide
- AngularJS Tutorial
- Learn to Build Modern Web Apps with MEAN

Videos

- AngularJS Fundamentals
- AngularJS Fundamentals In 60-ish Minutes

Thank you

Vihang Shah

Associate Project Manager @ Tatvasoft

http://vihang-shah.blogspot.com/

https://www.facebook.com/vihangshah.it