AngularJs Features

* **Directives** – команды, которые можно добавлять в HTML. выглядят как **HTML атрибуты**.
* **MVC** – AngularJs uses **Model-View-Controller** structure for its **framework**. **Model = data**, **View = template system**, that helps to add angular code **to HTML pages**. **Controller** = JavaScript links the data to the **templates**.
* **Data binding** – ability to bind a **modal** and **view** **together**. AngularJs has a unique way of doing this called **two-way data binding**. That means that if something in the template **causes the value of a variable to change**, that something will also **cause the script to do something related to that change**. They’re both tied or **bound together**.
* **Expressions** – AngularJs uses **double curly braces** to create **expressions**. Expressions is how you **output** something **from** the **Controller**, otherwise known as the **JavaScript** into the **HTML view**, which is just a **template**.

AngularJs vs Angular

* **Different paradigm** – Angular is a complete **rewrite of the framework**. Learning Angular is like a learning a completely **different framework**.
* **MVC vs components** – instead of using **Model-View-Controller** architecture is uses a **component-based** architecture, which is an approach that a lot of more modern frameworks like React and Vue use.
* **AngularJs** is more stable. The version of Angular is currently on a schedule that updates to a major revision on every six months and that can be really distractive if have a lot of code to maintain and don’t have time to do major revision of your code very often.
* **Less tooling** – the new versions of Angular requires a lot of tolling in order to work, it takes longer to setup. With AngularJs all you have to do is load up script tags into you HTML. So, it’s as easy as installing jQuery.

AngularJs Binding and directives

* **Binding** - AngularJs lets you create a relationship between data and the rest of your code. And in order to do this it uses something called **directives** – nothing more than a name of command. Directives almost always begins with the ng prefix. And they look like HTML attributes.
* **ng-app** directive **- declare** your **application** or part of the code to treat as the angular application using **ng-app** directive.
* **ng-model** directive – which **creates a variable** in your application or application scope. And it goes inside an HTML form element. The **variable** than **can be accessed within scripts as well as within template**. In AngularJS this is called **two-way data binding**.
* **{{}} – expression –** in order to display data, we can use double curly braces in HTML to create an expression. It’s a way to show variables and other things in HTML.
* **Put <script src=”lib/angular/angular.min.js”></script> in head section,** we need it to load before ng-app section.
* **ng-model** with **{{}} –** we can add **ng-model =**”**query** ”as input attribute and then we can add input value in any place of ng-app using expression {{**query**}}. We can add **JavaScript** code inside it and use with ng-modal variable: {{‘for: ’ + query}}

Modules and controllers

* **Modules** and **controllers** – in order to create better application AngularJS allows you to define something called modules and controllers. **Together they’re going to handle the functionality of application**.
* **module** – is a **container** for different **parts** **of** your **applications**, and **inside** a module, we can **have** **different** **controllers**. In order a module to work you’re going to connect the module to our HTML using **ng-app** **directive**. Once we have a **module** – we can specify one or more **controllers**.
* **Controllers – peace’s of code** that can handle specific functionality **within a module. Application** can have multiple controllers that do different things.
* **ng-controller directive –** in order to specify where a specific controller handles functionality we can use ng-controller directive. **Controller** also can be defined with different **dependencies**, and **dependencies** is just a **name** for something that a controller **needs in order to work**.
* **$scope** – one of the most **common** **ways** to **initialize** **components** is by using a special variable called **scope**. **Scope** – is a **global** **object** that we can use to **communicate** **between** **JavaScript** and our **HTML**. If we **insert** a **variable** in the **scope**, then it means that we can **use** it **within** our **HTML**.

**Example**:

|  |  |
| --- | --- |
| var maApp = angular.module(‘myApp’, [])  myApp.controller(‘MyController’, function MyController($scope){  $scope.artist = {  “name”: “Barot”,  “shortName”: “B\_t”  };  }) | // **creating** a **module** without dependency  // **creating** of **controller**  // **adding** **data** **to a global** var called **scope** |
| <html ng-app=”myApp”>  …  <script src=”lib/angular/angular.min.js”></script>  <script src=”js/app.js></script>  …  <div ng-controller=”MyController”>  <h3>{{ artist.name }}</h3>  <p>{{ artist.name }}</p>  </div>  …  </html> | // **declare** **application** or part of the code to treat as the angular application  // **connect** AngularJS **library**  // **connect** JavaScript **files**  // **define** the **section** **that** **uses** **controller**  // **use data** **added** to the global **scope** inside controller |

* **ns-src** – **directive** that allows as to use **{{}} inside URL**.
  + For example: ng-src=”images/{{artist.shortName}}\_tn.jpg”

Booleans and Loops directives

* **ng-show** and **ng-hide –** they do pretty much the same thing, depending on the value of an expression:
  + **ng-show** will display an element **if the value** of the **expression** is **true**
  + **ng-hide** will hide an element **if the value** of the **expression** is **true**
  + **Both** uses classes **to show or hide element**, which **means** that the element will **still exist in the DOM**.
* **ng-if –** it will **create** an **element** only if the **expression** you give it **is true**. **Main** **difference** is that the element **will not exist at all** **unless** the **expression** **is true**. So, it doesn’t just show or hide an item, it completely **creates** or **destroys** the **element.**
* **ng-repeat –** gives us the ability to loop through arrays or objects. Data is not often going to be as simple as one record, so we can use ng-repeat to loop through items