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**. For example: **ng-repeat=”item in artistsObj”**
* **Injection** – AngularJS will **automatically** **add** **and** **delete** **classes** for you. It’ll inject things in you HTML for you as different things happen inside application.

Other directives

* **ngBind** – replace the text content of the specified HTML element with the value of a given expression
* **ngBlur, ngFocus** – Specify custom behavior on blur/focus event
* **ngChange** – Evaluate the given expression when the user changes the input
* **ngChecked** – solves this problem for the checked attribute. Can add ng-modal variable to it.
* **ngClass** – example: ngClass=”expression ? ‘some-class : ‘other-class ”
* **ngClick, ngDblclick** – allows you to specify custom behavior when an element is clicked
* **ngCopy**, **ngCut**, **ngPaste** – Specify custom behavior on copy/cut/paste event.
* **ngDisabled** – sets the disabled attribute on the element if the expression inside is true.

Using AngularJS services

* We can use provided by **AngularJs** **services** or create our own.
* **$http** – one of the most **important AngularJs services**. Allows to handle **Ajax requests** to **load** files like **JSON** documents. Works only with a running server. Remember that is a request to a file doesn’t normally stop the browser from doing other things. This is why we call Ajax asynchronous. JavaScript uses promises which means that when the request is ready, we can execute some code afterwards.

**Example:**

|  |  |
| --- | --- |
| myApp.controller(‘MyController’, function MyController($scope, $http) {  $http.get(‘js/data.json’).then(function(response) {  $scope.artists = response.data;  })  } | // adding $http service  // making request and collect data from response, and then add it to the scope. |

* **$documen**t – A jQuery or jqLite wrapper for the browser's window.document object
* **$location –** service parses the URL in the browser address bar (based on the window.location) and makes the URL available to your application
* **$q –** service that helps you run functions asynchronously and use their return values (or exceptions) when they are done processing.

function asyncGreet(name) {

// perform some asynchronous operation, resolve, or reject the promise when appropriate.

return $q(function(resolve, reject) {

setTimeout(function() {

if (okToGreet(name)) {

resolve('Hello, ' + name + '!');

} else {

reject('Greeting ' + name + ' is not allowed.');

}

}, 1000);

});

}

* **$rootElement** – the root element of Angular application. This is the element where **ngApp** was declared.
* **$rootScope** – every application has a single root scope. All other scopes are descendant(потомок) scopes of the root scope.
* **$exceptionHandler** – any uncaught exception in angular expressions is delegated to this service
* **$timeout** – Angular's wrapper for **window.setTimeout**. The fn function is wrapped into a try/catch block and delegates any exceptions to **$exceptionHandler** service.
* **$window -** a reference to the browser's **window** **object**. While **window** is **globally** **available** in JavaScript, it **causes** **testability** **problems**, because it is a global variable. In angular we always **refer** **to** **it** through the **$window** service, so it may be **overridden**, **removed**, or **mocked** for **testing**.

Providers

* **$anchorScrollProvider** – Use **$anchorScrollProvider** to disable automatic scrolling whenever **$location.hash()** changes.

Basic Filters

* Simple filters:
* **currency, number, date, lowercase, uppercase** - list of basic **filters**, that we can use in AngularJS.
* **currency –** simple filter for formatting currency. Have options for different symbols and decimal places.
* **number –** allow to control decimal places in number
* **date –** is a really nice way to format dates.
* **lowercase and uppercase –** it’s understandable.
* **Use | and : characters** inside **{{}}** to apply any of the filters to variable. **Example**: {{item.name | **uppercase** }}
* Advanced filters:
* It’s a different group of filters that allow to work with arrays. We can create live search using it.
* **limitTo:qty:start** – good filter for **pagination**. Specify the number of elements that you want to show and then when you should start showing those elements. **Example** **ng-repeat=”item in artistsObj | limitTo:4:1”-**will show **four elements** starting **from the second one**.
* **filter**:**keyword** – allow to specify a **keyword** that will narrow the list of elements that show up in an array based on a certain keyword. We can use it to create search. **Example** **ng-repeat=”item in artistsObj | filter: query” -** will show only items close to search query. **So, it’s a live search.**
* **orderBy**:key:reverse – allows to control the order of the array, based on a key, and also whether or not this is in normal or reverse order. **Example** **ng-repeat=” item in artistsObj | orderBy: ‘name’ ”** will **order elements** in list **by name.**
* We can **combine** several **filters** by **dividing** them with **|** symbol**.**

Deep Linking (Routing)

* To create more complex applications, we can split our code into different sections through something called deep linking.
* **Deep linking** – is a process of making different URLs load up different content. And this content can be managed by different code.
* **ngRoute** – deep linking is handled by it. This a different module that we’ll have to load into our application using a different JavaScript file.
* **$routeProvider** – once we load a file, we can configure our application through this. Here we can specify what’s going to be loaded by which URLs, and which peace of JavaScript code is going to control the different content.
* **ng**-**view** – loaded module works with this special directive. That’s going to load appropriate code depending on URL.
* **Example**:

|  |  |
| --- | --- |
| App |  |
|  | // creating App and adding **‘ngToute’** and **controllers** to dependency array  // configuration the routing system  // template for this URL  // controller for this template  // adding parameters (:itemId) |
| Controllers |  |
|  | // declaration of controllers  // using **$http** as a fetch method for data  // using **$routeParams** for collecting URL parameters  // collecting itemId |