**AngularJS Intro**

AngularJS Directives

* AngularJS is a JavaScript framework. It can be added to an HTML page with a <script> tag.
* AngularJS extends HTML attributes ( <h1>, <a>, … ) with Directives (Dastorol amal) , and binds data to HTML with Expressions. (ebarat ha).
* AngularJS extends HTML with ng-directives:
* The ng-app directive defines an AngularJS application.
* The ng-model directive binds (vasl kardan) the value of HTML controls (input, select, textarea) to application data.
  + Simply it gave data from back end and show it in HTML file
* **The ng-bind** directive binds (vasl kardan) application data to the HTML view.
  + Simply it gave data from HTML and give it to the back end for processing on that data.
* The **ng-init**directive initializes AngularJS application variables.
* As you have already seen, AngularJS directives are HTML attributes with an ng prefix.

## AngularJS Expressions

* AngularJS expressions are written inside double braces: {{ expression }}.
* AngularJS will "output" data exactly where the expression is written.

AngularJS Applications

* AngularJS **modules** define AngularJS applications.
* AngularJS **controllers** control AngularJS applications.
* The **ng-app** directive defines the application, the **ng-controller** directive defines the controller.

Read more about this at:

<https://www.w3schools.com/angular/angular_intro.asp>

**AngularJS Expressions**

AngularJS binds data to HTML using Expressions.

AngularJS Expressions

* There are two ways to write expressions in AngualrJS:
  + AngularJS expressions can be written inside double braces: {{ expression }}.
  + AngularJS expressions can also be written inside a directive:

ng-bind="expression".

* **AngularJS expressions** are much like **JavaScript expressions:** They can contain literals, operators, and variables.
* Example {{ 5 + 5 }} or {{ firstName + " " + lastName }}
* If you remove the ng-app directive, HTML will display the expression as it is, without solving it. (very important)
* Some examples: ng-init="myCol=' lightblue' “ , ng-init="quantity=1; cost=5" .

AngularJS Objects, Arrays

See example: <https://www.w3schools.com/angular/angular_expressions.asp>

AngularJS Expressions vs. JavaScript Expressions

* Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.
* Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.
* AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.
* AngularJS expressions support filters, while JavaScript expressions do not.

**AngularJS 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.

Creating a Module

* A module is created by using the AngularJS function angular.module
* The "myApp" parameter refers to an HTML element in which the application will run.
* Now you can add controllers, directives, filters, and more, to your AngularJS application.

Adding a Controller

* Add a controller to your application, and refer to the controller with the

ng-controller directive:

Adding a Directive

* AngularJS has a set of built-in directives which you can use to add functionality to your application.
* For a full reference, visit our [AngularJS directive reference](https://www.w3schools.com/angular/angular_ref_directives.asp).

Modules and Controllers in Files

* 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:
* Global functions should be avoided in JavaScript. They can easily be overwritten or destroyed by other scripts.
* AngularJS modules reduces this problem, by keeping all functions local to the module.

When to Load the Library

* While it is common in HTML applications to place scripts at the end of the <body> element, it is recommended that you load the AngularJS library either in the <head> or at the start of the <body>.
* This is because calls to angular.module can only be compiled after the library has been loaded.

**AngularJS Directives**

* AngularJS lets you extend HTML with new attributes called Directives.
* AngularJS has a set of built-in directives which offers functionality to your applications.
* AngularJS also lets you define your own directives.
* AngularJS directives are extended HTML attributes with the prefix ng-.
* 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.

Repeating HTML Elements

* The ng-repeat directive repeats an HTML element.
* The ng-repeat directive used on an array of objects:

The ng-model Directive

The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

The ng-model directive can also:

* Provide type validation for application data (number, email, required).
* Provide status for application data (invalid, dirty, touched, error).
* Provide CSS classes for HTML elements.
* Bind HTML elements to HTML forms.

## Create New Directives

* In addition to all the built-in AngularJS directives, you can create your own directives.
* New directives are created by using the .directive function.
* To invoke the new directive, make an HTML element with the same tag name as the new directive.
* When naming a directive, you must use a camel case name, w3TestDirective, but when invoking it, you must use - separated name, w3-test-directive:
* You can invoke a directive by using:
  + Element name
  + Attribute
  + Class
  + Comment

**AngularJS ng-model Directive**

* The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.
* Simply it gave data from back end and show it in HTML file.
* The ng-model directive adds/removes the following classes, according to the status of the form field:
  + ng-empty
  + ng-not-empty
  + ng-touched
  + ng-untouched
  + ng-valid
  + ng-invalid
  + ng-dirty
  + ng-pending
  + ng-pristine

**AngularJS Data Binding**

* Data binding in AngularJS is the synchronization between the model and the view.
* The HTML container where the AngularJS application is displayed, is called the view.
* Because of the immediate synchronization of the model and the view, the controller can be completely separated from the view, and simply concentrate on the model data. Thanks to the data binding in AngularJS, the view will reflect any changes made in the controller.

**AngularJS Controllers**

* AngularJS controllers **control the data** of AngularJS applications.
* AngularJS controllers are regular **JavaScript Objects**.

Controllers In External Files

* See example in : <https://www.w3schools.com/angular/angular_controllers.asp>

**AngularJS Scope**

* The scope is the binding part between the HTML (view) and the JavaScript (controller).
* The scope is an object with the available properties and methods.
* The scope is available for both the view and the controller.
* When adding properties to the $scope object in the controller, the view (HTML) gets access to these properties.
* In the view, you do not use the prefix $scope, you just refer to a propertyname, like {{carname}}.

## Root Scope

All applications have a $rootScope which is the scope created on the HTML element that contains the ng-app directive.

The rootScope is available in the entire application.

If a variable has the same name in both the current scope and in the rootScope, the application use the one in the current scope.

**AngularJS Filters**

AngularJS provides filters to transform data:

* currency Format a number to a currency format.
* date Format a date to a specified format.
* filter Select a subset of items from an array.
* json Format an object to a JSON string.
* limitTo Limits an array/string, into a specified number of elements/characters.
* lowercase Format a string to lower case.
* number Format a number to a string.
* orderBy Orders an array by an expression.
* uppercase Format a string to upper case.

## Adding Filters to Expressions

Filters can be added to expressions by using the pipe character |, followed by a filter.

The uppercase filter format strings to upper case

## Adding Filters to Directives

Filters are added to directives, like ng-repeat, by using the pipe character |, followed by a filter

## The currency Filter

The currency filter formats a number as currency:

Read more here: <https://www.w3schools.com/angular/ng_filter_currency.asp>

## The filter Filter

The filter filter selects a subset of an array.

The filter filter can only be used on arrays, and it returns an array containing only the matching items.

## Filter an Array Based on User Input

By setting the ng-model directive on an input field, we can use the value of the input field as an expression in a filter.

## Sort an Array Based on User Input

## Custom Filters

You can make your own filters by registering a new filter factory function with your module:

For more intfomation see: <https://www.w3schools.com/angular/angular_filters.asp>

**AngularJS Services**

## What is a Service?

In AngularJS, a service is a function, or object, that is available for, and limited to, your AngularJS application.

AngularJS has about 30 built-in services. One of them is the $location service.

The $location service has methods which return information about the location of the current web page:

## Why use Services?

For many services, like the $location service, it seems like you could use objects that are already in the DOM, like the window.location object, and you could, but it would have some limitations, at least for your AngularJS application.

Read more : <https://www.w3schools.com/angular/angular_services.asp>

**AngularJS AJAX - $http**

**$http** is an AngularJS service for reading data from remote servers.

## AngularJS $http

The AngularJS $http service makes a request to the server, and returns a response.

Methods

The example above uses the .get method of the $http service.

The .get method is a shortcut method of the $http service. There are several shortcut methods:

* .delete()
* .get()
* .head()
* .jsonp()
* .patch()
* .post()
* .put()

Properties

The response from the server is an object with these properties:

* .config the object used to generate the request.
* .data a string, or an object, carrying the response from the server.
* .headers a function to use to get header information.
* .status a number defining the HTTP status.
* .statusText a string defining the HTTP status.

To handle errors, add one more functions to the .then method:

See exapme in: <https://www.w3schools.com/angular/angular_http.asp>

## JSON

The data you get from the response is expected to be in JSON format.

JSON is a great way of transporting data, and it is easy to use within AngularJS, or any other JavaScript.

Example: On the server we have a file that returns a JSON object containing 15 customers, all wrapped in array called records.

See the example: <https://www.w3schools.com/angular/angular_http.asp>

Application explained:

The application defines the customersCtrl controller, with a $scope and $http object.

$http is an **XMLHttpRequest object** for requesting external data.

$http.get() reads **JSON data** from <https://www.w3schools.com/angular/customers.php>.

On success, the controller creates a property, myData, in the scope, with JSON data from the server.

## Displaying Data in a Table

Displaying tables with angular is very simple:

See page: <https://www.w3schools.com/angular/angular_tables.asp>

## AngularJS Select Boxes

## Creating a Select Box Using ng-options

If you want to create a dropdown list, based on an object or an array in AngularJS, you should use the ng-options directive

## ng-options vs ng-repeat

You can also use the ng-repeat directive to make the same dropdown list

Because the ng-repeat directive repeats a block of HTML code for each item in an array, it can be used to create options in a dropdown list, but the ng-options directive was made especially for filling a dropdown list with options, and has at least one important advantage:

Dropdowns made with ng-options allows the selected value to be an **object**, while dropdowns made from ng-repeat has to be a string.

Read more: <https://www.w3schools.com/angular/angular_select.asp>

## AngularJS SQL

AngularJS is perfect for displaying data from a Database. Just make sure the data is in JSON format.

Server Code Examples

The following section is a listing of the server code used to fetch SQL data.

1. Using PHP and MySQL. Returning JSON.
2. Using PHP and MS Access. Returning JSON.
3. Using ASP.NET, VB, and MS Access. Returning JSON.
4. Using ASP.NET, Razor, and SQL Lite. Returning JSON.

## AngularJS HTML DOM

AngularJS has directives for binding application data to the attributes of HTML DOM elements.

The DOM is the way **Javascript** sees its containing pages' data. It is an object that includes how the HTML/XHTML/XML is formatted, as well as the browser state. A DOM element is something like a **DIV**, HTML, **BODY element** on a page. You can add classes to all of these using CSS, or interact with them using JS.

## The ng-disabled Directive

The **ng-disabled** directive binds AngularJS application data to the disabled attribute of HTML elements.

## The ng-show Directive

The **ng-show** directive shows or hides an HTML element.

The ng-show directive shows (or hides) an HTML element based on the **value** of ng-show.

You can use any expression that evaluates to true or false:

## The ng-hide Directive

The **ng-hide** directive hides or shows an HTML element.

## AngularJS Events

AngularJS has its own HTML events directives.

AngularJS Events

You can add AngularJS event listeners to your HTML elements by using one or more of these directives:

* ng-blur
* ng-change
* ng-click
* ng-copy
* ng-cut
* ng-dblclick
* ng-focus
* ng-keydown
* ng-keypress
* ng-keyup
* ng-mousedown
* ng-mouseenter
* ng-mouseleave
* ng-mousemove
* ng-mouseover
* ng-mouseup
* ng-paste

The event directives allows us to run AngularJS functions at certain user events.

An AngularJS event will not overwrite an HTML event, both events will be executed.

Mouse Events

Mouse events occur when the cursor moves over an element, in this order:

1. ng-mouseenter
2. ng-mouseover
3. ng-mousemove
4. ng-mouseleave

Or when a mouse button is clicked on an element, in this order:

1. ng-mousedown
2. ng-mouseup
3. ng-click

You can add mouse events on any HTML element.

## Toggle, True/False

If you want to show a section of HTML code when a button is clicked, and hide when the button is clicked again, like a dropdown menu, make the button behave like a toggle switch:

## AngularJS Forms

Forms in AngularJS provides data-binding and validation of input controls.

Input Controls

Input controls are the HTML input elements:

* input elements
* select elements
* button elements
* textarea elements

see example here: <https://www.w3schools.com/angular/angular_forms.asp>

## AngularJS Form Validation

## Form Validation

AngularJS offers client-side form validation.

AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.

AngularJS also holds information about whether they have been touched, or modified, or not.

You can use standard HTML5 attributes to validate input, or you can make your own validation functions.

## Required

Use the HTML5 attribute required to specify that the input field must be filled out:

## E-mail

Use the HTML5 type email to specify that the value must be an e-mail:

Form State and Input State

AngularJS is constantly updating the state of both the form and the input fields.

Input fields have the following states:

* $untouched The field has not been touched yet
* $touched The field has been touched
* $pristine The field has not been modified yet
* $dirty The field has been modified
* $invalid The field content is not valid
* $valid The field content is valid

They are all properties of the input field, and are either true or false.

Forms have the following states:

* $pristine No fields have been modified yet
* $dirty One or more have been modified
* $invalid The form content is not valid
* $valid The form content is valid
* $submitted The form is submitted

They are all properties of the form, and are either true or false.

You can use these states to show meaningful messages to the user. Example, if a field is required, and the user leaves it blank, you should give the user a warning:

CSS Classes

AngularJS adds CSS classes to forms and input fields depending on their states.

The following classes are added to, or removed from, input fields:

* ng-untouched The field has not been touched yet
* ng-touched The field has been touched
* ng-pristine The field has not been  modified yet
* ng-dirty The field has been modified
* ng-valid The field content is valid
* ng-invalid The field content is not valid
* ng-valid-*key* One *key* for each validation. Example: ng-valid-required, useful when there are more than one thing that must be validated
* ng-invalid-*key* Example: ng-invalid-required

The following classes are added to, or removed from, forms:

* ng-pristine No fields has not been modified yet
* ng-dirty One or more fields has been modified
* ng-valid The form content is valid
* ng-invalid The form content is not valid
* ng-valid-*key* One *key* for each validation. Example: ng-valid-required, useful when there are more than one thing that must be validated
* ng-invalid-*key* Example: ng-invalid-required

The classes are removed if the value they represent is false.

Add styles for these classes to give your application a better and more intuitive user interface.

## AngularJS API

API stands for **A**pplication **P**rogramming **I**nterface.

AngularJS Global API

The AngularJS Global API is a set of global JavaScript functions for performing common tasks like:

* Comparing objects
* Iterating objects
* Converting data

The Global API functions are accessed using the angular object.

Below is a list of some common API functions:

|  |  |
| --- | --- |
| **API** | **Description** |
| angular.lowercase() | Converts a string to lowercase |
| angular.uppercase() | Converts a string to uppercase |
| angular.isString() | Returns true if the reference is a string |
| angular.isNumber() | Returns true if the reference is a number |

## AngularJS Includes