**ANGULAR JS**

* AngularJS is a **JavaScript framework**. It can be added to an HTML page with a <script> tag.
* AngularJS extends HTML with new attributes.
* AngularJS extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**.
* AngularJS is a JavaScript framework written in JavaScript.
* AngularJS is perfect for Single Page Applications (SPAs).
* AngularJS is easy to learn.

First, you will learn the basics of AngularJS: directives, expressions, filters, modules, and controllers.

Then AngularJS: Events, DOM, Forms, Input, Validation, Http, and more.

**AngularJS Extends HTML**

AngularJS extends HTML with **ng-directives**.

* The **ng-app** directive defines an AngularJS application.
* The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.
* The **ng-bind** directive binds application data to the HTML view.

# AngularJS Expressions

AngularJS binds data to HTML using **Expressions**.

{{ expression }}.

# 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.

# 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 ng-model Directive

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

<div ng-app="myApp" ng-controller="myCtrl">  
  Name: <input ng-model="name">  
</div>  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
  $scope.name = "John Doe";  
});  
</script>

## Two-Way Binding

The binding goes both ways. If the user changes the value inside the input field, the AngularJS property will also change its value:

<div ng-app="myApp" ng-controller="myCtrl">  
  Name: <input ng-model="name">  
  <h1>You entered: {{name}}</h1>  
</div>

## Validate User Input

The ng-model directive can provide type validation for application data (number, e-mail, required):

<form ng-app="" name="myForm">  
  Email:  
  <input type="email" name="myAddress" ng-model="text">  
  <span ng-show="myForm.myAddress.$error.email">Not a valid e-mail address</span>  
</form>

## Application Status

The ng-model directive can provide status for application data (valid, dirty, touched, error):

Ex-1:

<form ng-app="" name="myForm" ng-init="myText = 'post@myweb.com'">  
  Email:  
  <input type="email" name="myAddress" ng-model="myText" required>  
  <h1>Status</h1>  
  {{myForm.myAddress.$valid}}  
  {{myForm.myAddress.$dirty}}  
  {{myForm.myAddress.$touched}}  
</form>

Ex-2:

<style>

input.ng-invalid {  
  background-color: lightblue;  
}

</style>  
<body>  
<form ng-app="" name="myForm">  
  Enter your name:  
  <input name="myName" ng-model="myText" required>  
</form>

</body>

Note: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.

## Data Model

AngularJS applications usually have a data model. The data model is a collection of data available for the application.

<p ng-bind="firstname"></p>

<input ng-model="firstname">

<p>First name: {{lastname}}</p>

<script>

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
  $scope.firstname = "John";  
  $scope.lastname = "Doe";  
});

</script>

## Two-way Binding

Data binding in AngularJS is the synchronization between the model and the view.

When data in the model changes, the view reflects the change, and when data in the view changes, the model is updated as well. This happens immediately and automatically, which makes sure that the model and the view is updated at all times.

## AngularJS Controllers

* AngularJS controllers **control the data** of AngularJS applications.
* AngularJS controllers are regular **JavaScript Objects**.
* AngularJS applications are controlled by controllers.
* The **ng-controller** directive defines the application controller.
* A controller is a **JavaScript Object**, created by a standard JavaScript **object constructor**.

<div ng-app="myApp" ng-controller="myCtrl">  
First Name: <input type="text" ng-model="firstName"><br>  
Last Name: <input type="text" ng-model="lastName"><br>  
<br>  
Full Name: {{firstName + " " + lastName}}  
</div>  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  $scope.firstName = "John";  $scope.lastName = "Doe";});  
</script>

# 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.

<div ng-app="myApp" ng-controller="myCtrl">  
<h1>{{carname}}</h1>  
</div>  
<script>

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
  $scope.carname = "Volvo";  
});  
</script>

# AngularJS Filters

Filters can be added in AngularJS to format data.

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

AngularJS Services(<https://www.w3schools.com/angular/angular_http.asp>)

In AngularJS you can make your own service, or use one of the many built-in 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.

## 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.

## $http Service

## $timeout Service

## The $interval Service

## Create Your Own Service

app.service('hexafy', function() {  
  this.myFunc = function (x) {  
    return x.toString(16);  
  }  
});

app.controller('myCtrl', function($scope, **hexafy**) {  
  $scope.hex = **hexafy**.myFunc(255);  
});

# AngularJS AJAX - $http

* **$http** is an AngularJS service for reading data from remote servers.
* The AngularJS $http service makes a request to the server, and returns a response.

<div ng-app="myApp" ng-controller="myCtrl">  
<p>Today's welcome message is:</p>  
<h1>{{myWelcome}}</h1>  
</div>  
<script>

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
  $http.get("welcome.htm")  
  .then(function(response) {  
    $scope.myWelcome = response.data;  
  });  
});

</script>

**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()

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
  $http({  
    method : "GET",      url : "welcome.htm"  }).then(function mySuccess(response) {    $scope.myWelcome = response.data;  }, function myError(response) {    $scope.myWelcome = response.statusText;  });  
});

$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.

# AngularJS SQL (<https://www.w3schools.com/angular/angular_sql.asp>)

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

## Fetching Data From a PHP Server Running MySQL

<div ng-app="myApp" ng-controller="customersCtrl">  
<table>  
  <tr ng-repeat="x in names">  
    <td>{{ x.Name }}</td>  
    <td>{{ x.Country }}</td>  
  </tr>  
</table>  
</div>  
  
<script>  
var app = angular.module('myApp', []);  
app.controller('customersCtrl', function($scope, $http) {  
  $http.get("customers\_mysql.php")  
  .then(function (response) {$scope.names = response.data.records;});  
});  
</script>

# AngularJS HTML DOM

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

## The ng-disabled Directive

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

# 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.

|  |  |
| --- | --- |
| **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 |