AngularJS Tutorial

AngularJS is a very powerful JavaScript library. It is used in Single Page Application (SPA) projects. It extends HTML DOM with additional attributes and makes it more responsive to user actions. AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0.

What is AngularJS?

AngularJS is an open source web application framework. It was originally developed in 2009 by Misko Hevery and Adam Abrons. It is now maintained by Google. Its latest version is 1.3.14.

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 and succinctly. Angular’s data binding and dependency injection eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner with any server technology.

Features

AngularJS is a powerful JavaScript based development framework to create RICH Internet Application (RIA).

AngulajJS provides developers options to write client side application (using JavaScript) in a clean MVC (Model View Controller) way.

Application written in AngularJS is cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.

AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache License version 2.0.

Core Features

Following are most important core features of AngularJS:

Data-binding: It is the automatic synchronization of data between model and view components.

Scope: These are objects that refer to the model. They act as a glue between controller and view.

Controller: These are JavaScript functions that are bound to a particular scope.

Services: AngularJS come with several built-in services for example $http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.

Filters: These select a subset of items from an array and returns a new array.

Directives: Directives are markers on DOM elements (such as elements, attributes, css, and more). These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives (ngBind, ngModel...)

Templates: These are the rendered view with information from the controller and model. These can be a single file (like index.html) or multiple views in one page using "partials".

Routing: It is concept of switching views.

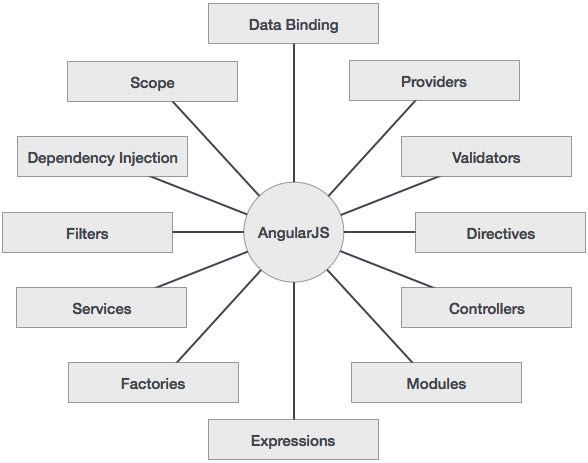
Model View Whatever: MVC is a design pattern for dividing an application into different parts (called Model, View and Controller), each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-View-Model). The Angular JS team refers it humorously as Model View Whatever.

Deep Linking: Deep linking allows you to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.

Dependency Injection: AngularJS has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand, and test.

Concepts

Following diagram depicts some important parts of AngularJS which we will discuss in detail in the subsequent chapters.



Advantages of AngularJS

AngularJS provides capability to create Single Page Application in a very clean and maintainable way.

AngularJS provides data binding capability to HTML thus giving user a rich and responsive experience

AngularJS code is unit testable.

AngularJS uses dependency injection and make use of separation of concerns.

AngularJS provides reusable components.

With AngularJS, developer write less code and get more functionality.

In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.

On top of everything, AngularJS applications can run on all major browsers and smart phones including Android and iOS based phones/tablets.

Disadvantages of AngulajJS

Though AngularJS comes with lots of plus points but same time we should consider the following points:

Not Secure: Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.

Not degradable: If your application user disables JavaScript then user will just see the basic page and nothing more.

AngularJS Directives

|  |  |
| --- | --- |
| **Directive** | **Description** |
| [ng-app](http://www.w3schools.com/angular/ng_ng-app.asp) | Defines the root element of an application. |
| [ng-bind](http://www.w3schools.com/angular/ng_ng-bind.asp) | Binds the content of an HTML element to application data. |
| [ng-bind-html](http://www.w3schools.com/angular/ng_ng-bind-html.asp) | Binds the innerHTML of an HTML element to application data, and also removes dangerous code from the HTML string. |
| [ng-bind-template](http://www.w3schools.com/angular/ng_ng-bind-template.asp) | Specifies that the text content should be replaced with a template. |
| [ng-blur](http://www.w3schools.com/angular/ng_ng-blur.asp) | Specifies a behavior on blur events. |
| [ng-change](http://www.w3schools.com/angular/ng_ng-change.asp) | Specifies an expression to evaluate when content is being changed by the user. |
| [ng-checked](http://www.w3schools.com/angular/ng_ng-checked.asp) | Specifies if an element is checked or not. |
| [ng-class](http://www.w3schools.com/angular/ng_ng-class.asp) | Specifies CSS classes on HTML elements. |
| [ng-class-even](http://www.w3schools.com/angular/ng_ng-class-even.asp) | Same as ng-class, but will only take effect on even rows. |
| [ng-class-odd](http://www.w3schools.com/angular/ng_ng-class-odd.asp) | Same as ng-class, but will only take effect on odd rows. |
| [ng-click](http://www.w3schools.com/angular/ng_ng-click.asp) | Specifies an expression to evaluate when an element is being clicked. |
| [ng-cloak](http://www.w3schools.com/angular/ng_ng-cloak.asp) | Prevents flickering when your application is being loaded. |
| [ng-controller](http://www.w3schools.com/angular/ng_ng-controller.asp) | Defines the controller object for an application. |
| [ng-copy](http://www.w3schools.com/angular/ng_ng-copy.asp) | Specifies a behavior on copy events. |
| [ng-csp](http://www.w3schools.com/angular/ng_ng-csp.asp) | Changes the content security policy. |
| [ng-cut](http://www.w3schools.com/angular/ng_ng-cut.asp) | Specifies a behavior on cut events. |
| [ng-dblclick](http://www.w3schools.com/angular/ng_ng-dblclick.asp) | Specifies a behavior on double-click events. |
| [ng-disabled](http://www.w3schools.com/angular/ng_ng-disabled.asp) | Specifies if an element is disabled or not. |
| [ng-focus](http://www.w3schools.com/angular/ng_ng-focus.asp) | Specifies a behavior on focus events. |
| ng-form | Specifies an HTML form to inherit controls from. |
| [ng-hide](http://www.w3schools.com/angular/ng_ng-hide.asp) | Hides or shows HTML elements. |
| [ng-href](http://www.w3schools.com/angular/ng_ng-href.asp) | Specifies a url for the <a> element. |
| [ng-if](http://www.w3schools.com/angular/ng_ng-if.asp) | Removes the HTML element if a condition is false. |
| [ng-include](http://www.w3schools.com/angular/ng_ng-include.asp) | Includes HTML in an application. |
| [ng-init](http://www.w3schools.com/angular/ng_ng-init.asp) | Defines initial values for an application. |
| ng-jq | Specifies that the application must use a library, like jQuery. |
| [ng-keydown](http://www.w3schools.com/angular/ng_ng-keydown.asp) | Specifies a behavior on keydown events. |
| [ng-keypress](http://www.w3schools.com/angular/ng_ng-keypress.asp) | Specifies a behavior on keypress events. |
| [ng-keyup](http://www.w3schools.com/angular/ng_ng-keyup.asp) | Specifies a behavior on keyup events. |
| [ng-list](http://www.w3schools.com/angular/ng_ng-list.asp) | Converts text into a list (array). |
| [ng-maxlength](http://www.w3schools.com/angular/ng_ng-maxlength.asp) | Specifies the maximum number of characters allowed in the input field. |
| [ng-minlength](http://www.w3schools.com/angular/ng_ng-minlength.asp) | Specifies the minimum number of characters allowed in the input field. |
| [ng-model](http://www.w3schools.com/angular/ng_ng-model.asp) | Binds the value of HTML controls to application data. |
| [ng-model-options](http://www.w3schools.com/angular/ng_ng-model-options.asp) | Specifies how updates in the model are done. |
| [ng-mousedown](http://www.w3schools.com/angular/ng_ng-mousedown.asp) | Specifies a behavior on mousedown events. |
| [ng-mouseenter](http://www.w3schools.com/angular/ng_ng-mouseenter.asp) | Specifies a behavior on mouseenter events. |
| [ng-mouseleave](http://www.w3schools.com/angular/ng_ng-mouseleave.asp) | Specifies a behavior on mouseleave events. |
| [ng-mousemove](http://www.w3schools.com/angular/ng_ng-mousemove.asp) | Specifies a behavior on mousemove events. |
| [ng-mouseover](http://www.w3schools.com/angular/ng_ng-mouseover.asp) | Specifies a behavior on mouseover events. |
| [ng-mouseup](http://www.w3schools.com/angular/ng_ng-mouseup.asp) | Specifies a behavior on mouseup events. |
| [ng-non-bindable](http://www.w3schools.com/angular/ng_ng-non-bindable.asp) | Specifies that no data binding can happen in this element, or its children. |
| [ng-open](http://www.w3schools.com/angular/ng_ng-open.asp) | Specifies the open attribute of an element. |
| [ng-options](http://www.w3schools.com/angular/ng_ng-options.asp) | Specifies <options> in a <select> list. |
| [ng-paste](http://www.w3schools.com/angular/ng_ng-paste.asp) | Specifies a behavior on paste events. |
| ng-pluralize | Specifies a message to display according to en-us localization rules. |
| [ng-readonly](http://www.w3schools.com/angular/ng_ng-readonly.asp) | Specifies the readonly attribute of an element. |
| [ng-repeat](http://www.w3schools.com/angular/ng_ng-repeat.asp) | Defines a template for each data in a collection. |
| [ng-required](http://www.w3schools.com/angular/ng_ng-required.asp) | Specifies the required attribute of an element. |
| [ng-selected](http://www.w3schools.com/angular/ng_ng-selected.asp) | Specifies the selected attribute of an element. |
| [ng-show](http://www.w3schools.com/angular/ng_ng-show.asp) | Shows or hides HTML elements. |
| [ng-src](http://www.w3schools.com/angular/ng_ng-src.asp) | Specifies the src attribute for the <img> element. |
| [ng-srcset](http://www.w3schools.com/angular/ng_ng-srcset.asp) | Specifies the srcset attribute for the <img> element. |
| [ng-style](http://www.w3schools.com/angular/ng_ng-style.asp) | Specifies the style attribute for an element. |
| [ng-submit](http://www.w3schools.com/angular/ng_ng-submit.asp) | Specifies expressions to run on onsubmit events. |
| [ng-switch](http://www.w3schools.com/angular/ng_ng-switch.asp) | Specifies a condition that will be used to show/hide child elements. |
| ng-transclude | Specifies a point to insert transcluded elements. |
| [ng-value](http://www.w3schools.com/angular/ng_ng-value.asp) | Specifies the value of an input element. |

<!DOCTYPEhtml>

<html>

<scriptsrc=*"/angularpart1/lib/angular.js"*></script>

<scriptsrc=*"/angularpart1/lib/sanitize.js"*></script>

<scriptsrc=*"/angularpart1/js/first.js"*></script>

<style>

*.sky* {

background-color:*lightblue*;

}

*.tomato* {

color:*green*;

}

*.striped* {

color:*white*;

background-color:*black*;

}

*.notstriped* {

color:*black*;

background-color:*yellow*;

}

</style>

<body>

<!--<div ng-app="" ng-init="myText='Hello World!'">

<p>ONE WAY DATA BINDING</p>

<p ng-bind="myText"></p>

<p class="ng-bind: myText"></p>

</div> -->

<h1>ng-app&ng-controller directives</h1>

<h1>ng-csp directives is used to change the security policy of AngularJS

Setting the value of the ng-csp directive to no-unsafe-eval, will stop

AngularJS from running any eval functions, but allow injecting inline styles.

Setting the value of the ng-csp directive to no-inline-style, will stop

AngularJS from injecting any inline styles, but allow eval functions.</h1>

<divng-app=*"myModule"*ng-controller=*"myController"*ng-csp>

<h1>ONE WAY DATA BINDING</h1>

<h1>ng-bind shows firstname value here</h1>

<png-bind=*"firstName"*></p>

<hr>

<h1>ONE WAY DATA BINDING</h1>

<h1>ng-model for both set & get twowayname value</h1>

<inputng-model=*"twowayname"*><span>{{twowayname}}</span>

<hr>

<h1>ng-bind shows myhtml value here</h1>

<png-bind=*"myhtml"*></p>

<hr>

<h1>ng-bind-html shows myhtml value here</h1>

<png-bind-html=*"myhtml"*></p>

<hr>

<h1>ng-bind-template shows firstname& lastName value here</h1>

<png-bind-template=*"{{firstName}} {{lastName}}"*></p>

<hr>

<h1>ng-blur events </h1>

<inputng-blur=*"blurFunc()"*/><br>

<hr>

<h1>ng-focus events </h1>

<inputng-focus=*"focusFunc()"*/><br>

<hr>

<h1>ng-change events </h1>

<inputng-change=*"changeFunc()"*ng-model=*"myValue"*/>{{myValue}}<br>

<hr>

<h1>ng-checked events </h1>

My cars:<inputtype=*"checkbox"*ng-model=*"allchecked"*> Check all<br>

<inputtype=*"checkbox"*ng-checked=*"allchecked"*>Volvo

<inputtype=*"checkbox"*ng-checked=*"allchecked"*>Ford

<inputtype=*"checkbox"*ng-checked=*"allchecked"*>Mercedes<br>

<hr>

<h1>ng-class for dynamic style </h1>

<p>Choose a class:</p>

<selectng-model=*"home"*>

<optionvalue=*"sky"*>Sky</option>

<optionvalue=*"tomato"*>Tomato</option>

<optionvalue=*"sky tomato"*>both</option>

</select>

<divng-class=*"home"*>

<h1>Welcome Home!</h1>

<p>I like it!</p>

</div>

<hr>

<h1>ng-class-even &ng-class-odd for dynamic style </h1>

<h1>ng-repeat for looping </h1>

<table>

<trng-repeat=*"x in records"*ng-class-even=*"'striped'"*ng-class-odd=*"'notstriped'"*>

<td>{{x.Name}}</td>

<td>{{x.Country}}</td>

</tr>

</table>

<hr>

<h1>ng-click event </h1>

<buttonng-click=*"clickFunc()"*>OK</button><br>

<hr>

<h1>ng-cloak for not showing angular code </h1>

<png-cloak>AngularJS applications can make HTML documents

flicker when the application is being loaded, showing the

AngularJS code for a second, before all code are executed.

Use the ng-cloak directive to prevent this.</p>

<hr>

<h1>ng-copy event </h1>

<inputng-copy=*"copyFunc()"*value=*"Copy this text"*/><br>

<hr>

<h1>ng-cut event </h1>

<inputng-cut=*"cutFunc()"*value=*"Cut this text"*/><br>

<hr>

<h1>ng-click event </h1>

<buttonng-dblclick=*"dblclickFunc()"*>Double Click</button><br>

<hr>

<h1>ng-disabled event </h1>

<inputtype=*"checkbox"*ng-model=*"alldisabled"*><br>

<selectng-disabled=*"alldisabled"*>

<option>Female</option>

<option>Male</option>

</select>

<hr>

<h1>ng-hide directive </h1>

<inputtype=*"checkbox"*ng-model=*"myhide"*>

<divng-hide=*"myhide"*>

<hr>

<h1>Welcome</h1>

<p>Welcome to my home.</p>

</div>

<hr>

<h1>ng-href directive </h1>

<divng-init=*"hrefLink= 'https://syntelligence.syntelinc.com/'"*>

<p>Go to <ang-href=*"{{hrefLink}}"*>{{hrefLink}}</a> to learn!</p>

</div>

<hr>

<h1>ng-if directive </h1>

<inputtype=*"checkbox"*ng-model=*"myif"*ng-init=*"myif=true"*>

<divng-if=*"myif"*>

<h1>Welcome</h1>

<p>Welcome to my home.</p>

<hr>

</div>

<divng-if=*"!myif"*>

<h1>Go back</h1>

<p>Please check the above checkbox</p>

<hr>

</div>

<hr>

<h1>ng-include directive </h1>

<divng-include=*"'second.html'"*onload="loadFunc()"style="width: *250px*;height: *100px*;border-style: *solid*;"></div>

<hr>

<h1>ng-keydown directive </h1>

<inputng-keydown=*"keydownFunc()"*/>

<h1>ng-keypress directive </h1>

<inputng-keypress=*"keypressFunc()"*/>

<h1>ng-keyup directive </h1>

<inputng-keyup=*"keyupFunc()"*/>

<hr>

<h1>ng-list directive will make input value become array separated by comma</h1>

<inputng-model=*"customers"*ng-list/>

<pre>{{customers}}</pre>

<hr>

<h1>ng-minlength & maxlength directive </h1>

<formname=*"myForm"*>

<inputname=*"myInput"*ng-model=*"myInput"*ng-minlength=*"5"*ng-maxlength=*"10"*>

<h1ng-if=*"!myForm.myInput.$valid"*>The value is too short or long</h1>

</form>

<hr>

<h1>ng-model-options directive </h1>

<inputng-model=*"modelOptionName1"*ng-model-options=*"{updateOn: 'blur'}"*><span>{{modelOptionName1}}</span><br>

<inputng-model=*"modelOptionName2"*ng-model-options=*"{debounce : 10000}"*><span>{{modelOptionName2}}</span><br>

<inputng-model=*"modelOptionName3"*ng-model-options=*"{allowInvalid : false}"*ng-minlength=*"5"*ng-maxlength=*"10"*><span>{{modelOptionName3}}</span><br>

<hr>

<h1>ng-mousedown directive </h1>

<divng-mousedown=*"mousedownFunc()"*style="width: *250px*;height: *100px*;border-style: *solid*;">Mouse Events!</div>

<h1>ng-mouseenter directive </h1>

<divng-mouseenter=*"mouseenterFunc()"*style="width: *250px*;height: *100px*;border-style: *solid*;">Mouse Events!</div>

<h1>ng-mouseleave directive </h1>

<divng-mouseleave=*"mouseleaveFunc()"*style="width: *250px*;height: *100px*;border-style: *solid*;">Mouse Events!</div>

<h1>ng-mousemove directive </h1>

<divng-mousemove=*"mousemoveFunc()"*style="width: *250px*;height: *100px*;border-style: *solid*;">Mouse Events!</div>

<h1>ng-mouseover directive </h1>

<divng-mouseover=*"mouseoverFunc()"*style="width: *250px*;height: *100px*;border-style: *solid*;">Mouse Events!</div>

<h1>ng-mouseup directive </h1>

<divng-mouseup=*"mouseupFunc()"*style="width: *250px*;height: *100px*;border-style: *solid*;">Mouse Events!</div>

<hr>

<h1>ng-non-bindable directive </h1>

<png-non-bindable>This code is not compiled by AngularJS: {{ 5+5 }}</p>

<hr>

<h1>ng-open directive </h1>

<inputtype=*"checkbox"*ng-model=*"openshowDetails"*>

<detailsng-open=*"openshowDetails"*>

<summary>Copyright 1999-2016.</summary>

<p> - by Refsnes Data. All Rights Reserved.</p>

</details>

<hr>

<h1>ng-options directive </h1>

<selectng-model=*"optionsselectedName"*ng-options=*"item for item in optionsnames"*></select>

<hr>

<h1>ng-paste directive </h1>

<inputng-paste=*"pastefunc()"*placeholder=*"Paste text here"*/>

<hr>

<h1>ng-readonly directive </h1>

<inputtype=*"checkbox"*ng-model=*"readonlyall"*><br>

<inputtype=*"text"*ng-readonly=*"readonlyall"*value=*"Sample"*>

<hr>

<h1>ng-readonly directive </h1>

<formname=*"requiredmyForm"*>

Click here to make the input field required:

<inputtype=*"checkbox"*ng-model=*"requiredmyVar"*><br>

<inputname=*"requiredmyInput"*ng-model=*"requiredmyInput"*ng-required=*"requiredmyVar"*>

<h1ng-if=*"!requiredmyForm.requiredmyInput.$valid"*>The input field cannot be empty</h1>

</form>

<hr>

<h1>ng-selected directive </h1>

<inputtype=*"checkbox"*ng-model=*"myselected"*>

<p>My Favourite car:</p>

<select>

<option>Volvo</option>

<optionng-selected=*"myselected"*>BMW</option>

<option>Ford</option>

</select>

<hr>

<h1>ng-src directive </h1>

<divng-init=*"myVarsrc= '/angularpart1/images/i1.jpg'"*>

<h1>Angular</h1>

<imgng-src=*"{{myVarsrc}}"*></img>

</div>

<hr>

<h1>ng-style directive </h1>

<h1ng-style=*"styleObj"*>Welcome</h1>

<hr>

<h1>ng-submit directive </h1>

<formng-submit=*"submitFunc()"*>

<inputtype=*"text"*>

<inputtype=*"submit"*>

</form>

<hr>

<h1>ng-swith & ng-switch-when directive </h1>

<selectng-model=*"myVar"*>

<optionvalue=*"dogs"*>Dogs

<optionvalue=*"tuts"*>Tutorials

<optionvalue=*"cars"*>Cars

</select>

<hr>

<divng-switch=*"myVar"*>

<divng-switch-when=*"dogs"*>

<h1>Dogs</h1>

<p>Welcome to a world of dogs.</p>

</div>

<divng-switch-when=*"tuts"*>

<h1>Tutorials</h1>

<p>Learn from examples.</p>

</div>

<divng-switch-when=*"cars"*>

<h1>Cars</h1>

<p>Read about cars.</p>

</div>

<divng-switch-default>

<h1>Switch</h1>

<p>Select topic from the dropdown, to switch the content of this DIV.</p>

</div>

</div>

<hr>

<h1>ng-value directive </h1>

<inputng-value=*"valueVar"*>

<hr>

</div>

</body>

</html>

**var** app = angular.module("myModule",['ngSanitize']);

app.controller("myController", **function**($scope) {

$scope.twowayname="hello";

$scope.firstName = "John";

$scope.lastName = "Steve";

$scope.modelOptionName1="Check with Binding value1";

$scope.modelOptionName2="Check with Binding value2";

$scope.modelOptionName3="Check with Binding value3";

$scope.myhtml = "My name is: <h1>John Doe</h1>";

$scope.optionsnames = ["Emil", "Tobias", "Linus"];

$scope.blurFunc=**function**(){

alert("blur event occured");

};

$scope.focusFunc=**function**(){

alert("focus event occured");

};

$scope.clickFunc=**function**(){

alert("click event occured");

};

$scope.copyFunc=**function**(){

alert("copy event occured");

};

$scope.cutFunc=**function**(){

alert("cut event occured");

};

$scope.dblclickFunc=**function**(){

alert("double click event occured");

};

$scope.loadFunc=**function**(){

alert("It is loaded");

};

$scope.keydownFunc=**function**(){

alert("keydown event occured");

};

$scope.keypressFunc=**function**(){

alert("keypress event occured");

};

$scope.keyupFunc=**function**(){

alert("keyup event occured");

};

$scope.mousedownFunc=**function**(){

alert("Mousedown event occured");

};

$scope.mouseenterFunc=**function**(){

alert("Mouseenter event occured");

};

$scope.mouseleaveFunc=**function**(){

alert("Mouseleave event occured");

};

$scope.mousemoveFunc=**function**(){

alert("Mousemove event occured");

};

$scope.mouseoverFunc=**function**(){

alert("Mouseover event occured");

};

$scope.mouseupFunc=**function**(){

alert("Mouseup event occured");

};

$scope.pasteFunc=**function**(){

alert("Paste event occured");

};

$scope.submitFunc=**function**(){

alert("Submit event occured");

};

$scope.records = [

{"Name" :"Alfreds Futterkiste","Country" : "Germany"},

{"Name" :"Berglunds snabbköp","Country" : "Sweden"},

{"Name" :"Centro comercial Moctezuma","Country" : "Mexico"},

{"Name" :"Ernst Handel","Country" : "Austria"}];

$scope.styleObj = {

"color" : "white",

"background-color" : "coral",

"font-size" : "60px",

"padding" : "50px"

};

$scope.valueVar="My Value from AngularJS";

});

## AngularJS Directives on HTML Elements

AngularJS modifies the default behavior of some HTML elements.

|  |  |
| --- | --- |
| **Element** | **Description** |
| [a](http://www.w3schools.com/angular/ng_a.asp) | AngularJS modifies the <a> element's default behaviors. |
| [form](http://www.w3schools.com/angular/ng_form.asp) | AngularJS modifies the <form> element's default behaviors. |
| [input](http://www.w3schools.com/angular/ng_input.asp) | AngularJS modifies the <input> element's default behaviors. |
| script | AngularJS modifies the <script> element's default behaviors. |
| select | AngularJS modifies the <select> element's default behaviors. |
| [textarea](http://www.w3schools.com/angular/ng_textarea.asp) | AngularJS modifies the <textarea> element's default behaviors. |

# **Form**

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

Forms inside an AngularJS application are given certain classes. These classes can be used to style forms according to their state.

The following classes are added:

* 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

# **Input**

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

The following classes are added:



<!DOCTYPEhtml>

<html>

<scriptsrc=*"/angularpart1/lib/angular.js"*></script>

<scriptsrc=*"/angularpart1/js/three.js"*></script>

<scripttype=*"text/ng-template"*id=*"/tpl.html"*>

Content of the template.

</script>

<style>

*.my-form* {

transition:*all linear 0.5s*;

background: *transparent*;

}

*.my-form.ng-invalid* {

background: *red*;

*}*

</style>

<bodyonload="alert('page loaded');">

<ahref=*""*>Click Here</a>

<divng-app=*"myModule1"*ng-controller=*"myController1"*>

<ahref=*""*>Click Here</a>

<hr>

<div>

Form Element

<formname=*"myForm"*class=*"my-form"*>

userType: <inputname=*"input"*ng-model=*"userType"*required>

<spanclass=*"error"*ng-show=*"myForm.input.$error.required"*>Required!</span><br>

<code>userType = {{userType}}</code><br>

<code>myForm.input.$valid = {{myForm.input.$valid}}</code><br>

<code>myForm.input.$error = {{myForm.input.$error}}</code><br>

<code>myForm.$valid = {{myForm.$valid}}</code><br>

<code>myForm.$error.required = {{!!myForm.$error.required}}</code><br>

<code>myForm.$error = {{myForm.$error}}</code><br>

</form>

</div>

<hr>

<div>

Input Element

<formname=*"myForm1"*>

<label>

User name:

<inputtype=*"text"*name=*"userName"*ng-model=*"user.name"*required>

</label>

<divrole=*"alert"*>

<spanclass=*"error"*ng-show=*"myForm1.userName.$error.required"*>

Required!</span>

</div>

<label>

Last name:

<inputtype=*"text"*name=*"lastName"*ng-model=*"user.last"*

ng-minlength=*"3"*ng-maxlength=*"10"*>

</label>

<labelfor=*"exampleInput"*>Pick a date in 2013:</label>

<inputtype=*"date"*id=*"exampleInput"*name=*"input"*ng-model=*"user.value"*ng-Trim=*"true"*

placeholder=*"yyyy-MM-dd"*min=*"2013-01-01"*max=*"2013-12-31"*required/>

<divrole=*"alert"*>

<spanclass=*"error"*ng-show=*"myForm1.input.$error.required"*>Required!</span>

<spanclass=*"error"*ng-show=*"myForm1.input.$error.date"*>Not a valid date!</span>

</div>

<label>Value1:

<inputtype=*"checkbox"*ng-model=*"user.value1"*ng-change=*"changeFunc()"*>

</label><br/>

<label>Value2:

<inputtype=*"checkbox"*ng-model=*"user.value2"*ng-true-value=*"'YES'"*ng-false-value=*"'NO'"*>

</label><br/>

<divrole=*"alert"*>

<spanclass=*"error"*ng-show=*"myForm1.lastName.$error.minlength"*>

Too short!</span>

<spanclass=*"error"*ng-show=*"myForm1.lastName.$error.maxlength"*>

Too long!</span>

</div>

</form>

<hr>

<tt>user = {{user}}</tt><br/>

<tt>value = {{user.value | date: "yyyy-MM-dd"}}</tt><br/>

<tt>myForm1.userName.$valid = {{myForm1.userName.$valid}}</tt><br/>

<tt>myForm1.userName.$error = {{myForm1.userName.$error}}</tt><br/>

<tt>myForm1.lastName.$valid = {{myForm1.lastName.$valid}}</tt><br/>

<tt>myForm1.lastName.$error = {{myForm1.lastName.$error}}</tt><br/>

<tt>myForm1.$valid = {{myForm1.$valid}}</tt><br/>

<tt>myForm1.$error.required = {{!!myForm1.$error.required}}</tt><br/>

<tt>myForm1.$error.minlength = {{!!myForm1.$error.minlength}}</tt><br/>

<tt>myForm1.$error.maxlength = {{!!myForm1.$error.maxlength}}</tt><br/>

</div>

<hr>

Script Element

<ang-click=*"currentTpl='/tpl.html'"*id=*"tpl-link"*>Load inlined template</a>

<divid=*"tpl-content"*ng-includesrc=*"currentTpl"*></div>

<hr>

Script Element

</div>

</body>

</html>

**var** app = angular.module("myModule1",[]);

app.controller("myController1", **function**($scope) {

$scope.userType = 'guest';

$scope.user = {name: 'guest', last: 'visitor',value1:**true**,value2:'YES',value: **new** Date(2013, 9, 22)};

$scope.changeFunc=**function**(){

alert('Change event occured');

};

});

Angular.Copy

<divng-controller="ExampleController">

<formnovalidateclass="simple-form">

<label>Name: <inputtype="text"ng-model="user.name"/></label><br/>

<label>Age: <inputtype="number"ng-model="user.age"/></label><br/>

Gender: <label><inputtype="radio"ng-model="user.gender"value="male"/>male</label>

<label><inputtype="radio"ng-model="user.gender"value="female"/>female</label><br/>

<buttonng-click="reset()">RESET</button>

<buttonng-click="update(user)">SAVE</button>

</form>

<pre>form = {{user | json}}</pre>

<pre>master = {{master | json}}</pre>

</div>

// Module: copyExample

angular.

module('copyExample',[]).

controller('ExampleController',['$scope',function($scope){

$scope.master ={};

$scope.reset =function(){

// Example with 1 argument

$scope.user =angular.copy($scope.master);

};

$scope.update =function(user){

// Example with 2 arguments

angular.copy(user, $scope.master);

};

$scope.reset();

}]);

Angular.equals

<divng-controller="ExampleController">

<formnovalidate>

<h3>User 1</h3>

Name: <inputtype="text"ng-model="user1.name">

Age: <inputtype="number"ng-model="user1.age">

<h3>User 2</h3>

Name: <inputtype="text"ng-model="user2.name">

Age: <inputtype="number"ng-model="user2.age">

<div>

<br/>

<inputtype="button"value="Compare"ng-click="compare()">

</div>

User 1: <pre>{{user1 | json}}</pre>

User 2: <pre>{{user2 | json}}</pre>

Equal: <pre>{{result}}</pre>

</form>

</div>

angular.module('equalsExample',[]).controller('ExampleController',['$scope',function($scope){

$scope.user1 ={};

$scope.user2 ={};

$scope.compare =function(){

$scope.result =angular.equals($scope.user1, $scope.user2);

};

}]);

Angular.forEach

var values ={name:'misko', gender:'male'};

var log =[];

angular.forEach(values,function(value, key){

this.push(key +': '+ value);

}, log);

expect(log).toEqual(['name: misko','gender: male']);

Angular.

// Create a new module

var myModule = angular.module('myModule',[]);

// register a new service

myModule.value('appName','MyCoolApp');

// configure existing services inside initialization blocks.

myModule.config(['$locationProvider',function($locationProvider){

// Configure existing providers

$locationProvider.hashPrefix('!');

}]);

## AngularJS Filters

|  |  |
| --- | --- |
| **Filter** | **Description** |
| [currency](http://www.w3schools.com/angular/ng_filter_currency.asp) | Format a number to a currency format. |
| [date](http://www.w3schools.com/angular/ng_filter_date.asp) | Format a date to a specified format. |
| [filter](http://www.w3schools.com/angular/ng_filter_filter.asp) | Select a subset of items from an array. |
| [json](http://www.w3schools.com/angular/ng_filter_json.asp) | Format an object to a JSON string. |
| [limitTo](http://www.w3schools.com/angular/ng_filter_limitto.asp) | Limits an array, or a string, into a specified number of elements/characters. |
| [lowercase](http://www.w3schools.com/angular/ng_filter_lowercase.asp) | Format a string to lower case. |
| [number](http://www.w3schools.com/angular/ng_filter_number.asp) | Format a number to a string. |
| [orderBy](http://www.w3schools.com/angular/ng_filter_orderby.asp) | Orders an array by an expression. |
| [uppercase](http://www.w3schools.com/angular/ng_filter_uppercase.asp) | Format a string to upper case. |

**Currency**

{{ currency\_expression | currency : symbol : fractionSize}}

$filter('currency')(amount, symbol, fractionSize)

<script>

angular.module('currencyExample',[])

.controller('ExampleController',['$scope',function($scope){

$scope.amount =1234.56;

}]);

</script>

<divng-controller="ExampleController">

<inputtype="number"ng-model="amount"aria-label="amount"><br>

default currency symbol ($): <spanid="currency-default">{{amount | currency}}</span><br>

custom currency identifier (USD$): <spanid="currency-custom">{{amount | currency:"USD$"}}</span><br>

no fractions (0): <spanid="currency-no-fractions">{{amount | currency:"USD$":0}}</span>

</div>

**Date**

format string can be composed of the following elements:

* 'yyyy': 4 digit representation of year (e.g. AD 1 => 0001, AD 2010 => 2010)
* 'yy': 2 digit representation of year, padded (00-99). (e.g. AD 2001 => 01, AD 2010 => 10)
* 'y': 1 digit representation of year, e.g. (AD 1 => 1, AD 199 => 199)
* 'MMMM': Month in year (January-December)
* 'MMM': Month in year (Jan-Dec)
* 'MM': Month in year, padded (01-12)
* 'M': Month in year (1-12)
* 'LLLL': Stand-alone month in year (January-December)
* 'dd': Day in month, padded (01-31)
* 'd': Day in month (1-31)
* 'EEEE': Day in Week,(Sunday-Saturday)
* 'EEE': Day in Week, (Sun-Sat)
* 'HH': Hour in day, padded (00-23)
* 'H': Hour in day (0-23)
* 'hh': Hour in AM/PM, padded (01-12)
* 'h': Hour in AM/PM, (1-12)
* 'mm': Minute in hour, padded (00-59)
* 'm': Minute in hour (0-59)
* 'ss': Second in minute, padded (00-59)
* 's': Second in minute (0-59)
* 'sss': Millisecond in second, padded (000-999)
* 'a': AM/PM marker
* 'Z': 4 digit (+sign) representation of the timezone offset (-1200-+1200)
* 'ww': Week of year, padded (00-53). Week 01 is the week with the first Thursday of the year
* 'w': Week of year (0-53). Week 1 is the week with the first Thursday of the year
* 'G', 'GG', 'GGG': The abbreviated form of the era string (e.g. 'AD')
* 'GGGG': The long form of the era string (e.g. 'Anno Domini')

format string can also be one of the following predefined [localizable formats](https://docs.angularjs.org/guide/i18n):

* 'medium': equivalent to 'MMM d, y h:mm:ss a' for en\_US locale (e.g. Sep 3, 2010 12:05:08 PM)
* 'short': equivalent to 'M/d/yy h:mm a' for en\_US locale (e.g. 9/3/10 12:05 PM)
* 'fullDate': equivalent to 'EEEE, MMMM d, y' for en\_US locale (e.g. Friday, September 3, 2010)
* 'longDate': equivalent to 'MMMM d, y' for en\_US locale (e.g. September 3, 2010)
* 'mediumDate': equivalent to 'MMM d, y' for en\_US locale (e.g. Sep 3, 2010)
* 'shortDate': equivalent to 'M/d/yy' for en\_US locale (e.g. 9/3/10)
* 'mediumTime': equivalent to 'h:mm:ss a' for en\_US locale (e.g. 12:05:08 PM)
* 'shortTime': equivalent to 'h:mm a' for en\_US locale (e.g. 12:05 PM)

format string can contain literal values. These need to be escaped by surrounding with single quotes (e.g."h 'in the morning'"). In order to output a single quote, escape it - i.e., two single quotes in a sequence (e.g. "h 'o''clock'").

{{ date\_expression | date : format : timezone}}

$filter('date')(date, format, timezone)

<spanng-non-bindable>{{1288323623006 | date:'medium'}}</span>:

<span>{{1288323623006 | date:'medium'}}</span><br>

<spanng-non-bindable>{{1288323623006 | date:'yyyy-MM-dd HH:mm:ss Z'}}</span>:

<span>{{1288323623006 | date:'yyyy-MM-dd HH:mm:ss Z'}}</span><br>

<spanng-non-bindable>{{1288323623006 | date:'MM/dd/yyyy @ h:mma'}}</span>:

<span>{{'1288323623006' | date:'MM/dd/yyyy @ h:mma'}}</span><br>

<spanng-non-bindable>{{1288323623006 | date:"MM/dd/yyyy 'at' h:mma"}}</span>:

<span>{{'1288323623006' | date:"MM/dd/yyyy 'at' h:mma"}}</span><br>

**Filter**

{{ filter\_expression | filter : expression : comparator : anyPropertyKey}}

$filter('filter')(array, expression, comparator, anyPropertyKey)

<divng-init="friends = [{name:'John', phone:'555-1276'},

{name:'Mary', phone:'800-BIG-MARY'},

{name:'Mike', phone:'555-4321'},

{name:'Adam', phone:'555-5678'},

{name:'Julie', phone:'555-8765'},

{name:'Juliette', phone:'555-5678'}]"></div>

<label>Search: <inputng-model="searchText"></label>

<tableid="searchTextResults">

<tr><th>Name</th><th>Phone</th></tr>

<trng-repeat="friend in friends | filter:searchText">

<td>{{friend.name}}</td>

<td>{{friend.phone}}</td>

</tr>

</table>

<hr>

<label>Any: <inputng-model="search.$"></label><br>

<label>Name only <inputng-model="search.name"></label><br>

<label>Phone only <inputng-model="search.phone"></label><br>

<label>Equality <inputtype="checkbox"ng-model="strict"></label><br>

<tableid="searchObjResults">

<tr><th>Name</th><th>Phone</th></tr>

<trng-repeat="friendObj in friends | filter:search:strict">

<td>{{friendObj.name}}</td>

<td>{{friendObj.phone}}</td>

</tr>

</table>

**Json**

{{ json\_expression | json : spacing}}

$filter('json')(object, spacing)

<preid="default-spacing">{{ {'name':'value'} | json }}</pre>

<preid="custom-spacing">{{ {'name':'value'} | json:4 }}</pre>

**LimitTo**

{{ limitTo\_expression | limitTo : limit :begin}}

$filter('limitTo')(input, limit,begin)

<script>

angular.module('limitToExample',[])

.controller('ExampleController',['$scope',function($scope){

$scope.numbers =[1,2,3,4,5,6,7,8,9];

$scope.letters ="abcdefghi";

$scope.longNumber =2345432342;

$scope.numLimit =3;

$scope.letterLimit =3;

$scope.longNumberLimit =3;

}]);

</script>

<divng-controller="ExampleController">

<label>

Limit {{numbers}} to:

<inputtype="number"step="1"ng-model="numLimit">

</label>

<p>Output numbers: {{ numbers | limitTo:numLimit }}</p>

<label>

Limit {{letters}} to:

<inputtype="number"step="1"ng-model="letterLimit">

</label>

<p>Output letters: {{ letters | limitTo:letterLimit }}</p>

<label>

Limit {{longNumber}} to:

<inputtype="number"step="1"ng-model="longNumberLimit">

</label>

<p>Output long number: {{ longNumber | limitTo:longNumberLimit }}</p>

</div>

**Lowercase**

{{ lowercase\_expression | lowercase}}

$filter('lowercase')()

**Number**

{{ number\_expression | number : fractionSize}}

$filter('number')(number, fractionSize)

<script>

angular.module('numberFilterExample',[])

.controller('ExampleController',['$scope',function($scope){

$scope.val =1234.56789;

}]);

</script>

<divng-controller="ExampleController">

<label>Enter number: <inputng-model='val'></label><br>

Default formatting: <spanid='number-default'>{{val | number}}</span><br>

No fractions: <span>{{val | number:0}}</span><br>

Negative number: <span>{{-val | number:4}}</span>

</div>

**Uppercase**

{{ lowercase\_expression | uppercase}}

$filter('uppercase')()

**orderBy**

{{ orderBy\_expression | orderBy : expression : reverse : comparator}}

$filter('orderBy')(collection, expression, reverse, comparator)

**Ordering a table with** ngRepeat

angular.module('orderByExample1',[])

.controller('ExampleController',['$scope',function($scope){

$scope.friends =[

{name:'John', phone:'555-1212', age:10},

{name:'Mary', phone:'555-9876', age:19},

{name:'Mike', phone:'555-4321', age:21},

{name:'Adam', phone:'555-5678', age:35},

{name:'Julie', phone:'555-8765', age:29}

];

}]);

<divng-controller="ExampleController">

<tableclass="friends">

<tr>

<th>Name</th>

<th>Phone Number</th>

<th>Age</th>

</tr>

<trng-repeat="friend in friends | orderBy:'-age'">

<td>{{friend.name}}</td>

<td>{{friend.phone}}</td>

<td>{{friend.age}}</td>

</tr>

</table>

</div>

**Changing parameters dynamically**

angular.module('orderByExample2',[])

.controller('ExampleController',['$scope',function($scope){

var friends =[

{name:'John', phone:'555-1212', age:10},

{name:'Mary', phone:'555-9876', age:19},

{name:'Mike', phone:'555-4321', age:21},

{name:'Adam', phone:'555-5678', age:35},

{name:'Julie', phone:'555-8765', age:29}

];

$scope.propertyName ='age';

$scope.reverse =true;

$scope.friends = friends;

$scope.sortBy =function(propertyName){

$scope.reverse =($scope.propertyName === propertyName)?!$scope.reverse :false;

$scope.propertyName = propertyName;

};

}]);

<divng-controller="ExampleController">

<pre>Sort by = {{propertyName}}; reverse = {{reverse}}</pre>

<hr/>

<buttonng-click="propertyName = null; reverse = false">Set to unsorted</button>

<hr/>

<tableclass="friends">

<tr>

<th>

<buttonng-click="sortBy('name')">Name</button>

<spanclass="sortorder"ng-show="propertyName === 'name'"ng-class="{reverse: reverse}"></span>

</th>

<th>

<buttonng-click="sortBy('phone')">Phone Number</button>

<spanclass="sortorder"ng-show="propertyName === 'phone'"ng-class="{reverse: reverse}"></span>

</th>

<th>

<buttonng-click="sortBy('age')">Age</button>

<spanclass="sortorder"ng-show="propertyName === 'age'"ng-class="{reverse: reverse}"></span>

</th>

</tr>

<trng-repeat="friend in friends | orderBy:propertyName:reverse">

<td>{{friend.name}}</td>

<td>{{friend.phone}}</td>

<td>{{friend.age}}</td>

</tr>

</table>

</div>

.friends {

border-collapse: collapse;

}

.friends th{

border-bottom:1px solid;

}

.friends td,.friends th{

border-left:1px solid;

padding:5px10px;

}

.friends td:first-child,.friends th:first-child {

border-left: none;

}

.sortorder:after{

content:'\25b2';// BLACK UP-POINTING TRIANGLE

}

.sortorder.reverse:after{

content:'\25bc';// BLACK DOWN-POINTING TRIANGLE

}

**Using orderBy inside a controller**

.friends {

border-collapse: collapse;

}

.friends th{

border-bottom:1px solid;

}

.friends td,.friends th{

border-left:1px solid;

padding:5px10px;

}

.friends td:first-child,.friends th:first-child {

border-left: none;

}

.sortorder:after{

content:'\25b2';// BLACK UP-POINTING TRIANGLE

}

.sortorder.reverse:after{

content:'\25bc';// BLACK DOWN-POINTING TRIANGLE

}

angular.module('orderByExample3',[])

.controller('ExampleController',['$scope','orderByFilter',function($scope, orderBy){

var friends =[

{name:'John', phone:'555-1212', age:10},

{name:'Mary', phone:'555-9876', age:19},

{name:'Mike', phone:'555-4321', age:21},

{name:'Adam', phone:'555-5678', age:35},

{name:'Julie', phone:'555-8765', age:29}

];

$scope.propertyName ='age';

$scope.reverse =true;

$scope.friends =orderBy(friends, $scope.propertyName, $scope.reverse);

$scope.sortBy =function(propertyName){

$scope.reverse =(propertyName !==null&& $scope.propertyName === propertyName)

?!$scope.reverse :false;

$scope.propertyName = propertyName;

$scope.friends =orderBy(friends, $scope.propertyName, $scope.reverse);

};

}]);

<divng-controller="ExampleController">

<pre>Sort by = {{propertyName}}; reverse = {{reverse}}</pre>

<hr/>

<buttonng-click="sortBy(null)">Set to unsorted</button>

<hr/>

<tableclass="friends">

<tr>

<th>

<buttonng-click="sortBy('name')">Name</button>

<spanclass="sortorder"ng-show="propertyName === 'name'"ng-class="{reverse: reverse}"></span>

</th>

<th>

<buttonng-click="sortBy('phone')">Phone Number</button>

<spanclass="sortorder"ng-show="propertyName === 'phone'"ng-class="{reverse: reverse}"></span>

</th>

<th>

<buttonng-click="sortBy('age')">Age</button>

<spanclass="sortorder"ng-show="propertyName === 'age'"ng-class="{reverse: reverse}"></span>

</th>

</tr>

<trng-repeat="friend in friends">

<td>{{friend.name}}</td>

<td>{{friend.phone}}</td>

<td>{{friend.age}}</td>

</tr>

</table>

</div>

**Using a custom comparator**

.friends-container {

display:inline-block;

margin:030px;

}

.friends {

border-collapse: collapse;

}

.friends th{

border-bottom:1px solid;

}

.friends td,.friends th{

border-left:1px solid;

padding:5px10px;

}

.friends td:first-child,.friends th:first-child {

border-left: none;

}

angular.module('orderByExample4',[])

.controller('ExampleController',['$scope',function($scope){

$scope.friends =[

{name:'John', favoriteLetter:'Ä'},

{name:'Mary', favoriteLetter:'Ü'},

{name:'Mike', favoriteLetter:'Ö'},

{name:'Adam', favoriteLetter:'H'},

{name:'Julie', favoriteLetter:'Z'}

];

$scope.localeSensitiveComparator =function(v1, v2){

// If we don't get strings, just compare by index

if(v1.type !=='string'|| v2.type !=='string'){

return(v1.index < v2.index)?-1:1;

}

// Compare strings alphabetically, taking locale into account

return v1.value.localeCompare(v2.value);

};

}]);

<divng-controller="ExampleController">

<divclass="friends-container custom-comparator">

<h3>Locale-sensitive Comparator</h3>

<tableclass="friends">

<tr>

<th>Name</th>

<th>Favorite Letter</th>

</tr>

<trng-repeat="friend in friends | orderBy:'favoriteLetter':false:localeSensitiveComparator">

<td>{{friend.name}}</td>

<td>{{friend.favoriteLetter}}</td>

</tr>

</table>

</div>

<divclass="friends-container default-comparator">

<h3>Default Comparator</h3>

<tableclass="friends">

<tr>

<th>Name</th>

<th>Favorite Letter</th>

</tr>

<trng-repeat="friend in friends | orderBy:'favoriteLetter'">

<td>{{friend.name}}</td>

<td>{{friend.favoriteLetter}}</td>

</tr>

</table>

</div>

</div>