Cheatography

Angular Js Directives v.1.3.0 Cheat Sheet

by Roman K. (Roman) via cheatography.com/19465/cs/2481/

A

Modifies the default behavior of the html A tag so that the default action is prevented when the href attribute is empty.

Priority level: 0

Example:

```
<a href="" ng-cli ck= " lis t.a ddI tem ()">Add Item</ a>
```

Form

Submitting a form and preventing the default action.

CSS classes:

ng-valid is set if the form is valid.

ng-invalid is set if the form is invalid.

ng-pristine is set if the form is pristine.

ng-dirty is set if the form is dirty.

ng-submitted is set if the form was submitted.

Arguments:

name(optional, string): If specified, the form controller will be published into related scope, under this name.

Priority level 0

ngApp

Use this directive to auto-bootstrap an AngularJS application. The ngApp directive designates the root element of the application and is typically placed near the root element of the page - e.g. on the <body> or <html> tags.

ngApp - {angular.Module} - an optional application module name to load.

ngStrictDi (optional) - {boolean} - if this attribute is present on the app element, the injector will be created in "strict-di" mode. This means that the application will fail to invoke functions which do not use explicit function annotation (and are thus unsuitable for minification), as described in the Dependency Injection guide, and useful debugging info will assist in tracking down the root of these bugs.

ngBind

The ngBind attribute tells Angular to replace the text content of the specified HTML element with the value of a given expression, and to update the text content when the value of that expression changes.

ngBind - {expression} - Expression to evaluate.

ngBindHtml

Creates a binding that will innerHTML the result of evaluating the expression into the current element in a secure way. By default, the innerHTML-ed content will be sanitized using the $sanitize service. To utilize this functionality, ensure that $sanitize is available, for example, by including ngSanitize in your module's dependencies (not in core Angular.) You may also bypass sanitization for values you know are safe. To do so, bind to an explicitly trusted value via $sce.trustAsHtml. See the example under Strict Contextual Escaping (SCE).. **ngBindHtml** - {expression} - Expression to evaluate.

ngBindTemplate

The ngBindTemplate directive specifies that the element text content should be replaced with the interpolation of the template in the ngBindTemplate attribute. Unlike ngBind, the ngBindTemplate can contain multiple {{ }} expressions. This directive is needed since some HTML elements (such as TITLE and OPTION) cannot contain SPAN elements.

ngBindTemplate - {string} - template of form {{ expression }} to eval.

ngBlur

Specify custom behavior on blur event.

Note: As the blur event is executed synchronously also during DOM manipulations (e.g. removing a focussed input), AngularJS executes the expression using scope.$evalAsync if the event is fired during an $apply to ensure a consistent state.

ngBlur - {expression} - Expression to evaluate upon blur. (Event object is available as $event)

ngChange

Evaluate the given expression when the user changes the input. The expression is evaluated immediately, unlike the JavaScript onchange event which only triggers at the end of a change (usually, when the user leaves the form element or presses the return key).
Note, this directive requires ngModel to be present.
ngChange - {expression} - Expression to evaluate upon change in input value.

ngChecked

The HTML specification does not require browsers to preserve the values of boolean attributes such as checked. (Their presence means true and their absence means false.) If we put an Angular interpolation expression into such an attribute then the binding information would be lost when the browser removes the attribute. The ngChecked directive solves this problem for the checked attribute. This complementary directive is not removed by the browser and so provides a permanent reliable place to store the binding information.. Argument:
ngChecked - {expression} - If the expression is truthy, then special attribute "checked" will be set on the element

ngClass

The ngClass directive allows you to dynamically set CSS classes on an HTML element by databinding an expression that represents all classes to be added.
The directive operates in three different ways, depending on which of three types the expression evaluates to:
- If the expression evaluates to a string, the string should be one or more space-delimited class names.
- If the expression evaluates to an array, each element of the array should be a string that is one or more space-delimited class names.
- If the expression evaluates to an object, then for each key-value pair of the object with a truthy value the corresponding key is used as a class name.
The directive won't add duplicate classes if a particular class was already set.
When the expression changes, the previously added classes are removed and only then the new classes are added.

ngClass (cont)

Animations:
add - happens just before the class is applied to the element remove - happens just before the class is removed from the element
Arguments:
ngClass - {expression} - Expression to eval. The result of the evaluation can be a string representing space delimited class names, an array, or a map of class names to boolean values. In the case of a map, the names of the properties whose values are truthy will be added as css classes to the element.

ngClassEven ngClassOdd

The ngClassOdd and ngClassEven directives work exactly as ngClass, except they work in conjunction with ngRepeat and take effect only on odd (even) rows.
Usage:
as attribute:
<ANY
ng-class-even="">
...
</ANY>
as CSS class:
<ANY class="ng-class-even: ;"> ... </ANY>
Arguments
ngClassEven | ngClassOdd - {expression} - Expression to eval. The result of the evaluation can be a string representing space delimited class names or an array.

ngClick

The ngClick directive allows you to specify custom behavior when an element is clicked.
Arguments:
ngClick - {expression} - Expression to evaluate upon click. (Event object is available as $event)

ngCloak

The ngCloak directive is used to prevent the Angular html template from being briefly displayed by the browser in its raw (uncompiled) form while your application is loading. Use this directive to avoid the undesirable flicker effect caused by the html template display.

ngController

The ngController directive attaches a controller class to the view. This is a key aspect of how angular supports the principles behind the Model-View-Controller design pattern.
MVC components in angular:
Model — Models are the properties of a scope; scopes are attached to the DOM where scope properties are accessed through bindings.
View — The template (HTML with data bindings) that is rendered into the View.
Controller — The ngController directive specifies a Controller class; the class contains business logic behind the application to decorate the scope with functions and values
Arguments
ngController - {expression} - Name of a constructor function registered with the current $controllerProvider or an expression that on the current scope evaluates to a constructor function. The controller instance can be published into a scope property by specifying ng-controller="as propertyName". If the current $controllerProvider is configured to use globals (via $controllerProvider.allowGlobals()), this may also be the name of a globally accessible constructor function (not recommended).

ngCopy

Specify custom behavior on copy event.
Arguments:
ngCopy - {expression} - Expression to evaluate upon copy. (Event object is available as $event)

ngCsp

Enables CSP (Content Security Policy) support.
This is necessary when developing things like Google Chrome Extensions.
CSP forbids apps to use eval or Function(string) generated functions (among other things). For Angular to be CSP compatible there are only two things that we need to do differently:
- don't use Function constructor to generate optimized value getters
- don't inject custom stylesheet into the document
AngularJS uses Function(string) generated functions as a speed optimization. Applying the ngCsp directive will cause Angular to use CSP compatibility mode. When this mode is on AngularJS will evaluate all expressions up to 30% slower than in non-CSP mode, but no security violations will be raised.

ngCut

Specify custom behavior on cut event.
ngCut - {expression} - Expression to evaluate upon cut. (Event object is available as $event)

ngKeydown

Specify custom behavior on keydown event.
Arguments:
ngKeydown - {expression} - Expression to evaluate upon keydown. (Event object is available as $event and can be interrogated for keyCode, altKey, etc.)

ngKeypress

Specify custom behavior on keypress event.
Arguments:
ngKeypress - {expression} - Expression to evaluate upon keypress. (Event object is available as $event and can be interrogated for keyCode, altKey, etc.)

ngKeyup

Specify custom behavior on keyup event.
Arguments:
ngKeyup - {expression} - Expression to evaluate upon keyup. (Event object is available as $event and can be interrogated for keyCode, altKey, etc.)

ngList

Text input that converts between a delimited string and an array of strings. The default delimiter is a comma followed by a space - equivalent to ng-list=", ". You can specify a custom delimiter as the value of the ngList attribute - for example, ng-list=" | ".
Arguments:
ngList (optional) - {string} - optional delimiter that should be used to split the value.

ngModel

The ngModel directive binds an input,select, textarea (or custom form control) to a property on the scope using NgModelController, which is created and exposed by this directive.

ngModelOptions

Allows tuning how model updates are done. Using ngModelOptions you can specify a custom list of events that will trigger a model update and/or a debouncing delay so that the actual update only takes place when a timer expires; this timer will be reset after another change takes place.
Arguments:
ngModelOptions - {Object} - options to apply to the current model. Valid keys are:
updateOn: string specifying which event should be the input bound to. You can set several events using an space delimited list. There is a special event called default that matches the default events belonging of the control.
debounce: integer value which contains the debounce model update value in milliseconds. A value of 0 triggers an immediate update. If an object is supplied instead, you can specify a custom value for each event. For example: ng-model-options="{ updateOn: 'default blur', debounce: {'default': 500, 'blur': 0} }"
getterSetter: boolean value which determines whether or not to treat functions bound to ngModel as getters/setters.
timezone: Defines the timezone to be used to read/write the Date instance in the model for <input type="date">, <input type="time">, ... Right now, the only supported value is 'UTC', otherwise the default timezone of the browser will be used.

ngMousedown

The ngMousedown directive allows you to specify custom behavior on mousedown event.
Arguments:
ngMousedown - {expression} - Expression to evaluate upon mousedown. (Event object is available as $event)

ngMouseenter

Specify custom behavior on mouseenter event.
Arguments:
ngMouseenter - {expression} - Expression to evaluate upon mouseenter. (Event object is available as $event)

ngMouseleave

Specify custom behavior on mouseleave event.
Arguments:
ngMouseleave - {expression} - Expression to evaluate upon mouseleave. (Event object is available as $event)

ngMousemove

Specify custom behavior on mousemove event.
Arguments:
ngMousemove - {expression} - Expression to evaluate upon mousemove. (Event object is available as $event)

ngMouseover

Specify custom behavior on mouseover event.
Arguments:
ngMouseover - {expression} - Expression to evaluate upon mouseover. (Event object is available as $event)

ngMouseup

Specify custom behavior on mouseup event.
Arguments:
ngMouseup - {expression} - Expression to evaluate upon mouseup. (Event object is available as $event)

ngNonBindable

The ngNonBindable directive tells Angular not to compile or bind the contents of the current DOM element. This is useful if the element contains what appears to be Angular directives and bindings but which should be ignored by Angular. This could be the case if you have a site that displays snippets of code, for instance.

ngOpen

The HTML specification does not require browsers to preserve the values of boolean attributes such as open. (Their presence means true and their absence means false.) If we put an Angular interpolation expression into such an attribute then the binding information would be lost when the browser removes the attribute.

ngOpen (cont)

The ngOpen directive solves this problem for the open attribute. This complementary directive is not removed by the browser and so provides a permanent reliable place to store the binding information..
Arguments:
ngOpen - {expression} - If the expression is truthy, then special attribute "open" will be set on the element.

ngPaste

Specify custom behavior on paste event.
Arguments:
ngPaste - {expression} - Expression to evaluate upon paste. (Event object is available as $event)

ngPluralize

ngPluralize is a directive that displays messages according to en-US localization rules. These rules are bundled with angular.js, but can be overridden (see Angular i18n dev guide). You configure ngPluralize directive by specifying the mappings between plural categories and the strings to be displayed.
Arguments:
count - {string | expression} - The variable to be bound to.
when - {string} - The mapping between plural category to its corresponding strings.
offset (optional) - {number} - Offset to deduct from the total number.

ngReadonly

The HTML specification does not require browsers to preserve the values of boolean attributes such as readonly. (Their presence means true and their absence means false.) If we put an Angular interpolation expression into such an attribute then the binding information would be lost when the browser removes the attribute. The ngReadonly directive solves this problem for the readonly attribute. This complementary directive is not removed by the browser and so provides a permanent reliable place to store the binding information.. Arguments:
ngReadonly - {expression} - If the expression is truthy, then special attribute "readonly" will be set on the element

ngRepeat

The ngRepeat directive instantiates a template once per item from a collection. Each template instance gets its own scope, where the given loop variable is set to the current collection item, and $index is set to the item index or key.
Special properties are exposed on the local scope of each template instance, including:
$index - *number* - iterator offset of the repeated element (0..length-1)
$first - *boolean* - true if the repeated element is first in the iterator.
$middle - *boolean* - true if the repeated element is between the first and last in the iterator.
$last - *boolean* - true if the repeated element is last in the iterator.
$even - *boolean* - true if the iterator position $index is even (otherwise false).
$odd - *boolean* - true if the iterator position $index is odd (otherwise false).
Arguments:
ngRepeat - {repeat_expression} - The expression indicating how to enumerate a collection. These formats are currently supported:
variable in expression – where variable is the user defined loop variable and expression is a scope expression giving the collection to enumerate.
For example: album in artist.albums.
(key, value) in expression – where key and value can be any user defined identifiers, and expression is the scope expression giving the collection to enumerate.
For example: (name, age) in {'adam':10, 'amalie':12}.
variable in expression track by tracking_expression – You can also provide an optional tracking function which can be used to associate the objects in the collection with the DOM elements. If no tracking function is specified the ng-repeat associates elements by identity in the collection. It is an error to have more than one tracking function to resolve to the same key. (This would mean that two distinct objects are mapped to the same DOM element, which is not possible.) Filters should be applied to the expression, before specifying a tracking expression.. For example: item in items is equivalent to item in items track by $id(item). This implies that the DOM elements will be associated by item identity in the array.
variable in expression as alias_expression – You can also provide an optional alias expression which will then store the intermediate results of the repeater after the filters have been applied. Typically this is used to render a special message when a filter is active on the repeater, but the filtered result set is empty.

ngRepeat (cont)

For example: item in items | filter:x as results will store the fragment of the repeated items as results, but only after the items have been processed through the filter.
For example: item in items track by $id(item). A built in $id() function can be used to assign a unique $$hashKey property to each item in the array. This property is then used as a key to associated DOM elements with the corresponding item in the array by identity. Moving the same object in array would move the DOM element in the same way in the DOM.
For example: item in items track by item.id is a typical pattern when the items come from the database. In this case the object identity does not matter. Two objects are considered equivalent as long as their id property is same.
For example: item in items | filter:searchText track by item.id is a pattern that might be used to apply a filter to items in conjunction with a tracking expression.

ngSelected

The HTML specification does not require browsers to preserve the values of boolean attributes such as selected. (Their presence means true and their absence means false.) If we put an Angular interpolation expression into such an attribute then the binding information would be lost when the browser removes the attribute. The ngSelected directive solves this problem for the selected attribute. This complementary directive is not removed by the browser and so provides a permanent reliable place to store the binding information.. Arguments:
ngSelected - {expression} - If the expression is truthy, then special attribute "selected" will be set on the element

ngShow

The ngShow directive shows or hides the given HTML element based on the expression provided to the ngShow attribute. The element is shown or hidden by removing or adding the .ng-hide CSS class onto the element. The .ng-hide CSS class is predefined in AngularJS and sets the display style to none (using an !important flag). For CSP mode please add angular-csp.css to your html file (see ngCsp).
Arguments:

ngShow (cont)

ngShow - {expression} - If the expression is truthy then the element is shown or hidden respectively.

input

HTML input element control with angular data-binding.
Arguments:
ngModel(string): Assignable angular expression to data-bind to.
name(optional, string): Property name of the form under which the control is published.
required(optional, string): Sets required validation error key if the value is not entered.
ngRequired(optional, boolean): Sets required attribute if set to true
ngMinlength(optional,number): Sets minlength validation error key if the value is shorter than minlength.
ngMaxlength(optional,number): Sets maxlength validation error key if the value is longer than maxlength.
ngPattern(optional, string): Sets pattern validation error key if the value does not match the RegExp pattern expression. Expected value is /regexp/ for inline patterns or regexp for patterns defined as scope expressions.
ngChange(optional,string) Angular expression to be executed when input changes due to user interaction with the input element.

priority level 0

input[checkbox]

HTML checkbox.
Arguments:
ngModel, *string*: Assignable angular expression to data-bind to.
name(optional), *string*: Property name of the form under which the control is published.
ngTrueValue(optional), *expression*: The value to which the expression should be set when selected.
ngFalseValue(optional), *expression*: The value to which the expression should be set when not selected.
ngChange(optional), *string*: Angular expression to be executed when input changes due to user interaction with the input element.

Priority level 0

input[dateTimeLocal]

Input with datetime validation and transformation. In browsers that do not yet support the HTML5 date input, a text element will be used. In that case, the text must be entered in a valid ISO-8601 local datetime format (yyyy-MM-ddTHH:mm), for example: 2010-12-28T14:57. The model must be a Date object.

ngModel, *string*: Assignable angular expression to data-bind to.

name(optional), *string*: Property name of the form under which the control is published.

min(optional), *string*: Sets the min validation error key if the value entered is less than min. This must be a valid ISO datetime format (yyyy-MM-ddTHH:mm).

max(optional), *string*: Sets the max validation error key if the value entered is greater than max. This must be a valid ISO datetime format (yyyy-MM-ddTHH:mm).

required(optional), *string*: Sets required validation error key if the value is not entered.

ngRequired (optional), *string*: Adds required attribute and required validation constraint to the element when the ngRequired expression evaluates to true. Use ngRequired instead of required when you want to data-bind to the required attribute.

ngChange(optional), *string*: Angular expression to be executed when input changes due to user interaction with the input element.

priority level 0

input[date]

Input with date validation and transformation. In browsers that do not yet support the HTML5 date input, a text element will be used. In that case, text must be entered in a valid ISO-8601 date format (yyyy--MM-dd), for example: 2009-01-06. The model must always be a Date object.

ngModel, *string*: Assignable angular expression to data-bind to.

name(optional), *string*: Property name of the form under which the control is published.

min(optional), *string*: Sets the min validation error key if the value entered is less than min. This must be a valid ISO date string (yyyy--MM-dd).

max(optional), *string*: Sets the max validation error key if the value entered is greater than max. This must be a valid ISO date string (yyyy-MM-dd).

required(optional), *string*: Sets required validation error key if the value is not entered.

ngRequired(optional), *string*: Adds required attribute and required validation constraint to the element when the ngRequired expression evaluates to true. Use ngRequired instead of required when you want to data-bind to the required attribute.

input[date] (cont)

ngChange(optional), *string*: Angular expression to be executed when input changes due to user interaction with the input element.

priority level 0

input[time]

Input with time validation and transformation. In browsers that do not yet support the HTML5 date input, a text element will be used. In that case, the text must be entered in a valid ISO-8601 local time format (HH:mm:ss), for example: 14:57:00. Model must be a Date object. This binding will always output a Date object to the model of January 1, 1970, or local date new Date(1970, 0, 1, HH, mm, ss).

ngModel - {string} - Assignable angular expression to data-bind to.

name (optional) - {string} - Property name of the form under which the control is published.

min (optional) - {string} - Sets the min validation error key if the value entered is less than min. This must be a valid ISO time format (HH:mm:ss).

max (optional) - {string} - Sets the max validation error key if the value entered is greater than max. This must be a valid ISO time format (HH:mm:ss).

required (optional) - {string} - Sets required validation error key if the value is not entered.

ngRequired (optional) - {string} - Adds required attribute and required validation constraint to the element when the ngRequired expression evaluates to true. Use ngRequired instead of required when you want to data-bind to the required attribute.

ngChange (optional) - {string} - Angular expression to be executed when input changes due to user interaction with the input element.

input[url]

Text input with URL validation. Sets the url validation error key if the content is not a valid URL.

ngModel - {string} - Assignable angular expression to data-bind to.

name (optional) - {string} - Property name of the form under which the control is published.

required (optional) - {string} - Sets required validation error key if the value is not entered.

ngRequired (optional) - {string} - Adds required attribute and required validation constraint to the element when the ngRequired expression evaluates to true. Use ngRequired instead of required when you want to data-bind to the required attribute.

input[url] (cont)

ngMinlength (optional) - {number} - Sets minlength validation error key if the value is shorter than minlength.
ngMaxlength (optional) - {number} - Sets maxlength validation error key if the value is longer than maxlength.
ngPattern (optional) - {string} - Sets pattern validation error key if the value does not match the RegExp pattern expression. Expected value is /regexp/ for inline patterns or regexp for patterns defined as scope expressions.
ngChange (optional) - {string} - Angular expression to be executed when input changes due to user interaction with the input element.

input[week]

Input with week-of-the-year validation and transformation to Date. In browsers that do not yet support the HTML5 week input, a text element will be used. In that case, the text must be entered in a valid ISO-8601 week format (yyyy-W##), for example: 2013-W02. The model must always be a Date object.
ngModel - {string} - Assignable angular expression to data-bind to.
name (optional) - {string} - Property name of the form under which the control is published.
min (optional) - {string} - Sets the min validation error key if the value entered is less than min. This must be a valid ISO week format (yyyy-W##).
max (optional) - {string} - Sets the max validation error key if the value entered is greater than max. This must be a valid ISO week format (yyyy-W##).
required (optional) - {string} - Sets required validation error key if the value is not entered.
ngRequired (optional) - {string} - Adds required attribute and required validation constraint to the element when the ngRequired expression evaluates to true. Use ngRequired instead of required when you want to data-bind to the required attribute.
ngChange (optional) - {string} - Angular expression to be executed when input changes due to user interaction with the input element.

input[email]

Text input with email validation. Sets the email validation error key if not a valid email address.
ngModel, *string*: Assignable angular expression to data-bind to.
name(optional), *string*: Property name of the form under which the control is published.

input[email] (cont)

required(optional), *string*: Sets required validation error key if the value is not entered.
ngRequired(optional), *string*: Adds required attribute and required validation constraint to the element when the ngRequired expression evaluates to true. Use ngRequired instead of required when you want to data-bind to the required attribute.
ngMinlength(optional), *number*: Sets minlength validation error key if the value is shorter than minlength.
ngMaxlength(optional), *number*: Sets maxlength validation error key if the value is longer than maxlength.
ngPattern(optional), *string*: Sets pattern validation error key if the value does not match the RegExp pattern expression. Expected value is /regexp/ for inline patterns or regexp for patterns defined as scope expressions.
ngChange(optional), *string*: Angular expression to be executed when input changes due to user interaction with the input element.

priority level 0

input[month]

Input with month validation and transformation. In browsers that do not yet support the HTML5 month input, a text element will be used. In that case, the text must be entered in a valid ISO-8601 month format (yyyy-MM), for example: 2009-01. The model must always be a Date object.
ngModel, *string*: Assignable angular expression to data-bind to.
name(optional), *string*: Property name of the form under which the control is published.
min(optional), *string*: Sets the min validation error key if the value entered is less than min. This must be a valid ISO month format (yyyy-MM).
max(optional), *string*: Sets the max validation error key if the value entered is greater than max. This must be a valid ISO month format (yyyy-MM).
required(optional), *string*: Sets required validation error key if the value is not entered.
ngRequired(optional), *string*: Adds required attribute and required validation constraint to the element when the ngRequired expression evaluates to true. Use ngRequired instead of required when you want to data-bind to the required attribute.
ngChange(optional), *string*: Angular expression to be executed when input changes due to user interaction with the input element.

priority level 0

input[number]

Text input with number validation and transformation. Sets the number validation error if not a valid number.
ngModel, *string*: Assignable angular expression to data-bind to.
name(optional), *string*: Property name of the form under which the control is published.
min(optional), *string*: Sets the min validation error key if the value entered is less than min.
max(optional), *string*: Sets the max validation error key if the value entered is greater than max.
required(optional), *string*: Sets required validation error key if the value is not entered.
ngRequired(optional), *string*: Adds required attribute and required validation constraint to the element when the ngRequired expression evaluates to true. Use ngRequired instead of required when you want to data-bind to the required attribute.
ngMinlength(optional), *number*: Sets minlength validation error key if the value is shorter than minlength.
ngMaxlength(optional), *number*: Sets maxlength validation error key if the value is longer than maxlength.
ngPattern(optional), *string*: Sets pattern validation error key if the value does not match the RegExp pattern expression. Expected value is /regexp/ for inline patterns or regexp for patterns defined as scope expressions.
ngChange(optional), *string*: Angular expression to be executed when input changes due to user interaction with the input element.

input[text]

Standard HTML text input with angular data binding, inherited by most of the input elements.
ngModel - {string} - Assignable angular expression to data-bind to.
name (optional) - {string} - Property name of the form under which the control is published.
required (optional) - {string} - Adds required validation error key if the value is not entered.
ngRequired (optional) - {string} - Adds required attribute and required validation constraint to the element when the ngRequired expression evaluates to true. Use ngRequired instead of required when you want to data-bind to the required attribute.
ngMinlength (optional) - {number} - Sets minlength validation error key if the value is shorter than minlength.
ngMaxlength (optional) - {number} - Sets maxlength validation error key if the value is longer than maxlength.

input[text] (cont)

ngPattern (optional) - {string} - Sets pattern validation error key if the value does not match the RegExp pattern expression. Expected value is /regexp/ for inline patterns or regexp for patterns defined as scope expressions.
ngChange (optional) - {string} - Angular expression to be executed when input changes due to user interaction with the input element.
ngTrim (optional) - {boolean} - If set to false Angular will not automatically trim the input. This parameter is ignored for input[type=password] controls, which will never trim the input. (default: true)

input[radio]

HTML radio button.
ngModel, *string*:Assignable angular expression to data-bind to.
value, *string*: The value to which the expression should be set when selected.
name(optional), *string*: Property name of the form under which the control is published.
ngChange(optional), *string*: Angular expression to be executed when input changes due to user interaction with the input element.
ngValue, *string*: Angular expression which sets the value to which the expression should be set when selected.

priority level 0.

ngDblclick

The ngDblclick directive allows you to specify custom behavior on a dblclick event.
Arguments:
ngDblclick - {expression} - Expression to evaluate upon a dblclick. (The Event object is available as $event)

ngDisabled

The HTML specification does not require browsers to preserve the values of boolean attributes such as disabled. (Their presence means true and their absence means false.) If we put an Angular interpolation expression into such an attribute then the binding information would be lost when the browser removes the attribute. The ngDisabled directive solves this problem for the disabled attribute.

ngDisabled (cont)

This complementary directive is not removed by the browser and so provides a permanent reliable place to store the binding information..
Arguments:
ngDisabled - {expression} - If the expression is truthy, then special attribute "disabled" will be set on the element

ngFocus

Specify custom behavior on focus event.
Arguments:
ngFocus - {expression} - Expression to evaluate upon focus. (Event object is available as $event)

ngForm

Nestable alias of form directive. HTML does not allow nesting of form elements. It is useful to nest forms, for example if the validity of a sub-group of controls needs to be determined.
Arguments:
ngForm | name (optional) - {string} - Name of the form. If specified, the form controller will be published into related scope, under this name.

ngHide

The ngHide directive shows or hides the given HTML element based on the expression provided to the ngHide attribute. The element is shown or hidden by removing or adding the ng-hide CSS class onto the element. The .ng-hide CSS class is predefined in AngularJS and sets the display style to none (using an !important flag). For CSP mode please add angular-csp.css to your html file (see ngCsp).
Arguments:
ngHide - {expression} - If the expression is truthy then the element is shown or hidden respectively.

ngHref

Using Angular markup like {{hash}} in an href attribute will make the link go to the wrong URL if the user clicks it before Angular has a chance to replace the {{hash}} markup with its value. Until Angular replaces the markup the link will be broken and will most likely return a 404 error.
Arguments:
ngHref - {template} - any string which can contain {{}} markup.

ngIf

The ngIf directive removes or recreates a portion of the DOM tree based on an {expression}. If the expression assigned to ngIf evaluates to a false value then the element is removed from the DOM, otherwise a clone of the element is reinserted into the DOM.
Arguments:
ngIf - {expression} - If the expression is falsy then the element is removed from the DOM tree. If it is truthy a copy of the compiled element is added to the DOM tree.

ngInclude

Fetches, compiles and includes an external HTML fragment.
Arguments:
ngInclude | src - {string} - angular expression evaluating to URL. If the source is a string constant, make sure you wrap it in single quotes, e.g. src="'myPartialTemplate.html'".
onload (optional) - {string} - Expression to evaluate when a new partial is loaded.
autoscroll (optional) - {string} - Whether ngInclude should call $anchorScroll to scroll the viewport after the content is loaded.
- If the attribute is not set, disable scrolling.
- If the attribute is set without value, enable scrolling.
- Otherwise enable scrolling only if the expression evaluates to truthy value.
Events:
$includeContentRequested - Emitted every time the ngInclude content is requested.
Type: emit
Target: the scope ngInclude was declared in
$includeContentLoaded - Emitted every time the ngInclude content is reloaded.
Type: emit
Target: the current ngInclude scope
$includeContentError - Emitted when a template HTTP request yields an erronous response (status < 200 || status > 299)
Type: emit
Target: the scope ngInclude was declared in

ngInit

The ngInit directive allows you to evaluate an expression in the current scope.
Arguments:
ngInit - {expression} - Expression to eval.

ngSrc

Using Angular markup like {{hash}} in a src attribute doesn't work right: The browser will fetch from the URL with the literal text {{hash}} until Angular replaces the expression inside {{hash}}. The ngSrc directive solves this problem.
Arguments:
ngSrc - {template} - any string which can contain {{}} markup.

ngSrcset

Using Angular markup like {{hash}} in a srcset attribute doesn't work right: The browser will fetch from the URL with the literal text {{hash}} until Angular replaces the expression inside {{hash}}. The ngSrcset directive solves this problem.
Arguments:
ngSrcset - {template} - any string which can contain {{}} markup.

ngStyle

The ngStyle directive allows you to set CSS style on an HTML element conditionally.
Arguments:
ngStyle - {expression} - Expression which evals to an object whose keys are CSS style names and values are corresponding values for those CSS keys.
Since some CSS style names are not valid keys for an object, they must be quoted. See the 'background-color' style in the example below.

ngSubmit

Enables binding angular expressions to onsubmit events.
Arguments:
ngSubmit - {expression} - Expression to eval. (Event object is available as $event)

ngSwitch

The ngSwitch directive is used to conditionally swap DOM structure on your template based on a scope expression. Elements within ngSwitch but without ngSwitchWhen or ngSwitchDefault directives will be preserved at the location as specified in the template.
Arguments:
ngSwitch | on - {*} - expression to match against ng-switch-when. On child elements add:
ngSwitchWhen: the case statement to match against. If match then this case will be displayed. If the same match appears multiple times, all the elements will be displayed.
ngSwitchDefault: the default case when no other case match. If there are multiple default cases, all of them will be displayed when no other case match.

ngTransclude

Directive that marks the insertion point for the transcluded DOM of the nearest parent directive that uses transclusion.
Any existing content of the element that this directive is placed on will be removed before the transcluded content is inserted.

ngValue

Binds the given expression to the value of input[select] or input[-radio], so that when the element is selected, the ngModel of that element is set to the bound value.
Arguments:
ngValue (optional) - {string} - angular expression, whose value will be bound to the value attribute of the input element

script

Load the content of a <script> element into $templateCache, so that the template can be used by ngInclude, ngView, or directives. The type of the <script> element must be specified as text/ng-template, and a cache name for the template must be assigned through the element's id, which can then be used as a directive's templateUrl.
Arguments:
type - {string} - Must be set to 'text/ng-template'.
id - {string} - Cache name of the template.

select

HTML SELECT element with angular data-binding.
ngOptions:
The ngOptions attribute can be used to dynamically generate a list of <option> elements for the <select> element using the array or object obtained by evaluating the ngOptions comprehension_expression.
Arguments:
ngModel - {string} - Assignable angular expression to data-bind to.
name (optional) - {string} - Property name of the form under which the control is published.
required(optional) - {string} - The control is considered valid only if value is entered.
ngRequired(optional) - {string} - Adds required attribute and required validation constraint to the element when the ngRequired expression evaluates to true. Use ngRequired instead of required when you want to data-bind to the required attribute.
ngOptions(optional) - {comprehension_expression} - in one of the following forms:
for array data sources:
label for value in array
select as label for value in array
label group by group for value in array
*select as label group by group for value in array track by trackexpr
for object data sources:
label for (key , value) in object
select as label for (key , value) in object
label group by group for (key, value) in object
select as label group by group for (key, value) in object
Where:
array / object: an expression which evaluates to an array / object to iterate over.
value: local variable which will refer to each item in the array or each property value of object during iteration.
key: local variable which will refer to a property name in object during iteration.
label: The result of this expression will be the label for <option> element. The expression will most likely refer to the value variable (e.g. value.propertyName).
select: The result of this expression will be bound to the model of the parent <select> element. If not specified, select expression will default to value.
group: The result of this expression will be used to group options using the <optgroup> DOM element.

select (cont)

trackexpr: Used when working with an array of objects. The result of this expression will be used to identify the objects in the array. The trackexpr will most likely refer to the value variable (e.g. value.propertyName).

textarea

HTML textarea element control with angular data-binding. The data-binding and validation properties of this element are exactly the same as those of the input element.
Arguments:
ngModel - {string} - Assignable angular expression to data-bind to.
name (optional) - {string} - Property name of the form under which the control is published.
required(optional) - {string} - Sets required validation error key if the value is not entered.
ngRequired(optional) - {string} - Adds required attribute and required validation constraint to the element when the ngRequired expression evaluates to true. Use ngRequired instead of required when you want to data-bind to the required attribute.
ngMinlength(optional) - {number} - Sets minlength validation error key if the value is shorter than minlength.
ngMaxlength(optional) - {number} - Sets maxlength validation error key if the value is longer than maxlength.
ngPattern (optional) - {string} - Sets pattern validation error key if the value does not match the RegExp pattern expression. Expected value is /regexp/ for inline patterns or regexp for patterns defined as scope expressions.
ngChange (optional) - {string} - Angular expression to be executed when input changes due to user interaction with the input element.
ngTrim (optional) - {boolean} - If set to false Angular will not automatically trim the input. (default: true)