v1.4.5-build.4193 (snapshot

/ API Reference (api) / ng (api/ng)
/ directive components in ng (api/ng/directive)

Show / Hide Table of Contents

Directive components in ng

Name	Description
ngJq (api/ng/directive/ngJq)	Use this directive to force the angular.element library. This should be used to force either jqLite by leaving ng-jq blank or setting the name of the jquery variable under window (eg. jQuery).
ngApp (api/ng/directive/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.
a (api/ng/directive/a)	Modifies the default behavior of the html A tag so that the default action is prevented when the href attribute is empty.
ngHref (api/ng/directive/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. The `ngHref` directive solves this problem.

ngSrc (api/ng/directive/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.
ngSrcset (api/ng/directive/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.
ngDisabled (api/ng/directive/ngDisabled)	This directive sets the `disabled` attribute on the element if the expression (guide/expression) inside `ngDisabled` evaluates to truthy.
ngChecked (api/ng/directive/ngChecked)	Sets the **`checked`** attribute on the element, if the expression inside `ngChecked` is truthy.
ngReadonly (api/ng/directive/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.
ngSelected	

(api/ng/directive/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.
ngOpen (api/ng/directive/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. 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.
ngForm (api/ng/directive/ngForm)	Nestable alias of `form` (api/ng/directive/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.
form (api/ng/directive/form)	Directive that instantiates FormController (api/ng/type/form.FormController).

textarea (api/ng/directive/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 (api/ng/directive/input).
input (api/ng/directive/input)	HTML input element control. When used together with `ngModel` (api/ng/directive/ngModel), it provides data-binding, input state control, and validation. Input control follows HTML5 input types and polyfills the HTML5 validation behavior for older browsers.
ngValue (api/ng/directive/ngValue)	Binds the given expression to the value of **`<option>`** or `input[radio]` (api/ng/input/input[radio]), so that when the element is selected, the `ngModel` (api/ng/directive/ngModel) of that element is set to the bound value.
ngBind (api/ng/directive/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.
ngBindTemplate (api/ng/directive/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.
ngBindHtml (api/ng/directive/ngBindHtml)	

	Evaluates the expression and inserts the resulting HTML into the element in a secure way. By default, the resulting HTML content will be sanitized using the $sanitize (api/ngSanitize/service/$sanitize) service. To utilize this functionality, ensure that `$sanitize` is available, for example, by including `ngSanitize` (api/ngSanitize) in your module's dependencies (not in core Angular). In order to use `ngSanitize` (api/ngSanitize) in your module's dependencies, you need to include "angular-sanitize.js" in your application.
ngChange (api/ng/directive/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).
ngClass (api/ng/directive/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.
ngClassOdd (api/ng/directive/ngClassOdd)	The `ngClassOdd` and `ngClassEven` directives work exactly as ngClass (api/ng/directive/ngClass), except they work in conjunction with `ngRepeat` and take effect only on odd (even) rows.
ngClassEven (api/ng/directive/ngClassEven)	The `ngClassOdd` and `ngClassEven` directives work exactly as ngClass (api/ng/directive/ngClass), except they work in conjunction with `ngRepeat` and take effect only on odd (even) rows.

ngCloak (api/ng/directive/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 (api/ng/directive/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.
ngCsp (api/ng/directive/ngCsp)	Angular has some features that can break certain CSP (Content Security Policy) (https://developer.mozilla.org/en/Security/CSP) rules.
ngClick (api/ng/directive/ngClick)	The ngClick directive allows you to specify custom behavior when an element is clicked.
ngDblclick (api/ng/directive/ngDblclick)	The `ngDblclick` directive allows you to specify custom behavior on a dblclick event.
ngMousedown (api/ng/directive/ngMousedown)	The ngMousedown directive allows you to specify custom behavior on mousedown event.
ngMouseup (api/ng/directive/ngMouseup)	Specify custom behavior on mouseup event.
ngMouseover (api/ng/directive/ngMouseover)	Specify custom behavior on mouseover event.
ngMouseenter (api/ng/directive/ngMouseenter)	Specify custom behavior on mouseenter event.

ngMouseleave (api/ng/directive/ngMouseleave)	Specify custom behavior on mouseleave event.
ngMousemove (api/ng/directive/ngMousemove)	Specify custom behavior on mousemove event.
ngKeydown (api/ng/directive/ngKeydown)	Specify custom behavior on keydown event.
ngKeyup (api/ng/directive/ngKeyup)	Specify custom behavior on keyup event.
ngKeypress (api/ng/directive/ngKeypress)	Specify custom behavior on keypress event.
ngSubmit (api/ng/directive/ngSubmit)	Enables binding angular expressions to onsubmit events.
ngFocus (api/ng/directive/ngFocus)	Specify custom behavior on focus event.
ngBlur (api/ng/directive/ngBlur)	Specify custom behavior on blur event.
ngCopy (api/ng/directive/ngCopy)	Specify custom behavior on copy event.
ngCut (api/ng/directive/ngCut)	Specify custom behavior on cut event.
ngPaste (api/ng/directive/ngPaste)	Specify custom behavior on paste event.
ngIf (api/ng/directive/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.
ngInclude (api/ng/directive/ngInclude)	Fetches, compiles and includes an external HTML fragment.
ngInit (api/ng/directive/ngInit)	The `ngInit` directive allows you to evaluate an expression in the current scope.
ngList (api/ng/directive/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=" | "` .
ngModel (api/ng/directive/ngModel)	The `ngModel` directive binds an `input` , **`select`** , `textarea` (or custom form control) to a property on the scope using NgModelController (api/ng/type/ngModel.NgModelController), which is created and exposed by this directive.
ngModelOptions (api/ng/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.
ngNonBindable (api/ng/directive/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.
ngOptions (api/ng/directive/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.
ngPluralize (api/ng/directive/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 (guide/i18n) dev guide). You configure ngPluralize directive by specifying the mappings between plural categories (http://unicode.org/repos/cldr-tmp/trunk/diff/supplemental/language_plural_rules.html) and the strings to be displayed.
ngRepeat (api/ng/directive/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.
ngShow (api/ng/directive/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 (api/ng/directive/ngCsp)).
ngHide (api/ng/directive/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 (api/ng/directive/ngCsp)).
ngStyle (api/ng/directive/ngStyle)	The `ngStyle` directive allows you to set CSS style on an HTML element conditionally.
ngSwitch (api/ng/directive/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.
ngTransclude (api/ng/directive/ngTransclude)	Directive that marks the insertion point for the transcluded DOM of the nearest parent directive that uses transclusion.
script (api/ng/directive/script)	Load the content of a **`<script>`** element into `$templateCache` (api/ng/service/$templateCache), so that the template can be used by `ngInclude` (api/ng/directive/ngInclude), `ngView` (api/ngRoute/directive/ngView), or directives (guide/directive). 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`.