**ngApp**

1. **- directive in module [ng](https://docs.angularjs.org/api/ng)**

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.

Only one AngularJS application can be auto-bootstrapped per HTML document. The first ngApp found in the document will be used to define the root element to auto-bootstrap as an application. To run multiple applications in an HTML document you must manually bootstrap them using [angular.bootstrap](https://docs.angularjs.org/api/ng/function/angular.bootstrap) instead. AngularJS applications cannot be nested within each other.

You can specify an **AngularJS module** to be used as the root module for the application. This module will be loaded into the [$injector](https://docs.angularjs.org/api/auto/service/$injector)when the application is bootstrapped and should contain the application code needed or have dependencies on other modules that will contain the code.

## Usage

as attribute:

* <ANY
* ng-app=""
* [ng-strict-di=""]>
* ...</ANY>

**Arguments:**

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| ngApp | [**angular.Module**](https://docs.angularjs.org/) | an optional application [module](https://docs.angularjs.org/api/ng/function/angular.module) name to load. |
| ngStrictDi  *(optional)* | [**boolean**](https://docs.angularjs.org/) | 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](https://docs.angularjs.org/guide/di), and useful debugging info will assist in tracking down the root of these bugs. |

**ngBind**

1. **- directive in module [ng](https://docs.angularjs.org/api/ng)**

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.

Typically, you don't use ngBind directly, but instead you use the double curly markup like {{ expression }} which is similar but less verbose.

It is preferable to use ngBind instead of {{ expression }} when a template is momentarily displayed by the browser in its raw state before Angular compiles it. Since ngBind is an element attribute, it makes the bindings invisible to the user while the page is loading.

An alternative solution to this problem would be using the [ngCloak](https://docs.angularjs.org/api/ng/directive/ngCloak) directive.

## Usage

* as attribute:
* <ANY
* ng-bind="">
* ...

</ANY>

* as CSS class:

<ANY class="ng-bind: ;"> ... </ANY>

### Arguments:

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| ngBind | [**expression**](https://docs.angularjs.org/) | [Expression](https://docs.angularjs.org/guide/expression) to evaluate. |

**ngBindHtml**

1. **- directive in module [ng](https://docs.angularjs.org/api/ng)**

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 [$sanitize](https://docs.angularjs.org/api/ngSanitize/service/$sanitize) service. To utilize this functionality, ensure that $sanitize is available, for example, by including [ngSanitize](https://docs.angularjs.org/api/ngSanitize) in your module's dependencies (not in core Angular.) You may also bypass sanitization for values you know are safe. To do so, bind to an explicitly trusted value via [$sce.trustAsHtml](https://docs.angularjs.org/api/ng/service/$sce#trustAsHtml). See the example under [Strict Contextual Escaping (SCE)](https://docs.angularjs.org/api/ng/service/$sce#Example).

Note: If a $sanitize service is unavailable and the bound value isn't explicitly trusted, you will have an exception (instead of an exploit.)

## Usage

* as attribute:
* <ANY
* ng-bind-html="">
* ...

</ANY>

### Arguments

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| ngBindHtml | [**expression**](https://docs.angularjs.org/) | [Expression](https://docs.angularjs.org/guide/expression) to evaluate. |

**ngBindTemplate**

1. **- directive in module [ng](https://docs.angularjs.org/api/ng)**

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.

## Usage

* as attribute:
* <ANY
* ng-bind-template="">
* ...

</ANY>

### Arguments

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| ngBindTemplate | [**string**](https://docs.angularjs.org/) | template of form {{ expression }} to eval. |

**ngBlur**

1. **- directive in module [ng](https://docs.angularjs.org/api/ng)**

Specify custom behavior on blur event.

**Usage**

as attribute:

* <window, input, select, textarea, a
* ng-blur="">
* ...

</window, input, select, textarea, a>

### Arguments

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| ngBlur | [**expression**](https://docs.angularjs.org/) | [Expression](https://docs.angularjs.org/guide/expression) to evaluate upon blur. ([Event object is available as $event](https://docs.angularjs.org/guide/expression#-event-)) |

**ngChange**

1. **- directive in module [ng](https://docs.angularjs.org/api/ng)**

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). The expression is not evaluated when the value change is coming from the model.

Note, this directive requires ngModel to be present.

## Usage

* as attribute:
* <input
* ng-change="">
* ...

</input>

### Arguments

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| ngChange | [**expression**](https://docs.angularjs.org/) | [Expression](https://docs.angularjs.org/guide/expression) to evaluate upon change in input value. |

**ngChecked**

1. **- directive in module [ng](https://docs.angularjs.org/api/ng)**

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.

## Usage

* as attribute:
* <INPUT
* ng-checked="">
* ...

</INPUT>

### Arguments

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| ngChecked | [**expression**](https://docs.angularjs.org/) | If the [expression](https://docs.angularjs.org/guide/expression) is truthy, then special attribute "checked" will be set on the element |

**ngClass**

1. **- directive in module [ng](https://docs.angularjs.org/api/ng)**

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:

1. If the expression evaluates to a string, the string should be one or more space-delimited class names.
2. 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.
3. 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.

## Usage

* as attribute:
* <ANY
* ng-class="">
* ...

</ANY>

* as CSS class:

<ANY class="ng-class: ;"> ... </ANY>

## Animations

add - happens just before the class is applied to the element remove - happens just before the class is removed from the element

[Click here](https://docs.angularjs.org/api/ngAnimate/service/$animate) to learn more about the steps involved in the animation.

### Arguments

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| ngClass | [**expression**](https://docs.angularjs.org/) | [Expression](https://docs.angularjs.org/guide/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**

1. **- directive in module [ng](https://docs.angularjs.org/api/ng)**

The ngClassOdd and ngClassEven directives work exactly as [ngClass](https://docs.angularjs.org/api/ng/directive/ngClass), except they work in conjunction withngRepeat and take effect only on odd (even) rows.

This directive can be applied only within the scope of an [ngRepeat](https://docs.angularjs.org/api/ng/directive/ngRepeat).

## Usage

* as attribute:
* <ANY
* ng-class-even="">
* ...

</ANY>

* as CSS class:

<ANY class="ng-class-even: ;"> ... </ANY>

### Arguments

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| ngClassEven | [**expression**](https://docs.angularjs.org/) | [Expression](https://docs.angularjs.org/guide/expression) to eval. The result of the evaluation can be a string representing space delimited class names or an array. |

**ngClassOdd**

1. **- directive in module [ng](https://docs.angularjs.org/api/ng)**

The ngClassOdd and ngClassEven directives work exactly as [ngClass](https://docs.angularjs.org/api/ng/directive/ngClass), except they work in conjunction withngRepeat and take effect only on odd (even) rows.

This directive can be applied only within the scope of an [ngRepeat](https://docs.angularjs.org/api/ng/directive/ngRepeat).

## Usage

* as attribute:
* <ANY
* ng-class-odd="">
* ...

</ANY>

* as CSS class:

<ANY class="ng-class-odd: ;"> ... </ANY>

### Arguments

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| ngClassOdd | [**expression**](https://docs.angularjs.org/) | [Expression](https://docs.angularjs.org/guide/expression) to eval. The result of the evaluation can be a string representing space delimited class names or an array. |

**ngCloak**

1. **- directive in module [ng](https://docs.angularjs.org/api/ng)**

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.

The directive can be applied to the <body> element, but the preferred usage is to apply multiple ngCloak directives to small portions of the page to permit progressive rendering of the browser view.

ngCloak works in cooperation with the following css rule embedded within angular.js and angular.min.js. For CSP mode please addangular-csp.css to your html file (see [ngCsp](https://docs.angularjs.org/api/ng/directive/ngCsp)).

[ng\:cloak], [ng-cloak], [data-ng-cloak], [x-ng-cloak], .ng-cloak, .x-ng-cloak {

display: none !important;

}

When this css rule is loaded by the browser, all html elements (including their children) that are tagged with the ngCloak directive are hidden. When Angular encounters this directive during the compilation of the template it deletes the ngCloak element attribute, making the compiled element visible.

For the best result, the angular.js script must be loaded in the head section of the html document; alternatively, the css rule above must be included in the external stylesheet of the application.

Legacy browsers, like IE7, do not provide attribute selector support (added in CSS 2.1) so they cannot match the [ng\:cloak] selector. To work around this limitation, you must add the css class ng-cloak in addition to the ngCloak directive as shown in the example below.

**Usage**

* as attribute:
* <ANY>
* ...

</ANY>

* as CSS class:

<ANY class=""> ... </ANY>