UI Bootstrap

Bootstrap components written in pure AngularJS (http://angularjs.org) by the AngularUI Team (http://angular-ui.github.io)

Code on Github (https://github.com/angular-ui/bootstrap)

Download (1.3.1) Create a Build

Star 12,277 Fork 6,324 Tweet G+1 1.845

Getting started

Dependencies

This repository contains a set of **native AngularJS directives** based on Bootstrap's markup and CSS. As a result no dependency on jQuery or Bootstrap's JavaScript is required. The **only required dependencies** are:

- AngularJS (http://angularjs.org) (requires AngularJS 1.4.x or higher, tested with 1.5.3). 0.14.3 is the last version of this library that supports AngularJS 1.3.x and 0.12.0 is the last version that supports AngularJS 1.2.x.
- Angular-animate (http://angularjs.org) (the version should match with your angular's, tested with 1.5.3) if you

plan in using animations, you need to load angular-animate as well.
- Angular-touch (http://angularjs.org) (the version should match with your angular's, tested with 1.5.3) if you plan in using swipe actions, you need to load angular-touch as well.
- Bootstrap CSS (http://getbootstrap.com) (tested with version 3.3.6). This version of the library (1.3.1) works only with Bootstrap CSS in version 3.x. 0.8.0 is the last version of this library that supports Bootstrap CSS in version 2.3.x.

Files to download

Build files for all directives are distributed in several flavours: minified for production usage, un-minified for development, with or without templates. All the options are described and can be downloaded from here (https://github.com/angular-ui/bootstrap/tree/gh-pages). It should be noted that the `-tpls` files contain the templates bundled in JavaScript, while the regular version does not contain the bundled templates. For more information, check out the FAQ here (https://github.com/angular-ui/bootstrap/wiki/FAQ#full-explanation) and the README here (https://github.com/angular-ui/bootstrap/tree/gh-pages#build-files).

Alternativelly, if you are only interested in a subset of directives, you can create your own build.

Whichever method you choose the good news that the overall size of a download is very small: <76kB for all directives (~20kB with gzip compression!)

Installation

As soon as you've got all the files downloaded and included in your page you just need to declare a dependency on the `ui.bootstrap` module (http://docs.angularjs.org/guide/module):

```
angular.module('myModule', ['ui.bootstrap']);
```

If you are using UI Bootstrap in the CSP mode, e.g. in an extension, make sure you link to the `ui-bootstrap-csp.css` in your HTML manually.

You can fork one of the plunkers from this page to see a working example of what is described here.

Migration to prefixes

Since version 0.14.0 we started to prefix all our components. If you are upgrading from ui-bootstrap 0.13.4 or earlier, check our migration guide (https://github.com/angular-ui/bootstrap/wiki/Migration-guide-for-prefixes).

CSS

Original Bootstrap's CSS depends on empty `href` attributes to style cursors for several components (pagination, tabs etc.). But in AngularJS adding empty `href` attributes to link tags will cause unwanted route changes. This is why we need to remove empty `href` attributes from directive templates and as a result styling is not applied correctly. The remedy is simple, just add the following styling to your application:

```
.nav, .pagination, .carousel, .panel-title a { cursor: pointer; }
```

FAQ

Please check our FAQ section (https://github.com/angular-ui/bootstrap/wiki/FAQ) for common problems / solutions.

Reading the documentation

Each of the components provided in `ui-bootstrap` have documentation and interactive Plunker examples.

For the directives, we list the different attributes with their default values. In addition to this, some settings have a badge on it:

- 👁 - This setting has an angular $watch listener applied to it.
- **B** - This setting is a boolean. It doesn't need a parameter.
- **C** - This setting can be configured globally in a constant service*.
- **$** - This setting expects an angular expression instead of a literal string. If the expression support a boolean / integer, you can pass it directly.
- **readonly** - This setting is readonly.

For the services (you will recognize them with the `$` prefix), we list all the possible parameters you can pass to them and their default values if any.

* Some directives have a config service that follows the next pattern: `uibDirectiveConfig`. The service's settings use camel case. The services can be configured in a `.config` function for example.

Accordion (ui.bootstrap.accordion (https://github.com/angular-ui/bootstrap/tree/master/src/accordion))

Toggle last panel | Enable / Disable first panel

✓ Open only one at a time

Static Header, initially expanded ()

This content is straight in the template.

Dynamic Group Header - 1 ()

Dynamic Group Header - 2 ()

Dynamic Body Content ()

Custom template ()

Delete account ()

I can have markup, too! () ›

The **accordion directive** builds on top of the collapse directive to provide a list of items, with collapsible bodies that are collapsed or expanded by clicking on the item's header.

The body of each accordion group is transcluded into the body of the collapsible element.

uib-accordion settings

- `close-others` **$** **C** *(Default: `true`)* - Control whether expanding an item will cause the other items to close.
- `template-url` *(Default: `template/accordion/accordion.html`)* - Add the ability to override the template used on the component.

uib-accordion-group settings

- `heading` *(Default: `none`)* - The clickable text on the group's header. You need one to be able to click on the header for toggling.
- `is-disabled` **$** 👁 *(Default: `false`)* - Whether the accordion group is disabled or not.
- `is-open` **$** 👁 *(Default: `false`)* - Whether accordion group is open or closed.
- `panel-class` 👁 *(Default: `panel-default`)* - Add ability to use Bootstrap's contextual panel classes (panel-primary, panel-success, panel-info, etc...) or your own. This must be a string.
- `template-url` *(Default: `uib/template/accordion/accordion-group.html`)* - Add the ability to override the template used on the component.

Accordion heading

Instead of the `heading` attribute on the `uib-accordion-group`, you can use an `uib-accordion-heading` element inside a group that will be used as the group's header.

If you're using a custom template for the `uib-accordion-group`, you'll need to have an element for the heading to be transcluded into using `uib-accordion-header` (e.g. `<div uib-accordion-header></div>`).

Known issues

To use clickable elements within the accordion, you have override the accordion-group template to use div elements instead of anchor elements, and add `cursor: pointer` in your CSS. This is due to browsers interpreting anchor elements as the target of any click event, which triggers routing when certain elements such as buttons are nested inside the anchor element.

If custom classes on the accordion-group element are desired, one needs to either modify the template to remove the `ng-class` usage in the accordion-group template and use ng-class on the accordion-group element (not recommended), or use an interpolated expression in the class attribute, i.e. `<uib-accordion-group class="{{customClass()}}"></uib-accordion-group>`.

Markup () JavaScript () Edit in plunker

```
<div ng-controller="AccordionDemoCtrl">
  <script type="text/ng-template" id="group-template.html">
    <div class="panel {{panelClass || 'panel-default'}}">
      <div class="panel-heading">
        <h4 class="panel-title" style="color:#fa39c3">
          <a href tabindex="0" class="accordion-toggle" ng-click="toggleOpen()" uib-acc
ordion-transclude="heading"><span
            ng-class="{'text-muted': isDisabled}">{{heading}}</span></a>
        </h4>
      </div>
      <div class="panel-collapse collapse" uib-collapse="!isOpen">
        <div class="panel-body" style="text-align: right" ng-transclude></div>
      </div>
    </div>
  </script>

  <p>
    <button type="button" class="btn btn-default btn-sm" ng-click="status.open = !statu
s.open">Toggle last panel</button>
    <button type="button" class="btn btn-default btn-sm" ng-click="status.isFirstDisabl
ed = ! status.isFirstDisabled">Enable / Disable first panel</button>
  </p>

  <div class="checkbox">
    <label>
      <input type="checkbox" ng-model="oneAtATime">
      Open only one at a time
    </label>
  </div>
  <uib-accordion close-others="oneAtATime">
    <uib-accordion-group heading="Static Header, initially expanded" is-open="status.is
FirstOpen" is-disabled="status.isFirstDisabled">
      This content is straight in the template.
    </uib-accordion-group>
    <uib-accordion-group heading="{{group.title}}" ng-repeat="group in groups">
      {{group.content}}
    </uib-accordion-group>
    <uib-accordion-group heading="Dynamic Body Content">
      <p>The body of the uib-accordion group grows to fit the contents</p>
      <button type="button" class="btn btn-default btn-sm" ng-click="addItem()">Add Ite
m</button>
      <div ng-repeat="item in items">{{item}}</div>
    </uib-accordion-group>
    <uib-accordion-group heading="Custom template" template-url="group-template.html">
      Hello
    </uib-accordion-group>
    <uib-accordion-group heading="Delete account" panel-class="panel-danger">
      <p>Please, to delete your account, click the button below</p>
      <button class="btn btn-danger">Delete</button>
    </uib-accordion-group>
    <uib-accordion-group is-open="status.open">
      <uib-accordion-heading>
```

```
        I can have markup, too! <i class="pull-right glyphicon" ng-class="{'glyphicon-c
hevron-down': status.open, 'glyphicon-chevron-right': !status.open}"></i>
      </uib-accordion-heading>
      This is just some content to illustrate fancy headings.
    </uib-accordion-group>
  </uib-accordion>
</div>
```

Alert (ui.bootstrap.alert (https://github.com/angular-ui/bootstrap/tree/master/src/alert))

Oh snap! Change a few things up and try submitting again. ×

Well done! You successfully read this important alert message. ×

A happy alert!

Add Alert

This directive can be used both to generate alerts from static and dynamic model data (using the `ng-repeat` directive).

uib-alert settings

- `close()` **$** - A callback function that gets fired when an `alert` is closed. If the attribute exists, a close button is displayed as well.
- `dismiss-on-timeout` *(Default: `none`)* - Takes the number of milliseconds that specify the timeout duration, after which the alert will be closed. This attribute requires the presence of the `close` attribute.
- `template-url` *(Default: `uib/template/alert/alert.html`)* - Add the ability to override the template used in the component.
- `type` *(Default: `warning`)* - Defines the type of the alert. Go to bootstrap page (http://getbootstrap.com/components/#alerts) to see the type of alerts available.

Markup () JavaScript () Edit in plunker

```
<div ng-controller="AlertDemoCtrl">
  <script type="text/ng-template" id="alert.html">
    <div class="alert" style="background-color:#fa39c3;color:white" role="alert">
      <div ng-transclude></div>
    </div>
  </script>

  <uib-alert ng-repeat="alert in alerts" type="{{alert.type}}" close="closeAlert($index
)">{{alert.msg}}</uib-alert>
  <uib-alert template-url="alert.html">A happy alert!</uib-alert>
  <button type="button" class='btn btn-default' ng-click="addAlert()">Add Alert</button
>
</div>
```

Buttons (ui.bootstrap.buttons (https://github.com/angular-ui/bootstrap/tree/master/src/buttons))

Single toggle

```
1
```

Single Toggle

Checkbox

```
Model: {"left":false,"middle":true,"right":false}
```

```
Results: ["middle"]
```

Left Middle Right

Radio & Uncheckable Radio

```
Middle
```

Left Middle Right Left Middle Right

Toggle uncheckable

With the buttons directive, we can make a group of buttons behave like a set of checkboxes (`uib-btn-checkbox`) or behave like a set of radio buttons (`uib-btn-radio`).

uib-btn-checkbox settings

- `btn-checkbox-false` *(Default: `false`)* - Sets the value for the unchecked status.
- `btn-checkbox-true` *(Default: `true`)* - Sets the value for the checked status.
- `ng-model` **$** 👁 - Model where we set the checkbox status. By default `true` or `false` .

uib-btn-radio settings

- `ng-model` **$** 👁 - Model where we set the radio status. All radio buttons in a group should use the same `ng-model` .
- `uib-btn-radio` - **$** Value to assign to the `ng-model` if we check this radio button.
- `uib-uncheckable` **$** *(Default: `null`)* - An expression that evaluates to a truthy or falsy value that determines whether the `uncheckable` attribute is present.
- `uncheckable` **B** - Whether a radio button can be unchecked or not.

Additional settings `uibButtonConfig`

- `activeClass` *(Default: `active`)* - Class to apply to the checked buttons.
- `toggleEvent` *(Default: `click`)* - Event used to toggle the buttons.

Markup () JavaScript () Edit in plunker

```
<div ng-controller="ButtonsCtrl">
    <h4>Single toggle</h4>
    <pre>{{singleModel}}</pre>
    <button type="button" class="btn btn-primary" ng-model="singleModel" uib-btn-checkb
ox btn-checkbox-true="1" btn-checkbox-false="0">
        Single Toggle
    </button>
    <h4>Checkbox</h4>
    <pre>Model: {{checkModel}}</pre>
    <pre>Results: {{checkResults}}</pre>
    <div class="btn-group">
        <label class="btn btn-primary" ng-model="checkModel.left" uib-btn-checkbox>Left
</label>
        <label class="btn btn-primary" ng-model="checkModel.middle" uib-btn-checkbox>Mi
ddle</label>
        <label class="btn btn-primary" ng-model="checkModel.right" uib-btn-checkbox>Rig
ht</label>
    </div>
    <h4>Radio &amp; Uncheckable Radio</h4>
    <pre>{{radioModel || 'null'}}</pre>
    <div class="btn-group">
        <label class="btn btn-primary" ng-model="radioModel" uib-btn-radio="'Left'">Lef
t</label>
        <label class="btn btn-primary" ng-model="radioModel" uib-btn-radio="'Middle'">M
iddle</label>
        <label class="btn btn-primary" ng-model="radioModel" uib-btn-radio="'Right'">Ri
ght</label>
    </div>
    <div class="btn-group">
        <label class="btn btn-success" ng-model="radioModel" uib-btn-radio="'Left'" unc
heckable>Left</label>
        <label class="btn btn-success" ng-model="radioModel" uib-btn-radio="'Middle'" u
ncheckable>Middle</label>
        <label class="btn btn-success" ng-model="radioModel" uib-btn-radio="'Right'" ui
b-uncheckable="uncheckable">Right</label>
    </div>
    <div>
        <button class="btn btn-default" ng-click="uncheckable = !uncheckable">
            Toggle uncheckable
        </button>
    </div>
</div>
```

Carousel (ui.bootstrap.carousel (https://github.com/angular-ui/bootstrap/tree/master/src/carousel))

Add Slide Randomize slides

Disable Slide Looping

Interval, in milliseconds:

5000

Enter a negative number or 0 to stop the interval.

Carousel creates a carousel similar to bootstrap's image carousel.

The carousel also offers support for touchscreen devices in the form of swiping. To enable swiping, load the `ngTouch` module as a dependency.

Use a `<uib-carousel>` element with `<uib-slide>` elements inside it.

uib-carousel settings

- `active` *(Default: `Index of first slide`)* - Index of current active slide.
- `interval` $ *(Default: `none`)* - Sets an interval to cycle through the slides. You need a number bigger than 0 to make the interval work.
- `no-pause` $ *(Default: `false`)* - The interval pauses on mouseover. Setting this to truthy, disables this pause.
- `no-transition` $ *(Default: `false`)* - Whether to disable the transition animation between slides. Setting this to truthy, disables this transition.
- `no-wrap` $ *(Default: `false`)* - Disables the looping of slides. Setting `no-wrap` to an expression which evaluates to a truthy value will prevent looping.
- `template-url` *(Default: `uib/template/carousel/carousel.html`)* - Add the ability to override the template used on the component.

uib-slide settings

- `actual` $ *(Default: `none`)* - Use this attribute to bind the slide model (or any object of interest) onto the slide scope, which makes it available for customization in the carousel template.

- `index` $ 👁 *(Default:* `none` *)* - The index of the slide. Must be unique.
- `template-url` *(Default:* `uib/template/carousel/slide.html` *)* - Add the ability to override the template used on the component.

Markup () JavaScript () Edit in plunker

```
<div ng-controller="CarouselDemoCtrl">
  <div style="height: 305px">
    <uib-carousel active="active" interval="myInterval" no-wrap="noWrapSlides">
      <uib-slide ng-repeat="slide in slides track by slide.id" index="slide.id">
        <img ng-src="{{slide.image}}" style="margin:auto;">
        <div class="carousel-caption">
          <h4>Slide {{slide.id}}</h4>
          <p>{{slide.text}}</p>
        </div>
      </uib-slide>
    </uib-carousel>
  </div>
  <div class="row">
    <div class="col-md-6">
      <button type="button" class="btn btn-info" ng-click="addSlide()">Add Slide</butto
n>
      <button type="button" class="btn btn-info" ng-click="randomize()">Randomize slide
s</button>
      <div class="checkbox">
        <label>
          <input type="checkbox" ng-model="noWrapSlides">
          Disable Slide Looping
        </label>
      </div>
    </div>
    <div class="col-md-6">
      Interval, in milliseconds: <input type="number" class="form-control" ng-model="my
Interval">
      <br />Enter a negative number or 0 to stop the interval.
    </div>
  </div>
</div>
```

Collapse (ui.bootstrap.collapse (https://github.com/angular-ui/bootstrap/tree/master/src/collapse))

Toggle collapse

Some content

uib-collapse provides a simple way to hide and show an element with a css transition

uib-collapse settings

- `collapsed()` $ - An optional expression called after the element finished collapsing.
- `collapsing()` $ - An optional expression called before the element begins collapsing. If the expression returns a promise, animation won't start until the promise resolves. If the returned promise is rejected, collapsing will be cancelled.
- `expanded()` $ - An optional expression called after the element finished expanding.
- `expanding()` $ - An optional expression called before the element begins expanding. If the expression returns a promise, animation won't start until the promise resolves. If the returned promise is rejected, expanding will be cancelled.
- `uib-collapse` $ *(Default: `false`)* - Whether the element should be collapsed or not.

Markup () JavaScript () Edit in plunker

```
<div ng-controller="CollapseDemoCtrl">
	<button type="button" class="btn btn-default" ng-click="isCollapsed = !isCollap
sed">Toggle collapse</button>
	<hr>
	<div uib-collapse="isCollapsed">
		<div class="well well-lg">Some content</div>
	</div>
</div>
```

Dateparser (ui.bootstrap.dateparser (https://github.com/angular-ui/bootstrap/tree/master/src/dateparser))

Formatting codes playground

Define your format

yyyy/MM/dd

Result

2016/05/06

The `uibDateParser` is what the `uib-datepicker` uses internally to parse the dates. You can use it standalone by injecting the `uibDateParser` service where you need it.

The public API for the dateParser is a single method called `parse`.

Certain format codes support i18n. Check this guide (https://docs.angularjs.org/guide/i18n) for more information.

uibDateParser's parse function

parameters

- `input` *(Type: `string`, Example: `2004/Sep/4`)* - The input date to parse.
- `format` *(Type: `string`, Example: `yyyy/MMM/d`)* - The format we want to use. Check all the supported formats below.
- `baseDate` *(Type: `Date`, Example: `new Date()`)* - If you want to parse a date but maintain the timezone, you can pass an existing date here.

return

- If the specified input matches the format, a new date with the input will be returned, otherwise, it will return undefined.

uibDateParser's format codes

- `yyyy` *(Example: `2015`)* - Parses a 4 digits year.
- `yy` *(Example: `15`)* - Parses a 2 digits year.
- `y` *(Example: `15`)* - Parses a year with 1, 2, 3, or 4 digits.
- `MMMM` *(Example: `February`, i18n support)* - Parses the full name of a month.
- `MMM` *(Example: `Feb`, i18n support)* - Parses the short name of a month.
- `MM` *(Example: `12`, Leading 0)* - Parses a numeric month.
- `M` *(Example: `3`)* - Parses a numeric month.
- `M!` *(Example: `3` or `03`)* - Parses a numeric month, but allowing an optional leading zero
- `dd` *(Example: `05`, Leading 0)* - Parses a numeric day.
- `d` *(Example: `5`)* - Parses a numeric day.
- `d!` *(Example: `3` or `03`)* - Parses a numeric day, but allowing an optional leading zero
- `EEEE` *(Example: `Sunday`, i18n support)* - Parses the full name of a day.
- `EEE` *(Example: `Mon`, i18n support)* - Parses the short name of a day.
- `HH` *(Example: `14`, Leading 0)* - Parses a 24 hours time.
- `H` *(Example: `3`)* - Parses a 24 hours time.

- `hh` *(Example: `11` , Leading 0)* - Parses a 12 hours time.
- `h` *(Example: `3`)* - Parses a 12 hours time.
- `mm` *(Example: `09` , Leading 0)* - Parses the minutes.
- `m` *(Example: `3`)* - Parses the minutes.
- `sss` *(Example: `094` , Leading 0)* - Parses the milliseconds.
- `ss` *(Example: `08` , Leading 0)* - Parses the seconds.
- `s` *(Example: `22`)* - Parses the seconds.
- `a` *(Example: `10AM`)* - Parses a 12 hours time with AM/PM.
- `Z` *(Example: `-0800`)* - Parses the timezone offset in a signed 4 digit representation
- `ww` *(Example: `03` , Leading 0)* - Parses the week number
- `w` *(Example: `03`)* - Parses the week number
- `G` , `GG` , `GGG` *(Example: `AD`)* - Parses the era (`AD` or `BC`)
- `GGGG` *(Example: `Anno Domini`)* - Parses the long form of the era (`Anno Domini` or `Before Christ`)

* The ones marked with `Leading 0` , needs a leading 0 for values less than 10. Exception being milliseconds which needs it for values under 100.

** It also supports `fullDate|longDate|medium|mediumDate|mediumTime|short|shortDate|shortTime` as the format for parsing.

*** It supports template literals as a string between the single quote `'` character, i.e. `'The Date is' MM/DD/YYYY` . If one wants the literal single quote character, one must use `''''` .

Markup () JavaScript () Edit in plunker

```
<div ng-controller="DateParserDemoCtrl">
  <h4>Formatting codes playground</h4>
  <p class="form-group">
    <label>Define your format</label>
    <input type="text" ng-model="format" class="form-control">
  </p>
  <p class="form-group">
    <label>Result</label>
    <input type="text" class="form-control" uib-datepicker-popup="{{format}}" ng-model=
"date" />
  </p>
</div>
```

Datepicker (ui.bootstrap.datepicker (https://github.com/angular-

ui/bootstrap/tree/master/src/datepicker))

```
Selected date is: Friday, May 6, 2016
```

Inline

‹ **May 2016** ›

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
18	01	02	03	04	05	06	07
19	08	09	10	11	12	13	14
20	15	16	17	18	19	20	21
21	22	23	24	25	26	27	28
22	29	30	31	01	02	03	04
23	05	06	07	08	09	10	11

Popup

06-May-2016

2016-05-06

Format: (manual alternate *M!/d!/yyyy*)

dd-MMMM-yyyy

Today | 2009-08-24 | Clear | Min date

Our datepicker is flexible and fully customizable.

You can navigate through days, months and years.

The datepicker has 3 modes:

- `day` - In this mode you're presented with a 6-week calendar for a specified month.
- `month` - In this mode you can select a month within a selected year.
- `year` - In this mode you are presented with a range of years (20 by default).

uib-datepicker settings

- `ng-model` **$** - The date object. Needs to be a Javascript Date object.
- `ng-model-options` **$** **C** *(Default:* `{}` *)* - Supported angular ngModelOptions (https://docs.angularjs.org

/api/ng/directive/ngModelOptions):

  - allowInvalid
  - timezone
- `template-url` *(Default: `uib/template/datepicker/datepicker.html`)* - Add the ability to override the template used on the component.

Apart from the previous settings, to configure the uib-datepicker you need to create an object in Javascript with all the options and use it on the `datepicker-options` attribute:

- `datepicker-options` **$** - An object to configure the datepicker in one place.
  - `customClass (date, mode)` - An optional expression to add classes based on passing a date and current mode.
  - `dateDisabled (date, mode)` - An optional expression to disable visible options based on passing a date and current mode.
  - `datepickerMode` **C** 👁 *(Default: `day`)* - Current mode of the datepicker *(day|month|year)*. Can be used to initialize the datepicker in a specific mode.
  - `formatDay` **C** *(Default: `dd`)* - Format of day in month.
  - `formatMonth` **C** *(Default: `MMMM`)* - Format of month in year.
  - `formatYear` **C** *(Default: `yyyy`)* - Format of year in year range.
  - `formatDayHeader` **C** *(Default: `EEE`)* - Format of day in week header.
  - `formatDayTitle` **C** *(Default: `MMMM yyyy`)* - Format of title when selecting day.
  - `formatMonthTitle` **C** *(Default: `yyyy`)* - Format of title when selecting month.
  - `initDate` 👁 *(Default: `null`)* - The initial date view when no model value is specified.
  - `maxDate` **C** 👁 *(Default: `null`)* - Defines the maximum available date. Requires a Javascript Date object.
  - `maxMode` **C** 👁 *(Default: `year`)* - Sets an upper limit for mode.
  - `minDate` **C** 👁 *(Default: `null`)* - Defines the minimum available date. Requires a Javascript Date object.
  - `minMode` **C** 👁 *(Default: `day`)* - Sets a lower limit for mode.
  - `shortcutPropagation` **C** *(Default: `false`)* - An option to disable the propagation of the keydown event.
  - `showWeeks` **C** *(Default: `true`)* - Whether to display week numbers.
  - `startingDay` **C** *(Default: `$locale.DATETIME_FORMATS.FIRSTDAYOFWEEK`)* - Starting day of the week from 0-6 (0=Sunday, ..., 6=Saturday).
  - `yearRows` **C** *(Default: `4`)* - Number of rows displayed in year selection.
  - `yearColumns` **C** *(Default: `5`)* - Number of columns displayed in year selection.

Keyboard support

Depending on datepicker's current mode, the date may refer either to day, month or year. Accordingly, the term view refers either to a month, year or year range.

- `Left` : Move focus to the previous date. Will move to the last date of the previous view, if the current date is the first date of a view.
- `Right` : Move focus to the next date. Will move to the first date of the following view, if the current date is the last date of a view.
- `Up` : Move focus to the same column of the previous row. Will wrap to the appropriate row in the previous view.
- `Down` : Move focus to the same column of the following row. Will wrap to the appropriate row in the following view.
- `PgUp` : Move focus to the same date of the previous view. If that date does not exist, focus is placed on the last date of the month.
- `PgDn` : Move focus to the same date of the following view. If that date does not exist, focus is placed on the last date of the month.
- `Home` : Move to the first date of the view.
- `End` : Move to the last date of the view.
- `Enter` / `Space` : Select date.
- `Ctrl` + `Up` : Move to an upper mode.
- `Ctrl` + `Down` : Move to a lower mode.
- `Esc` : Will close popup, and move focus to the input.

Notes

If the date a user enters falls outside of the min-/max-date range, a `dateDisabled` validation error will show on the form.

Markup () JavaScript () Edit in plunker

```
<style>
  .full button span {
    background-color: limegreen;
    border-radius: 32px;
    color: black;
  }
  .partially button span {
    background-color: orange;
    border-radius: 32px;
    color: black;
  }
</style>
<div ng-controller="DatepickerDemoCtrl">
    <pre>Selected date is: <em>{{dt | date:'fullDate' }}</em></pre>

    <h4>Inline</h4>
    <div style="display:inline-block; min-height:290px;">
      <uib-datepicker ng-model="dt" class="well well-sm" datepicker-options="inlineOpti
ons"></uib-datepicker>
    </div>

    <h4>Popup</h4>
    <div class="row">
      <div class="col-md-6">
        <p class="input-group">
          <input type="text" class="form-control" uib-datepicker-popup="{{format}}" ng-
model="dt" is-open="popup1.opened" datepicker-options="dateOptions" ng-required="true"
close-text="Close" alt-input-formats="altInputFormats" />
          <span class="input-group-btn">
            <button type="button" class="btn btn-default" ng-click="open1()"><i class="
glyphicon glyphicon-calendar"></i></button>
          </span>
        </p>
      </div>

      <div class="col-md-6">
        <p class="input-group">
          <input type="text" class="form-control" uib-datepicker-popup ng-model="dt" is
-open="popup2.opened" datepicker-options="dateOptions" ng-required="true" close-text="C
lose" />
          <span class="input-group-btn">
            <button type="button" class="btn btn-default" ng-click="open2()"><i class="
glyphicon glyphicon-calendar"></i></button>
          </span>
        </p>
      </div>
    </div>
    <div class="row">
      <div class="col-md-6">
        <label>Format: <span class="muted-text">(manual alternate <em>{{altInputFormats
[0]}}</em>)</span></label> <select class="form-control" ng-model="format" ng-options="f
 for f in formats"><option></option></select>
```

```
      </div>
    </div>

    <hr />
    <button type="button" class="btn btn-sm btn-info" ng-click="today()">Today</button>
    <button type="button" class="btn btn-sm btn-default" ng-click="setDate(2009, 7, 24)
">2009-08-24</button>
    <button type="button" class="btn btn-sm btn-danger" ng-click="clear()">Clear</butto
n>
    <button type="button" class="btn btn-sm btn-default" ng-click="toggleMin()" uib-too
ltip="After today restriction">Min date</button>
</div>
```

Datepicker Popup (ui.bootstrap.datepickerPopup (https://github.com/angular-ui/bootstrap/tree/master/src/datepickerPopup))

```
Selected date is: Friday, May 6, 2016
```

Popup

06-May-2016

2016-05-06

Format: (manual alternate *M!/d!/yyyy*)

dd-MMMM-yyyy

Today | 2009-08-24 | Clear | Min date

The datepicker popup is meant to be used with an input element. To understand usage of the datepicker, please refer to its documentation here (https://angular-ui.github.io/bootstrap/#/datepicker).

uib-datepicker-popup settings

The popup is a wrapper that you can use in an input to toggle a datepicker. To configure the datepicker, use `datepicker-options` as documented in the inline datepicker (https://angular-ui.github.io/bootstrap/#/datepicker).

- `alt-input-formats` **$** **C** *(Default:* `[]` *)* - A list of alternate formats acceptable for manual entry.
- `clear-text` **C** *(Default:* `Clear` *)* - The text to display for the clear button.

- `close-on-date-selection` **$** **C** *(Default:* `true` *)* - Whether to close calendar when a date is chosen.
- `close-text` **C** *(Default:* `Done` *)* - The text to display for the close button.
- `current-text` **C** *(Default:* `Today` *)* - The text to display for the current day button.
- `datepicker-append-to-body` **$** **C** *(Default:* `false` *, Config:* `appendToBody` *)* - Append the datepicker popup element to `body`, rather than inserting after `datepicker-popup`.
- `datepicker-options` **$** - An object with any combination of the datepicker settings (in camelCase) used to configure the wrapped datepicker.
- `datepicker-popup-template-url` **C** *(Default:* `uib/template/datepickerPopup/popup.html` *)* - Add the ability to override the template used on the component.
- `datepicker-template-url` **C** *(Default:* `uib/template/datepicker/datepicker.html` *)* - Add the ability to override the template used on the component (inner uib-datepicker).
- `is-open` **$** 👁 *(Default:* `false` *)* - Whether or not to show the datepicker.
- `on-open-focus` **$** **C** *(Default:* `true` *)* - Whether or not to focus the datepicker popup upon opening.
- `show-button-bar` **$** **C** *(Default:* `true` *)* - Whether or not to display a button bar underneath the uib-datepicker.
- `type` **C** *(Default:* `text` *, Config:* `html5Types` *)* - You can override the input type to be *(date|datetime-local|month)*. That will change the date format of the popup.
- `popup-placement` **C** *(Default:* `auto bottom-left` *, Config: 'placement')* - Passing in 'auto' separated by a space before the placement will enable auto positioning, e.g: "auto bottom-left". The popup will attempt to position where it fits in the closest scrollable ancestor. Accepts:
  - `top` - popup on top, horizontally centered on input element.
  - `top-left` - popup on top, left edge aligned with input element left edge.
  - `top-right` - popup on top, right edge aligned with input element right edge.
  - `bottom` - popup on bottom, horizontally centered on input element.
  - `bottom-left` - popup on bottom, left edge aligned with input element left edge.
  - `bottom-right` - popup on bottom, right edge aligned with input element right edge.
  - `left` - popup on left, vertically centered on input element.
  - `left-top` - popup on left, top edge aligned with input element top edge.
  - `left-bottom` - popup on left, bottom edge aligned with input element bottom edge.
  - `right` - popup on right, vertically centered on input element.
  - `right-top` - popup on right, top edge aligned with input element top edge.
  - `right-bottom` - popup on right, bottom edge aligned with input element bottom edge.
- `uib-datepicker-popup` **C** *(Default:* `yyyy-MM-dd` *, Config:* `datepickerConfig` *)* - The format for displayed dates. This string can take string literals by surrounding the value with single quotes, i.e. `yyyy-MM-dd h 'o\'clock'`.

Notes

If using this directive on input type date, a native browser datepicker could also appear.

Markup () JavaScript () Edit in plunker

```
<style>
  .full button span {
    background-color: limegreen;
    border-radius: 32px;
    color: black;
  }
  .partially button span {
    background-color: orange;
    border-radius: 32px;
    color: black;
  }
</style>
<div ng-controller="DatepickerPopupDemoCtrl">
    <pre>Selected date is: <em>{{dt | date:'fullDate' }}</em></pre>

    <h4>Popup</h4>
    <div class="row">
      <div class="col-md-6">
        <p class="input-group">
          <input type="text" class="form-control" uib-datepicker-popup="{{format}}" ng-
model="dt" is-open="popup1.opened" datepicker-options="dateOptions" ng-required="true"
close-text="Close" alt-input-formats="altInputFormats" />
          <span class="input-group-btn">
            <button type="button" class="btn btn-default" ng-click="open1()"><i class="
glyphicon glyphicon-calendar"></i></button>
          </span>
        </p>
      </div>

      <div class="col-md-6">
        <p class="input-group">
          <input type="text" class="form-control" uib-datepicker-popup ng-model="dt" is
-open="popup2.opened" datepicker-options="dateOptions" ng-required="true" close-text="C
lose" />
          <span class="input-group-btn">
            <button type="button" class="btn btn-default" ng-click="open2()"><i class="
glyphicon glyphicon-calendar"></i></button>
          </span>
        </p>
      </div>
    </div>
    <div class="row">
      <div class="col-md-6">
        <label>Format: <span class="muted-text">(manual alternate <em>{{altInputFormats
[0]}}</em>)</span></label> <select class="form-control" ng-model="format" ng-options="f
 for f in formats"><option></option></select>
      </div>
    </div>

    <hr />
    <button type="button" class="btn btn-sm btn-info" ng-click="today()">Today</button>
    <button type="button" class="btn btn-sm btn-default" ng-click="setDate(2009, 7, 24)
```

```
">2009-08-24</button>
    <button type="button" class="btn btn-sm btn-danger" ng-click="clear()">Clear</butto
n>
    <button type="button" class="btn btn-sm btn-default" ng-click="toggleMin()" uib-too
ltip="After today restriction">Min date</button>
</div>
```

Dropdown (ui.bootstrap.dropdown (https://github.com/angular-ui/bootstrap/tree/master/src/dropdown))

Click me for a dropdown, yo! () Button dropdown ▾ Action ▾ Dropdown on Body ▾
Dropdown using template ▾

Toggle button dropdown Enable/Disable

Dropdown with keyboard navigation ▾

append-to vs. append-to-body vs. inline example

Dropdown in Container ▾ Dropdown on Body ▾ Inline Dropdown ▾

Dropdown is a simple directive which will toggle a dropdown menu on click or programmatically.

This directive is composed by three parts:

- `uib-dropdown` which transforms a node into a dropdown.
- `uib-dropdown-toggle` which allows the dropdown to be toggled via click. This directive is optional.
- `uib-dropdown-menu` which transforms a node into the popup menu.

Each of these parts need to be used as attribute directives.

uib-dropdown settings

- `auto-close` *(Default: `always`)* - Controls the behavior of the menu when clicked.
  - `always` - Automatically closes the dropdown when any of its elements is clicked.
  - `disabled` - Disables the auto close. You can control it manually with `is-open`. It still gets closed if the toggle is clicked, `esc` is pressed or another dropdown is open.
  - `outsideClick` - Closes the dropdown automatically only when the user clicks any element outside the dropdown.
- `dropdown-append-to` **$** *(Default: `null`)* - Appends the inner dropdown-menu to an arbitrary DOM element.
- `dropdown-append-to-body` **B** *(Default: `false`)* - Appends the inner dropdown-menu to the body element.
- `is-open` **$** 👁 *(Default: `false`)* - Defines whether or not the dropdown-menu is open. The `uib-dropdown-toggle` will toggle this attribute on click.
- `keyboard-nav`: **B** *(Default: `false`)* - Enables navigation of dropdown list elements with the arrow keys.
- `on-toggle(open)` **$** - An optional expression called when the dropdown menu is opened or closed.

uib-dropdown-menu settings

- `template-url` *(Default: `none`)* - You may specify a template for the dropdown menu. Check the demos for an example.

Additional settings `uibDropdownConfig`

- `appendToOpenClass` *(Default: `uib-dropdown-open`)* - Class to apply when the dropdown is open and appended to a different DOM element.
- `openClass` *(Default: `open`)* - Class to apply when the dropdown is open.

Known issues

For usage with ngTouch, it is recommended to use the programmatic `is-open` trigger with ng-click - this is due to ngTouch decorating ng-click to prevent propagation of the event.

Markup () JavaScript () Edit in plunker

```
<div ng-controller="DropdownCtrl">
    <!-- Simple dropdown -->
    <span uib-dropdown on-toggle="toggled(open)">
      <a href id="simple-dropdown" uib-dropdown-toggle>
        Click me for a dropdown, yo!
      </a>
      <ul class="dropdown-menu" uib-dropdown-menu aria-labelledby="simple-dropdown">
        <li ng-repeat="choice in items">
          <a href>{{choice}}</a>
        </li>
      </ul>
    </span>

    <!-- Single button -->
    <div class="btn-group" uib-dropdown is-open="status.isopen">
      <button id="single-button" type="button" class="btn btn-primary" uib-dropdown-tog
gle ng-disabled="disabled">
        Button dropdown <span class="caret"></span>
      </button>
      <ul class="dropdown-menu" uib-dropdown-menu role="menu" aria-labelledby="single-b
utton">
        <li role="menuitem"><a href="#">Action</a></li>
        <li role="menuitem"><a href="#">Another action</a></li>
        <li role="menuitem"><a href="#">Something else here</a></li>
        <li class="divider"></li>
        <li role="menuitem"><a href="#">Separated link</a></li>
      </ul>
    </div>

    <!-- Split button -->
    <div class="btn-group" uib-dropdown>
      <button id="split-button" type="button" class="btn btn-danger">Action</button>
      <button type="button" class="btn btn-danger" uib-dropdown-toggle>
        <span class="caret"></span>
        <span class="sr-only">Split button!</span>
      </button>
      <ul class="dropdown-menu" uib-dropdown-menu role="menu" aria-labelledby="split-bu
tton">
        <li role="menuitem"><a href="#">Action</a></li>
        <li role="menuitem"><a href="#">Another action</a></li>
        <li role="menuitem"><a href="#">Something else here</a></li>
        <li class="divider"></li>
        <li role="menuitem"><a href="#">Separated link</a></li>
      </ul>
    </div>

    <!-- Single button using append-to-body -->
    <div class="btn-group" uib-dropdown dropdown-append-to-body>
      <button id="btn-append-to-body" type="button" class="btn btn-primary" uib-dropdow
n-toggle>
        Dropdown on Body <span class="caret"></span>
      </button>
      <ul class="dropdown-menu" uib-dropdown-menu role="menu" aria-labelledby="btn-appe
nd-to-body">
        <li role="menuitem"><a href="#">Action</a></li>
        <li role="menuitem"><a href="#">Another action</a></li>
        <li role="menuitem"><a href="#">Something else here</a></li>
        <li class="divider"></li>
        <li role="menuitem"><a href="#">Separated link</a></li>
      </ul>
    </div>

    <!-- Single button using template-url -->
    <div class="btn-group" uib-dropdown>
      <button id="button-template-url" type="button" class="btn btn-primary" uib-dropdo
wn-toggle ng-disabled="disabled">
        Dropdown using template <span class="caret"></span>
      </button>
      <ul class="dropdown-menu" uib-dropdown-menu template-url="dropdown.html" aria-lab
elledby="button-template-url">
      </ul>
    </div>

    <hr />
    <p>
        <button type="button" class="btn btn-default btn-sm" ng-click="toggleDropdown($
event)">Toggle button dropdown</button>
        <button type="button" class="btn btn-warning btn-sm" ng-click="disabled = !disa
bled">Enable/Disable</button>
    </p>

    <hr>
    <!-- Single button with keyboard nav -->
    <div class="btn-group" uib-dropdown keyboard-nav>
        <button id="simple-btn-keyboard-nav" type="button" class="btn btn-primary" uib-
dropdown-toggle>
            Dropdown with keyboard navigation <span class="caret"></span>
        </button>
        <ul class="dropdown-menu" uib-dropdown-menu role="menu" aria-labelledby="simple
-btn-keyboard-nav">
            <li role="menuitem"><a href="#">Action</a></li>
            <li role="menuitem"><a href="#">Another action</a></li>
            <li role="menuitem"><a href="#">Something else here</a></li>
            <li class="divider"></li>
            <li role="menuitem"><a href="#">Separated link</a></li>
        </ul>
    </div>

    <hr>
    <!-- AppendTo use case -->
    <h4>append-to vs. append-to-body vs. inline example</h4>
    <div id="dropdown-scrollable-container" style="height: 15em; overflow: auto;">
      <div id="dropdown-long-content">
```

```
        <div id="dropdown-hidden-container">
          <div class="btn-group" uib-dropdown keyboard-nav dropdown-append-to="appendTo
El">
            <button id="btn-append-to" type="button" class="btn btn-primary" uib-dropdo
wn-toggle>
              Dropdown in Container <span class="caret"></span>
            </button>
            <ul class="dropdown-menu" uib-dropdown-menu role="menu" aria-labelledby="bt
n-append-to">
              <li role="menuitem"><a href="#">Action</a></li>
              <li role="menuitem"><a href="#">Another action</a></li>
              <li role="menuitem"><a href="#">Something else here</a></li>
              <li class="divider"></li>
              <li role="menuitem"><a href="#">Separated link</a></li>
            </ul>
          </div>
          <div class="btn-group" uib-dropdown dropdown-append-to-body>
            <button id="btn-append-to-to-body" type="button" class="btn btn-primary" ui
b-dropdown-toggle>
              Dropdown on Body <span class="caret"></span>
            </button>
            <ul class="dropdown-menu" uib-dropdown-menu role="menu" aria-labelledby="bt
n-append-to-to-body">
              <li role="menuitem"><a href="#">Action</a></li>
              <li role="menuitem"><a href="#">Another action</a></li>
              <li role="menuitem"><a href="#">Something else here</a></li>
              <li class="divider"></li>
              <li role="menuitem"><a href="#">Separated link</a></li>
            </ul>
          </div>
          <div class="btn-group" uib-dropdown>
            <button id="btn-append-to-single-button" type="button" class="btn btn-prima
ry" uib-dropdown-toggle>
              Inline Dropdown <span class="caret"></span>
            </button>
            <ul class="dropdown-menu" uib-dropdown-menu role="menu" aria-labelledby="bt
n-append-to-single-button">
              <li role="menuitem"><a href="#">Action</a></li>
              <li role="menuitem"><a href="#">Another action</a></li>
              <li role="menuitem"><a href="#">Something else here</a></li>
              <li class="divider"></li>
              <li role="menuitem"><a href="#">Separated link</a></li>
            </ul>
          </div>
        </div>
      </div>
    </div>

    <script type="text/ng-template" id="dropdown.html">
        <ul class="dropdown-menu" uib-dropdown-menu role="menu" aria-labelledby="button
-template-url">
          <li role="menuitem"><a href="#">Action in Template</a></li>
          <li role="menuitem"><a href="#">Another action in Template</a></li>
          <li role="menuitem"><a href="#">Something else here</a></li>
          <li class="divider"></li>
          <li role="menuitem"><a href="#">Separated link in Template</a></li>
        </ul>
    </script>
</div>
```

Modal (ui.bootstrap.modal (https://github.com/angular-ui/bootstrap/tree/master/src/modal))

Open me! Large modal Small modal Toggle Animation (true)

`$uibModal` is a service to create modal windows. Creating modals is straightforward: create a template, a controller and reference them when using `$uibModal`.

The `$uibModal` service has only one method: `open(options)`.

$uibModal's open function

options parameter

- `animation` *(Type: `boolean`, Default: `true`)* - Set to false to disable animations on new modal/backdrop. Does not toggle animations for modals/backdrops that are already displayed.
- `appendTo` *(Type: `angular.element`, Default: `body`: Example: `$document.find('aside').eq(0)`)* - Appends the modal to a specific element.
- `backdrop` *(Type: `boolean|string`, Default: `true`)* - Controls presence of a backdrop. Allowed values: `true` (default), `false` (no backdrop), `'static'` (disables modal closing by click on the backdrop).
- `backdropClass` *(Type: `string`)* - Additional CSS class(es) to be added to a modal backdrop template.
- `bindToController` *(Type: `boolean`, Default: `false`)* - When used with `controllerAs` & set to `true`, it will bind the $scope properties onto the controller.
- `controller` *(Type: `function|string|array`, Example: `MyModalController`)* - A controller for the modal instance, either a controller name as a string, or an inline controller function, optionally wrapped in array notation for dependency injection. Allows the controller-as syntax. Has a special `$uibModalInstance` injectable to access the modal instance.
- `controllerAs` *(Type: `string`, Example: `ctrl`)* - An alternative to the controller-as syntax. Requires the `controller` option to be provided as well.
- `keyboard` - *(Type: `boolean`, Default: `true`)* - Indicates whether the dialog should be closable by hitting the ESC key.
- `openedClass` *(Type: `string`, Default: `modal-open`)* - Class added to the `body` element when the

modal is opened.

- `resolve` *(Type:* `Object` *)* - Members that will be resolved and passed to the controller as locals; it is equivalent of the `resolve` property in the router.
- `scope` *(Type:* `$scope` *)* - The parent scope instance to be used for the modal's content. Defaults to `$rootScope`.
- `size` *(Type:* `string` *, Example:* `lg` *)* - Optional suffix of modal window class. The value used is appended to the `modal-` class, i.e. a value of `sm` gives `modal-sm`.
- `template` *(Type:* `string` *)* - Inline template representing the modal's content.
- `templateUrl` *(Type:* `string` *)* - A path to a template representing modal's content. You need either a `template` or `templateUrl`.
- `windowClass` *(Type:* `string` *)* - Additional CSS class(es) to be added to a modal window template.
- `windowTemplateUrl` *(Type:* `string` *, Default:* `uib/template/modal/window.html` *)* - A path to a template overriding modal's window template.
- `windowTopClass` *(Type:* `string` *)* - CSS class(es) to be added to the top modal window.

Global defaults may be set for `$uibModal` via `$uibModalProvider.options`.

return

The `open` method returns a modal instance, an object with the following properties:

- `close(result)` *(Type:* `function` *)* - Can be used to close a modal, passing a result.
- `dismiss(reason)` *(Type:* `function` *)* - Can be used to dismiss a modal, passing a reason.
- `result` *(Type:* `promise` *)* - Is resolved when a modal is closed and rejected when a modal is dismissed.
- `opened` *(Type:* `promise` *)* - Is resolved when a modal gets opened after downloading content's template and resolving all variables.
- `closed` *(Type:* `promise` *)* - Is resolved when a modal is closed and the animation completes.
- `rendered` *(Type:* `promise` *)* - Is resolved when a modal is rendered.

The scope associated with modal's content is augmented with:

- `$close(result)` *(Type:* `function` *)* - A method that can be used to close a modal, passing a result.
- `$dismiss(reason)` *(Type:* `function` *)* - A method that can be used to dismiss a modal, passing a reason.

Those methods make it easy to close a modal window without a need to create a dedicated controller.

Also, when using `bindToController`, you can define an `$onInit` method in the controller that will fire upon initialization.

Events fired:

- `$uibUnscheduledDestruction` - This event is fired if the $scope is destroyed via unexpected mechanism, such as it being passed in the modal options and a $route/$state transition occurs. The modal will also be dismissed.

- `modal.closing` - This event is broadcast to the modal scope before the modal closes. If the listener calls preventDefault() on the event, then the modal will remain open. Also, the `$close` and `$dismiss` methods returns true if the event was executed. This event also includes a parameter for the result/reason and a boolean that indicates whether the modal is being closed (true) or dismissed.

UI Router resolves

If one wants to have the modal resolve using UI Router's (https://github.com/angular-ui/ui-router) pre-1.0 resolve mechanism, one can call `$uibResolve.setResolver('$resolve')` in the configuration phase of the application. One can also provide a custom resolver as well, as long as the signature conforms to UI Router's $resolve (http://angular-ui.github.io/ui-router/site/#/api/ui.router.util.$resolve).

Markup () JavaScript () Edit in plunker

```
<div ng-controller="ModalDemoCtrl">
    <script type="text/ng-template" id="myModalContent.html">
        <div class="modal-header">
            <h3 class="modal-title">I'm a modal!</h3>
        </div>
        <div class="modal-body">
            <ul>
                <li ng-repeat="item in items">
                    <a href="#" ng-click="$event.preventDefault(); selected.item = item
">{{ item }}</a>
                </li>
            </ul>
            Selected: <b>{{ selected.item }}</b>
        </div>
        <div class="modal-footer">
            <button class="btn btn-primary" type="button" ng-click="ok()">OK</button>
            <button class="btn btn-warning" type="button" ng-click="cancel()">Cancel</b
utton>
        </div>
    </script>

    <button type="button" class="btn btn-default" ng-click="open()">Open me!</button>
    <button type="button" class="btn btn-default" ng-click="open('lg')">Large modal</bu
tton>
    <button type="button" class="btn btn-default" ng-click="open('sm')">Small modal</bu
tton>
    <button type="button" class="btn btn-default" ng-click="toggleAnimation()">Toggle A
nimation ({{ animationsEnabled }})</button>
    <div ng-show="selected">Selection from a modal: {{ selected }}</div>
</div>
```

Pager (ui.bootstrap.pager (https://github.com/angular-ui/bootstrap

/tree/master/src/pager))

Pager

```
You are currently on page 4
```

« Previous () Next » ()

A lightweight pager directive that is focused on providing previous/next paging functionality

uib-pager settings

- `align` **C** *(Default:* `true` *)* - Whether to align each link to the sides.
- `items-per-page` **$** **C** 👁 *(Default:* `10` *)* - Maximum number of items per page. A value less than one indicates all items on one page.
- `next-text` **C** *(Default:* `Next »` *)* - Text for Next button.
- `ng-disabled` **$** 👁 *(Default:* `false` *)* - Used to disable the pager component.
- `ng-model` **$** 👁 - Current page number. First page is 1.
- `num-pages` **$** **readonly** *(Default:* `angular.noop` *)* - An optional expression assigned the total number of pages to display.
- `previous-text` **C** *(Default:* `« Previous` *)* - Text for Previous button.
- `template-url` *(Default:* `uib/template/pager/pager.html` *)* - Override the template for the component with a custom provided template.
- `total-items` **$** 👁 - Total number of items in all pages.

Markup () JavaScript () Edit in plunker

```
<div ng-controller="PagerDemoCtrl">
  <h4>Pager</h4>
  <pre>You are currently on page {{currentPage}}</pre>
  <uib-pager total-items="totalItems" ng-model="currentPage"></uib-pager>
</div>
```

Pagination (ui.bootstrap.pagination (https://github.com/angular-ui/bootstrap/tree/master/src/pagination))

Default

Previous () | 1 () | 2 () | 3 () | 4 () | 5 () | 6 () | 7 () | Next ()

« () | ‹ () | 1 () | 2 () | 3 () | 4 () | 5 () | 6 () | 7 () | › () | » ()

First () | 1 () | 2 () | 3 () | 4 () | 5 () | 6 () | 7 () | Last ()

1 () | 2 () | 3 () | 4 () | 5 () | 6 () | 7 ()

```
The selected page no: 4
```

Set current page to: 3

Limit the maximum visible buttons

`rotate` defaulted to `true` :

First () | Previous () | 1 () | 2 () | 3 () | 4 () | 5 () | Next () | Last ()

`rotate` defaulted to `true` and `force-ellipses` set to `true` :

First () | Previous () | 1 () | 2 () | 3 () | 4 () | 5 () | ... () | Next () | Last ()

`rotate` set to `false` :

First () | Previous () | 1 () | 2 () | 3 () | 4 () | 5 () | ... () | Next () | Last ()

`boundary-link-numbers` set to `true` and `rotate` defaulted to `true` :

Previous () | 1 () | 2 () | 3 () | 4 () | 5 () | ... () | 18 () | Next ()

`boundary-link-numbers` set to `true` and `rotate` set to `false` :

Previous () | 1 () | 2 () | 3 () | 4 () | 5 () | ... () | 18 () | Next ()

```
Page: 1 / 18
```

A lightweight pagination directive that is focused on ... providing pagination & will take care of visualising a pagination bar and enable / disable buttons correctly!

uib-pagination settings

- `boundary-links` **C** *(Default: `false`)* - Whether to display First / Last buttons.
- `boundary-link-numbers` **$** **C** *(Default: `false`)* - Whether to always display the first and last page numbers. If `max-size` is smaller than the number of pages, then the first and last page numbers are still shown with ellipses in-between as necessary. NOTE: `max-size` refers to the center of the range. This option may add up to 2 more numbers on each side of the displayed range for the end value and what would be an ellipsis but is replaced by a number because it is sequential.
- `direction-links` **$** **C** *(Default: `true`)* - Whether to display Previous / Next buttons.
- `first-text` **C** *(Default: `First`)* - Text for First button.
- `force-ellipses` **$** **C** *(Default: `false`)* - Also displays ellipses when `rotate` is true and `max-size` is smaller than the number of pages.
- `items-per-page` **$** **C** 👁 *(Default: `10`)* - Maximum number of items per page. A value less than one indicates all items on one page.
- `last-text` **C** *(Default: `Last`)* - Text for Last button.
- `max-size` **$** 👁 *(Default: `null`)* - Limit number for pagination size.
- `next-text` **C** *(Default: `Next`)* - Text for Next button.
- `ng-change` **$** - This can be used to call a function whenever the page changes.
- `ng-disabled` **$** 👁 *(Default: `false`)* - Used to disable the pagination component.
- `ng-model` **$** 👁 - Current page number. First page is 1.
- `num-pages` **$** **readonly** *(Default: `angular.noop`)* - An optional expression assigned the total number of pages to display.
- `page-label` *(Default: `angular.identity`)* - An optional expression to override the page label based on

passing the current page indexes. Supports page number with `$page` in the template.

- `previous-text` **C** *(Default: `Previous`)* - Text for Previous button.
- `rotate` **$** **C** *(Default: `true`)* - Whether to keep current page in the middle of the visible ones.
- `template-url` *(Default: `uib/template/pagination/pagination.html`)* - Override the template for the component with a custom provided template
- `total-items` **$** 👁 - Total number of items in all pages.

Markup () JavaScript () Edit in plunker

```
<div ng-controller="PaginationDemoCtrl">
    <h4>Default</h4>
    <uib-pagination total-items="totalItems" ng-model="currentPage" ng-change="pageChan
ged()"></uib-pagination>
    <uib-pagination boundary-links="true" total-items="totalItems" ng-model="currentPag
e" class="pagination-sm" previous-text="&lsaquo;" next-text="&rsaquo;" first-text="&laq
uo;" last-text="&raquo;"></uib-pagination>
    <uib-pagination direction-links="false" boundary-links="true" total-items="totalIte
ms" ng-model="currentPage"></uib-pagination>
    <uib-pagination direction-links="false" total-items="totalItems" ng-model="currentP
age" num-pages="smallnumPages"></uib-pagination>
    <pre>The selected page no: {{currentPage}}</pre>
    <button type="button" class="btn btn-info" ng-click="setPage(3)">Set current page t
o: 3</button>

    <hr />
    <h4>Limit the maximum visible buttons</h4>
    <h6><code>rotate</code> defaulted to <code>true</code>:</h6>
    <uib-pagination total-items="bigTotalItems" ng-model="bigCurrentPage" max-size="max
Size" class="pagination-sm" boundary-links="true" num-pages="numPages"></uib-pagination
>
    <h6><code>rotate</code> defaulted to <code>true</code> and <code>force-ellipses</co
de> set to <code>true</code>:</h6>
    <uib-pagination total-items="bigTotalItems" ng-model="bigCurrentPage" max-size="max
Size" class="pagination-sm" boundary-links="true" force-ellipses="true"></uib-paginatio
n>
    <h6><code>rotate</code> set to <code>false</code>:</h6>
    <uib-pagination total-items="bigTotalItems" ng-model="bigCurrentPage" max-size="max
Size" class="pagination-sm" boundary-links="true" rotate="false"></uib-pagination>
    <h6><code>boundary-link-numbers</code> set to <code>true</code> and <code>rotate</c
ode> defaulted to <code>true</code>:</h6>
    <uib-pagination total-items="bigTotalItems" ng-model="bigCurrentPage" max-size="max
Size" class="pagination-sm" boundary-link-numbers="true"></uib-pagination>
    <h6><code>boundary-link-numbers</code> set to <code>true</code> and <code>rotate</c
ode> set to <code>false</code>:</h6>
    <uib-pagination total-items="bigTotalItems" ng-model="bigCurrentPage" max-size="max
Size" class="pagination-sm" boundary-link-numbers="true" rotate="false"></uib-paginatio
n>
    <pre>Page: {{bigCurrentPage}} / {{numPages}}</pre>

</div>
```

Popover (ui.bootstrap.popover (https://github.com/angular-ui/bootstrap/tree/master/src/popover))

Dynamic

Popup Text:

Hello, World!

Popup Title:

Title

Popup Template:

myPopoverTemplate.html

Dynamic Popover Popover With Template

Positional

Popover placement

top

Popover top

Triggers

Mouseenter

Click me!

Other

fading title Toggle popover HTML Popover

A lightweight, extensible directive for fancy popover creation. The popover directive supports multiple placements, optional transition animation, and more.

Like the Bootstrap jQuery plugin, the popover **requires** the tooltip module.

Note to mobile developers: Please note that while popovers may work correctly on mobile devices (including tablets), we have made the decision to not officially support such a use-case because it does not make sense from a UX perspective.

There are three versions of the popover: `uib-popover` and `uib-popover-template`, and `uib-popover-html`:

- `uib-popover` - Takes text only and will escape any HTML provided for the popover body.
- `uib-popover-html` **$** - Takes an expression that evaluates to an HTML string. Note that this HTML is not compiled. If compilation is required, please use the `uib-popover-template` attribute option instead. *The user is responsible for ensuring the content is safe to put into the DOM!*

- `uib-popover-template` **$** - A URL representing the location of a template to use for the popover body. Note that the contents of this template need to be wrapped in a tag, e.g., `<div></div>`.

uib-popover-* settings

All these settings are available for the three types of popovers.

- `popover-animation` **$** **C** *(Default: `true`, Config: `animation`)* - Should it fade in and out?
- `popover-append-to-body` **$** **C** *(Default: `false`, Config: `appendToBody`)* - Should the popover be appended to '$body' instead of the parent element?
- `popover-enable` **$** *(Default: `true`)* - Is it enabled? It will enable or disable the configured popover-trigger.
- `popover-is-open` 👁 *(Default: `false`)* - Whether to show the popover.
- `popover-placement` **C** *(Default: `top`, Config: `placement`)* - Passing in 'auto' separated by a space before the placement will enable auto positioning, e.g: "auto bottom-left". The popover will attempt to position where it fits in the closest scrollable ancestor. Accepts:
  - `top` - popover on top, horizontally centered on host element.
  - `top-left` - popover on top, left edge aligned with host element left edge.
  - `top-right` - popover on top, right edge aligned with host element right edge.
  - `bottom` - popover on bottom, horizontally centered on host element.
  - `bottom-left` - popover on bottom, left edge aligned with host element left edge.
  - `bottom-right` - popover on bottom, right edge aligned with host element right edge.
  - `left` - popover on left, vertically centered on host element.
  - `left-top` - popover on left, top edge aligned with host element top edge.
  - `left-bottom` - popover on left, bottom edge aligned with host element bottom edge.
  - `right` - popover on right, vertically centered on host element.
  - `right-top` - popover on right, top edge aligned with host element top edge.
  - `right-bottom` - popover on right, bottom edge aligned with host element bottom edge.
- `popover-popup-close-delay` **C** *(Default: `0`, Config: `popupCloseDelay`)* - For how long should the popover remain open after the close trigger event?
- `popover-popup-delay` **C** *(Default: `0`, Config: `popupDelay`)* - Popup delay in milliseconds until it opens.
- `popover-title` - A string to display as a fancy title.
- `popover-trigger` *(Default: `click`)* - What should trigger a show of the popover? Supports a space separated list of event names (see below).

Note: To configure the tooltips, you need to do it on `$uibTooltipProvider` (also see below).

Triggers

The following show triggers are supported out of the box, along with their provided hide triggers:

- `mouseenter` : `mouseleave`
- `click` : `click`
- `outsideClick` : `outsideClick`
- `focus` : `blur`
- `none`

The `outsideClick` trigger will cause the popover to toggle on click, and hide when anything else is clicked.

For any non-supported value, the trigger will be used to both show and hide the popover. Using the 'none' trigger will disable the internal trigger(s), one can then use the `popover-is-open` attribute exclusively to show and hide the popover.

$uibTooltipProvider

Through the `$uibTooltipProvider`, you can change the way tooltips and popovers behave by default; the attributes above always take precedence. The following methods are available:

- `setTriggers(obj)` *(Example:* `{ 'openTrigger': 'closeTrigger' }` *)* - Extends the default trigger mappings mentioned above with mappings of your own.
- `options(obj)` - Provide a set of defaults for certain tooltip and popover attributes. Currently supports the ones with the **C** badge.

Known issues

For Safari 7+ support, if you want to use **focus** `popover-trigger`, you need to use an anchor tag with a tab index. For example:

```
<a tabindex="0" uib-popover="Test" popover-trigger="focus" class="btn btn-default">
  Click Me
</a>
```

Markup () JavaScript () Edit in plunker

```
<div ng-controller="PopoverDemoCtrl">
    <h4>Dynamic</h4>
    <div class="form-group">
      <label>Popup Text:</label>
      <input type="text" ng-model="dynamicPopover.content" class="form-control">
    </div>
    <div class="form-group">
      <label>Popup Title:</label>
      <input type="text" ng-model="dynamicPopover.title" class="form-control">
    </div>
    <div class="form-group">
      <label>Popup Template:</label>
      <input type="text" ng-model="dynamicPopover.templateUrl" class="form-control">
    </div>
    <button uib-popover="{{dynamicPopover.content}}" popover-title="{{dynamicPopover.ti
tle}}" type="button" class="btn btn-default">Dynamic Popover</button>

    <button uib-popover-template="dynamicPopover.templateUrl" popover-title="{{dynamicP
opover.title}}" type="button" class="btn btn-default">Popover With Template</button>

    <script type="text/ng-template" id="myPopoverTemplate.html">
        <div>{{dynamicPopover.content}}</div>
        <div class="form-group">
          <label>Popup Title:</label>
          <input type="text" ng-model="dynamicPopover.title" class="form-control">
        </div>
    </script>
    <hr />
    <h4>Positional</h4>
    <div class="form-group">
      <label>Popover placement</label>
      <select class="form-control" ng-model="placement.selected" ng-options="o as o for
 o in placement.options"></select>
    </div>
    <button popover-placement="{{placement.selected}}" uib-popover="On the {{placement.
selected}}" type="button" class="btn btn-default">Popover {{placement.selected}}</butto
n>

    <hr />
    <h4>Triggers</h4>
    <p>
      <button uib-popover="I appeared on mouse enter!" popover-trigger="mouseenter" typ
e="button" class="btn btn-default">Mouseenter</button>
    </p>
    <input type="text" value="Click me!" uib-popover="I appeared on focus! Click away a
nd I'll vanish..."  popover-trigger="focus" class="form-control">

    <hr />
    <h4>Other</h4>
    <button popover-animation="true" uib-popover="I fade in and out!" type="button" cla
ss="btn btn-default">fading</button>
    <button uib-popover="I have a title!" popover-title="The title." type="button" clas
```

```
s="btn btn-default">title</button>
    <button uib-popover="I am activated manually" popover-is-open="popoverIsOpen" ng-cl
ick="popoverIsOpen = !popoverIsOpen" type="button" class="btn btn-default">Toggle popov
er</button>
    <button uib-popover-html="htmlPopover" class="btn btn-default">HTML Popover</button
>
</div>
```

Position (ui.bootstrap.position (https://github.com/angular-ui/bootstrap/tree/master/src/position))

$uibPosition service

✓ Parent scrollable
✓ Parent relative

Get values

Demo element

offsetParent:
scrollParent:
scrollbarWidth:
position:
offset:
viewportOffset:
positionElements:

The `$uibPosition` service provides a set of DOM utilities used internally to absolute-position an element in relation to another element (tooltips, popovers, typeaheads etc...).

getRawNode(element)

Takes a jQuery/jqLite element and converts it to a raw DOM element.

parameters

- `element` *(Type:* `object` *)* - The element to convert.

returns

- *(Type:* `element` *)* - A raw DOM element.

parseStyle(element)

Parses a numeric style value to a number. Strips units and will return 0 for invalid (NaN) numbers.

parameters

- `value` *(Type:* `string` *)* - The style value to parse.

returns

- *(Type:* `number` *)* - The numeric value of the style property.

offsetParent(element)

Gets the closest positioned ancestor.

parameters

- `element` *(Type:* `element` *)* - The element to get the offset parent for.

returns

- *(Type:* `element` *)* - The closest positioned ancestor.

scrollbarWidth(isBody)

Calculates the browser scrollbar width and caches the result for future calls. Concept from the TWBS measureScrollbar() function in modal.js (https://github.com/twbs/bootstrap/blob/master/js/modal.js).

parameters

- `isBody` *(Type:* `boolean` *, Default:* `false` *, optional)* - Is the requested scrollbar width for the body/html element. IE and Edge overlay the scrollbar on the body/html element and should be considered 0.

returns

- *(Type:* `number` *)* - The width of the browser scrollbar.

scrollbarPadding(element)

Calculates the padding required to replace the scrollbar on an element.

parameters

- 'element' *(Type:* `element` *)* - The element to calculate the padding on (should be a scrollable element).

returns

An object with the following properties:

- `scrollbarWidth` *(Type:* `number` *)* - The width of the scrollbar.
- `widthOverflow` *(Type:* `boolean` *)* - Whether the width is overflowing.
- `right` *(Type:* `number` *)* - The total right padding required to replace the scrollbar.
- `originalRight` *(Type:* `number` *)* - The oringal right padding on the element.
- `heightOverflow` *(Type:* `boolean` *)* - Whether the height is overflowing.
- `bottom` *(Type:* `number` *)* - The total bottom padding required to replace the scrollbar.
- `originalBottom` *(Type:* `number` *)* - The oringal bottom padding on the element.

isScrollable(element, includeHidden)

Determines if an element is scrollable.

parameters

- `element` *(Type: `element`)* - The element to check.
- `includeHidden` *(Type: `boolean` , Default: `false` , optional)* - Should scroll style of 'hidden' be considered.

returns

- *(Type: `boolean`)* - Whether the element is scrollable.

scrollParent(element, includeHidden, includeSelf)

Gets the closest scrollable ancestor. Concept from the jQueryUI scrollParent.js (https://github.com/jquery/jquery-ui/blob/master/ui/scroll-parent.js).

parameters

- `element` *(Type: `element`)* - The element to get the closest scrollable ancestor for.
- `includeHidden` *(Type: `boolean` , Default: `false` , optional)* - Should scroll style of 'hidden' be considered.
- `includeSelf` *(Type: `boolean` , Default: `false` , optional)* - Should the element passed in be included in the scrollable lookup.

returns

- *(Type: `element`)* - The closest scrollable ancestor.

position(element, includeMargins)

A read-only equivalent of jQuery's position (http://api.jquery.com/position/) function, distance to closest positioned ancestor. Does not account for margins by default like jQuery's position.

parameters

- `element` *(Type: `element`)* - The element to get the position for.
- `includeMargins` *(Type: `boolean` , Default: `false` , optional)* - Should margins be accounted for.

returns

An object with the following properties:

- `width` *(Type: `number`)* - The width of the element.
- `height` *(Type: `number`)* - The height of the element.
- `top` *(Type: `number`)* - Distance to top edge of offset parent.
- `left` *(Type: `number`)* - Distance to left edge of offset parent.

offset(element)

A read-only equivalent of jQuery's offset (http://api.jquery.com/offset/) function, distance to viewport.

parameters

- `element` *(Type: `element`)* - The element to get the offset for.

returns

An object with the following properties:

- `width` *(Type: `number`)* - The width of the element.
- `height` *(Type: `number`)* - The height of the element.
- `top` *(Type: `number`)* - Distance to top edge of the viewport.
- `left` *(Type: `number`)* - Distance to left edge of the viewport.

viewportOffset(element, useDocument, includePadding)

Gets the elements available space relative to the closest scrollable ancestor. Accounts for padding, border, and scrollbar width. Right and bottom dimensions represent the distance to the respective edge of the viewport element, not the top and left edge. If the element edge extends beyond the viewport, a negative value will be reported.

parameters

- `element` *(Type: `element`)* - The element to get the viewport offset for.
- `useDocument` *(Type: `boolean` , Default: `false` , optional)* - Should the viewport be the document element instead of the first scrollable element.
- `includePadding` *(Type: `boolean` , Default: `true` , optional)* - Should the padding on the viewport element be accounted for, default is true.

returns

An object with the following properties:

- `top` *(Type: `number`)* - Distance to top content edge of the viewport.
- `bottom` *(Type: `number`)* - Distance to bottom content edge of the viewport.
- `left` *(Type: `number`)* - Distance to left content edge of the viewport.
- `right` *(Type: `number`)* - Distance to right content edge of the viewport.

parsePlacement(placement)

Gets an array of placement values parsed from a placement string. Along with the 'auto' indicator, supported placement strings are:

- top: element on top, horizontally centered on host element.
- top-left: element on top, left edge aligned with host element left edge.
- top-right: element on top, right edge aligned with host element right edge.
- bottom: element on bottom, horizontally centered on host element.
- bottom-left: element on bottom, left edge aligned with host element left edge.
- bottom-right: element on bottom, right edge aligned with host element right edge.
- left: element on left, vertically centered on host element.
- left-top: element on left, top edge aligned with host element top edge.
- left-bottom: element on left, bottom edge aligned with host element bottom edge.
- right: element on right, vertically centered on host element.
- right-top: element on right, top edge aligned with host element top edge.
- right-bottom: element on right, bottom edge aligned with host element bottom edge.

A placement string with an 'auto' indicator is expected to be space separated from the placement, i.e: 'auto bottom-left'. If the primary and secondary placement values do not match 'top, bottom, left, right' then 'top' will be the primary placement and 'center' will be the secondary placement. If 'auto' is passed, true will be returned as the 3rd value of the array.

parameters

- `placement` *(Type: `string` , Example: `auto top-left`)* - The placement string to parse.

returns

An array with the following values:

- `[0]` *(Type: `string`)* - The primary placement.
- `[1]` *(Type: `string`)* - The secondary placement.
- `[2]` *(Type: `boolean`)* - Is auto place enabled.

positionElements(hostElement, targetElement, placement, appendToBody)

Gets gets coordinates for an element to be positioned relative to another element.

parameters

- `hostElement` *(Type: `element`)* - The element to position against.
- `targetElement` *(Type: `element`)* - The element to position.
- `placement` *(Type: `string` , Default: `top` , optional)* - The placement for the target element. See the parsePlacement() function for available options. If 'auto' placement is used, the viewportOffset() function is used to decide where the targetElement will fit.
- `appendToBody` *(Type: `boolean` , Default: `false` , optional)* - Should the coordinates be cacluated from the body element.

returns

An object with the following properties:

- `top` *(Type: `number`)* - The targetElement top value.
- `left` *(Type: `number`)* - The targetElement left value.
- `right` *(Type: `number`)* - The resolved placement with 'auto' removed.

positionArrow(element, placement)

Positions the tooltip and popover arrow elements when using placement options beyond the standard top, left, bottom, or right.

parameters

- `element` *(Type: `element`)* - The element to position the arrow element for.
- `placement` *(Type: `string`)* - The placement for the element.

Markup () JavaScript () Edit in plunker

```
<div ng-controller="PositionDemoCtrl">
  <h4>$uibPosition service</h4>
  <div id="posdemoparent" ng-style="{'overflow': (parentScrollable && 'scroll'), 'posit
ion': (parentRelative && 'relative')}" style="border: 1px solid #ccc; padding: 15px;">
    <div class="checkbox">
      <label>
        <input type="checkbox" ng-model="parentScrollable"> Parent scrollable
      </label>
    </div>
    <div class="checkbox">
      <label>
        <input type="checkbox" ng-model="parentRelative"> Parent relative
      </label>
    </div>
    <button id="posdemobtn" class="btn btn-default" ng-click="getValues()">Get values</
button>

    <div id="posdemodiv" style="width: 100px; height: 100px; margin: 15px 0; padding: 1
0px; background-color: #f8f8f8; border: 1px solid #ccc;">
      Demo element
    </div>
  </div>
  <br />
  offsetParent: {{elemVals.offsetParent}}
  <br />
  scrollParent: {{elemVals.scrollParent}}
  <br />
  scrollbarWidth: {{scrollbarWidth}}
  <br />
  position: {{elemVals.position}}
  <br />
  offset: {{elemVals.offset}}
  <br />
  viewportOffset: {{elemVals.viewportOffset}}
  <br />
  positionElements: {{elemVals.positionElements}}
</div>
```

Progressbar (ui.bootstrap.progressbar (https://github.com/angular-ui/bootstrap/tree/master/src/progressbar))

Static

22% *166 / 200*

Dynamic Randomize

56 / 200

No animation

56%

Object (changes type based on value)

warning *!!! Watch out !!!*

Stacked Randomize

23%

A progress bar directive that is focused on providing feedback on the progress of a workflow or action.

It supports multiple (stacked) `<uib-bar>` into the same `<uib-progress>` element or a single `<uib-progressbar>` element with optional `max` attribute and transition animations.

uib-progressbar settings

- `value` **$** 👁 - The current value of progress completed.
- `type` *(Default: `null`)* - Bootstrap style type. Possible values are 'success', 'info', 'warning', and, 'danger' to use Bootstrap's pre-existing styling, or any desired custom suffix.
- `max` **$** **C** 👁 *(Default: `100`)* - A number that specifies the total value of bars that is required.
- `animate` **$** **C** *(Default: `true`)* - Whether bars use transitions to achieve the width change.
- `title` *(Default: `progressbar`)* - Title to use as label (for accessibility).

uib-progress settings

- `max` **$** **C** 👁 *(Default: `100`)* - A number that specifies the total value of bars that is required.
- `animate` **$** **C** *(Default: `true`)* - Whether bars use transitions to achieve the width change.
- `title` *(Default: `progressbar`)* - Title to use as label (for accessibility).

uib-bar settings

- `value` **$** 👁 - The current value of progress completed.
- `type` *(Default: `null`)* - Bootstrap style type. Possible values are 'success', 'info', 'warning', and, 'error' to use Bootstrap's pre-existing styling, or any desired custom suffix.
- `title` *(Default: `progressbar`)* - Title to use as label (for accessibility).

Markup () JavaScript () Edit in plunker

```
<div ng-controller="ProgressDemoCtrl">

    <h3>Static</h3>
    <div class="row">
        <div class="col-sm-4"><uib-progressbar value="55"></uib-progressbar></div>
        <div class="col-sm-4"><uib-progressbar class="progress-striped" value="22" type
="warning">22%</uib-progressbar></div>
        <div class="col-sm-4"><uib-progressbar class="progress-striped active" max="200
" value="166" type="danger"><i>166 / 200</i></uib-progressbar></div>
    </div>

    <hr />
    <h3>Dynamic <button type="button" class="btn btn-sm btn-primary" ng-click="random()
">Randomize</button></h3>
    <uib-progressbar max="max" value="dynamic"><span style="color:white; white-space:no
wrap;">{{dynamic}} / {{max}}</span></uib-progressbar>

    <small><em>No animation</em></small>
    <uib-progressbar animate="false" value="dynamic" type="success"><b>{{dynamic}}%</b>
</uib-progressbar>

    <small><em>Object (changes type based on value)</em></small>
    <uib-progressbar class="progress-striped active" value="dynamic" type="{{type}}">{{
type}} <i ng-show="showWarning">!!! Watch out !!!</i></uib-progressbar>

    <hr />
    <h3>Stacked <button type="button" class="btn btn-sm btn-primary" ng-click="randomSt
acked()">Randomize</button></h3>
    <uib-progress><uib-bar ng-repeat="bar in stacked track by $index" value="bar.value"
 type="{{bar.type}}"><span ng-hide="bar.value < 5">{{bar.value}}%</span></uib-bar></uib
-progress>

</div>
```

Rating (ui.bootstrap.rating (https://github.com/angular-ui/bootstrap/tree/master/src/rating))

Default

★★★★★★★☆☆☆

```
Rate: 7 - Readonly is: false - Hovering over: none
```

Clear Toggle Readonly

Custom icons

✔ ✔ ✔ ✔ ✔ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ (***Rate:*** **5**)
✔ ★ ⊘ ☆ ⏻ (***Rate:*** **2**)

Rating directive that will take care of visualising a star rating bar.

uib-rating settings

- `max` **$** **C** *(Default:* `5` *)* - Changes the number of icons.
- `ng-model` **$** 👁 - The current rate.
- `on-hover(value)` **$** - An optional expression called when user's mouse is over a particular icon.
- `on-leave()` **$** - An optional expression called when user's mouse leaves the control altogether.
- `rating-states` **$** *(Default:* `null` *)* - An array of objects defining properties for all icons. In default template, `stateOn` & `stateOff` property is used to specify the icon's class.
- `read-only` **$** *(Default:* `false` *)* - Prevent user's interaction.
- `titles` **$** **C** *(Default: ['one', 'two', 'three', 'four', 'five']`)* - An array of strings defining titles for all icons.
- `enable-reset` **$** *(Default:* `true` *)* - Clicking the icon of the current rating will reset the rating to 0.
- `state-off` **$** **C** *(Default:* `null` *)* - A variable used in the template to specify the state for unselected icons.
- `state-on` **$** **C** *(Default:* `null` *)* - A variable used in the template to specify the state (class, src, etc) for selected icons.

Markup () JavaScript () Edit in plunker

```
<div ng-controller="RatingDemoCtrl">
    <h4>Default</h4>
    <uib-rating ng-model="rate" max="max" read-only="isReadonly" on-hover="hoveringOver
(value)" on-leave="overStar = null" titles="['one','two','three']" aria-labelledby="def
ault-rating"></uib-rating>
    <span class="label" ng-class="{'label-warning': percent<30, 'label-info': percent>=
30 && percent<70, 'label-success': percent>=70}" ng-show="overStar && !isReadonly">{{pe
rcent}}%</span>

    <pre style="margin:15px 0;">Rate: <b>{{rate}}</b> - Readonly is: <i>{{isReadonly}}<
/i> - Hovering over: <b>{{overStar || "none"}}</b></pre>

    <button type="button" class="btn btn-sm btn-danger" ng-click="rate = 0" ng-disabled
="isReadonly">Clear</button>
    <button type="button" class="btn btn-sm btn-default" ng-click="isReadonly = ! isRea
donly">Toggle Readonly</button>
    <hr />

    <h4>Custom icons</h4>
    <div ng-init="x = 5"><uib-rating ng-model="x" max="15" state-on="'glyphicon-ok-sign
'" state-off="'glyphicon-ok-circle'" aria-labelledby="custom-icons-1"></uib-rating> <b>
(<i>Rate:</i> {{x}})</b></div>
    <div ng-init="y = 2"><uib-rating ng-model="y" rating-states="ratingStates" aria-lab
elledby="custom-icons-2"></uib-rating> <b>(<i>Rate:</i> {{y}})</b></div>
</div>
```

Tabs (ui.bootstrap.tabs (https://github.com/angular-ui/bootstrap/tree/master/src/tabs))

Select a tab by setting active binding to true:

Select second tab Select third tab

Enable / Disable third tab

Static title () Dynamic Title 1 () Dynamic Title 2 () Alert! ()

Static content

Vertical 1 ()

Vertical 2 ()

Vertical content 1

Justified () SJ () Long Justified ()

Justified content

Tabbed pills with CSS classes

Default Size () Small Button ()

Tab 1 content

Tabs using nested forms:

Form Tab () Tab One () Tab Two ()

Name

Tabs

Model:

```
{
  "name": "Tabs"
}
```

Nested Form:

```
{
  "$error": {},
  "$name": "nestedForm",
  "$dirty": false,
  "$pristine": true,
  "$valid": true,
  "$invalid": false,
  "$submitted": false
}
```

AngularJS version of the tabs directive.

uib-tabset settings

- `active` 👁 *(Default:* `Index of first tab` *)* - Active index of tab. Setting this to an existing tab index will make that tab active.
- `justified` **$** *(Default:* `false` *)* - Whether tabs fill the container and have a consistent width.

  - `template-url` *(Default: `uib/template/tabs/tabset.html`)* - A URL representing the location of a template to use for the main component.
- `type` *(Defaults: `tabs`)* - Navigation type. Possible values are 'tabs' and 'pills'.
- `vertical` $ *(Default: `false`)* - Whether tabs appear vertically stacked.

uib-tab settings

- `classes` $ - An optional string of space-separated CSS classes.
- `deselect()` $ - An optional expression called when tab is deactivated. Supports $event in template for expression. You may call `$event.preventDefault()` in this event handler to prevent a tab change from occurring.
- `disable` $ 👁 *(Default: `false`)* - Whether tab is clickable and can be activated.
- `heading` - Heading text.
- `index` - Tab index. Must be unique.
- `select()` $ - An optional expression called when tab is activated. Supports $event in template for expression.
- `template-url` *(Default: `uib/template/tabs/tab.html`)* - A URL representing the location of a template to use for the tab heading.

Tabset heading

Instead of the `heading` attribute on the `uib-tabset`, you can use an `uib-tab-heading` element inside a tabset that will be used as the tabset's header. There you can use HTML as well.

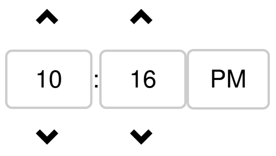
Known issues

To use clickable elements within the tab, you have override the tab template to use div elements instead of anchor elements, and replicate the desired styles from Bootstrap's CSS. This is due to browsers interpreting anchor elements as the target of any click event, which triggers routing when certain elements such as buttons are nested inside the anchor element.

Markup () JavaScript () Edit in plunker

```
<style type="text/css">
  form.tab-form-demo .tab-pane {
    margin: 20px 20px;
  }
</style>

<div ng-controller="TabsDemoCtrl">
  <p>Select a tab by setting active binding to true:</p>
  <p>
    <button type="button" class="btn btn-default btn-sm" ng-click="active = 1">Select s
econd tab</button>
    <button type="button" class="btn btn-default btn-sm" ng-click="active = 2">Select t
hird tab</button>
  </p>
  <p>
    <button type="button" class="btn btn-default btn-sm" ng-click="tabs[1].disabled = !
 tabs[1].disabled">Enable / Disable third tab</button>
  </p>
  <hr />

  <uib-tabset active="active">
    <uib-tab index="0" heading="Static title">Static content</uib-tab>
    <uib-tab index="$index + 1" ng-repeat="tab in tabs" heading="{{tab.title}}" active=
"tab.active" disable="tab.disabled">
      {{tab.content}}
    </uib-tab>
    <uib-tab index="3" select="alertMe()">
      <uib-tab-heading>
        <i class="glyphicon glyphicon-bell"></i> Alert!
      </uib-tab-heading>
      I've got an HTML heading, and a select callback. Pretty cool!
    </uib-tab>
  </uib-tabset>

  <hr />

  <uib-tabset active="activePill" vertical="true" type="pills">
    <uib-tab index="0" heading="Vertical 1">Vertical content 1</uib-tab>
    <uib-tab index="1" heading="Vertical 2">Vertical content 2</uib-tab>
  </uib-tabset>

  <hr />

  <uib-tabset active="activeJustified" justified="true">
    <uib-tab index="0" heading="Justified">Justified content</uib-tab>
    <uib-tab index="1" heading="SJ">Short Labeled Justified content</uib-tab>
    <uib-tab index="2" heading="Long Justified">Long Labeled Justified content</uib-tab
>
  </uib-tabset>

  <hr />
  Tabbed pills with CSS classes
  <uib-tabset type="pills">
    <uib-tab heading="Default Size">Tab 1 content</uib-tab>
    <uib-tab heading="Small Button" classes="btn-sm">Tab 2 content</uib-tab>
  </uib-tabset>

  <hr />

  Tabs using nested forms:
  <form name="outerForm" class="tab-form-demo">
    <uib-tabset active="activeForm">
      <uib-tab index="0" heading="Form Tab">
        <ng-form name="nestedForm">
          <div class="form-group">
            <label>Name</label>
            <input type="text" class="form-control" required ng-model="model.name"/>
          </div>
        </ng-form>
      </uib-tab>
      <uib-tab index="1" heading="Tab One">
        Some Tab Content
      </uib-tab>
      <uib-tab index="2" heading="Tab Two">
        More Tab Content
      </uib-tab>
    </uib-tabset>
  </form>
  Model:
  <pre>{{ model | json }}</pre>
  Nested Form:
  <pre>{{ outerForm.nestedForm | json }}</pre>
</div>
```

Timepicker (ui.bootstrap.timepicker (https://github.com/angular-ui/bootstrap/tree/master/src/timepicker))



```
Time is: 10:16 PM
```

Hours step is:

1

Minutes step is:

15

12H / 24H Set to 14:00 Clear

A lightweight & configurable timepicker directive.

uib-timepicker settings

- `arrowkeys` $ C *(Default: `true`)* - Whether user can use up/down arrow keys inside the hours & minutes input to increase or decrease its values.
- `hour-step` $ C 👁 *(Default: `1`)* - Number of hours to increase or decrease when using a button.
- `max` $ 👁 *(Default: `undefined`)* - Maximum time a user can select.
- `meridians` $ C *(Default: `null`)* - Meridian labels based on locale. To override you must supply an array like `['AM', 'PM']`.
- `min` $ 👁 *(Default: `undefined`)* - Minimum time a user can select
- `minute-step` $ C 👁 *(Default: `1`)* - Number of minutes to increase or decrease when using a button.
- `mousewheel` $ C *(Default: `true`)* - Whether user can scroll inside the hours & minutes input to increase or decrease its values.
- `ng-disabled` $ 👁 *(Default: `false`)* - Whether or not to disable the component.
- `ng-model` $ 👁 - Date object that provides the time state.
- `pad-hours` $ *(Default: true)* - Whether the hours column is padded with a 0.
- `readonly-input` $ C *(Default: `false`)* - Whether user can type inside the hours & minutes input.
- `second-step` $ C 👁 *(Default: `1`)* - Number of seconds to increase or decrease when using a button.
- `show-meridian` $ C 👁 *(Default: `true`)* - Whether to display 12H or 24H mode.
- `show-seconds` $ C 👁 *(Default: `false`)* - Show seconds input.
- `show-spinners` $ C *(Default: `true`)* - Show spinner arrows above and below the inputs.
- `tabindex` *(Defaults: `0`)* - Sets tabindex for each control in the timepicker.
- `template-url` C *(Defaults: `uib/template/timepicker/timepicker.html`)* - Add the ability to override the template used on the component.

Notes

This component makes no claims of absolutely supporting the preservation of dates in all cases, and it is highly recommended that model tracking of dates is encapsulated in a different object. This component should not be used with the same model as the datepicker. This is due to edge cases with situations such as Daylight Savings timezone changes which require a modification of the date in order to prevent an impossible to increment or decrement situation. See #5485 (https://github.com/angular-ui/bootstrap/issues/5485) for details.

If the model value is updated (i.e. via `Date.prototype.setDate`), you must update the model value by breaking the reference by `modelValue = new Date(modelValue)` in order to have the timepicker update.

Markup () JavaScript () Edit in plunker

```
<div ng-controller="TimepickerDemoCtrl">

  <uib-timepicker ng-model="mytime" ng-change="changed()" hour-step="hstep" minute-step
="mstep" show-meridian="ismeridian"></uib-timepicker>

  <pre class="alert alert-info">Time is: {{mytime | date:'shortTime' }}</pre>

  <div class="row">
    <div class="col-xs-6">
        Hours step is:
      <select class="form-control" ng-model="hstep" ng-options="opt for opt in options.
hstep"></select>
    </div>
    <div class="col-xs-6">
        Minutes step is:
      <select class="form-control" ng-model="mstep" ng-options="opt for opt in options.
mstep"></select>
    </div>
  </div>

  <hr>

  <button type="button" class="btn btn-info" ng-click="toggleMode()">12H / 24H</button>
  <button type="button" class="btn btn-default" ng-click="update()">Set to 14:00</butto
n>
  <button type="button" class="btn btn-danger" ng-click="clear()">Clear</button>

</div>
```

Tooltip (ui.bootstrap.tooltip (https://github.com/angular-ui/bootstrap/tree/master/src/tooltip))

Tooltip placement

top

Tooltip top

Dynamic Tooltip Text

dynamic

Dynamic Tooltip Popup Text

Hello, World!

Pellentesque dynamic, sit amet venenatis urna cursus eget nunc scelerisque viverra mauris, in aliquam. Tincidunt lobortis feugiat vivamus at fading eget arcu dictum varius duis at consectetur lorem. Vitae elementum curabitur show delay nunc sed velit dignissim sodales ut eu sem integer vitae. Turpis egestas hide delay pharetra convallis posuere morbi leo urna, Custom template at elementum eu, facilisis sed odio morbi quis commodo odio.

I can even contain HTML. Check me out!

I can have a custom class. Check me out!

Or use custom triggers, like focus:

Click me!

Disable tooltips conditionally:

Hover over this for a tooltip until this is filled

Open tooltips conditionally. Toggle tooltip

A lightweight, extensible directive for fancy tooltip creation. The tooltip directive supports multiple placements, optional transition animation, and more.

Note to mobile developers: Please note that while tooltips may work correctly on mobile devices (including tablets), we have made the decision to not officially support such a use-case because it does not make sense from a UX perspective.

There are three versions of the tooltip: `uib-tooltip`, `uib-tooltip-template`, and `uib-tooltip-html`:

- `uib-tooltip` - Takes text only and will escape any HTML provided.
- `uib-tooltip-html` **$** - Takes an expression that evaluates to an HTML string. Note that this HTML is not compiled. If compilation is required, please use the `uib-tooltip-template` attribute option instead. *The user is responsible for ensuring the content is safe to put into the DOM!*
- `uib-tooltip-template` **$** - Takes text that specifies the location of a template to use for the tooltip. Note that this needs to be wrapped in a tag.

uib-tooltip-* settings

All these settings are available for the three types of tooltips.

- `tooltip-animation` **$** **C** *(Default: `true`, Config: `animation`)* - Should it fade in and out?
- `tooltip-append-to-body` **$** **C** *(Default: `false`, Config: `appendToBody`)* - Should the tooltip be appended to '$body' instead of the parent element?
- `tooltip-class` - Custom class to be applied to the tooltip.
- `tooltip-enable` **$** *(Default: `true`)* - Is it enabled? It will enable or disable the configured tooltip-trigger.
- `tooltip-is-open` 👁 *(Default: `false`)* - Whether to show the tooltip.
- `tooltip-placement` **C** *(Default: `top`, Config: `placement`)* - Passing in 'auto' separated by a space before the placement will enable auto positioning, e.g: "auto bottom-left". The tooltip will attempt to position where it fits in the closest scrollable ancestor. Accepts:

    - `top` - tooltip on top, horizontally centered on host element.
    - `top-left` - tooltip on top, left edge aligned with host element left edge.
    - `top-right` - tooltip on top, right edge aligned with host element right edge.
    - `bottom` - tooltip on bottom, horizontally centered on host element.
    - `bottom-left` - tooltip on bottom, left edge aligned with host element left edge.
    - `bottom-right` - tooltip on bottom, right edge aligned with host element right edge.
    - `left` - tooltip on left, vertically centered on host element.
    - `left-top` - tooltip on left, top edge aligned with host element top edge.
    - `left-bottom` - tooltip on left, bottom edge aligned with host element bottom edge.
    - `right` - tooltip on right, vertically centered on host element.
    - `right-top` - tooltip on right, top edge aligned with host element top edge.
    - `right-bottom` - tooltip on right, bottom edge aligned with host element bottom edge.
- `tooltip-popup-close-delay` **C** *(Default: `0`, Config: `popupCloseDelay`)* - For how long should the tooltip remain open after the close trigger event?
- `tooltip-popup-delay` **C** *(Default: `0`, Config: `popupDelay`)* - Popup delay in milliseconds until it opens.
- `tooltip-trigger` *(Default: `mouseenter`)* - What should trigger a show of the tooltip? Supports a space separated list of event names (see below).

Note: To configure the tooltips, you need to do it on `$uibTooltipProvider` (also see below).

Triggers

The following show triggers are supported out of the box, along with their provided hide triggers:

- `mouseenter` : `mouseleave`
- `click` : `click`
- `outsideClick` : `outsideClick`
- `focus` : `blur`
- `none`

The `outsideClick` trigger will cause the tooltip to toggle on click, and hide when anything else is clicked.

For any non-supported value, the trigger will be used to both show and hide the tooltip. Using the 'none' trigger will disable the internal trigger(s), one can then use the `tooltip-is-open` attribute exclusively to show and hide the tooltip.

$uibTooltipProvider

Through the `$uibTooltipProvider`, you can change the way tooltips and popovers behave by default; the attributes above always take precedence. The following methods are available:

- `setTriggers(obj)` *(Example: `{ 'openTrigger': 'closeTrigger' }`)* - Extends the default trigger mappings mentioned above with mappings of your own.
- `options(obj)` - Provide a set of defaults for certain tooltip and popover attributes. Currently supports the ones with the **C** badge.

Known issues

For Safari 7+ support, if you want to use the **focus** `tooltip-trigger`, you need to use an anchor tag with a tab index. For example:

```
<a tabindex="0" uib-tooltip="Test" tooltip-trigger="focus" class="btn btn-default">
  Click Me
</a>
```

Markup () JavaScript () Edit in plunker

```
<div ng-controller="TooltipDemoCtrl">
    <div class="form-group">
      <label>Tooltip placement</label>
      <select class="form-control" ng-model="placement.selected" ng-options="o as o for
 o in placement.options"></select>
    </div>
    <button tooltip-placement="{{placement.selected}}" uib-tooltip="On the {{placement.
selected}}" type="button" class="btn btn-default">Tooltip {{placement.selected}}</butto
n>

    <hr />
    <div class="form-group">
      <label>Dynamic Tooltip Text</label>
      <input type="text" ng-model="dynamicTooltipText" class="form-control">
    </div>
    <div class="form-group">
      <label>Dynamic Tooltip Popup Text</label>
      <input type="text" ng-model="dynamicTooltip" class="form-control">
    </div>
    <p>
      Pellentesque <a href="#" uib-tooltip="{{dynamicTooltip}}">{{dynamicTooltipText}}<
/a>,
      sit amet venenatis urna cursus eget nunc scelerisque viverra mauris, in
      aliquam. Tincidunt lobortis feugiat vivamus at
      <a href="#" tooltip-animation="false" uib-tooltip="I don't fade. :-(">fading</a>
      eget arcu dictum varius duis at consectetur lorem. Vitae elementum curabitur
      <a href="#" tooltip-popup-delay='1000' uib-tooltip='appears with delay'>show dela
y</a>
      nunc sed velit dignissim sodales ut eu sem integer vitae. Turpis egestas
      <a href="#" tooltip-popup-close-delay='1000' uib-tooltip='hides with delay'>hide
delay</a>
      pharetra convallis posuere morbi leo urna,
      <a href="#" uib-tooltip-template="'myTooltipTemplate.html'">Custom template</a>
      at elementum eu, facilisis sed odio morbi quis commodo odio.
    </p>

    <p>
        I can even contain HTML. <a href="#" uib-tooltip-html="htmlTooltip">Check me ou
t!</a>
    </p>

    <p>
      <style>
        /* Specify styling for tooltip contents */
        .tooltip.customClass .tooltip-inner {
          color: #880000;
          background-color: #ffff66;
          box-shadow: 0 6px 12px rgba(0,0,0,.175);
        }
        /* Hide arrow */
        .tooltip.customClass .tooltip-arrow {
          display: none;
        }
      </style>
      I can have a custom class. <a href="#" uib-tooltip="I can have a custom class app
lied to me!" tooltip-class="customClass">Check me out!</a>
    </p>


    <div class="form-group">
      <label>Or use custom triggers, like focus: </label>
      <input type="text" value="Click me!" uib-tooltip="See? Now click away..." tooltip
-trigger="focus" tooltip-placement="right" class="form-control" />
    </div>

    <div class="form-group" ng-class="{'has-error' : !inputModel}">
      <label>Disable tooltips conditionally:</label>
      <input type="text" ng-model="inputModel" class="form-control"
             placeholder="Hover over this for a tooltip until this is filled"
             uib-tooltip="Enter something in this input field to disable this tooltip"
             tooltip-placement="top"
             tooltip-trigger="mouseenter"
             tooltip-enable="!inputModel" />
    </div>
    <div class="form-group">
      <label>
        Open tooltips <span uib-tooltip="Hello!" tooltip-is-open="tooltipIsOpen" toolti
p-placement="bottom">conditionally.</span>
      </label>
      <button ng-click="tooltipIsOpen = !tooltipIsOpen">Toggle tooltip</button>
    </div>
    <script type="text/ng-template" id="myTooltipTemplate.html">
      <span>Special Tooltip with <strong>markup</strong> and {{ dynamicTooltipText }}</
span>
    </script>
</div>
```

Typeahead (ui.bootstrap.typeahead (https://github.com/angular-ui/bootstrap/tree/master/src/typeahead))

Static arrays

```
Model:
```

Asynchronous results

`Model:`

Locations loaded via $http

ngModelOptions support

`Model:`

Custom templates for results

`Model:`

Custom template

Custom popup templates for typeahead's dropdown

`Model:`

Custom popup template

Typeahead is a AngularJS version of Bootstrap v2's typeahead plugin (http://getbootstrap.com/2.3.2 /javascript.html#typeahead). This directive can be used to quickly create elegant typeaheads with any form text input.

It is very well integrated into AngularJS as it uses a subset of the select directive (http://docs.angularjs.org /api/ng.directive:select) syntax, which is very flexible. Supported expressions are:

- *label* for *value* in *sourceArray*
- *select* as *label* for *value* in *sourceArray*

The `sourceArray` expression can use a special `$viewValue` variable that corresponds to the value entered inside the input.

This directive works with promises, meaning you can retrieve matches using the `$http` service with minimal effort.

uib-typeahead settings

- `ng-model` **$** 👁 - Assignable angular expression to data-bind to.
- `ng-model-options` **$** - Options for ng-model (see ng-model-options directive (https://docs.angularjs.org /api/ng/directive/ngModelOptions)). Currently supports the `debounce` and `getterSetter` options.
- `typeahead-append-to` **$** *(Default:* `null` *)* - Should the typeahead popup be appended to an element instead of the parent element?
- `typeahead-append-to-body` **$** 👁 *(Default:* `false` *)* - Should the typeahead popup be appended to $body instead of the parent element?

- `typeahead-editable` $ 👁 *(Default: `true`)* - Should it restrict model values to the ones selected from the popup only?
- `typeahead-focus-first` $ *(Default: `true`)* - Should the first match automatically be focused as you type?
- `typeahead-focus-on-select` *(Default: `true`)* - On selection, focus the input element the typeahead directive is associated with.
- `typeahead-input-formatter` 👁 *(Default: `undefined`)* - Format the ng-model result after selection.
- `typeahead-is-open` $ 👁 *(Default: `angular.noop`)* - Binding to a variable that indicates if the dropdown is open.
- `typeahead-loading` $ 👁 *(Default: `angular.noop`)* - Binding to a variable that indicates if matches are being retrieved asynchronously.
- `typeahead-min-length` $ 👁 *(Default: `1`)* - Minimal no of characters that needs to be entered before typeahead kicks-in. Must be greater than or equal to 0.
- `typeahead-no-results` $ 👁 *(Default: `angular.noop`)* - Binding to a variable that indicates if no matching results were found.
- `typeahead-on-select($item, $model, $label, $event)` $ *(Default: `null`)* - A callback executed when a match is selected. $event can be undefined if selection not triggered from a user event.
- `typeahead-popup-template-url` *(Default: `uib/template/typeahead/typeahead-popup.html`)* - Set custom popup template.
- `typeahead-select-on-blur` $ *(Default: `false`)* - On blur, select the currently highlighted match.
- `typeahead-select-on-exact` $ *(Default: `false`)* - Automatically select the item when it is the only one that exactly matches the user input.
- `typeahead-show-hint` $ *(Default: `false`)* - Show hint when the first option matches.
- `typeahead-template-url` *(Default: `uib/template/typeahead/typeahead-match.html`)* - Set custom item template.
- `typeahead-wait-ms` $ 👁 *(Default: `0`)* - Minimal wait time after last character typed before typeahead kicks-in.
- `uib-typeahead` $ 👁 - Comprehension Angular expression (see select directive (http://docs.angularjs.org /api/ng.directive:select)).

Markup () JavaScript () Edit in plunker

```
<style>
  .typeahead-demo .custom-popup-wrapper {
    position: absolute;
    top: 100%;
    left: 0;
    z-index: 1000;
    display: none;
    background-color: #f9f9f9;
  }

  .typeahead-demo .custom-popup-wrapper > .message {
    padding: 10px 20px;
    border-bottom: 1px solid #ddd;
    color: #868686;
  }

  .typeahead-demo .custom-popup-wrapper > .dropdown-menu {
    position: static;
    float: none;
    display: block;
    min-width: 160px;
    background-color: transparent;
    border: none;
    border-radius: 0;
    box-shadow: none;
  }
</style>

<script type="text/ng-template" id="customTemplate.html">
  <a>
      <img ng-src="http://upload.wikimedia.org/wikipedia/commons/thumb/{{match.model.fl
ag}}" width="16">
      <span ng-bind-html="match.label | uibTypeaheadHighlight:query"></span>
  </a>
</script>

<script type="text/ng-template" id="customPopupTemplate.html">
  <div class="custom-popup-wrapper"
     ng-style="{top: position().top+'px', left: position().left+'px'}"
     style="display: block;"
     ng-show="isOpen() && !moveInProgress"
     aria-hidden="{{!isOpen()}}">
    <p class="message">select location from drop down.</p>

    <ul class="dropdown-menu" role="listbox">
        <li ng-repeat="match in matches track by $index" ng-class="{active: isActive($i
ndex) }"
            ng-mouseenter="selectActive($index)" ng-click="selectMatch($index)" role="o
ption" id="{{::match.id}}">
            <div uib-typeahead-match index="$index" match="match" query="query" templat
e-url="templateUrl"></div>
        </li>
    </ul>
  </div>
</script>

<div class='container-fluid typeahead-demo' ng-controller="TypeaheadCtrl">

    <h4>Static arrays</h4>
    <pre>Model: {{selected | json}}</pre>
    <input type="text" ng-model="selected" uib-typeahead="state for state in states | f
ilter:$viewValue | limitTo:8" class="form-control">

    <h4>Asynchronous results</h4>
    <pre>Model: {{asyncSelected | json}}</pre>
    <input type="text" ng-model="asyncSelected" placeholder="Locations loaded via $http
" uib-typeahead="address for address in getLocation($viewValue)" typeahead-loading="loa
dingLocations" typeahead-no-results="noResults" class="form-control">
    <i ng-show="loadingLocations" class="glyphicon glyphicon-refresh"></i>
    <div ng-show="noResults">
      <i class="glyphicon glyphicon-remove"></i> No Results Found
    </div>

    <h4>ngModelOptions support</h4>
    <pre>Model: {{ngModelOptionsSelected | json}}</pre>
    <input type="text" ng-model="ngModelOptionsSelected" ng-model-options="modelOptions
" uib-typeahead="state for state in states | filter:$viewValue | limitTo:8" class="form
-control">

    <h4>Custom templates for results</h4>
    <pre>Model: {{customSelected | json}}</pre>
    <input type="text" ng-model="customSelected" placeholder="Custom template" uib-type
ahead="state as state.name for state in statesWithFlags | filter:{name:$viewValue}" typ
eahead-template-url="customTemplate.html" class="form-control" typeahead-show-hint="tru
e" typeahead-min-length="0">

    <h4>Custom popup templates for typeahead's dropdown</h4>
    <pre>Model: {{customPopupSelected | json}}</pre>
    <input type="text" ng-model="customPopupSelected" placeholder="Custom popup templat
e" uib-typeahead="state as state.name for state in statesWithFlags | filter:{name:$view
Value}" typeahead-popup-template-url="customPopupTemplate.html" class="form-control">
</div>
```