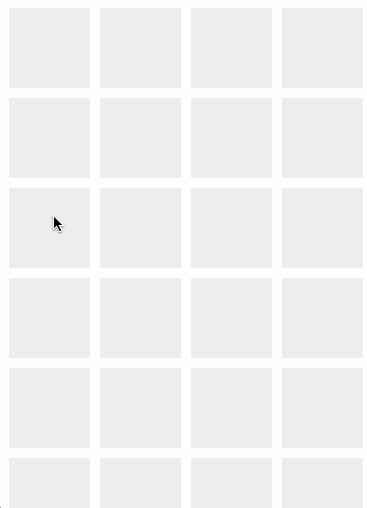
$ionicActionSheet

The Action Sheet is a slide-up pane that lets the user choose from a set of options. Dangerous options are highlighted in red and made obvious.

There are easy ways to cancel out of the action sheet, such as tapping the backdrop or even hitting escape on the keyboard for desktop testing.



Usage

To trigger an Action Sheet in your code, use the $ionicActionSheet service in your angular controllers:

angular.module('mySuperApp', ['ionic'])

.controller(**function**($scope, $ionicActionSheet, $timeout) {

*// Triggered on a button click, or some other target*

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

*// Show the action sheet*

**var** hideSheet **=** $ionicActionSheet.show({

buttons: [

{ text: '<b>Share</b> This' },

{ text: 'Move' }

],

destructiveText: 'Delete',

titleText: 'Modify your album',

cancelText: 'Cancel',

cancel: **function**() {

*// add cancel code..*

},

buttonClicked: **function**(index) {

**return** **true**;

}

});

*// For example's sake, hide the sheet after two seconds*

$timeout(**function**() {

hideSheet();

}, 2000);

};

});

Methods

show(options)

Load and return a new action sheet.

A new isolated scope will be created for the action sheet and the new element will be appended into the body.

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| options | object | The options for this ActionSheet. Properties:   * [Object] buttons Which buttons to show. Each button is an object with a text field. * {string} titleText The title to show on the action sheet. * {string=} cancelText the text for a 'cancel' button on the action sheet. * {string=} destructiveText The text for a 'danger' on the action sheet. * {function=} cancel Called if the cancel button is pressed, the backdrop is tapped or the hardware back button is pressed. * {function=} buttonClicked Called when one of the non-destructive buttons is clicked, with the index of the button that was clicked and the button object. Return true to close the action sheet, or false to keep it opened. * {function=} destructiveButtonClicked Called when the destructive button is clicked. Return true to close the action sheet, or false to keep it opened. * {boolean=} cancelOnStateChange Whether to cancel the actionSheet when navigating to a new state. Default true. * {string} cssClass The custom CSS class name. |

$ionicBackdrop

Shows and hides a backdrop over the UI. Appears behind popups, loading, and other overlays.

Often, multiple UI components require a backdrop, but only one backdrop is ever needed in the DOM at a time.

Therefore, each component that requires the backdrop to be shown calls $ionicBackdrop.retain() when it wants the backdrop, then $ionicBackdrop.release() when it is done with the backdrop.

For each time retain is called, the backdrop will be shown until release is called.

For example, if retain is called three times, the backdrop will be shown until release is called three times.

**Notes:**

* The backdrop service will broadcast ‘backdrop.shown’ and ‘backdrop.hidden’ events from the root scope, this is useful for alerting native components not in html.

Usage

**function** MyController($scope, $ionicBackdrop, $timeout, $rootScope) {

*//Show a backdrop for one second*

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

$ionicBackdrop.retain();

$timeout(**function**() {

$ionicBackdrop.release();

}, 1000);

};

*// Execute action on backdrop disappearing*

$scope.$on('backdrop.hidden', **function**() {

*// Execute action*

});

*// Execute action on backdrop appearing*

$scope.$on('backdrop.shown', **function**() {

*// Execute action*

});

}

Methods

retain()

Retains the backdrop.

release()

Releases the backdrop.

ion-content   
Delegate: [$ionicScrollDelegate](http://ionicframework.com/docs/v1/api/service/$ionicScrollDelegate/)

The ionContent directive provides an easy to use content area that can be configured to use Ionic’s custom Scroll View, or the built in overflow scrolling of the browser.

While we recommend using the custom Scroll features in Ionic in most cases, sometimes (for performance reasons) only the browser’s native overflow scrolling will suffice, and so we’ve made it easy to toggle between the Ionic scroll implementation and overflow scrolling.

You can implement pull-to-refresh with the [ionRefresher](http://ionicframework.com/docs/v1/api/directive/ionRefresher/) directive, and infinite scrolling with the [ionInfiniteScroll](http://ionicframework.com/docs/v1/api/directive/ionInfiniteScroll/) directive.

If there is any dynamic content inside the ion-content, be sure to call .resize() with [$ionicScrollDelegate](http://ionicframework.com/docs/v1/api/service/$ionicScrollDelegate/) after the content has been added.

Be aware that this directive gets its own child scope. If you do not understand why this is important, you can read <https://docs.angularjs.org/guide/scope>.

Usage

<ion-content

[delegate-handle=""]

[direction=""]

[locking=""]

[padding=""]

[scroll=""]

[overflow-scroll=""]

[scrollbar-x=""]

[scrollbar-y=""]

[start-x=""]

[start-y=""]

[on-scroll=""]

[on-scroll-complete=""]

[has-bouncing=""]

[scroll-event-interval=""]>

...

</ion-content>

API

| Attr | Type | Details |
| --- | --- | --- |
| delegate-handle  *(optional)* | string | The handle used to identify this scrollView with [$ionicScrollDelegate](http://ionicframework.com/docs/v1/api/service/$ionicScrollDelegate/). |
| direction  *(optional)* | string | Which way to scroll. 'x' or 'y' or 'xy'. Default 'y'. |
| locking  *(optional)* | boolean | Whether to lock scrolling in one direction at a time. Useful to set to false when zoomed in or scrolling in two directions. Default true. |
| padding  *(optional)* | boolean | Whether to add padding to the content. Defaults to true on iOS, false on Android. |
| scroll  *(optional)* | boolean | Whether to allow scrolling of content. Defaults to true. |
| overflow-scroll  *(optional)* | boolean | Whether to use overflow-scrolling instead of Ionic scroll. See [$ionicConfigProvider](http://ionicframework.com/docs/v1/api/provider/$ionicConfigProvider/) to set this as the global default. |
| scrollbar-x  *(optional)* | boolean | Whether to show the horizontal scrollbar. Default true. |
| scrollbar-y  *(optional)* | boolean | Whether to show the vertical scrollbar. Default true. |
| start-x  *(optional)* | string | Initial horizontal scroll position. Default 0. |
| start-y  *(optional)* | string | Initial vertical scroll position. Default 0. |
| on-scroll  *(optional)* | expression | Expression to evaluate when the content is scrolled. |
| on-scroll-complete  *(optional)* | expression | Expression to evaluate when a scroll action completes. Has access to 'scrollLeft' and 'scrollTop' locals. |
| has-bouncing  *(optional)* | boolean | Whether to allow scrolling to bounce past the edges of the content. Defaults to true on iOS, false on Android. |
| scroll-event-interval  *(optional)* | number | Number of milliseconds between each firing of the 'on-scroll' expression. Default 10. |

ion-refresher   
Child of [ionContent](http://ionicframework.com/docs/v1/api/directive/ionContent/) or [ionScroll](http://ionicframework.com/docs/v1/api/directive/ionScroll/)

Allows you to add pull-to-refresh to a scrollView.

Place it as the first child of your [ionContent](http://ionicframework.com/docs/v1/api/directive/ionContent/) or [ionScroll](http://ionicframework.com/docs/v1/api/directive/ionScroll/) element.

When refreshing is complete, $broadcast the ‘scroll.refreshComplete’ event from your controller.

Usage

<ion-content ng-controller="MyController">

<ion-refresher

pulling-text="Pull to refresh..."

on-refresh="doRefresh()">

</ion-refresher>

<ion-list>

<ion-item ng-repeat="item in items"></ion-item>

</ion-list>

</ion-content>

angular.module('testApp', ['ionic'])

.controller('MyController', **function**($scope, $http) {

$scope.items **=** [1,2,3];

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

$http.get('/new-items')

.success(**function**(newItems) {

$scope.items **=** newItems;

})

.**finally**(**function**() {

*// Stop the ion-refresher from spinning*

$scope.$broadcast('scroll.refreshComplete');

});

};

});

API

| **Attr** | **Type** | **Details** |
| --- | --- | --- |
| on-refresh  *(optional)* | expression | Called when the user pulls down enough and lets go of the refresher. |
| on-pulling  *(optional)* | expression | Called when the user starts to pull down on the refresher. |
| pulling-text  *(optional)* | string | The text to display while the user is pulling down. |
| pulling-icon  *(optional)* | string | The icon to display while the user is pulling down. Default: 'ion-android-arrow-down'. |
| spinner  *(optional)* | string | The [ionSpinner](http://ionicframework.com/docs/v1/api/directive/ionSpinner/) icon to display after user lets go of the refresher. The SVG [ionSpinner](http://ionicframework.com/docs/v1/api/directive/ionSpinner/) is now the default, replacing rotating font icons. Set to none to disable both the spinner and the icon. |
| refreshing-icon  *(optional)* | string | The font icon to display after user lets go of the refresher. This is deprecated in favor of the SVG [ionSpinner](http://ionicframework.com/docs/v1/api/directive/ionSpinner/). |
| disable-pulling-rotation  *(optional)* | boolean | Disables the rotation animation of the pulling icon when it reaches its activated threshold. To be used with a custom pulling-icon. |

# ion-pane

A simple container that fits content, with no side effects. Adds the ‘pane’ class to the element.

## Usage

<ion-pane>

...

</ion-pane>

Forms

# ion-checkbox

The checkbox is no different than the HTML checkbox input, except it’s styled differently.

The checkbox behaves like any [AngularJS checkbox](http://docs.angularjs.org/api/ng/input/input%5bcheckbox%5d).

## Usage

<ion-checkbox ng-model="isChecked">Checkbox Label</ion-checkbox>

ion-radio

The radio directive is no different than the HTML radio input, except it’s styled differently.

Radio behaves like [AngularJS radio](http://docs.angularjs.org/api/ng/input/input%5bradio%5d).

Usage

<ion-radio ng-model="choice" ng-value="'A'">Choose A</ion-radio>

<ion-radio ng-model="choice" ng-value="'B'">Choose B</ion-radio>

<ion-radio ng-model="choice" ng-value="'C'">Choose C</ion-radio>

API

| **Attr** | **Type** | **Details** |
| --- | --- | --- |
| name  *(optional)* | string | The name of the radio input. |
| value  *(optional)* | expression | The value of the radio input. |
| disabled  *(optional)* | boolean | The state of the radio input. |
| icon  *(optional)* | string | The icon to use when the radio input is selected. |
| ng-value  *(optional)* | expression | Angular equivalent of the value attribute. |
| ng-model  *(optional)* | expression | The angular model for the radio input. |
| ng-disabled  *(optional)* | boolean | Angular equivalent of the disabled attribute. |
| ng-change  *(optional)* | expression | Triggers given expression when radio input's model changes |

ion-toggle

A toggle is an animated switch which binds a given model to a boolean.

Allows dragging of the switch’s nub.

The toggle behaves like any [AngularJS checkbox](http://docs.angularjs.org/api/ng/input/input%5bcheckbox%5d) otherwise.

Usage

Below is an example of a toggle directive which is wired up to the airplaneModemodel and has the toggle-calm CSS class assigned to the inner element.

<ion-toggle ng-model="airplaneMode" toggle-class="toggle-calm">Airplane Mode</ion-toggle>

API

| **Attr** | **Type** | **Details** |
| --- | --- | --- |
| toggle-class  *(optional)* | string | Sets the CSS class on the inner label.toggle element created by the directive. |

$ionicModal

Related: [ionicModal controller](http://ionicframework.com/docs/v1/api/controller/ionicModal/).

The Modal is a content pane that can go over the user’s main view temporarily. Usually used for making a choice or editing an item.

Put the content of the modal inside of an <ion-modal-view> element.

**Notes:**

* A modal will broadcast ‘modal.shown’, ‘modal.hidden’, and ‘modal.removed’ events from its originating scope, passing in itself as an event argument. Both the modal.removed and modal.hidden events are called when the modal is removed.
* This example assumes your modal is in your main index file or another template file. If it is in its own template file, remove the script tags and call it by file name.

Usage

<script id="my-modal.html" type="text/ng-template">

**<**ion**-**modal**-**view**>**

**<**ion**-**header**-**bar**>**

**<**h1 **class=**"title"**>**My Modal title**<**/h1>

**<**/ion-header-bar>

**<**ion**-**content**>**

Hello**!**

**<**/ion-content>

**<**/ion-modal-view>

</script>

angular.module('testApp', ['ionic'])

.controller('MyController', **function**($scope, $ionicModal) {

$ionicModal.fromTemplateUrl('my-modal.html', {

scope: $scope,

animation: 'slide-in-up'

}).then(**function**(modal) {

$scope.modal **=** modal;

});

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

$scope.modal.show();

};

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

$scope.modal.hide();

};

*// Cleanup the modal when we're done with it!*

$scope.$on('$destroy', **function**() {

$scope.modal.remove();

});

*// Execute action on hide modal*

$scope.$on('modal.hidden', **function**() {

*// Execute action*

});

*// Execute action on remove modal*

$scope.$on('modal.removed', **function**() {

*// Execute action*

});

});

Methods

fromTemplate(templateString, options)

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| templateString | string | The template string to use as the modal's content. |
| options | object | Options to be passed [ionicModal#initialize](http://ionicframework.com/docs/v1/api/controller/ionicModal/" \l "initialize) method. |

* Returns: object An instance of an [ionicModal](http://ionicframework.com/docs/v1/api/controller/ionicModal/) controller.

fromTemplateUrl(templateUrl, options)

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| templateUrl | string | The url to load the template from. |
| options | object | Options to be passed [ionicModal#initialize](http://ionicframework.com/docs/v1/api/controller/ionicModal/" \l "initialize) method. options object. |

* Returns: promise A promise that will be resolved with an instance of an [ionicModal](http://ionicframework.com/docs/v1/api/controller/ionicModal/) controller.

ionicModal

Instantiated by the [$ionicModal](http://ionicframework.com/docs/v1/api/service/$ionicModal/) service.

Be sure to call [remove()](http://ionicframework.com/docs/v1/api/controller/ionicModal/#remove) when you are done with each modal to clean it up and avoid memory leaks.

Note: a modal will broadcast ‘modal.shown’, ‘modal.hidden’, and ‘modal.removed’ events from its originating scope, passing in itself as an event argument. Note: both modal.removed and modal.hidden are called when the modal is removed.

Methods

initialize(options)

Creates a new modal controller instance.

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| options | object | An options object with the following properties:   * {object=} scope The scope to be a child of. Default: creates a child of $rootScope. * {string=} animation The animation to show & hide with. Default: 'slide-in-up' * {boolean=} focusFirstInput Whether to autofocus the first input of the modal when shown. Will only show the keyboard on iOS, to force the keyboard to show on Android, please use the [Ionic keyboard plugin](https://github.com/ionic-team/ionic-plugin-keyboard#keyboardshow). Default: false. * {boolean=} backdropClickToClose Whether to close the modal on clicking the backdrop. Default: true. * {boolean=} hardwareBackButtonClose Whether the modal can be closed using the hardware back button on Android and similar devices. Default: true. |

show()

Show this modal instance.

* Returns: promise A promise which is resolved when the modal is finished animating in.

hide()

Hide this modal instance.

* Returns: promise A promise which is resolved when the modal is finished animating out.

remove()

Remove this modal instance from the DOM and clean up.

* Returns: promise A promise which is resolved when the modal is finished animating out.

isShown()

* Returns: boolean Whether this modal is currently shown.

ionicPopover

Instantiated by the [$ionicPopover](http://ionicframework.com/docs/v1/api/service/$ionicPopover/) service.

Be sure to call [remove()](http://ionicframework.com/docs/v1/api/controller/ionicPopover/#remove) when you are done with each popover to clean it up and avoid memory leaks.

Note: a popover will broadcast ‘popover.shown’, ‘popover.hidden’, and ‘popover.removed’ events from its originating scope, passing in itself as an event argument. Both the popover.removed and popover.hidden events are called when the popover is removed.

Methods

initialize(options)

Creates a new popover controller instance.

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| options | object | An options object with the following properties:   * {object=} scope The scope to be a child of. Default: creates a child of $rootScope. * {boolean=} focusFirstInput Whether to autofocus the first input of the popover when shown. Default: false. * {boolean=} backdropClickToClose Whether to close the popover on clicking the backdrop. Default: true. * {boolean=} hardwareBackButtonClose Whether the popover can be closed using the hardware back button on Android and similar devices. Default: true. |

show($event)

Show this popover instance.

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| $event | $event | The $event or target element which the popover should align itself next to. |

* Returns: promise A promise which is resolved when the popover is finished animating in.

hide()

Hide this popover instance.

* Returns: promise A promise which is resolved when the popover is finished animating out.

remove()

Remove this popover instance from the DOM and clean up.

* Returns: promise A promise which is resolved when the popover is finished animating out.

isShown()

* Returns: boolean Whether this popover is currently shown.

$ionicPopup

The Ionic Popup service allows programmatically creating and showing popup windows that require the user to respond in order to continue.

The popup system has support for more flexible versions of the built in alert(), prompt(), and confirm() functions that users are used to, in addition to allowing popups with completely custom content and look.

An input can be given an autofocus attribute so it automatically receives focus when the popup first shows. However, depending on certain use-cases this can cause issues with the tap/click system, which is why Ionic prefers using the autofocus attribute as an opt-in feature and not the default.

Usage

A few basic examples, see below for details about all of the options available.

angular.module('mySuperApp', ['ionic'])

.controller('PopupCtrl',**function**($scope, $ionicPopup, $timeout) {

*// Triggered on a button click, or some other target*

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

$scope.data **=** {};

*// An elaborate, custom popup*

**var** myPopup **=** $ionicPopup.show({

template: '<input type="password" ng-model="data.wifi">',

title: 'Enter Wi-Fi Password',

subTitle: 'Please use normal things',

scope: $scope,

buttons: [

{ text: 'Cancel' },

{

text: '<b>Save</b>',

type: 'button-positive',

onTap: **function**(e) {

**if** (**!**$scope.data.wifi) {

*//don't allow the user to close unless he enters wifi password*

e.preventDefault();

} **else** {

**return** $scope.data.wifi;

}

}

}

]

});

myPopup.then(**function**(res) {

console.log('Tapped!', res);

});

$timeout(**function**() {

myPopup.close(); *//close the popup after 3 seconds for some reason*

}, 3000);

};

*// A confirm dialog*

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

**var** confirmPopup **=** $ionicPopup.confirm({

title: 'Consume Ice Cream',

template: 'Are you sure you want to eat this ice cream?'

});

confirmPopup.then(**function**(res) {

**if**(res) {

console.log('You are sure');

} **else** {

console.log('You are not sure');

}

});

};

*// An alert dialog*

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

**var** alertPopup **=** $ionicPopup.alert({

title: 'Don\'t eat that!',

template: 'It might taste good'

});

alertPopup.then(**function**(res) {

console.log('Thank you for not eating my delicious ice cream cone');

});

};

});

Methods

show(options)

Show a complex popup. This is the master show function for all popups.

A complex popup has a buttons array, with each button having a text and type field, in addition to an onTap function. The onTapfunction, called when the corresponding button on the popup is tapped, will by default close the popup and resolve the popup promise with its return value. If you wish to prevent the default and keep the popup open on button tap, call event.preventDefault() on the passed in tap event. Details below.

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| options | object | The options for the new popup, of the form:  {  title: '', // String. The title of the popup.  cssClass: '', // String, The custom CSS class name  subTitle: '', // String (optional). The sub-title of the popup.  template: '', // String (optional). The html template to place in the popup body.  templateUrl: '', // String (optional). The URL of an html template to place in the popup body.  scope: null, // Scope (optional). A scope to link to the popup content.  buttons: [{ // Array[Object] (optional). Buttons to place in the popup footer.  text: 'Cancel',  type: 'button-default',  onTap: function(e) {  // e.preventDefault() will stop the popup from closing when tapped.  e.preventDefault();  }  }, {  text: 'OK',  type: 'button-positive',  onTap: function(e) {  // Returning a value will cause the promise to resolve with the given value.  return scope.data.response;  }  }]  } |

* Returns: object A promise which is resolved when the popup is closed. Has an additional close function, which can be used to programmatically close the popup.

alert(options)

Show a simple alert popup with a message and one button that the user can tap to close the popup.

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| options | object | The options for showing the alert, of the form:  {  title: '', // String. The title of the popup.  cssClass: '', // String, The custom CSS class name  subTitle: '', // String (optional). The sub-title of the popup.  template: '', // String (optional). The html template to place in the popup body.  templateUrl: '', // String (optional). The URL of an html template to place in the popup body.  okText: '', // String (default: 'OK'). The text of the OK button.  okType: '', // String (default: 'button-positive'). The type of the OK button.  } |

* Returns: object A promise which is resolved when the popup is closed. Has one additional function close, which can be called with any value to programmatically close the popup with the given value.

confirm(options)

Show a simple confirm popup with a Cancel and OK button.

Resolves the promise with true if the user presses the OK button, and false if the user presses the Cancel button.

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| options | object | The options for showing the confirm popup, of the form:  {  title: '', // String. The title of the popup.  cssClass: '', // String, The custom CSS class name  subTitle: '', // String (optional). The sub-title of the popup.  template: '', // String (optional). The html template to place in the popup body.  templateUrl: '', // String (optional). The URL of an html template to place in the popup body.  cancelText: '', // String (default: 'Cancel'). The text of the Cancel button.  cancelType: '', // String (default: 'button-default'). The type of the Cancel button.  okText: '', // String (default: 'OK'). The text of the OK button.  okType: '', // String (default: 'button-positive'). The type of the OK button.  } |

* Returns: object A promise which is resolved when the popup is closed. Has one additional function close, which can be called with any value to programmatically close the popup with the given value.

prompt(options)

Show a simple prompt popup, which has an input, OK button, and Cancel button. Resolves the promise with the value of the input if the user presses OK, and with undefined if the user presses Cancel.

$ionicPopup.prompt({

title: 'Password Check',

template: 'Enter your secret password',

inputType: 'password',

inputPlaceholder: 'Your password'

}).then(**function**(res) {

console.log('Your password is', res);

});

| **Param** | **Type** | **Details** |
| --- | --- | --- |
| options | object | The options for showing the prompt popup, of the form:  {  title: '', // String. The title of the popup.  cssClass: '', // String, The custom CSS class name  subTitle: '', // String (optional). The sub-title of the popup.  template: '', // String (optional). The html template to place in the popup body.  templateUrl: '', // String (optional). The URL of an html template to place in the popup body.  inputType: // String (default: 'text'). The type of input to use  defaultText: // String (default: ''). The initial value placed into the input.  maxLength: // Integer (default: null). Specify a maxlength attribute for the input.  inputPlaceholder: // String (default: ''). A placeholder to use for the input.  cancelText: // String (default: 'Cancel'. The text of the Cancel button.  cancelType: // String (default: 'button-default'). The type of the Cancel button.  okText: // String (default: 'OK'). The text of the OK button.  okType: // String (default: 'button-positive'). The type of the OK button.  } |

* Returns: object A promise which is resolved when the popup is closed. Has one additional function close, which can be called with any value to programmatically close the popup with the given value.

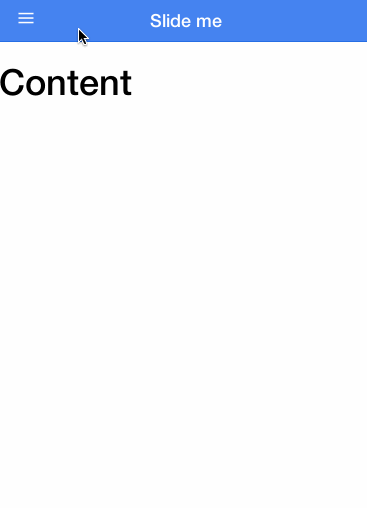
ion-side-menus   
Delegate: [$ionicSideMenuDelegate](http://ionicframework.com/docs/v1/api/service/$ionicSideMenuDelegate/)

A container element for side menu(s) and the main content. Allows the left and/or right side menu to be toggled by dragging the main content area side to side.

To automatically close an opened menu, you can add the [menuClose](http://ionicframework.com/docs/v1/api/directive/menuClose/) attribute directive. The menu-close attribute is usually added to links and buttons within ion-side-menu-content, so that when the element is clicked, the opened side menu will automatically close.

“Burger Icon” toggles can be added to the header with the [menuToggle](http://ionicframework.com/docs/v1/api/directive/menuToggle/) attribute directive. Clicking the toggle will open and close the side menu like the menu-close directive. The side menu will automatically hide on child pages, but can be overridden with the enable-menu-with-back-views attribute mentioned below.

By default, side menus are hidden underneath their side menu content and can be opened by swiping the content left or right or by toggling a button to show the side menu. Additionally, by adding the [exposeAsideWhen](http://ionicframework.com/docs/v1/api/directive/exposeAsideWhen/) attribute directive to an [ionSideMenu](http://ionicframework.com/docs/v1/api/directive/ionSideMenu/) element directive, a side menu can be given instructions about “when” the menu should be exposed (always viewable).



For more information on side menus, check out:

* [ionSideMenuContent](http://ionicframework.com/docs/v1/api/directive/ionSideMenuContent/)
* [ionSideMenu](http://ionicframework.com/docs/v1/api/directive/ionSideMenu/)
* [menuToggle](http://ionicframework.com/docs/v1/api/directive/menuToggle/)
* [menuClose](http://ionicframework.com/docs/v1/api/directive/menuClose/)
* [exposeAsideWhen](http://ionicframework.com/docs/v1/api/directive/exposeAsideWhen/)

Usage

To use side menus, add an <ion-side-menus> parent element. This will encompass all pages that have a side menu, and have at least 2 child elements: 1 <ion-side-menu-content> for the center content, and one or more <ion-side-menu> directives for each side menu(left/right) that you wish to place.

<ion-side-menus>

*<!-- Left menu -->*

<ion-side-menu side="left">

</ion-side-menu>

<ion-side-menu-content>

*<!-- Main content, usually <ion-nav-view> -->*

</ion-side-menu-content>

*<!-- Right menu -->*

<ion-side-menu side="right">

</ion-side-menu>

</ion-side-menus>

**function** ContentController($scope, $ionicSideMenuDelegate) {

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

$ionicSideMenuDelegate.toggleLeft();

};

}

API

| **Attr** | **Type** | **Details** |
| --- | --- | --- |
| enable-menu-with-back-views  *(optional)* | bool | Determines whether the side menu is enabled when the back button is showing. When set to false, any [menuToggle](http://ionicframework.com/docs/v1/api/directive/menuToggle/) will be hidden, and the user cannot swipe to open the menu. When going back to the root page of the side menu (the page without a back button visible), then any menuToggle buttons will show again, and menus will be enabled again. |
| delegate-handle  *(optional)* | string | The handle used to identify this side menu with [$ionicSideMenuDelegate](http://ionicframework.com/docs/v1/api/service/$ionicSideMenuDelegate/). |

ion-side-menu   
Child of [ionSideMenus](http://ionicframework.com/docs/v1/api/directive/ionSideMenus/)

A container for a side menu, sibling to an [ionSideMenuContent](http://ionicframework.com/docs/v1/api/directive/ionSideMenuContent/) directive.

Usage

<ion-side-menu

side="left"

width="myWidthValue + 20"

is-enabled="shouldLeftSideMenuBeEnabled()">

</ion-side-menu>

For a complete side menu example, see the [ionSideMenus](http://ionicframework.com/docs/v1/api/directive/ionSideMenus/) documentation.

API

| **Attr** | **Type** | **Details** |
| --- | --- | --- |
| side | string | Which side the side menu is currently on. Allowed values: 'left' or 'right'. |
| is-enabled  *(optional)* | boolean | Whether this side menu is enabled. |
| width  *(optional)* | number | How many pixels wide the side menu should be. Defaults to 275. |

# menu-toggle

Toggle a side menu on the given side.

## Usage

Below is an example of a link within a nav bar. Tapping this button would open the given side menu, and tapping it again would close it.

<ion-nav-bar>

<ion-nav-buttons side="left">

*<!-- Toggle left side menu -->*

<button menu-toggle="left" class="button button-icon icon ion-navicon"></button>

</ion-nav-buttons>

<ion-nav-buttons side="right">

*<!-- Toggle right side menu -->*

<button menu-toggle="right" class="button button-icon icon ion-navicon"></button>

</ion-nav-buttons>

</ion-nav-bar>

### Button Hidden On Child Views

By default, the menu toggle button will only appear on a root level side-menu page. Navigating in to child views will hide the menu- toggle button. They can be made visible on child pages by setting the enable-menu-with-back-views attribute of the [ionSideMenus](http://ionicframework.com/docs/v1/api/directive/ionSideMenus/) directive to true.

<ion-side-menus enable-menu-with-back-views="true">

# menu-close

menu-close is an attribute directive that closes a currently opened side menu. Note that by default, navigation transitions will not animate between views when the menu is open. Additionally, this directive will reset the entering view’s history stack, making the new page the root of the history stack. This is done to replicate the user experience seen in most side menu implementations, which is to not show the back button at the root of the stack and show only the menu button. We recommend that you also use the enable-menu-with-back-views="false"[ionSideMenus](http://ionicframework.com/docs/v1/api/directive/ionSideMenus/) attribute when using the menuClose directive.

## Usage

Below is an example of a link within a side menu. Tapping this link would automatically close the currently opened menu.

<a menu-close href="#/home" class="item">Home</a>

Note that if your destination state uses a resolve and that resolve asynchronously takes longer than a standard transition (300ms), you’ll need to set the nextViewOptions manually as your resolve completes.

$ionicHistory.nextViewOptions({

historyRoot: **true**,

disableAnimate: **true**,

expire: 300

});

# expose-aside-when  Child of [ionSideMenus](http://ionicframework.com/docs/v1/api/directive/ionSideMenus/)

It is common for a tablet application to hide a menu when in portrait mode, but to show the same menu on the left side when the tablet is in landscape mode. The exposeAsideWhen attribute directive can be used to accomplish a similar interface.

By default, side menus are hidden underneath its side menu content, and can be opened by either swiping the content left or right, or toggling a button to show the side menu. However, by adding the exposeAsideWhen attribute directive to an [ionSideMenu](http://ionicframework.com/docs/v1/api/directive/ionSideMenu/) element directive, a side menu can be given instructions on “when” the menu should be exposed (always viewable). For example, the expose-aside-when="large"attribute will keep the side menu hidden when the viewport’s width is less than 768px, but when the viewport’s width is 768px or greater, the menu will then always be shown and can no longer be opened or closed like it could when it was hidden for smaller viewports.

Using large as the attribute’s value is a shortcut value to (min-width:768px) since it is the most common use-case. However, for added flexibility, any valid media query could be added as the value, such as (min-width:600px) or even multiple queries such as(min-width:750px) and (max-width:1200px).

## Usage

<ion-side-menus>

*<!-- Center content -->*

<ion-side-menu-content>

</ion-side-menu-content>

*<!-- Left menu -->*

<ion-side-menu expose-aside-when="large">

</ion-side-menu>

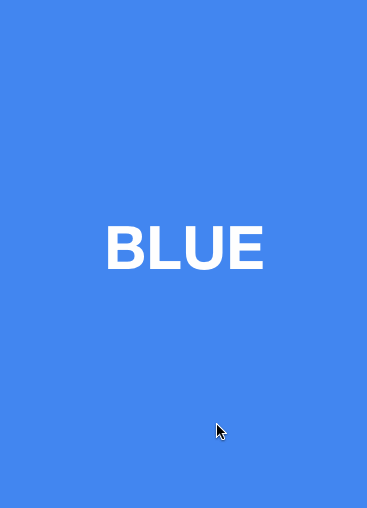
</ion-side-menus>

For a complete side menu example, see the [ionSideMenus](http://ionicframework.com/docs/v1/api/directive/ionSideMenus/) documentation.

ion-slides   
Delegate: [$ionicSlideBoxDelegate](http://ionicframework.com/docs/v1/api/service/$ionicSlideBoxDelegate/)

The Slides component is a powerful multi-page container where each page can be swiped or dragged between.

Note: this is a new version of the Ionic Slide Box based on the [Swiper](http://www.idangero.us/swiper/" \l ".Vmc1J-ODFBc) widget from [idangerous](http://www.idangero.us/).



Usage

<ion-content scroll="false">

<ion-slides options="options" slider="data.slider">

<ion-slide-page>

<div class="box blue"><h1>BLUE</h1></div>

</ion-slide-page>

<ion-slide-page>

<div class="box yellow"><h1>YELLOW</h1></div>

</ion-slide-page>

<ion-slide-page>

<div class="box pink"><h1>PINK</h1></div>

</ion-slide-page>

</ion-slides>

</ion-content>

$scope.options **=** {

loop: **false**,

effect: 'fade',

speed: 500,

}

$scope.$on("$ionicSlides.sliderInitialized", **function**(event, data){

*// data.slider is the instance of Swiper*

$scope.slider **=** data.slider;

});

$scope.$on("$ionicSlides.slideChangeStart", **function**(event, data){

console.log('Slide change is beginning');

});

$scope.$on("$ionicSlides.slideChangeEnd", **function**(event, data){

*// note: the indexes are 0-based*

$scope.activeIndex **=** data.slider.activeIndex;

$scope.previousIndex **=** data.slider.previousIndex;

});

Slide Events

The slides component dispatches events when the active slide changes

|  |  |
| --- | --- |
| $ionicSlides.slideChangeStart | This event is emitted when a slide change begins |
| $ionicSlides.slideChangeEnd | This event is emitted when a slide change completes |
| $ionicSlides.sliderInitialized | This event is emitted when the slider is initialized. It provides access to an instance of the slider. |

Updating Slides Dynamically

When applying data to the slider at runtime, typically everything will work as expected.

In the event that the slides are looped, use the updateLoop method on the slider to ensure the slides update correctly.

$scope.$on("$ionicSlides.sliderInitialized", function(event, data){

// grab an instance of the slider

$scope.slider = data.slider;

});

function dataChangeHandler(){

// call this function when data changes, such as an HTTP request, etc

if ( $scope.slider ){

$scope.slider.updateLoop();

}

}

ion-slide-box   
Delegate: [$ionicSlideBoxDelegate](http://ionicframework.com/docs/v1/api/service/$ionicSlideBoxDelegate/)

The Slide Box is a multi-page container where each page can be swiped or dragged between:

Deprecated API

will be removed in the next Ionic release in favor of the new ion-slides component. Don't depend on the internal behavior of this widget.

Usage

<ion-slide-box on-slide-changed="slideHasChanged($index)">

<ion-slide>

<div class="box blue"><h1>BLUE</h1></div>

</ion-slide>

<ion-slide>

<div class="box yellow"><h1>YELLOW</h1></div>

</ion-slide>

<ion-slide>

<div class="box pink"><h1>PINK</h1></div>

</ion-slide>

</ion-slide-box>

# ion-slide  Child of [ionSlideBox](http://ionicframework.com/docs/v1/nightly/api/directive/ionSlideBox/)

Displays a slide inside of a slidebox.

For more complete examples, see [ionSlideBox](http://ionicframework.com/docs/v1/nightly/api/directive/ionSlideBox/).

## Usage

<ion-slide-box>

<ion-slide>1</ion-slide>

<ion-slide>2</ion-slide>

</ion-slide-box>