# 基本布局

### 标题栏 : ion-header-bar

****ion-header-bar****指令声明一个标题栏元素，标题栏总是位于屏幕的顶部：

1. <ion-header-bar>...</ion-header-bar>

ion-header-bar指令有两个可选的属性：

* ****align-title**** - 设置标题文字的对齐方式。允许值：left | right | center，分别对应左对齐、 右对齐和居中对齐。
* ****no-tap-scroll**** - 当点击标题时，是否将内容区域自动滚动到最开始。允许值：true | false，默认为false。

### 页脚栏 : ion-footer-bar

****ion-footer-bar****指令声明一个页脚栏元素，页脚栏总是位于屏幕的底部：

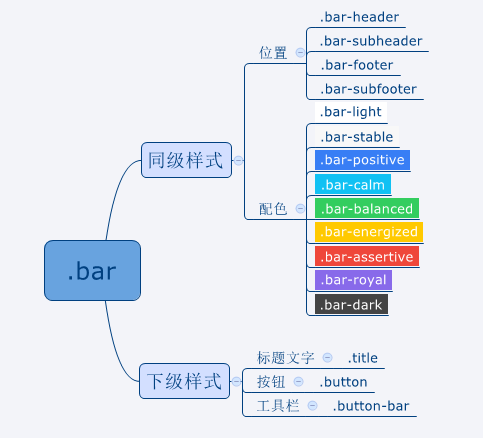
1. <ion-footer-bar>...</ion-footer-bar>

ion-footer-bar指令有一个可选的属性：

* ****align-title**** - 设置标题文本的对齐方式。允许值：left | right | center 。

### header/footer : 样式及内容

ion-header-bar和ion-footer-bar经过编译后其样式类将分别被设置为.bar.bar-header 和.bar.bar-footer，回忆下我们在CSS框架课程中已经了解到的内容：



显然，你可以使用这些样式调整ion-header-bar/ion-footer-bar的外观！

### 内容区 : ion-content

使用****ion-content****指令定义内容区域：

1. <ion-content>...</ion-content>

ion-content占据header和footer以外的剩余区域。当内容超过可视区域时，ion-content 可以滚动以显示被隐藏的部分。

试着滚动右边示例效果的内容区域，你会发现浮现的滚动条。当滚动停止时，浮动条消失。 这是ionic定制的滚动视图，可以使用overflow-scroll属性设置使用系统内置的滚动条：

1. <ion-content overflow-scroll="true">...</ion-content>

### 滚动框 : ion-scroll

****ion-scroll****指令声明一个可滚动的容器元素，用户可以按住内容进行拖动：

1. <ion-scroll>
2. <!--content-->
3. </ion-scroll>

ion-scroll指令有两个常用的可选属性：

* direction - 内容可以滚动的方向。允许值：x|y|xy。默认为 y。
* zooming - 是否支持pinch-to-zoom(捏拉缩放)。允许值：true | false。

在使用ion-scroll时，需要显式指定滚动框元素及内容元素 的大小（高度和宽度）：



### 拉动刷新 : ion-refresher

使用指令****ion-refresher****可以为滚动容器（ion-scroll或ion-content）增加 拉动刷新/pull-to-refresh的功能：

1. <ion-refresher></ion-refresher>

ion-refresher指令有以下可选的属性：

* on-refresh - 当用户向下拉动足够的距离并松开时，执行此表达式
* on-pulling - 当用户开始向下拉动时，执行此表达式
* pulling-text - 当用户向下拉动时，显示此文本
* pulling-icon - 当用户向下拉动时，显示此图标
* refreshing-icon - 当用户向下拉动并松开后，显示的等待图标。ionic推荐使用spinner 代替这个属性
* spinner - 和refreshing-icon的作用一样，但spinner是基于SVG的动画
* disable-pulling-rotation - 禁止下拉图标旋转动画

注意在刷新完毕后，应当使用作用域的$broadcast()方法通知框架：

1. $scope.$broadcast("scoll.refreshComplete")

### 滚动刷新 : ion-infinite-scroll

使用****ion-infinite-scroll****指令可以为滚动容器（ion-scroll或ion-content）增加 滚动刷新功能：

1. <ion-infinite-scroll on-infinite = "">...</ion-infinite-scroll>

ion-infinite-scroll指令有如下属性：

* on-infinite - 必须。当滚动到底部时执行此表达式\* distance - 可选。距底部距离百分比。当距离底部超过此数值时，执行on-infinite。默认为1%
* icon - 可选。载入时显示的图标。默认是ion-load-d。ionic推荐使用spinner代替icon属性
* spinner - 可选。载入时的spinner。默认是ionSpinner
* immediate-check - 可选。是否在载入时立即检查滚动框范围

# Tab切换

### 选项卡 : ion-tabs

使用****ion-tabs****指令声明选项卡，使用****ion-tab****声明选项页：

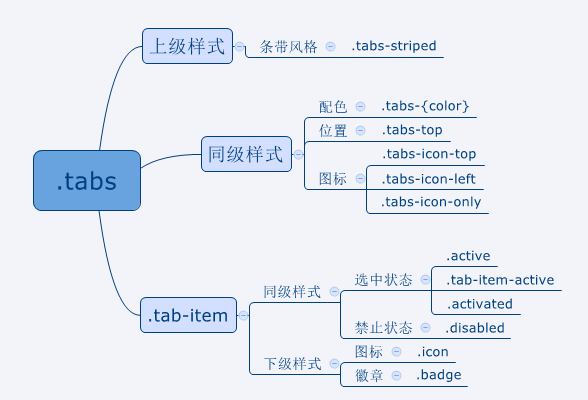
1. <ion-tabs>
2. <ion-tab title="...">...</ion-tab>
3. <ion-tab title="...">...</ion-tab>
4. ...
5. </ion-tabs>

每个ion-tab元素的title属性值将作为选项页的标题其 内容将填充选项卡书签栏之外的剩余区域（被应用.pane样式）。

****注意****：

1. 不要把ion-tabs指令放在ion-content之内
2. ion-tab的内容应当放入ion-view指令内，否则ionic在计算布局时可能出错

AngularJS编译后，ion-tabs元素将被应用****.tabs****样式，因此我们可以使用 相关的样式调整ion-tabs的外观：



### ion-tabs : 声明条带风格

如果学习过课程：ionic之CSS框架，应该记得使用****.tabs-striped****样式可以将选项卡 声明为条带风格：

1. <ion-tabs class="tabs-striped">...</ion-tabs>

也可以通过****$ionicConfigProvider****在AngularJS的配置阶段，将选项卡设置为条带风格：

1. app.config(function($ionicConfigProvider){
2. $ionicConfigProvider.tabs.style("striped"); // 参数可以是： standard | striped
3. })

### ion-tabs : 声明位置

如果学习过课程：ionic之CSS框架，应该记得使用****.tabs-top****可以将选项卡置于 顶部标题栏之下：

1. <ion-tabs class="tabs-top">...</ion-tabs>

也可以通过****$ionicConfigProvider****，在配置阶段设置选项卡的位置是在顶部还是底部：

1. app.config(function($ionicConfigProvider){
2. $ionicConfigProvider.tabs.position("top"); //参数可以是：top | bottom
3. });

### ion-tab ： 标题文字、图标和徽章

ion-tab指令有以下属性用于设置文本、图标和徽章：

* ****title**** - 标题文字

标题文字是必须的。该属性值将作为选项页的标题文字。

* ****icon**** - 标题图标

使用icon属性是可选的，该属性值将用来在标题文字旁边添加一个指定的图标。 这个属性的值将被作为icon-on和icon-off的默认值

* ****icon-on**** - 被选中状态的标题图标

如果一个选项页被选中，ion-tabs将使用icon-on属性的值绘制图标。如果icon-on 没有设置，那么ion-tabs就使用icon属性的值绘制图标

* ****icon-off**** - 未选中状态的标题图标

如果一个选项页没有被选中，ion-tabs将使用icon-off属性的值绘制图标。如果icon-off 没有设置，那么ion-tabs就使用icon属性的值绘制图标

* ****badge**** - 标题徽章

ion-tabs使用badge属性的值在标题文字旁边添加一个圆形的文字标识，通常用来 显示数字。这个属性是可选的，可以是一个具体的值，也可以是当前作用域上的 一个变量

* ****badge-style**** - 标题徽章样式

使用badge-style属性设置徽章的样式， 比如配色方案：barge-{color}

### on-tab : 隐藏、禁止

有些情况下，可能需要临时性地隐藏或禁止某个选项页，ion-tab提供了两个 属性用来实现这个功能：

* ****hidden**** - 隐藏

hidden属性是当前作用域上的表达式。 当其值为true时，选项页将不可见

* ****disabled**** - 禁止

disabled属性是当前作用域上的表达式。当值为true时，选项页将不响应 用户的点击

### ion-tab : 事件

****ion-tab****指令提供了挂接事件的属性：

* ****on-select**** - 选中事件

可选。选项页从未选中状态切换到选中状态时执行此表达式

* ****on-deselect**** - 未选中事件

可选。选项页从选中状态切换到未选中状态时执行此表达式

* ****ng-click**** - 点击事件

可选。选项页被点击时执行此表达式。如果这个属性被设置，那么ion-tabs将不会 自动切换选项页，调用者需要手动调用$ionicTabsDelegate的select()方法进行 选项页切换

### 脚本接口 : $ionicTabsDelegate

使用****$ionicTabsDelegate****服务，我们可以在脚本中控制选项卡对象：

* ****select(index)**** - 选中指定的选项页

index参数从0开始，第一个选项页的index为0，第二个为1，依次类推。

* ****selectedIndex()**** - 返回当前选中选项页的索引号

如果当前没有选中的选项页，则返回 -1。

# 列表

### 列表 : ion-list

列表是常用的信息组织方式。在ionic中，使用****ion-list****指令声明列表 元素，使用****ion-item****指令声明成员元素：

1. <ion-list>
2. <ion-item>...</ion-item>
3. <ion-item>...</ion-item>
4. ...
5. </ion-list>

ion-list指令提供以下属性用来定制列表的外观：

* ****type**** - 列表种类

type属性是可选的，可用来设置列表的种类： list-inset | card。这两种列表 都产生内嵌的效果，区别在于card列表有边框的阴影效果。

* ****show-delete**** - 是否显示成员内的delete按钮

show-delete属性是可选的。如果在成员内有delete按钮（ion-delete-button），使用这个 属性来通知列表是否显示元素删除按钮。允许的值为：true | false

* ****show-reorder**** - 是否显示成员内的reorder按钮

show-reorder属性是可选的。如果在成员内有reorder按钮（ion-reorder-button），使用这个 属性来通知列表是否显示元素重排序按钮。允许的值为：true | false

* ****can-swipe**** - 是否支持滑动方式显示成员option按钮

can-swipe属性是可选的。如果在成员内有option按钮（ion-option-button），使用这个 属性来允许或禁止通过向左滑动成员来打开option按钮。允许的值为：true | false ，默认为true。

### 脚本接口 : $ionicListDelegate

如果需要从脚本中控制列表元素，可以使用****$ionicListDelegate****服务：

* ****showReorder([showReorder])**** - 显示/关闭排序按钮

showReorder的允许值为：true | false。可以使用一个作用域上的表达式

* ****showDelete([showDelete])**** - 显示/关闭删除按钮

showDelete的允许值为： true | false。可以使用一个作用域上的表达式

* ****canSwipeItems([canSwipeItems])**** - 是否允许通过滑动方式来显示成员选项按钮

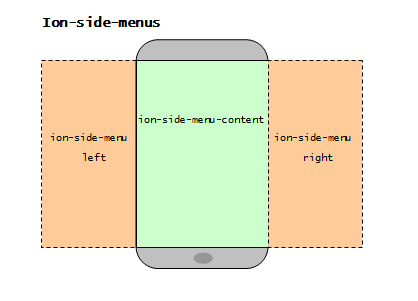
canSwipeItems的允许值为：true | false。可以使用一个作用域上的表达式

* ****closeOptionButtons()**** - 关闭所有选项按钮

# 侧边栏

### 侧边栏菜单 : ion-side-menus

侧边栏菜单是一个最多包含三个子容器的元素：



默认情况下，侧边栏菜单将只显示ion-side-menu-content容器的内容。向左 滑动时，将显示右边栏ion-side-menu容器的内容，向右滑动时，将显示左边栏 ion-side-menu容器的内容。

在ionic中，使用****ion-side-menus****指令定义侧边栏菜单：

1. <ion-side-menus>
2. <ion-side-menu-content>...</ion-side-menu-content>
3. <ion-side-menu side="left">...</ion-side-menu>
4. <ion-side-menu side="right">...</ion-side-menu>
5. </ion-side-menus>

### 切换指令 : menu-toggle/menu-close

****menu-toggle****指令用来给元素增加切换侧栏内容显示状态的功能：

1. <!--切换左侧栏显示状态-->
2. <any menu-toggle="left"></any>
3. <!--切换右侧栏显示状态-->
4. <any menu-toggle="right"></any>

****menu-close****指令用来给元素增加关闭侧栏内容的功能：

1. <any menu-close></any>

与menu-toggle指令不同，menu-close不需要指定要关闭的侧栏是哪一边，它 直接将当前打开的侧栏关闭。

### 侧边栏区域容器 : ion-side-menu

使用指令****ion-side-menu****声明侧边栏区域容器：

1. <ion-side-menu>...</ion-side-menu>

ion-side-menu指令有以下属性：

* ****side**** - 位于内容区的左边或右边

side属性是必须的。允许值：left | right。默认值为left。

* ****width**** - 侧边栏的宽度

width属性是可选的，表示以像素为单位的侧栏菜单宽度。默认为275px。

* ****is-enabled**** - 是否可用

is-enabled属性是可选的，声明该侧边栏是否可用，允许值为：true | false ，默认为true。 当侧边栏不可用时，拖拽内容或使用切换按钮都无法打开侧栏菜单。

* ****expose-aside-when**** - 侧边栏自动显示条件表达式

默认情况下，侧边栏是隐藏的，需要用户向左或向右拖动内容，或者通过一个切换按钮来打开。 但是在有些场景下（比如，横放的平板）屏幕宽度足够大，这时，自动地显示侧边栏内容会更 合理。

expose-aside-when属性就是处理这种情况的，和CSS3的@meida类似，expose-aside-when需要 一个CSS表达式，例如：expose-aside-when="(min-width:500px)"，这意味着当屏幕宽度大于 500px时将自动显示侧栏菜单。

ionic为expose-aside-when提供了一个快捷选项：large ， 这等同于 (min-width:768px)。

### 脚本接口 : $ionicSideMenuDelegate

如果需要在脚本中控制侧边栏菜单，可以使用服务****$ionicSideMenuDelegate****：

* ****toggleLeft([isOpen])**** - 是否打开左侧栏菜单

参数isOpen是可选的，默认为true ，表示打开左侧栏菜单。

* ****toggleRight([isOpen])**** - 是否打开右侧栏菜单

参数isOpen是可选的，默认为true ，表示打开右侧栏菜单。

* ****getOpenRatio()**** - 侧栏菜单打开的宽度占其总宽度比例

例如，一个100px宽的侧栏菜单，如果打开50px，那么其比例为50%，getOpenRatio()将 返回0.5 。

* ****isOpen()**** - 当前侧栏菜单是否打开

不管是左侧栏菜单，还是右侧栏菜单，只要处于打开状态，isOpen()都返回 true。

* ****isOpenLeft()**** - 左侧栏菜单是否打开

当左侧栏菜单处于打开状态时，isOpenLeft()返回true。

* ****isOpenRight()**** - 右侧栏菜单是否打开

当右侧栏菜单处于打开状态时，isOpenRight()返回true。

* ****canDragContent([canDrag])**** - 是否允许拖拽内容以打开侧栏菜单

canDrag参数是可选的，如果canDrag为true，表示允许通过拖拽内容打开侧栏菜单。

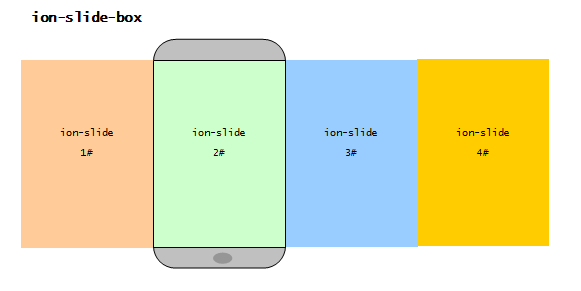
* ****edgeDragThreshold(value)**** - 设置边框距离阈值

当参数value为false或0时，意味着在内容区域任何位置进行拖拽都可以打开侧栏菜单。 如果参数value为一个数值，意味着只有当拖拽发生的位置距边框不大于此数值时，才能打开 侧栏菜单。 参数value为true等同于将value设置为25。

# 幻灯片

### 幻灯片 : ion-slide-box

幻灯片也是一种常见的UI表现方式，它从一组元素中选择一个投射到屏幕可视区域，用户 可以通过滑动方式（向左或向右）进行切换：



在ionic中，使用****ion-slide-box****指令声明幻灯片元素，使用****ion-slide**** 指令声明幻灯页元素：

1. <ion-slide-box>
2. <ion-slide>...</ion-slide>
3. <ion-slide>...</ion-slide>
4. ...
5. </ion-slide-box>

### ion-slide-box : 定制播放行为

指令****ion-slide-box****有一些可选的属性可以定制其播放行为：

* ****does-continue**** - 是否循环切换

你可能注意到，上一节的示例中，开头的幻灯页只能向左滑动，最后的幻灯页只能向右滑动。 将does-continue属性值设为true，就可以让幻灯页组首尾连接起来，循环切换。

* ****auto-play**** - 是否自动播放

通过将auto-play属性设置为true，可以让幻灯页自动切换。切换的间隔默认是4000ms，可以 通过属性****slide-interval****进行调整。

* ****slide-interval**** - 自动播放的间隔时间，默认为4000ms
* ****show-pager**** - 是否显示分页器

分页器用来指示幻灯页的选中状态，位于幻灯片的底部。 允许值为：true | false

### ion-slide-box : 事件及变量

指令****ion-slide-box****提供了可选的用于事件监听的属性：

* ****pager-click**** - 分页器点击事件

pager-click属性应当设置为一个当前作用域上的函数调用表达式，这个函数将被 传入被点击的分页按钮对应的幻灯页序号：index

* ****on-slide-changed**** - 幻灯页切换事件

on-slide-changed属性应当设置为一个当前作用域上的函数调用表达式，这个函数 将被传入当前幻灯页的序号：$index

* ****active-slide**** - 当前幻灯页索引

active-slide属性应当设置为一个当前作用域上的变量，当幻灯片切换时，这个变 量同步的反应当前的幻灯页索引号

### 脚本接口 : $ionicSlideBoxDelegate

可以使用服务****$ionicSlideBoxDelegate****在脚本中操作幻灯片对象：

* ****update()**** - 重绘幻灯片

有时，比如当容器尺寸发生变化时，需要调用update()方法重绘幻灯片。

* ****slide(to[,speed])**** - 切换到指定幻灯页

参数to表示切换的目标幻灯页序号，参数speed是可选的，表示以毫秒 为单位的切换时间

* ****enableSlide([shouldEnable])**** - 幻灯片使能

参数shouldEnable的允许值为：true | false 。

* ****previous()**** - 切换到前一张幻灯页
* ****next()**** - 切换到后一张幻灯页
* ****currentIndex()**** - 获得当前幻灯页的序号
* ****slideCount()**** - 获得全部幻灯页的数量

# 表单输入

### 复选按钮 : ion-checkbox

ionic的复选按钮脱胎于HTML标准的****checkbox****元素，可以在一组选项中 同时选中多个：



使用****ion-checkbox****指令声明复选按钮元素：

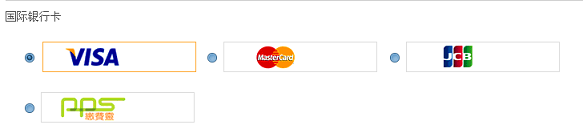
1. <ion-checkbox>...</ion-checkbox>

与标准的checkbox相比，使用ionic的****ion-checkbox****有几个优势：

1. 样式更美观，与移动端的UI更匹配
2. 支持数据绑定。使用可选的ng-model属性，可以直接将选中状态绑定到作用域上的变量
3. 直接使用文本子元素作为标签

### 单选按钮 ：ion-radio

和复选按钮类似，ionic的单选按钮脱胎于HTML标准的****radio****元素，用来在一组互斥的 选项中进行选择：



使用****ion-radio****声明单选按钮元素：

1. <ion-radio>...</ion-radio>

和HTML的radio相比，ion-radio的改进也是明显的：

1. 与移动端的UI更匹配
2. 使用可选的ng-model属性，实现与作用域变量的数据绑定
3. 使用可选的ng-value属性，可以使用作用域变量设置单选按钮对应的值

### 开关按钮 : ion-toggle

开关和复选按钮的作用一致，都是从一组选项中进行多重选择。

在ionic中，使用****ion-toggle****指令声明开关元素：

1. <ion-toggle></ion-toggle>

ion-toggle有两个可选的属性：

* ****ng-model**** - 模型变量

和复选按钮一样，开关按钮也可以使用可选的ng-model属性实现与作用域变量 的双向绑定。

* ****toggle-class**** - 样式类

可以使用可选的toggle-class属性为开关按钮声明额外的样式。比如：toggle-{color} 用来声明配色方案。

### 等待指示器 : ion-spinner

当App在进行耗时地处理时，给用户一个明显的提示反馈，可以营造良好的体验。

ionic的spinner使用SVG和JavaScript实现，因此可以缩放、添加样式并具有动画效果。

使用****ion-spinner****指令声明spinner对象：

1. <ion-spinner></ion-spinner>

ion-spinner的icon属性是可选的，用来指定SVG图标名称。请注意，这里的名称不是 ionioncs的图标名称。当前支持的SVG图标名称请参见示例代码。

ionic的预置样式类：spinner-{color}可以用来声明配色方案，也可以 自定义样式，比如设置大小等：

1. .spinner svg{
2. width: 28px;
3. height: 28px;
4. stroke: #444;
5. fill: #444;
6. }

### 模态对话框 : $ionicModal

模态对话框常用来供用户进行选择或编辑，在模态对话框关闭之前，其他 的用户交互行为被阻止。

在ionic中使用模态对话框有三个步骤：

****1.声明对话框模板****

使用ion-modal-view指令声明对话框模板，对话框模板通常置入 script内以构造内联模板：

1. <script id="a.html" type="text/ng-template">
2. <ion-modal-view>
3. <!--对话框内容-->
4. </ion-modal-view>
5. </script>

****2.创建对话框对象****

服务$ionicModal有两个方法用来创建对话框对象：

* fromTemplate(templateString,options) - 使用字符串模板
* fromTemplateUrl(templateUrl,options) - 使用内联模板

这两个方法返回的都是一个对话框对象。

****3.操作对话框对象****

对象框对象有以下方法用于显示、隐藏或删除对话框：

* show() - 显示对话框
* hide() - 隐藏对话框
* remove() - 移除对话框
* isShown() - 对话框是否可视？

### 上拉菜单 : $ionicActionSheet

上拉菜单是一个自屏幕底部向上滑出的菜单，通常用来让用户做出选择。

ionic的上拉菜单由三种按钮组成，点击任何按钮都自动关闭上拉菜单：

* 取消按钮 - 取消按钮总是位于菜单的底部，用户点击该按钮将关闭。一个上拉菜单 最多有一个取消按钮。
* 危险选项按钮 - 危险选项按钮文字被标红以明显提示。一个上拉菜单最多有一个 危险选项按钮。
* 自定义按钮 - 用户定义的任意数量的按钮。

在ionic中使用上拉菜单需要遵循以下步骤：

****1.定义上拉菜单选项****

使用一个JSON对象定义上拉菜单选项，包括以下字段：

* titleText - 上拉菜单的标题文本
* buttons - 自定义按钮数组。每个按钮需要一个描述对象，其text字段用于按钮显示
* cancelText - 取消按钮的文本。如果不设置此字段，则上拉菜单中不出现取消按钮
* destructiveText - 危险选项按钮的文本。如果不设置此字段，则上拉菜单中不出现危险选项按钮
* buttonClicked - 自定义按钮的回调函数，当用户点击时触发
* cancel - 取消按钮回调函数，当用户点击时触发
* destructiveButtonClicked - 危险选项按钮的回调函数，当用户点击时触发
* cancelOnStateChange - 当切换到新的视图时是否关闭此上拉菜单。默认为true
* cssClass - 附加的CSS样式类名称

****2.创建上拉菜单****

****$ionicActionSheet****服务的show()方法用来创建上拉菜单，返回一个函数，调用该 返回函数可以关闭此菜单。

### 弹出框 : $ionicPopup

弹出框通常用于提醒、警告等，在用户响应之前其他交互行为不能继续。与模态 对话框覆盖整个屏幕空间不同，弹出框通常仅占据一部分屏幕空间。

在ionic中，使用****$ionicPopup****服务管理弹出框：

1. $ionicPopup.show(options)
2. .then(function(){
3. //这个函数在弹出框关闭时被调用
4. });

****show()****方法返回的是一个promise对象，当弹出框关闭后，该对象被解析，这意味着 then()方法指定的参数函数此时将被调用。

show()方法的参数****options****是一个JSON对象，可以包括以下字段：

* title - 弹出框标题文本
* subTitle - 弹出框副标题文本
* template - 弹出框内容的字符串模板
* templateUrl - 弹出框内容的内联模板URL
* scope - 要关联的作用域对象
* buttons - 自定义按钮数组。按钮总是被置于弹出框底部
* cssClass - 附加的CSS样式类

### 简化的特定弹出框

除了show()方法，$ionicPopup还针对一些特定场景提供了简化的方法，这些方法不需要 自定义按钮，只需要设置title和template即可：

* ****alert(options)**** - 警告弹出框，仅包含一个按钮供关闭弹出框
* ****confirm(options)**** - 确认弹出框，包含一个取消按钮和一个确认按钮
* ****prompt(options)**** - 输入提示弹出框，包含一个文本输入框、一个取消按钮和一个确认按钮