**Ionic2 Css组件总结**

# Buttons(按钮)

按钮默认：

<button *ion-button color=*"secondary">上拉菜单</button>

其中 ion-button 必须加上，该指令是用来强化按钮标志，如果去掉该指令，按钮的样式无效果。color 属性用来修改按钮的颜色，其颜色值可以是已有的scss变量，也可以是js定义的变量，不能是直接的颜色值，如‘#ddd’。可以赋值的属性加上[]则可以是js中定义的变量或者表达式，如[color]="isDanger ? 'danger' : 'primary'".

## 带外边框按钮

增加 outline属性，可赋值boolean 类型，如 [outline]=”true”：

<button *ion-button* *outline*>outline按钮</button>



## 圆角按钮

增加round 属性，可赋值boolean 类型：

<button *ion-button color=*"secondary" *round*>round按钮</button>



## 干净的按钮

增加clear 属性，可赋值boolean 类型：

<button *ion-button color=*"secondary" *clear*>clear按钮</button>



## 块级按钮

增加block属性，可赋值boolean 类型：

<button *ion-button color=*"secondary" *block*>block按钮</button>



## 全宽按钮

增加full属性，可赋值boolean 类型：

<button *ion-button color=*"secondary" *full*>full按钮</button>



## 小按钮

增加small属性，可赋值boolean 类型：

<button *ion-button color=*"secondary" *small*>small按钮</button>



## 大按钮

增加large属性，可赋值boolean 类型：

<button *ion-button color=*"secondary" *large*>large按钮</button>



## 带图标的按钮

增加icon-left 或者icon-right属性：

<button *ion-button icon-left*><ion-icon *name=*"home"></ion-icon>按钮</button>



增加icon-only属性，表示只有图标的按钮：

<button *ion-button icon-only*><ion-icon *name=*"home"></ion-icon></button>



## 列表中的按钮

<ion-list>  
 <ion-item>  
 Left Icon Button  
 <button *ion-button outline item-end icon-left*>  
 <ion-icon *name=*"star"></ion-icon>  
 Left Icon  
 </button>  
 </ion-item>  
</ion-list>



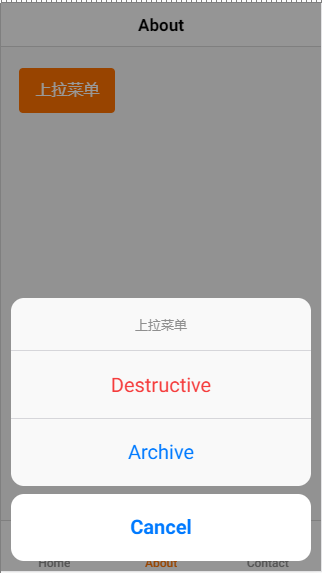
# 上拉菜单

上拉菜单是由一组按钮组成，其中危险角色的按钮在ios上会突出显示。可以通过点击背景或者esc按钮或者取消角色按钮关闭上拉菜单。还可以通过setTitle()方法设置标题，addButton()方法添加按钮。代码如下：

import { ActionSheetController } from 'ionic-angular';

constructor(public actionSheetCtrl: ActionSheetController) {}

presentActionSheet() {  
 let actionSheet = this.actionSheetCtrl.create({  
 title: '上拉菜单',  
 buttons: [  
 {  
 text:'Destructive',  
 role:'destructive',  
 handler: () => {  
 console.log('点击了按钮Destructive');  
 }  
 }, {  
 text:'Archive',  
 handler: () => {  
 console.log('点击了按钮Archive');  
 }  
 }, {  
 text:'Cancel',  
 role: 'cancel',  
 handler: () => {  
 console.log('Cancel');  
 }  
 }  
 ]  
 });  
 actionSheet.present();  
}



上拉菜单默认点击按钮关闭，如果按钮的handle()方法中return false ，则点击该按钮菜单不会消失。如果涉及到关闭后页面跳转等逻辑处理，可以采用以下方法：

handler: () => {  
 console.log('点击了按钮Destructive');  
 let navTransition = actionSheet.dismiss();  
 navTransition.then(() => {  
 console.log('关闭');  
 });  
 return false; //此句代码很关键，不可少  
}

**创建上拉菜单选项create(opts):**

|  |  |  |
| --- | --- | --- |
| 选项Option | 类型 | 备注 |
| title | string | Action sheet 的标题 |
| subTitle | string | 副标题 |
| cssClass | string | 定制样式，空格隔开 |
| enableBackdropDismiss | boolean | 是否通过点击背景关闭上拉菜单 |
| buttons | Array<any> | 要显示的一组按钮 |

**上拉菜单按钮选项**

|  |  |  |
| --- | --- | --- |
| 选项Option | 类型 | 备注 |
| text | string | 按钮内容 |
| icon | icon | 按钮的图标，填写图标名，如‘ios-bowtie’ |
| handler | any | 点击按钮时的处理方法 |
| cssClass | string | 按钮样式 |
| role | string | 按钮角色，可让按钮按照角色来展示样式。destructive或者cancel，如果没有配置该选项，默认按钮没有任何的样式 |

当关闭上拉菜单后需要页面切换时，最好先等待上拉菜单关闭之后再切换，否则会出现跳转异常。如下：

let actionSheet = this.actionSheetCtrl.create({

title: 'Hello',

buttons: [{

text: 'Ok',

handler: () => {

// user has clicked the action sheet button

// begin the action sheet's dimiss transition

let navTransition = actionSheet.dismiss();

// start some async method

someAsyncOperation().then(() => {

// once the async operation has completed

// then run the next nav transition after the

// first transition has finished animating out

navTransition.then(() => {

this.nav.pop();

});

});

return false;

}

}]

});

actionSheet.present();

# 弹出框

引入AlertController 控制器，并在class构造器中注入。

## 基础弹窗

一般显示提示或者警告信息。

alerts() {  
 let alert = this.alertCtrl.create({  
 title: '提示',  
 subTitle: '提示副标题！',  
 message: '详细信息',  
 buttons: ['确定']  
 });  
 alert.present();  
}



## 对话弹出框

prompt() {  
 let prompt = this.alertCtrl.create({  
 title:'密码输入',  
 message:'请输入密码',  
 inputs: [{  
 name:'pws',  
 placeholder:'六位数字'  
 }],  
 buttons: [{  
 text:'取消',  
 handler: data => {  
 console.log('点击取消');  
 }  
 },{  
 text:'确定',  
 handler: data => {  
 console.log('点击确定');  
 }  
 }]  
 });  
  
 prompt.present();  
}



## 确认弹出框

confirm() {  
 let confirm = this.alertCtrl.create({  
 title:'是否确认？',  
 message:'确认后将不能撤销，是否确认？',  
 buttons: [{  
 text:'取消',  
 handler: data => {  
 console.log('点击取消');  
 }  
 },{  
 text:'确认',  
 handler: data => {  
 console.log('点击确定');  
 }  
 }]  
 });  
  
 confirm.present();  
}



## 单选弹出框

radio() {  
 let dataList = [{  
 label: 'Blue',  
 value: 'blue',  
 checked: false  
 },{  
 label: 'Green',  
 value: 'green',  
 checked: true  
 }]  
 let alert = this.alertCtrl.create();  
 alert.setTitle('颜色类型');  
  
 for(var radio of dataList){  
 alert.addInput({  
 type: 'radio',  
 label: radio.label,  
 value: radio.value,  
 checked: radio.checked  
 });  
 }  
 alert.addButton('取消');  
 alert.addButton({  
 text: '确定',  
 handler: data => {  
 console.log(data);  
 }  
 });  
 alert.present();  
}



## 复选弹出框

checkbox() {  
 let dataList = [{  
 label: 'Alderaan',  
 value: 'value1',  
 checked: false  
 },{  
 label: 'Fraden',  
 value: 'value2',  
 checked: true  
 }]  
 let alert = this.alertCtrl.create();  
 alert.setTitle('你想参观哪里？');  
  
 for(var checkbox of dataList){  
 alert.addInput({  
 type: 'checkbox',  
 label: checkbox.label,  
 value: checkbox.value,  
 checked: checkbox.checked  
 });  
 }  
  
 alert.addButton('取消');  
 alert.addButton({  
 text: '确定',  
 handler: data => {  
 console.log('Checkbox data:', data);  
 }  
 });  
 alert.present();  
}



# Badges标识符

标识符是一种小的组件，通常会向用户传递数值。它们通常在一个列表中使用。

<ion-item>  
 <ion-icon *name=*"logo-twitter" *item-start*></ion-icon>  
 Followers  
 <ion-badge *item-end*>260k</ion-badge>  
</ion-item>



# 五、卡片

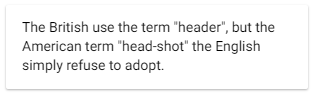
卡片(card)默认样式带有box-shadow(阴影)，主要是一个css组件。卡片组件需添加<ion-card>标签包裹，卡片的内容需放在<ion-card-content>中。

## 1.基础卡片

<ion-card>  
 <ion-card-content>

The British use the term "header", but the American term "head-shot" the English simply refuse to adopt.

</ion-card-content>  
</ion-card>



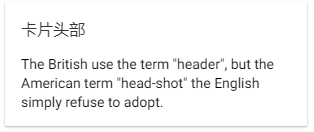
## 带有头部的卡片

卡片头部内容放在<ion-card-header>标签中。Ionic2中没有卡片底部的样式。

<ion-card>  
 <ion-card-header>卡片头部</ion-card-header>  
 <ion-card-content>

The British use the term "header", but the American term "head-shot" the English simply refuse to adopt.

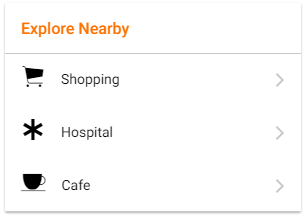
</ion-card-content>  
</ion-card>



## 列表卡片

卡片中可以包含列表，添加一个ion-list组件代替ion-card-content来显示一个列表。

<ion-card>  
 <ion-card-header *color=*"primary">Explore Nearby</ion-card-header>  
  
 <ion-list>  
 <button *ion-item*>  
 <ion-icon *name=*"cart" *item-start*></ion-icon>  
 Shopping  
 </button>  
 </ion-list>  
</ion-card>

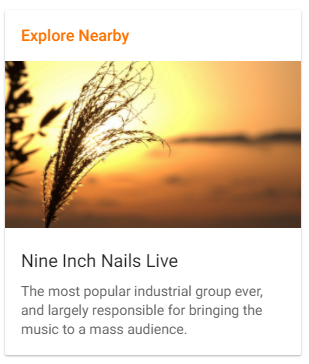


## 图像卡片

图像通常大小不一，所以在整个应用程序中采用一致的样式是很重要的。在卡片上添加图像将给图像一个不变的宽度和一个可变的高度。

<ion-card>

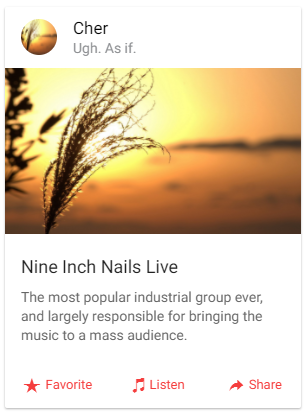
<ion-card-header *color=*"primary">Explore Nearby</ion-card-header>  
 <img *src=*"assets/imgs/card.jpg" />  
 <ion-card-content>  
 <ion-card-title>Nine Inch Nails Live</ion-card-title>  
 <p>  
 The most popular industrial group ever, and largely  
 responsible for bringing the music to a mass audience.  
 </p>  
 </ion-card-content>  
</ion-card>



## 实例

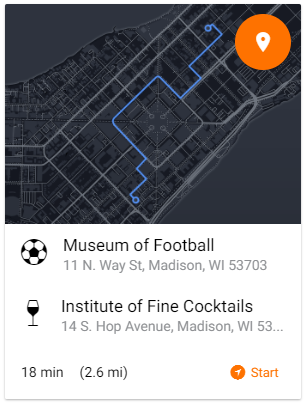
卡片可以被用于各种设计中，也可与其它样式组件搭配使用。如下使用图像列表、图像卡片与栅格系统：

<ion-card>  
 <ion-item>  
 <ion-avatar *item-start*><img *src=*"assets/imgs/card.jpg"></ion-avatar>  
 <h2>Cher</h2>  
 <p>Ugh. As if.</p>  
 </ion-item>  
 <img *src=*"assets/imgs/card.jpg" />  
 <ion-card-content>  
 <ion-card-title>Nine Inch Nails Live</ion-card-title>  
 <p>  
 The most popular industrial group ever, and largely  
 responsible for bringing the music to a mass audience.  
 </p>  
 </ion-card-content>  
 <ion-row *no-padding*>  
 <ion-col>  
 <button *ion-button clear small color=*"danger" *icon-start*>  
 <ion-icon *name=*'star'></ion-icon>  
 Favorite  
 </button>  
 </ion-col>  
 <ion-col *text-center*>  
 <button *ion-button clear small color=*"danger" *icon-start*>  
 <ion-icon *name=*'musical-notes'></ion-icon>  
 Listen  
 </button>  
 </ion-col>  
 <ion-col *text-right*>  
 <button *ion-button clear small color=*"danger" *icon-start*>  
 <ion-icon *name=*'share-alt'></ion-icon>  
 Share  
 </button>  
 </ion-col>  
 </ion-row>  
</ion-card>



如下使用各种组件混合打造地图卡片：

<ion-card *style=*"position:relative;">  
 <img *src=*"assets/imgs/map.png">  
 <ion-fab *right top*>  
 <button *ion-fab*><ion-icon *name=*"pin"></ion-icon></button>  
 </ion-fab>  
  
 <ion-item>  
 <ion-icon *name=*"football" *item-start large*></ion-icon>  
 <h2>Museum of Football</h2>  
 <p>11 N. Way St, Madison, WI 53703</p>  
 </ion-item>  
  
 <ion-item>  
 <ion-icon *name=*"wine" *item-left large* ></ion-icon>  
 <h2>Institute of Fine Cocktails</h2>  
 <p>14 S. Hop Avenue, Madison, WI 53703</p>  
 </ion-item>  
  
 <ion-item>  
 <span *item-left*>18 min</span>  
 <span *item-left*>(2.6 mi)</span>  
 <button *ion-button icon-left clear item-end*>  
 <ion-icon *name=*"navigate"></ion-icon>  
 Start  
 </button>  
 </ion-item>  
  
</ion-card>



# 六、表单

Ionic更倾向于在元素<label> 外创建 item-input , 这样无论任何某一行的任何部位被点击的时候都可以进入输入状态。

## 1. 输入框

### 默认样式

基础表单默认为100%宽度（左右两侧没有边框），并使用 placeholder 属性设置输入字段预期值的提示信息。一般与<ion-item>组合使用。

<ion-list>  
 <ion-item>  
 <ion-input *type=*"text" *placeholder=*"userName"></ion-input>  
 </ion-item>  
</ion-list>



### 固定行内标签

如果输入框带有标签，则需要在<ion-label>标签中添加‘fixed’属性，将标签固定在左侧，标签左侧宽度固定为100px，超出省略，无论标签的长度如何，输入都将在相同的位置上对齐。

<ion-list>  
 <ion-item>  
 <ion-label *fixed*>Username</ion-label>  
 <ion-input *type=*"text" *value=*""></ion-input>  
 </ion-item>  
 <ion-item>  
 <ion-label *fixed*>Password</ion-label>  
 <ion-input *type=*"password"></ion-input>  
 </ion-item>  
</ion-list>



### 行内标签

行内标签输入框与固定行内标签样式相同，两者区别在于行内标签宽度不固定，最大宽度为200px，更适用于输入框内容右对齐的排版。

<ion-list>  
 <ion-item>  
 <ion-label>Username</ion-label>  
 <ion-input *placeholder=*"Username" *type=*"text"></ion-input>  
 </ion-item>  
 <ion-item>  
 <ion-label>Password</ion-label>  
 <ion-input *placeholder=*"Password" *type=*"password"></ion-input>  
 </ion-item>  
</ion-list>

### 内嵌标签

默认情况下，每个输入项将填充父元素的100%宽度。通过将inset属性添加到ion-list组件，使输入内置。内嵌样式与行内样式差不多，仅仅是外层list多了16px的margin距离。

<ion-list *inset*>  
 <ion-item>  
 <ion-label>Username</ion-label>  
 <ion-input *placeholder=*"Username" *type=*"text"></ion-input>  
 </ion-item>  
 <ion-item>  
 <ion-label>Password</ion-label>  
 <ion-input *placeholder=*"Password" *type=*"password"></ion-input>  
 </ion-item>  
</ion-list>



### 堆叠标签

堆叠标签通常位于输入框的头部。每个选项在<ion-label>标签中添加‘stacked’属性。 每个输入框需要指定 input-label。

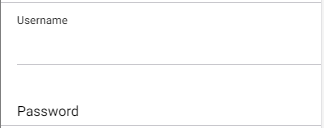
<ion-list>  
 <ion-list-header>堆叠输入框</ion-list-header>  
 <ion-item>  
 <ion-label *stacked*>Username</ion-label>  
 <ion-input *placeholder=*"Username" *type=*"text"></ion-input>  
 </ion-item>  
 <ion-item>  
 <ion-label *stacked*>Password</ion-label>  
 <ion-input *placeholder=*"Password" *type=*"password"></ion-input>  
 </ion-item>  
</ion-list>



### 浮动标签

浮动标签类似于堆叠标签，但浮动标签有一个动画的效果，当输入被激活时标签会往上浮动。在<ion-label>标签中添加‘floating’属性即可。

<ion-list>  
 <ion-list-header>浮动输入框</ion-list-header>  
 <ion-item>  
 <ion-label *floating*>Username</ion-label>  
 <ion-input *type=*"text"></ion-input>  
 </ion-item>  
 <ion-item>  
 <ion-label *floating*>Password</ion-label>  
 <ion-input *type=*"password"></ion-input>  
 </ion-item>  
</ion-list>



### 搜索输入框

搜索输入框标签为<ion-searchbar>，输入框中包含一个搜索的图标，通过‘showCancelButton’属性来控制是否显示取消按钮，这个属性只在输入框聚焦时显示取消按钮。该标签也可通过ionInput属性来绑定change事件。

<ion-searchbar *showCancelButton (ionInput)=*"onInput($event)"></ion-searchbar>

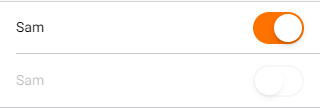




## 2. 切换开关

切换开关包含一个布尔值输入，表示滑块的状态。‘value’、‘disabled’、‘checked’属性均可应用在该组件上。

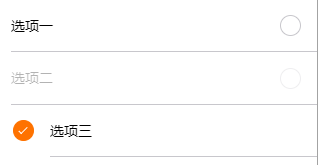
<ion-item>  
 <ion-label>Sam</ion-label>  
 <ion-toggle *checked=*"true"></ion-toggle>  
</ion-item>  
<ion-item>  
 <ion-label>Sam</ion-label>  
 <ion-toggle *disabled*></ion-toggle>  
</ion-item>



## 3. 复选框

复选框的标签为<ion-checkbox>，一般与<ion-item>一起组合使用。与<ion-item>组合使用时，默认选择框放在左边，如果需要将选择框放在右边，则在<ion-checkbox>中添加‘item-right’或者‘item-end’。添加‘disabled’表示复选框不可用。

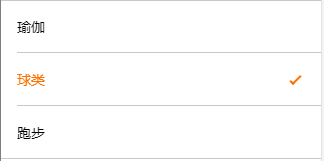
<ion-item>  
 <ion-label>选项一</ion-label>  
 <ion-checkbox *item-right*></ion-checkbox>  
</ion-item>  
<ion-item>  
 <ion-label>选项二</ion-label>  
 <ion-checkbox *disabled item-end*></ion-checkbox>  
</ion-item>  
<ion-item>  
 <ion-label>选项三</ion-label>  
 <ion-checkbox *checked=*"true"></ion-checkbox>  
</ion-item>



## 4. 单选框

与复选框一样，单选框是一个包含布尔值的输入组件，在不同的平台上运行会有不同的样式。ionic 单选按钮与标准 type="radio" 的 input元素类似。 通过‘checked’属性设置是否选中。在<ion-list>标签上添加‘radio-group’属性表示该列表下的单选为同一组。

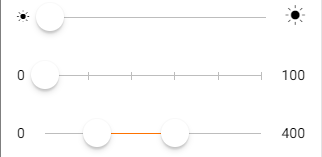
<ion-list *radio-group*>  
 <ion-item>  
 <ion-label>瑜伽</ion-label>  
 <ion-radio></ion-radio>  
 </ion-item>  
 <ion-item>  
 <ion-label>球类</ion-label>  
 <ion-radio></ion-radio>  
 </ion-item>  
 <ion-item>  
 <ion-label>跑步</ion-label>  
 <ion-radio></ion-radio>  
 </ion-item>  
</ion-list>



## 5. 滑块控件

Range控件允许用户通过移动滑块或轨道上的滑块按钮来选择值。滑块两端的值可以通过‘range-left’、‘rang-right’来指定位置。滑块控件上可以通过‘min’和‘max’来指定滑块能滑动的最大和最小值。‘step’属性表示每次滑动增加的步骤值，‘snap’表示每次滑动可以设置为根据步骤属性值自动将旋钮移动到最近的刻度标记。‘dualKnobs’属性用来开启双滑块，使用这个属性时，需要与ngModel一起使用才会有效果，ngModel定义格式为：structure: any = { lower: 33, upper: 60 }; 必须包含lower和upper两个参数。

<ion-range *min=*"20" *max=*"40" *[(ngModel)]=*"sunny">  
 <ion-icon *range-left small name=*"sunny"></ion-icon>  
 <ion-icon *range-right name=*"sunny"></ion-icon>  
</ion-range>  
  
<ion-range *step=*"20" *snaps=*"true">  
 <ion-label *range-left*>0</ion-label>  
 <ion-label *range-right*>100</ion-label>  
</ion-range>  
  
<ion-range *dualKnobs=*"true" *[(ngModel)]=*"structure">  
 <ion-label *range-left*>0</ion-label>  
 <ion-label *range-right*>400</ion-label>  
</ion-range>



## 6. 选择器

ionic 的select 用法与html中的select相同，但展现是采用弹出框，包括弹窗、上拉菜单等形式。可以通过<ion-select>标签中的‘interface’属性去指定弹窗的样式，它的值包括‘action-sheet’、‘popover’、‘alert’。默认为‘alert’弹窗。

<ion-item>  
 <ion-label>省</ion-label>  
 <ion-select *cancelText=*"取消" *okText=*"确定">  
 <ion-option>广东省</ion-option>  
 <ion-option>湖南省</ion-option>  
 <ion-option>江苏省</ion-option>  
 </ion-select>  
</ion-item>



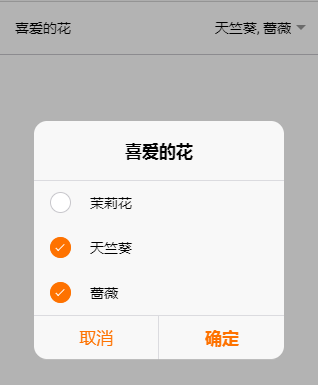
<ion-item>  
 <ion-label>市</ion-label>  
 <ion-select *cancelText=*"取消" *okText=*"确定" *interface=*"popover">  
 <ion-option>长沙市</ion-option>  
 <ion-option>岳阳市</ion-option>  
 <ion-option>湘潭市</ion-option>  
 </ion-select>  
</ion-item>



<ion-item>  
 <ion-label>区</ion-label>  
 <ion-select *cancelText=*"取消" *okText=*"确定" *interface=*"action-sheet">  
 <ion-option>岳麓区</ion-option>  
 <ion-option>开福区</ion-option>  
 <ion-option>天心区</ion-option>  
 </ion-select>  
</ion-item>



Select控件还可以多选，在控件上添加‘multipe=true’属性即可。但这种多选模式仅限于‘alert’模式的弹窗，选中的内容会以逗号隔开显示。



## 7. 日期时间选择

ionic2的日期时间组件样式类似ios上原生的日期时间选择器，从底部滑出，显示可滚动的列，这些列可用于单独选择年份、月份、日、小时和分钟值，标签为<ion-datetime>。

<ion-item>  
 <ion-label>日期选择</ion-label>  
 <ion-datetime *doneText=*"确定" *cancelText=*"取消"

*displayFormat=*"YYYY-MM-DD" *[(ngModel)]=*"myDate">

</ion-datetime>  
</ion-item>  
<ion-item>  
 <ion-label>日期时间选择</ion-label>  
 <ion-datetime *displayFormat=*"YYYY-MM-DD HH:mm:ss"></ion-datetime>  
</ion-item>  
<ion-item>  
 <ion-label>时间选择</ion-label>  
 <ion-datetime *displayFormat=*"HH:mm:ss"></ion-datetime>  
</ion-item>  
<ion-item>  
 <ion-label>日期选择最小最大限制</ion-label>  
 <ion-datetime *displayFormat=*"YYYY-MM-DD" *min=*"2017-12-30" *max=*"2018-07-01">

</ion-datetime>  
</ion-item>



日期选择器默认显示的是英文，如果需要修改成中文，或者修改日期显示的语言格式等，可在main.ts中配置。也可在当前的<ion-datetime>标签下配置。具体配置方式与内容参考官网API。

### 常用属性说明

**displayFormat**：指定在ion-datetime组件中如何将datetime的值输出为格式化文本显示在界面上。常用的Y表示年，M为月份，D为日，h为1-12小时数值，H为24小时数值，m为分，s为秒，还有写其余的可参考官网地址：<https://ionicframework.com/docs/api/components/datetime/DateTime/> 。

格式字符之间可以用任何字符来隔开，如“YYYY/MM/DD”、“YYYY-MM-DD”。

**pickerFormat**：指定在ion-datetime组件中显示哪些列以及显示的顺序以供选择，并且提供列显示的格式，如“MM DD”。如果这个值没有指定，会默认使用displayFormat的格式。

**min**：日期选择的最小值，可只指定最小的年份，但不能只指定月份等，最好与displayFormat格式一致。默认为当前年份的前100年

**max**：日期选择的最大值。默认为当年最后一天。

**doneText**：确定按钮的文字内容，默认为‘Done’。

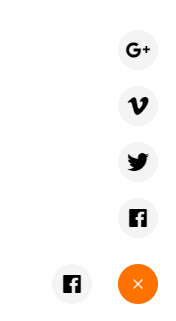
**cancelText**：取消按钮的文字内容，默认为‘Cancel’。

# 七、浮动按钮组

浮动按钮组形状为圆形，当按下时，它可能包含更多相关的动作。而更多的按钮动作会围绕浮动按钮展开。浮动按钮组的内容都必须写在<ion-fab>标签中，而其中的子按钮则包含在<ion-fab-list>标签中。

<ion-fab>标签采用的是绝对定位，可通过left、right、top、bottom、middle、center、edge来进行定位。浮动的主按钮button上必须增加ion-fab属性，否则会使按钮组失效。<ion-fab-list>可通过slide属性来指定展示的方向，可设置的值包括 left、right、top、bottom。

<ion-fab *bottom right*>  
 <button *ion-fab mini*><ion-icon *name=*"add"></ion-icon></button>  
 <ion-fab-list *side=*"left">  
 <button *ion-fab*><ion-icon *name=*"logo-facebook"></ion-icon></button>  
 </ion-fab-list>  
 <ion-fab-list *side=*"top">  
 <button *ion-fab*><ion-icon *name=*"logo-facebook"></ion-icon></button>  
 <button *ion-fab*><ion-icon *name=*"logo-twitter"></ion-icon></button>  
 <button *ion-fab*><ion-icon *name=*"logo-vimeo"></ion-icon></button>  
 <button *ion-fab*><ion-icon *name=*"logo-googleplus"></ion-icon></button>  
 </ion-fab-list>  
</ion-fab>



# 八、手势Gestures

一些基础的手势事件可以直接在html中进行绑定，事件列表如下：

tap：快速点击事件

press：长按事件，按住元素300ms左右触发该事件

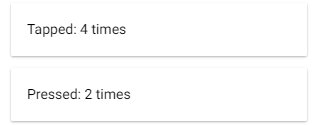
pan：滑动事件，当手指在元素上滑动时触发，一次滑动会触发多次

swipe：滑动事件，当手指在元素上左右滑动时触发，一次滑动触发一次。可在幻灯片切换等场景使用

rotate：旋转事件

pinch：捏合手势

<ion-card *(tap)=*"tapEvent($event)">  
 <ion-item>  
 Tapped: {{tap}} times  
 </ion-item>  
</ion-card>  
  
<ion-card *(press)=*"pressEvent($event)">  
 <ion-item>  
 Pressed: {{press}} times  
 </ion-item>  
</ion-card>



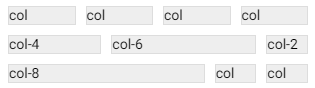
# 网格Grid

网格系统是基于css的flex布局来实现，网格中由三部分组成，分别是网格、行和列。网格系统会将一行分成12列，每一列ion-col可以通过设置col-<width>属性来设置宽度。

网格需要注意的地方：

1. 网格是行和列的容器；
2. 列是行的直接子元素；
3. 没有指定宽度的网格列将自动具有相同的宽度。例如，col-sm的4个实例列将自动设置为25%的宽度；

<ion-grid>  
 <ion-row>  
 <ion-col><div>col</div></ion-col>  
 <ion-col><div>col</div></ion-col>  
 <ion-col><div>col</div></ion-col>  
 <ion-col><div>col</div></ion-col>  
 </ion-row>  
 <ion-row>  
 <ion-col *col-4*><div>col-4</div></ion-col>  
 <ion-col *col-6*><div>col-6</div></ion-col>  
 <ion-col *col-2*><div>col-2</div></ion-col>  
 </ion-row>  
 <ion-row>  
 <ion-col *col-8*><div>col-8</div></ion-col>  
 <ion-col><div>col</div></ion-col>  
 <ion-col><div>col</div></ion-col>  
 </ion-row>  
</ion-grid>



# Icon图标

图标使用标签为<ion-icon>，标签中的name属性表示图标的名称，还可通过isActive来表示是否激活该图标，取值为true或false，激活状态的图标一般是饱满和厚重的，目前只在ios上测试有效果。也可分平台填入要展示的图标，如下：

<ion-icon *name=*"heart" *isActive=*"false"></ion-icon>

<ion-icon *ios=*"logo-apple" *md=*"logo-android"></ion-icon>



<ion-icon *name=*"heart" *isActive=*"true"></ion-icon>

<ion-icon *ios=*"logo-apple" *md=*"logo-android"></ion-icon>



# 十一、列表

列表用于显示信息行，如联系人列表、播放列表或菜单。

列表项可以是任何的HTML元素。容器元素需要list类，每个列表项需要使用ion-item属性。Item列表行默认左侧padding为16px，且内容不换行。

## 基础列表

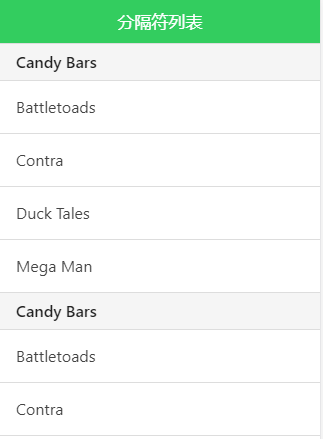
默认的列表存在分割线，如果列表项为button元素，则默认在右侧添加右向箭头图标；其他元素作为列表项，则需要根据需求手动添加箭头图标。

<ion-list>  
 <button *ion-item*>基础列表一</button>  
 <button *ion-item*>基础列表二</button>  
 <div *ion-item*>基础列表三</div>  
 <div *ion-item*>  
 基础列表四  
 <ion-icon *item-end name=*"arrow-forward"></ion-icon>  
 </div>  
</ion-list>



## 2. 列表分隔符 item-divider

<ul *class=*"list">  
 <li *class=*"item item-divider">Candy Bars</li>  
 <li *class=*"item">Battletoads</li>  
 </ul>



## 图标列表

列表项的左侧或右侧都可以指定图标，使用 item-icon-left图标在左侧，item-icon-right 设置图标在右侧。如果你需要在两边都有图标，则两个类都添加上即可。下述例子中省略外层“list”。

### 左侧图标 item-icon-left

<div *class=*"item item-icon-left">  
 <i *class=*"icon ion-email"></i>  
 Email  
 </div>



### 右侧图标 item-icon-right

<div *class=*"item item-icon-right">  
 Call  
 <i *class=*"icon ion-ios-telephone-outline"></i>  
 </div>



### 左右两侧均有图标 item-icon-left item-icon-right

<div *class=*"item item-icon-left item-icon-right">  
 <i *class=*"icon ion-email"></i>  
 Call  
 <i *class=*"icon ion-ios-telephone-outline"></i>  
 </div>



### 图标+注释 item-note

<div *class=*"item item-icon-left item-icon-right">  
 <i *class=*"icon ion-email"></i>  
 Call  
 <span *class=*"item-note">  
 Grammy  
 </span>  
 <i *class=*"icon ion-ios-telephone-outline"></i>  
</div>



### 图标+标记 badge

<div *class=*"item item-icon-left item-icon-right">  
 <i *class=*"icon ion-email"></i>  
 Call  
 <span *class=*"badge badge-assertive">60</span>  
 </div>



## 按钮列表 item-button-left / item-button-right

按钮的位置摆放与图标使用一致，不再赘述。

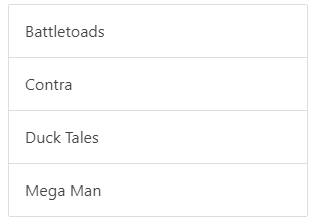
<div *class=*"item item-button-right">  
 Call  
 <button *class=*"button button-balanced">Btn</button>  
 </div>



## 清单列表 list-inset

在容器当中内嵌列表，列表不会显示完整的宽度。 内嵌列表的样式为：list list-inset，与常规列表区别是，它设置了外边距（marign）, 类似于选项卡。 内嵌列表是没有阴影效果的，滚动时效果会更好。

<ul *class=*"list list-inset">  
 <li *class=*"item">Battletoads</li>  
 <li *class=*"item">Contra</li>  
 <li *class=*"item">Duck Tales</li>  
 <li *class=*"item">Mega Man</li>  
 </ul>



## 头像列表 item-avatar

<ul *class=*"list">  
 <li *class=*"item item-avatar">  
 <img *src=*"img/c.png">  
 <h2>Venkman</h2>  
 <p>Back off, man. I'm a scientist.</p>  
 </li>  
 <li *class=*"item item-avatar item-icon-right">  
 <img *src=*"img/c.png">  
 <h2>Venkman</h2>  
 <p>Back off, man. I'm a scientist.</p>  
 <i *class=*"icon ion-ios-arrow-right"></i>  
 </li>  
</ul>



## 7. 缩略图列表 item-thumbnail-left / item-thumbnail-right

item-thumbnail-left 类用于添加左侧对齐的缩略图， item-thumbnail-right 类用于添加右侧对齐的缩略图。

<ul *class=*"list">  
 <li *class=*"item item-thumbnail-left">  
 <img *src=*"img/c.png">  
 <h2>Venkman</h2>  
 <p>Back off, man. I'm a scientist.</p>  
 </li>  
 <li *class=*"item item-thumbnail-left item-icon-right">  
 <img *src=*"img/c.png">  
 <h2>Venkman</h2>  
 <p>Back off, man. I'm a scientist.</p>  
 <i *class=*"icon ion-ios-arrow-right"></i>  
 </li>  
</ul>



# 栅格

ionic 的网格(Grid)采用了弹性盒子模型(Flexible Box Model)，而且在移动端，基本上的手机都支持flexbox。row 样式指定行，col 样式指定列。在row简单添加columns,columns会被平分row的空间. 如果有三列,就添加三列, 有五列就添加五列。没有列数的限制,也没有列大小的限制，还可以在每个列中设置垂直居中。

## 1. 同等大小网格

在带有row的样式的元素里如果包含了col的样式，col就会设置为同等大小。

<div *class=*"row">  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
</div>



## 2. 指定列宽

使用栅格系统的好处在于只需要指明需要的列占位百分比即可，其他的列就会被平均分配使用剩下的空间。

下面列出了指定列宽的一些百分比的样式名:

.col-10 10%

.col-20 20%

.col-25 25%

.col-33 33.3333%

.col-50 50%

.col-67 66.6666%

.col-75 75%

.col-80 80%

.col-90 90%

<div *class=*"row">  
 <div *class=*"col col-50">.col.col-50</div>  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
</div>



## 有偏移量的网格

列可以设置左侧偏移量，下面列出了指定列宽的一些百分比的样式名:

.col-offset-10 10%

.col-offset-20 20%

.col-offset-25 25%

.col-offset-33 33.3333%

.col-offset-50 50%

.col-offset-67 66.6666%

.col-offset-75 75%

.col-offset-80 80%

.col-offset-90 90%

实例如下：

<div *class=*"row">  
 <div *class=*"col col-50 col-offset-20">.col-50 .col-offset-20</div>  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
</div>



## 纵向对齐网格

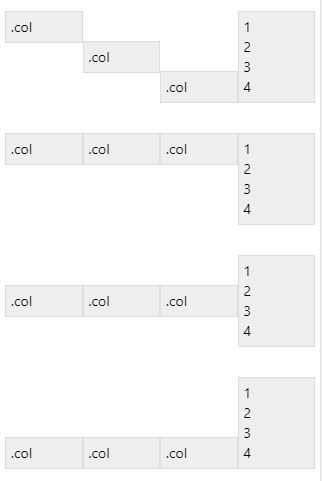
纵向对齐包含顶部，中间部分，底部，可以应用到每一行的列，或者指定的某列。其中 col-top、col-center、col-bottom用来设置指定的列，row-top、row-center、row-bottom用来设置行的内容，实例如下：

<div *class=*"row">  
 <div *class=*"col col-top">.col</div>  
 <div *class=*"col col-center">.col</div>  
 <div *class=*"col col-bottom">.col</div>  
 <div *class=*"col">1<br>2<br>3<br>4</div>  
</div>

<div *class=*"row row-top">  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
 <div *class=*"col">1<br>2<br>3<br>4</div>  
</div>

<div *class=*"row row-center">  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
 <div *class=*"col">1<br>2<br>3<br>4</div>  
</div>

<div *class=*"row row-bottom">  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
 <div *class=*"col">1<br>2<br>3<br>4</div>  
</div>



## 响应式网格

响应式类：

.responsive-sm 小于手机横屏

.responsive-md 小于平板竖屏

.responsive-lg 小于平板横屏

<div *class=*"row responsive-sm">  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
 <div *class=*"col">.col</div>  
</div>



# 其他实用样式

|  |  |
| --- | --- |
| 样式名 | 说明 |
| padding | 在四周添加10px内边距 |
| padding-vertical | 添加10px内边距到顶和底 |
| padding-horizontal | 添加10px内边距到左和右 |
| padding-top | 添加10px内边距到顶 |
| padding-right | 添加10px内边距到右 |
| padding-bottom | 添加10px内边距到底 |
| padding-left | 添加10px内边距到左 |
| text-left | 内容左对齐 |
| text-center | 内容居中 |
| text-right | 内容右对齐 |
| clearfix | 清除浮动 |

# 六、路由跳转

## 根路由修改

跟路由修改在app.ts文件中修改，此处为进入应用默认显示的页面。引入相应的页面组件，将其赋值给rootPage变量即可，该rootPage变量对应 app.html中的[root]值。

import { ButtonPage } from '../pages/component/button/button';  
  
@Component({  
 templateUrl: 'app.html'  
})  
export class MyApp {  
 rootPage:any = ButtonPage;  
}

## 页面路由跳转

路由跳转采用NavController组件，调用push方法，NavController组件需在构造器中生成。

import { TabsPage } from '../../tabs/tabs';  
  
@Component({  
 templateUrl: 'button.html'  
})  
export class ButtonPage {  
  
 constructor(public navCtrl: NavController) {  
  
 }  
  
 goPage() {  
 this.navCtrl.push(TabsPage);  
 }  
}

## 从tab页跳转到其他页面

从tab中跳出到其他页面，与常用的路由跳转有点区别。Tab一般会有自己的子路由，如果按NavController的push方法跳转，会将下一个页面当成tab的子页面。如果需要跳出tab，从根路由处渲染，可用App组件的getRootNav()方法，如下：

import { App } from 'ionic-angular';  
import { ButtonPage } from '../component/button/button';  
  
export class HomePage {  
 constructor(public appCtrl: App) {  
  
 }  
  
 goPage(type) {  
 this.appCtrl.getRootNav().push(ButtonPage);  
 }  
}