**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>



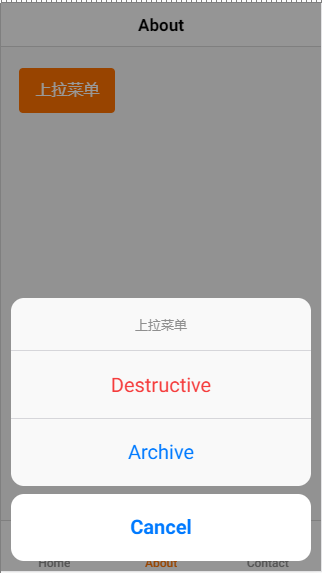
# 上拉菜单

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>

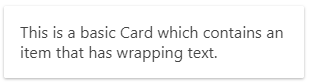


# 五、卡片

卡片(card)默认样式带有box-shadow(阴影)，由于性能的原因，和他类似的元素像 list list-inset 并没有阴影。 如果你有很多的卡片，每个卡片都有很多子元素，建议使用内嵌列表。

## 1. 基础卡片

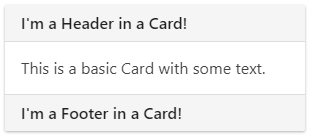
<div *class=*"card">  
 <div *class=*"item item-text-wrap">  
 This is a basic Card which contains an item that has wrapping text.  
 </div>  
</div>



## 带有头部与底部的卡片

通过添加 item-divider 类为卡片添加头部与底部： 添加带 item-divider 类的 card 的元素在内容区域的上方或者下方。

<div *class=*"card">  
 <div *class=*"item item-divider">  
 I'm a Header in a Card!  
 </div>  
 <div *class=*"item item-text-wrap">  
 This is a basic Card with some text.  
 </div>  
 <div *class=*"item item-divider">  
 I'm a Footer in a Card!  
 </div>  
</div>

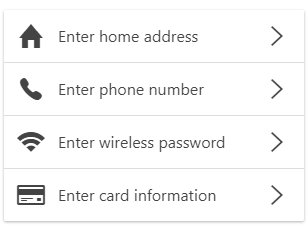


## 列表卡片

使用 list card 类来设置卡片列表。

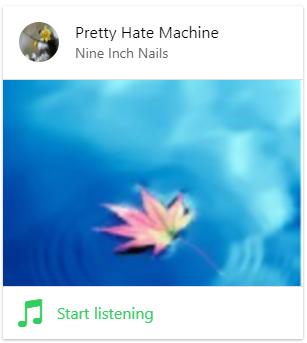
<div *class=*"list card">  
 <div *class=*"item item-icon-left item-icon-right">  
 <i *class=*"icon ion-home"></i>  
 Enter home address  
 <i *class=*"icon ion-ios-arrow-right"></i>  
 </div>

</div>



## 图像卡片

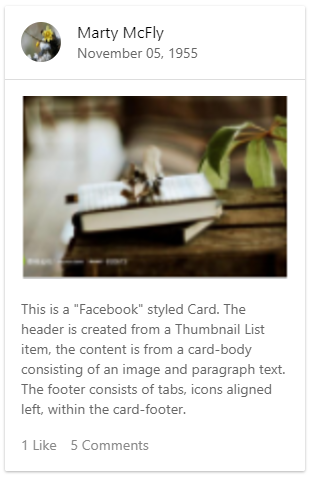
<div *class=*"list card">  
 <div *class=*"item item-avatar">  
 <img *src=*"img/c.png">  
 <h2>Pretty Hate Machine</h2>  
 <p>Nine Inch Nails</p>  
 </div>  
  
 <div *class=*"item item-image">  
 <img *src=*"img/h.png">  
 </div>  
  
 <div *class=*"item item-icon-left balanced">  
 <i *class=*"icon ion-music-note"></i>  
 Start listening  
 </div>  
</div>



## 实例

以下实例中，开始使用了 list card 元素，并使用了 item-avatar , item-body 元素用于展示图片和文本信息。

<div *class=*"list card">  
  
 <div *class=*"item item-avatar">  
 <img *src=*"img/c.png">  
 <h2>Marty McFly</h2>  
 <p>November 05, 1955</p>  
 </div>  
  
 <div *class=*"item item-body">  
 <img *class=*"full-image" *src=*"img/d.png">  
 <p>  
 This is a "Facebook" styled Card.   
 </p>  
 <p>  
 <a *class=*"subdued">1 Like</a>  
 <a *class=*"subdued">5 Comments</a>  
 </p>  
 </div>  
  
</div>



# 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);  
 }  
}

# 按钮

## 基础按钮 button

### 默认按钮样式

默认情况下，按钮显示样式为：display: inline-block。

<button *class=*"button">默认按钮</button>



### 带有颜色的按钮 button-calm

颜色的名称与标题颜色名称一致，将“bar-”改成“button-”即可，例如“button-calm”。

<button *class=*"button button-balanced">

button-balanced

</button>



### 块级按钮 button-block

添加button-block到按钮使按钮以display: block方式显示。一个block的按钮会100%集成他的父元素的宽度。 按钮包含的内容同时也应用了padding, 所以在按钮和显示屏边框之间会有一定的空隙。

<button *class=*"button button-balanced button-block">

块级按钮

</button>



### 全宽按钮 button-full

添加 button-full 到按钮不仅将应用 display: block到按钮，同时还将移除按钮的左右边距，任何边框圆角也可能被应用。当按钮需要撑满整个屏幕的时候，这种样式将非常有用。另外, 按钮的父元素不会有 padding 被应用。

<button *class=*"button button-balanced button-full">

全宽按钮

</button>



### 小按钮 button-small

<button *class=*"button button-balanced button-small">

小按钮

</button>



### 大按钮 button-large

<button *class=*"button button-balanced button-large">

大按钮

</button>



### 干净的按钮 button-clear

添加 button-clear 会移除按钮的边框.

注意: 按钮文字和边框颜色会使用按钮的颜色样式，就是说 button-positive 会使文字的颜色变成蓝色，蓝色背景也将会被无边框取代。

<button *class=*"button button-balanced button-clear">

干净的按钮

</button>



### 图标按钮

图标能够以子元素的形式被放在按钮内部，不过，直接把图标分配给按钮将减少DOM的数量。

<button *class=*"button button-balanced">  
 <i *class=*"icon ion-load-c"></i> Loading...  
 </button>

或者

<button *class=*"button button-balanced icon ion-load-c">  
 Loading...  
 </button>



<button *class=*"button button-balanced icon ion-load-c">

</button>



## 导航按钮

<div *class=*"button-bar">  
 <button *class=*"button button-balanced">First</button>  
 <button *class=*"button button-assertive">Second</button>  
 <button *class=*"button button-calm">Third</button>  
 </div>



# 五、列表

列表可以是基本文字、按钮，开关，图标和缩略图等。

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item类。Item 类默认padding为16px，字体大小为16px，且内容不换行。

ionList和ionItem可以很容易的支持各种交互方式，比如，滑动编辑，拖动排序，以及删除项。

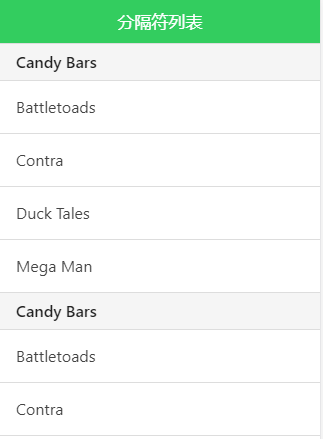
## 1. 基础列表 item

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



## 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。

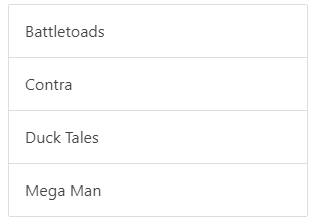
<div *class=*"item item-button-right">  
 Call  
 <button *class=*"button button-balanced">Btn</button>  
 </div>



## 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>



# 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. 基础表单

### 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。

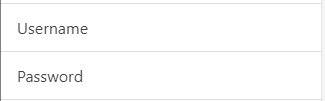
<label *class=*"item-input">  
 <input *type=*"text" *placeholder=*"First Name">  
</label>

IMG_256

### 带有标签

list 类同样可以用于 input 元素。item-input 和 item 类指定了文本框及其标签，使用 input-label 将标签放置于输入框 input 的左侧。

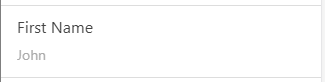
<div *class=*"list">  
 <label *class=*"item item-input">  
 <span *class=*"input-label">Username</span>  
 <input *type=*"text">  
 </label>  
</div>



### 堆叠标签

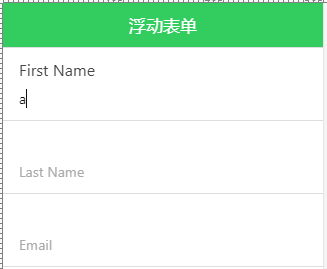
堆叠标签通常位于输入框的头部。每个选项使用 item-stacked-label 类指定。 每个输入框需要指定 input-label。

<div *class=*"list">  
 <label *class=*"item item-input item-stacked-label">  
 <span *class=*"input-label">First Name</span>  
 <input *type=*"text" *placeholder=*"John">  
 </label>  
</div>



## 2. 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，每个选项需要指定 item-floating-label 类，输入标签需要指定 input-label。

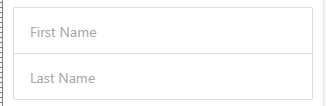


## 3. 列表表单

### 内嵌表单

默认情况下每个输入域宽度都是100%，但我们可以使用 list list-inset 或 card 类设置表单的内边距(padding)，card 类带有阴影。 另外, 如果列表的父元素有 padding 那么当前元素同样会有一个内嵌的表现。

<div *class=*"list list-inset">  
 <label *class=*"item item-input">  
 <input *type=*"text" *placeholder=*"First Name">  
 </label>  
 <label *class=*"item item-input">  
 <input *type=*"text" *placeholder=*"Last Name">  
 </label>  
</div>



### 内嵌输入域

使用 item-input-inset 样式可以在item中内嵌，例如一个按钮。

<div *class=*"list">  
 <div *class=*"item item-input-inset">  
 <label *class=*"item-input-wrapper">  
 <input *type=*"text" *placeholder=*"Email">  
 </label>  
 <button *class=*"button button-small button-balanced">  
 Submit  
 </button>  
 </div>  
</div>



## 图标表单

item-input 输入框可以很简单的添加图标，图标可以在 <input> 前添加。 默认的图标会使用label文字的颜色, 但你也可以添加placeholder-icon来改变placeholder的颜色。

<div *class=*"list list-inset">  
 <label *class=*"item item-input">  
 <i *class=*"icon ion-search placeholder-icon"></i>  
 <input *type=*"text" *placeholder=*"Search">  
 </label>  
</div>



## 5. 头部表单

<div *class=*"bar bar-header bar-balanced item-input-inset">  
 <label *class=*"item-input-wrapper">  
 <i *class=*"icon ion-ios-search placeholder-icon"></i>  
 <input *type=*"search" *placeholder=*"Search">  
 </label>  
 <button *class=*"button button-clear">取消</button>  
</div>



## 6. 切换开关

切换开关类似与 HTML 的 checkbox 标签，但它更易于在移动设备上使用。切换开关可以使用 toggle-assertive 来指定颜色。与item搭配使用时，每个选项的 item 类后需要添加 item-toggle类。

<ul *class=*"list">  
 <li *class=*"item item-toggle">  
 HTML5

*<!--开关代码-->*

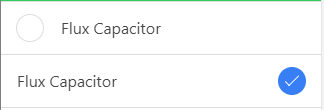
<label *class=*"toggle toggle-balanced">  
 <input *type=*"checkbox">  
 <div *class=*"track">  
 <div *class=*"handle"></div>  
 </div>  
 </label>  
 </li>  
</ul>



## 7. 复选框

每个选项的item类后需要添加 item-checkbox 类。复选框可以使用 checkbox-assertive 来指定颜色。默认复选框在左，也可通过在 item-checkbox 类后添加 item-checkbox-right 将复选框放置右侧。

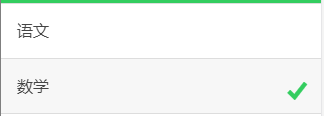
<div *class=*"list">  
 <div *class=*"item item-checkbox">  
 <label *class=*"checkbox">  
 <input *type=*"checkbox">  
 </label>  
 Flux Capacitor  
 </div>  
 <div *class=*"item item-checkbox item-checkbox-right">  
 <label *class=*"checkbox">  
 <input *type=*"checkbox">  
 </label>  
 Flux Capacitor  
 </div>  
</div>



## 8. 单选框

ionic 单选按钮与标准 type="radio" 的 input元素类似。 每个 item-radio 中的 type="radio" 的 input 元素的 name 属性都相同。radio-content 类用来包裹要显示的内容，包括标签与图标等。radio-icon 类用于是否显示图标。 ionic 在单选项中使用了 <label> 元素，使其更易点击。

<div *class=*"list">  
 <label *class=*"item item-radio">  
 <input *type=*"radio" *name=*"group" *checked=*"checked">  
 <div *class=*"radio-content">  
 <div *class=*"item-content">  
 语文  
 </div>  
 <i *class=*"radio-icon ion-checkmark balanced"></i>  
 </div>  
 </label>  
 <label *class=*"item item-radio">  
 <input *type=*"radio" *name=*"group">  
 <div *class=*"radio-content">  
 <div *class=*"item-content">  
 数学  
 </div>  
 <i *class=*"radio-icon ion-checkmark balanced"></i>  
 </div>  
 </label>  
</div>



## 9. 滑块控件

每个选项的item类后需要添加 range 类。

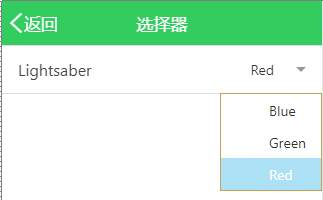
<div *class=*"item range">  
 <i *class=*"icon ion-volume-low balanced"></i>  
 <input *type=*"range" *name=*"volume">  
 <i *class=*"icon ion-volume-high balanced"></i>  
</div>



## 10. 选择器

ionic select 修改了原生样式，但是弹出的可选选项样式是浏览器默认的，每个平台上的可选项样式都是不一样的，在PC电脑的浏览器上，是传统的下拉界面，Android 上是弹出单选按钮选项，iOS 有个滚轮操作界面。

<label *class=*"item item-input item-select">  
 <div *class=*"input-label">  
 Lightsaber  
 </div>  
 <select>  
 <option>Blue</option>  
 <option *selected*>Green</option>  
 <option>Red</option>  
 </select>  
</label>



# 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

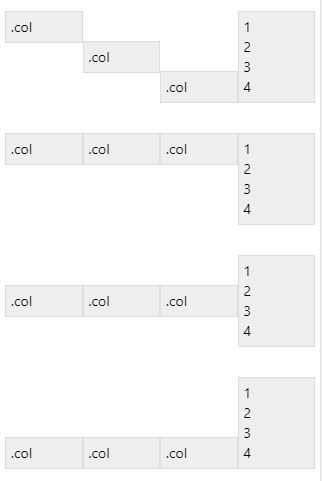
纵向对齐包含顶部，中间部分，底部，可以应用到每一行的列，或者指定的某列。其中 col-top、col-center、col-bottom用来设置指定的列，row-top、row-center、row-bottom用来设置行的内容，实例如下：

<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 |