

accordion（折叠面板）

折叠面板从二级列表中演化而来，dom结构和二级列表类似，如下：

```
<ul class="mui-table-view">
  <li class="mui-table-view-cell mui-collapse mui-active">
    <a class="mui-navigate-right" href="#">面板1</a>
    <div class="mui-collapse-content">
      <p>面板1子内容</p>
    </div>
  </li>
  <li class="mui-table-view-cell mui-collapse">
    <a class="mui-navigate-right" href="#">面板</a>
    <div class="mui-collapse-content">
      <p>面板2子内容</p>
    </div>
  </li>
  <li class="mui-table-view-cell mui-collapse">
    <a class="mui-navigate-right" href="#">面板3</a>
    <div class="mui-collapse-content">
      <p>面板3子内容</p>
    </div>
  </li>
</ul>
```

可以在折叠面板中放置任何内容；折叠面板默认收缩，若希望某个面板默认展开，只需要在包含 `.mui-collapse` 类的 `li` 节点上，增加 `.mui-active` 类即可；mui官网中的方法说明，使用的就是折叠面板控件。

扩展阅读

代码块激活字符: `maccordion`

actionsheet（操作表）

actionsheet一般从底部弹出，显示一系列可供用户选择的操作按钮；actionsheet是从popover控件基础上演变而来，实际上就是一个固定从底部弹出的popover，故DOM结构和popove类似，只是需要在含 `.mui-popover` 类的节点上增加 `.mui-popover-bottom`、`.mui-popover-action` 类；

```
<div id="sheet1" class="mui-popover mui-popover-bottom mui-popover-action ">
  <!-- 可选择菜单 -->
  <ul class="mui-table-view">
    <li class="mui-table-view-cell">
      <a href="#">菜单1</a>
    </li>
    <li class="mui-table-view-cell">
      <a href="#">菜单2</a>
    </li>
  </ul>
  <!-- 取消菜单 -->
  <ul class="mui-table-view">
    <li class="mui-table-view-cell">
      <a href="#sheet1"><b>取消</b></a>
    </li>
  </ul>
</div>
```

和popover一样，推荐使用锚点方式显示、隐藏actionsheet；若要使用js代码动态显示、隐藏actionsheet，同样在popover插件的构造方法中传入“toggle”参数即可，如下：

```
//传入toggle参数，用户无需关心当前是显示还是隐藏状态，mui会自动识别处理；
mui('#sheet1').popover('toggle');
```

扩展阅读

问答社区话题讨论：[actionsheet](http://ask.dcloud.net.cn/topic/actionsheet) (<http://ask.dcloud.net.cn/topic/actionsheet>)

代码块激活字符：`mactionsheet`

badge（数字角标）

数字角标一般和其它控件（列表、9宫格、选项卡等）配合使用，用于进行数量提示。角标的核心类是 `.mui-badge`，默认为实心灰色背景；同时，mui还内置了蓝色(blue)、绿色(green)、黄色(yellow)、红色(red)、紫色(purple)五种色系的数字角标，如下：

```
<span class="mui-badge">1</span>
<span class="mui-badge mui-badge-primary">12</span>
<span class="mui-badge mui-badge-success">123</span>
<span class="mui-badge mui-badge-warning">3</span>
<span class="mui-badge mui-badge-danger">45</span>
<span class="mui-badge mui-badge-purple">456</span>
```

若无需底色，则增加 `.mui-badge-inverted` 类即可，如下：

```
<span class="mui-badge mui-badge-inverted">1</span>
<span class="mui-badge mui-badge-primary mui-badge-inverted">2</span>
<span class="mui-badge mui-badge-success mui-badge-inverted">3</span>
<span class="mui-badge mui-badge-warning mui-badge-inverted">4</span>
<span class="mui-badge mui-badge-danger mui-badge-inverted">5</span>
<span class="mui-badge mui-badge-royal mui-badge-inverted">6</span>
```

扩展阅读

代码块激活字符：`mbadge`

button（按钮）

mui默认按钮为灰色，另外还提供了蓝色(blue)、绿色(green)、黄色(yellow)、红色(red)、紫色(purple)五种色系的按钮，五种色系对应五种场景，分别为primary、success、warning、danger、royal；使用 `.mui-btn` 类即可生成一个默认按钮，继续添加 `.mui-btn-颜色值` 或 `.mui-btn-场景` 可生成对应色系的按钮，例如：通过 `.mui-btn-blue` 或 `.mui-btn-primary` 均可生成蓝色按钮；

普通按钮

在button节点上增加 `.mui-btn` 类，即可生成默认按钮；若需要其他颜色的按钮，则继续增加对应class即可，比如 `.mui-btn-blue` 即可变成蓝色按钮

```
<button type="button" class="mui-btn">默认</button>
<button type="button" class="mui-btn mui-btn-primary">蓝色</button>
<button type="button" class="mui-btn mui-btn-success">绿色</button>
<button type="button" class="mui-btn mui-btn-warning">黄色</button>
<button type="button" class="mui-btn mui-btn-danger">红色</button>
<button type="button" class="mui-btn mui-btn-royal">紫色</button>
```

默认按钮有底色，运行效果如下：



若希望无底色、有边框的按钮，仅需增加 `.mui-btn-outlined` 类即可，代码如下：

```
<button type="button" class="mui-btn mui-btn-outlined">默认</button>
<button type="button" class="mui-btn mui-btn-primary mui-btn-outlined">蓝色</button>
<button type="button" class="mui-btn mui-btn-success mui-btn-outlined">绿色</button>
<button type="button" class="mui-btn mui-btn-warning mui-btn-outlined">黄色</button>
<button type="button" class="mui-btn mui-btn-danger mui-btn-outlined">红色</button>
<button type="button" class="mui-btn mui-btn-royal mui-btn-outlined">紫色</button>
```

运行效果如下：



扩展阅读

代码块激活字符: `mbutton`

checkbox（复选框）

checkbox常用于多选的情况，比如批量删除、添加群聊等；

DOM结构

```
<div class="mui-input-row mui-checkbox">
  <label>checkbox示例</label>
  <input name="checkboxbox1" value="Item 1" type="checkbox" checked>
</div>
```

默认checkbox在右侧显示，若希望在左侧显示，只需增加 `.mui-left` 类即可，如下：

```
<div class="mui-input-row mui-checkbox mui-left">
  <label>checkbox左侧显示示例</label>
  <input name="checkboxbox1" value="Item 1" type="checkbox">
</div>
```

若要禁用checkbox，只需在checkbox上增加disabled属性即可；

扩展阅读

代码块激活字符: `mcheckbox`

dialog（对话框）

*此dialog api支持mui v2.7+

创建并显示对话框，弹出的对话框为非阻塞模式，用户点击对话框上的按钮后关闭(h5模式的对话框也可通过closepopup关闭)，并通过callback函数返回用户点击按钮的索引值或输入框中的值。

mui会根据 ua 判断,弹出原生对话框还是h5绘制的对话框,在基座中默认会弹出原生对话框,可以配置type属性,使得弹出h5模式对话框

两者区别:1.原生对话框可以跨webview,2.h5对话框样式统一而且可以修改对话框属性或样式

mui v3.0 版本的dialog控件支持换行（\n）显示

.alert(message, title, title, btnValue, callback, [, type])

message
Type: String ()
提示对话框上显示的内容

title

Type: String ()

提示对话框上显示的标题

btnValue

Type: String ()

提示对话框上按钮显示的内容

callback

Type: Function ()

提示对话框上关闭后的回调函数

type

Value: 'div' ()

是否使用h5绘制的对话框

.confirm(message, title, title, btnValue, callback, [, type])**message**

Type: String ()

提示对话框上显示的内容

title

Type: String ()

提示对话框上显示的标题

btnValue

Type: Array ()

提示对话框上按钮显示的内容

callback

Type: Function ()

提示对话框上关闭后的回调函数

type

Value: 'div' ()

是否使用h5绘制的对话框

.prompt(message, placeholder, title, title, btnValue, callback, [, type])**message**

Type: String ()

提示对话框上显示的内容

placeholder

Type: String ()

编辑框显示的提示文字

title

Type: String ()

提示对话框上显示的标题

btnValue

Type: Array ()

提示对话框上按钮显示的内容

callback

Type: Function ()

提示对话框上关闭后的回调函数

type

Value: 'div' ()

是否使用h5绘制的对话框

如果有定制对话框样式的需求([只能修改h5模式对话框](#))可以在mui.css中修改 .mui-popup 类下的样式

如果需要修改 DOM 结构可以按照以下方式处理.

```
//修改弹出框默认input类型为password
mui.prompt('text','deftext','title',['true','false'],null,'div')
document.querySelector('.mui-popup-input input').type='password'
```

.toast(message)

toast
Type: String ()
自动消失提示框

.closePopup()(只能关闭h5模式的对话框)

关闭最后一次弹出的对话框

.closePopups()(只能关闭h5模式的对话框)

关闭所有对话框

扩展阅读

代码块激活字符: malert mconfirm mprompt mtoast mclosepopup mclosepopups

图片轮播

图片轮播继承自slide插件，因此其DOM结构、事件均和slide插件相同；

DOM结构

默认不支持循环播放，DOM结构如下：

```
<div class="mui-slider">
  <div class="mui-slider-group">
    <div class="mui-slider-item"><a href="#"></a></div>
    <div class="mui-slider-item"><a href="#"></a></div>
    <div class="mui-slider-item"><a href="#"></a></div>
    <div class="mui-slider-item"><a href="#"></a></div>
  </div>
</div>
```

假设当前图片轮播中有1、2、3、4四张图片，从第1张图片起，依次向左滑动切换图片，当切换到第4张图片时，继续向左滑动，接下来会有两种效果：

- 支持循环：左滑，直接切换到第1张图片；
- 不支持循环：左滑，无反应，继续显示第4张图片，用户若要显示第1张图片，必须连续向右滑动切换到第1张图片；

当显示第1张图片时，继续右滑是否显示第4张图片，是同样问题；这个问题的实现需要通过 .mui-slider-loop 类及DOM节点来控制；

若要支持循环，则需要在 .mui-slider-group 节点上增加 .mui-slider-loop 类，同时需要重复增加2张图片，图片顺序变为：4、1、2、3、4、1，代码示例如下：

```
<div class="mui-slider">
  <div class="mui-slider-group mui-slider-loop">
    <!--支持循环，需要重复图片节点-->
    <div class="mui-slider-item mui-slider-item-duplicate"><a href="#"></a></div>
    <div class="mui-slider-item"><a href="#"></a></div>
    <div class="mui-slider-item"><a href="#"></a></div>
    <div class="mui-slider-item"><a href="#"></a></div>
    <div class="mui-slider-item"><a href="#"></a></div>
    <!--支持循环，需要重复图片节点-->
    <div class="mui-slider-item mui-slider-item-duplicate"><a href="#"></a></div>
  </div>
</div>
```

JS Method

mui框架内置了图片轮播插件，通过该插件封装的JS API，用户可以设定是否自动轮播及轮播周期，如下为代码示例：

```
//获得slider插件对象
var gallery = mui('.mui-slider');
gallery.slider({
  interval:5000//自动轮播周期，若为0则不自动播放，默认为0；
});
```

因此若希望图片轮播不要自动播放，而是用户手动滑动才切换，只需要通过如上方法，将interval参数设为0即可。

若要跳转到第x张图片，则可以使用图片轮播插件的gotoItem方法，例如：

```
//获得slider插件对象
var gallery = mui('.mui-slider');
gallery.slider().gotoItem(index);//跳转到第index张图片，index从0开始；
```

注意：mui框架会默认初始化当前页面的图片轮播组件；若轮播组件内容为js动态生成时（比如通过ajax动态获取的营销信息），则需要在动态DOM生成后，手动调用图片轮播的初始化方法；代码如下：

```
//获得slider插件对象
var gallery = mui('.mui-slider');
gallery.slider({
  interval:5000//自动轮播周期，若为0则不自动播放，默认为0；
});
```

扩展阅读

问答社区话题讨论：图片轮播 (<http://ask.dcloud.net.cn/topic/图片轮播>)

代码块激活字符：`mslider`

icon(图标)

mui默认提供了手机App开发常用的字体图标，如下：



mui-icon-map	mui-icon-compose	mui-icon-trash	mui-icon-upload	mui-icon-download	mui-icon-close
mui-icon-closeempty	mui-icon-redo	mui-icon-undo	mui-icon-refresh	mui-icon-refreshempty	mui-icon-reload
mui-icon-loop	mui-icon-spinner mui-spin	mui-icon-spinner-cycle	mui-icon-star	mui-icon-starhalf	mui-icon-plus
mui-icon-plusempty	mui-icon-minus	mui-icon-checkmarkempty	mui-icon-search	mui-icon-home	mui-icon-navigate
mui-icon-gear	mui-icon-settings	mui-icon-list	mui-icon-bars	mui-icon-paperplane	mui-icon-info
mui-icon-help	mui-icon-locked	mui-icon-more	mui-icon-locked	mui-icon-flag	mui-icon-paperclip
mui-icon-back	mui-icon-forward	mui-icon-arrowup	mui-icon-arrowdown	mui-icon-arrowleft	mui-icon-arrowright
mui-icon-arrowthinup	mui-icon-arrowthindown	mui-icon-arrowthinleft	mui-icon-arrowthinright	mui-icon-pulldown	

使用时，只需要在 `span` 节点上分别增加 `.mui-icon`、`.mui-icon-name` 两个类即可（`name`为图标名称，例如：`weixin`、`weibo`等），如下代码即可显示一个微信图标：

```
<span class="mui-icon mui-icon-weixin"></span>
```

关联阅读

- mui如何增加自定义icon图标 (<http://ask.dcloud.net.cn/article/128>)

扩展阅读

代码块激活字符: `micon`

输入增强

`mui`目前提供的输入增强包括：快速删除和语音输入两项功能。要删除输入框中的内容，使用输入法键盘上的删除按键，只能逐个删除字符，`mui`提供了快速删除能力，只需要在对应`input`控件上添加 `.mui-input-clear` 类，当`input`控件中有内容时，右侧会有一个删除图标，点击会清空当前`input`的内容；另外，为了方便快速输入，`mui`集成了HTML5+的语音输入，只需要在对应`input`控件上添加 `.mui-input-speech` 类，就会在该控件右侧显示一个语音输入的图标，点击会启用科大讯飞语音输入界面。

扩展阅读

代码块激活字符: `minput`

list（列表）

列表是常用的UI控件，mui封装的列表组件比较简单，只需要在 `ul` 节点上添加 `.mui-table-view` 类、在 `li` 节点上添加 `.mui-table-view-cell` 类即可，如下为示例代码

```
<ul class="mui-table-view">
  <li class="mui-table-view-cell">Item 1</li>
  <li class="mui-table-view-cell">Item 2</li>
  <li class="mui-table-view-cell">Item 3</li>
</ul>
```

点击列表，对应列表项显示灰色高亮，若想自定义高亮颜色，只需要重写 `.mui-table-view-cell.mui-active` 即可，如下：

```
/*点击变蓝色高亮*/
.mui-table-view-cell .mui-active{
  background-color: #0062CC;
}
```

若右侧需要增加导航箭头，变成导航链接，则只需在 `li` 节点下增加 `a` 子节点，并为该 `a` 节点增加 `.mui-navigate-right` 类即可，如下：

```
<ul class="mui-table-view">
  <li class="mui-table-view-cell">
    <a class="mui-navigate-right">Item 1</a>
  </li>
  <li class="mui-table-view-cell">
    <a class="mui-navigate-right">Item 2</a>
  </li>
  <li class="mui-table-view-cell">
    <a class="mui-navigate-right">Item 3</a>
  </li>
</ul>
```

mui支持将数字角标、按钮、开关等控件放在列表中；mui默认将数字角标放在列表右侧显示，代码如下：

```
<ul class="mui-table-view">
  <li class="mui-table-view-cell">Item 1
    <span class="mui-badge mui-badge-primary">11</span>
  </li>
  <li class="mui-table-view-cell">Item 2
    <span class="mui-badge mui-badge-success">22</span>
  </li>
  <li class="mui-table-view-cell">Item 3
    <span class="mui-badge">33</span>
  </li>
</ul>
```

扩展阅读

代码块激活字符：`mList`

遮罩蒙版

在popover、侧滑菜单等界面，经常会用到蒙版遮罩；比如popover弹出后，除popover控件外的其它区域都会遮罩一层蒙版，用户点击蒙版不会触发蒙版下方的逻辑，而会关闭popover同时关闭蒙版；再比如侧滑菜单界面，菜单划出后，除侧滑菜单之外的其它区域都会遮罩一层蒙版，用户点击蒙版会关闭侧滑菜单同时关闭蒙版。

遮罩蒙版常用的操作包括：创建、显示、关闭，如下代码：

```
var mask = mui.createMask(callback);//callback为用户点击蒙版时自动执行的回调；
mask.show();//显示遮罩
mask.close();//关闭遮罩
```

注意：关闭遮罩仅会关闭，不会销毁；关闭之后可以再次调用 `mask.show()`；打开遮罩：

mui默认的蒙版遮罩使用 `.mui-backdrop` 类定义（如下代码），若需自定义遮罩效果，只需覆盖定义 `.mui-backdrop` 即可；


```
.mui-backdrop {
  position: fixed;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
  z-index: 998;
  background-color: rgba(0,0,0,.3);
}
```

扩展阅读

代码块激活字符: `mmask`

media list (图文列表)

图文列表继承自列表组件，主要添加了 .mui-media 、 .mui-media-object 、 .mui-media-body 几个类，如下为示例代码

```
<ul class="mui-table-view">
  <li class="mui-table-view-cell mui-media">
    <a href="javascript:;">
      
      <div class="mui-media-body">
        幸福
        <p class='mui-ellipsis'>能和心爱的人一起睡觉，是件幸福的事情；可是，打呼噜怎么办？</p>
      </div>
    </a>
  </li>
  <li class="mui-table-view-cell mui-media">
    <a href="javascript:;">
      
      <div class="mui-media-body">
        木屋
        <p class='mui-ellipsis'>想要这样一间小木屋，夏天挫冰吃瓜，冬天围炉取暖.</p>
      </div>
    </a>
  </li>
  <li class="mui-table-view-cell mui-media">
    <a href="javascript:;">
      
      <div class="mui-media-body">
        CBD
        <p class='mui-ellipsis'>烤炉模式的城，到黄昏，如同打翻的调色盘一般.</p>
      </div>
    </a>
  </li>
</ul>
```

扩展阅读

代码块激活字符: `mList_Media`

numbox(数字输入框)

mui提供了数字输入框控件，可直接输入数字，也可以点击“+”、“-”按钮变换当前数值；默认numbox控件dom结构比较简单，如下：

```
<div class="mui-numbox">
  <!-- "-"按钮，点击可减小当前数值 -->
  <button class="mui-btn mui-numbox-btn-minus" type="button">-</button>
  <input class="mui-numbox-input" type="number" />
  <!-- "+"按钮，点击可增大当前数值 -->
  <button class="mui-btn mui-numbox-btn-plus" type="button">+</button>
</div>
```

可通过 data-* 自定义属性设置数字输入框的参数，如下：

numbox自定义参数

data-numbox-min
输入框允许使用的最小值，默认无限制

data-numbox-max
输入框允许使用的最大值，默认无限制

data-numbox-step
每次点击“+”、“-”按钮变化的步长，默认步长为1

示例：设置取值范围为0~100，每次变化步长为10，则代码如下

```
<div class="mui-numbox" data-numbox-step='10' data-numbox-min='0' data-numbox-max='100'>
  <button class="mui-btn mui-numbox-btn-minus" type="button">-</button>
  <input class="mui-numbox-input" type="number" />
  <button class="mui-btn mui-numbox-btn-plus" type="button">+</button>
</div>
```

扩展阅读

代码块激活字符：`numbox`

侧滑导航

mui提供了两种侧滑导航实现：webview模式和div模式，两种模式各有优劣，适用于不同的场景。

webview模式

主页面和菜单内容在不同的webview中，两个页面根据内容需求分别组织DOM结构，mui对其DOM结构无特殊要求，故其有如下优点：

- 菜单内容是单独的webview，故可被多个页面复用；
- 菜单内容在单独的webview中，菜单区域的滚动不影响主界面，故可使用原生滚动，滚动更为流畅；

另一方面，webview模式也有其缺点：

- 不支持拖动手势（跟手拖动）；
- 主页面、菜单不同webview实现，因此若需交互（如：点击菜单触发主页面内容变化），需使用自定义事件实现跨webview通讯；

div模式

主页面和菜单内容在同一个webview下，嵌套在特定结构的div中，通过div的移动动画模拟菜单移动；故该模式有如下优点：

- 支持拖动手势（跟手拖动）；
- 主页面、菜单在一个页面中，可通过JS轻松实现两者交互（如：点击菜单触发主页面内容变化），没有跨webview通讯的烦恼；

另一方面，div模式也有其缺点：

- 不支持菜单内容在多页面的复用，需每个页面都生成对应的菜单节点；
- 主界面和菜单内容的滚动互不影响，因此会使用div区域滚动，在低端Android手机且滚动内容较多时，可能会稍显卡顿；

div模式支持不同的动画效果，每种动画效果需遵从不同的DOM构造；下面我们以右滑菜单为例（左滑菜单仅需将菜单父节点上的 mui-off-canvas-left 换成 mui-off-canvas-right 即可），说明每种动画对应的DOM结构。

动画1：主界面移动、菜单不动

```
<!-- 侧滑导航根容器 -->
<div class="mui-off-canvas-wrap mui-draggable">
  <!-- 菜单容器 -->
  <aside class="mui-off-canvas-left">
    <div class="mui-scroll-wrapper">
      <div class="mui-scroll">
        <!-- 菜单具体展示内容 -->
        ...
      </div>
    </div>
  </aside>
  <!-- 主页面容器 -->
  <div class="mui-inner-wrap">
    <!-- 主页面标题 -->
    <header class="mui-bar mui-bar-nav">
      <a class="mui-icon mui-action-menu mui-icon-bars mui-pull-left"></a>
      <h1 class="mui-title">标题</h1>
    </header>
    <div class="mui-content mui-scroll-wrapper">
      <div class="mui-scroll">
        <!-- 主界面具体展示内容 -->
        ...
      </div>
    </div>
  </div>
</div>
```

动画2：主界面不动、菜单移动

该种动画要求的DOM结构和动画1的DOM结构基本相同，唯一差别就是需在侧滑导航根容器class上增加一个 mui-slide-in 类

动画3：主界面、菜单同时移动

该种动画要求的DOM结构较特殊，需将菜单容器放在主页面容器之下

```
<!-- 侧滑导航根容器 -->
<div class="mui-off-canvas-wrap mui-draggable">
  <!-- 主页面容器 -->
  <div class="mui-inner-wrap">
    <!-- 菜单容器 -->
    <aside class="mui-off-canvas-left">
      <div class="mui-scroll-wrapper">
        <div class="mui-scroll">
          <!-- 菜单具体展示内容 -->
          ...
        </div>
      </div>
    </aside>
    <!-- 主页面标题 -->
    <header class="mui-bar mui-bar-nav">
      <a class="mui-icon mui-action-menu mui-icon-bars mui-pull-left"></a>
      <h1 class="mui-title">标题</h1>
    </header>
    <!-- 主页面内容容器 -->
    <div class="mui-content mui-scroll-wrapper">
      <div class="mui-scroll">
        <!-- 主界面具体展示内容 -->
        ...
      </div>
    </div>
  </div>
</div>
```

mui支持多种方式显示div模式的侧滑菜单：1、在主界面向右拖动（drag）；2、点击含有 mui-action-menu 类的控件；3、Android手机按menu键；4、通过JS API触发，如下：

```
mui('.mui-off-canvas-wrap').offCanvas('show');
```

同样，mui支持多种方式关闭div模式的侧滑菜单：1、在手机屏幕上任意位置向左拖动（drag）；2、点击主界面内任意位置；3、Android手机按menu键；4、Android手机按back键；5、通过JS API触发，如下：

```
mui('.mui-off-canvas-wrap').offCanvas('close');
```

扩展阅读

问答社区 话题讨论： 侧滑菜单 (<http://ask.dcloud.net.cn/topic/侧滑菜单>)

代码块激活字符：`moffcanvas`

弹出菜单

mui框架内置了弹出菜单插件，弹出菜单显示内容不限，但必须包裹在一个含 `.mui-popover` 类的div中，如下即为一个弹出菜单内容：

```
<div id="popover" class="mui-popover">
  <ul class="mui-table-view">
    <li class="mui-table-view-cell"><a href="#">Item1</a></li>
    <li class="mui-table-view-cell"><a href="#">Item2</a></li>
    <li class="mui-table-view-cell"><a href="#">Item3</a></li>
    <li class="mui-table-view-cell"><a href="#">Item4</a></li>
    <li class="mui-table-view-cell"><a href="#">Item5</a></li>
  </ul>
</div>
```

要显示、隐藏如上菜单，mui推荐使用锚点方式，例如：

```
<a href="#popover" id="openPopover" class="mui-btn mui-btn-primary mui-btn-block">打开弹出菜单</a>
```

点击如上定义的按钮，即可显示弹出菜单，再次点击弹出菜单之外的其他区域，均可关闭弹出菜单；这种使用方式最为简洁。

若希望通过js的方式控制弹出菜单，则通过如下一个方法即可：

```
mui('.bottomPopover').popover(status,[anchor]);
```

status

- 'show'**
显示popover
- 'hide'**
隐藏popover
- 'toggle'**
自动识别处理显示隐藏状态

```
mui('.bottomPopover').popover('toggle');//show hide toggle
```

[anchor]

- anchorElement**
锚点元素

```
//传入toggle参数，用户也无需关心当前是显示还是隐藏状态，mui会自动识别处理；
mui('.mui-popover').popover('toggle',document.getElementById("openPopover"));
```

扩展阅读

问答社区 话题讨论： popover (<http://ask.dcloud.net.cn/topic/popover>)

代码块激活字符：`mpopover`

picker（选择器）

mui框架扩展了pipcker组件,可用于弹出选择器,在各平台上都有统一表现.poppicker和dtpicker是对picker的具体实现

*poppicker组件依赖 mui.picker.js/.css mui.poppicker.js/.css 请务必在mui.js/css后中引用,也可统一引用 压缩版本: mui.picker.min.js

popPicker

适用于弹出单级或多级选择器

快速体验

通过 `new mui.PopPicker()` 初始化popPicker组件

```
var picker = new mui.PopPicker();
```

给picker对象添加数据

setDate() 支持数据格式为: 数组

```
picker.setData([{value:'zz',text:'智子'}]);
```

显示picker

```
picker.show( SelectedItemsCallback )
```

实例

```
var picker = new mui.PopPicker();
picker.setData([{value:'zz',text:'智子'}]);
picker.show(function (selectItems) {
    console.log(selectItems[0].text);//智子
    console.log(selectItems[0].value);//zz
})
```

API详解

new PopPicker({PopPicker.options})

layer

Type: Int ()

picker显示列数

buttons

Type: Array ()

picker显示按钮

如: new mui.PopPicker({button:['cancle','ok']})

setData([data])

参数:data

Type: Array ()

填充数据

如: picker.setData([{value:'zz',text:'智子'}])

data格式

getSelectedItems()

返回值[data]

Type: Array ()
获取选中的项（数组）
如: picker.getSelectedItems()

show(getSelectedItems)

返回值:**[data]**
Type: Array ()
获取选中的项（数组）
如:

```
picker.show(function(getSelectedItems){console.log(getSelectedItems)})
```

* return false; 可以阻止选择框的关闭

hide()

隐藏picker
如: picker.hide()

dispose()

释放组件资源(释放后将不能再操作组件)
如: picker.dispose()
* 通常情况下，不需要释放组件资源，初始化之后，可以一直使用。
* 当内容较多，如不释放组件资源，在某些设备上可能会卡顿。
* 所以每次用完便立即调用 dispose() 进行释放,下次用时再创建新实例。

dtpicker

dtpicker组件适用于弹出日期选择器

快速体验

通过 `new mui.DtPicker()` 初始化DtPicker组件

```
var dtPicker = new mui.DtPicker(options);
```

显示picker

```
dtPicker.show( SelectedItemsCallback )
```

实例

```
var dtPicker = new mui.DtPicker();
dtPicker.show(function (selectItems) {
    console.log(selectItems.y);//{text: "2016",value: 2016}
    console.log(selectItems.m);//{text: "05",value: "05"}
})
```

API详解

new DtPicker({options})

1.参数:type

Type: JSON ()
设置日历初始视图模式
支持的值:

可选值	视图模式
'datetime'	完整日期视图(年月日时分)
'date'	年视图(年月日)
'time'	时间视图(时分)
'month'	月视图(年月)
'hour'	时视图(年月日时)

*暂不支持指定其他视图,
如有特殊需求可在 dtpicker.js 中修改 getSelected() 方法中 selected 对象值

2.参数:customData

Type: JSON ()
设置时间/日期别名
设置格式:

```
"customData":{
  "date":[
    {value:"",text:""}
  ]
}
```

示例:

```
var dtpicker = new mui.DtPicker({
  "type": "time",
  "customData": {
    "h": [
      {
        value: "am",
        text: "上午"
      }, {
        value: "pm",
        text: "下午"
      },
    ],
  }
})
dtpicker.show(function(e) {
  console.log(e);
})
```

支持的值:

可选值	视图模式
'y'	可设置 ^年 别名
'm'	可设置 ^月 别名
'd'	可设置 ^日 别名
'h'	可设置 ^时 别名
'i'	可设置 ^分 别名

* customData 值生效的前提需要有指定的日期视图,如设置'y',需要在视图使用'年'视图

3.参数:labels

Type: Array ()
设置默认标签区域提示语
可设置 ["年", "月", "日", "时", "分"] 这五个字段,
可以根据视图模式选择设置个别标签,也可以设置所有标签,
dtpicker 显示的时候只会根据当前视图显示设置项,
*建议不要设置空字符串,会影响美观哦

4.参数:beginDate

Type: Date ()

设置可选择日期最小范围

可单独设置最小年范围, 如: beginYear:2015 ,

其他日期支持Date格式,如: new Date(2016,5) 可指定最小月份6月

5.参数:endDate

Type: Date ()

设置可选择日期最大范围

可单独设置最大年范围, 如: endYear:2017 ,

其他日期支持Date格式,如: new Date(2016,10) 可指定最大月份11月

完整示例:

```
var dtpicker = new mui.DtPicker({
  type: "datetime", //设置日历初始视图模式
  beginDate: new Date(2015, 04, 25), //设置开始日期
  endDate: new Date(2016, 04, 25), //设置结束日期
  labels: ['Year', 'Mon', 'Day', 'Hour', 'min'], //设置默认标签区域提示语
  customData: {
    h: [{
      value: 'AM',
      text: 'AM'
    }, {
      value: 'PM',
      text: 'PM'
    }]
  } //时间/日期别名
})
dtpicker.show(function(e) {
  console.log(e);
})
```

getSelectedItems()

返回值Date

Type: Date ()

获取选中的项

如: var iTems = dtPicker.getSelectedItems()

返回值:

如:

```
/*
 * iTems.value 拼合后的 value
 * iTems.text 拼合后的 text
 * iTems.y 年, 可以通过 rs.y.vaue 和 rs.y.text 获取值和文本
 * iTems.m 月, 用法同年
 * iTems.d 日, 用法同年
 * iTems.h 时, 用法同年
 * iTems.i 分 (minutes 的第二个字母), 用法同年
 */
```

show(getSelectedItems)

返回值:[data]

Type: Array ()

获取选中的项 (数组)

如:


```
dtpicker.show(function(items) {  
    /*  
    * items.value 拼合后的 value  
    * items.text 拼合后的 text  
    * items.y 年, 可以通过 rs.y.vaue 和 rs.y.text 获取值和文本  
    * items.m 月, 用法同年  
    * items.d 日, 用法同年  
    * items.h 时, 用法同年  
    * items.i 分 (minutes 的第二个字母), 用法同年  
    */  
    console.log(items);  
})
```

* return false; 可以阻止选择框的关闭

hide()

隐藏dtPicker
如:dtPicker.hide()

dispose()

释放组件资源(释放后将不能再操作组件)

如:dtPicker.dispose()

- * 通常情况下, 不需要释放组件资源, 初始化之后, 可以一直使用。
- * 当内容较多, 如不释放组件资源, 在某些设备上可能会卡顿。
- * 所以每次用完便立即调用 dispose() 进行释放,下次用时再创建新实例。

扩展阅读

* picker 组件会触发webview硬件加速,在部分手机上可以能出现卡顿或变形的情况此时需要开启硬件加速,硬件加速请参考硬件加速章节(<http://ask.dcloud.net.cn/article/55>)

代码块激活字符: `mpoppicker` `mdtpicker`

progressbar (滚动条)

有准确值的进度条

- 有准确值的进度条会显示当前进度正处于整体进度的占比位置, 用户可以更直观的预期完成时间;
- 使用进度条控件, 需要一个进度条控件容器, mui会自动识别该容器下是否有进度条控件, 若没有, 则自动创建。

进度条控件DOM结构:

```
<div id="demo1" class="mui-progressbar">  
    <span></span>  
</div>
```

备注: 若想在页面顶部显示进度条, 则无需编写任何HTML代码, mui会自动创建顶部进度条控件, 但同样需要调用JS方法进行初始化;

初始化:

```
mui(container).progressbar({progress:20}).show();
```

例如:

```
mui("#demo1").progressbar({progress:20}).show();
```

mui初始化的逻辑为:

检查当前容器(container控件)自身是否含有.mui-progressbar类:

- 若有, 则以当前容器为目标控件, 直接显示进度;
- 没有, 检查当前容器的直接孩子节点是否含有.mui-progressbar类, 若存在, 则以遍历到的第一个含有.mui-progressbar类的孩子节点为目标控件, 显示进度;
- 若当前容器的直接孩子节点, 均不含.mui-progressbar类,则自动创建进度条控件;

更改显示进度条:

```
mui(container).progressbar().setProgress(50);
```

关闭进度条:

```
mui(container).progressbar().hide();
```

备注: 关闭进度条一般用于动态创建 (DOM中预先未定义) 的进度条, 调用hide方法后, 会直接删掉对应的DOM节点; 若已提前创建好DOM节点的进度条, 调用hide方法无效;

无限循环进度条:

若无法准确提供当前进度, 可以提供无限循环进度条 (无限循环进度条类似于waiting等待框,参考[HTML5+规范] (http://www.html5plus.org/doc/zh_cn/nativeui.html#plus.nativeUI.showWaiting))

无限循环进度条和准确值的进度条的用法基本相同, 有如下差异:

进度条控件DOM结构 (多了 .mui-progressbar-infinite):

```
<div id="demo1" class="mui-progressbar mui-progressbar-infinite">
  <span></span>
</div>
```

初始化

```
mui("#demo1").progressbar().show();
```

注意: 无限循环进度条不显示具体进度, 因此初始化时, 不能传入progress参数; mui框架也是根据是否progressbar构造方法中是否含有progress参数来区分当前进度条为有准确值的进度条还是无限循环进度条;

同样因为不显示具体进度的原因, 无限循环进度条调用setProgress()方法无效。

关闭进度条

```
mui("#demo1").progressbar().hide();
```

扩展阅读

代码块激活字符: `mprogressbar`

radio (单选框)

radio用于单选的情况

DOM结构

```
<div class="mui-input-row mui-radio">
  <label>radio</label>
  <input name="radio1" type="radio">
</div>
```

默认radio在右侧显示, 若希望在左侧显示, 只需增加 .mui-left 类即可, 如下:

```
<div class="mui-input-row mui-radio">
  <label>radio</label>
  <input name="radio1" type="radio">
</div>
```

若要禁用radio，只需在radio上增加disabled属性即可；

mui基于列表控件，提供了列表式单选实现；在列表根节点上增加 .mui-table-view-radio 类即可，若要默认选中某项，只需要在对应li节点上增加 .mui-selected 类即可，dom结构如下：

```
<ul class="mui-table-view mui-table-view-radio">
  <li class="mui-table-view-cell">
    <a class="mui-navigate-right">Item 1</a>
  </li>
  <li class="mui-table-view-cell mui-selected">
    <a class="mui-navigate-right">Item 2</a>
  </li>
  <li class="mui-table-view-cell">
    <a class="mui-navigate-right">Item 3</a>
  </li>
</ul>
```

列表式单选在切换选中项时会触发selected事件，在事件参数（e.detail.el）中可以获得当前选中的dom节点，如下代码打印当前选中项的innerHTML：

```
var list = document.querySelector('.mui-table-view.mui-table-view-radio');
list.addEventListener('selected',function(e){
  console.log("当前选中的为："+e.detail.el.innerHTML);
});
```

扩展阅读

代码块激活字符：`mradio`

range（滑块）

滑块常用于区间数字选择

DOM结构

```
<div class="mui-input-row mui-input-range">
  <label>Range</label>
  <input type="range" min="0" max="100">
</div>
```

扩展阅读

代码块激活字符：`mrange`

scroll(区域滚动)

在App开发中，div区域滚动的需求是普遍存在的，但系统默认实现又有如下问题：

- Android平台4.0以下不支持div滚动
- Android平台4.0以上支持div滚动，但不显示滚动条
- 弹出层的div滚动在iOS平台存在事件透传的问题

因此，mui额外提供了区域滚动组件，使用时需要遵循如下DOM结构

```
<div class="mui-scroll-wrapper">
  <div class="mui-scroll">
    <!--这里放置真实显示的DOM内容-->
  </div>
</div>
```

区域滚动组件默认为absolute定位，全屏显示；在实际使用过程中，需要手动设置滚动区域的位置（top和height）

若使用区域滚动组件，需手动初始化scroll控件

*常用配置项:

scroll.options

```
options = {
  scrollY: true, //是否竖向滚动
  scrollX: false, //是否横向滚动
  startX: 0, //初始化时滚动至x
  startY: 0, //初始化时滚动至y
  indicators: true, //是否显示滚动条
  deceleration: 0.0006 //阻尼系数,系数越小滑动越灵敏
  bounce: true, //是否启用回弹
}
```

示例：初始化scroll控件：

```
mui('.mui-scroll-wrapper').scroll({
  deceleration: 0.0005 //flick 减速系数，系数越大，滚动速度越慢，滚动距离越小，默认值0.0006
});
```

mui中iOS平台的下拉刷新（Android平台的下拉刷新使用的是双webview+原生滚动方案）、popover、可拖动式选项卡均使用了scroll组件。为方便大家使用，mui还额外为scroll插件封装了部分方法。

滚动到特定位置

scrollTo(xpos , ypos [, duration])

xpos

Type: Integer ()

要在窗口文档显示区左上角显示的文档的 x 坐标

ypos

Type: Integer ()

要在窗口文档显示区左上角显示的文档的 y 坐标

duration

Type: Integer ()

滚动时间周期，单位是毫秒

示例：快速回滚到该区域顶部，代码如下：

```
mui('.mui-scroll-wrapper').scroll().scrollTo(0,0,100);//100毫秒滚动到顶
```

滚动到底部位置

滚动到顶部的代码比较容易实现,坐标值设为0、0即可；但滚动到底部，需要计算该区域的实际高度，因此mui封装了scrollToBottom方法。

scrollToBottom(duration)

duration

Type: Integer ()

滚动时间周期，单位是毫秒

横向滚动

横向滚动只需在scroll组件基础上添加 `mui-slider-indicator``mui-segmented-control` `mui-segmented-control-inverted` 这三个class即可.(给予元素添加 `mui-control-item` 可加大文字间距增强体验如:)

```
<div class="mui-scroll-wrapper mui-slider-indicator mui-segmented-control mui-segmented-control-inverted">
  <div class="mui-scroll">
    <a class="mui-control-item mui-active">
      推荐
    </a>
    <a class="mui-control-item">
      热点
    </a>
    <a class="mui-control-item">
      北京
    </a>
    <a class="mui-control-item">
      社会
    </a>
    <a class="mui-control-item">
      娱乐
    </a>
    <a class="mui-control-item">
      科技
    </a>
  </div>
</div>
```

扩展阅读

代码块激活字符: `mscroll` `mscrollsegmented`

slide（轮播组件）

轮播组件是mui提供的一个核心组件，在该核心组件基础上，衍生出了图片轮播、可拖动式图文表格、可拖动式选项卡、左右滑动9宫格等组件，这些组件有较多共同点。首先，Dom构造基本相同，如下：

```
<div class="mui-slider">
  <div class="mui-slider-group">
    <!-- 第一个内容区容器 -->
    <div class="mui-slider-item">
      <!-- 具体内容 -->
    </div>
    <!-- 第二个内容区 -->
    <div class="mui-slider-item">
      <!-- 具体内容 -->
    </div>
  </div>
</div>
```

当拖动切换显示内容时，会触发slide事件，通过该事件的detail.slideNumber参数可以获得当前显示项的索引（第一项索引为0，第二项为1，以此类推），利用该事件，可在显示内容切换时，动态处理一些业务逻辑。

如下为一个可拖动式选项卡示例，为提高页面加载速度，页面加载时，仅显示第一个选项卡的内容，第二、第三选项卡内容为空。

当切换到第二、第三个选项卡时，再动态获取相应内容进行显示：

```
var item2Show = false,item3Show = false;//子选项卡是否显示标志
document.querySelector('.mui-slider').addEventListener('slide', function(event) {
  if (event.detail.slideNumber === 1&&!item2Show) {
    //切换到第二个选项卡
    //根据具体业务，动态获得第二个选项卡内容；
    var content = ....
    //显示内容
    document.getElementById("item2").innerHTML = content;
    //改变标志位，下次直接显示
    item2Show = true;
  } else if (event.detail.slideNumber === 2&&!item3Show) {
    //切换到第三个选项卡
    //根据具体业务，动态获得第三个选项卡内容；
    var content = ....
    //显示内容
    document.getElementById("item3").innerHTML = content;
    //改变标志位，下次直接显示
    item3Show = true;
  }
});
```

图片轮播、可拖动式图文表格等均可按照同样方式监听内容变化，比如我们可以在图片轮播界面显示当前正在看的是第几张图片：

```
document.querySelector('.mui-slider').addEventListener('slide', function(event) {
  //注意slideNumber是从0开始的；
  document.getElementById("info").innerText = "你正在看第" +(event.detail.slideNumber+1) + "张图片";
});
```

扩展阅读

代码块激活字符：`mslider`

switch(开关)

mui提供了开关控件，点击滑动两种手势都可以对开关控件进行操作,UI如下：



默认开关控件,带on/off文字提示，打开时为绿色背景，基本class类为 `.mui-switch` 、 `.mui-switch-handle` ，DOM结构如下：

```
<div class="mui-switch">
  <div class="mui-switch-handle"></div>
</div>
```

若希望开关默认为打开状态，只需要在 `.mui-switch` 节点上增加 `.mui-active` 类即可，如下：

```
<!-- 开关打开状态，多了一个.mui-active类 -->
<div class="mui-switch mui-active">
  <div class="mui-switch-handle"></div>
</div>
```

若希望隐藏on/off文字提示，变成简洁模式，需要在 `.mui-switch` 节点上增加 `.mui-switch-mini` 类，如下：

```
<!-- 简洁模式开关关闭状态 -->
<div class="mui-switch mui-switch-mini">
  <div class="mui-switch-handle"></div>
</div>
<!-- 简洁模式开关打开状态 -->
<div class="mui-switch mui-switch-mini mui-active">
  <div class="mui-switch-handle"></div>
</div>
```

mui默认还提供了蓝色开关控件，只需在 `.mui-switch` 节点上增加 `.mui-switch-blue` 类即可，如下：

```
<!-- 蓝色开关关闭状态 -->
<div class="mui-switch mui-switch-blue">
  <div class="mui-switch-handle"></div>
</div>
<!-- 蓝色开关打开状态 -->
<div class="mui-switch mui-switch-blue mui-active">
  <div class="mui-switch-handle"></div>
</div>
```

蓝色开关上增加 `.mui-switch-mini` 即可变成无文字的简洁模式

方法

若要获得当前开关状态，可通过判断当前开关控件是否包含 `.mui-active` 类来实现，若包含，则为打开状态，否则即为关闭状态；如下为代码示例：

```
var isActive = document.getElementById("mySwitch").classList.contains("mui-active");
if(isActive){
  console.log("打开状态");
}else{
  console.log("关闭状态");
}
```

若使用js打开、关闭开关控件，可使用switch插件的 `toggle()` 方法，如下为示例代码：

```
mui("#mySwitch").switch().toggle();
```

事件

开关控件在打开/关闭两种状态之间进行切换时，会触发toggle事件,通过事件的detail.isActive属性可以判断当前开关状态。可通过监听toggle事件，可以在开关切换时执行特定业务逻辑。如下为使用示例：

```
document.getElementById("mySwitch").addEventListener("toggle", function(event){
  if(event.detail.isActive){
    console.log("你启动了开关");
  }else{
    console.log("你关闭了开关");
  }
})
```

扩展阅读

问答社区话题讨论：[switch](http://ask.dcloud.net.cn/topic/switch) (<http://ask.dcloud.net.cn/topic/switch>)、[开关](http://ask.dcloud.net.cn/topic/开关) (<http://ask.dcloud.net.cn/topic/开关>)

代码块激活字符：`mswitch`

mui遵循 MIT License (<https://github.com/dcloudio/mui/blob/master/LICENSE>)

最新版本 v2.8.0 · [问答社区](http://ask.dcloud.net.cn/explore/category-3) (<http://ask.dcloud.net.cn/explore/category-3>) · [Issues](https://github.com/dcloudio/mui/issues) (<https://github.com/dcloudio/mui/issues>) · [Releases](https://github.com/dcloudio/mui/releases) (<https://github.com/dcloudio/mui/releases>) ·

-->

页面初始化

在app开发中，若要使用HTML5+扩展api (<http://www.html5plus.org/#specification>)，必须等plusready事件发生后才能正常使用，mui将该事件封装成了 `mui.plusReady()` 方法，涉及到HTML5+的api，建议都写在mui.plusReady方法中。如下为打印当前页面URL的示例：

```
mui.plusReady(function(){
    console.log("当前页面URL: "+plus.webview.currentWebview().getURL());
});
```

扩展阅读

`mui.init()` (<http://dev.dcloud.net.cn/mui/util/#init>) mui插件初始化

`mui.ready()` 当DOM准备就绪时，指定一个函数来执行。

代码块激活字符: `minit`

创建子页面

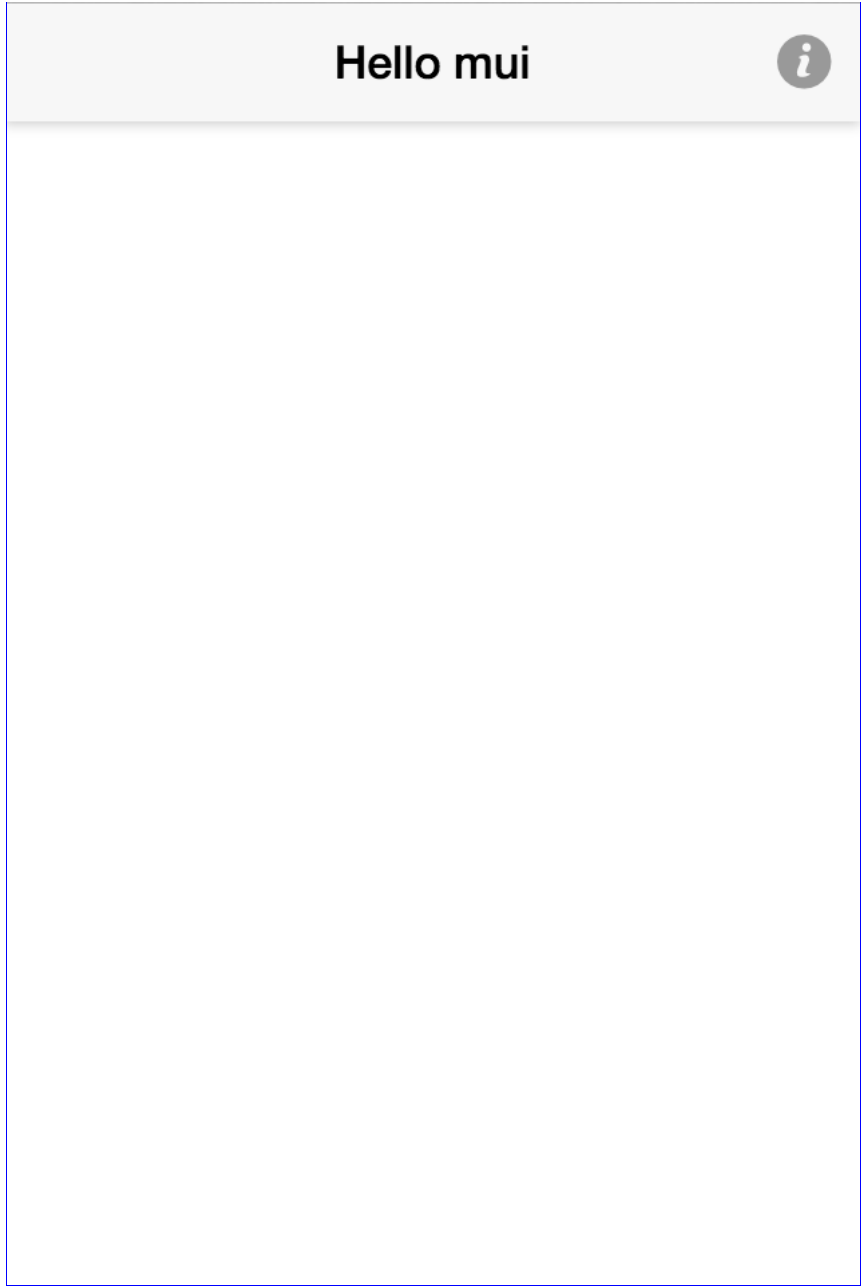
在mobile app开发过程中，经常遇到卡头卡尾的页面，此时若使用局部滚动，在android手机上会出现滚动不流畅的问题；mui的解决思路是：将需要滚动的区域通过单独的webview实现，完全使用原生滚动。具体做法则是：将目标页面分解为主页面和内容页面，主页面显示卡头卡尾区域，比如顶部导航、底部选项卡等；内容页面显示具体需要滚动的内容，然后在主页面中调用mui.init方法初始化内容页面。

```
mui.init({
    subpages:[{
        url:your-subpage-url,//子页面HTML地址，支持本地地址和网络地址
        id:your-subpage-id,//子页面标志
        styles:{
            top:subpage-top-position,//子页面顶部位置
            bottom:subpage-bottom-position,//子页面底部位置
            width:subpage-width,//子页面宽度，默认为100%
            height:subpage-height,//子页面高度，默认为100%
            .....
        },
        extras:{}//额外扩展参数
    }]
});
```

参数说明：styles表示窗口属性，参考5+规范中的WebViewStyle

(http://www.dcloud.io/docs/api/zh_cn/webview.shtml#plus.webview.WebviewStyle)；特别注意，height和width两个属性,即使不设置，也默认按100%计算；因此若设置了top值为非"0px"的情况，建议同时设置bottom值，否则5+ runtime根据高度100%计算，可能会造成页面真实底部位置超出屏幕范围的情况；left、right同理。

示例：Hello mui的首页其实就是index.html加list.html合并而成的，如下：



index.html

badge（数字角标）>

button（按钮）v

checkbox（复选框）>

dialog（消息框）>

gallery slider（图片轮播）v

gallery table（图文表格）v

grid（9宫格）v

icon（图标）>

input（输入框）>

list（列表）v

badge（数字角标）>

button（按钮）v

checkbox（复选框）>

dialog（消息框）>

gallery slider（图片轮播）v

gallery table（图文表格）v

grid（9宫格）v

icon（图标）>

input（输入框）>

list（列表）v

合并后的首页

list.html

index.html的作用就是显示固定导航，list.html显示具体列表内容，列表项的滚动是在list.html所在webview中使用原生滚动，既保证了滚动条不会穿透顶部导航，符合app的体验，也保证了列表流畅滚动，解决了区域滚动卡顿的问题。list.html就是index.html的子页面，创建代码比较简单，如下：

```
mui.init({
  subpages:[{
    url:'list.html',
    id:'list.html',
    styles:{
      top:'45px',//mui标题栏默认高度为45px;
      bottom:'0px'//默认为0px，可不定义;
    }
  }]
});
```

扩展阅读

代码块激活字符: misubpage

打开新页面

做web app，一个无法避开的问题就是转场动画；web是基于链接构建的，从一个页面点击链接跳转到另一个页面，如果通过有刷新的打开方式，用户要面对一个空白的页面等待；如果通过无刷新的方式，用Javascript移入DOM节点（常见的SPA解决方案），会碰到很高的性能挑战：DOM节点繁多，页面太大，转场动画不流畅甚至导致浏览器崩溃；mui的解决思路是：单webview只承载单个页面的dom，减少dom层级及页面大小；页面切换使用原生动画，将最耗性能的部分交给原生实现。

```
mui.openWindow({
  url:new-page-url,
  id:new-page-id,
  styles:{
    top:newpage-top-position,//新页面顶部位置
    bottom:newage-bottom-position,//新页面底部位置
    width:newpage-width,//新页面宽度，默认为100%
    height:newpage-height,//新页面高度，默认为100%
    .....
  },
  extras:{
    .....//自定义扩展参数，可以用来处理页面间传值
  },
  createNew:false,//是否重复创建同样id的webview，默认为false:不重复创建，直接显示
  show:{
    autoShow:true,//页面loaded事件发生后自动显示，默认为true
    aniShow:animationType,//页面显示动画，默认为"slide-in-right";
    duration:animationTime//页面动画持续时间，Android平台默认100毫秒，iOS平台默认200毫秒；
  },
  waiting:{
    autoShow:true,//自动显示等待框，默认为true
    title:'正在加载...',//等待对话框上显示的提示内容
    options:{
      width:waiting-dialog-widht,//等待框背景区域宽度，默认根据内容自动计算合适宽度
      height:waiting-dialog-height,//等待框背景区域高度，默认根据内容自动计算合适高度
      .....
    }
  }
})
```

参数说明：

- styles表示窗口参数，参考5+规范中的WebViewStyle (http://www.dcloud.io/docs/api/zh_cn/webview.shtml#plus.webview.WebviewStyle)；特别注意，height和width两个属性,即使不设置，也默认按100%计算；因此若设置了top值为非"0px"的情况，建议同时设置bottom值，否则5+ runtime根据高度100%计算，可能会造成页面真实底部位置超出屏幕范围的情况；left、right同理。
- extras:新窗口的额外扩展参数，可用来处理页面间传值；例如：var webview = mui.openWindow({url:'info.html',extras:{name:'mui'}});console.log(webview.name);，会输出"mui"字符串；注意：扩展参数仅在打开新窗口时有效，若目标窗口为预加载页面，则通过mui.openWindow方法打开时传递的extras参数无效。
- createNew：是否重复创建相同id的webview；为优化性能、避免app中重复创建webview，mui v1.7.0开始增加createNew参数，默认为false；判断逻辑如下：若createNew为true，则不判断重复，每次都新建webview；若为false，则先计算当前App中是否已存在同样id的webview，若存在则直接显示；否则则新建并根据show参数执行显示逻辑；该参数可能导致的影响：若业务写在plusReady事件中，而plusReady事件仅首次创建时会触发，则下次再次通过 mui.openWindow 方法打开同样webview时，是不会再次触发plusReady事件的，此时可通过自定义事件触发；案例参考：<http://ask.dcloud.net.cn/question/6514> (<http://ask.dcloud.net.cn/question/6514>)；
- show表示窗口显示控制。autoShow：目标窗口loaded事件发生后，是否自动显示；若目标页面为预加载页面，则该参数无效；aniShow表示页面显示动画，比如从右侧划入、从下侧划入等，具体可参考5+规范中的AnimationTypeShow (http://www.dcloud.io/docs/api/zh_cn/webview.shtml#plus.webview.AnimationTypeShow)
- waiting表示系统等待框；mui框架在打开新页面时等待框的处理逻辑为：显示等待框-->创建目标页面webview-->目标页面loaded事件发生-->关闭等待框；因此，只有当新页面为新建页面（webview）时，会显示等待框，否则若为预加载好的页面，则直接显示目标页面，不会显示等待框。waiting中的参数：autoShow表示自动显示等待框，默认为true，若为false，则不显示等待框；注意：若显示了等待框，但目标页面不自动显示，则需在目标页面中通过如下代码关闭等待框 plus.nativeUI.closeWaiting();。title表示等待框上的提示文字，options表示等待框显示参数，比如宽高、背景色、提示文字颜色等，具体可参考5+规范中的WaitingOption (http://www.dcloud.io/docs/api/zh_cn/nativeUI.shtml#plus.nativeUI.WaitingOption)。

示例1：Hello mui中，点击首页右上角的图标，会打开关于页面，实现代码如下：

```
//tap为mui封装的单击事件，可参考手势事件章节
document.getElementById('info').addEventListener('tap', function() {
  //打开关于页面
  mui.openWindow({
    url: 'examples/info.html',
    id: 'info'
  });
});
```

因没有传入styles参数，故默认全屏显示；也没有传入show参数，故使用slide-in-right动画，新页面从右侧滑入。

示例2：从A页面打开B页面，B页面为一个需要从服务端加载的列表页面，若在B页面loaded事件发生时就将其显示出来，因服务器数据尚未加载完毕，列表页面为空，用户体验不好；可通过如下方式改善用户体验（最好的用户体验应该是通过预加载的方式）：第一步，B页面loaded事件发生后，不自动显示；

```
//A页面中打开B页面，设置show的autoShow为false，则B页面在其loaded事件发生后，不会自动显示；
mui.openWindow({
  url: 'B.html',
  show:{
    autoShow:false
  }
});
```

第二步，在B页面获取列表数据后，再关闭等待框、显示B页面

```
//B页面onload从服务器获取列表数据；
window.onload = function(){
  //从服务器获取数据
  ....
  //业务数据获取完毕，并已插入当前页面DOM；
  //注意：若为ajax请求，则需将如下代码放在处理完ajax响应数据之后；
  mui.plusReady(function(){
    //关闭等待框
    plus.nativeUI.closeWaiting();
    //显示当前页面
    mui.currentWebview.show();
  });
}
```

扩展阅读

代码块激活字符：`mopenwindow`

关闭页面

mui框架将窗口关闭功能封装在 `mui.back` 方法中，具体执行逻辑是：

- 若当前webview为预加载页面，则hide当前webview；
- 否则，close当前webview；

在mui框架中，有三种操作会触发页面关闭（执行mui.back方法）：

- 点击包含 `.mui-action-back` 类的控件
- 在屏幕内，向右快速滑动
- Android手机按下back按键

iOS平台原生支持从屏幕边缘右滑关闭

iOS平台可通过popGesture参数实现从屏幕边缘右滑关闭webview，参考5+规范 (http://www.html5plus.org/doc/zh_cn/webview.html#plus.webview.WebviewStyle)，若想禁用该功能，可通过setStyle方法设置popGesture为none。

hbuilder中敲 `mheader` 生成的代码块，会自动生成带有返回导航箭头的标题栏，点击返回箭头可关闭当前页面，原因就是该返回箭头包含 `.mui-action-back` 类，代码如下：

```
<header class="mui-bar mui-bar-nav">
  <a class="mui-action-back mui-icon mui-icon-left-nav mui-pull-left"></a>
  <h1 class="mui-title">标题</h1>
</header>
```

若希望在顶部导航栏之外的其它区域添加关闭页面的控件，只需要在对应控件上添加 `.mui-action-back` 类即可，如下为一个关闭按钮示例：

```
<button type="button" class='mui-btn mui-btn-danger mui-action-back'>关闭</button>
```

mui框架封装的页面右滑关闭功能，默认未启用，若要使用右滑关闭功能，需要在 `mui.init()`；方法中设置`swipeBack`参数，如下：

```
mui.init({
  swipeBack:true //启用右滑关闭功能
});
```

mui框架默认会监听Android手机的back按键，然后执行页面关闭逻辑；若不希望mui自动处理back按键，可通过如下方式关闭mui的back按键监听：

```
mui.init({
  keyEventBind: {
    backbutton: false //关闭back按键监听
  }
});
```

除了如上三种操作外，也可以直接调用 `mui.back()` 方法，执行窗口关闭逻辑：

`mui.back()` 仅处理窗口逻辑，若希望在窗口关闭之前再处理一些其它业务逻辑，则可将业务逻辑抽象成一个具体函数，然后注册为 `mui.init` 方法的 `beforeback` 参数，`beforeback` 的执行逻辑为：

- 执行 `beforeback` 参数对应的函数若返回 `false`，则不再执行 `mui.back()` 方法；
- 否则（返回 `true` 或无返回值），继续执行 `mui.back()` 方法；

示例：从列表打开详情页面，从详情页面再返回后希望刷新列表界面，此时可注册 `beforeback` 参数，然后通过自定义事件 (`../event/#customevent`) 通知列表页面刷新数据，示例代码如下：

```
mui.init({
  beforeback: function(){
    //获得列表界面的webview
    var list = plus.webview.getWebviewById('list');
    //触发列表界面的自定义事件（refresh），从而进行数据刷新
    mui.fire(list,'refresh');
    //返回true，继续页面关闭逻辑
    return true;
  }
});
```

注意：`beforeback` 的执行返回必须是同步的（阻塞模式），若使用 `nativeUI` 这种异步js（非阻塞模式），则可能会出现意想不到的结果；比如：通过 `plus.nativeUI.confirm()` 弹出确认框，可能用户尚未选择，页面已经返回了（`beforeback` 同步执行完毕，无返回值，继续执行 `mui.back()` 方法，`nativeUI` 不会阻塞js进程）：在这种情况下，若要自定义业务逻辑，就需要复写 `mui.back` 方法了；如下为一个自定义示例，每次都需要用户确认后，才会关闭当前页面

```
//备份mui.back，mui.back已将窗口关闭逻辑封装的比较完善（预加载及父子窗口），因此最好复用mui.back
var old_back = mui.back;
mui.back = function(){
  var btn = ["确定","取消"];
  mui.confirm('确认关闭当前窗口?', 'Hello MUI', btn, function(e){
    if(e.index==0){
      //执行mui封装好的窗口关闭逻辑；
      old_back();
    }
  });
}
```

为何设置了 `swipeBack: false`，在iOS上依然可以右滑关闭？

iOS平台原生支持从屏幕边缘右滑关闭，这个是通过 `popGesture` 参数控制的，参考5+规范 (http://www.html5plus.org/doc/zh_cn/webview.html#plus.webview.WebviewStyle)，若需禁用，可通过 `setStyle` 方法设置 `popGesture` 为 `none`。

能否通过 `addEventListener` 增加back按键监听实现自定义关闭逻辑？

`addEventListener` 只会增加新的执行逻辑，老的监听逻辑(`mui.back`)依然会执行，因此，若需实现自定义关闭逻辑，一定要重写 `mui.back`。

扩展阅读

代码块激活字符: `mback`

预加载

所谓的预加载技术就是在用户尚未触发页面跳转时，提前创建目标页面，这样当用户跳转时，就可以立即进行页面切换，节省创建新页面的时间，提升app使用体验。mui提供两种方式实现页面预加载。

方式一：通过mui.init方法中的preloadPages参数进行配置。

```
mui.init({
  preloadPages:[
    {
      url:prelaod-page-url,
      id:preload-page-id,
      styles:{},//窗口参数
      extras:{},//自定义扩展参数
      subpages:[{}],{}//预加载页面的子页面
    }
  ],
  preloadLimit:5//预加载窗口数量限制(一旦超出,先进先出)默认不限制
});
```

该种方案使用简单、可预加载多个页面，但不会返回预加载每个页面的引用，若要获得对应webview引用，还需要通过plus.webview.getWebviewById 方式获得；另外，因为mui.init是异步执行，执行完mui.init方法后立即获得对应webview引用，可能会失败，例如如下代码：

```
mui.init({
  preloadPages:[
    {
      url:'list.html',
      id:'list'
    }
  ]
});
var list = plus.webview.getWebviewByid('list');//这里可能返回空；
```

方式二：通过mui.preload方法预加载。

```
var page = mui.preload({
  url:new-page-url,
  id:new-page-id,//默认使用当前页面的url作为id
  styles:{},//窗口参数
  extras:{},//自定义扩展参数
});
```

通过 mui.preload() 方法预加载，可立即返回对应webview的引用，但一次仅能预加载一个页面；若需加载多个webview，则需多次调用 mui.preload() 方法；

如上两种方案，各有优劣，需根据具体业务场景灵活选择；

判断预加载是否成功

方式一、通过直观现象分析

预加载页面会立即打开，不会显示等待框；非预加载页面默认会先显示等待框，再显示新页面；/p>

方式二、增加log分析预加载页面是否已创建

比如：A页面中预加载B页面，则在A页面完全加载（可通过setTimeout模拟）后，打印当前应用所有webview，看是否包含B页面的url，以此来分析。

例如：在A页面增加如下代码：

```
mui.plusReady(function(){
    setTimeout(function(){
        var array = plus.webview.all();
        if(array){
            for(var i=0,len=array.length;i<len;i++){
                console.log(array[i].getURL());
            }
        }
    },5000)
});
```

扩展阅读

代码块激活字符:

`miniptpreload`

`mpreload(单个webview)`

mui遵循 MIT License (<https://github.com/dcloudio/mui/blob/master/LICENSE>)

最新版本 v2.8.0 · 问答社区 (<http://ask.dcloud.net.cn/explore/category-3>) · Issues (<https://github.com/dcloudio/mui/issues>) · Releases (<https://github.com/dcloudio/mui/releases>) ·

事件绑定

除了可以使用 `addEventListener()` 方法监听某个特定元素上的事件外， 也可以使用 `.on()` 方法实现批量元素的事件绑定。

.on(event , selector , handler)

event
Type: String ()
需监听的事件名称，例如： 'tap'

selector
Type: String ()
选择器

handler
Type: Function ()(Event () event)
事件触发时的回调函数，通过回调中的event参数可以获得事件详情

示例

点击新闻列表，获取当前列表项的id，并将该id传给新闻详情页面，然后打开新闻详情页面

```
mui(".mui-table-view").on('tap', '.mui-table-view-cell',function(){
  //获取id
  var id = this.getAttribute("id");
  //传值给详情页面，通知加载新数据
  mui.fire(detail,'getDetail',{id:id});
  //打开新闻详情
  mui.openWindow({
    id:'detail',
    url:'detail.html'
  });
})
```

扩展阅读

代码块激活字符:

mmon

事件取消

使用 `on()` 方法绑定事件后，若希望取消绑定，则可以使用 `off()` 方法。 `off()` 方法根据传入参数的不同，有不同的实现逻辑。

.off(event , selector , handler)

version added: 2.0.0

event
Type: String ()
需取消绑定的事件名称，例如： 'tap'

selector
Type: String ()
选择器

handler
Type: Function ()
之前绑定到该元素上的事件函数，不支持匿名函数

.off(event , selector)

version added: 2.0.0

event

<div>Type: String () 需取消绑定的事件名称，例如：'tap'</div> <div>selector Type: String () 选择器</div>	
<div>.off(event)</div> <div>event Type: String () 需取消绑定的事件名称，例如：'tap'</div>	version added: 2.2.0
<div>.off()</div> <div>空参数，删除该元素上所有事件</div>	version added: 2.2.0

示例

off(event,selector,handle) 适用于取消对应选择器上特定事件所执行的特定回调，例如：

```
//点击li时，执行foo_1函数
mui("#list").on("tap","li",foo_1);
//点击li时，执行foo_2函数
mui("#list").on("tap","li",foo_2);

function foo_1(){
  console.log("foo_1 execute");
}

function foo_2(){
  console.log("foo_2 execute");
}
//点击li时，不再执行foo_1函数，但会继续执行foo_2函数
mui("#list").off("tap","li",foo_1);
```

off(event,selector) 适用于取消对应选择器上特定事件的所有回调，例如：

```
//点击li时，执行foo_1函数
mui("#list").on("tap","li",foo_1);
//点击li时，执行foo_2函数
mui("#list").on("tap","li",foo_2);

function foo_1(){
  console.log("foo_1 execute");
}

function foo_2(){
  console.log("foo_2 execute");
}
//点击li时，foo_2、foo_2两个函数均不再执行
mui("#list").off("tap","li");
```

off(event) 适用于取消当前元素上绑定的特定事件的所有回调，例如：

```
//点击li时，执行foo_1函数
mui("#list").on("tap","li",foo_1);
//点击p时，执行foo_3函数
mui("#list").on("tap","p",foo_3);

function foo_1(){
  console.log("foo_1 execute");
}

function foo_3(){
  console.log("foo_3 execute");
}
//点击li时，不再执行foo_1函数；点击p时，也不再执行foo_3函数
mui("#list").off("tap");
```

`off()` 适用于取消当前元素上绑定的所有事件回调，例如：

```
// 点击li时，执行foo_1函数
mui("#list").on("tap", "li", foo_1);
// 双击li时，执行foo_4函数
mui("#list").on("doubletap", "li", foo_4);
// 点击p时，执行foo_3函数
mui("#list").on("tap", "p", foo_3);

function foo_1(){
  console.log("foo_1 execute");
}

function foo_3(){
  console.log("foo_3 execute");
}

function foo_4(){
  console.log("foo_4 execute");
}
// 点击li时，不再执行foo_1函数；点击p时，也不再执行foo_3函数；双击li时，也不再执行foo_4函数；
mui("#list").off();
```

扩展阅读

代码块激活字符: `mmoff`

事件触发

使用 `mui.trigger()` 方法可以动态触发特定DOM元素上的事件。

.trigger(element , event , data)

element
Type: Element ()
触发事件的DOM元素

event
Type: String ()
事件名字，例如： 'tap'、 'swipeleft'

data
Type: Object ()
需要传递给事件的业务参数

示例

自动触发按钮的点击事件：

```
var btn = document.getElementById("submit");
// 监听点击事件
btn.addEventListener("tap", function () {
  console.log("tap event trigger");
});
// 触发submit按钮的点击事件
mui.trigger(btn, 'tap');
```

部分mui控件监听的事件无法通过 `mui.trigger` 触发
比如无法实现自动触发mui返回图标，实现关闭当前页面的功能，该部分逻辑正在优化中

扩展阅读

代码块激活字符: `mtrigger`

手势事件

在开发移动端的应用时，会用到很多的手势操作，比如滑动、长按等，为了方便开放者快速集成这些手势，mui内置了常用的手势事件，目前支持的手势事件见如下列表：

分类	参数	描述
点击	tap	单击屏幕
	doubletap	双击屏幕
长按	longtap	长按屏幕
	hold	按住屏幕
	release	离开屏幕
滑动	swipeleft	向左滑动
	swiperight	向右滑动
	swipeup	向上滑动
	swipedown	向下滑动
拖动	dragstart	开始拖动
	drag	拖动中
	dragend	拖动结束

手势事件配置

根据使用频率，mui默认会监听部分手势事件，如点击、滑动事件；为了开发出更高性能的moble App，mui支持用户根据实际业务需求，通过mui.init方法中的gestureConfig参数，配置具体需要监听的手势事件，。

```
mui.init({
  gestureConfig:{
    tap: true, //默认为true
    doubletap: true, //默认为false
    longtap: true, //默认为false
    swipe: true, //默认为true
    drag: true, //默认为true
    hold:false,//默认为false，不监听
    release:false//默认为false，不监听
  }
});
```

注意:dragstart、drag、dragend共用drag开关，swipeleft、swiperight、swipeup、swipedown共用swipe开关

事件监听

单个元素上的事件监听，直接使用 addEventListener() 即可，如下：

```
elem.addEventListener("swipeleft",function(){
  console.log("你正在向左滑动");
});
```

若多个元素执行相同逻辑，则建议使用事件绑定(on())。

扩展阅读

代码块激活字符: `minitgesture`

自定义事件

在App开发中，经常会遇到页面间传值的需求，比如从新闻列表页进入详情页，需要将新闻id传递过去； Html5Plus规范设计了evalJS (http://www.html5plus.org/doc/zh_cn/webview.html#plus.webview.WebviewObject.evalJS)方法来解决该问题； 但evalJS方法仅接收字符串参数，涉及多个参数时，需要开发人员手动拼字符串； 为简化开发，mui框架在evalJS方法的基础上，封装了自定义事件，通过自定义事件，用户可以轻松实现多webview间数据传递。

仅能在5+ App及流应用中使用

因为是多webview之间传值，故无法在手机浏览器、微信中使用；

监听自定义事件

添加自定义事件监听操作和标准js事件监听类似，可直接通过window对象添加，如下：

```
window.addEventListener('customEvent',function(event){
    //通过event.detail可获得传递过来的参数内容
    ....
});
```

触发自定义事件

通过 mui.fire() 方法可触发目标窗口的自定义事件：

.fire(target , event , data)

target

Type: WebviewObject (http://www.html5plus.org/doc/zh_cn/webview.html#plus.webview.WebviewObject)
需传值的目标webview

event

Type: String ()
自定义事件名称

data

Type: JSON ()
json格式的数据

目标webview必须触发loaded事件后才能使用自定义事件

若新创建一个webview，不等该webview的loaded事件发生，就立即使用webview.evalJS()或mui.fire(webview,'eventName',{})，则可能无效；案例参考：[这里](http://ask.dcloud.net.cn/question/11022) (<http://ask.dcloud.net.cn/question/11022>)

示例

假设如下场景：从新闻列表页面进入新闻详情页面，新闻详情页面为共用页面，通过传递新闻ID通知详情页面需要显示具体哪个新闻，详情页面再动态向服务器请求数据，mui要实现类似需求可通过如下步骤实现：

- 在列表页面中预加载详情页面（假设为detail.html）
- 列表页面在点击新闻标题时，首先，获得该新闻id，触发详情页面的newsId事件，并将新闻id作为事件参数传递过去；然后再打开详情页面；
- 详情页面监听newsId自定义事件

列表页面代码如下：

```
//初始化预加载详情页面
mui.init({
  preloadPages:[{
    id:'detail.html',
    url:'detail.html'
  }]
});

var detailPage = null;
//添加列表项的点击事件
mui('.mui-content').on('tap', 'a', function(e) {
  var id = this.getAttribute('id');
  //获得详情页面
  if(!detailPage){
    detailPage = plus.webview.getWebviewById('detail.html');
  }
  //触发详情页面的newsId事件
  mui.fire(detailPage,'newsId',{
    id:id
  });
});
//打开详情页面
mui.openWindow({
  id:'detail.html'
});
});
```

详情页面代码如下：

```
//添加newId自定义事件监听
window.addEventListener('newsId',function(event){
  //获得事件参数
  var id = event.detail.id;
  //根据id向服务器请求新闻详情
  .....
});
```

扩展阅读

代码块激活字符: `mfire`

mui遵循 MIT License (<https://github.com/dcloudio/mui/blob/master/LICENSE>)

最新版本 v2.8.0 · 问答社区 (<http://ask.dcloud.net.cn/explore/category-3>) · Issues (<https://github.com/dcloudio/mui/issues>) · Releases (<https://github.com/dcloudio/mui/releases>) ·

init

mui框架将很多功能配置都集中在mui.init方法中, 要使用某项功能, 只需要在mui.init方法中完成对应参数配置即可, 目前支持在mui.init方法中配置的功能包括: 创建子页面 (<http://dev.dcloud.net.cn/mui/window/#subpage>)、关闭页面 (<http://dev.dcloud.net.cn/mui/window/#closewindow>)、手势事件配置 (<http://dev.dcloud.net.cn/mui/event/#gesture>)、预加载 (<http://dev.dcloud.net.cn/mui/window/#preload>)、下拉刷新 (<http://dev.dcloud.net.cn/mui/pulldown/#pullrefresh-down>)、上拉加载 (<http://dev.dcloud.net.cn/mui/pullup/#pullrefresh-up>)、设置系统状态栏背景颜色。

mui需要在页面加载时初始化很多基础控件,如监听返回键,因此务必在每个页面中调用

以下各配置模块在其对应文档中有详细阐述,请点击链接查看,这里只列出所有可配置项

```
mui.init({
//子页面
    subpages: [{
        //...
    }],
//预加载
    preloadPages:[
        //...
    ],
//下拉刷新、上拉加载
    pullRefresh : {
        //...
    },
//手势配置
    gestureConfig:{
        //...
    },
//侧滑关闭
    swipeBack:true, //Boolean(默认false)启用右滑关闭功能

//监听Android手机的back、menu按键
    keyEventBind: {
        backbutton: false, //Boolean(默认true)关闭back按键监听
        menubutton: false //Boolean(默认true)关闭menu按键监听
    },
//处理窗口关闭前的业务
    beforeback: function() {
        //... //窗口关闭前处理其他业务详情点击 + "关闭页面"链接查看
    },
//设置状态栏颜色
    statusBarBackground: '#9defbcg', //设置状态栏颜色,仅iOS可用
    preloadLimit:5//预加载窗口数量限制(一旦超出,先进先出)默认 unlimited
})
```

以上各配置模块在其对应文档中有详细阐述,请点击链接查看,以下只补充单独配置项

通过 `statusBarBackground: rgb` 属性设置状态栏颜色 (iOS7.0+、安卓不支持) 格式为#RRGGBB。

```
mui.init({
    statusBarBackground: '#9defbcg',
})
```

mui默认会监听Android手机的物理按键 (back&menu) ,若不希望自动处理按键可通过以下方式关闭

```
mui.init({
  //监听Android手机的back、menu按键
  keyEventBind: {
    backbutton: true, //Boolean(默认true)关闭back按键监听
    menubutton: true //Boolean(默认true)关闭menu按键监听
  },
})
```

扩展阅读

代码块激活字符: `minit`

mui()

mui使用css选择器获取HTML元素，返回mui对象数组。

mui("p")：选取所有 <p> 元素

mui(".p.title")：选取所有包含 .title 类的 <p> 元素

若要将mui对象转化成dom对象，可使用如下方法（类似jquery对象转成dom对象）：

```
//obj1是mui对象
var obj1 = mui("#title");
//obj2是dom对象
var obj2 = obj1[0];
```

MUI框架的定位是“最接近原生体验的移动App的UI框架”，因此和jQuery有所区别，很少为简化DOM操作而封装API，具体可参考MUI产品概述 (<http://ask.dcloud.net.cn/article/91>)；该函数的设计目的，更多是为了配合MUI插件使用，比如图片轮播、下拉刷新、区域滚动等，如下为详细示例：

示例1：跳转到图片轮播的第二张图片

```
mui('.mui-slider').slider().gotoItem(1);
```

示例2：重新开启上拉加载

```
mui('#pullup-container').pullRefresh().refresh(true);
```

扩展阅读

代码块激活字符: `mmui`

each()

each既是一个类方法，同时也是一个对象方法，两个方法适用场景不同；换言之，你可以使用 mui.each() 去遍历数组或json对象，也可以使用 mui(selector).each() 去遍历DOM结构。

mui.each(obj , handler)

obj

Type: Array ()|JSONObj ()

需遍历的对象或数组；若为对象，仅遍历对象根节点下的key

handler

Type: Function ()(Integer ()|String () index,Anything () element)

为每个元素执行的回调函数；其中，index表示当前元素的下标或key，element表示当前匹配元素

mui(selector).each(handler)

handler

Type: Function ()(Integer () index,Element () element)

为每个匹配元素执行的回调函数；其中，**index**表示当前元素在匹配元素中的位置（下标，从0开始），**element**表示当前匹配元素，可用 **this** 关键字代替

示例1

输出当前数组中每个元素的平方

```
var array = [1,2,3]
mui.each(array,function(index,item){
    console.log(item*item);
})
```

示例2

当前页面中有三个字段，如下：

```
<div class="mui-input-group">
  <div class="mui-input-row">
    <label>字段1: </label>
    <input type="text" class="mui-input-clear" id="col1" placeholder="请输入">
  </div>
  <div class="mui-input-row">
    <label>字段2: </label>
    <input type="text" class="mui-input-clear" id="col2" placeholder="请输入">
  </div>
  <div class="mui-input-row">
    <label>字段3: </label>
    <input type="text" class="mui-input-clear" id="col3" placeholder="请输入">
  </div>
</div>
```

提交时校验三个字段均不能为空，若为空则提醒并终止业务逻辑运行，使用 **each()** 方法循环校验，如下：

```
var check = true;
mui(".mui-input-group input").each(function () {
    //若当前input为空，则alert提醒
    if(!this.value||trim(this.value)==""){
        var label = this.previousElementSibling;
        mui.alert(label.innerText+"不允许为空");
        check = false;
        return false;
    }
});
//校验通过，继续执行业务逻辑
if(check){
    //.....
}
```

扩展阅读

代码块激活字符: `meach` `mmeach(遍历DOM)`

extend()

将两个对象合并成一个对象。

.extend(target , object1 [, objectN])

target

Type: Object ()

需合并的目标对象

object1

Type: Object ()

需合并的对象

objectN

Type: Object ()

需合并的对象

.extend(deep , target , object1 [, objectN])

deep

Type: Boolean ()

若为true，则递归合并

target

Type: Object ()

需合并的目标对象

object1

Type: Object ()

需合并的对象

objectN

Type: Object ()

需合并的对象

示例

```
var target = {
  company:"dcloud",
  product:{
    mui:"小巧、高效"
  }
}
var obj1 = {
  city:"beijing",
  product:{
    HBuilder:"飞一样的编码"
  }
}
mui.extend(target,obj1);
//输出: {"company":"dcloud","product":{"HBuilder":"飞一样的编码"},"city":"beijing"}
console.log(JSON.stringify(target));
```

从如上输出可以看到，product节点下的mui被替换成了HBuilder，因为默认仅合并对象根节点下的key、value；如果想深度合并，则可以传入 deep 参数，如下：

```
var target = {
  company:"dcloud",
  product:{
    mui:"小巧、高效"
  }
}
var obj1 = {
  city:"beijing",
  product:{
    HBuilder:"飞一样的编码"
  }
}
//支持深度合并
mui.extend(true,target,obj1);
//输出: {"company":"dcloud","product":{"mui":"小巧、高效","HBuilder":"飞一样的编码"},"city":"beijing"}
console.log(JSON.stringify(target));
```

扩展阅读

代码块激活字符: `mextend`

later()

setTimeout封装

.later(func , delay [, context, data])

func

Type: Function ()

delay毫秒后要执行的函数

delay

Type: Int ()

延迟的毫秒数

context

Type: Object ()

上下文

扩展阅读

代码块激活字符: `mLater`

scrollTo()

滚动窗口屏幕到指定位置，该方法是对 `window.scrollTo()` 方法在手机端的增强实现，可设定滚动动画时间及滚动结束后的回调函数;鉴于手机屏幕大小，该方法仅可实现屏幕纵向滚动。

.scrollTo(ypos [, duration] [, handler])

ypos

Type: Integer ()

要在窗口文档显示区左上角显示的文档的 y 坐标

duration

Type: Integer ()

滚动时间周期，单位是毫秒

handler

Type: Function ()

滚动结束后执行的回调函数

示例

1秒钟之内滚动到页面顶部

```
mui.scrollTo(0,1000);
```

扩展阅读

代码块激活字符: `mscrollto`

os

我们经常会有通过 navigator.userAgent 判断当前运行环境的需求,mui对此进行了封装,通过调用mui.os.XXX即可

<p>Android(可以访问的参数为:)</p> <p>.wechat 返回是否为微信端</p> <p>.android 返回是否为安卓手机</p> <p>.version 安卓版本号</p> <p>.isBadAndroid android非Chrome环境</p>
<p>iOS(可以访问的参数为:)</p> <p>.iOS 返回是否为苹果设备</p> <p>.version 返回手机版本号</p> <p>.iphone 返回是否为苹果手机</p> <p>.ipad 返回是否为ipad</p>
<p>plus(可以访问的参数为:)</p> <p>.plus 返回是否在基座中运行</p> <p>.stream 返回是否为流应用</p>

示例

检测是否为iOS或安卓系统版本是否小于4.4

```
if(mui.os.ios||(mui.os.android&&parseFloat(mui.os.version)<4.4)){  
  //...  
}
```

代码块激活字符:

mui遵循 MIT License (<https://github.com/dcloudio/mui/blob/master/LICENSE>)

最新版本 v2.8.0 · 问答社区 (<http://ask.dcloud.net.cn/explore/category-3>) · Issues (<https://github.com/dcloudio/mui/issues>) · Releases (<https://github.com/dcloudio/mui/releases>) ·

Ajax

mui框架基于html5plus的XMLHttpRequest (<http://www.html5plus.org/#specification/#specification/XMLHttpRequest.html>)，封装了常用的Ajax函数，支持GET、POST请求方式，支持返回json、xml、html、text、script数据类型；本着极简的设计原则，mui提供了mui.ajax方法，并在mui.ajax方法基础上，进一步简化出最常用的mui.get()、mui.getJSON()、mui.post()三个方法。

mui.ajax(url[, settings])

url
Type: String ()
请求发送的目标地址

settings
Type: PlainObject ()
key/value格式的json对象，用来配置ajax请求参数，支持的完整参数参考如下 `mui.ajax([settings])` 方法

mui.ajax([settings])

settings
Type: PlainObject ()
key/value格式的json对象，用来配置ajax请求参数，支持的详细参数如下：

data
Type: PlainObject ()|String ()
发送到服务器的业务数据

dataType
Type: String ()
预期服务器返回的数据类型；如果不指定，mui将自动根据HTTP包的MIME头信息自动判断；支持设置的dataType可选值：

"xml": 返回XML文档

"html": 返回纯文本HTML信息；

"script": 返回纯文本JavaScript代码

"json": 返回JSON数据

"text": 返回纯文本字符串

error
Type: Function ()（XMLHttpRequest xhr,String type,String errorThrown）
请求失败时触发的回调函数，该函数接收三个参数：

xhr：xhr实例对象

type：错误描述，可取值："timeout","error","abort","parsererror"、"null"

errorThrown：可捕获的异常对象

success
Type: Function ()（Anything data,String textStatus,XMLHttpRequest xhr）
请求成功时触发的回调函数，该函数接收三个参数：

data：服务器返回的响应数据，类型可以是json对象、xml对象、字符串等；

textStatus：状态描述，默认值为'success'

xhr：xhr实例对象

timeout
Type: Number ()

请求超时时间（毫秒），默认值为0，表示永不超时；若超过设置的超时时间(非0的情况)，依然未收到服务器响应，则触发`error`回调

type

Type: String ()

请求方式，目前仅支持'GET'和'POST'，默认为'GET'方式

headers

Type: Json ()

指定HTTP请求的Header

```
headers: {'Content-Type', 'application/json'}
```

crossDomain ^{*5+ only}

Type: Boolean ()

强制使用5+跨域

代码示例：如下为通过post方式向某服务器发送鉴权登录的代码片段

```
mui.ajax('http://server-name/login.php',{
  data:{
    username:'username',
    password:'password'
  },
  dataType:'json',//服务器返回json格式数据
  type:'post',//HTTP请求类型
  timeout:10000,//超时时间设置为10秒;
  headers: {'Content-Type', 'application/json'}
  success:function(data){
    //服务器返回响应，根据响应结果，分析是否登录成功;
    ...
  },
  error:function(xhr,type,errorThrown){
    //异常处理;
    console.log(type);
  }
});
```

`mui.post()` 方法是对 `mui.ajax()` 的一个简化方法，直接使用POST请求方式向服务器发送数据、且不处理`timeout`和异常（若需处理异常及超时，请使用 `mui.ajax()` 方法），使用方法：`mui.post(url[,data][,success][,dataType]`），如上登录鉴权代码换成 `mui.post()` 后，代码更为简洁，如下：

```
mui.post('http://server-name/login.php',{
  username:'username',
  password:'password'
},function(data){
  //服务器返回响应，根据响应结果，分析是否登录成功;
  ...
},'json'
);
```

`mui.get()` 方法和 `mui.post()` 方法类似，只不过是直接使用GET请求方式向服务器发送数据、且不处理`timeout`和异常（若需处理异常及超时，请使用 `mui.ajax()` 方法），使用方法：`mui.get(url[,data][,success][,dataType]`），如下为获得某服务器新闻列表的代码片段，服务器以json格式返回数据列表

```
mui.get('http://server-name/list.php',{category:'news'},function(data){
  //获得服务器响应
  ...
},'json'
);
```

如上 `mui.get()` 方法和如下 `mui.ajax()` 方法效果是一致的：

```
mui.ajax('http://server-name/list.php',{
    data:{
        category:'news'
    },
    dataType:'json',//服务器返回json格式数据
    type:'get',//HTTP请求类型
    success:function(data){
        //获得服务器响应
        ...
    }
});
```

`mui.getJSON()` 方法是在 `mui.get()` 方法基础上的更进一步简化，限定返回json格式的数据，其它参数和 `mui.get()` 方法一致，使用方法：`mui.get(url[,data][,success])`，如上获得新闻列表的代码换成 `mui.getJSON()` 方法后，更为简洁，如下：

```
mui.getJSON('http://server-name/list.php',{category:'news'},function(data){
    //获得服务器响应
    ...
});
```

扩展阅读

问答社区话题讨论：[Ajax \(http://ask.dcloud.net.cn/topic/Ajax\)](http://ask.dcloud.net.cn/topic/Ajax)

代码块激活字符：`majax` `mget` `mjson`

mui遵循 MIT License (<https://github.com/dcloudio/mui/blob/master/LICENSE>)

最新版本 v2.8.0 · [问答社区 \(http://ask.dcloud.net.cn/explore/category-3\)](http://ask.dcloud.net.cn/explore/category-3) · [Issues \(https://github.com/dcloudio/mui/issues\)](https://github.com/dcloudio/mui/issues) · [Releases \(https://github.com/dcloudio/mui/releases\)](https://github.com/dcloudio/mui/releases)

下拉刷新

为实现下拉刷新功能，大多H5框架都是通过DIV模拟下拉回弹动画，在低端android手机上，DIV动画经常出现卡顿现象（特别是图文列表的情况）；mui通过双webview解决这个DIV的拖动流畅度问题：拖动时，拖动的不是div，而是一个完整的webview（子webview），回弹动画使用原生动画；在iOS平台，H5的动画已经比较流畅，故依然使用H5方案。两个平台实现虽有差异，但mui经过封装，可使用一套代码实现下拉刷新。

主页面内容比较简单，只需要创建子页面即可：

```
mui.init({
  subpages:[{
    url:pullrefresh-subpage-url,//下拉刷新内容页面地址
    id:pullrefresh-subpage-id,//内容页面标志
    styles:{
      top:subpage-top-position,//内容页面顶部位置,需根据实际页面布局计算，若使用标准mui导航，顶部默认为48px；
      .....//其它参数定义
    }
  }]
});
```

iOS平台的下拉刷新，使用的是mui封装的区域滚动组件（../ui/index.html#scroll），为保证两个平台的DOM结构一致，内容页面需统一按照如下DOM结构构建：

```
<!--下拉刷新容器-->
<div id="refreshContainer" class="mui-content mui-scroll-wrapper">
  <div class="mui-scroll">
    <!--数据列表-->
    <ul class="mui-table-view mui-table-view-chevron">

      </ul>
    </div>
  </div>
```

其次，通过mui.init方法中pullRefresh参数配置下拉刷新各项参数，如下：

```
mui.init({
  pullRefresh : {
    container:"#refreshContainer",//下拉刷新容器标识，querySelector能定位的css选择器均可，比如：id、.class等
    down : {
      height:50,//可选，默认50.触发下拉刷新拖动距离，
      auto: true,//可选，默认false.自动下拉刷新一次
      contentdown : "下拉可以刷新",//可选，在下拉可刷新状态时，下拉刷新控件上显示的标题内容
      contentover : "释放立即刷新",//可选，在释放可刷新状态时，下拉刷新控件上显示的标题内容
      contentrefresh : "正在刷新...",//可选，正在刷新状态时，下拉刷新控件上显示的标题内容
      callback :pullfresh-function //必选，刷新函数，根据具体业务来编写，比如通过ajax从服务器获取新数据；
    }
  }
});
```

下拉刷新是mui框架的一个插件，该插件目前有下拉刷新结束、滚动到特定位置两个方法：

下拉刷新结束

在下拉刷新过程中，当获取新数据后，需要执行endPulldownToRefresh方法，该方法的作用是关闭“正在刷新”的雪花进度提示，内容区域回滚顶部位置，如下：。

```
function pullfresh-function() {
  //业务逻辑代码，比如通过ajax从服务器获取新数据；
  .....
  //注意，加载完新数据后，必须执行如下代码，注意：若为ajax请求，则需将如下代码放置在处理完ajax响应数据之后
  mui('#refreshContainer').pullRefresh().endPulldownToRefresh();
}
```

滚动到特定位置

下拉刷新组件滚动到特定位置的方法类似区域滚动组件

scrollTo(xpos , ypos [, duration])**xpos**

Type: Integer ()

要在窗口文档显示区左上角显示的文档的 x 坐标

ypos

Type: Integer ()

要在窗口文档显示区左上角显示的文档的 y 坐标

duration

Type: Integer ()

滚动时间周期，单位是毫秒

示例：在hello mui下拉刷新示例中，实现了双击标题栏，让列表快速回滚到顶部的功能；代码如下：

```
var contentWebview = null;
//监听标题栏的双击事件
document.querySelector('header').addEventListener('doubletap',function () {
    if(contentWebview==null){
        contentWebview = plus.webview.currentWebview().children()[0];
    }
    //内容区滚动到顶部
    contentWebview.evalJS("mui('#pullrefresh').pullRefresh().scrollTo(0,0,100)");
});
```

更改下拉刷新文字位置

根据实际需求在父页面给mui-content设置top属性

```
.mui-bar-nav ~ .mui-content .mui-pull-top-pocket{
    top: 180px !important;
}
```

扩展阅读

问答社区话题讨论： 下拉刷新 (<http://ask.dcloud.net.cn/topic/下拉刷新>)

代码块激活字符： `mipull(DOM结构)` `minitpull(初始化组件)` `mmpull(组件方法)`

mui遵循 MIT License (<https://github.com/dcloudio/mui/blob/master/LICENSE>)

最新版本 v2.8.0 · 问答社区 (<http://ask.dcloud.net.cn/explore/category-3>) · Issues (<https://github.com/dcloudio/mui/issues>) · Releases (<https://github.com/dcloudio/mui/releases>) ·

概述

mui的上拉加载和下拉刷新类似，都属于 pullRefresh 插件，使用过程如下：

- 1、页面滚动到底，显示“正在加载...”提示（mui框架提供）
- 2、执行加载业务数据逻辑（开发者提供）
- 3、加载完毕，隐藏“正在加载”提示（mui框架提供）

开发者只需关心业务逻辑，实现加载更多数据即可。

初始化

初始化方法类似下拉刷新，通过mui.init方法中pullRefresh参数配置上拉加载各项参数，如下：

```
mui.init({
  pullRefresh : {
    container:refreshContainer,//待刷新区域标识，querySelector能定位的css选择器均可，比如：id、.class等
    up : {
      height:50,//可选。默认50。触发上拉加载拖动距离
      auto:true,//可选，默认false。自动上拉加载一次
      contentrefresh : "正在加载...",//可选，正在加载状态时，上拉加载控件上显示的标题内容
      contentnomore:'没有更多数据了',//可选，请求完毕若没有更多数据时显示的提醒内容；
      callback :pullfresh-function //必选，刷新函数，根据具体业务来编写，比如通过ajax从服务器获取新数据；
    }
  }
});
```

结束上拉加载

加载完新数据后，需要执行endPullupToRefresh()方法，结束转雪花进度条的“正在加载...”过程

.endPullupToRefresh(nomore)

nomore

Type: Boolean ()

是否还有更多数据；若还有更多数据，则传入false；否则传入true，之后滚动条滚动到底时，将不再显示“上拉显示更多”的提示语，而显示“没有更多数据了”的提示语；

示例：

```
function pullfresh-function() {
  //业务逻辑代码，比如通过ajax从服务器获取新数据；
  .....
  //注意：
  //1、加载完新数据后，必须执行如下代码，true表示没有更多数据了；
  //2、若为ajax请求，则需将如下代码放置在处理完ajax响应数据之后
  this.endPullupToRefresh(true|false);
}
```

重置上拉加载

若部分业务中，有重新触发上拉加载的需求（比如当前类别已无更多数据，但切换到另外一个类别后，应支持继续上拉加载），此时调用.refresh(true)方法，可重置上拉加载控件，如下代码：

```
//pullup-container为在mui.init方法中配置的pullRefresh节点中的container参数；
//注意：refresh()中需传入true
mui('#pullup-container').pullRefresh().refresh(true);
```

禁用上拉刷新

在部分场景下希望禁用上拉加载，比如在列表数据过少时，不想显示“上拉显示更多”、“没有更多数据”的提示语，开发者可以通过调用disablePullupToRefresh()方法实现类似需求，代码如下：

```
//pullup-container为在mui.init方法中配置的pullRefresh节点中的container参数；
mui('#pullup-container').pullRefresh().disablePullupToRefresh();
```

启用上拉刷新

使用 `disablePullupToRefresh()` 方法禁用上拉加载后, 可通过 `enablePullupToRefresh()` 方法再次启用上拉加载, 代码如下:

```
//pullup-container为在mui.init方法中配置的pullRefresh节点中的container参数;  
mui('#pullup-container').pullRefresh().enablePullupToRefresh();
```

扩展阅读

- 1、上拉加载时, 怎么隐藏底部的“没有更多数据了”? (<http://ask.dcloud.net.cn/question/5882>)
- 2、问答社区话题讨论: 上拉加载 (<http://ask.dcloud.net.cn/topic/上拉加载>)


mui遵循 MIT License (<https://github.com/dcloudio/mui/blob/master/LICENSE>)

最新版本 v2.8.0 · 问答社区 (<http://ask.dcloud.net.cn/explore/category-3>) · Issues (<https://github.com/dcloudio/mui/issues>) · Releases (<https://github.com/dcloudio/mui/releases>) ·

html

此底色代表最小触发字符 此底色代表非必要完整触发字符

*需HBuilder7.1+,或者下载mui_html_snippets.rb (<https://github.com/dcloudio/HBuilderRubyBundle/tree/master/html>) mui_js_snippets.rb (<https://github.com/dcloudio/HBuilderRubyBundle/tree/master/js>)替换使用

我们只整理了部分常用到的代码块.欢迎大家提交更多的代码块  (<https://github.com/dcloudio/HBuilderRubyBundle>)方便大家开发.

组件	触发字符
mDoctype(mui-dom结构)	<div>m</div> do <div>c</div> type
mBody(主体)	<div>m</div> bo <div>d</div> y
mScroll(区域滚动容器)	<div>m</div> sc <div>r</div> oll
mrefreshContainer(刷新容器)	<div>m</div> re <div>f</div> resh
mHeader(标题栏)	<div>m</div> he <div>a</div> der
mHeader(带返回箭头的标题栏)	<div>m</div> he <div>a</div> derwithBack
mCheckbox(复选框)	<div>m</div> ch <div>e</div> ckbox
mCheckbox(复选框居左)	<div>m</div> ch <div>e</div> ckbox_ <div>l</div> eft
mCheckbox(复选框禁用选项)	<div>m</div> ch <div>e</div> ckbox_ <div>d</div> isabled
mlcon(图标)	<div>m</div> i <div>c</div> on
mOffcanvas(侧滑导航-主界面、菜单同时移动)	<div>m</div> of <div>f</div> canvasall
mOffcanvas(侧滑导航-主界面移动、菜单不动)	<div>m</div> of <div>f</div> canvasmain
mOffcanvas(侧滑导航-主界面不动、菜单移动)	<div>m</div> of <div>f</div> canvasmenu
mOffcanvas(侧滑导航-缩放式侧滑（类手机QQ）)	<div>m</div> of <div>f</div> canvasscalable
mText(文本框)	<div>m</div> i <div>n</div> <div>p</div> uttext
mText_Search(搜索框)	<div>m</div> i <div>n</div> <div>p</div> utsearch
mText_Clear(带清除按钮的文本框)	<div>m</div> i <div>n</div> <div>p</div> utclear
mText_Speech(语音输入)	<div>m</div> i <div>n</div> <div>p</div> utspeech
mForm(表单)	<div>m</div> f <div>o</div> <div>r</div> m
mRadio(单选框)	<div>m</div> r <div>a</div> <div>d</div> io
mRadio(单选框居左)	<div>m</div> r <div>a</div> <div>d</div> io_ <div>l</div> eft
mRadio(禁用单选框)	<div>m</div> r <div>a</div> <div>d</div> io_ <div>d</div> isable
mRadios(默认选中指定项)	<div>m</div> r <div>a</div> <div>d</div> io_ <div>s</div> electd
mPopover(弹出菜单)	<div>m</div> p <div>o</div> <div>p</div> over
mprogressbar(进度条-无限循环)	<div>m</div> p <div>r</div> <div>o</div> gressbarinfinite
mprogressbar(进度条-有准确值)	<div>m</div> p <div>r</div> <div>o</div> gressbar
mActionsheet(H5模式弹出菜单)	<div>m</div> a <div>c</div> <div>t</div> ionsheet
mRange(Label+滑块)	<div>m</div> r <div>a</div> <div>n</div> gelabel
mSwitch(开关)	<div>m</div> s <div>w</div> <div>i</div> tch
mSwitch(开关 - 蓝色)	<div>m</div> s <div>w</div> <div>i</div> tch_ <div>b</div> lue
mSwitch(开关Mini)	<div>m</div> s <div>w</div> <div>i</div> tchmini

mSwitch(开关Mini - blue)	<div>msw</div> <div>itchmini_blue</div>
mbadge(数字角标)	<div>mba</div> <div>dge</div>
mbadge(数字角标无底色)	<div>mba</div> <div>dge_inverted</div>
mTab(底部选项卡)	<div>mta</div> <div>b</div>
mTabSegmented(div选项卡)	<div>mta</div> <div>bsegmented</div>
mTabSegmented(可左右拖动的选项卡)	<div>mta</div> <div>bviewpage</div>
mPagination(分页)	<div>mpa</div> <div>gination</div>
mList(列表)	<div>mli</div> <div>st</div>
mListMedia(图文列表图片居左)	<div>mli</div> <div>st_Media_left</div>
mListMedia(图文列表图片居右)	<div>mli</div> <div>st_Media_right</div>
mGrid(九宫格)	<div>mgr</div> <div>id</div>
mGallery-Table(图文表格)	<div>msl</div> <div>ider_gallery_table</div>
mGallery(图片轮播)	<div>msl</div> <div>ider_gallery</div>
slide(轮播组件)	<div>msl</div> <div>ider</div>
mactionsheet(操作表)	<div>act</div> <div>ionsheet</div>
maccordion(折叠面板)	<div>mac</div> <div>ordion</div>
mnumbox(数字输入框)	<div>mnu</div> <div>mbox</div>
mrefreshContainer(刷新容器)	<div>mpu</div> <div>llrefresh</div>
mButton(按钮)	<div>mbu</div> <div>tton</div>
mButton(按钮无底色,有边框)	<div>mbu</div> <div>tton_outline</div>
mButton(块状按钮)	<div>mbu</div> <div>tton_block</div>

init

组件	触发字符
mui.init	<div>min</div> <div></div>
创建子页面(subpage)	<div>mins</div> <div>ubpage</div>
预加载页面(preload)	<div>minp</div> <div>reload</div>
刷新组件(pullRefresh)	<div>minp</div> <div>ullRefresh</div>
手势事件(getures)	<div>ming</div> <div>esture</div>
侧滑返回(swipeback)	<div>mins</div> <div>wipeback</div>
按键绑定(keyeventbind)	<div>mink</div> <div>eyevent</div>
重写窗口关闭逻辑(beforeBack)	<div>minb</div> <div>eforeback</div>
设置状态栏颜色(setStatusbar)	<div>mins</div> <div>tatusbar</div>
预加载数量(preloadlimit)	<div>minp</div> <div>relimit</div>

js组件

组件	触发字符
mui.plusReady()	<div>mpl</div> usready
mui.ready	<div>mre</div> ady

event

组件	触发字符
mui.on(事件绑定)	<div>mmon</div>
mui.off(事件取消)	<div>mmoff</div>
mui.trigger()(事件触发)	<div>mtrigger</div>
mui.fire()(自定义事件)	<div>mfire</div>
documentgetElementById	<div>dg</div>
documentquerySelector	<div>ds</div>
documentquerySelectoraddEventListener	<div>dsa</div>
documentgetElementByIdaddEventListener	<div>dga</div>
windowaddEventListener	<div>wad</div>
documentaddEventListener	<div>dad</div>

dialog

组件	触发字符
mui.alert()(弹出框)	<div>mda</div> lert
mui.confirm()(确认弹出框)	<div>mdc</div> onfirm
mui.prompt()(输入弹出框)	<div>mdp</div> rompt
mui.toast()(自动消失提示框)	<div>mdt</div> oast
mui.closePopup()(关闭最外层对话框)	<div>mdc</div> losePopup
mui.closePopups()(关闭全部对话框)	<div>mdc</div> losePopups

utils

组件	触发字符
mui() (mui对象选择器)	<div>mmu</div> i
mui.each()(数组,对象遍历)	<div>mea</div> ch

mui.each()(DOM遍历)	<div>mme</div> <div>ach</div>
mui.extends(对象合并)	<div>mex</div> <div>tends</div>
mui.later() (setTimeout封装)	<div>mle</div> <div>ter</div>
mui.scrollTo() (滚动到指定位置)	<div>msc</div> <div>rollto</div>
mui.os()(判断当前运行环境)	<div>mos</div> <div></div>

ajax

组件	触发字符
mui.ajax()	<div>maj</div> <div>ax</div>
mui.post()	<div>mpo</div> <div>st</div>
mui.get()	<div>mge</div> <div>t</div>
mui.getJSON()	<div>mjs</div> <div>on</div>

webview

组件	触发字符
mui.open(打开新页面)	<div>mop</div> <div>en</div>
mui.currentWebview (当前页面)	<div>mcu</div> <div>rrent</div>
mui.back()(关闭窗口)	<div>mba</div> <div>ck</div>
mui.backFunction()(重写返回逻辑)	<div>mba</div> <div>ckfunction</div>
mui.backDouble()(双击退出应用)	<div>mba</div> <div>ckDouble</div>
mui.backTast(双击进入后台)	<div>mba</div> <div>cktast</div>
mui.preload()(预加载)	<div>mpr</div> <div>eload</div>

plus

组件	触发字符
plusReady	<div>pre</div> <div>ady</div>
plus.accelerometer	<div>pac</div> <div>ce</div>
plus.audio	<div>pau</div> <div>dio</div>
plus.barcode	<div>pba</div> <div>rcode</div>
plus.camera	<div>pca</div> <div>mera</div>
plus.contacts	<div>pco</div> <div>ntacts</div>

plus.device	<div>pdevice</div>
plus.gallery	<div>pgallery</div>
plus.geolocation	<div>pgeolocation</div>
plus.io	<div>pio</div>
plus.key	<div>pkey</div>
plus.maps	<div>pmaps</div>
plus.messaging	<div>pmessage</div>
plus.nativeObj	<div>pnativeObj</div>
plus.nativeUI	<div>pnativeUI</div>
plus.navigator	<div>pnavigator</div>
plus.orientation	<div>porientation</div>
plus.payment	<div>ppayment</div>
plus.proximity	<div>pproximity</div>
plus.push	<div>ppush</div>
plus.runtime	<div>pruntime</div>



mui遵循 MIT License (<https://github.com/dcloudio/mui/blob/master/LICENSE>)

最新版本 v2.8.0 · [问答社区 \(http://ask.dcloud.net.cn/explore/category-3\)](http://ask.dcloud.net.cn/explore/category-3) · [Issues \(https://github.com/dcloudio/mui/issues\)](https://github.com/dcloudio/mui/issues) · [Releases \(https://github.com/dcloudio/mui/releases\)](https://github.com/dcloudio/mui/releases)