- 快速上手
  - 部署
  - 目录说明
  - 宿主页面
  - 全局配置
  - 侧边菜单
- 路由
  - 路由规则
  - 路由跳转
  - 路由结尾
- 视图
  - 视图与路由的关系
  - 视图中加载 JS 模块
- 动态模板
  - 定义模板
  - 模板基础属性
  - 模板语法
- 登录与接口鉴权
- 基础方法
  - admin模块
  - view模块
- ID 唯一性
- 实用组件
- 事件监听
- 兼容性
- 缓存问题
- 升级事项
- 源码构建

目录

- 快速上手
  - 部署
  - 目录说明
  - 宿主页面
  - 全局配置
  - 侧边菜单
- 路由
  - 路由规则
  - 路由跳转
  - 路由结尾
- 视图
  - 视图与路由的关系
  - 视图中加载 JS 模块
- 动态模板
  - 定义模板
  - 模板基础属性
  - 模板语法
- 登录与接口鉴权
- 基础方法
  - admin模块
  - view模块
- ID 唯一性
- 实用组件
- 事件监听
- 兼容性
- 缓存问题
- 升级事项
- 源码构建

更新：4小时前 阅读：13834

layuiAdmin pro v1.x 【单页版】开发者文档

更新日志 常见问题 在线演示 会员专区 切换到：iframe版文档

> layuiAdmin pro 是完全基于 layui 架构而成的后台管理模板系统，可以更轻松地实现前后端分离，它是 mvc 的简化版，全面接管 *视图* 和 *页面路由*，并可自主完成数据渲染，服务端通常只负责数据接口，而前端只需专注视图和事件交互，所有的页面动作都是在一个宿主页面中完成，因此这赋予了 layuiAdmin 单页面应用开发的能力。

题外

- 该文档适用于 **layuiAdmin 专业版（单页面）**，阅读之前请务必确认是否与你使用的版本对应。
- 熟练掌握 layuiAdmin 的前提是熟练掌握 layui，因此除了本篇文档，layui 的文档 也是必不可少的存在。

快速上手

部署

1. 解压文件后，将 *layuiAdmin* 完整放置在任意目录
2. 通过本地 web 服务器去访问 *./start/index.html* 即可运行 Demo

> 由于 layuiAdmin 可采用前后端分离开发模式，因此你无需将其放置在你的服务端 MVC 框架中，你只需要给 layuiAdmin 主入口页面（我们也称之为：*宿主页面*）进行访问解析，它即可全权完成自身路由的跳转和视图的呈现，而数据层则完全通过服务端提供的异步接口来完成。

目录说明

- **src/**
  layuiAdmin 源代码，通常用于开发环境（如本地），推荐你在本地开发时，将 *./start/index.html* 中的 `layui.css` 和 `layui.js` 的引入路径由 dist 改为 src 目录。
  - **src/controller/**：存放 JS 业务模块，即对视图进行事件等交互性处理
  - **src/lib/**：layuiAdmin 的核心模块，一般不推荐修改
  - **src/style/**：存放样式，其中 *admin.css* 是核心样式
  - **src/views/**：存放视图文件。其中 *layout.html* 是整个框架结构的承载，一般不推荐做大量改动。
  - **src/config.js**：layuiAdmin 的全局配置文件，可随意修改。
  - **src/index.js**：layuiAdmin 的入口模块，一般不推荐修改
- **dist/**
  通过 gulp 将 layuiAdmin src 目录的源代码进行构建后生成的目录（即：将 JS 和 CSS 文件进行了压缩等处理），通常用于线上环境。关于 gulp 的使用，下文也有介绍。
- **start/**
  存放 layuiAdmin 的入口页面、模拟接口数据、layui

宿主页面

你所看到的 *start/index.html* 是我们提供好的宿主页面，它是整个单页面的承载，所有的界面都是在这一个页面中完成跳转和渲染的。事实上，宿主页面可以放在任何地方，但是要注意修改里面的 `<link>` `<script>` 的 src 和 layui.config 中 `base` 的路径。

全局配置

当你已经顺利在本地预览了 layuiAdmin 后，你一定迫不及待关注更深层的结构。打开 src 目录，你将看到 `config.js`，里面存储着所有的默认配置。你可以按照实际需求选择性修改，下面是 layuiAdmin 默认提供的配置：

```
layui.define(['laytpl', 'layer', 'element', 'util'], function(exports){
  exports('setter', {
    container: 'LAY_app' //容器ID
    ,base: layui.cache.base //记录layuiAdmin文件夹所在路径
    ,views: layui.cache.base + 'views/' //视图所在目录
    ,entry: 'index' //默认视图文件名
    ,engine: '.html' //视图文件后缀名
    ,pageTabs: false //是否开启页面选项卡功能。单页版不推荐开启

    ,name: 'layuiAdmin Pro'
    ,tableName: 'layuiAdmin' //本地存储表名
    ,MOD_NAME: 'admin' //模块事件名

    ,debug: true //是否开启调试模式。如开启，接口异常时会抛出异常 URL 等信息

    ,interceptor: false //是否开启未登入拦截

    //自定义请求字段
    ,request: {
      tokenName: 'access_token' //自动携带 token 的字段名。可设置 false 不携带。
    }

    //自定义响应字段
    ,response: {
      statusName: 'code' //数据状态的字段名称
      ,statusCode: {
        ok: 0 //数据状态一切正常的状态码
        ,logout: 1001 //登录状态失效的状态码
      }
      ,msgName: 'msg' //状态信息的字段名称
      ,dataName: 'data' //数据详情的字段名称
    }

    //独立页面路由，可随意添加（无需写参数）
    ,indPage: [
      '/user/login' //登入页
      ,'/user/reg' //注册页
      ,'/user/forget' //找回密码
      ,'/template/tips/test' //独立页的一个测试 demo
    ]

    //扩展的第三方模块
    ,extend: [
      'echarts', //echarts 核心包
      'echartsTheme' //echarts 主题
    ]

    //主题配置
    ,theme: {
      //配色方案，如果用户未设置主题，第一个将作为默认
      color: [{
        main: '#20222A' //主题色
        ,selected: '#009688' //选中色
        ,logo: '' //logo区域背景色
        ,header: '' //头部区域背景色
        ,alias: 'default' //默认别名
      }] //为了减少篇幅，更多主题此处不做列举，可直接参考 config.js

      //初始的颜色索引，对应上面的配色方案数组索引
      //如果本地已经有主题色记录，则以本地记录为优先，除非请求本地数据（localStorage）
      // 1.0 正式版开始新增
      ,initColorIndex: 0
    }
  });
});
```

侧边菜单

- 在 **start/json/menu.js** 文件中，我们放置了默认的侧边菜单数据，你可以去随意改动它。
- 如果你需要动态加载菜单，你需要将 **views/layout.html** 中的对应地址改成你的真实接口地址

侧边菜单最多可支持到三级。无论你采用静态的菜单还是动态的，菜单的数据格式都必须是一段合法的 JSON，且必须符合以下规范：

```
{
  "code": 0 //状态码，key 名可以通过 config.js 去重新配置
  ,"msg": "" //提示信息
  ,"data": [{ //菜单数据，key名可以通过 config.js 去重新配置
    "name": "component" //一级菜单名称（与视图的文件夹名称和路由路径对应）
    ,"title": "组件" //一级菜单标题
    ,"icon": "layui-icon-component" //一级菜单图标样式
    ,"jump": '' //自定义一级菜单路由地址，默认按照 name 解析。一旦设置，将优先按照 jump 设定的路由跳转
    ,"spread": true //是否默认展子菜单（1.0.0-beta9 新增）
    ,"list": [{ //二级菜单
      "name": "grid" //二级菜单名称（与视图的文件夹名称和路由路径对应）
      ,"title": "栅格" //二级菜单标题
      ,"jump": ''  //自定义二级菜单路由地址
      ,"spread": true //是否默认展子菜单（1.0.0-beta9 新增）
      ,"list": [{ //三级菜单
        "name": "list" //三级菜单名（与视图中最终的文件名和路由对应），如：component/grid/list
        ,"title": "等比例列表排列" //三级菜单标题
      },{
        "name": "mobile"
        ,"title": "按移动端排列"
      }]
    }]
}
```

> TIPS：实际运用时，切勿出现上述中的注释，否则将不是合法的 JSON ，会出现解析错误。

需要注意的是以下几点：
1. 当任意级菜单有子菜单，点击该菜单都只是收缩和展开操作，而并不会跳转，只有没有子菜单的菜单才被允许跳转。
2. 菜单的路由地址默认是按照菜单层级的 name 来设定的。
   我们假设一级菜单的 name 是：`a`，二级菜单的是：`b`，三级菜单的 name 是 `c`，那么：
   - 三级菜单最终的路由地址就是：`/a/b/c`
   - 如果二级菜单没有三级菜单，那么二级菜单就是最终路由，地址就是：`/a/b/`
   - 如果一级菜单没有二级菜单，那么一级菜单就是最终路由，地址就是：`/a/`
3. 但如果你设置了 参数 *jump*，那么就会优先读取 jump 设定的路由地址，如：`"jump": "/user/set"`

路由

layuiAdmin 的路由是采用 *location.hash* 的机制，即路由地址是放在 `./#/` 后面，并通过 layui 自带的方法：`layui.router()` 来进行解析。每一个路由都对应一个真实存在的视图文件，且路由地址和视图文件的路径是一致的（相对 *views* 目录）。因此，你不再需要通过配置服务端的路由去访问一个页面，也无需在 layuiAdmin 内部代码中去定义路由，而是直接通过 layuiAdmin 的前端路由去访问，即可匹配相应目录的视图，从而呈现出页面结果。

路由规则

```
./#/path1/path2/path3/key1=value1/key2=value2…
```

一个实际的示例：

```
./#/user/set
./#/user/set/uid=123/type=1#xxx（下面将以这个为例继续讲解）
```

当你需要对路由结构进行解析时，你只需要通过 layui 内置的方法 `layui.router()` 即可完成。如上面的路由解析出来的结果是：

```
{
 path: ['user','set']
 ,search: {uid: 123, type: 1}
 ,href: 'user/set/uid=123/type=1'
 ,hash: 'xxx'
}
```

可以看到，不同的结构会自动归纳到相应的参数中，其中：

> - path：存储的是路由的目录结构
> - search：存储的是路由的参数部分
> - href：存储的是 layuiAdmin 的完整路由地址
> - hash：存储的是 layuiAdmin 自身的锚记，跟系统自带的 `location.hash` 有点类似

通过 `layui.router()` 得到路由对象后，你就可以对页面进行个性化操作、异步参数传值等等。如：

```
//在 JS 中获取路由参数
var router = layui.router();
admin.req({
  url: 'xxx'
  ,data: {
    uid: router.search.uid
  }
});
```

```
<!--  在动态模板中获取路由参数 -->
<script type="text/html" template lay-url="./xxx/?uid={{ layui.router().search.uid }}">
  …
</script>

<!-- 或 -->
<script type="text/html" template lay-url="./xxx/" lay-data="{uid:'{{ layui.router().search.uid }}'}">
  …
</script>
```

路由跳转

通过上文的路由规则，你已经大致清楚了 layuiAdmin 路由的基本原理和解析方法。那么如何完成路由的跳转呢？
1. 在视图文件的 HTML 代码中，通过对任意元素设定 `lay-href="/user/set/uid=123/type=1"`，**好处是**：任意元素都可以触发跳转。**缺点是**：只能在浏览器当前选项卡完成跳转（注意：不是 layuiAdmin 的选项卡）
2. 直接对 a 标签设定 href，如：`<a href="#/user/set">text</a>`。**好处是**：你可以通过设定 `target="_blank"` 来打开一个浏览器新选项卡。**缺点是**：只能设置 `a` 标签，且前面必须加 `/#/`
3. 在 JS 代码中，还可通过 `location.hash = '/user/set';` 来跳转。前面无需加 `#`，它会自动追加。

路由结尾

在路由结尾部分出现的 `/` 与不出现，是两个完全不同的路由。比如下面这个：
1. user/set
   读取的视图文件是：.views/user/set.html
2. user/set/
   读取的视图文件是：./views/user/set/index.html （TIPS：这里的 *index.html* 即是目录下的默认主视图，下文会有讲解）

因此一定要注意结尾处的 `/`，避免视图读取错误。

视图

这或许是你应用 layuiAdmin 时的主要焦点，在开发过程中，你的大部分精力都可能会聚焦在这里。它取代了服务端 MVC 架构中的 *view* 层，使得应用开发变得更具扩展性。因此如果你采用 layuiAdmin 的 SPA（单页应用）模式，请务必要抛弃服务端渲染视图的思想，让页面的控制权限重新回归到前端吧！

> **views** 目录存放的正是视图文件，你可以在该目录添加任意的新目录和新文件，通过对应的路由即可访问。

注意：如果是单页面模式，视图文件通常是一段 HTML 碎片，而不能是一个完整的 html 代码结构。

视图与路由的关系

每一个视图文件，都对应一个路由。其中 `index.html` 是默认文件（你也可以通过 config.js 去重新定义）。视图文件的所在目录决定了路由的访问地址，如：

视图路径	对应的路由地址
./views/user/index.html	/user/
./views/user.html	/user
./views/user/set/index.html	/user/set/
./views/user/set.html	/user/set
./views/user/set/base.html	/user/set/base

通过上述的表格列举的对应关系，可以总结出：
- 当视图文件是 index.html，那么路由地址就是它的上级目录（相对 *views*），以 `/` 结尾
- 当视图文件不是 index.html，那么路由地址就是它的上级目录+视图文件名，不以 `/` 结尾

> 值得注意的是：路由路径并非最多只能三级，它可以无限极。但对应的视图也必须存放在相应的层级目录下

视图中加载 JS 模块

在视图文件中，除了写 HTML，也可以写 JavaScript 代码。如：

```
<div id="LAY-demo-hello">Hello layuiAdmin</div>
<script>
layui.use('admin', function(){
  var $ = layui.jquery;
  admin.popup({
    content: $('#LAY-demo-hello').html()
  });
});
</script>
```

如果该视图对应的 JS 代码量太大，我们更推荐你在 *controller* 目录下新增一个业务模块，并在视图中直接 layui.use 去加载该模块。下面以控制台主页 `index.html` 为例：

```
<div>html区域<div>

<script>
//加载 controller 目录下的对应模块
/*
  小贴士：
    这里 console 模块对应 的 console.js 并不会重复加载，
    然而该页面的视图可能会重新插入到容器，那如何保证脚本能重新控制视图呢？有两种方式：
      1): 借助 layui.factory 方法获取 console 模块的工厂（回调函数）给 layui.use
      2): 直接在 layui.use 方法的回调中书写业务代码，即:
          layui.use('console', function(){
            //同 console.js 中的 layui.define 回调中的代码
          });

  这里我们采用的是方式1。其它很多视图中采用的其实都是方式2，因为更简单些，也减少了一个请求数。

*/
layui.use('console', layui.factory('console'));
</script>
```

当视图被渲染后，layui.factory 返回的函数也会被执行，从而保证在不重复加载 JS 模块文件的前提下，保证脚本能重复执行。

动态模板

layuiAdmin 的视图是一个“动静结合”的载体，除了常规的静态模板，你当然还可以在视图中存放动态模板，因此它可谓是焦点中的焦点。

定义模板

在视图文件中，通过下述规则定义模板：

```
<script type="text/html" template>
  <!-- 动态模板碎片 -->
</script>
```

下面是一个简单的例子：

```
<script type="text/html" template>
  当前 layuiAdmin 的版本是：{{ layui.admin.v }}
  路由地址：{{ layui.router().href }}
</script>
```

在不对动态模板设定数据接口地址的情况下，它能读取到全局对象。但更多时候，一个动态模板应该是对应一个接口地址，如下所示：

```
<script type="text/html" template lay-url="接口地址">
    我叫：{{ d.data.username }}
    {{# if(d.data.sex === '男'){ }}
        公的
    {{# } else { }}
        母的
    {{# } }}
</script>
```

模板中的 `d` 对应的是你接口返回的 json 转化后的一维对象，如：

```
{
  "code": 0
  ,"data": {
    "username": "贤心"
    ,"sex": "男"
  }
}
```

那么，上述动态模板最终输出的结果就是：

```
我叫：贤心
公的
```

模板基础属性

动态模板支持以下基础属性

- **lay-url**
  用于绑定模板的数据接口地址，支持动态模板解析，如：

```
<script type="text/html" template lay-url="https://api.xxx.com?id={{ layui.router().search.id }}">
  <!-- 动态模板碎片 -->
</script>
```

- **lay-type**
  用于设定模板的接口请求类型（默认：get），如：

```
<script type="text/html" template lay-url="接口地址" lay-type="post">
  <!-- 动态模板碎片 -->
</script>
```

- **lay-data**
  用于定义接口请求的参数，其值是一个 JavaScript object 对象，同样支持动态模板解析，如：

```
<script type="text/html" template lay-url="接口地址" lay-data="{id: '{{ layui.router().search.id }}', type: 1}">
  <!-- 动态模板碎片 -->
</script>
```

- **lay-headers**
  用户定义接口请求的 Request Headers 参数，用法与 lay-data 的完全类似，支持动态模板解析。
- **lay-done**
  接口请求完毕并完成视图渲染的回调脚本，里面支持写任意的 JavaScript 语句。事实上它是一个封闭的函数作用域，通过给 Function 实例返回的函数传递一个参数 `d`，用于得到接口返回的数据：

```
<script type="text/html" template lay-url="接口地址" lay-done="console.log(d);">
  <!-- 动态模板碎片 -->
</script>
```

很多时候，你在动态模板中可能会放入一些类似于 layui 的 form 元素，而有些控件需要执行 `form.render()` 才会显示，这时，你可以对 lay-done 赋值一个全局函数，如：

```
<script type="text/html" template lay-url="接口地址" lay-done="layui.data.done(d);">
   <div class="layui-form" lay-filter="LAY-filter-demo-form">
     <input type="checkbox" title="复选框">
  </div>
</script>

<!--  注意：别看眼花了，下面可不是动态模板，而是 JS 脚本区域 -->
<script>
layui.data.done = function(d){
  layui.use(['form'], function(){
    var form = layui.form;
     form.render(null, 'LAY-filter-demo-form'); //渲染该模板下的动态表单
  });
};
</script>
```

TIPS:

> - 如果模板渲染完毕需要处理过多的交互，我们强烈推荐你采用上述的方式定义一个全局函数赋值给 lay-done，会极大地减少维护成本。
> - 无需担心该全局函数的冲突问题，该函数是一次性的。其它页面即便声明了一个同样的函数，也只是用于新的视图，丝毫不会对之前的视图造成任何影响。
> - layui.data.done 中的 *done* 可以随意命名，但需与 lay-done 的赋值对应上。

模板语法

动态模板基于 layui 的 laytpl 模块，详细语法可见：
http://www.layui.com/doc/modules/laytpl.html#syntax

登录与接口鉴权

由于 layuiAdmin 接管了视图层，所以不必避免可能会与服务端分开部署，这时你有必要了解一下 layuiAdmin 默认提供的：从 *登录* 到 *接口鉴权*，再到 *注销* 的整个流程。

登录拦截器

进入登入页面登入成功后，会在 localStorage 的本地表中写入一个字段。如：access_token（名称可以在 config.js 自定义）。拦截器判断没有 access_token 时，则会跳转到登入页。尽管可以通过伪造一个假的 access_token 绕过视图层的拦截，但在请求接口时，会自动带上 access_token，服务端应再次做一层校验。

流程

1. 打开 `config.js`，将 `interceptor` 参数设置为 `true`（该参数为 1.0.0-beta6 开始新增）。那么，当其未检查到 `access_token` 值时，会强制跳转到登录页面，以获取 access_token。
2. 打开登录对应的视图文件 `views/user/login.html`，在代码最下面，你将看到一段已经写好的代码，你需要的是将接口地址改为服务端的真实接口，并返回 `access_token` 值。
3. layuiAdmin 会将服务端返回的 `access_token` 值进行本地存储，这时你会发现 layuiAdmin 不再强制跳转到登录页面。并在后面每次请求服务端接口时，都会自动在参数和 Request Headers 中带上 `access_token`，以便服务端进行鉴权。
4. 若鉴权成功，顺利返回数据；若鉴权失败，服务端的 `code` 应返回 `1001`（可在 config.js 自定义），layuiAdmin 将会自动清空本地无效 token 并跳转到登入页。
5. 退出登录：重新打开 `controller/common.js`，搜索 `logout`，配上注销接口即可。

> 如果是在其它场景请求的接口（如：table.render()），那么你需要获取本地存储的 token 复制给接口参数，如：

```
table.render({
  elem: '#xxxx'
  ,url: 'url'
  ,where: {
    access_token: layui.data('layuiAdmin').access_token
  }
})
```

事实上，layuiAdmin 的所有 Ajax 请求都是采用 `admin.req(options)`，它会自动传递 `access_token`，因此推荐你在 JS 执行 Ajax 请求时直接使用它。其中参数 *options* 和 `$.ajax(options)` 的参数完全一样。

接口鉴权

我们推荐服务端遵循 **JWT**（JSON Web Token）标准进行鉴权。对 JWT 不甚了解的同学，可以去搜索一些相关资料，会极大地增加应用的可扩展性。当然，你也可以直接采用传统的 cookie / session 机制。

基础方法

- **config 模块**

> 你可以在任何地方通过 `layui.setter` 得到 *config.js* 中的配置信息

- **admin 模块**

> var admin = layui.admin;

- **admin.req(options)**
  Ajax 请求，用法同 $.ajax(options)，只是该方法会进行错误处理和 token 的自动传递
- **admin.screen()**
  获取屏幕类型，根据当前屏幕大小，返回 0 - 3 的值
  0: 低于768px的屏幕
  1：768px到992px之间的屏幕
  2：992px到1200px之间的屏幕
  3：高于1200px的屏幕
- **admin.exit()**
  清除本地 token，并跳转到登入页
- **admin.sideFlexible(status)**
  侧边伸缩。status 为 null：收缩；status为 “spread”：展开
- **admin.on(eventName, callback)**
  事件监听，下文会有讲解
- **admin.popup(options)**
  弹出一个 layuiAdmin 主题风格的 layer 层，参数 options 跟 layer.open(options) 完全相同
- **admin.popupRight(options)**
  在屏幕右侧呼出一个面板层。options 同上。

```
admin.popupRight({
  id: 'LAY-popup-right-new1' //定义唯一ID，防止重复弹出
  ,success: function(){
    //将 views 目录下的某视图文件内容渲染给该面板
    layui.view(this.id).render('视图文件所在路径');
  }
});
```

- **admin.resize(callback)**
  窗口 resize 事件处理，我们推荐你使用该方法取代 jQuery 的 resize 事件，以避免多页面标签下可能存在的冲突。
- **admin.events**
  - admin.events.refresh()
    刷新当前右侧区域
  - admin.events.closeThisTabs()
    关闭当前标签页
  - admin.events.closeOtherTabs()
    关闭其它标签页
  - admin.events.closeAllTabs()
    关闭全部标签页
- **view 模块**

> var view = layui.view;

- **view(id)**
  获取指定容器，并返回一些视图渲染的方法，如：

```
//渲染视图，viewPath 即为视图路径
view('id').render(viewPath).then(function(){
    //视图文件请求完毕，视图内容渲染前的回调
}).done(function(){
   //视图文件请求完毕和内容渲染完毕的回调
});

//直接向容器插入 html，tpl 为 模板字符；data 是传入的数据。该方法会自动完成动态模板解析
view('id').send(tpl, data);
```

另外，`render` 方法支持动态传参，以用于视图内容接受。如：

```
admin.popup({
  id: 'LAY-popup-test1'
  ,success: function(){
    view(this.id).render('视图文件所在路径', {
       id: 123 //这里的 id 值你可以在一些事件中动态获取（如 table 模块的编辑）
    });
  }
})
```

那么，在视图文件中，你可以在动态模板中通过 `{{ d.params.xxx }}` 得到传入的参数，如：

```
<script type="text/html" template lay-url="http://api.com?id={{ d.params.id }}">
  配置了接口的动态模板，且接口动态获取了 render 传入的参数：{{ d.params.id }}
</script>

<script type="text/html" template>
  也可以直接获取：<input type="hidden" name="id" value="{{ d.params.id }}">
</script>
```

驾驭好 `view().render().done(callback)` 对您的项目开发至关重要。

ID唯一性

如果你开启了标签页功能，请务必注意 ID 的冲突，尤其是在你自己绑定事件的情况。ID 的命令可以遵循以下规则来规避冲突：

```
LAY-路由-任意名
```

以 *消息中心* 页面为例，假设它的路由为：`/app/message/`，那么 ID 应该命名为：

```
<button class="layui-btn" id="LAY-app-message-del">删除</button>
```

实用组件

Hover 提示层

通过对元素设置 `lay-tips="提示内容"` 来开启一个 hover 提示，如：

```
<i class="layui-icon layui-icon-tips" lay-tips="要支持的喔" lay-offset="5"></i>
```

其中 `lay-offset` 用于定于水平偏移距离（单位px），以调整箭头让其对准元素

事件监听

- **hash**
  监听路由地址改变

```
// 下述中的 xxx 可随意定义，不可与已经定义的 hash 事件同名，否则会覆盖上一事件
admin.on('hash(xxx)', function(router){
  console.log(router); //得到路由信息
});
```

- **side**
  监听侧边伸缩

```
// 下述中的 xxx 可随意定义，不可与已经定义的 side 事件同名，否则会覆盖上一事件
admin.on('side(xxx)', function(obj){
  console.log(obj.status); //得到伸缩状态：spread 为展开状态，其它值为收缩状态
});
```

兼容性

layuiAdmin 使用到了 layui 的栅格系统，而栅格则是基于浏览器的媒体查询。ie8、9不支持。
所以要在宿主页面（如 start/index.html ）加上下面这段保证兼容：

```
<!-- 让IE8/9支持媒体查询，从而兼容栅格 -->
<!--[if lt IE 9]>
  <script src="https://cdn.staticfile.org/html5shiv/r29/html5.min.js"></script>
  <script src="https://cdn.staticfile.org/respond.js/1.4.2/respond.min.js"></script>
<![endif]-->
```

缓存问题

由于单页面版本的视图文件和静态资源模块都是动态加载的，所以可能存在浏览器的本地缓存问题，事实上我们也考虑到这个，因此，为了避免改动后的文件未及时生效，你只需在入口页面（默认为 `start/index.html`）中，找到 `layui.config`，修改其 `version` 的值即可。

> **我们推荐你分场景来更新缓存：**
> - 场景一：如果项目是在本地开发。你可以设置 version 为动态毫秒数，如：
>
> ```
> version: new Date().getTime() //这样你每次刷新页面，都会更新一次缓存
> ```
>
> - 场景二：如果项目是在线上运行。建议你手工更新 `version`，如：
>
> ```
> version: '1.0.0' //每次发布项目时，跟着改动下该属性值即可更新静态资源的缓存
> ```

升级事项

从官网更新资源包后，除了 src 和 dist 目录需要注意一下，其它目录和文件均可覆盖，下面以 src 为例（dist 由于是 src 构建后生成的目录，所以本质是和 src 一样的）
src 目录下可以直接覆盖的有：

> - src/lib/
> - src/style/
> - src/index.js

需要灵活调配的有：

> - src/controller/
> - src/views/
> - src/config.js
>   如果没有改动默认配置，事实上 config.js 也可以覆盖升级

开发过程中，建议同时运行两个 layuiAdmin 。一个是已经实际运用的，一个是 layuiAdmin 本身的 Demo。以便从 Demo 中获取参考和提取示例。

源码构建

> 当你在 src 目录完成开发后，你可通过 gulp 对 src 源码进行自动化构建，以生成用于线上环境的 dist 目录。并在入口页面中的 `layui.config` 的 base 参数指向 dist。

在资源包中根目录下看到的 gulpfile.js 是 layuiAdmin 写好的任务脚本，package.json 是任务配置文件，你只需按照以下步骤：
1. step1：确保你的电脑已经安装好了 Node.js，如果未安装，可去官网下载安装
2. step2: 命令行安装 gulp：npm install gulp -g
3. step3：切换到 layuiAdmin 项目根目录（即 gulpfile.js 所在目录），命令行安装任务所依赖的包：npm install

安装完成后，后续只需直接执行命令：`gulp` 即可完成 src 到 dist 目录的构建

关于版权

> layuiAdmin 受国家计算机软件著作权保护，未经官网正规渠道授权擅自公开产品源文件、以及直接对产品二次出售的，我们将追究相应的法律责任。

© 2018 layui.com 版权所有