

全局配置

方法：*layui.config(options)*

你可以在使用模块之前，全局化配置一些参数，尽管大部分时候它不是必须的。所以我们目前提供的全局配置项非常少，这也是为了减少一些不必要的工作，尽可能让使用变得更简单。目前支持的全局配置项如下：

定义模块

方法：*layui.define([mods], callback)*

通过该方法可定义一个 Layui*模块*。参数mods是可选的，用于声明该模块所依赖的模块。callback即为模块加载完毕的回调函数，它返回一个exports参数，用于输出该模块的接口。

1. layui.define(function(exports){
2. //do something
3. exports('demo', function(){
4. alert('Hello World!');
5. });
6. });

exports是一个函数，它接受两个参数，第一个参数为模块名，第二个参数为模块接口，当你声明了上述的一个模块后，你就可以在外部使用了，demo就会注册到layui对象下，即可通过 *layui.demo()* 去执行该模块的接口。

你也可以在定义一个模块的时候，声明该模块所需的依赖



上述的*['layer', 'laypage']*即为本模块所依赖的模块，它并非只能是一个数组，你也可以直接传一个字符型的模块名，但是这样只能依赖一个模块。

加载所需模块

方法：*layui.use([mods], callback)*

Layui的内置模块并非默认就加载的，他必须在你执行该方法后才会加载。它的参数跟上述的 define方法完全一样。   
另外请注意，mods里面必须是一个合法的模块名，不能包含目录。如果需要加载目录，建议采用extend建立别名（详见模块规范）

1. layui.use(['laypage', 'layedit'], function(laypage, layedit){
2. //使用分页
3. laypage();
4. //建立编辑器
5. layedit.build();
6. });

该方法的函数其实返回了所加载的模块接口，所以你其实也可以不通过layui对象赋值获得接口（这一点跟Sea.js很像哈）：

动态加载CSS

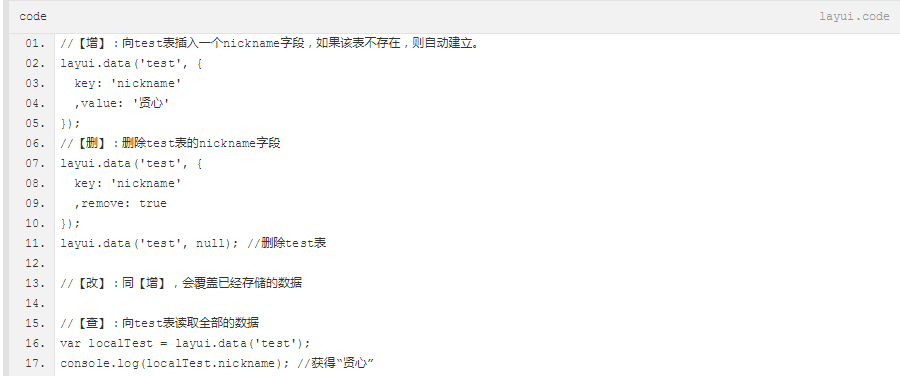
方法：*layui.link(href)*

href即为css路径。注意：该方法并非是你使用Layui所必须的，它一般只是用于动态加载你的外部CSS文件。

本地存储

方法：*layui.data(table, settings)*

参数*table*为表名，*settings*是一个对象，用于设置key、value。   
该方法对localStorage进行了良好的封装，在Layui的多个内置模块（比如layim）中发挥了重要的作用，因此你可以使用它存储一些本地数据。



获取设备信息

方法：*layui.device(key)*，参数key是可选的

由于Layui的一些功能进行了兼容性处理和响应式支持，因此该方法同样发挥了至关重要的作用。尤其是在layui mobile模块中的作用可谓举足轻重。该方法返回了丰富的设备信息：

你要验证当前的WebView是否在你的App环境，即可通过上述的*myapp*（即为Native给Webview插入的标识，可以随意定义）来判断。

### code[layui.code](http://www.layui.com/doc/modules/code.html" \t "http://www.layui.com/doc/base/_blank)

1. var device = layui.device('myapp');
2. if(device.myapp){
3. alert('在我的App环境');
4. }

其它

其实除此之外，layui.js内部还提供了许多底层引擎，他们同样是整个Layui框架体系的有力支撑，所以有必要露个脸，尽管你可能并不会用到：



第三方支撑

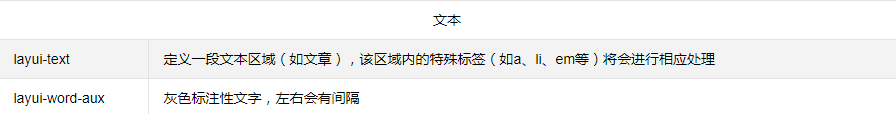
Layui**部分模块**依赖*jQuery*（比如layer），但是你并不用去额外加载jQuery。Layui已经将jQuery最稳定的一个版本改为Layui的内部模块，当你去使用 layer 之类的模块时，它会首先判断你的页面是否已经引入了jQuery，如果没有，则加载内部的jQuery模块，如果有，则不会加载。

# 页面元素规范与说明

Layui提倡返璞归真，遵循于原生态的元素书写规则，所以通常而言，你仍然是在写基本的HTML和CSS代码，不同的是，在HTML结构上及CSS定义上需要小小遵循一定的规范。

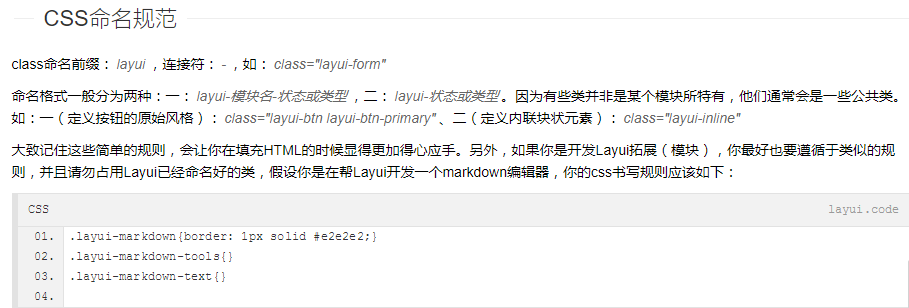






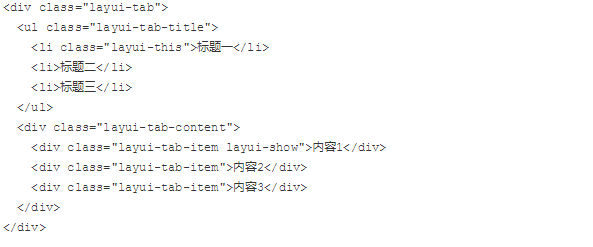


其它的类一般都是某个元素或模块所特有，因此不作为我们所定义的公共类。



HTML规范：结构

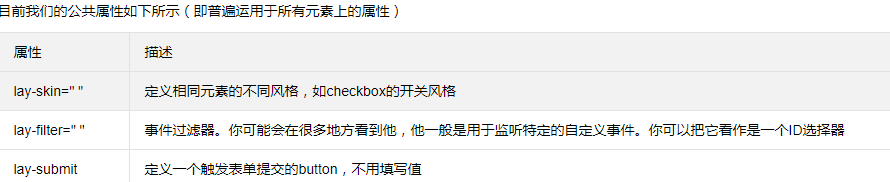
Layui在解析HTML元素时，必须充分确保其结构是被支持的。以Tab选项卡为例：



你如果改变了结构，极有可能会导致Tab功能失效。所以在嵌套HTML的时候，你应该细读各个元素模块的相关文档（如果你不是拿来主义）

HTML规范：常用公共属性

很多时候，元素的基本交互行为，都是由模块自动开启。但不同的区域可能需要触发不同的动作，这就需要你设定我们所支持的自定义属性来作为区分。如下面的 *lay-submit*、*lay-filter*即为公共属性（即以 *lay-* 作为前缀的自定义属性）：



# 模块规范

layui 的模块是基于 layui.js 内部实现的异步模块加载方式，它并不遵循于AMD（没有为什么，毕竟任性呀！），而是自己定义了一套更轻量的模块规范。并且这种方式在经过了大量的实践后，成为 layui 最核心的模块加载引擎。

预先加载

开门见山，还是直接说使用比较妥当。Layui的模块加载采用核心的 *layui.use(mods, callback)*方法，当你的JS 需要用到Layui模块的时候，我们更推荐你采用预先加载，因为这样可以避免到处写layui.use的麻烦。你应该在最外层如此定义：

模块命名空间

Layui的全部模块绑定在 layui对象下，内部由layui.define()方法来完成。每一个模块都会一个特有的名字，并且无法被占用。所以你无需担心模块的空间被污染，除非是你 delete layui.mod; 调用一个模块也必须借助layui对象的赋值。如：

一个模块一旦加载后，就会注册在layui对象下，所以你无法直接用模块名来获得，而同样借助layui对象。譬如有时你可能会直接在元素的事件属性上去调用一个模块，如：



扩展一个Layui模块

第一步：确认模块名，假设为：*test*，然后新建一个*test.js* 文件放入项目任意目录下（注意：不用放入layui目录）

第二步：编写test.js 如下：

第三步：设定扩展模块所在的目录，然后就可以在别的JS文件中使用了

其实关于模块的核心，就是layui.js的两个底层方法：一个用于定义模块的*layui.define()*，一个加载模块的*layui.use()*。

如何使用内部jQuery？

由于Layui部分内置模块依赖jQuery，所以我们将jQuery1.11最稳定的一个版本作为一个内置的DOM模块（唯一的一个第三方模块）。只有你所使用的模块有依赖到它，它才会加载，并且如果你的页面已经script引入了jquery，它并不会重复加载。内置的jquery模块去除了全局的$和jQuery，是一个符合layui规范的标准模块。所以你必须通过以下方式得到：



为什么表单不显示？

当你使用表单时，Layui会对select、checkbox、radio等原始元素隐藏，从而进行美化修饰处理。但这需要依赖于form组件，所以你必须加载 *form*，并且执行一个实例。值得注意的是：导航的Hover效果、Tab选项卡等同理（它们需依赖 *element* 模块）

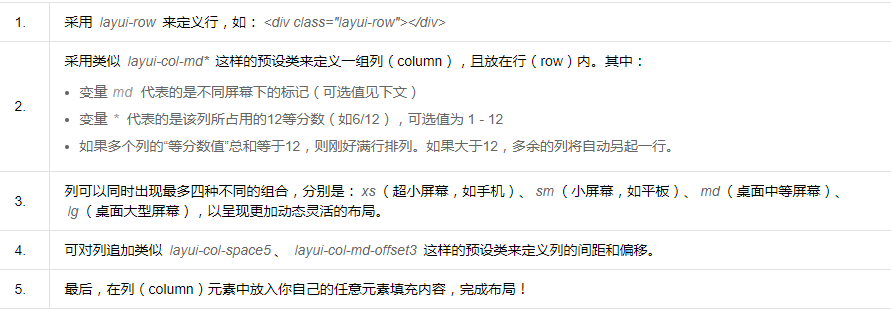
1. layui.use('form', function(){
2. var form = layui.form; //只有执行了这一步，部分表单元素才会自动修饰成功
3. //但是，如果你的HTML是动态生成的，自动渲染就会失效
4. //因此你需要在相应的地方，执行下述方法来手动渲染，跟这类似的还有 element.init();
5. form.render();

<http://fly.layui.com/jie/5366/> 社区地址

# 栅格系统与后台布局

为了丰富网页布局，简化 HTML/CSS 代码的耦合，并提升多终端的适配能力，layui 在 2.0 的版本中引进了自己的一套具备响应式能力的栅格系统。我们将容器进行了 12 等分，预设了 4\*12 种CSS排列类，它们在移动设备、平板、桌面中/大尺寸四种不同的屏幕下发挥着各自的作用。

1. 栅格布局规则：



<http://www.layui.com/demo/grid.html> 案例查看

二、响应式规则：

栅格的响应式能力，得益于CSS3媒体查询（Media Queries）的强力支持，从而针对四类不同尺寸的屏幕，进行相应的适配处理



1. 响应式公共类：



四．布局容器：

将栅格放入一个带有 *class="layui-container"* 的特定的容器中，以便在小屏幕以上的设备中固定宽度，让列可控。

1. <div class="layui-container">
2. <div class="layui-row">
3. ……
4. </div>
5. </div>

当然，你还可以不固定容器宽度。将栅格或其它元素放入一个带有 *class="layui-fluid"* 的容器中，那么宽度将不会固定，而是 100% 适应

1. <div class="layui-fluid">
2. ……
3. </div>
4. 列间距：

|  |
| --- |
| 通过“列间距”的预设类，来设定列之间的间距。且一行中最左的列不会出现左边距，最右的列不会出现右边距。列间距在保证排版美观的同时，还可以进一步保证分列的宽度精细程度。我们结合网页常用的边距，预设了 12 种不同尺寸的边距，分别是： |

1. 列偏移：

对列追加 类似 *layui-col-md-offset\** 的预设类，从而让列向右偏移。其中 *\** 号代表的是偏移占据的列数，可选中为 1 - 12。   
如：*layui-col-md-offset3*，即代表在“中型桌面屏幕”下，让该列向右偏移3个列宽度

请注意，列偏移可针对不同屏幕的标准进行设定，比如上述的例子，只会在桌面屏幕下有效，当低于桌面屏幕的规定的临界值，就会堆叠

1. 栅格嵌套：

理论上，你可以对栅格进行无穷层次的嵌套，这更加增强了栅格的表现能力。而嵌套的使用非常简单。在列元素（*layui-col-md\**）中插入一个行元素（*layui-row*），即可完成嵌套。下面是一个简单的例子：

1. 让IE8/9兼容栅格：

事实上IE8和IE9并不支持媒体查询（Media Queries），但你可以使用下面的补丁完美兼容！该补丁来自于开源社区：

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

后台布局

layui 之所以赢得如此多人的青睐，更多是在于它前后台系统通吃的能力。既可编织出绚丽的前台页面，又可满足繁杂的后台功能需求。layui 致力于让每一位开发者都能轻松搭建自己的后台。下面是 layui 提供的一个现场的方案，你可以前往示例页面

http://www.layui.com/demo/layuiAdmin.html

# layui 颜色设计感

常用主色

 通常用于按钮、及任何修饰元素

 一般用于选中状态

 通常用于导航

 比较适合一些鲜艳色系的页面

layui 主要是以象征包容的墨绿作为主色调，由于它给人以深沉感，所以通常会以浅黑色的作为其陪衬，又会以蓝色这种比较鲜艳的色调来弥补它的色觉疲劳，整体让人清新自然，愈发耐看。【取色意义】：我们执着于务实，不盲目攀比，又始终不忘绽放活力。这正是 layui 所追求的价值

场景色

暖色系，一般用于提示性元素

比较引人注意的颜色

 用于文字着色，如链接文本

 侧边或底部普遍采用的颜色

极简中性色



layui 认为灰色系代表极简，因为这是一种神奇的颜色，几乎可以与任何元素搭配，不易形成视觉疲劳，且永远不会过时。低调而优雅！

内置的背景色CSS类

layui 内置了七种背景色，以便你用于各种元素中，如：徽章、分割线、导航等等



 “不热衷于视觉设计的程序猿不是一个好作家！” ——贤心

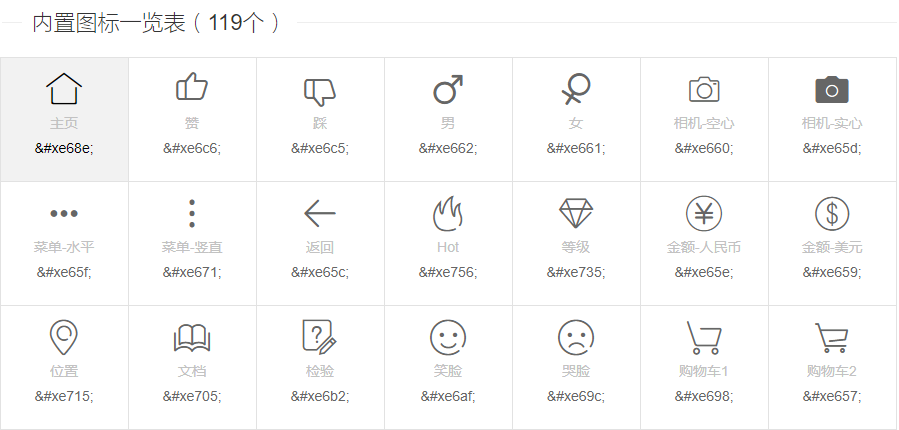
# 字体图标

layui 的所有图标全部采用字体形式，取材于阿里巴巴矢量图标库（iconfont）。因此你可以把一个icon看作是一个普通的文字，这意味着你直接用css控制文字属性，如color、font-size，就可以改变图标的颜色和大小。而区分不同的图标，我们主要是采用 *Unicode* 字符

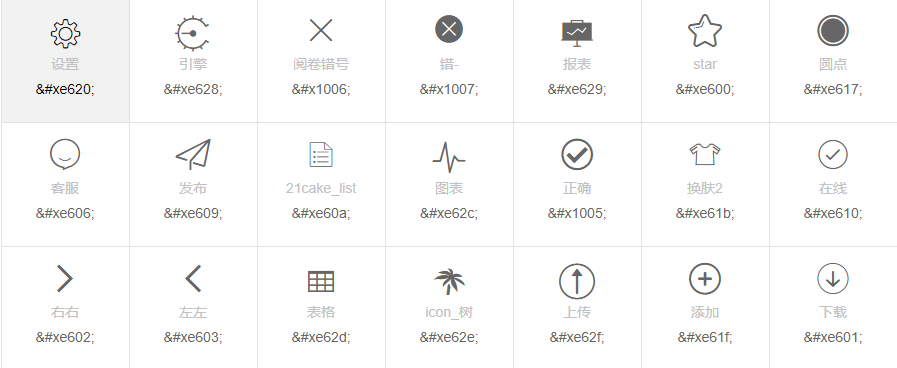
使用方式

通过对一个内联元素（一般推荐用 *i*标签）设定 *class="layui-icon"*，来定义一个图标，然后对元素加上图标对应的 *Unicode* 字符，即可显示出你想要的图标，譬如：

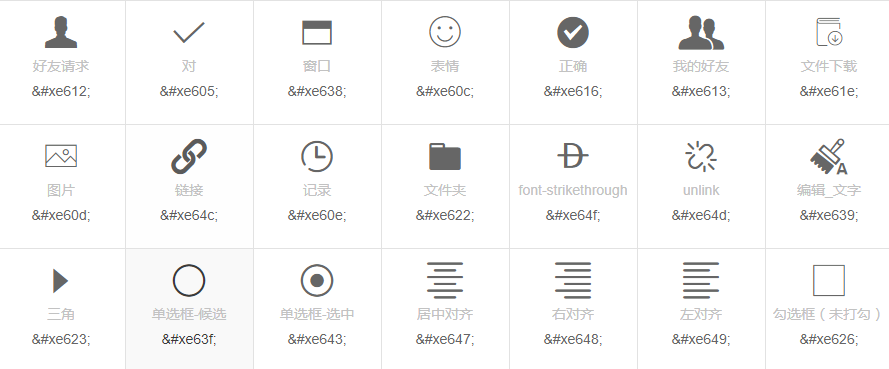
### 









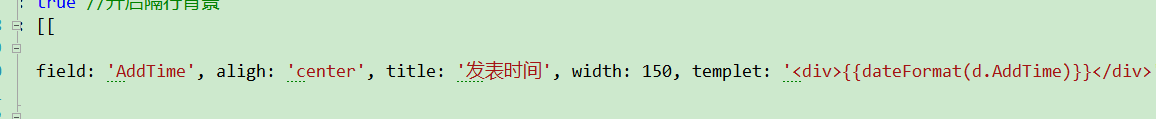




跨域问题的解决

由于浏览器存在同源策略，所以如果layui（里面含图标字体文件）所在的地址与你当前的页面地址*不在同一个域下*，即会出现图标跨域问题。所以要么你就把Layui与网站放在同一服务器，要么就对Layui所在的资源服务器的Response Headers加上属性：*Access-Control-Allow-Origin: \**

模板格式化时间



layer.closeAll(); //疯狂模式，关闭所有层

layer.closeAll('dialog'); //关闭信息框

layer.closeAll('page'); //关闭所有页面层

layer.closeAll('iframe'); //关闭所有的iframe层

layer.closeAll('loading'); //关闭加载层

layer.closeAll('tips'); //关闭所有的tips层

var index = parent.layer.getFrameIndex(window.name); //获取窗口索引

//parent.layer.closeAll(index);

parent.layer.closeAll();

index.gettext可以富文本判断为空的