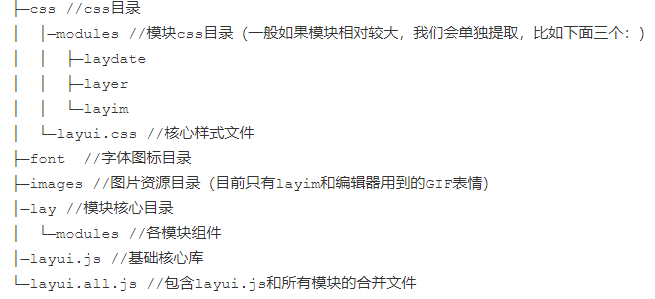
# Layui

## 1、简介：

是一款国人开发采用自身模块规范编写的前端 UI 框架元素操作在element需要那个引入哪个，门槛极低。外在极简，体积轻盈，但是组件丰盈。

把https://www.layui.com/doc/全看完



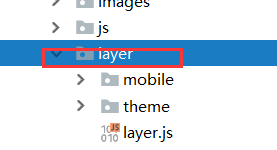
## 2、使用方式

1、 把整个layui放到static里 引入js与cs即可

|  |
| --- |
| <**link rel="stylesheet" href="layui/css/layui.css"**> <**script src="layui/layui.js"**></**script**> |

## 3、内置模块layer

1、可以下载一个layer把整个包导入项目



然后引入两个js，layerjs依赖jquery

|  |
| --- |
| <**script src="/js/jquery-1.12.4.js"**></**script**> <**script src="layer/layer.js"**></**script**> |

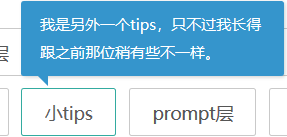
1. 或者直接导入layui的js和css直接加载layer模块

|  |
| --- |
| layui.use('layer', function(){  var layer = layui.layer; //写一个laydate变量用于操作  }); |

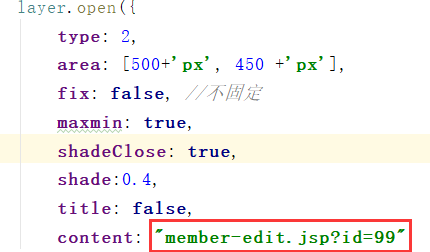
3、然后再官网找代码直接引入即可 第一种直接copy到script 标签里，第二种copy到layui.use里

|  |
| --- |
| **layer**.confirm(**'您是如何看待前端开发？'**, {  **btn**: [**'重要'**,**'奇葩'**] *//按钮* }, **function**(){  **layer**.msg(**'的确很重要'**, {**icon**: 1}); }, **function**(){  **layer**.msg(**'也可以这样'**, {  **time**: 20000, *//20s后自动关闭* **btn**: [**'明白了'**, **'知道了'**]  }); }); |

4、还有这种效果的 必须设置吸附于哪个元素



|  |
| --- |
| 1. layer.tips('我是另外一个tips，只不过我长得跟之前那位稍有些不一样。', '吸附元素选择器(#aa)', { 2. tips: [1, '#3595CC'], 3. time: 4000 4. }); |



编辑的话layer有iframe弹框，只需要连接一个修改页传入id即可，那边用${param.id}来接收这个还有end:用于关闭弹框后执行的方法，还有success

## 引入内置jquery方法



但是如果你的页面已经script引入了jquery，它并不会重复加载

## 注意\*\*\*

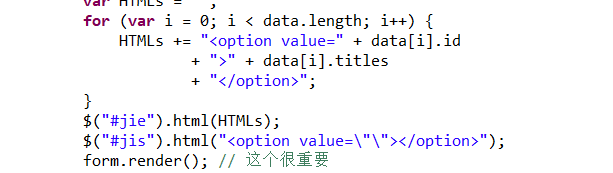
### 1、原始元素隐藏

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

|  |
| --- |
| 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. form.render(); 就加到这即可 4. //…… 5. //但是，如果你的HTML是动态生成的，自动渲染就会失效 6. //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(); 7. form.render(); 8. }); |

加到use不管用那么就加到动态生成代码后面试试

|  |
| --- |
| **for** (**var *i*** = 0; ***i*** <***data***.**length** ; ***i***++) {  *$*(**"#firstCategory"**).append(**"<option value="** + ***data***[***i***].**id** + **">"**+***data***[***i***].**value**+**"</option>"**) } **form**.render(); |



应该是加到动态生成代码后

### 2、为表单提供了一些属性



第一个为让多选框变成开关形式

第二个类似于加个idclass为layui里的样式，用于事件监听

第三个不用解释了吧

第四个如果监听某个标签比如select为form.on('select(filter)')，而监听表单提交的时候就

<button lay-submit lay-filter="\*">提交</button> 监听就可以这么写了 submit(\*)

### 3、监听元素

如果只需要监听某一个元素，使用过滤器即可 形式都如下 *<select lay-filter="test"></select>*

|  |
| --- |
| 1. form.on('select(test)', function(data){ 2. console.log(data); 3. }); |



1、监听select选择

1. form.on('select(filter)', function(data){
2. console.log(data.elem); //得到select原始DOM对象
3. console.log(data.value); //得到被选中的值
4. console.log(data.othis); //得到美化后的DOM对象
5. });

一般只获取value即可

2、监听checkbox多选 一般只要data.**elem**.**checked即可**

3、监听switch开关(多选框变成开关形式) 一般只要data.**elem**.**checked即可**

1. 监听redio 一般只要data.**value即可(男代表0女代表1)**

**5、监听submit**

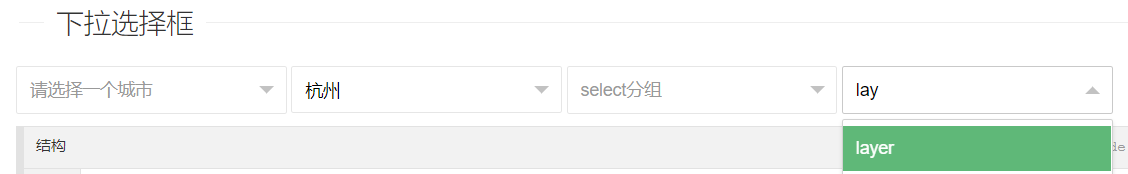
|  |
| --- |
| 1. form.on('submit(\*)', function(data){ 2. console.log(data.field) //一般用这个即可，名值对形式：{name: value}所以?sex=data.field.sex 3. return false; //阻止表单跳转。如果需要表单跳转，去掉这段即可。 4. }); 5. <button lay-submit lay-filter="\*">提交</button> |

如果你想监听所有的select，去掉过滤器*(filter)*即可。每个监听都一个意思

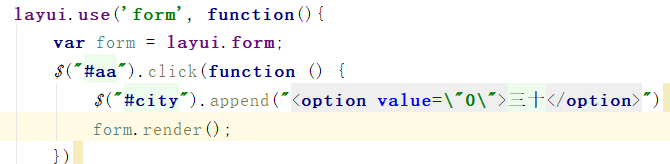
## 5、表单\*\*\*

表单直接copy即可 关键是script代码来拿到数据

|  |
| --- |
| **layui**.use(**'form'**, **function**(){  **var** form = **layui**.**form**;   *//监听提交* form.on(**'submit(formDemo)'**, **function**(data){  *alert*(***JSON***.stringify(data.**field**))  **layer**.msg(***JSON***.stringify(data.**field**));  **return false**;  }); }); |



有一个带搜索的下拉框



可以监听各种然后form.rander()动态刷新数据即可

<**input type="checkbox" value="asd" title="锁定" checked** >

<**input type="checkbox" checked lay-skin="switch" lay-filter="switchTest" lay-text="可用|已锁定" value={{ d.id }}**>



$(data.othis).find(".layui-unselect").val() 能直接转换成jquery对象来获取里面的值(得到美化后的DOM对象)

## 6、静态表格直接copy即可\*\*\*



可以再td标签里加按钮编辑删除什么的

## 分页

|  |
| --- |
| <**div id="test1"**></**div**> <**div id="test2"**></**div**> <**div id="aa"**></**div**> *<!--测试jump的obj参数-->* <**script**>  **var *data*** = [  **'北京'**,  **'上海'**,  **'广州'**,  **'深圳'**,  **'杭州'**,  **'长沙'**,  **'合肥'**,  **'宁夏'**,  **'成都'**,  **'西安'**,  **'南昌'**,  **'上饶'**,  **'沈阳'**,  **'济南'**,  **'厦门'**,  **'福州'**,  **'九江'**,  **'宜春'**,  **'赣州'**,  **'宁波'**,  **'绍兴'**,  **'无锡'**,  **'苏州'**,  **'徐州'**,  **'东莞'**,  **'佛山'**,  **'中山'**,  **'成都'**,  **'武汉'**,  **'青岛'**,  **'天津'**,  **'重庆'**,  **'南京'**,  **'九江'**,  **'香港'**,  **'澳门'**,  **'台北'** ];  **layui**.use(**'laypage'**, **function**(){  **var** laypage = **layui**.**laypage**;   laypage.render({  **elem**: **'test1'** *//注意，这里的 test1 是 ID，不用加 # 号* ,**count**: ***data***.**length** *//数据总数，从服务端得到* ,jump: **function**(obj){  *//模拟渲染* **var** arr = []  ,thisData = ***data***.concat().splice(obj.**curr**\*obj.**limit** - obj.**limit**, obj.**limit**);  **layui**.each(thisData, **function**(index, item){  arr.push(**'<li>'**+ item +**'</li>'**);  });  *$*(**"#aa"**).html(arr)  *$*(**"#test2"**).html(obj.**curr**)  *//如果为第一次进来就会执行，一般用于初始加载的判断* **if**(first){  *alert*(**"s"**) }  }  ,**first**: **'首页'** *//往后很多页这个才会显示  //显示 的位置 每页页数、一共多少条数据，刷新按钮，选择页数来跳转* ,**layout**: [**'next'**, **'page'**, **'prev'**,**'limit'**,**'count'**,**'refresh'**,**'skip'**]  ,**limit** : 3 *//每页显示三条* ,**limits**: [10, 20, 30, 40, 50] *//选择每页显示多少条数据* ,**theme**: **'#c00'** *//变成红色*  ,**groups**: 2 *//只显示 2 个连续页码*  });  }); </**script**> |

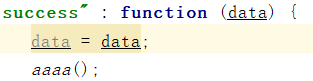
### 真实情况只是显示数据下

可以先进入controller返回类似模拟data的样子(request.set返回test.jsp),页面上直接var data = ${Categorys}即可



### 会有各种筛选显示数据就

1. 把render封装到一个方法里内容都跟(1里面只是显示数据)一样，页面执行的时候先调用aaaa()方法
2. 然后筛选就写一个ajax方法，进入control的一个responsbody方法，成功之后把值赋给data然后再重新重新执行一下aaaa方法即可重新渲染



还有一个类似的素材在案例项目 前端里面

## 表格

### 最普通的

返回值需要code为0，格式为{"code":0,”data”: [{},{},{}] },但是分页不管用，分页的页数是由传过来的count来判断的

|  |
| --- |
| <**table id="demo" lay-filter="test"**></**table**> <**script**>  **layui**.**use**(**'table'**, **function**(){  **var** table = **layui**.**table**;   *//第一个实例* table.render({  **elem**: **'#demo'** ,**height**: 312  ,**url**: **'category1'** *//数据接口* ,**page**: **true** *//开启分页* ,**cols**: [[ *//表头* {**field**: **'id'**, **title**: **'ID'**, **width**:80, **sort**: **true**, **fixed**: **'left'**}  ,{**field**: **'parentId'**, **title**: **'父级id'**, **width**:80}  ,{**field**: **'value'**, **title**: **'值'**, **width**:80, **sort**: **true**}  ,{**field**: **'parentname'**, **title**: **'父级名字'**, **width**:80}  ]]  });   }); </**script**>  @ResponseBody @RequestMapping(**"category1"**) **public** Object category1(String parentId, HttpServletRequest request){  **if** (parentId == **null**){  parentId = **"1"**;  }  HashMap<String,Object> hashMap = **new** HashMap<String,Object>();  hashMap.put(**"code"**,0);  hashMap.put(**"msg"**,**""**);  hashMap.put(**"count"**,1000);  hashMap.put(**"data"**,**categorysService**.parentIdSelectCategorysList(parentId));  **return** JSON.*toJSONString*(hashMap); } |

### 2、另一种方式渲染

|  |
| --- |
| <**table class="layui-table" lay-data="{height:315, url:'category1', page:true, id:'test1'}" lay-filter="test1"**>  <**thead**>  <**tr**>  <**th lay-data="{field:'id', width:150, sort: true}"**>ID</**th**>  <**th lay-data="{field:'parentId', width:150}"**>父级id</**th**>  <**th lay-data="{field:'value', width:150, sort: true}"**>值</**th**>  <**th lay-data="{field:'parentname'}"**>父级名字</**th**>  </**tr**>  </**thead**> </**table**> |

一个更专注js文件，一个专注html表头

### 3、page分页属性：

|  |
| --- |
| ,**page**: { *//支持传入 laypage 组件的所有参数（某些参数除外，如：jump/elem）* **layout**: [ **'limit'**,**'prev'**, **'page'**, **'next'**, **'skip'**] *//自定义分页布局* ,**groups**: 2 *//只显示 2 个连续页码* ,**limit**: 3 *//每页三条数据* } |

//可以直接写true有个默认的 调整limit只对分页那出现几个页数有改变，数据不会变

### 4、data属性对已知数据进行展示(就不需要写url属性了)

分页会成功

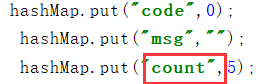
|  |
| --- |
| ,data: [{  "id": "10001"  ,"username": "杜甫"  ,"email": "xianxin@layui.com"  ,"sex": "男"  ,"city": "浙江杭州"  ,"sign": "人生恰似一场修行"  ,"experience": "116"  ,"ip": "192.168.0.8"  ,"logins": "108"  ,"joinTime": "2016-10-14"  }, {  "id": "10002"  ,"username": "李白"  ,"email": "xianxin@layui.com"  ,"sex": "男"  ,"city": "浙江杭州"  ,"sign": "人生恰似一场修行"  ,"experience": "12"  ,"ip": "192.168.0.8"  ,"logins": "106"  ,"joinTime": "2016-10-14"  ,"LAY\_CHECKED": true  }] |

### done属性回调

无论是异步请求数据，还是直接赋值数据，都会触发该回调。你可以利用该回调做一些表格以外元素的渲染。

|  |
| --- |
| ,done: function(res, curr, count){  //如果是异步请求数据方式，res即为你接口返回的信息。  //如果是直接赋值的方式，res即为：{data: [], count: 99} data为当前页数据、count为数据总长度  alert(res.data[0].id)  //得到当前页码  console.log(curr);  //得到数据总量  console.log(count);  } |

Res.data为当页数据 curr为当前页数 count为服务层传过来的count



 一般只要动了这些，那么这个回调函数就会执行

### 6、totalRow:true 开启合计行

开启后 在cols属性里也写true就计算了



### 7、toolbar:标头工具栏属性

toolbar: '#toolbarDemo' //开启工具栏而且左侧显示自定义工具栏

toolbar: true //仅开启工具栏，不显示左侧模板

toolbar: 'default' //开启工具栏而且左侧显示默认的内置模板

工具栏就是“列显示隐藏”、“导出”、“打印”等功能，属性为第一个的话就是绑定自定义额外的功能

1、属性为'#toolbarDemo'

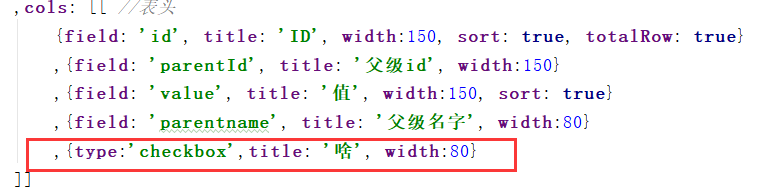
|  |
| --- |
| <**script type="text/html" id="toolbarDemo"**>  <**div class="layui-btn-container"**>  <**button class="layui-btn layui-btn-sm" lay-event="getCheckData"**>获取选中行数据</**button**>  <**button class="layui-btn layui-btn-sm" lay-event="getCheckLength"**>获取选中数目</**button**>  <**button class="layui-btn layui-btn-sm" lay-event="isAll"**>验证是否全选</**button**>  </**div**> </**script**> |

这样就头部会显示了那三个功能了

2、<**table id="demo" lay-filter="test"**></**table**> 表格为这个所以现在需要监控表格filter属性为test里面的toolbar

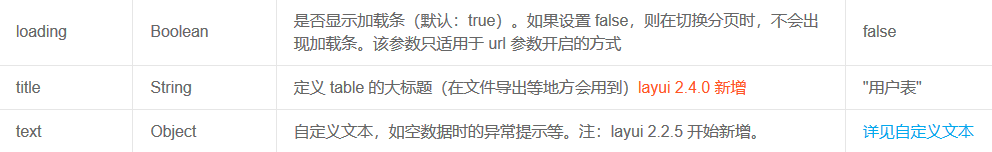
|  |
| --- |
| table.on(**'toolbar(test)'**, **function**(obj){  **var** checkStatus = table.checkStatus(obj.**config**.**id**); *//获取表格选中行*  **switch**(obj.**event**){  **case 'getCheckData'**:  **var** data = checkStatus.**data**;  **layer**.alert(***JSON***.stringify(data));  **break**;  **case 'getCheckLength'**:  **var** data = checkStatus.**data**;  **layer**.msg(**'选中了：'**+ data.**length** + **' 个'**);  **break**;  **case 'isAll'**:  **layer**.msg(checkStatus.**isAll** ? **'全选'**: **'未全选'**);  **break**;  }; }); |

必须有一栏为多选



### 8、其他属性

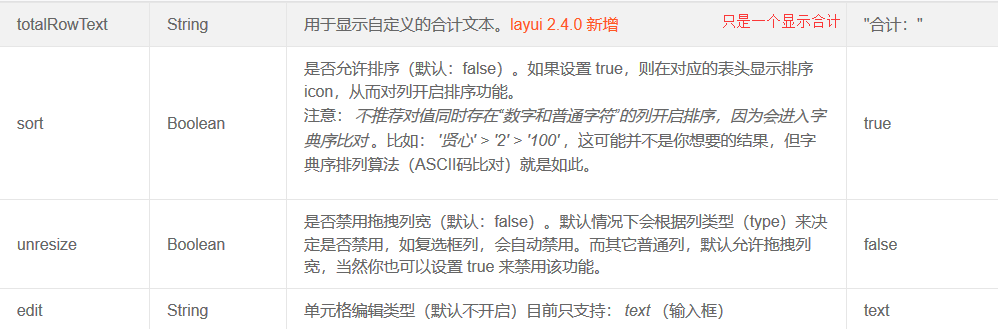






### 9、Cols的参数







### 10、event:"id" 属性 单元格点击事件

|  |  |
| --- | --- |
| 自定义单元格点击事件名，以便在 [tool](https://www.layui.com/doc/modules/table.html" \l "ontool) 事件中完成对该单元格的业务处理 |  |

|  |
| --- |
| table.on(**'tool(test)'**, **function**(obj){ *//注：tool是工具条事件名，test是table原始容器的属性 lay-filter="对应的值"* **var** layEvent = obj.**event**; *//获得 lay-event 对应的值（也可以是表头的 event 参数对应的值）  alert*(layEvent) }); |

**给哪个属性加那么那一列一点击就会触发这个方法***alert*(layEvent)

### initSort - 初始排序

用于在数据表格渲染完毕时，默认按某个字段排序

|  |
| --- |
| ,**initSort**: {  **field**: **'id'** *//排序字段，对应 cols 设定的各字段名* ,**type**: **'desc'** *//排序方式 asc: 升序、desc: 降序、null: 默认排序* } |

### colspan 与 rowspan案例

|  |
| --- |
| ,**cols**: [[  ,{**type**:**'space'**,**title**: **'分类信息'**, **align**: **'center'**,**colspan**: 4} ],[  {**field**: **'id'**, **title**: **'ID'**, **width**:150, **sort**: **true**,**totalRow**: **true**}  ,{**field**: **'parentId'**, **title**: **'父级id'**, **width**:150}  ,{**field**: **'value'**, **title**: **'值'**, **width**:150, **sort**: **true**}  ,{**field**: **'parentname'**, **title**: **'父级名字'**, **width**:80} ]] |

### 12、templet - 自定义列模板

在默认情况下，单元格的内容是完全按照数据接口返回的content原样输出的，如果你想对某列的单元格添加链接等其它元素，或是根据值做一些条件来判断显示什么可以用自定义模板，例如如果是1那就选中是2不选中

|  |
| --- |
| <**script type="text/html" id="titleTpl"**>  {{# if(d.id < 100){ }}  <**a href="/detail/{{d.id}}" class="layui-table-link"**>是</**a**>  {{# } else { }}  <**a href="/detail/{{d.id}}" class="layui-table-link"**>否</**a**>  {{# } }} </**script**>  ,{**field**: **'parentId'**, **title**: **'是否成功'**, **width**:150,**templet**: **'#titleTpl'**} |

### 13、绑定列工具条

表格的每一行加上 查看、编辑、删除 这样类似的操作按钮，而 tool 参数就是为此而生

|  |
| --- |
| ,{**fixed**: **'right'**, **width**:150, **align**:**'center'**,**title**:**"操作"**, **toolbar**: **'#barDemo'**}  <**script type="text/html" id="barDemo"**>  <**a class="layui-btn layui-btn-xs" lay-event="detail"**>查看</**a**>  <**a class="layui-btn layui-btn-xs" lay-event="edit"**>编辑</**a**>  <**a class="layui-btn layui-btn-danger layui-btn-xs" lay-event="del"**>删除</**a**>   *<!-- 这里同样支持 laytpl 语法，如： -->* {{# if(d.auth > 2){ }}  <**a class="layui-btn layui-btn-xs" lay-event="check"**>审核</**a**>  {{# } }} </**script**> |

|  |
| --- |
| *//监听工具条* table.on(**'tool(test)'**, **function**(obj){ *//注：tool是工具条事件名，test是table原始容器的属性 lay-filter="对应的值"* **var** data = obj.**data**; *//获得当前行数据* **var** layEvent = obj.**event**; *//获得 lay-event 对应的值（也可以是表头的 event 参数对应的值）* **var** tr = obj.**tr**; *//获得当前行 tr 的DOM对象* **if**(layEvent === **'detail'**){ *//查看  //do somehing* } **else if**(layEvent === **'del'**){ *//删除* **layer**.confirm(**'真的删除行么'**, **function**(index){  obj.del(); *//删除对应行（tr）的DOM结构，并更新缓存* **layer**.close(index);  *//向服务端发送删除指令* });  } **else if**(layEvent === **'edit'**){ *//编辑  //do something   //同步更新缓存对应的值* obj.update({  **id**: **'123'** ,**title**: **'xxx'** });  } }); |

表头工具栏指定的区域里面按钮有lay-event 然后toolbar(test)监听

单元格单机事件给列加event属性然后tool(test)监听

工具条指定的区域里面按钮有lay-event tool(test)来监听

tool(test)监听工具条和单元格事件

*alert*(layEvent)  
*alert*(***JSON***.stringify(data))  
*alert*(tr.html())

每个里面都可以输出这些信息

### 异步数据接口

异步请求参数的属性有：

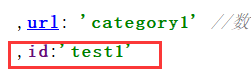
|  |  |  |
| --- | --- | --- |
|  | Url地址。默认会自动传递两个参数：?page=1&limit=30 page 代表当前页码、limit 代表每页数据量 contr直接getParame即可   |  | | --- | | table.render({ *//其它参数在此省略* **url**: **'/api/data/'** *//where: {token: 'sasasas', id: 123} //如果无需传递额外参数，可不加该参数  //method: 'post' //如果无需自定义HTTP类型，可不加该参数  //request: {} //如果无需自定义请求参数，可不加该参数  //response: {} //如果无需自定义数据响应名称，可不加该参数* }); | |

如果返回的数据名字格式不一样可以使用下列方法



### 表格重载

很多时候，你需要对表格进行重载。比如数据全局搜索。 只要不设置的属性都是按照原来的参数来请求的

Test1为表格加载设置的属性 ，

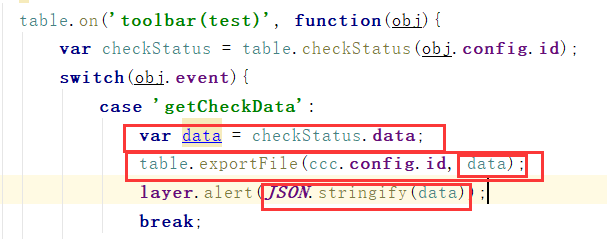
|  |
| --- |
| *$*(**"#aa"**).click(**function** () {  table.reload(**'test1'**, {  **url**: **'category1'** ,**where**: {**parentId** : **'0'**} *//设定异步数据接口的额外参数* ,**page**: {  **curr**: 1 *//重新从第 1 页开始* }  }); }) |

### 导出任意数据

尽管 table 的工具栏内置了数据导出按钮，但有时你可能需要通过方法去导出任意数据，那么可以借助以下方法：

1. 依赖table实例的情况下

|  |
| --- |
| **var** ins1 = table.render({  **elem**: **'#demo'** ,**id**: **'test'** *//,…… //其它参数* })  *//将上述表格示例导出为 csv 文件* table.exportFile(ins1.**config**.**id**, ***data***); |

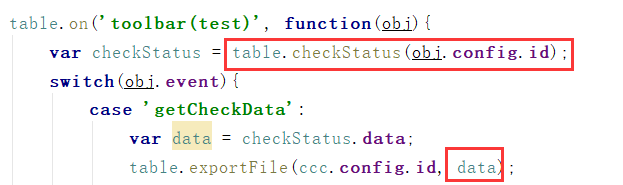


可以放到自定义标题栏上获取选中的信息后直接传入data(是对象如果传入的为下方那个json方法处理过的就不行)

1. 不依赖table实例的情况下

|  |
| --- |
| table.exportFile([**'名字'**,**'性别'**,**'年龄'**], [  [**'张三'**,**'男'**,**'20'**],  [**'李四'**,**'女'**,**'18'**],  [**'王五'**,**'女'**,**'19'**] ], **'csv'**); *//默认导出 csv，也可以为：xls* |

放到上方标头工具栏直接 获取选中的



放到下方有obj.data获取当前行数据 获取当前行



把data弄成[{"id":5,"parentId":0,"parentname":"招标信息","value":"交通运输"}]类型的

### 监听复选框 得用from方式的监听

|  |
| --- |
| form.on(**'checkbox(lockDemo)'**, **function**(obj){  *alert*(obj.**elem**.**checked**); *//当前是否选中状态* });  <**script type="text/html" id="titleTpl"**>  {{# if(d.id < 7){ }}  <**input type="checkbox" name="lock" value="{{d.id}}" title="锁定" lay-filter="lockDemo" checked**>  {{# } else { }}  <**input type="checkbox" name="lock" value="{{d.id}}" title="锁定" lay-filter="lockDemo"** >  {{# } }} </**script**> |

### 18、监听edit

|  |
| --- |
| table.on(**'edit(test)'**, **function**(obj){ *//注：edit是固定事件名，test是table原始容器的属性 lay-filter="对应的值"  alert*(obj.**value**); *//得到修改后的值  alert*(obj.**field**); *//当前编辑的字段名  alert*(***JSON***.stringify(obj.**data**)); *//所在行的所有相关数据* }); |

1. 监听行点击事件

|  |
| --- |
| *//监听行单击事件* table.on(**'row(test)'**, **function**(obj){ *test是table原始容器的属性 lay-filter="对应的值"*  *alert*(obj.**tr**) *//得到当前行元素对象  alert*(***JSON***.stringify(obj.**data**)) *//得到当前行数据* **var** aa = obj.**data**;  aa.**id**=4;  *//obj.del(); //删除当前行  //obj.update(aa) //修改当前行数据* });  双击把row换成rowDouble |

### 监听排序切换

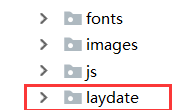
|  |
| --- |
| *//监听排序事件* table.on(**'sort(test)'**, **function**(obj){ *//注：tool是工具条事件名，test是table原始容器的属性 lay-filter="对应的值"* ***console***.log(obj.**field**); *//当前排序的字段名* ***console***.log(obj.**type**); *//当前排序类型：desc（降序）、asc（升序）、null（空对象，默认排序）* ***console***.log(**this**); *//当前排序的 th 对象   //尽管我们的 table 自带排序功能，但并没有请求服务端。  //有些时候，你可能需要根据当前排序的字段，重新向服务端发送请求，从而实现服务端排序，如：* table.reload(**'idTest'**, {  **initSort**: obj *//记录初始排序，如果不设的话，将无法标记表头的排序状态。* ,**where**: { *//请求参数（注意：这里面的参数可任意定义，并非下面固定的格式）* **field**: obj.**field** *//排序字段* ,**order**: obj.**type** *//排序方式* }  });   **layer**.msg(**'服务端排序。order by '**+ obj.**field** + **' '** + obj.**type**);  }); |

## 轮播

|  |
| --- |
| <**div class="layui-carousel" id="test1"**>  <**div carousel-item**>  <**div style="background-color**: **black"**>条目1</**div**>  <**div**>条目2</**div**>  <**div**>条目3</**div**>  <**div**>条目4</**div**>  <**div**>条目5</**div**>  </**div**> </**div**> *<!-- 条目中可以是任意内容，如：<img src=""> -->* <**script**>  **layui**.**use**(**'carousel'**, **function**(){  **var** carousel = **layui**.**carousel**;  *//建造实例* carousel.render({  **elem**: **'#test1'** ,**width**: **'100%'** *//设置容器宽度* ,**arrow**: **'always'** *//始终显示箭头  //,anim: 'updown' //切换动画方式* });  }); </**script**> |

## 4、内置模块laydate

https://www.layui.com/laydate/

1、引入这个laydate，在引入<**script src="/laydate/laydate.js"**></**script**>即可

2、或者引入layui后直接

|  |
| --- |
| 1. layui.use('laydate', function(){ 2. var laydate = layui.laydate;   }); |

3、然后再官网找代码直接引入即可 第一种直接copy到script 标签里，第二种copy到layui.use里

|  |
| --- |
| <**input type="text" class="date"**>   laydate.render({  **elem**: **'.date'** ,**calendar**: **true** });  <**input type="text" class="date"**> 2个 <**input type="text" class="date"**>  **lay**(**'.date'**).each(**function**(){  laydate.render({  **elem**: **this** ,**trigger**: **'click'** }); }); |

## 5、颜色

只是提供一些颜色的选择，颜色的搭配，让配色有比较好的参考



## 6、图标



不同的图标引入不同class样式即可， 必须都加layui-icon

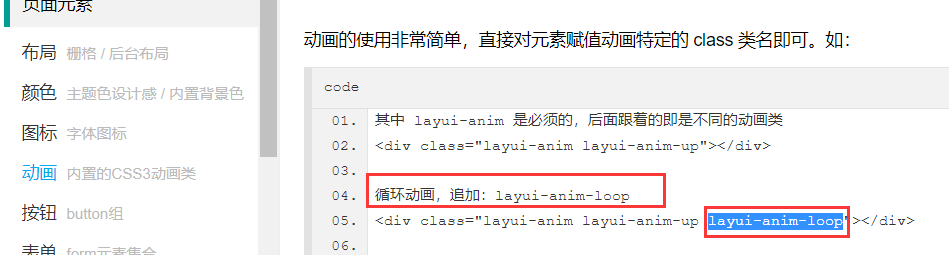
而且可以用控制文字的方式控制他设置大小和颜色 例如变成红色 大小30px

<**i class="layui-icon layui-icon-rate-half" style="font-size**: 30**px**; **color**: **red**;**"**></**i**>

## 7、动画效果

动画效果 ，直接把动画class加到里面即可 例如加到上方字体里

<**i class="layui-anim layui-anim-rotate ayui-icon layui-icon-rate-half" style="font-size**: 30**px**; **color**: **red**;**"**></**i**>



如果让他一直执行 可以再加个loop 样式，动画样式可以加给任意元素div button等

## 8、按钮

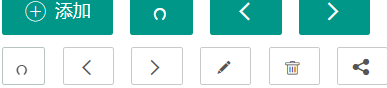


按钮也很简单有标准 和 链接按钮 换class即可，还有大小禁止，最大适应化按钮

<**a href="http://www.layui.com" class="layui-btn layui-btn-lg layui-btn-normal"**>一个可跳转的按钮</**a**>

<button class="layui-btn">一个标准的按钮</button>

按钮样式可以加给任意元素div button等

还可以组合在里面加图标

|  |
| --- |
| <button class="layui-btn layui-btn-sm layui-btn-primary">  <i class="layui-icon">*&*#x1002;</i>  </button> |



|  |
| --- |
| <**div class="layui-btn-group"**>  <**button class="layui-btn layui-btn-sm"**>增加</**button**>  <**button class="layui-btn layui-btn-sm"**>编辑</**button**>  <**button class="layui-btn layui-btn-sm"**>删除</**button**> </**div**> |

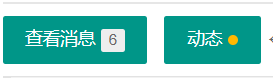
## 10、导航也是直接copy即可

## 11、选项卡直接copy即可

## 12、进度条(动态需要学习element模块)

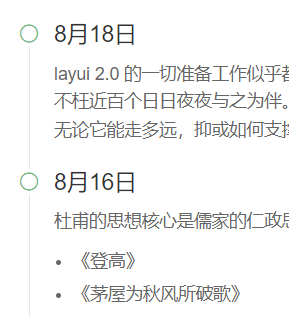


## 14、小徽章

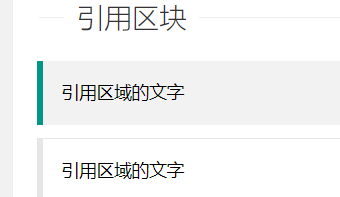


<**button class="layui-btn"**>动态  
 <**span class="layui-badge-dot layui-bg-orange"**></**span**>  
</**button**>

## 15、时间线



## 16、引用区块和分割线



、

# 扩展

JSON.stringify(data) 对象转化为json

无论什么事件想要获取当前行信息都可以试试**{{ d.id }}**