EasyAdmin用户手册集合

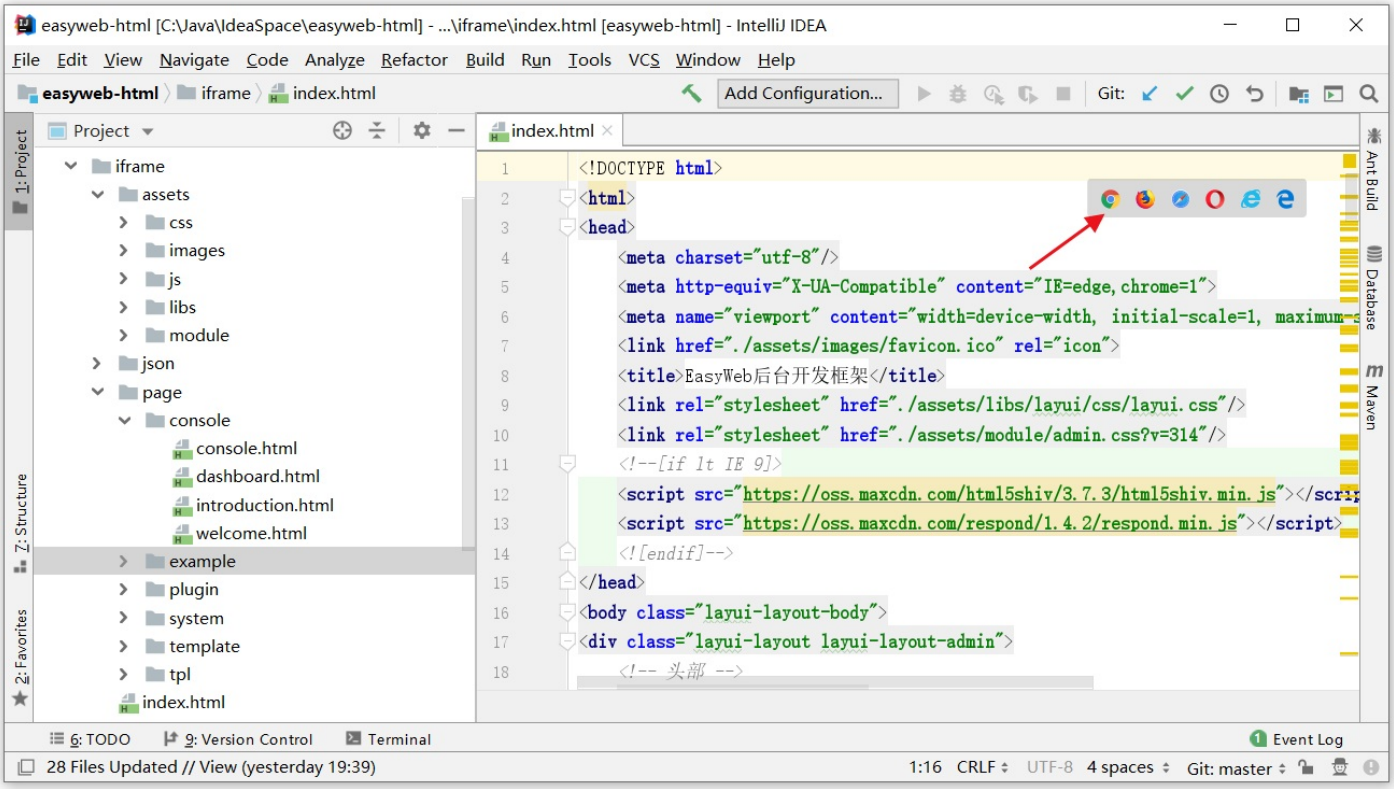
wirror800

目　录

1、开始使用

1.1.导入项目 :id=import

1. 下载项目后进行解压
2. 使用IDEA、WebStorm、HBuilder等前端开发工具打开
3. 打开index.html点击右上角图标运行到浏览器：



!> 注意： 必须以http://的形式访问，而不是file://的形式访问。

1.2.项目结构 :id=structure

```
|-assets
|     |-css                // 样式
|     |-images             // 图片
|     |-js
|         |-main.js        // 入口js
|     |-libs               // 第三方库, echarts(图表)、layui
|     |-module             // layui扩展模块, 版本更新只用替换此目录
|         |-theme              // 主题资源
|         |-img                // easyweb框架用到的图片
|         |-admin.css          // easyweb框架核心样式
|         |-admin.js           // admin模块
|         |-index.js           // index模块
|         |-********           // 其他扩展模块, 不一一列举
|-components               // html子页面
```

```
|-json              // 模拟数据
|-index.html        // 主页面
```

> main.js为入口js，上手使用项目前最好先看看它的说明。

1.3.index.html结构说明 :id=declare

```
<html>
<head>
    <link rel="stylesheet" href="assets/libs/layui/css/layui.css"/>
    <link rel="stylesheet" href="assets/module/admin.css"/>
</head>
<body class="layui-layout-body">
    <div class="layui-layout layui-layout-admin">
        <!-- 头部 -->
        <div class="layui-header">...</div>
        <!-- 侧边栏 -->
        <div class="layui-side">...</div>
        <!-- 主体部分 -->
        <div class="layui-body">...</div>
        <!-- 底部 -->
        <div class="layui-footer">...</div>
    </div>

    <script type="text/javascript" src="assets/libs/layui/layui.js"></script>
    <script type="text/javascript" src="assets/js/main.js"></script>
</body>
</html>
```

index.html就是这样固定的结构，一般不需要修改，只用在components下写子页面即可。

1.4.添加一个菜单 :id=add_menu

1. 打开json/menus.json，在合适的位置添加一个菜单：

   ```
   {
      "name": "xx管理",
      "url": "#/xxx.html"
   }
   ```

2. 在components下面新建一个xxx.html页面
3. 运行index.html，查看效果

!> 注意： url以"#/"开头才会受路由控制，实现局部刷新。

1.5.main.js说明 :id=main_js

```
layui.config({
    base: 'assets/module/'  // 配置layui扩展模块目录
}).extend({  // 配置每个模块分别所在的目录
    notice: 'notice/notice',
    step: 'step-lay/step'
}).use(['layer', 'element', 'config', 'index', 'admin', 'laytpl'], function ()
{
    var $ = layui.jquery;
    var layer = layui.layer;
    var element = layui.element;
    var config = layui.config;
    var index = layui.index;
    var admin = layui.admin;
    var laytpl = layui.laytpl;

    // 加载侧边栏
    admin.req('menus.json', {}, function (res) {
        laytpl(sideNav.innerHTML).render(res, function (html) {
            $('.layui-layout-admin>.layui-side .layui-nav').html(html);
            element.render('nav');
        });
        index.regRouter(res);  // 注册路由
        index.loadHome({  // 加载主页
            url: '#/console/console1',
            name: '<i class="layui-icon layui-icon-home"></i>'
        });
    }, 'get');

    // 移除loading动画
    setTimeout(function () {
        admin.removeLoading();
    }, 300);
});
```

- layui.config是告诉layui扩展模块所在目录
- layui.extend是配置每个模块具体js位置
- admin.removeLoading()是移除页面加载动画
- 侧边栏通过ajax加载，然后注册路由，加载主页

像 `admin.js` 、 `index.js` 这些没有用子目录存放的模块不需要配置layui.extend，延时移除加载动画是给切换主题预留时间。

!> 注意： 只有注册了路由，点击"#/xxx"的时候，才能找到对应的地址并实现局部刷新

1.6.config模块 :id=config

config模块主要是用于给index模块和admin模块提供配置，可根据自己需要进行更改。

参数配置：

配置名	默认	说明
version	312	版本号，加载子页面会带上,为true则随机，意味着不会有缓存
base_server	'json/'	接口地址，admin.req方法会自动加
tableName	'easyweb-spa'	存储表名
pageTabs	false	是否开启多标签
openTabCtxMenu	true	是否开启Tab右键菜单
maxTabNum	20	最多打开多少个tab
viewPath	'components'	视图位置
viewSuffix	'.html'	视图后缀
defaultTheme	'theme-admin'	默认主题
reqPutToPost	true	req请求put方法变成post
cacheTab	true	是否记忆Tab

方法：

配置名	默认
getToken()	获取token
putToken(token)	缓存token
removeToken()	清除token
getUser()	获取当前登录的用户信息
putUser(user)	缓存当前登录的用户信息
getUserAuths()	获取用户所有权限
getAjaxHeaders(requestUrl)	ajax请求的header
ajaxSuccessBefore(res, requestUrl)	ajax请求结束后的处理，返回false阻止代码执行
routerNotFound(r)	路由不存在处理

解决缓存问题配置version版本号在加载子页面时会带上版本号，为true会随机生成，

如果config.js被缓存，请检查main.js里面layui.config有没有写version: true。

2、index模块

2.index模块 :id=index

index模块主要是负责管理选项卡的打开、关闭，使用方法：

```
layui.use(['index'], function () {
    var index = layui.index;

    index.xxx();  // 调用index模块的方法
});
```

2.1.批量注册路由 :id=reg

```
index.regRouter([{
    name: '用户管理',
    url: '#/system/user'
}]);
```

参数是一个数组，可以批量注册路由，当你访问 `#/system/user` 的时候，就会打开一个“用户管理”的标签页，“url”是路由关键字，
对应的页面地址是“components/system/user.html”，“components”和“.html”可以在config模块中配置。

2.2.加载默认主页 :id=load_home

```
index.loadHome({
    url: '#/console/console1',
    name: '<i class="layui-icon layui-icon-home"></i>',
    loadSetting: true,
    onlyLast: false
});
```

- url： 必填 主页路径
- name： 必填 Tab标题
- loadSetting： 非必填 加载缓存的设置
- onlyLast： 非必填 是否仅恢复最后一个Tab

> 主页的地址可以不在侧边栏中，因为“loadHome()”方法会检查并注册路由。

2.3.打开一个选项卡 :id=open_tab

```
index.openNewTab({
    name: '用户管理',
    url: '#/system/user'
});

// 也可以加参数
index.openNewTab({
    name: '用户管理',
    url: '#/system/user/id=1'
});
```

- name： 选项卡的标题
- url： 路由关键字

也可以使用 `<a href="#/system/user">用户管理</a>` ，这种写法必须保证“#/system/user”已经注册了路由，而“openNewTab()”方法会自动注册路由。

!> 注意：openNewTab方法只是在调用这个方法的时候注册路由，如果刷新页面，路由就会不存在了，如果想刷新页面也能使用，需要提前在main.js中注册路由。

```
// 在main.js的index.loadHome()方法之前注册即可，注册路由不用加参数
index.regRouter([{
    name: '用户管理',
    url: '#/system/user'
}]);
```

2.4.关闭指定选项卡 :id=close_tab

```
index.closeTab('/system/user');
```

!> 注意：关闭选项卡不要带“#”号。

2.5.跳到指定选项卡 :id=go_tab

```
index.go('/system/user');
```

跳转页面不要带“#”号，相当于点击了 `<a href="#/system/user">` 。

2.6.修改Tab标题 :id=set_tab

```
index.setTabTitle('Hello');  // 修改当前Tab标题文字，也支持单标签模式
```

```
index.setTabTitle('Hello', tabId);  // 修改指定Tab标题文字

index.setTabTitleHtml('<span>Hello</span>');  // 修改整个标题栏的html，此方法只在单标签模式有效
```

关闭多标签时框架会自动生成一个标题栏，可使用此方法修改标题栏内容，参数为undefined时为隐藏标题栏。

2.7.获取hash路径 :id=hash_path

```
index.getHashPath('#/system/user');

index.getHashPath('#/system/user/id=1/name=aa');
```

这两种hash最后返回的都是“/system/user”，后面属于参数传递，不属于视图的路径。

2.8.侧边栏手风琴折叠 :id=accordion

默认是开启的，如要关闭配置如下：

```
<!-- 侧边栏 -->
<div class="layui-side">
    <div class="layui-side-scroll">
        <ul class="layui-nav layui-nav-tree" lay-accordion="false">
            ......省略其他部分
        </ul>
    </div>
</div>
```

修改layui-nav-tree上的 `lay-accordion` 属性为false或者直接去掉即可，所谓手风琴效果就是一个菜单展开的时候，其他菜单自动折叠。

2.9.切换Tab自动刷新

```
// 选项卡切换自动刷新
element.on('tab(admin-pagetabs)', function (data) {
    var tabId = $(this).attr('lay-id');
    var $content = $('.layui-layout-admin>.layui-body>.layui-tab>.layui-tab-content>.layui-tab-item>div[lay-id="' + tabId + '"]');
    var layHash = $content.attr('lay-hash');
    if ($content.html() && !layui.layRouter.isRefresh) {
        layui.layRouter.refresh(layHash);
    }
});
```

上面代码写在main.js中即可。

3、admin模块(常用)

3.1.全部方法 :id=method

方法	参数	描述
flexible(expand)	true和false	折叠/展开侧导航
activeNav(url)	a标签里面的lay-href值	设置侧导航栏选中
refresh(url)	url，可为空	刷新指定Tab或当前Tab
closeAllTabs()	无	关闭所有选项卡
closeOtherTabs(url)	url	关闭除url外所有选项卡
closeThisTabs(url)	url，可为空	关闭url或当前选项卡
rollPage(d)	方向(left、right、auto)	滚动选项卡tab
iframeAuto()	无	让当前的ifram弹层自适应高度
closeThisDialog()	无	关闭当前iframe弹窗
closeDialog(elem)	dom选择器	关闭elem元素所在的弹窗(页面层弹窗)
putTempData(key, value)	key,value	缓存临时数据
getTempData(key)	key	获取缓存的临时数据
parseJSON(string)	字符串	解析json，解析失败返回undefined
getPageHeight()	无	获取浏览器高度
getPageWidth()	无	获取浏览器宽度
btnLoading(elem)	dom选择器	设置按钮为加载状态
openSideAutoExpand()	无	开启鼠标移入侧边栏自动展开
openCellAutoExpand()	无	开启鼠标移入表格单元格超出内容自动展开
modelForm(layero, btnFilter, formFilter)		把弹窗自带的按钮跟弹窗内的表单绑定一起
hasPerm(auth)	权限	判断当前登录的用户是否有权限
renderPerm()	无	移除没有权限的dom元素

使用示例：

```
layui.use(['admin'], function () {
    var admin = layui.admin;

    var pageHeight = admin.getPageHeight();    // 获取浏览器高度
});
```

admin.iframeAuto()方法：

针对type:2的弹窗自适应弹窗高度，写在弹窗的子页面中，此方法是调用一次做一次高度自适应，如果你用js动态修改了弹窗子页面的高度，需要再调用一次。

admin.closeThisDialog()：

关闭当前iframe类型弹窗，针对type:2的弹窗，在弹窗的子页面调用即可关闭当前的iframe弹窗。

admin.closeDialog(elem)：

关闭非iframe类型的弹窗，调用需要传递弹窗页面里面任意一个的元素：

```
admin.closeDialog('#xxx');  // 关闭id为xxx元素所在的页面层的弹窗
```

关闭弹窗还可以使用ew-event操作：

```
<!-- 关闭页面层的弹窗 -->
<button ew-event="closeDialog"></button>
<!-- 关闭iframe类型的弹窗 -->
<button ew-event="closeIframeDialog"></button>
```

admin.openSideAutoExpand()方法需要在index.html中调用，admin.openCellAutoExpand()可在任何页面调用。

3.2.判断权限hasPerm :id=has_perm

通过“admin.hasPerm(auth)”可以控制按钮级别的权限隐藏：

```
if(!admin.hasPerm('user:add')) {
    $('#btnUserAdd').remove();
}
```

- “hasPerm”方法需要通过“config.getUserAuths”方法来获取全部权限；

- “config.getUserAuths”是通过“config.getUser”方法从用户信息中获取权限的；
- "config.getUser"从本地缓存中获取用户信息的，"config.putUser"把用户信息放入缓存中。

在main.js中有获取用户信息并通过"config.putUser"放入缓存中的写法，请参考。

还可以通过加属性的方式来隐藏：

```
<button perm-show="user:add">添加</button>
```

> “perm-show”可以用于任意元素，对于动态添加的元素调用admin.renderPerm()来更新渲染。

3.3.popupRight和open :id=open

快速使用：

```
// 打开弹窗
admin.open({
    type: 2,
    content: 'tpl-theme.html'
});

// 打开右侧面板
admin.popupRight({
    type: 2,
    content: 'tpl-theme.html'
});
```

这两个方法只是对layer.open进行了一层封装，用法和layer一样，查看layer文档。

新增参数url：

```
admin.open({
    title: 'Hello',
    url: 'tpl-theme.html'
});


admin.popupRight({
    url: 'tpl-theme.html'
});
```

`type:2, content:xxx` 这种是iframe类型的弹窗，使用 `url` 会通过ajax加载页面到弹窗中，而不是iframe嵌入。

当使用url方式的时候，弹窗页面应该是代码片段，而不是完整的html，如下所示：

```
<style>
* { color: #333; }
</style>
<form id="modelRoleForm" lay-filter="modelRoleForm" class="layui-form model-for
m">
    <input name="roleId" type="hidden"/>
    <div class="layui-form-item">
        <label class="layui-form-label">角色名</label>
        <div class="layui-input-block">
            <input name="roleName" placeholder="请输入角色名" type="text" class="
layui-input" maxlength="20"
                   lay-verType="tips" lay-verify="required" required/>
        </div>
    </div>
    <div class="layui-form-item text-right">
        <button class="layui-btn layui-btn-primary" type="button" ew-event="clo
seDialog">取消</button>
        <button class="layui-btn" lay-filter="modelSubmitRole" lay-submit>保存<
/button>
    </div>
</form>
<script>
    layui.use(['layer', 'form'], function () {
        var $ = layui.jquery;
        var layer = layui.layer;
        var form = layui.form;

        // 表单提交事件
        form.on('submit(modelSubmitRole)', function (data) {
            console.log(data.field);
            return false;
        });

    });
</script>
```

页面不需要html、body这些东西，并且可以直接用 `<script>` 标签来写事件，至于参数传递，请到弹窗专题查看。

> 使用url方式更加符合单页面应用

3.4.加载动画loading :id=loading

快速使用：

```
admin.showLoading('#xxx');  // 在id为xxx的元素中显示加载层
admin.showLoading('#xxx', 1, '.8');  // 显示type为1、透明度为0.8的遮罩层
```

- 参数一 `elem` 非必填 元素选择器，不填为body；
- 参数二 `type` 非必填 动画类型(1 小球，2 魔方，3信号)，默认是1；
- 参数三 `opacity` 非必填 透明度(0 - 1)，默认不透明；

尺寸控制，提供有两种尺寸，用法：

```
admin.showLoading({
    elem: '#xxx',
    type: 1,
    size: 'sm'  // 默认是sm小型尺寸，还可以选md大型尺寸
});
```

移除加载动画：

```
admin.removeLoading('#xxx');
admin.removeLoading('#xxx', true, true);
```

- 参数一 非必填 元素选择器，不填为body；
- 参数二 非必填 true是淡出效果，false直接隐藏，默认是true；
- 参数三 非必填 true是删除，false是隐藏不删除，默认是false；

页面载入的加载动画：

```
<body>
    <!-- 小球样式 -->
    <div class="page-loading">
        <div class="ball-loader">
            <span></span><span></span><span></span><span></span>
        </div>
    </div>

    <!-- 魔方样式 -->
    <div class="page-loading">
        <div class="rubik-loader"></div>
    </div>

    <!-- 信号样式 -->
    <div class="page-loading">
        <div class="signal-loader">
            <span></span><span></span><span></span><span></span>
        </div>
    </div>

    <!-- 加sm是小型尺寸 -->
```

```
    <div class="page-loading">
        <div class="signal-loader sm">
            <span></span><span></span><span></span><span></span>
        </div>
    </div>
</body>
```

写在页面中需要在js中调用 `admin.removeLoading()` 移除加载动画，common.js中已经写好了。

按钮loading：

```
admin.btnLoading('#btn1');   // 设置按钮为loading状态
admin.btnLoading('#btnLoading', false);  // 移除按钮的loading状态

admin.btnLoading('#btn1', '加载中');   // 设置按钮为loading状态，同时修改按钮文字
admin.btnLoading('#btnLoading', '保存', false);  // 移除按钮的loading状态，同时修改
按钮文字
```

3.5.ajax封装 :id=ajax

req方法：

```
admin.req('url',{
    username: 'admin',
    password: '123456'
}, function(res){
    alert(res.code + '-' + res.msg);
}, 'get');
```

- 参数一　请求的url
- 参数二　请求参数
- 参数三　请求回调（失败也进此回调，404、403等）
- 参数四　请求方式，get、post、put、delete等

ajax方法，参数同$.ajax：

```
admin.ajax({
    url: 'url',
    data: {},
    headers： {},
    type: 'post',
    dataType: 'json',
    success: function(res){
        alert(res.code + '-' + res.msg);
    }
```

```
});
```

req和ajax都实现了自动传递header、预处理、系统错误依然回调到success等功能。

> 自动传递header是通过config.getAjaxHeaders()方法，请求回调预处理是通过config.ajaxSuccessBefore()方法。

3.6.ew-event事件绑定 :id=event

使用示例：

```
<a ew-event="fullScreen">全屏</a>
<a ew-event="flexible">折叠导航</a>
```

事件	描述
flexible	折叠侧导航
refresh	刷新主体部分
closeThisTabs	关闭当前选项卡
closeOtherTabs	关闭其他选项卡
closeAllTabs	关闭全部选项卡
leftPage	左滚动选项卡
rightPage	右滚动选项卡
closeDialog	关闭当前页面层弹窗
closeIframeDialog	关闭当前iframe弹窗
theme	打开主题设置弹窗
note	打开便签弹窗
message	打开消息弹窗
psw	打开修改密码弹窗
logout	退出登录
fullScreen	全屏切换
back	浏览器后退
open	打开弹窗
popupRight	打开右侧弹窗

ew-event属性可用于任何元素，不仅仅是a标签，theme、note等可以通过 `data-url` 属性配置对应的url，还可以通过 `data-window="top"` 属性配置在父页面处理事件。

```
<a ew-event="theme" data-url="xxx.html">主题</a>
```

3.7.open和popupRight事件 :id=event_open

这两个事件是用来支持非js方式打开弹窗：

```
<button ew-event="open" data-type="2" data-content="http://baidu.com">iframe弹窗</button>

<button ew-event="open" data-type="1" data-url="form.html">页面弹窗</button>

<button ew-event="open" data-type="1" data-content="#userForm">页面弹窗</button>
<form id="userForm">......省略</form>

<!-- 设置area和offset -->
<button ew-event="open" data-type="1" data-content="Hello" data-area="80px,60px" data-offset="10px,10px">页面弹窗</button>

<!-- popupRight一样的用法 -->
<button ew-event="popupRight" data-type="2" data-url="http://baidu.com" data-title="百度一下，你就知道">右侧弹窗</button>

<!-- function类型参数写法 -->
<button ew-event="open" data-type="1" data-content="Hello" data-success="onDialogSuccess">页面弹窗</button>
<script>
    layui.use(['layer'], function(){
        var layer = layui.layer;

        // 方法需要加window
        window.onDialogSuccess = function(){
            layer.msg('弹窗被成功打开了');
        };
    });
</script>
```

layer支持的参数大部分都可以通过data属性来设置，数组类型用逗号分隔，function类型需要把作用域放在window对象下。

4、admin模块(进阶)

4.1.文字提示 :id=tips

鼠标滑过弹出tips，使用示例：

```
<button lay-tips="大家好！">按钮</button>

<button lay-tips="大家好！" lay-direction="2" lay-bg="#009788">按钮</button>

<button lay-tips="大家好！" lay-offset="10px">按钮</button>

<button lay-tips="大家好！" lay-offset="10px,10px">按钮</button>
```

- lay-direction： 设置位置，1上面(默认)、2右边、3下面、4左边
- lay-bg： 设置背景颜色
- lay-offset： 设置偏移距离，(上下，左右)

4.2.地图选择位置 :id=location

快速使用：

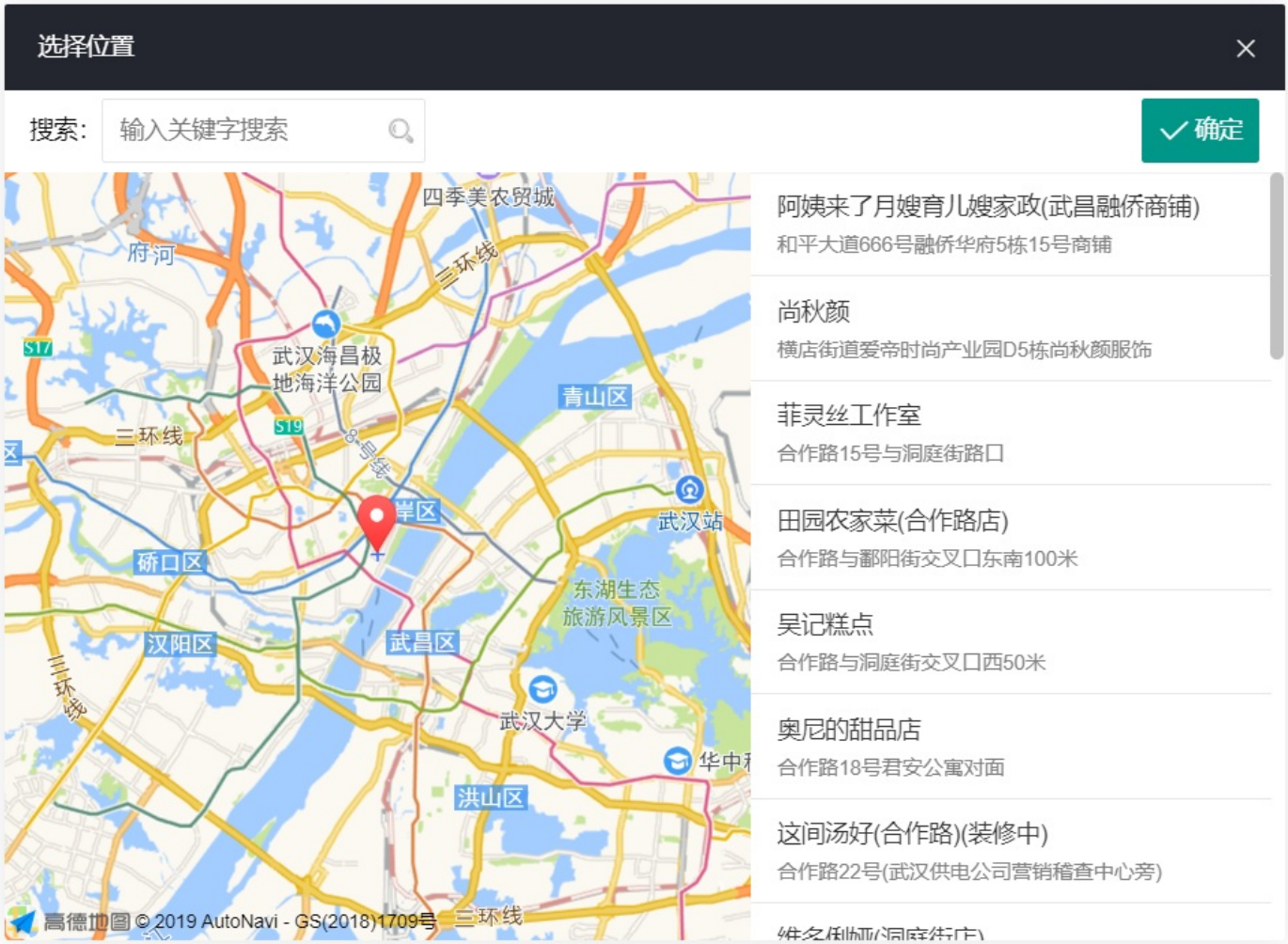
```
admin.chooseLocation({
    needCity: true,
    onSelect: function (res) {
        layer.msg(JSON.stringify(res), {icon: 1});
    }
});
```

参数	默认	描述
title	"选择位置"	弹窗标题
needCity	false	是否返回行政区，省市区默认不返回
center	定位当前城市	地图默认的中心点
defaultZoom	11	地图默认缩放级别
pointZoom	17	选中时地图的缩放级别

keywords	无	poi检索关键字，例如：建筑、写字楼
pageSize	30	poi检索最大数量
onSelect	无	选择回调
mapJsUrl	内置	高德地图js的url

- 地图默认中心点参考值：[116.397428, 39.90923]，经度，纬度
- 地图url参考值： `https://webapi.amap.com/maps?v=1.4.14&key=xxxxxxx`
- 返回结果说明：
  - res.name; // 地点名称
  - res.address; // 详细地址
  - res.lat; // 纬度
  - res.lng; // 经度
  - res.city; // 城市，是一个对象
  - res.city.province; // 省
  - res.city.city; // 市
  - res.city.district; // 区
  - res.city.citycode; // 城市代码

地图相关参数的详细介绍请前往高德地图API查看。



4.3.裁剪图片 :id=crop_img

快速使用：

```
admin.cropImg({
    aspectRatio: 1/1,
    imgSrc: '../../assets/images/15367146917869444.jpg',
    onCrop: function (res) {
        // 返回的res是base64编码的裁剪后的图片
        layer.msg('<img src="' + res + '" width="220px" height="220px"/>');
    }
});
```

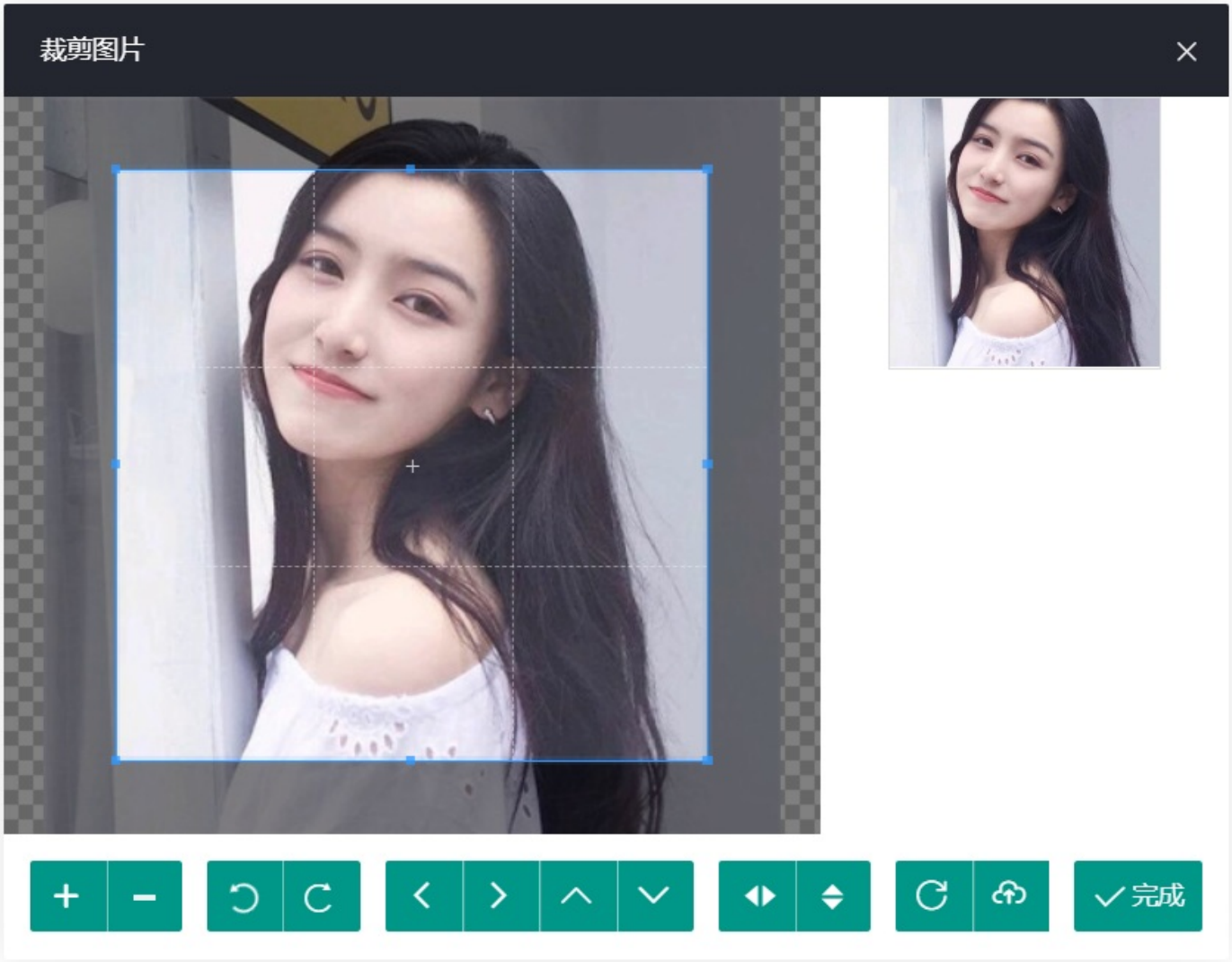
参数	默认	描述
title	"裁剪图片"	弹窗标题
aspectRatio	1/1	裁剪比例，例如：16/9
imgSrc	无	要裁剪的图片，无则先弹出选择图片
imgType	'image/jpeg'	

imgType	'image/jpeg'	
onCrop	无	裁剪完成回调
limitSize	不限制	限制选择的图片大小
acceptMime	'image/*'	限制选择的图片类型
exts	不限制	限制选择的图片后缀

- acceptMime参考值：'image/jpg, image/png'（只显示 jpg 和 png 文件）
- exts参考值：jpg|png|gif|bmp|jpeg

后面三个参数配置可参考upload模块，

如果想单独在页面中使用图片裁剪功能请参考Cropper插件文档



4.2.动画数字 :id=anim_num

快速使用：

```
<h2 id="demoAnimNum1">12345</h2>
```

```
<h2 id="demoAnimNum2">￥2373467.342353</h2>
<h2 id="demoAnimNum3">上浮99.98%</h2>

<script>
layui.use(['admin'], function () {
    var $ = layui.jquery;
    var admin = layui.admin;

    admin.util.animateNum('#demoAnimNum1');
    admin.util.animateNum('#demoAnimNum2');
    admin.util.animateNum('#demoAnimNum3', true, 500, 100);

});
</script>
```

- 参数一　需要动画的元素
- 参数二　是否开启千分位，开启后每进千加逗号分隔
- 参数三　动画间隔，默认500
- 参数四　动画粒度，默认100

此方法会智能过滤除数字外的字符。

4.3.经纬度转换 :id=jwdzh

快速使用：

```
// GCJ02转BD09
var point = admin.util.Convert_GCJ02_To_BD09({
   lat: 30.505674,
   lng: 114.40043
});
console.log(point.lng + ',' + point.lat);

// BD09转GCJ02
var point = admin.util.Convert_BD09_To_GCJ02({
   lat: 30.512004,
   lng: 114.405701
});
console.log(point.lng + ',' + point.lat);
```

- 高德地图、腾讯地图以及谷歌中国区地图使用的是GCJ-02坐标系
- 百度地图使用的是BD-09坐标系

不同的坐标系之间会有几十到几百米的偏移，所以在开发基于地图的产品时，可以通过此方法修正不同坐标系之间的偏差，

详细了解国内各坐标系，当然即使用此方法转换也会存在几米的误差，
因为浮点运算精度难免会丢失。

4.4.深度克隆对象 :id=deepclone

快速使用：

```
var o1 = {
    name: 'xxx',
    role: ['admin', 'user']
};

var o2 = admin.util.deepClone(o1);
```

对象型在参数传递是通常是引用传递，有时需要把对象克隆一份再传递，避免传递的参数被修改导致第二次传递出现问题。

5、公共样式

5.1.公共类 :id=public

类名（class）	说明
pull-left	左浮动
pull-right	右浮动
text-left	内容居左
text-center	内容居中
text-right	内容居右
inline-block	设置display为inline-block
bg-white	设置背景为白色
layui-link	设置a标签颜色为主题色
layui-text	.layui-text下面的a标签为蓝色
text-muted	文字颜色为灰色
text-success	文字颜色为绿色，成功色
text-warning	文字颜色为黄色警告色
text-danger	文字颜色为红色危险色
text-info	文字颜色为蓝色信息色
text-primary	文字颜色为主题色

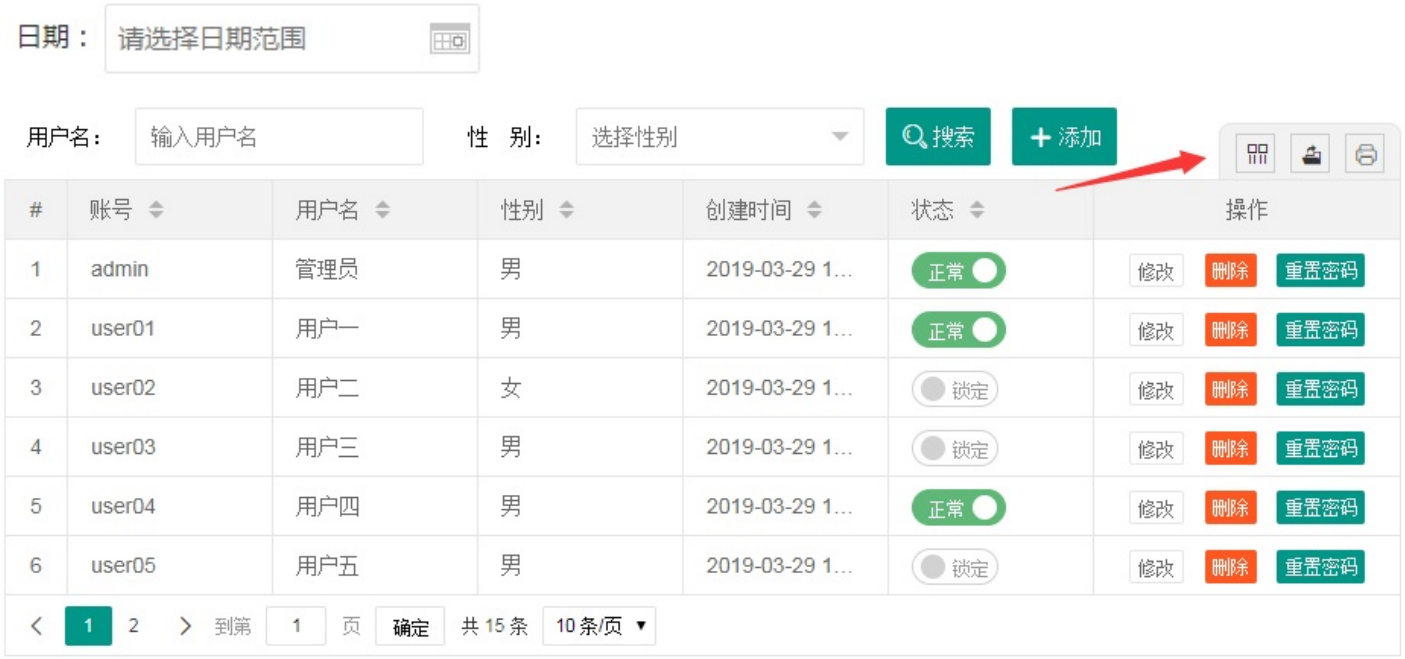
以上是easyweb增加的公共类，当然也可以使用Layui公共类。

5.2.组件样式 :id=module

类名（class）	说明
icon-btn	带图标的按钮，会缩小边距
date-icon	在元素的右边加入日期的图标
icon-search	在元素的右边加入搜索的图标
btn-circle	圆形按钮，参见便签界面
layui-form-select-top	控制下拉框上弹出，加载select父元素上
xm-select-nri	多选下拉框去掉最后那个没用的图标，加在父元素上
mini-bar	如果有滚动条，使用细的风格

arrow2	设置侧边栏小三角为箭头图标，加在layui-nav上
arrow3	设置侧边栏小三角为加减号图标，加在layui-nav上
close-footer	关闭页脚，加在body上
table-tool-mini	数据表格工具栏mini样式，加在table父元素上
full-table	针对full-xxx的table的工具栏mini样式
hide-body-title	全局隐藏单标签模式标题栏，加在body上

```
<!-- 图标按钮 -->
<button class="layui-btn icon-btn"><i class="layui-icon">&#xe615;</i>搜索</butt
on>

<!-- 日期图标 -->
<input class="layui-input date-icon" type="text"/>

<!-- 圆形按钮 -->
<div class="btn-circle">
    <i class="layui-icon layui-icon-add-1"></i>
</div>

<!-- 下拉框上弹出 -->
<div class="layui-form-select-top">
    <select>....</select>
</div>

<!-- 多选下拉框去掉最后那个没用的图标 -->
<div class="xm-select-nri">
    <select xm-select="xx">....</select>
</div>

<!-- 关闭页脚 -->
<body class="layui-layout-body close-footer">

<!-- 表格工具栏mini样式 -->
<div class="table-tool-mini full-table">
    <table id="xxTable" lay-filter="xxTable"></table>
</div>
```



5.3.表单弹窗 :id=form

类名（class）	说明
model-form	调整弹窗内的表单的间距使之更好看
model-form-body	表单内容部分，高度自适应，超过屏幕高度显示滚动条
model-form-footer	表单底部按钮部分，用于固定底部按钮

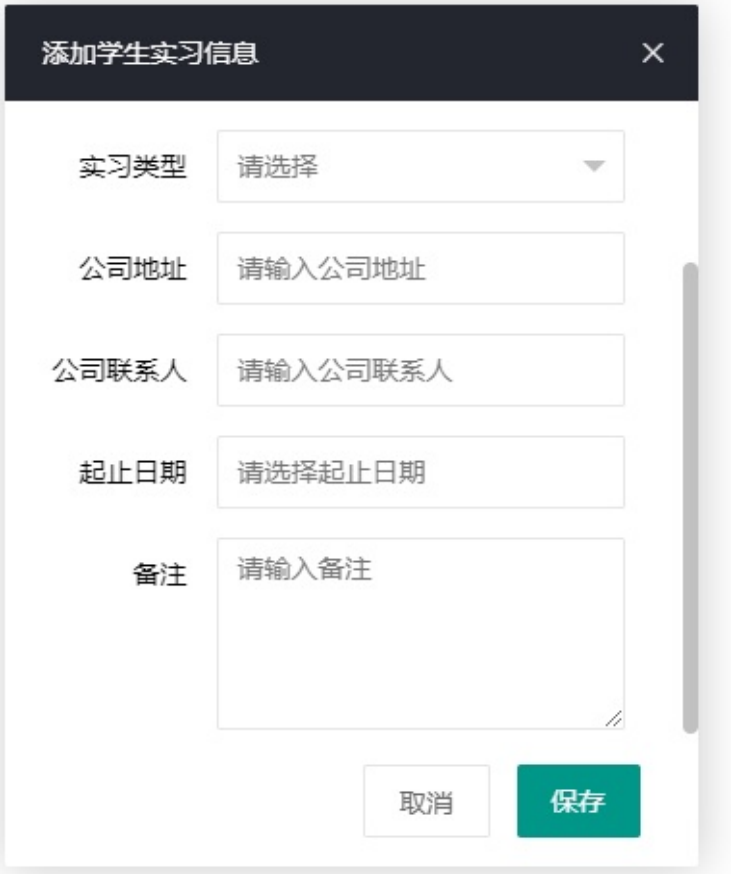
表单弹窗示例：

```
<script type="text/html" id="modelUser">
    <form id="modelUserForm" lay-filter="modelUserForm" class="layui-form model
-form">
        <input name="userId" type="hidden"/>
        <div class="layui-form-item">
            <label class="layui-form-label">账号</label>
            <div class="layui-input-block">
                <input name="username" placeholder="请输入账号" type="text" clas
s="layui-input" maxlength="20"
                       lay-verType="tips" lay-verify="required" required/>
            </div>
        </div>
        <div class="layui-form-item">
            <label class="layui-form-label">性别</label>
            <div class="layui-input-block">
                <input type="radio" name="sex" value="男" title="男" checked/>
                <input type="radio" name="sex" value="女" title="女"/>
            </div>
        </div>
```

```
        <div class="layui-form-item">
            <label class="layui-form-label">角色</label>
            <div class="layui-input-block">
                <select name="roleId" lay-verType="tips" lay-verify="required">
                    <option value="">请选择角色</option>
                    <option value="1">管理员</option>
                    <option value="2">普通用户</option>
                </select>
            </div>
        </div>
        <div class="layui-form-item text-right">
            <button class="layui-btn layui-btn-primary" type="button" ew-event=
"closePageDialog">取消</button>
            <button class="layui-btn" lay-filter="modelSubmitUser" lay-submit>
保存</button>
        </div>
    </form>
</script>
<script>
    layui.use(['admin'],function(){
        var admin = layui.admin;

        admin.open({
            type: 1,
            title: '添加用户',
            content: $('#modelUser').html(),
            success: function (layero, dIndex) {
                // 表单的操作，事件绑定等都写在success回调里面
            }
        });
    });
</script>
```

固定底部操作按钮示例：

```
<script type="text/html" id="modelUser">
    <form id="modelUserForm" lay-filter="modelUserForm" class="layui-form model
-form no-padding">
        <div class="model-form-body" style="max-height: 320px;"> <!-- 如果要超出
屏幕才固定底部，不要写max-height -->
            <div class="layui-form-item">
                <label class="layui-form-label">实习公司</label>
                <div class="layui-input-block">
                    <input name="companyName" class="layui-input"/>
                </div>
            </div>
            <!-- ......省略 -->
        </div>
        <div class="layui-form-item text-right model-form-footer">
            <button class="layui-btn layui-btn-primary" type="button" ew-event=
"closePageDialog">取消</button>
            <button class="layui-btn" lay-filter="modelSubmitUser" lay-submit>
保存</button>
        </div>
    </form>
</script>
```



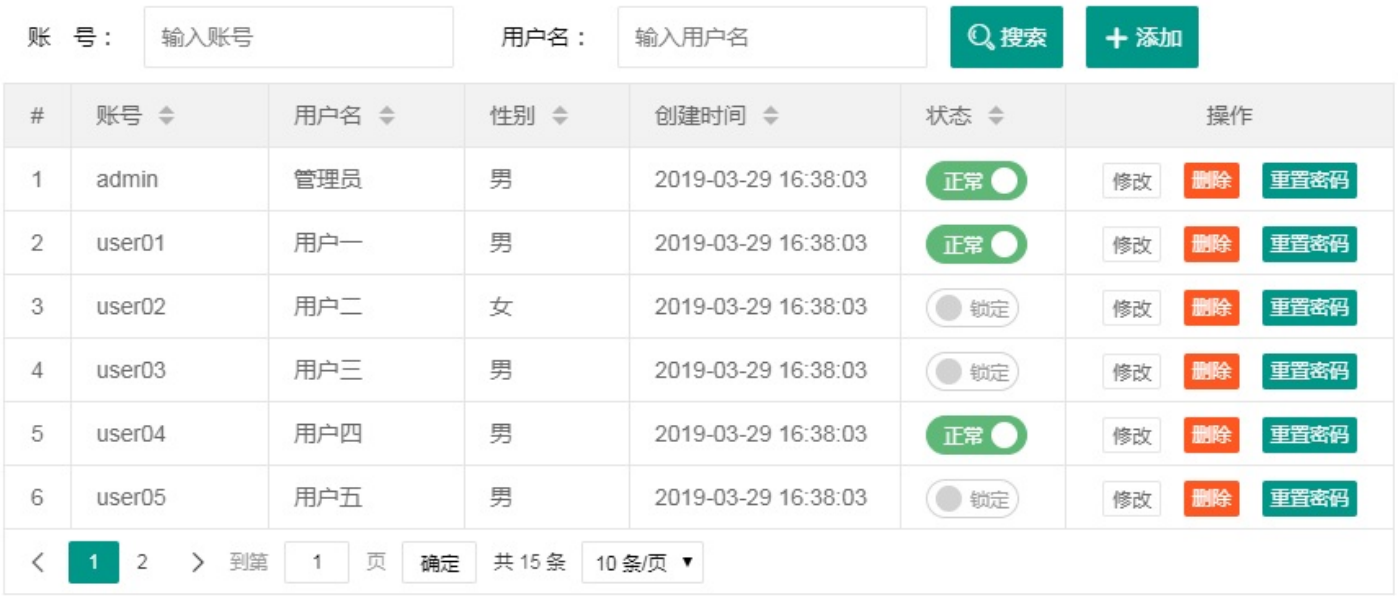
> 两者的区别就在于固定底部操作按钮需要增加model-form-body和model-form-footer，而普通的表单弹窗只需要model-form

5.4.表格工具栏 :id=toolbar

类名（class）	说明
toolbar	调整表格上面的表单间距使之更好看
w-auto	设置width:auto，用于重置一些有固定宽度表单元素
mr0	设置margin-right:0，用于重置一些表单元素的样式

```
<!-- 表格顶部工具栏区域 -->
<div class="layui-form toolbar">
    <div class="layui-form-item">
        <div class="layui-inline">
            <label class="layui-form-label w-auto">账 号：</label>
            <div class="layui-input-inline mr0">
                <input name="username" class="layui-input" type="text" placehol
der="输入账号"/>
            </div>
        </div>
```

```
        <div class="layui-inline">
            <label class="layui-form-label w-auto">用户名：</label>
            <div class="layui-input-inline mr0">
                <input name="nickName" class="layui-input" type="text" placehol
der="输入用户名"/>
            </div>
        </div>
        <div class="layui-inline">
            <button class="layui-btn icon-btn" lay-filter="formSubSearchUser" l
ay-submit>
                <i class="layui-icon">&#xe615;</i>搜索
            </button>
            <button id="btnAddUser" class="layui-btn icon-btn"><i class="layui-
icon">&#xe654;</i>添加</button>
        </div>
    </div>
</div>

<!-- 表格 -->
<table class="layui-table" id="tableUser" lay-filter="tableUser"></table>
```



移动端自动适配效果：

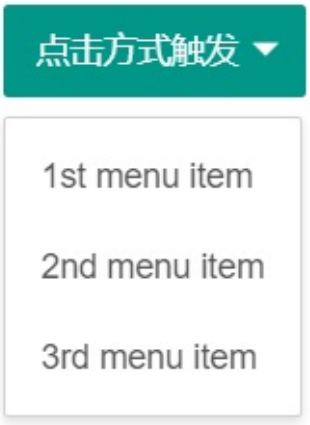

账　号：
输入账号
用户名：
输入用户名
搜索
+ 添加
# 账号 用户名 性别
1 admin 管理员 男
2 user01 用户一 男
3 user02 用户二 女
4 user03 用户三 男
5 user04 用户四 男
6 user05 用户五 男
1 2 到第 1 页 确定 共

6、扩展组件(常用)

6.1、下拉菜单

6.1.1.快速使用 :id=start

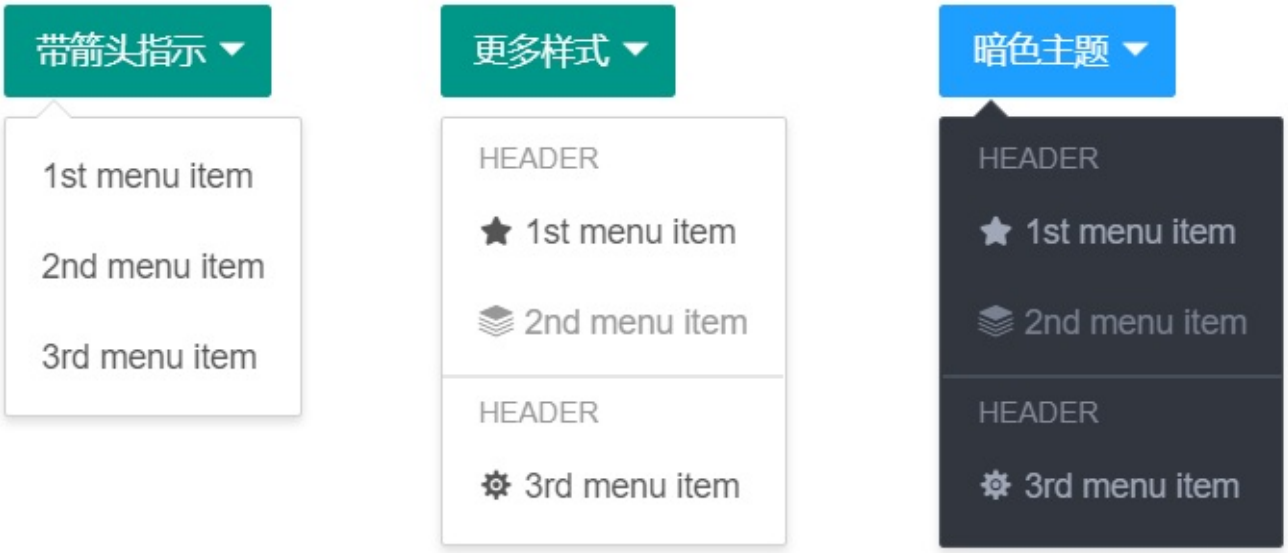
```
<!-- click模式触发 -->
<div class="dropdown-menu">
    <button class="layui-btn icon-btn">
         Click me <i class="layui-icon layui-icon-drop"></i>
    </button>
    <ul class="dropdown-menu-nav">
        <li><a>1st menu item</a></li>
        <li><a>2nd menu item</a></li>
        <li><a>3rd menu item</a></li>
    </ul>
</div>

<!-- hover模式触发，增加dropdown-hover即可 -->
<div class="dropdown-menu dropdown-hover">
    <button class="layui-btn icon-btn">
         Hover me <i class="layui-icon layui-icon-drop"></i></button>
    <ul class="dropdown-menu-nav">
        <li><a>1st menu item</a></li>
        <li><a>2nd menu item</a></li>
        <li><a>3rd menu item</a></li>
    </ul>
</div>

<script>
    layui.use(['dropdown'], function () {
        var dropdown = layui.dropdown;  // 加载模块

    });
</script>
```



6.1.2.更多样式 :id=style

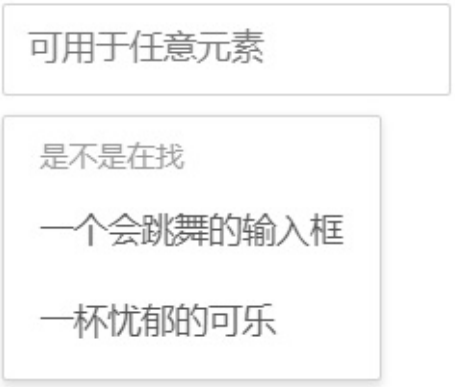
```
<!-- 标题及禁用样式 -->
<div class="dropdown-menu dropdown-hover">
    <button class="layui-btn icon-btn">
         更多样式 <i class="layui-icon layui-icon-drop"></i></button>
    <ul class="dropdown-menu-nav">
        <li class="title">HEADER</li>
        <li><a><i class="layui-icon layui-icon-star-fill"></i>1st menu item</a>
</li>
        <li class="disabled">
            <a><i class="layui-icon layui-icon-template-1"></i>2nd menu item</a
></li>
        <hr>
        <li class="title">HEADER</li>
        <li><a><i class="layui-icon layui-icon-set-fill"></i>3rd menu item</a><
/li>
    </ul>
</div>

<!-- 带小三角样式 -->
<div class="dropdown-menu dropdown-hover">
    <button class="layui-btn icon-btn">
         带小三角 <i class="layui-icon layui-icon-drop"></i>
    </button>
    <ul class="dropdown-menu-nav">
        <div class="dropdown-anchor"></div>
        <li><a>1st menu item</a></li>
        <li><a>2nd menu item</a></li>
        <li><a>3rd menu item</a></li>
    </ul>
</div>

<!-- 暗色主题 -->
<div class="dropdown-menu">
    <button class="layui-btn layui-btn-normal icon-btn">
         暗色主题 <i class="layui-icon layui-icon-drop"></i></button>
    <ul class="dropdown-menu-nav dark">
        <div class="dropdown-anchor"></div>
        <li class="title">HEADER</li>
        <li><a><i class="layui-icon layui-icon-star-fill"></i>1st menu item</a>
</li>
        <li class="disabled">
            <a><i class="layui-icon layui-icon-template-1"></i>2nd menu item</a
></li>
        <hr>
        <li class="title">HEADER</li>
        <li><a><i class="layui-icon layui-icon-set-fill"></i>3rd menu item</a><
/li>
```

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



6.1.3.对任意元素使用 :id=target

```
<div class="dropdown-menu dropdown-hover">
    <input type="text" placeholder="一个会跳舞的输入框" class="layui-input"/>
    <ul class="dropdown-menu-nav">
        <li class="title">是不是在找</li>
        <li><a>另一个会跳舞的下拉框</a></li>
        <li><a>一杯忧郁的可乐</a></li>
    </ul>
</div>
```



6.1.4.带遮罩层 :id=shade

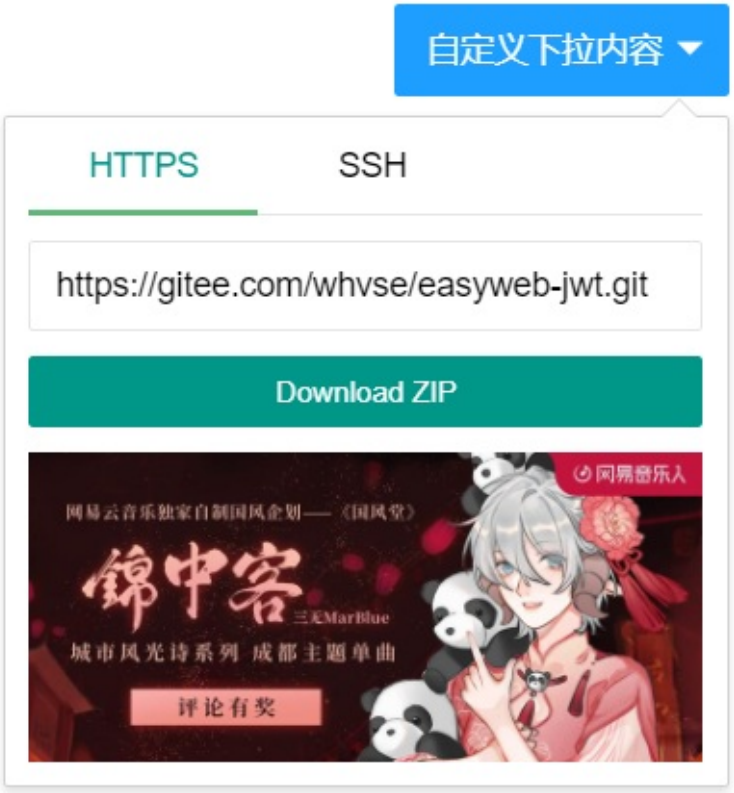
遮罩层是分离式的绑定，通过data-dropdown绑定：

```
<button class="layui-btn layui-btn-normal icon-btn" data-dropdown="#dropdown1">
   带遮罩层 <i class="layui-icon layui-icon-drop"></i>
</button>

<!-- 下拉菜单 -->
<ul class="dropdown-menu-nav dropdown-bottom-right layui-hide" id="dropdown1">
    <div class="dropdown-anchor"></div>
    <li class="title">HEADER</li>
    <li><a><i class="layui-icon layui-icon-star-fill"></i>1st menu item</a></li
>
    <li class="disabled">
        <a><i class="layui-icon layui-icon-template-1"></i>2nd menu item</a></l
i>
    <hr>
    <li class="title">HEADER</li>
    <li><a><i class="layui-icon layui-icon-set-fill"></i>3rd menu item</a></li>
</ul>
```

6.1.5.自定义下拉内容 :id=div_drop

```
<div class="dropdown-menu">
    <button class="layui-btn layui-btn-normal icon-btn">
         克隆/下载 <i class="layui-icon layui-icon-drop"></i></button>
    <div class="dropdown-menu-nav dropdown-bottom-right"
         style="width: 280px;padding: 0 10px 10px 10px;">
        <div class="dropdown-anchor"></div>
        <!-- 下面是自定义内容 -->
        <div class="layui-tab layui-tab-brief">
            <ul class="layui-tab-title">
                <li class="layui-this">HTTPS</li>
                <li>SSH</li>
            </ul>
            <div class="layui-tab-content" style="padding: 10px 0 10px 0;">
                <div class="layui-tab-item layui-show">
                    <input class="layui-input" value="https://gitee.com/whvse/e
asyweb-jwt.git"/>
                </div>
                <div class="layui-tab-item">
                    <input class="layui-input" value="git@gitee.com:whvse/easyw
eb-jwt.git"/>
                </div>
            </div>
        </div>
        <button class="layui-btn layui-btn-sm layui-btn-fluid" style="margin-bo
ttom: 10px;">
            Download ZIP
        </button>
        <img src="http://p1.music.126.net/voV3yPduAhNATICMRJza1A==/109951164017
```

```
919367.jpg"
             width="100%">
        <!-- //end.自定义内容结束 -->
    </div>
</div>
```



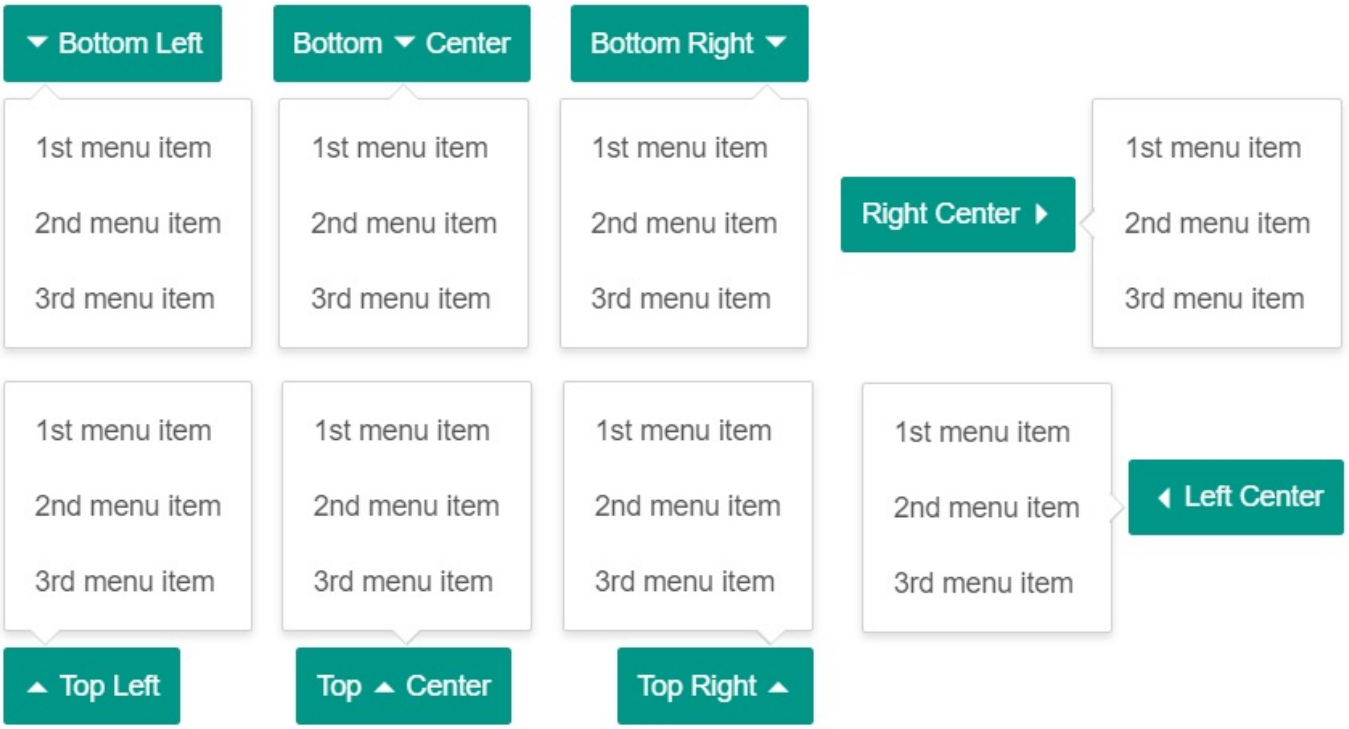
6.1.6.控制显示方向 :id=anchor

在dropdown-menu-nav上加dropdown-bottom-center等class控制位置：

类名	位置
dropdown-bottom-left	下左弹出
dropdown-bottom-center	下中弹出
dropdown-bottom-right	下右弹出
dropdown-top-left	上左弹出
dropdown-top-center	上中弹出
dropdown-top-right	上右弹出
dropdown-left-top	左上弹出
dropdown-left-center	左中弹出
dropdown-left-bottom	左下弹出
dropdown-right-top	右上弹出

dropdown-right-center	右中弹出
dropdown-right-bottom	右下弹出

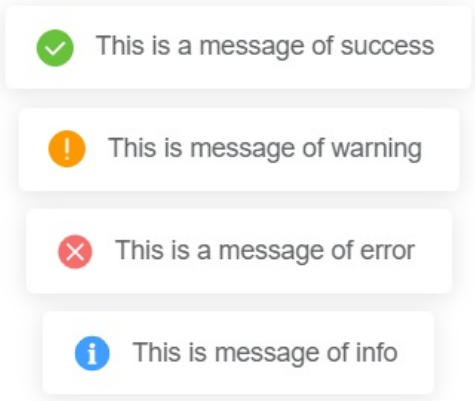
```
<!-- Bottom Center -->
<div class="dropdown-menu dropdown-hover">
    <button class="layui-btn layui-btn-primary icon-btn">
        Bottom <i class="layui-icon layui-icon-drop"></i> Center
    </button>
    <ul class="dropdown-menu-nav dropdown-bottom-center"><!-- 这里加控制方向的类 -
->
        <div class="dropdown-anchor"></div>
        <li><a>1st menu item</a></li>
        <li><a>2nd menu item</a></li>
        <li><a>3rd menu item</a></li>
    </ul>
</div>

<!-- Bottom Right -->
<div class="dropdown-menu dropdown-hover">
    <button class="layui-btn layui-btn-primary icon-btn">
        Bottom Right <i class="layui-icon layui-icon-drop"></i>
    </button>
    <ul class="dropdown-menu-nav dropdown-bottom-right"><!-- 这里加控制方向的类 --
>
        <div class="dropdown-anchor"></div>
        <li><a>1st menu item</a></li>
        <li><a>2nd menu item</a></li>
        <li><a>3rd menu item</a></li>
    </ul>
</div>
```


Bottom Left
Bottom Center
Bottom Right
1st menu item
2nd menu item
3rd menu item
Right Center
Left Center
Top Left
Top Center
Top Right

6.2、消息通知

6.2.1.快速使用 :id=start

```
layui.use(['notice'], function(){
    var notice = layui.notice;

    // 消息通知
    notice.success({
        title: '消息通知',
        message: '你有新的消息，请注意查收!'
    });

    // 提示框，1成功、2失败、3警告、4加载、5信息(蓝色图标)
    notice.msg('Hello', {icon: 1});
});
```



6.2.2.全部方法 :id=method

方法	参数	说明
success(object)	见下方	成功消息（绿色）
warning(object)	见下方	警告消息（黄色）
error(object)	见下方	错误消息（红色）
info(object)	见下方	通知消息（蓝色）
show(object)	见下方	自定义样式
msg(string, object)	文本,其他参数	提示框

destroy()	无	关闭全部
hide(object, toast, closedBy)	见下方	关闭指定的通知
settings(object)	见下方	统一设置默认值

统一设置默认值：

```
notice.settings({
    timeout: 1500,
    transitionIn: 'flipInX',
    onOpened: function(){
        console.log('notice opend!');
    }
});
```

关闭指定的通知：

```
notice.hide({}, document.querySelector('.toast'));
```

- 参数一　重写一些参数，比如关闭动画等
- 参数二　根据自定义的className选择关闭的对象

6.2.3.参数列表 :id=options

参数	说明	默认值	可选值
title	标题	无	string类型
message	内容	无	string类型
position	显示位置	topRight	见下方
transitionIn	进入动画	fadeInLeft	见下方
transitionOut	退出动画	fadeOutRight	见下方
timeout	消失时间	5000	单位毫秒，false永不消失
progressBar	进度条	true	true显示、false不显示
balloon	气泡效果	false	true开启、false关闭
close	关闭按钮	true	true显示，false不显示
pauseOnHover	鼠标滑过暂停消失时间	true	true、false

resetOnHover	鼠标滑过重置消失时间	false	true、false
animateInside	文字动画效果	false	true开启、false关闭
className	自定义class	无	多个用空格分隔
theme	主题	light	light、dark
audio	音效	无	1，2，3，4，5，6
image	显示图片	无	图片地址
imageWidth	图片宽度	60	数字
buttons	显示按钮	[]	[['btn1', function(){}], ['btn2', function(){}]]
overlay	遮罩层	false	true显示,false不显示
drag	滑动关闭	true	true开启，false关闭
layout	布局类型	2	1标题和内容并排，2两排显示
rtl	布局方向	false	false内容居左，true居右
displayMode	显示模式	0	0无限制，1同类型存在不显示，2同类型存在先移除
targetFirst	插入方式	自动	true从上插入，false下插入
onOpened	打开后回调函数	无	function
onClosed	关闭后回调函数	无	function
titleColor	标题颜色	默认	颜色单位
titleSize	标题大小	默认	尺寸单位
messageColor	文字颜色	默认	颜色单位
messageSize	文字大小	默认	尺寸单位
backgroundColor	背景颜色	默认	颜色单位
progressBarColor	进度条颜色	默认	颜色单位

maxWidth	最大宽度	空	尺寸单位

参数position显示位置可选值：

属性	说明
bottomRight	右下角
bottomLeft	左下角
topRight	右上角
topLeft	左上角
topCenter	顶部中间
bottomCenter	底部中间
center	正中间

参数transitionIn进入动画可选值：

属性	说明
bounceInLeft	向左反弹
bounceInRight	向右反弹
bounceInUp	向上反弹
bounceInDown	向下反弹
fadeIn	淡入
fadeInDown	向下淡入
fadeInUp	向上淡入
fadeInLeft	向左淡入
fadeInRight	向右淡入
flipInX	翻转进入

参数transitionOut退出动画可选值：

属性	说明
fadeOut	淡出
fadeOutUp	向上淡出
fadeOutDown	向下淡出
fadeOutLeft	向左淡出
fadeOutRight	向右淡出
flipOutX	翻转退出

6.3、级联选择器

6.3.1.快速使用 :id=start

```
<input id="demoCascader1" placeholder="请选择" class="layui-hide"/>

<script>
layui.use(['cascader'], function () {
    var $ = layui.jquery;
    var cascader = layui.cascader;

    cascader.render({
        elem: '#demoCascader1',
        data: [{
              value: 'beijing',
              label: '北京',
              children: [
                  {
                      value: 'gugong',
                      label: '故宫'
                  },
                  {
                      value: 'tiantan',
                      label: '天坛'
                  },
                  {
                      value: 'wangfujing',
                      label: '王府井'
                  }
              ]
          }]
    });
});
</script>
```

6.3.2.异步加载 :id=async

```
<input id="demoCascader1" placeholder="请选择" class="layui-hide"/>

<script>
layui.use(['cascader'], function () {
    var $ = layui.jquery;
    var cascader = layui.cascader;

    cascader.render({
        elem: '#demoCascader1',
        reqData: function (values, callback, data) {
            // values是当前所有选中的值，data是当前选中的对象
            $.get('xxxx.json', { id: data.value }, function(res){
                callback(res.data);  // 数据请求完成通过callback回调
            },'json');
        }
    });
});
</script>
```

异步加载的数据格式为：

```
[
    {value: 'beijing', label: '北京', haveChildren: true},
    {value: 'jiangsu', label: '江苏', haveChildren: true}
]
```

通过haveChildren字段来标识是否还有子节点，如果你的后台数据格式不是这样，可以在callback之前格式化：

```
cascader.render({
    elem: '#demoCascader1',
    reqData: function (values, callback, data) {
        // values是当前所有选中的值，data是当前选中的对象
        $.get('xxxx.json', { id: data.value }, function(res){
            var newList = [];
            for(var i=0;i<res.data.length;i++){
                  var item = res.data[i];
                  newList.push({
                       value: item.id,
                       label: item.name,
                       haveChildren: item.haveChildren
                  });
            }
            callback(newList);  // 数据请求完成通过callback回调
        },'json');
    }
});
```

6.3.3.自定义分隔符 :id=format

```
<input id="demoCascader1" placeholder="请选择" class="layui-hide"/>

<script>
layui.use(['cascader'], function () {
    var $ = layui.jquery;
    var cascader = layui.cascader;

    cascader.render({
        elem: '#demoCascader1',
        data: [],
        renderFormat: function (labels, values) {
            return labels.join(' / ');  // 默认是用斜杠分割，可以自定义
        }
    });
});
</script>
```

6.3.4.搜索功能 :id=search

```
<input id="demoCascader1" placeholder="请选择" class="layui-hide"/>

<script>
layui.use(['cascader'], function () {
    var $ = layui.jquery;
    var cascader = layui.cascader;
```

```
    cascader.render({
        elem: '#demoCascader1',
        data: [],
        filterable: true   // 这个参数是开启搜索功能
    });

    // 自定义搜索
    cascader.render({
        elem: '#demoCascader1',
        data: [],
        filterable: true,
        reqSearch: function (keyword, callback, dataList) {
            // keyword是搜索的关键字, dataList是当前的全部数据集合
            $.get('xxx.json', { keyword: keyword }, function(res){
                callback(res);
            },'json')
        }
    });
});
</script>
```

搜索后端接口返回的数据格式为:

```
[
  {"value": "1,2", "label": "北京 / 王府井"},
  {"value": "1,3", "label": "北京 / 故宫"}
]
```

如果要标记关键字为红色： `<span class=\"search-keyword\">北京</span> / 王府井`

6.3.5.省市区级联选择 :id=city

```
<input id="demoCascader1" placeholder="请选择" class="layui-hide"/>

<script type="text/javascript" src="/assets/module/cascader/citys-data.js"></sc
ript>
<script>
layui.use(['cascader'], function () {
    var $ = layui.jquery;
    var cascader = layui.cascader;

    cascader.render({
        elem: '#demoCascader1',
        data: citysData,
        itemHeight: '250px',
        filterable: true
    });
});
</script>
```

省市区的数据已经封装好了，只需要引入数据的js即可，项目里面的数据value是区号，如果你的数据库直接存中文，可以使用下面的数据：

citys-data.js 未压缩(311kb)、citys-data.min.js 压缩(136kb)。

如果想去掉区域，只显示省和市，可以用js处理一下数据：

```
// 去掉第三级的数据
for (var i = 0; i < citysData.length; i++) {
    for (var j = 0; j < citysData[i].children.length; j++) {
        delete citysData[i].children[j].children;
    }
}
cascader.render({
    elem: '#demoCascader11',
    data: citysData
});
```

如果一个页面同时有省市和省市区选择，去掉第三级的时候应该克隆一下数据：

```
var citysData2 = admin.util.deepClone(citysData);
for (var i = 0; i < citysData2.length; i++) {
    for (var j = 0; j < citysData2[i].children.length; j++) {
        delete citysData2[i].children[j].children;
    }
}
cascader.render({
    elem: '#demoCascader11',
    data: citysData2
});
```

6.3.6.全部方法 :id=method

```
<input id="demoCascader1" placeholder="请选择" class="layui-hide"/>

<script>
layui.use(['cascader'], function () {
    var $ = layui.jquery;
    var cascader = layui.cascader;

    var ins1 = cascader.render({
        elem: '#demoCascader1',
        data: []
    });

    ins1.data;  // 获取当前的数据
    ins1.open();  // 展开
    ins1.hide();  // 关闭
    ins1.removeLoading();  // 移除加载中的状态
    ins1.setDisabled(true);  // 禁用或取消禁用
    ins1.getValue();  // 获取选中的数据值
    ins1.getLabel();  // 选取选中的数据名称
    ins1.setValue('1,2');   // 设置值
```

```
});
</script>
```

6.3.7.全部参数 :id=options

参数名称	介绍	默认值
elem	需要渲染的元素	
data	数据	
clearable	是否开启清除	true
clearAllActive	清除时清除所有列选中	false
trigger	次级菜单触发方式，可选'hover'	'click'
disabled	是否禁用	false
changeOnSelect	是否点击每一项都改变值	false
filterable	是否开启搜索功能	false
notFoundText	搜索为空是提示文字	'没有匹配数据'
itemHeight	下拉列表的高度	'180px'
reqData(values, callback, data)	异步获取数据的方法	
reqSearch(keyword, callback, dataList)	自定义搜索的方法	
renderFormat(labels, values)	选择后用于展示的函数	
onChange(values, data)	数据选择改变的回调	
onVisibleChange(isShow)	展开和关闭的回调	

6.4、标签输入框

6.4.1.快速使用 :id=start

```
<input id="demoTagsInput" value="辣妹子,大长腿" class="layui-hide"/>

<script>
    layui.use(['jquery', 'tagsInput'], function () {
        var $ = layui.jquery;

        // 输入框样式
        $('#demoTagsInput').tagsInput();

        // 无边框样式
        $('#demoTagsInput').tagsInput({skin: 'tagsinput-default'});

        // BackSpace键可删除标签
        $('#demoTagsInput').tagsInput({removeWithBackspace: true});

        // 输入列表提示
        $('#demoTagsInput').tagsInput({
            skin: 'tagsinput-default',
            autocomplete_url: '../../json/tagsInput.json'
        });

    });
</script>
```

只需要写一个input框，调用tagsInput方法渲染即可，回显数据写在value中用逗号分隔。

6.4.2.全部参数 :id=options

参数	说明	默认值
defaultText	提示文字	+请输入
skin	样式风格	
removeWithBackspace	回退键可删除已添加的标签	false
focusWithClick	点击已添加标签输入框获取焦点	true
autocomplete_url	自动提示接口url	
autocomplete	接口配置	

autocomplete参数：

```
$('#demoTagsInput').tagsInput({
    autocomplete_url: '../../json/tagsInput.json',
    autocomplete: {
        type: 'post',
        data: {
            access_token: 'xxxxx'
        }
    }
});
```

type是请求方式，默认是get请求，data是额外参数，请求autocomplete_url会传递 `name` 参数(输入框的值)。

更详细的使用文档可以参考jQuery-Tags-Input。

> 此插件基于 jQuery-Tags-Input 二次修改。

6.5、分割面板

6.5.1.快速使用 :id=start

```
<div class="split-group">
    <div class="split-item" id="demoSplit1">
        面板一
    </div>
    <div class="split-item" id="demoSplit2">
        面板二
    </div>
</div>

<script>
    layui.use(['Split'], function () {
        var $ = layui.jquery;
        var Split = layui.Split;

        // 水平分割，需要分割的元素(id)、默认大小(百分比)、最小值(单位px)
        Split(['#demoSplit1', '#demoSplit2'], {sizes: [25, 75], minSize: 100});
    });
</script>
```

6.5.2.垂直分割 :id=vertical

```
<div class="split-group-vertical">
    <div class="split-item" id="demoSplit3">
        面板一
    </div>
    <div class="split-item" id="demoSplit4">
        面板二
    </div>
</div>

<script>
    layui.use(['Split'], function () {
        var $ = layui.jquery;
        var Split = layui.Split;

        // 垂直分割
        Split(['#demoSplit3', '#demoSplit4'], {direction: 'vertical'});
    });
</script>
```

6.5.3.嵌套使用 :id=nest

```
<div class="split-group" style="height: 600px;">
    <div class="split-item" id="demoSplit8">
        <div class="split-group-vertical">
            <div class="split-item" id="demoSplit10">
                面板一
            </div>
            <div class="split-item" id="demoSplit11">
                面板二
            </div>
        </div>
    </div>
    <div class="split-item" id="demoSplit9">
        面板三
    </div>
</div>

<script>
    layui.use(['Split'], function () {
        var $ = layui.jquery;
        var Split = layui.Split;

        // 垂直水平分割
        Split(['#demoSplit8', '#demoSplit9'], {sizes: [25, 75], minSize: 100});
        Split(['#demoSplit10', '#demoSplit11'], {direction: 'vertical'});
    });
</script>
```

6.6、步骤条

6.6.1.快速使用 :id=start

step分布表单作为最早推出的组件之一，后续会重写，目前版本使用比较简单，暂时不写文档...

6.7、环形进度条

6.7.1.快速使用 :id=start

```
<div id="demoProgress1"></div>

<script>
layui.use(['CircleProgress'], function () {
    var CircleProgress = layui.CircleProgress;

    // 快速使用
    new CircleProgress('#demoProgress1', {
        max: 100,
        value: 20
    });
});
</script>
```

6.7.2.全部参数 :id=option

参数名称	说明	默认值
max	最大值	
value	当前值	
clockwise	是否顺时针方向	true
startAngle	起始角度	0
textFormat	文字样式	

文字样式：

- vertical　垂直
- percent　百分比
- value　只显示值
- valueOnCircle　值显示在进度条上
- none　不显示文字

自定义文字样式：

```
new CircleProgress('#demoProgress9', {
    max: 12,
    value: 9,
    textFormat: function (value, max) {
        return value + ' dots';
```

```
    }
});
```

6.7.3.自定义样式 :id=style

使用css自定义样式：

```
<div id="demoProgress1"></div>

<script>
layui.use(['CircleProgress'], function () {
    var CircleProgress = layui.CircleProgress;

    // 快速使用
    new CircleProgress('#demoProgress1', {
        max: 100,
        value: 20,
        textFormat: 'percent'
    });
});
</script>

<style>
/* 进度条选中的样式 */
#demoProgress1 .circle-progress-value {
    stroke-width: 8px;  /* 粗度 */
    stroke: #3FDABA;  /* 颜色 */
}

/* 进度条未选中的样式 */
#demoProgress1 .circle-progress-circle {
    stroke-width: 6px;
    stroke: #E0FAF1;
}
</style>
```

> 此插件是使用svg实现的，想要实现更丰富的样式，可先了解下svg

6.8、下拉选择树

6.8.1.快速使用 :id=start

后续推出...

7、扩展组件(进阶)

7.1、树形表格

7.1.树形表格 :id=treeTable

treeTable树形表格已进行了重写，模块名是treeTable，旧版模块名是treetable，不要搞混了，

查看演示。

7.1.1.快速使用 :id=start

方法渲染：

```
<table id="demoTb1"></table>

<script>
    layui.use(['treeTable'], function () {
        var $ = layui.jquery;
        var treeTable = layui.treeTable;
        var data = [{
            id: '1',
            name: 'xxx',
            createTime: '2019/11/18 10:44:00',
            children: [ {
                id: '1_1',
                name: 'xxx',
                createTime: '2019/11/18 10:44:00'
            }]
        }, {
            id: '2',
            name: 'xxx',
            createTime: '2019/11/18 10:44:00',
            children: [{
                id: '2_1',
                name: 'xxx',
                state: 0,
                createTime: '2019/11/18 10:44:00',
                children: []
            }]
        }];

        // 渲染表格
        var insTb = treeTable.render({
            elem: '#demoTb1',
            data: data,  // 数据
            tree: {
                iconIndex: 1  // 折叠图标显示在第几列
            },
```

```
            cols: [
                {type: 'numbers'},
                {field: 'id', title: 'ID'},
                {field: 'name', title: 'name', width: 160},
                {field: 'createTime', title: '创建时间', width: 180}
            ]
        });

    });
</script>
```

使用pid形式的数据：

```
var data = [{
    id: '1',
    name: 'xxx',
    createTime: '2019/11/18 10:44:00',
}, {
    pid: '1',
    id: '1_1',
    name: 'xxx',
    createTime: '2019/11/18 10:44:00'
},{
    id: '2',
    name: 'xxx',
    createTime: '2019/11/18 10:44:00',
},{
    pid: '2',
    id: '2_1',
    name: 'xxx',
    state: 0,
    createTime: '2019/11/18 10:44:00',
    children: []
}];

// 渲染表格
var insTb = treeTable.render({
    elem: '#demoTb1',
    data: data,  // 数据
    tree: {
        iconIndex: 1,  // 折叠图标显示在第几列
        isPidData: true  // 是否是pid形式数据
    },
    cols: [
        {type: 'numbers'},
        {field: 'id', title: 'ID'},
        {field: 'name', title: 'name', width: 160},
        {field: 'createTime', title: '创建时间', width: 180}
    ]
```

```
});
```

请求网络数据、懒加载：

```
var insTb = treeTable.render({
    elem: '#demoTb1',
    tree: {
        iconIndex: 1,  // 折叠图标显示在第几列
        idName: 'id',  // 自定义id字段的名称
        pidName: 'pid',  // 自定义标识是否还有子节点的字段名称
        haveChildName: 'haveChild',  // 自定义标识是否还有子节点的字段名称
        isPidData: true  // 是否是pid形式数据
    },
    cols: [
        {type: 'numbers'},
        {field: 'id', title: 'ID'},
        {field: 'name', title: 'name', width: 160},
        {field: 'createTime', title: '创建时间', width: 180}
    ],
    reqData: function(data, callback) {
        // 在这里写ajax请求，通过callback方法回调数据
        $.get('list.json', function (res) {
            callback(res.data);  // 参数是数组类型
        });
    }
});
```

!> 用法大部分与数据表格table模块一致，需要注意：cols是一维数组，不是二维数组。

7.1.2.基础参数一览表 :id=options

参数	类型	说明	示例值
elem	String/DOM	指定原始 table 容器的选择器或 DOM	'#demo'
cols	Array	设置表头。值是一维数组	详见表头参数
data	Array	直接赋值数据	[{}, {}, {}, {}, ...]
width	Number	设定容器宽度	350
height	Number/String	设定容器高度	300 / 'full-150'
cellMinWidth	Number	定义所有单元格的最小宽度	100
text	Object	自定义文本，如空数据提示等	详见自定义文本

skin	String	设定表格风格	line行边框、row列边框、nob无边框
even	Boolean	开启隔行背景	true/false
size	String	设定表格尺寸	sm 小尺寸、lg 大尺寸
tree	Object	设定树相关参数	详见树相关参数
style	String	设定容器的样式	'margin-top: 0;'
getThead	Function	自定义表头	详见自定义表头
reqData	Function	懒加载数据	详见懒加载数据

7.1.3.表头参数一览表cols :id=cols

参数	类型	说明	示例值
field	String	设定字段名	'username'
title	String	设定标题名称	用户名
width	Number	设定列宽，若不填写，则自动分配	150
minWidth	Number	单元格的最小宽度	100
type	String	设定列类型	checkbox复选框、radio单选框、numbers序号列、space空列
edit	String	单元格编辑类型	text（输入框）
style	String	自定义单元格样式	color: red;
align	String	单元格排列方式	center居中、right居右
templet	String	自定义列模板	详见自定义列模板
toolbar	String	绑定工具条模板	详见行工具事件

> templet和toolbar用法与数据表格table模块一致

7.1.4.树相关参数一览表tree :id=tree

参数	类型	说明	示例值
iconIndex	Number	图标列的索引	默认0
onlyIconControl	Boolean	仅允许点击图标折叠	默认false

arrowType	String	箭头类型	可选'arrow2'
getIcon	Function	自定义树形图标	详见自定义树形图标
isPidData	Boolean	是否是pid形式的数据，懒加载方式不需要	默认false
idName	String	设定id的字段名	默认'id'
pidName	String	设定pid的字段名，children形式数据不需要	默认'pid'
childName	String	设定children的字段名，pid形式数据不需要	默认'children'
haveChildName	String	设定是否有子集的字段名，用于懒加载	默认'haveChild'
openName	String	设定是否默认展开的字段名	默认'open'

7.1.5.自定义文本text :id=text

目前只支持自定义空数据提示：

```
treeTable.render({
    text: {
        none: '<div style="padding: 15px 0;">哎呀，一条数据都没有~</div>'
    }
});
```

7.1.6.自定义表头 :id=tb_head

treeTable实现复杂表头的做法与table模块不同，方法如下：

```
treeTable.render({
    getThead: function() {
        return '<tr><td colspan="6">我是表头</td></tr>';
    }
});
```

就是这么简单粗暴，直接返回表头的html即可，不用在cols里面纠结半天，所以treeTable的cols是一维数组。

7.1.7.自定义树形图标 :id=tree_icon

内置了两种风格图标，你也可以很灵活的自己扩展风格：

```
treeTable.render({
    tree: {
        arrowType: 'arrow2',    // 自定义箭头风格
        getIcon: function(d) {  // 自定义图标
            // d是当前行的数据
            if (d.haveChild) {  // 判断是否有子集
                return '<i class="ew-tree-icon ew-tree-icon-folder"></i>';
            } else {
                return '<i class="ew-tree-icon ew-tree-icon-file"></i>';
            }
        }
    }
});
```

ew-tree-icon-folder(文件夹)和ew-tree-icon-file(文件)这两个class是treeTable内置的，

你可以换成其他clss自己加样式，ew-tree-icon这个class是必须的。

!> 判断是否有子集d.haveChild这个写法根据你的实际情况写，如果是children形式数据可以写

(d.children&&d.children.length>0)

7.1.8.懒加载数据 :id=req_data

实现reqData参数即可实现懒加载数据：

```
treeTable.render({
    elem: '#demoTable',
    reqData: function(data, callback) {
        // data是当前行的数据，通过callback回调子集数据
        callback([]);
    },
    tree: {
        iconIndex: 2,  // 折叠图标显示在第几列
        idName: 'id',  // 自定义id字段的名称
        haveChildName: 'haveChild'  // 自定义标识是否还有子节点的字段名称
    }
});
```

你可以在reqData里面写ajax请求：

```
treeTable.render({
    reqData: function(data, callback) {
        var pid = data?data.id:'';
        $.get('list.json?pid='+pid, function (res) {
            callback(res.data);
```

```
        });
    }
});
```

json数据参考格式：

```
{"code": 200, "data": [{"id": "1", "name": "xxx", "haveChild": true}]}
```

通过haveChild标识是否还有子节点，id也是必须的字段，这两个字段的名字可以在tree参数里面自定义。

!> reqData这个方法里面也可以一次性把所有数据都返回，也可以懒加载。

7.1.9.默认展开 :id=open

通过在数据里面增加open字段来控制是否默认展开：

```
var data = [{
    id: '1',
    name: 'xxx',
    open: true,  // 默认展开
    children: [ {
        id: '1_1',
        name: 'xxx',
        createTime: '2019/11/18 10:44:00'
    }]
}, {
    id: '2',
    name: 'xxx',
    open: true,  // 默认展开
    children: [{
        id: '2_1',
        name: 'xxx',
        state: 0,
        createTime: '2019/11/18 10:44:00',
        children: []
    }]
}];
```

字段名字也可以自定义：

```
treeTable.render({
    tree: {
        openName: 'open',    // 自定义默认展开的字段名
    }
});
```

7.1.10.事件监听 :id=event

监听工具条点击事件：

```
treeTable.on('tool(test)', function(obj){
    var data = obj.data;  // 获得当前行数据
    var event = obj.event; // 获得lay-event对应的值
    obj.del(); // 删除对应行，并更新缓存
    // 同步更新缓存对应的值
    obj.update({
        username: '123',
        title: 'xxx'
    });
});
```

监听复选框选择：

```
treeTable.on('checkbox(test)', function(obj){
    console.log(obj.checked);  // 当前是否选中状态
    console.log(obj.data);  // 选中行的相关数据
    console.log(obj.type);  // 如果触发的是全选，则为：all，如果触发的是单选，则为：one
});
```

监听单元格编辑：

```
treeTable.on('edit(test)', function(obj){
    console.log(obj.value); //得到修改后的值
    console.log(obj.field); //当前编辑的字段名
    console.log(obj.data); //所在行的所有相关数据
});
```

监听行单双击事件：

```
// 监听行单击事件
treeTable.on('row(test)', function(obj){
    console.log(obj.tr) //得到当前行元素对象
    console.log(obj.data) //得到当前行数据
    // obj.del(); // 删除当前行
    // obj.update(fields) // 修改当前行数据
});

// 监听行双击事件
treeTable.on('rowDouble(test)', function(obj){
    // obj 同上
});
```

监听单元格单双击事件：

```
// 监听行单击事件
treeTable.on('cell(test)', function(obj){
    console.log(obj.field); //当前单元格的字段名
    console.log(obj.data) // 得到当前行数据
});

// 监听行双击事件
treeTable.on('cellDouble(test)', function(obj){
    // obj 同上
});
```

7.1.11.其他方法 :id=other

```
// pid形式数据转children形式数据
treeTable.pidToChildren(data, idName, pidName, childName);
```

7.1.12.实例方法 :id=method

```
var insTb = treeTable.render({ });

// 刷新
insTb.reload(options);   // 重载表格
insTb.refresh();  // 刷新(异步模式)
insTb.refresh(id);  // 刷新指定节点下的数据
insTb.refresh(id, data);  // 刷新指定节点下的数据为data

// 复选框
insTb.checkStatus();  // 获取选中数据
insTb.setChecked(['1','2']);  // 设置选中数据
insTb.removeAllChecked();  // 移除全部选中

// 折叠/展开
insTb.expand(id);  // 展开指定节点
insTb.fold(id);  // 折叠指定节点
insTb.expandAll();  // 展开全部
insTb.foldAll();  // 折叠全部

// 搜索
insTb.filterData('keywords');   // 根据关键字模糊搜索
insTb.filterData(['1','2']);   // 根据id搜索
insTb.clearFilter();   // 清除搜索

// 更新数据(只更新数据，不更新界面)
insTb.del(id);   // 根据id删除
```

```
insTb.update(id, fields);  // 根据id更新
```

7.2、文件选择器

7.2.1.快速使用 :id=start

```
layui.use(['fileChoose'], function () {
    var fileChoose = layui.fileChoose;

    fileChoose.open({
        fileUrl: '',  // 文件查看的url
        listUrl: '../template/file/files.json',  // 文件列表的url
        where: {
          access_token: 'xxxxxx'
        },
        num: 3,  // 最多选择数量
        dialog: {
            offset: '60px'
        },
        onChoose: function (urls) {
            layer.msg('你选择了：' + JSON.stringify(urls), {icon: 1});
        }
    });
});
```

7.2.2.全部参数 :id=options

参数	描述	默认值
fileUrl	文件查看的url	
listUrl	文件列表的url	
where	文件列表请求参数	{}
num	文件选择的数量	1
onChoose	选择后回调	
upload	文件上传配置(同layui配置)	{}
dialog	弹窗配置(同layui配置)	{}
menu	点击弹出的菜单	数组类型
menuClick	菜单点击事件处理	
response	接口数据格式化	

菜单配置及点击事件：

```
fileChoose.open({
```

```
    menu: [{
        name: '预览',
        event: 'preview'
    }, {
        name: '复制',
        event: 'copy'
    }, {
        name: '<span style="color: red;">删除</span>',
        event: 'del'
    }],
    menuClick: function(event, item) {
        // event  事件名称
        // item   当前数据
    }
});
```

name菜单项名称，event点击事件名称

接口数据格式化：

```
fileChoose.open({
    response: {
        method: 'get',  // 请求方式
        code: 0,  // 成功码，默认200
        name: 'name',  // 文件名称字段名称
        url: 'url',  // 文件url字段名称
        smUrl: 'smUrl',  // 文件缩略图字段名称
        isDir: 'isDir',  // 是否是文件夹字段名称,boolean类型
        dir: 'dir'  // 当前文件夹参数名称
    }
});
```

接口数据返回的格式需要为：

```
{
    "code": 200,
    "msg": "请求成功",
    "data": [
        {
            "name": "图片一",
            "url": "2019/07/11/001.png",
            "smUrl": "sm/2019/07/11/001.png",
            "isDir": false
        }
    ]
}
```

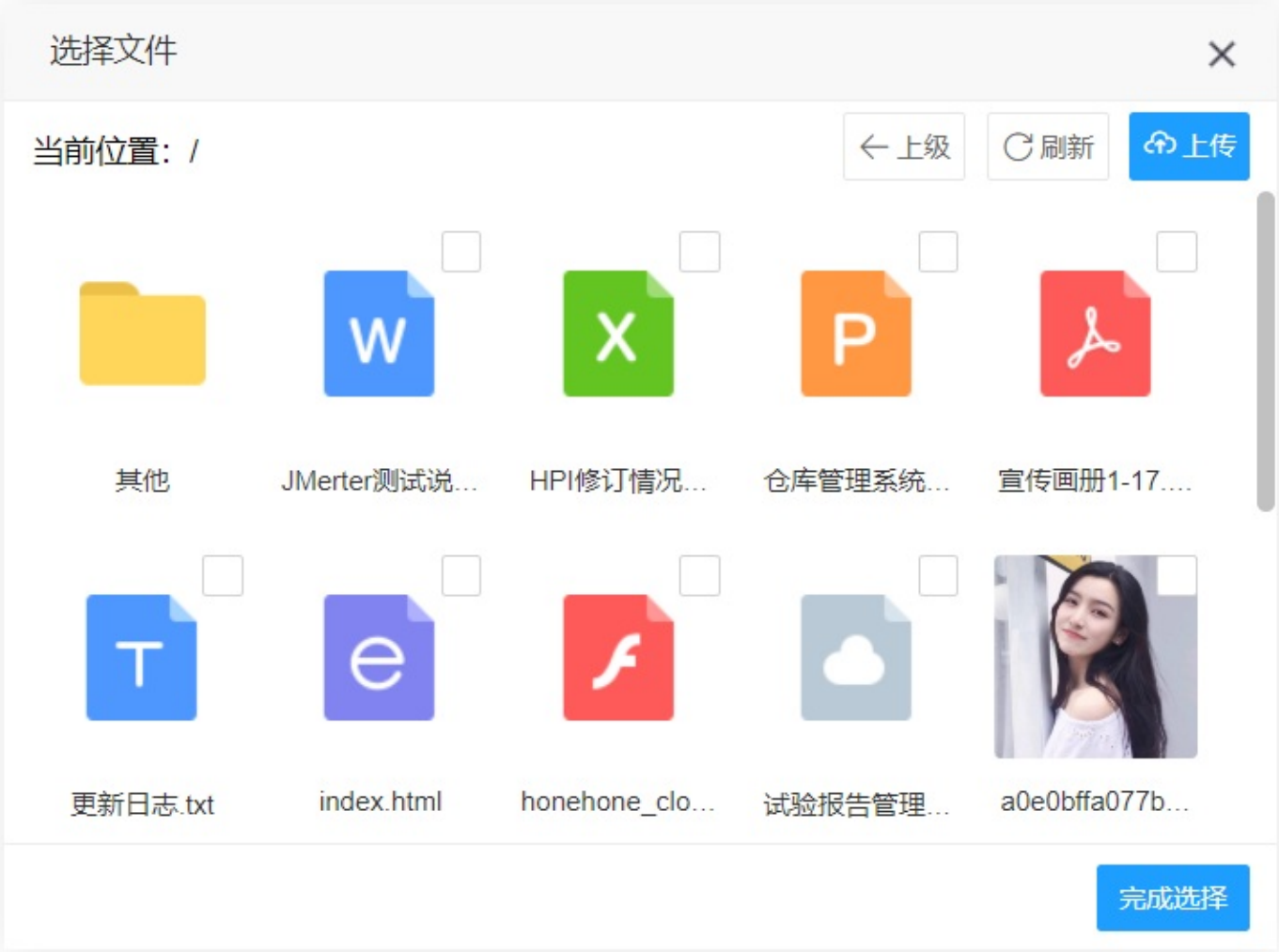
code、msg、data是必须按这个名字的，name、url、smUrl、isDir这几个字段的名称可以通过response参数配置，也可以加其他字段，
比如id、create_time等，这些字段会在菜单点击事件和选择回调事件中返回。

如果你的接口返回的数据不是code、msg，是其他的，比如status、message，可以使用parseData参数格式化：

```
fileChoose.open({
    response: {
        parseData: function(res){
            return {
                code: res.status,
                msg: res.message,
                data: res.list
            }
        }
    }
});
```

如果是文件夹，点击文件夹会重新请求接口，并且传递文件夹的名称，传递的字段名称可以通过response.dir修改。

不同文件显示不同的图标是前端根据文件url的后缀名称来判断的，在之前版本是服务器根据文件的content-type判断的。


选择文件
当前位置：/
上级
刷新
上传
其他
JMerter测试说...
HPI修订情况...
仓库管理系统...
宣传画册1-17....
更新日志.txt
index.html
honehone_clo...
试验报告管理...
a0e0bffa077b...
完成选择

7.3、表格扩展tableX

7.3.1.全部方法 :id=method

方法	参数	描述
merges(tableId, indexs, fields)	见单独说明	合并单元格
bindCtxMenu(tableId, items)	见单独说明	给表格行绑定鼠标右键
render(object)	同layui表格	渲染表格，带后端排序功能
renderFront(object)	同layui表格	渲染表格，前端分页、排序、搜索
exportData(object)	见单独说明	导出任意数据为excel
loadUrl(object, callback)	见单独说明	请求表格数据
parseTbData(cols, dataList, overwrite)	见单独说明	获取解析templet后数据
filterData(dataList, searchName, searchValue)	见单独说明	前端搜索数据

查看在线演示

7.3.2.合并单元格 :id=merge

```
layui.use(['tableX'], function () {
    var tableX = layui.tableX;

    table.render({
        elem: '#xTable2',
        url: '../../json/tablex1.json',
        cols: [[
            {type: 'numbers'},
            {field: 'parentName', title: '模块名称', sort: true},
            {field: 'authorityName', title: '菜单名称', sort: true}
        ]],
        done: function () {
            tableX.merges('xTable2', [1]);  // 在done回调里面调用
        }
    });
});
```

参数说明：

```
tableX.merges('xTable2', [1]);  // 合并第2列相同的单元格
tableX.merges('xTable2', [1], ['parentName']);  // 合并第2列相同的单元格
tableX.merges('xTable2', [1, 2]);   // 合并第2、3列相同的单元格
tableX.merges('xTable2', [1, 2], ['parentName', 'authorityName']);   // 合并第2
、3列相同的单元格

tableX.merges('xTable2', [1, 2], false);   // 在后面加一个false可解决排序冲突的问题
```

- 参数一　必填　表格的lay-filter
- 参数二　必填　要合并列的索引，数组类型
- 参数三　非必填　根据数据字段判断是否相同，为空时根据单元格的html内容判断

7.3.3.行绑定鼠标右键 :id=ctx

```
table.render({
    elem: '#xTable3',
    url: '../../json/user.json',
    cols: [[
        {field: 'nickName', title: '用户名', sort: true},
        {field: 'sex', title: '性别', sort: true}
    ]],
    done: function () {
        // 在done回调里面绑定
        tableX.bindCtxMenu('xTable3', [{
            icon: 'layui-icon layui-icon-edit',
            name: '修改此用户',
            click: function (d) {
                layer.msg('点击了修改，userId：' + d.userId);
            }
        }, {
            icon: 'layui-icon layui-icon-close text-danger',
            name: '<span class="text-danger">删除此用户</span>',
            click: function (d) {
                layer.msg('点击了删除，userId：' + d.userId);
            }
        }]);
    }
});
```

- 参数一　表格的lay-filter
- 参数二　右键菜单
  - icon　图标

- name　　标题
- click　　点击事件，d是当前行的数据

7.3.4.后端排序 :id=render

```
tableX.render({
    elem: '#xTable3',
    url: '../../json/user.json',
    cols: [[
        {type: 'numbers'},
        {field: 'nickName', title: '用户名', sort: true},
        {field: 'sex', title: '性别', sort: true}
    ]]
});
```

仅仅是把 `table.render` 换成 `tableX.render` 就会在点击排序时自动传递 `sort` 和 `order` 参数，例如：user?page=1&limit=10&sort=sex&order=asc。

- sort　　排序字段，值是点击排序列的field
- order　　排序方式，升序是asc，降序是desc

> 注意：如果写了sort: true开启排序，一定要写field: 'xxx'。

7.3.5.前端分页排序 :id=render_f

```
tableX.renderFront({
    elem: '#xTable1',
    url: '../../json/userAll.json',
    page: { groups: 6 },
    cols: [[
        {type: 'checkbox'},
        {field: 'nickName', title: '用户名', sort: true},
        {field: 'sex', title: '性别', sort: true}
    ]]
});
```

仅仅是把 `table.render` 换成 `tableX.renderFront` 就可以有前端分页和排序功能了，参数跟layui表格的参数一摸一样，url方式你的接口可以返回全部数据，前端来分页，也支持data方式。

前端模糊搜索：

```
<input tb-search="xTable1" class="layui-input icon-search" type="text"/>
```

页面中加入上面的输入框，通过 `tb-search` 关联表格，就实现了对表格的模糊搜索功能，你可以对该input

加任意样式，放在任意位置。

还可以增加name属性来设置搜索时只搜索某些字段，多个字段通过逗号分隔：

```
<input tb-search="xTable1" name="sex,phone" class="layui-input icon-search" typ
e="text"/>
```

刷新功能：

```
<!-- 通过tb-refresh绑定刷新按钮 -->
<button tb-refresh="xTable1" class="layui-btn">刷新</button>
```

也可以通过js刷新：

```
var insTb = tableX.renderFront('.....省略');

insTb.reloadUrl();  // url方式刷新

insTb.reloadData({data: dataList, page: {curr: 1}});  // data方式的刷新
```

> 注意：url方式的前端分页使用reloadUrl()方法，reloadUrl()方法暂时不支持加参数，reloadData方法跟table.reload()方法参数一致。

前端排序：

前端排序如果有field字段，会根据field字段的值来排序，如果没有field有templet会根据templet转换后的值排序，

templet可能会返回表单元素，比如switch开关等，这些元素根本无法用来排序，那么可以通过 `export-show` 和 `export-hide` 这两个东西来控制。

```
<!-- 开关，state=0开关打开,state=1开关关闭 -->
<script type="text/html" id="tableState">
    <input type="checkbox" lay-skin="switch" lay-text="正常|锁定" lay-filter="ck
State" value="{{d.userId}}" {{d.state==0?'checked':''}}/>
    <div class="export-show">{{d.state==0?'正常':'锁定'}}</div>
</script>
<!-- 图标，state=0显示ok的图标，state=1显示叉叉的图标 -->
<script type="text/html" id="tableState">
    <div class="export-hide">{{d.state==0?'<i class="layui-icon layui-icon-ok">
</i>':'<i class="layui-icon layui-icon-close"></i>'}}</div>
    <div class="export-show">{{d.state==0?'正常':'锁定'}}</div>
</script>
<!-- 图标和开关没法用于排序，可以写两份，一份用于表格展示，一份用于排序和导出功能 -->
```

- `export-hide` 用于表格展示，但是对排序和导出时屏蔽
- `export-show` 用于排序和导出功能获取单元格数据，但是对表格展示时屏蔽

> 排序建议最好是通过field来根据数据的字段排序，field不支持d.xxx.xxx这种嵌套的对象，就需要用上面的方法

7.3.6.导出数据 :id=export

```
tableX.exportData({
    cols: insTb.config.cols,   // 表头配置
    data: table.cache.xTable3,  // 数据，支持url方式
    fileName: '用户表'          // 文件名称
});
```

参数	必填	说明	默认
cols	是	表头配置	
data	是	导出的数据，支持数组和string的url	
fileName	否	导出的文件名称	table
expType	否	导出的文件类型	xls
option	否	url方式的配置	

如果data是string类型会把data当url请求数据，option是请求的配置，跟表格的配置一样，配置method、where、headers等，

接口返回的格式也要跟表格一样包含code、count、data等信息。

cols的配置也跟表格一样，是一个多维数组，可以通过 `insTb3.config.cols` 来获取表格的cols，也可以重写。

导出的数据会包含templet转换：

```
tableX.renderFront({
    elem: '#xTable1',
    url: '../../json/userAll.json',
    page: { groups: 6 },
    cols: [[
        {type: 'checkbox'},
        {field: 'nickName', title: '用户名', sort: true},
        {field: 'sex', title: '性别', sort: true},
        {
            title: '状态', templet: function (d) {
                var stateStr = ['正常', '冻结', '欠费'];
                return stateStr[d.state];
```

```
            }
        }
    ]]
});
```

像上面这种state数据存的是0和1，但是显示是文字，导出数据会自动转成文字(layui2.5后自带的带出也包含此功能)。

如果列是表单元素，比如switch开关，导出的方法：

```
<!-- 状态列 -->
<script type="text/html" id="tableState">
    <input type="checkbox" lay-skin="switch" lay-text="正常|锁定" lay-filter="ck
State" value="{{d.userId}}" {{d.state==0?'checked':''}}/>
    <div class="export-show">{{d.state==0?'正常':'锁定'}}</div>
</script>
<script>
layui.use(['tableX'], function () {
    var tableX = layui.tableX;

    tableX.render({
        elem: '#xTable3',
        url: '../../json/user.json',
        cols: [[
            {field: 'nickName', title: '用户名', sort: true},
            {field: 'phone', title: '手机号', sort: true},
            {field: 'state', templet: '#tableState', title: '状态', sort: true}
        ]]
    });
});
</script>
```

通过加一个 `export-show` 类可以保证里面的内容只作为导出的时候用，不作为表格展示，还可以加 `export-hide` 来设置导出的时候屏蔽的内容。

```
<!-- 图标，state=0显示ok的图标，state=1显示叉叉的图标 -->
<script type="text/html" id="tableState">
    <div class="export-hide">{{d.state==0?'<i class="layui-icon">&#xe605;</i>':
'<i class="layui-icon">&#x1006;</i>'}}</div>
    <div class="export-show">{{d.state==0?'正常':'锁定'}}</div>
</script>
<!-- 不可能把图标的unicode字符导出吧，那就写两份，一份用于表格展示，一份用于排序和导出功能 -
->
```

- `export-hide` 用于表格展示，但是对排序和导出时屏蔽
- `export-show` 用于排序和导出功能获取单元格数据，但是对表格展示时屏蔽

7.3.7.导出全部、搜索 :id=export_adv

导出当前页数据：

```
tableX.exportData({
    cols: insTb.config.cols,
    data: table.cache.xTable3,
    fileName: '用户表'
});
```

insTb是表格渲染后的实例，table.cache获取表格当前页的数据。

导出全部数据：

```
tableX.exportData({
    cols: insTb.config.cols,
    data: 'listAll.json',
    fileName: '用户表'
});
```

导出全部数据后端再写一个查询全部的接口，导出时只需要把data写接口地址即可。

导出搜索后的全部数据：

```
tableX.exportData({
    cols: insTb.config.cols,
    data: 'listAll.json',
    option: {
        where: lastWhere
    },
    fileName: '用户表'
});
```

在options里面加一个where把搜索条件传给后端，搜索条件可以这样获取：

```
var lastWhere;
// 搜索
form.on('submit(formSubSearchUser)', function (data) {
    lastWhere = data.field;
    insTb.reload({where: data.field}, 'data');
});
```

在表格的搜索里面把搜索条件传给lastWhere。

cols也可以重写：

```
tableX.exportData({
    cols: [[
        {field: 'username', title: '账号'},
        {field: 'nickName', title: '用户名'}
    ]],
    data: table.cache.xTable3,
    fileName: '用户表'
});
```

7.3.8.后端导出 :id=export_back

后端导出一般是只用window.open(url)即可，如果要传递参数，并且要post提交，那就麻烦了，tableX进行了完美的封装：

```
tableX.exportDataBack('user/export', {sex: '男'});

tableX.exportDataBack('user/export', {sex: '男'}, 'post');
```

- 参数一：　导出的url
- 参数二：　参数
- 参数三：　请求方式

如果是get方式，会使用?和&拼接参数，如果是post方式，会创建一个隐藏的表单来提交，如果你的参数有复杂的类型，比如json格式，建议用post的形式

7.4、表单扩展formX

7.4.1.验证规则 :id=start

规则	描述	提示信息
phoneX	手机号	请输入正确的手机号
emailX	邮箱	邮箱格式不正确
urlX	网址	链接格式不正确
numberX	数字	只能填写数字
dateX	日期	日期格式不正确
identityX	身份证	请输入正确的身份证号
psw	密码	密码必须5到12位，且不能出现空格
equalTo	重复	两次输入不一致
digits	整数	只能输入整数
digitsP	正整数	只能输入正整数
digitsN	负整数	只能输入负整数
digitsPZ	非负整数	只能输入正整数和0
digitsNZ	非正整数	只能输入负整数和0
h5	兼容h5的规则	

使用示例：

```
<form class="layui-form">
    <input class="layui-input" placeholder="请输入手机号"
           lay-verType="tips" lay-verify="phoneX"/>
    <input class="layui-input" placeholder="请输入手机号"
           lay-verType="tips" lay-verify="required|phoneX"/>
    <input class="layui-input" placeholder="请输入整数"
           lay-verType="tips" lay-verify="digits"/>
    <input class="layui-input" placeholder="请输入正整数"
           lay-verType="tips" lay-verify="digitsP"/>
</form>
```

equalTo用法：

```
<form class="layui-form">
    <input id="demoPsw" class="layui-input" placeholder="请输入密码"
```

```
            lay-verType="tips" lay-verify="required|psw"/>
    <input class="layui-input" placeholder="请再次输入密码"
           lay-verType="tips" lay-verify="equalTo" lay-equalTo="#demoPsw"
           lay-equalToText="两次输入密码不一致"/>
</form>
```

equalTo用来验证两次输入是否一致。

属性	描述
lay-equalTo	关联输入框的dom选择器
lay-equalToText	自定义提示文本

h5用法：

属性	描述
minlength	最少输入字符长度
maxlength	最多输入字符长度
min	最小输入数值
max	最大输入数值

```
<form class="layui-form">
    <input class="layui-input" placeholder="最少输入5个字符" minlength="5"
           lay-verType="tips" lay-verify="required|h5"/>
    <input class="layui-input" placeholder="最多输入10个字符" maxlength="10"
           lay-verType="tips" lay-verify="h5"/>
    <input class="layui-input" type="number" placeholder="值只能在-9到9之间" min=
"-9" max="9"
           lay-verType="tips" lay-verify="required|numberX|h5"/>
</form>
```

> phoneX、emailX等与layui自带phone、email等的区别是如果没有输入不会验证，输入了才验证格式。

7.4.2.扩展方法 :id=method

方法	描述
formVal(filter, object)	赋值表单，解决top.layui.form.val()方法无效的bug

使用方法：

```
layui.use(['formX'],function(){
    var formX = layui.formX;
```

```
    formX.val('userForm', {name: 'user01'});  // 赋值表单，支持top.formX.val()用法
});
```

7.5、数据列表dataGrid

7.5.1.快速使用 :id=start

大多数的后台系统页面全是数据表格，表格看多了，难免喜欢其他布局，例如网格、瀑布流、卡片列表等形式，

如果使用这些布局，分页、自动渲染、搜索、排序等功能都需要自己实现，相比数据表格，大大增加工作量，dataGrid插件就是对非表格形式的列表页面实现自动分页、自动渲染的功能。

```
<div id="demoGrid"></div>

<!-- 模板 -->
<script type="text/html" id="demoGridItem">
    <div>
        <h2>{{d.title}}</h2>
        <p>{{d.content}}</p>
        <a lay-event="edit">修改</a>
    </div>
</script>

<script>
layui.use(['dataGrid'], function () {
    var dataGrid = layui.dataGrid;

    // 渲染
    var ins = dataGrid.render({
        elem: '#demoGrid',  // 容器
        templet: '#demoGridItem',  // 模板
        data: '../../../json/data-grid.json',  // 数据接口
        page: {limit: 5},  // 开启分页
        onItemClick: function (obj) {
            // item点击事件监听
        },
        onToolBarClick: function (obj) {
            // item子元素点击事件监听
        }
    });

});
</script>
```

使用方法与数据表格table类似。

7.5.2.事件监听 :id=event

```
dataGrid.render({
    // item点击事件监听
    onItemClick: function (obj) {
        var index = obj.index;
        console.log('点击了第' + (index + 1) + '个');
    },
    onToolBarClick: function (obj) {
        // item子元素点击事件监听
        var event = obj.event;
        console.log(event);
    }
});
```

onToolBarClick就是item里面加了lay-event属性的子元素的点击事件。

obj详细介绍：

```
obj.data;           // 当前操作的数据对象
obj.index;          // 当前操作的数据索引
obj.elem;           // 当前操作的dom元素
obj.event;          // 当前操作的lay-event值
obj.e;              // 原生点击事件的e

obj.del();          // 删除当前item
obj.update(o);      // 修改当前item
```

obj.update()用法：

```
dataGrid.render({
    onToolBarClick: function (obj) {
        var data = obj.data;
        data.title = '我是新的值';
        obj.update(data);
    }
});
```

7.5.3.分页功能 :id=page

```
dataGrid.render({
    elem: '#demoGrid',  // 容器
    templet: '#demoGridItem',  // 模板
    data: '../../../json/data-grid.json',  // 数据接口
    page: {limit: 5}  // 开启分页
});
```

page参数同layui分页组件的参数一致，前往查看。

7.5.4.加载更多功能 :id=load_more

```
dataGrid.render({
    elem: '#demoGrid',  // 容器
    templet: '#demoGridItem',  // 模板
    data: '../../../json/data-grid.json',  // 数据接口
    loadMore: {limit: 5}  // 开启加载更多
});
```

loadMore的可选配置：

参数	说明	默认值
class	自定义class	''
limit	每页多少条	10
text	显示文字	'加载更多'
loadingText	加载中文字	'加载中...'
noMoreText	无数据文字	'没有更多数据了~'
errorText	加载失败文字	'加载失败，请重试'

> loadMore和page只能二选一

7.5.5.重载功能 :id=reload

```
var ins = dataGrid.render({});

ins.reload();  // 重载

ins.reload({page: {curr: 2} });  // 跳到第二页
```

7.5.6.渲染完成的回调 :id=done

```
dataGrid.render({
    done: function(dataList, curr, count) {
        dataList;  // 当前页数据
        curr;      // 当前第几页
        count;     // 总数量
    }
});
```

7.5.7.自动渲染 :id=auto_render

```
<div id="demoGrid" lay-data="{url: 'json/data-grid.json', page: {limit: 5} }" d
ata-grid>
    <script type="text/html" data-grid-tpl>
        <div>
            <h2>{{d.title}}</h2>
            <p>{{d.content}}</p>
            <a lay-event="edit">修改</a>
        </div>
    </script>
</div>
```

通过添加data-grid和data-grid-tpl属性，dataGrid组件便会自动渲染，通过lay-data属性进行参数配置。

7.5.8.绑定事件 :id=bind_event

```
// 绑定item事件
dataGrid.onItemClick('demoGrid', function(obj){
    console.log(obj);
});

// 绑定item子元素事件
dataGrid.onToolBarClick('demoGrid', function(obj){
    console.log(obj);
});
```

参数一是容器的id，这种方式绑定事件的优先级比直接在渲染的时候绑定事件低。

7.6、鼠标右键

7.6.1.快速使用 :id=start

```
layui.use(['contextMenu'], function () {
    var contextMenu = layui.contextMenu;

    // 重写整个页面右键菜单
    contextMenu.bind('html', [{
        icon: 'layui-icon layui-icon-snowflake',
        name: '菜单一',
        click: function (e, event) {
            // 通过$(event.currentTarget)可获取事件触发的元素
            alert('点击了菜单一');
        }
    }, {
        name: '菜单二',
        click: function (e) {
            alert('点击了菜单二');
        }
    }]);
});
```

- 参数一： 绑定元素；
- 参数二：菜单数组；

菜单数组可支持无限极：

```
[{
    icon: 'xxxxx',
    name: '菜单三',
    subs: [{
        icon: 'xxxxx',
        name: '子菜单一',
        click: function (e, event) {
            alert('点击了子菜单一');
        }
    }]
}]
```

- icon： 图标
- name： 菜单名
- click： 点击事件
- subs： 子菜单(支持无限极)

!> click事件里面的e是右键菜单的事件对象，event才是绑定的目标元素的事件对象

7.6.2.自定义使用 :id=div

```
// 用于点击事件
$('#btn').click(function (e) {
    var x = $(this).offset().left;
    var y = $(this).offset().top + $(this).outerHeight();
    contextMenu.show([{
        name: '按钮菜单一',
        click: function () {
        }
    }], x, y, e);
    e.preventDefault();
    e.stopPropagation();
});
```

你可以自己绑定事件，通过show方法显示出来，contextMenu.show(item, x, y, e)方法参数说明：

- 参数一： 菜单数组
- 参数二： x坐标；
- 参数三： y坐标；
- 参数四： e

7.6.3.动态元素绑定 :id=show

对于动态生成的元素可以使用事件委托的方式来绑定：

```
// 对.btn元素绑定鼠标右键
$ (document). bind('contextmenu', '.btn', function (e) {
    contextMenu.show([{
        icon: 'layui-icon layui-icon-set',
        name: '删除',
        click: function (e, event) {
            layer.msg('点击了删除');
        }
    }],e. clientX, e. clientY, e);
    return false;
});
```

你需要自己写contextmenu事件绑定，然后使用contextMenu的show方法显示出来。

7.7、打印插件

7.7.1.打印当前页面 :id=print_this

```
printer.print();

printer.print({
    hide: ['.layui-btn', '#btn01'],  // 打印时隐藏的元素
    horizontal: true,  // 是否横向打印
    blank: true,  // 是否打开新页面打印
    close: true,  // 如果是打开新页面，打印完是否关闭
    iePreview: true  // 是否兼容ie打印预览
});
```

> 参数都是可以选参数，默认值已经符合大多数需求。

7.7.2.设置不打印的元素 :id=hide

```
<div class="hide-print">非打印内容</div>
```

> 加 `hide-print` 这个class即可，可以跟 `hide` 参数叠加使用。

7.7.3.打印自定义内容 :id=print_html

```
var pWindow = printer.printHtml({
    html: '<span>xxxx</span>',  // 要打印的内容
    horizontal: true,  // 是否横向打印
    blank: true,  // 是否打开新页面打印
    close: true,  // 如果是打开新页面，打印完是否关闭
    iePreview: true,  // 是否兼容ie打印预览
    print: true  // 如果是打开新窗口是否自动打印
});
```

打印表格时可以给table加 `print-table` 这个class来设置表格的边框样式：`<table class="print-table"></table>` 。

> 如果print为false关闭自动打印，可以使用pWindow.print()触发打印。

7.7.4.分页打印 :id=print_page

```
printer.printPage({
```

```
    htmls: [
        '<span>xxxx</span>',    // 要打印的内容
        '<span>xxxx</span>'
    ],
    style: '<style> span { color: red; } </style>',  // 页面样式
    horizontal: true,  // 是否横向打印
    padding: undefined,  // 页面间距，例如写'10px'
    debug: false,  // 调试模式
    blank: true,  // 是否打开新页面打印
    close: true,  // 如果是打开新页面，打印完是否关闭
    iePreview: true,  // 是否兼容ie打印预览
    print: true  // 如果是打开新窗口是否自动打印
});
```

参数都是可选参数：

- htmls 数组，代表每一页；
- style 样式，也可以写 `<link rel="stylesheet">` 的形式；
- padding 间距，undefined表示使用打印机默认间距，如果自定义间距，打印机间距要选择无；
- debug 调试模式，每一页会加红色边框，便于调试大小；

> `padding` 为undefined表示使用打印机默认间距，插件内部会根据不同浏览器做兼容处理，因为不同浏览器的默认间距不同，
>
> 目前对谷歌、IE、Edge、火狐做了兼容处理，safari、欧朋后续会考虑。

7.7.5.拼接html :id=make_html

```
var htmlStr = printer.makeHtml({
    title: '网页标题',
    style: '<link rel="stylesheet" href="layui.css"><script type="text/javascri
pt" src="layui.js"><\/script>',
    body: $('#xxx').html()
});
```

7.7.6.其他方法 :id=method

```
printer.isIE();
printer.isEdge();
printer.isFirefox();
```

7.7.7.打印任意url :id=print_url

目前没有提供直接的方法，可以使用ajax请求到url的html内容，在使用printHtml方法打印：

```
$.ajax({
    url: 'xxx.html',
    type: 'get',
    dataType: 'html',
    success: function (res) {
        printer.printHtml({
            html: res,  // 要打印的内容
            horizontal: true  // 是否横向打印
        });
    }
});
```

> 另外js直接控制横向打印只支持谷歌内核的浏览器，火狐不支持js调用打印预览，
> 火狐打印预览可以通过设置 `blank:true,print:false` 然后手动选择打印预览。

8、第三方插件

8.1.鼠标滚轮监听 :id=mousewheel

模块名：mousewheel，使用方式：

```
layui.use(['mousewheel'], function () {
    var $ = layui.jquery;

    // 滚动监听
    $('#xxx').on('mousewheel', function (event) {
        console.log(event.deltaX, event.deltaY, event.deltaFactor);
        event.stopPropagation();  // 阻止事件冒泡
        event.preventDefault();  // 阻止默认事件
    });

});
```

event对象中可以获取如下三个属性值：

- deltaX：值为负的（-1），则表示滚轮向左滚动。值为正的（1），则表示滚轮向右滚动。
- deltaY：值为负的（-1），则表示滚轮向下滚动。值为正的（1），则表示滚轮向上滚动。
- deltaFactor：增量因子。通过 deltaFactor * deltaX 或者 deltaFactor * deltaY 可以得到浏览器实际的滚动距离。

> 此插件来源于 jquery-mousewheel 。

8.2.二维码模块 :id=qrcode

模块名：QRCode，使用方式：

```
<div id="xxx"></div>
<script>
    layui.use(['QRCode'], function () {
        var $ = layui.jquery;
        var QRCode = layui.QRCode;

        // 二维码
        var demoQrCode = new QRCode(document.getElementById("xxx"), {
            text: "Hello Word!",
            width: 101,  // 宽度
            height: 101,  // 高度
```

```
            colorDark: "#000000",  // 颜色
            colorLight: "#ffffff",  // 背景颜色
            correctLevel: QRCode.CorrectLevel.H
        });

        // 更换内容
        demoQrCode.makeCode("Easyweb");

    });
</script>
```

> 此插件来源于 qrcodejs 。

8.3.引导插件 :id=introjs

模块名：introJs，使用方式：

```
<div data-step="1" data-intro="这是步骤一">步骤一</div>
<div data-step="2" data-intro="这是步骤二">步骤二</div>
<script>
    layui.use(['introJs'], function () {
        var introJs = layui.introJs;

        // 初始化
        introJs().start();

    });
</script>
```

> 此插件来源于 intro.js，对样式进行了微调。

8.4.剪贴板 :id=clipboardjs

模块名：ClipboardJS，使用方式：

```
<input id="foo" value="https://github.com/zenorocha/clipboard.js.git">
<button id="btnCopy" data-clipboard-target="#foo">复制</button>
<script>
    layui.use(['ClipboardJS'], function () {
        var ClipboardJS = layui.ClipboardJS;

        var clipboard = new ClipboardJS('#btnCopy');
        clipboard.on('success', function(e) {
```

```
            e.clearSelection();
        });

        clipboard.on('error', function(e) {
            console.error('Action:', e.action);
        });

    });
</script>
```

> 此插件来源于 clipboard.js 。

8.5.视频播放器 :id=player

模块名：Player，使用方式：

```
<div id="demoVideo"></div>

<script>
    layui.use(['Player'], function () {
        var Player = layui.Player;

        // 视频播放器
        var player = new Player({
            id: "demoVideo",
            url: "//s1.pstatp.com/cdn/expire-1-M/byted-player-videos/1.0.0/xgpl
ayer-demo.mp4",  // 视频地址
            poster: "https://imgcache.qq.com/open_proj/proj_qcloud_v2/gateway/s
olution/general-video/css/img/scene/1.png",  // 封面
            fluid: true,  // 宽度100%
            playbackRate: [0.5, 1, 1.5, 2],  // 开启倍速播放
            pip: true,  // 开启画中画
            lang: 'zh-cn'
        });

        // 开启弹幕
        var dmStyle = {
            color: '#ffcd08', fontSize: '20px'
        };
        var player = new Player({
            id: "demoVideo2",
            url: "http://demo.htmleaf.com/1704/201704071459/video/2.mp4",  //
视频地址
            autoplay: false,
            fluid: true,  // 宽度100%
            lang: 'zh-cn',
```

```
            danmu: {
                comments: [
                    {id: '1', start: 0, txt: '空降', color: true, style: dmStyl
e, duration: 15000},
                    {id: '2', start: 1500, txt: '前方高能', color: true, style:
dmStyle, duration: 15000},
                    {id: '3', start: 3500, txt: '弹幕护体', color: true, style:
dmStyle, duration: 15000},
                ]
            }
        });

    });
</script>
```

> 视频播放器使用的是西瓜播放器，更多介绍请前往查看xgplayer。

8.6.富文本编辑器 :id=ckeditor

模块名：CKEDITOR，使用方式：

```
<textarea id="demoCkEditor"></textarea>

<script>
    layui.use(['CKEDITOR'], function () {
        var $ = layui.jquery;
        var CKEDITOR = layui.CKEDITOR;

        // 渲染富文本编辑器
        CKEDITOR.replace('demoCkEditor', {height: 450});
        var insEdt = CKEDITOR.instances.demoCkEditor;  // 获取渲染后的实例

        var content = insEdt.getData();  // 获取内容
        insEdt.setData('<h1>Hello Word!</h1>');  // 设置内容
        insEdt.insertHtml('<h1>EasyWeb</h1>');  // 插入内容

    });
</script>
```

图片上传配置：

打开 `ckeditor/config.js` 修改 `config.filebrowserImageUploadUrl` 为你的后端地址。

文件上传配置(包括视频、音频、flash等)：

打开 `ckeditor/config.js` 修改 `config.filebrowserUploadUrl` 为你的后端地址。

上传接口返回的数据格式：

成功：

```
{
  "uploaded": 1,
  "fileName": "demo img",
  "url": "https://pic.qqtn.com/up/2018-9/15367146917869444.jpg"
}
```

失败：

```
{
  "uploaded": 0,
  "error": {
    "message": "演示环境，不允许上传，请替换为自己的url"
  }
}
```

配置参数还可以在渲染的时候配置：

```
CKEDITOR.replace('demoCkEditor', {
   height: 450,
   filebrowserImageUploadUrl: '你的接口地址'
});
```

修改操作按钮布局：

前往toolbarconfigurator在线配置，并修改 `ckeditor/config.js` 。

> 富文本编辑器使用的是 `CKEditor4` ，并做好了配置及样式调整。

9、更多功能

9.1.主题功能 :id=theme

1. 打开 主题生成器 在线定制主题
2. 将生成的css放在 `assets/moudel/theme` 目录下
3. 打开“page/tpl/tpl-theme.html”添加生成的主题：

```
var themes = [
    {title: '黑白主题', theme: 'admin'},
    {title: '黑色主题', theme: 'black'},
    {title: '你的主题', theme: 'xxx'}
];
```

主题css以 `theme-xxx.css` 的形式命名，主题预览图跟主题css名字保持一致，预览图可在主题生成器中获取。

> 更强大的主题生成器正在开发中，敬请期待...

9.2.自定义扩展模块 :id=expand

集成社区模块：

Layui社区有很多优秀的扩展模块，前去查看，

集成到easyweb只需要把下载好的模块放在 `/assets/module/` 目录下面，如果模块只是一个单纯的js，不用做任何配置就可以使用了，

例如admin.js、index.js，如果模块是一个文件夹，还需要在 `main.js` 中配置模块的具体位置，例如 `treetable-lay/treetable.js` 。

自定义扩展模块：

如果你需要自己写一个扩展模块，请前往阅读Layui定义模块的开发文档，

或者参考admin.js、index.js等模块。

集成jquery插件：

jquery作为老牌框架，有着丰富的第三方插件，layui基于jquery.js，同样可以使用jquery插件，

最简单的使用方法就是把插件放入libs下，页面中先引入jquery.js，再引入插件js，即可使用。

当然更友好的集成方式是把它改造成layui扩展模块，改造方式请参考layui 封装第三方组件。

10、弹窗专题

10.弹窗专题

Layui没有像Bootstrap那样的直接写在页面中的模态弹窗，Layui的弹窗是通过layer模块用js去弹出窗口，这就意味着Layui的弹窗功能更丰富、操作更灵活，当然灵活也就意味着对于初学者来说使用更为复杂，下面介绍了实现表单弹窗的几种不同方式。

10.1.第一种 页面层弹窗 :id=type1

页面层弹窗就是弹窗页面和列表页面在一个文件中，弹窗类型type=1：

```
<button id="btnAddUser" class="layui-btn">添加</button>
<table id="tableUser" lay-filter="tableUser"></table>

<!-- 表单弹窗 -->
<script type="text/html" id="modelUser">
    <form id="modelUserForm" lay-filter="modelUserForm" class="layui-form model
-form">
        <input name="userId" type="hidden"/>
        <div class="layui-form-item">
            <label class="layui-form-label">用户名</label>
            <div class="layui-input-block">
                <input name="nickName" placeholder="请输入用户名" type="text" cla
ss="layui-input" maxlength="20"
                       lay-verType="tips" lay-verify="required" required/>
            </div>
        </div>
        <div class="layui-form-item">
            <label class="layui-form-label">性别</label>
            <div class="layui-input-block">
                <input type="radio" name="sex" value="男" title="男" checked/>
                <input type="radio" name="sex" value="女" title="女"/>
            </div>
        </div>
        <div class="layui-form-item text-right">
            <button class="layui-btn layui-btn-primary" type="button" ew-event=
"closeDialog">取消</button>
            <button class="layui-btn" lay-filter="modelSubmitUser" lay-submit>
保存</button>
        </div>
    </form>
</script>

<!-- 表格操作列 -->
<script type="text/html" id="tableBarUser">
```

```
    <a class="layui-btn layui-btn-primary layui-btn-xs" lay-event="edit">修改</
a>
    <a class="layui-btn layui-btn-danger layui-btn-xs" lay-event="del">删除</a>
</script>

<!-- js部分 -->
<script>
    layui.use(['layer', 'form', 'table', 'admin'], function () {
        var $ = layui.jquery;
        var layer = layui.layer;
        var form = layui.form;
        var table = layui.table;
        var admin = layui.admin;

        // 渲染表格
        var insTb = table.render({
            elem: '#tableUser',
            url: '../../json/user.json',
            cols: [[
                {type: 'numbers', title: '#'},
                {field: 'nickName', sort: true, title: '用户名'},
                {field: 'sex', sort: true, title: '性别'},
                {align: 'center', toolbar: '#tableBarUser', title: '操作', minW
idth: 200}
            ]]
        });

        // 添加
        $('#btnAddUser').click(function () {
            showEditModel();
        });

        // 工具条点击事件
        table.on('tool(tableUser)', function (obj) {
            var data = obj.data;
            var layEvent = obj.event;
            if (layEvent === 'edit') { // 修改
                showEditModel(data);
            } else if (layEvent === 'del') { // 删除
                layer.msg('点击了删除', {icon: 2});
            }
        });

        // 显示表单弹窗
        function showEditModel(mUser) {
            admin.open({
                type: 1,
                title: (mUser ? '修改' : '添加') + '用户',
                content: $('#modelUser').html(),
                success: function (layero, dIndex) {
```

```
                    var url = mUser ? '/updateUser' : '/addUser';
                    // 回显数据
                    form.val('modelUserForm', mUser);
                    // 表单提交事件
                    form.on('submit(modelSubmitUser)', function (data) {
                        layer.load(2);
                        $.post(url, data.field, function (res) {
                            layer.closeAll('loading');
                            if (res.code == 200) {
                                layer.close(dIndex);
                                layer.msg(res.msg, {icon: 1});
                                insTb.reload();
                            } else {
                                layer.msg(res.msg, {icon: 2});
                            }
                        }, 'json');
                        return false;
                    });
                }
            });
        }

    });
</script>
```

弹窗的参数 `type: 1` 表示弹窗类型是页面层类型而不是iframe层类型，
`content: $("#modelUser").html()` 是指定弹窗的内容，
表单使用 `<script type="text/html">` 而不是使用div，这样表单页面会默认隐藏，需要注意的是，操作表单的代码，
包括表单的事件监听都要写在弹窗的 `success` 回调里面，因为layer弹窗是把表单的html复制一份动态插入到页面上，而不是把那一段代码变成模态窗，
所以事件绑定必须写在success里面。

> 这种方式表单和表格在一个页面上面，数据传递、页面相互操作比较方便，不涉及到iframe父子页面传值的问题，推荐初学者使用这种方式

!> 切记： 对于type=1的弹窗，所有操作弹窗内元素的代码都要放在弹窗的success里面。

10.2.第二种 iframe层弹窗 :id=type2

使用iframe类型的弹窗可以让表单页面和表格页面分开，逻辑更清晰。
弹窗页面，userForm.html：

```
<html>
<head>
    <link rel="stylesheet" href="../../assets/libs/layui/css/layui.css"/>
    <link rel="stylesheet" href="../../assets/module/admin.css"/>
</head>
<body>
<form id="modelUserForm" lay-filter="modelUserForm" class="layui-form model-for
m">
    <input name="userId" type="hidden"/>
    <div class="layui-form-item">
        <label class="layui-form-label">用户名</label>
        <div class="layui-input-block">
            <input name="nickName" placeholder="请输入用户名" type="text" class="
layui-input" maxlength="20"
                   lay-verType="tips" lay-verify="required" required/>
        </div>
    </div>
    <div class="layui-form-item">
        <label class="layui-form-label">性别</label>
        <div class="layui-input-block">
            <input type="radio" name="sex" value="男" title="男" checked/>
            <input type="radio" name="sex" value="女" title="女"/>
        </div>
    </div>
    <div class="layui-form-item text-right">
        <button class="layui-btn layui-btn-primary" type="button" ew-event="clo
seIframeDialog">取消</button>
        <button class="layui-btn" lay-filter="modelSubmitUser" lay-submit>保存<
/button>
    </div>
</form>

<!-- js部分 -->
<script type="text/javascript" src="../../assets/libs/layui/layui.js"></script>
<script type="text/javascript" src="../../assets/js/common.js"></script>
<script>
    layui.use(['layer', 'form', 'admin'], function () {
        var $ = layui.jquery;
        var layer = layui.layer;
        var form = layui.form;
        var admin = layui.admin;

        var mUser = admin.getLayerData();  // 获取列表页面传递的数据
        // 回显数据
        form.val('modelUserForm', mUser);
        // 表单提交事件
        form.on('submit(modelSubmitUser)', function (data) {
            layer.load(2);
            var url = mUser ? '/updateUser' : '/addUser';
```

```
            $.post(url, data.field, function (res) {
                layer.closeAll('loading');
                if (res.code == 200) {
                    layer.msg(res.msg, {icon: 1});
                    admin.putLayerData('formOk', true);  // 设置操作成功的标识
                    admin.closeThisDialog();  // 关闭当前iframe弹窗
                } else {
                    layer.msg(res.msg, {icon: 2});
                }
            }, 'json');
            return false;
        });

    });
</script>
</body>
</html>
```

表格页面，list.html：

```
<button id="btnAddUser" class="layui-btn">添加</button>
<table id="tableUser" lay-filter="tableUser"></table>

<!-- 表格操作列 -->
<script type="text/html" id="tableBarUser">
    <a class="layui-btn layui-btn-primary layui-btn-xs" lay-event="edit">修改</
a>
    <a class="layui-btn layui-btn-danger layui-btn-xs" lay-event="del">删除</a>
</script>

<!-- js部分 -->
<script>
    layui.use(['layer', 'table', 'admin'], function () {
        var $ = layui.jquery;
        var layer = layui.layer;
        var table = layui.table;
        var admin = layui.admin;

        // 渲染表格
        var insTb = table.render({
            elem: '#tableUser',
            url: '../../json/user.json',
            cols: [[
                {type: 'numbers', title: '#'},
                {field: 'nickName', sort: true, title: '用户名'},
                {field: 'sex', sort: true, title: '性别'},
                {align: 'center', toolbar: '#tableBarUser', title: '操作', minW
idth: 200}
            ]]
```

```
        });

        // 添加
        $('#btnAddUser').click(function () {
            showEditModel();
        });

        // 工具条点击事件
        table.on('tool(tableUser)', function (obj) {
            var data = obj.data;
            var layEvent = obj.event;
            if (layEvent === 'edit') { // 修改
                showEditModel(data);
            } else if (layEvent === 'del') { // 删除
                layer.msg('点击了删除', {icon: 2});
            }
        });

        // 显示表单弹窗
        function showEditModel(mUser) {
            var layIndex = admin.open({
                type: 2,
                title: (mUser ? '修改' : '添加') + '用户',
                content: 'userForm.html',
                data: mUser,       // 使用data参数传值给弹窗页面
                end: function () {
                    if (admin.getLayerData(layIndex, 'formOk')) {  // 判断表单操
作成功标识
                        insTb.reload();  // 成功刷新表格
                    }
                }
            });
        }

    });
</script>
```

弹窗参数 `type: 2` 就表示是一个iframe类型的弹窗，content参数写表单的页面url，
然后在end回调里面判断操作成功的标识然后刷新表格，注意是end而不是success，这样做的好处就是，
弹窗页面跟表格独立开了，大大减少了每个页面的代码量，将业务逻辑分开，坏处就是，
如果页面有下拉框、日期选择控件等元素，它们的范围不能超出弹窗的范围，会导致弹窗出现滚动条，甚至不显示出来。

那么有没有办法既能让表单页面独立，又能让表单里面的下拉框、日期等组件能够超出弹窗的范围呢，请看第三种方式。

10.3.第三种 url方式弹窗 :id=type3

url方式弹窗就是使用url参数将弹窗页面独立出来：

弹窗页面，userForm.html：

```
<form id="modelUserForm" lay-filter="modelUserForm" class="layui-form model-for
m">
    <input name="userId" type="hidden"/>
    <div class="layui-form-item">
        <label class="layui-form-label">用户名</label>
        <div class="layui-input-block">
            <input name="nickName" placeholder="请输入用户名" type="text" class="
layui-input" maxlength="20"
                   lay-verType="tips" lay-verify="required" required/>
        </div>
    </div>
    <div class="layui-form-item">
        <label class="layui-form-label">性别</label>
        <div class="layui-input-block">
            <input type="radio" name="sex" value="男" title="男" checked/>
            <input type="radio" name="sex" value="女" title="女"/>
        </div>
    </div>
    <div class="layui-form-item text-right">
        <button class="layui-btn layui-btn-primary" type="button" ew-event="clo
seDialog">取消</button>
        <button class="layui-btn" lay-filter="modelSubmitUser" lay-submit>保存<
/button>
    </div>
</form>

<script>
    layui.use(['layer', 'form', admin'], function () {
        var $ = layui.jquery;
        var layer = layui.layer;
        var form = layui.form;
        var admin = layui.admin;

        var mUser = admin.getLayerData('#modelUserForm');  // 列表页面传递的数据,
#modelUserForm这个只要写弹窗内任意一个元素的id即可
        // 回显数据
        form.val('modelUserForm', mUser);
        // 表单提交事件
        form.on('submit(modelSubmitUser)', function (data) {
        layer.load(2);
        var url = mUser ? '/updateUser' : '/addUser';
            $.post(url, data.field, function (res) {
                layer.closeAll('loading');
                if (res.code == 200) {
                    layer.msg(res.msg, {icon: 1});
                    admin.putLayerData('formOk', true, '#modelUserForm');  //
```

```
设置操作成功的标识，#modelUserForm这个只要写弹窗内任意一个元素的id即可
                    admin.closeDialog('#modelUserForm');  // 关闭页面层弹窗
                } else {
                    layer.msg(res.msg, {icon: 2});
                }
            }, 'json');
            return false;
        });

    });
</script>
```

注意这里，页面不需要写 `<html><body>` 这些东西，它是一个html片段，不是完整的html页面。

表格页面，list.html：

```
<button id="btnAddUser" class="layui-btn">添加</button>
<table id="tableUser" lay-filter="tableUser"></table>

<!-- 表格操作列 -->
<script type="text/html" id="tableBarUser">
    <a class="layui-btn layui-btn-primary layui-btn-xs" lay-event="edit">修改</
a>
    <a class="layui-btn layui-btn-danger layui-btn-xs" lay-event="del">删除</a>
</script>

<!-- js部分 -->
<script>
    layui.use(['layer', table', 'admin'], function () {
        var $ = layui.jquery;
        var layer = layui.layer;
        var table = layui.table;
        var admin = layui.admin;

        // 渲染表格
        var insTb = table.render({
            elem: '#tableUser',
            url: '../../json/user.json',
            cols: [[
                {type: 'numbers', title: '#'},
                {field: 'nickName', sort: true, title: '用户名'},
                {field: 'sex', sort: true, title: '性别'},
                {align: 'center', toolbar: '#tableBarUser', title: '操作', minW
idth: 200}
            ]]
        });

        // 添加
        $('#btnAddUser').click(function () {
```

```
            showEditModel();
        });

        // 工具条点击事件
        table.on('tool(tableUser)', function (obj) {
            var data = obj.data;
            var layEvent = obj.event;
            if (layEvent === 'edit') { // 修改
                showEditModel(data);
            } else if (layEvent === 'del') { // 删除
                layer.msg('点击了删除', {icon: 2});
            }
        });

        // 显示表单弹窗
        function showEditModel(mUser) {
            var layIndex = admin.open({
                title: (mUser ? '修改' : '添加') + '用户',
                url: 'userForm.html',
                data: mUser,     // 传递数据到表单页面
                end: function () {
                    if (admin.getLayerData(layIndex, 'formOk')) {  // 判断表单操
作成功标识
                        insTb.reload();  // 成功刷新表格
                    }
                },
                success: function (layero, dIndex) {
                    // 弹窗超出范围不出现滚动条
                    $(layero).children('.layui-layer-content').css('overflow',
'visible');
                }
            });
        }

    });
</script>
```

这里与iframe弹窗的区别是，使用 `url` 参数指定表单的页面链接，而不是使用content，url这个参数是admin.open扩展的，layer不具备，

这样做会把表单页面的html内容使用ajax加载到弹窗中，而不是iframe嵌入，这样表单里面的下拉框、日期等组件就可以超出弹窗的范围，不会导致弹窗出现滚动条。

!> 注意：表单页面是html片段，不是完整的html，这点不要忘记了，另外表单页面和表格页面不要出现重复的id，因为最终是在一个页面上

10.4.四种方式选择指南 :id=choose

方式	推荐	理由

第一、四种	建议初学者使用这种	不涉及两个页面传值问题
第二种	推荐表单跟表格无交互使用，比如详情	下拉框、日期不能超出弹窗
第三种	表单弹窗、页面有交互建议使用	页面传值方便，不存在问题

10.5.admin.modelForm方法 :id=model_form

此方法是把layer弹窗自带的确定按钮绑定成表单的提交按钮。

```
<!-- 表单弹窗 -->
<script type="text/html" id="modelUser">
    <div class="layui-form-item">
        <label class="layui-form-label">用户名</label>
        <div class="layui-input-block">
            <input name="nickName" placeholder="请输入用户名" type="text" class="
layui-input" maxlength="20"
                   lay-verType="tips" lay-verify="required" required/>
        </div>
    </div>
    <div class="layui-form-item">
        <label class="layui-form-label">性别</label>
        <div class="layui-input-block">
            <input type="radio" name="sex" value="男" title="男" checked/>
            <input type="radio" name="sex" value="女" title="女"/>
        </div>
    </div>
</script>

<!-- js部分 -->
<script>
    layui.use(['layer', 'form', admin'], function () {
        var $ = layui.jquery;
        var layer = layui.layer;
        var form = layui.form;
        var admin = layui.admin;

        admin.open({
            type: 1,
            title: '添加用户',
            btn: ['确定', '取消'],
            content: $('#modelUser').html(),
            success: function (layero, dIndex) {
                // 把确定按钮绑定表单提交，参数二是给按钮起一个lay-filter，参数三是给表单
起一个lay-filter
                admin.modelForm(layero, 'demoFormSubmit', 'demoForm');
```

```
                // 给表单赋值
                form.val('demoForm', {nickName: '张三', sex: '男'});

                // 监听表单提交
                form.on('submit(demoFormSubmit)', function (data) {
                    layer.msg(JSON.stringify(data.field));
                    return false;
                });

            },
            yes: function () {
                // 确定按钮方法什么都不要操作
            }
        });

    });
</script>
```

admin.modelForm()这个方法会把弹窗外面包一个form，然后把确定按钮加lay-submit，所以你的表单页面不需要写form和确定、关闭按钮，只需要写表单项。

这个方法的使用场景为你想要表单的按钮固定，只滚动表单内容部分，可以用这个操作，
当然那种自己写确定和取消按钮的表单弹窗的写法也是支持固定按钮的，
前面css组件样式中有介绍。

10.6.参数传递方法详解 :id=model_param

参数传递：

```
admin.open({
    type: 2,
    content: 'userForm.html',
    data: {
        name: '李白',
        sex: '男'
    }
});
```

通过data属性进行参数传递，data同样是admin.open扩展的，layer不具备。

获取参数：

方法	说明	参数
admin.getLayerData(index)	获取某弹窗的全部参数	layer的index

admin.getLayerData(index, key)	参数某弹窗参数的某个字段	index，字段
admin.getLayerData()	iframe弹窗子页面获取参数	无任何参数
admin.getLayerData('#xxForm')	url方式弹窗子页面获取参数	弹窗内任意元素id

如果是在iframe弹窗的子页面中可以使用admin.getLayerData()直接获取父页面传递的全部参数，如果是在url方式打开的弹窗中可以使用admin.getLayerData('#xx')直接获取父页面的全部参数，'#xx'是弹窗内任意元素的id。

增加参数：

方法	说明	参数
admin.putLayerData(key, value, index)	增加参数	字段名，值，index
admin.putLayerData(key, value)	iframe弹窗子页面增加参数	字段名，值
admin.putLayerData(key, value, '#xx')	url方式弹窗子页面增加参数	弹窗内任意元素id

!> **注意：**data参数必须是对象的形式，data:1、data:'aa'这种写法会导致无法put新参数。

10.7.第四种 捕获层弹窗

第一种页面层弹窗由于所有关于弹窗内元素的操作代码都要写在弹窗的success里面，大部分人不适应这种方式，所以介绍第四种捕获层弹窗：

```
<!-- 表单弹窗 -->
<form id="modelRoleForm" lay-filter="modelRoleForm" class="layui-form model-for
m" style="display: none;">
    <input name="roleId" type="hidden"/>
    <div class="layui-form-item">
        <label class="layui-form-label">角色名</label>
        <div class="layui-input-block">
            <input name="roleName" placeholder="请输入角色名" type="text" class="
layui-input"
                   lay-verType="tips" lay-verify="required" required/>
        </div>
    </div>
    <div class="layui-form-item">
        <label class="layui-form-label">备注</label>
        <div class="layui-input-block">
            <textarea name="comments" placeholder="请输入内容" class="layui-text
```

```
area"></textarea>
        </div>
    </div>
    <div class="layui-form-item text-right">
        <button class="layui-btn layui-btn-primary" type="button" ew-event="clo
seDialog">取消</button>
        <button class="layui-btn" lay-filter="modelSubmitRole" lay-submit>保存<
/button>
    </div>
</form>

<script>
    layui.use(['layer', 'form', 'table', 'admin'], function () {
        var $ = layui.jquery;
        var layer = layui.layer;
        var form = layui.form;
        var table = layui.table;
        var admin = layui.admin;
        var formUrl;

        // 渲染表格
        var insTb = table.render({...});

        // 添加
        $('#btnAddRole').click(function () {
            showEditModel();
        });

        // 表格工具条点击事件
        table.on('tool(tableRole)', function (obj) {
            var data = obj.data;
            var layEvent = obj.event;
            if (layEvent === 'edit') { // 修改
                showEditModel(data);
            }
        });

        // 显示编辑弹窗
        function showEditModel(mRole) {
            // 如果是单标签版本可以使用下面代码把表单移到body中
            /* if ($('#modelRoleForm').parent('body').length == 0) {
                $('body').append($('#modelRoleForm'));
            } */
            $('#modelRoleForm')[0].reset();  // 重置表单
            form.val('modelRoleForm', mRole);  // 回显数据
            formUrl = mRole ? 'role/update' : 'role/add';
            admin.open({
                type: 1,
                title: (mRole ? '修改' : '添加') + '角色',
                content: $('#modelRoleForm')
```

```
            });
        }

        // 表单提交事件
        form.on('submit(modelSubmitRole)', function (data) {
            layer.load(2);
            $.post(formUrl, data.field, function (res) {
                layer.closeAll('loading');
                if (res.code == 200) {
                    admin.closeDialog('#modelRoleForm');
                    layer.msg(res.msg, {icon: 1});
                    insTb.reload();
                } else {
                    layer.msg(res.msg, {icon: 2});
                }
            }, 'json');
            return false;
        });

    });
</script>
```

与第一种的区别是form不用 `<script>` 包裹，加 `style="display:none"` 隐藏，admin.open的content是$('#roleForm')而不是$('#roleForm').html()，

表单的提交事件可以直接写在外面，而不用写在弹窗的success里面。

!> 捕获层的弊端就是弹窗的页面代码最好是写在body下面，不然样式会被其他影响，所以spa版本不建议使用这种方式，单标签版、iframe版本可以使用。

11、常见问题

11.1.跨页面操作 :id=page_oper

单页面是可以直接跨页面操作的：

```
table.reload('userTab', {});  // 刷新其他页面表格
```

layui.use里面定义的变量想在其他页面访问：

```
layui.use(['layer'], function() {

    window.xxx = 'xxxx';  // 变量给其他页面访问

    // 方法给其他页面访问
    window.xxx = function(){
        alert('aa');
    };
});
```

11.2.nginx部署解决跨域 :id=nginx_cors

部署只需要把前端所有代码放在nginx的html中，修改config.js的base_server为接口地址即可，

如果接口没有做跨域支持，可使用nginx反向代理来解决跨域问题：

1. 打开“nginx/conf/nginx.conf”配置文件
2. 设置反向代理：

```
http {
    server {
        # 加入以下配置，之前的配置全部不要动，这个location是新加入的
        location /api/ {
            proxy_pass  http://11.11.111.111:8088/; # 这个是后台接口所在的地址
        }
    }
}
```

3. 修改config.js里面的base_server为/api/。

> 这样配置接口就会访问localhost:80/api/，然后nginx再反向代理到实际接口

11.3.多系统模式 :id=more_side

在header中有几个选项：“xx系统”、“xx系统”，点击不同的系统切换不同的侧边菜单，

在线演示：side-more.html

侧边栏代码，使用 `nav-id` 标识不同的菜单：

```
<div class="layui-side">
    <div class="layui-side-scroll">
        <!-- 系统一的菜单，每个ul都加nav-id -->
        <ul nav-id="xt1" class="layui-nav layui-nav-tree" lay-filter="admin-sid
e-nav">
            <!-- ...省略代码... -->
        </ul>
        <!-- 系统二的菜单，加layui-hide隐藏 -->
        <ul nav-id="xt2" class="layui-nav layui-nav-tree layui-hide" lay-filter
="admin-side-nav">
            <!-- ...省略代码... -->
        </ul>
    </div>
</div>
```

header.html代码，使用 `nav-bind` 绑定侧边栏菜单：

```
<div class="layui-header">
    <ul class="layui-nav layui-layout-left">
        <li class="layui-nav-item"><a nav-bind="xt1">系统一</a></li>
        <li class="layui-nav-item"<a nav-bind="xt2">系统二</a></li>
    </ul>
</div>
```

11.4.弹窗下拉框出现滚动条 :id=dialog_scroll

非iframe类型的弹窗才能解决，解决办法如下：

```
 admin.open({
        title: '添加用户',
        content: $('#model').html(),
        success: function (layero, index) {
            // 禁止出现滚动条
            $(layero).children('.layui-layer-content').css('overflow', 'visible
');
        }
    });
```

> 关键代码就是在success回调中写上面那句话

11.5.弹窗宽度不能超出屏幕 :id=dialog_overflow

这个是针对手机屏幕下做了让弹窗宽度自适应，如果你需要让弹窗宽度超出屏幕如下代码即可：

```
admin.open({
        title: '添加用户',
        content: $('#model').html(),
        success: function (layero, index) {
            $(layero).css('max-width', 'unset');   // 去掉max-width属性
        }
});
```

> success回调中去掉max-width属性

11.6.表单文字出现换行 :id=form_text

layui的表单左边的标题文字最多显示5个字，超出会换行，通过添加css修改宽度：

```
#userForm .layui-form-label {
    width: 100px;  // 这里修改标题宽度
}

#userForm .layui-input-block {
    margin-left: 130px;  // 这里要比上面始终大30px
}
```

`#userForm` 是表单的id，加id避免影响其他表单样式：

```
<form id="userForm" class="layui-form">
    <div class="layui-form-item">
        <label class="layui-form-label">活动起止时间</label>
        <div class="layui-input-block">
            <input type="text" class="layui-input"/>
        </div>
    </div>
    <div class="layui-form-item">
        <label class="layui-form-label">活动详细介绍</label>
        <div class="layui-input-block">
            <textarea class="layui-textarea" maxlength="200"></textarea>
        </div>
    </div>
</form>
```

11.7.侧边栏折叠图标放大 :id=side_icon

侧边栏折叠后图标会进行放大，如果要修改大小，添加如下css：

```
@media screen and (min-width: 750px) {
    .layui-layout-admin.admin-nav-mini .layui-side .layui-nav .layui-nav-item >
 a > .layui-icon {
        font-size: 18px;
    }
}
```

修改font-size即可，如果不想放大，改成14px即可。

11.8.select、radio不显示 :id=select_none

select、radio在layui中会被美化，对于动态生成的元素需要重新渲染才能美化：

```
$('div').appen('<select><option value="1">xxxx</option></select>');
form.render('select');  // 重新渲染select
form.render('radio');  // 重新渲染radio

// 对于弹窗内select不显示
admin.open({
    type: 1,
    content: '<select><option value="1">xxxx</option></select>',
    success: function(){
        form.render('select');  // 弹窗要在success里重新渲染
    }
});
```

12、路由模块

12.1.快速了解

使用示例：

```
<div id="m"></div>
<a href="#/user">go user</a>
<a href="#/home">go home</a>

<script>
layui.use(['layRouter'],function(){
    var layRouter = layui.layRouter;

    layRouter.reg('/home', function(){
        document.getElementById('m').innerHTML = 'Hello World';
    }).reg('/user', function() {
        document.getElementById('m').innerHTML = 'This is user page';
    });

    layRouter.init({
        index: '/home' /* 首页地址 */
    });
});
</script>
```

打开例子后，浏览器会从 http://xxx.com/ 跳转到 http://xx.com/#/home ，并且在id为m的div中显示 Hello World。

然后点击 `go user` 的链接，会跳转到 http://xx.com/#/user ，并且在id为m的div中显示 his is user page。

12.2.注册路由

```
layRouter.reg('/home', function(){
    alert('xxx');
});

// 在框架中要使用这种
index.regRouter([{
    name: '用户管理',
    url: '#/system/user'
}]);
```

上面是基础的写法，下面是index模块封装的写法，使用index.regRouter注册后访问 `#/system/user` ，就会打开一个“用户管理”的标签页，使用上面的写法不会。

> layRouter是路由的核心实现，提供给index模块作为支撑的，一般不需要去直接用它。

12.3.路由参数传递

请参考页面路由参数传递。

参数传递的规则：

```
#/system/user                    // 无参数

#/system/user/id=1               // 参数id=1

#/system/user/id=1/name=aaa      // 参数id=1, name=aaa
```

这三种类型的url注册路由的时候只会注册 `#/system/user` 这个url，后面两个都是加了参数，而且后面的参数可以无限加，

也就是说如果已经注册了 `#/system/user` ， `#/system/user` 和 `#/system/user/id=1/name=aaa` 是可以直接访问的。

如何获取当前传递的参数：

```
layui.router().search;

layui.router('#/system/user/id=1/name=aa').search;
```

```
var param = layui.router().search
console.log(param.id);
console.log(param.name);
```

> 注意：不要传递太过复杂的参数，如 `'/'` 和 `'='` 会影响路由解析的符号不能传递。

12.4.路由不存在处理

在config.js中有一个 `routerNotFound` 方法用于处理404路由。

12.5.路由切换监听

注册 `pop` 方法即可监听路由的切换：

```
layRouter.reg('pop', function(r){
    console.log(r);
});
```

13、变更记录

更新日志 :id=log

> 当前版本： `spa v3.1.5` ，更新于： `2019-10-05` ，查看 在线演示。

2019-11-18 :id=log_tb

- 升级树形表格treeTable组件
- 在强烈呼吁下已支持懒加载
- 框架本次并未升级，如要使用新版treeTable请到这里下载

2019-10-05 （V3.1.5）:id=log_315

- 增加新组件圆形进度条“CircleProgress”
- 增加新组件“dataGrid”，实现非表格形式的列表自动渲染
- 增加新组件“formX”，实现了一大堆表单验证
- tableX模块增加后端导出的方法，支持传参数、post请求
- admin.open增加参数传递、获取参数的封装
- contextMenu模块click事件增加可获取绑定的目标元素
- 三套内置的白色主题进行重新设计
- admin模块增加判断富文本是否为空的方法
- 增加了又一个经典实例页面
- admin.css部分样式微调

> 升级替换assets/module目录即可

2019-08-07 （V3.1.4）:id=log_314

- 增加新组件“级联选择器”，支持无限级、懒加载、模糊搜索、清除
- 增加新组件“分割面板”，支持水平、垂直、嵌套分割
- 侧边栏、折叠面板展开/折叠增加过渡效果，告别生硬的方式
- 侧边栏箭头样式优化、折叠时悬浮弹出的效果优化
- 集成富文本编辑器CKEditor、视频播放器
- ew-event的open和popupRight支持function类型的参数
- admin模块增加动画数字、经纬度坐标系转换等工具方法
- 增加经典实例，并会持续增加实际项目中遇到的经典例子
- 修复上个版本对tips样式重写、toolbar的mini样式产生的一些bug

> 升级指南：侧边栏、折叠面板展开/折叠过渡效果这个功能修改了layui的element模块，
> 所以除了升级assets/module目录外还需要升级layui/lay/modules/element.js

2019-07-12（V3.1.3）:id=log_313

- 增加新的控制台页面
- 侧边栏折叠下子菜单实现无限悬浮
- 文件选择弹窗封装为组件
- 增加标签输入框组件
- 将二维码、引导插件、鼠标滚轮、剪贴板复制等插件封装为layui模块
- 数据表格自带的toolbar增加mini样式，查看效果
- ew-event增加open和popupRight，可实现无js打开弹窗
- 增加第三种loading样式，loading增加大、小两种尺寸
- 增加单标签模式下全局隐藏标题栏功能
- tips吸附层样式优化
- 解决tableX合并单元格跟排序冲突问题
- 侧边栏折叠后增加图标放大效果
- 主框架布局css重写，以解决引导插件被覆盖的问题
- 移动端下侧边栏抽屉效果改为侧滑效果

> 本次升级module目录全部替换(每次更新基本都是)，layui内置的layer.js(修改了tips方法)和form.js(表单验证可配置tips颜色)进行了修改，
> 也需要替换，common.js别忘了配置新增加的扩展模块。

2019-06-05（V3.1.2）:id=log_312

- 增加tableX模块
  - 后端排序自动传递sort和order
  - 前端分页、排序、模糊搜索，分页支持url方式
  - 合并单元格
  - 导出数据，支持导出temple列，支持自定义格式
  - 对行绑定鼠标右键菜单
- 增加选择地图经纬度弹窗
- 增加裁剪图片弹窗
- 下拉菜单dropdown模块重写，支持hover模式，自定义下拉内容
- 消息通知notice模块升级，增加音效、滑动清除、双主题
- 表单弹窗增加固定底部按钮栏的样式

- 解决折叠侧边栏表格横向滚动条闪现的问题
- ztree封装为layui扩展模块
- 侧边栏箭头可选三种样式，可在主题界面设置
- 增加设置按钮的loading状态
- 模板页面复杂表单固定底部按钮栏
- 表格上方搜索栏全部改成表单提交方式
- config.js增加版本号配置，可解决页面缓存问题
- 增加Tab记忆功能
- 三个版本基本已经稳定，版本号开始统一

> 本地升级替换assets/module目录，需要注意dropdown模块不兼容旧版，
> 如果升级dropdown之前的下拉菜单请修改写法，或者不升级dropdown模块，
> 另外main.js中加了version:true，config.js最下面加了设置layui的version为config.version，请注意添加。

2019-03-24（V3.0.8）:id=log_308

- 增加打印模块，支持ie打印预览、分页打印
- 对自带的几套主题进行重新配色
- 侧边栏选中增加自动滚动到选中位置
- 关闭选项卡时自动生成标题栏，无需手动添加
- index中遮罩层、选项卡代码可自动生成，简化结构
- 加载动画增加新样式，颜色可随主题改变，并可配置透明度

> 本次升级替换assets/module目录，index.html中可移除遮罩层、选项卡的代码，加载动画要使用新样式可参考开发文档。

2019-01-17（V3.0.7）:id=log_307

- 增加鼠标右键扩展模块“contextMenu”，支持无限级
- 选项卡增加鼠标右键刷新关闭，可自由配置是否开启
- 多系统功能进行封装，无需写js，并且增加联动控制
- 主题位置进行规范，支持设置默认主题，加载切换主题由admin.js控制，与common.js解耦
- 限制选项卡最大打开数量

> 本次升级admin.css和theme都移入到module下面，js和css全部替换。

2018-12-24（V3.0.6）:id=log_306

- 增加个人中心、左树右表等模板页面

- 消息通知模块增加提示框风格
- 修复手风琴侧边栏选中错位的问题
- 便签增加删除功能

> 本次升级替换admin.css、index.js、notice.css、notice.js。

2018-12-10（V3.0.5）:id=log_305

- 完善扩展插件dropdown，增加嵌入式下拉菜单

> 本次升级替换admin.css、dropdown.css、dropdown.js。

2018-12-04（V3.0.4）:id=log_304

- 基于q.js核心代码重写路由模块layRouter
- 解决q.js路由参数传递困难、路由注册不灵活等问题
- 增加可配置是否开启选项卡，关闭选项卡样式跟静态版保持一致

> 本次更新增加了layRouter.js，index.js、admin.js、main.js、admin.css都有修改，移除q.js。

2018-11-20（V3.0.3）:id=log_303

- 增加下拉菜单插件
- 增加消息通知插件
- admin.open弹窗标题颜色可随主题控制
- 增加本地便签功能
- 解决三级以上菜单无法折叠的bug
- 增加常用模板页面
- 解决选项卡样式受框架影响的bug

2018-10-27（V3.0.2）:id=log_302

- 优化admin模块一些方法的封装
- 简化index模块，让上手更容易
- 主题弹窗增加可配置url功能
- 增加可配置侧边栏手风琴折叠效果

2018-10-11（V3.0）:id=log_300

- 发布iframe版本、静态单标签版本、spa单页面版本
- 切换主题不需要刷新页面，内置多套主题
- 更新主题生成器，增加实时预览功能

2018-02-11（V1.0-2.0）:id=log_1020

- 基于layui、q.js实现的spa单页面后台框架，前后端分离架构
- 不需要webpack、npm等知识就可以实现前后端分离、模块化
- 1.x、2.x版本为免费开源项目，吸引不少用户了解并加入Layui