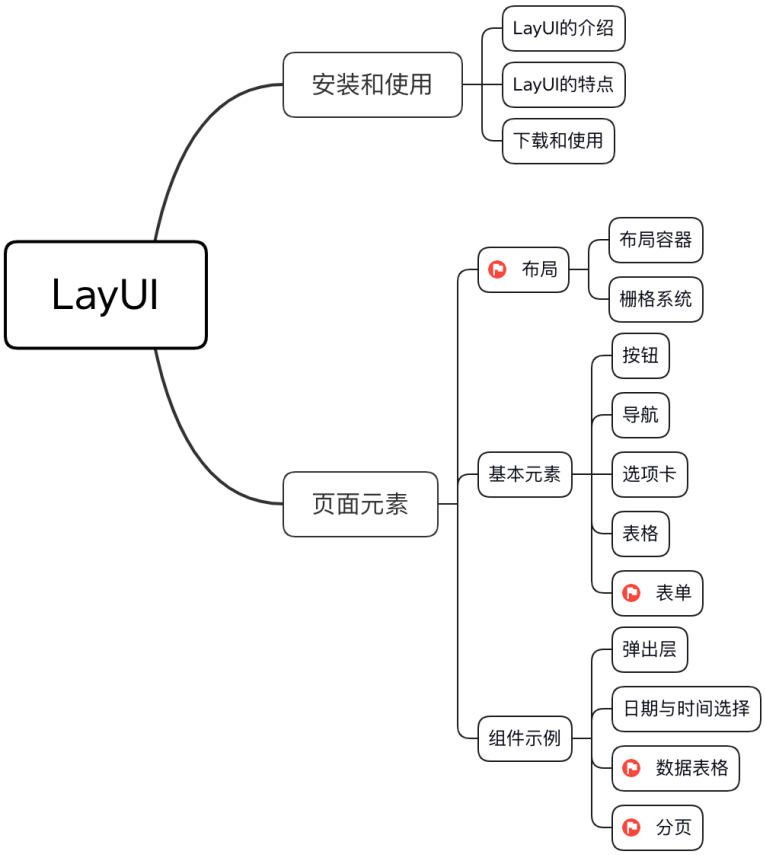
LayUI

1. 主要内容



2. LayUI 的安装及使用

2.1. LayUI 的介绍

layui（谐音：类UI) 是一款采用自身模块规范编写的前端 UI 框架，遵循原生 HTML/CSS/JS 的书写与组织形式，门槛极低，拿来即用。

由国人开发，16年出厂的框架，其主要提供了很多好看、方便的样式，并且基本拿来即用，和Bootstrap有些相似，但该框架有个极大的好处就是定义了很多前后端交互的样式接口，如分页表格，只需在前端配置好接口，后端则按照定义好的接口规则返回数据，即可完成页面的展示，极大减少了后端人员的开发成本。

官网：https://www.layui.com

官方文档：https://www.layui.com/doc/

2.2. LayUI 的特点

（1）layui属于轻量级框架，简单美观。适用于开发后端模式，它在服务端页面上有非常好的效果。

（2）layui是提供给后端开发人员的ui框架，基于DOM驱动。

2.3. 下载与使用

1. 在 官网首页 下载到 layui 的最新版。目录结构如下：

```
├─css // css目录
 │   ├─modules // 模块css目录（一般如果模块相对较大，我们会单独提取，比如下面三个：）
 │   │   ├─laydate
 │   │   ├─layer
 │   │   └─layim
 │   └─layui.css // 核心样式文件
 ├─font  // 字体图标目录
 ├─images // 图片资源目录（目前只有layim和编辑器用到的GIF表情）
 ├─lay // 模块核心目录
 │   └─modules // 各模块组件
 ├─layui.js // 基础核心库
 └─layui.all.js // 包含layui.js和所有模块的合并文件
```

2. 获得 layui 后，将其完整地部署（拷贝到项目中）到你的项目目录，你只需要引入下述两个文件：

```
./layui/css/layui.css
./layui/layui.js // 提示：如果是采用非模块化方式，此处可换成：./layui/layui.all.js
```

3. 基本的入门页面

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-
scale=1">
  <title>开始使用layui</title>
  <link rel="stylesheet" href="layui/css/layui.css">
</head>
<body>

<!-- 你的HTML代码 -->
```

```
<script src="layui/layui.js"></script>
<script>
    // 模块和回调函数
    // 一般直接写在一个js文件中
    layui.use(['layer', 'form'], function(){
      var layer = layui.layer
      ,form = layui.form;

      layer.msg('Hello World');
    });
</script>
</body>
</html>
```

4. 还需要声明需要使用的 **模块** 和 **回调函数**。参照官方文档，选择自己想要的效果就行。

   比如：

   ```
   <script>
       // 注意：导航 依赖 element 模块，否则无法进行功能性操作
       layui.use('element', function(){
           var element = layui.element;

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

3. 页面元素

3.1. 布局

3.1.1. 布局容器

3.1.1.1. 固定宽度

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

```
<div class="layui-container">
  <div class="layui-row">
    ……
  </div>
</div>
```

3.1.1.2. 完整宽度

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

```
<div class="layui-fluid">
  ……
</div>
```

3.1.2. 栅格系统

为了丰富网页布局，简化 HTML/CSS 代码的耦合，并提升多终端的适配能力，layui 引进了一套具备响应式能力的栅格系统。将容器进行了 12 等分，预设了 4*12 种 CSS 排列类，它们在移动设备、平板、桌面中/大尺寸四种不同的屏幕下发挥着各自的作用。

3.1.2.1. 栅格布局规则

1. 采用 *layui-row* 来定义行，如：

   ```
   <div class="layui-row"></div>
   ```

2. 采用类似 *layui-col-md** 这样的预设类来定义一组列（column），且放在行（row）内。其中：
   - 变量 **md** 代表的是不同屏幕下的标记
   - 变量 **** 代表的是该列所占用的12等分数（如6/12），可选值为 1 - 12
   - 如果多个列的“等分数值”总和等于12，则刚好满行排列。如果大于12，多余的列将自动另起一行。
3. 列可以同时出现最多四种不同的组合，分别是：*xs*（超小屏幕，如手机）、*sm*（小屏幕，如平板）、*md*（桌面中等屏幕）、*lg*（桌面大型屏幕）。
4. 可对列追加类似 *layui-col-space5*、 *layui-col-md-offset3* 这样的预设类来定义列的间距和偏移。
5. 可以在列（column）元素中放入你自己的任意元素填充内容

示例：

```
<h3>常规布局（以中型屏幕桌面为例）：</h3>
<div class="layui-row">
  <div class="layui-col-md9" style="background-color: #00F7DE;">
    你的内容 9/12
  </div>
  <div class="layui-col-md3" style="background-color: rosybrown;">
    你的内容 3/12
  </div>
</div>
```

3.1.2.2. 响应式规则

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

	超小屏幕 (手机 <768px)	**小屏幕 (平板 ≥768px)**	**中等屏幕 (桌面 ≥992px)**	**大型屏幕(桌面 ≥1200px)**
*.layui-container*的值	auto	750px	970px	1170px
标记	xs	sm	md	lg
列对应类 * 为1-12的等分数值	layui-col-xs*	layui-col-sm*	layui-col-md*	layui-col-lg*
总列数	12	12	12	12
响应行为	始终按设定的比例水平排列	在当前屏幕下水平排列，如果屏幕大小低于临界值则堆叠排列	在当前屏幕下水平排列，如果屏幕大小低于临界值则堆叠排列	在当前屏幕下水平排列，如果屏幕大小低于临界值则堆叠排列

```
<h3>平板、桌面端的不同表现：</h3>
<div class="layui-row">
    <div class="layui-col-sm6 layui-col-md4"
         style="background-color: thistle">
        平板≥768px：6/12 | 桌面端≥992px：4/12
    </div>
</div>

<div class="layui-row">
    <div class="layui-col-sm4 layui-col-md6"
         style="background-color: mediumaquamarine;">
        平板≥768px：4/12 | 桌面端≥992px：6/12
    </div>
</div>

<div class="layui-row">
    <div class="layui-col-sm12 layui-col-md8"
         style="background-color: coral">
        平板≥768px：12/12 | 桌面端≥992px：8/12
    </div>
</div>
```

3.1.2.3. 列边距

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

```
/* 支持列之间为 1px-30px 区间的所有双数间隔，以及 1px、5px、15px、25px 的单数间隔 */
layui-col-space1
layui-col-space2
layui-col-space4
layui-col-space5
layui-col-space6
layui-col-space8
layui-col-space10
layui-col-space12
layui-col-space14
layui-col-space15
layui-col-space16
layui-col-space18
layui-col-space20
layui-col-space22
layui-col-space24
layui-col-space25
layui-col-space26
layui-col-space28
layui-col-space30
```

示例：

```
<h3>列间距</h3>
<div class="layui-row layui-col-space10">
    <div class="layui-col-md4" >
        <!-- 需要在layui-col-md4里面再加一层div -->
        <div style="background-color: #009688;">
            1/3
        </div>
    </div>
    <div class="layui-col-md4">
        <div style="background-color: burlywood;">
            1/3
        </div>
    </div>
    <div class="layui-col-md4">
        <div style="background-color: silver;">
            1/3
        </div>
    </div>
</div>
```

注：

1. layui-col-space：设置后不起作用主要是因为**设置的是padding**,也就是说是**向内缩**，所以设置背景色padding也是会添上颜色，看起来好像没有间距一样。可以在里面在加一个div，来达到目的。
2. 间距一般不高于30px，如果超过30，建议使用列偏移。

3.1.2.4. 列偏移

对列追加 类似 *layui-col-md-offset** 的预设类，从而让列向右偏移。其中 * 号代表的是偏移占据的列数，可选中为 1 - 12。

如：*layui-col-md-offset3*，即代表在“中型桌面屏幕”下，让该列向右偏移 3 个列宽度

```
<h3>列偏移</h3>
<div class="layui-row">
    <div class="layui-col-md4" style="background-color: rosybrown;">
        4/12
    </div>
    <div class="layui-col-md4 layui-col-md-offset4"
         style="background-color: cornflowerblue;">
        偏移4列，从而在最右
    </div>
</div>
```

注：列偏移可针对不同屏幕的标准进行设定，在当前设定的屏幕下有效，当低于桌面屏幕的规定的临界值，就会堆叠排列。

3.1.2.5. 列嵌套

可以对栅格进行无穷层次的嵌套。在列元素（*layui-col-md**）中插入行元素（*layui-row*），即可完成嵌套。

```
<h3>列嵌套</h3>
<div class="layui-row layui-col-space5">
    <div class="layui-col-md5" style="background-color: thistle;">
        <div class="layui-row">
            <div class="layui-col-md3" style="background-color: burlywood;" >
                内部列
            </div>
            <div class="layui-col-md5" style="background-color: indianred;">
                内部列
            </div>
            <div class="layui-col-md4" style="background-color:
mediumaquamarine;">
                内部列
            </div>
        </div>
    </div>
```

```
</div>
```

3.2. 基本元素

3.2.1. 按钮

3.2.1.1. 用法

向任意HTML元素设定*class="layui-btn"*，建立一个基础按钮。通过追加格式为*layui-btn-{type}*的class来定义其它按钮风格。

```
<!-- 基础按钮 -->
<button type="button" class="layui-btn">一个标准的按钮</button>
<a href="http://www.layui.com" class="layui-btn">一个可跳转的按钮</a>
<div class="layui-btn">一个按钮</div>
```

3.2.1.2. 主题

名称	组合
原始	class="layui-btn layui-btn-primary"
默认	class="layui-btn"
百搭	class="layui-btn layui-btn-normal"
暖色	class="layui-btn layui-btn-warm"
警告	class="layui-btn layui-btn-danger"
禁用	class="layui-btn layui-btn-disabled"

示例：

```
<!-- 不同主题的按钮 -->
<button class="layui-btn">默认按钮</button>
<button class="layui-btn layui-btn-primary">原始按钮</button>
<button class="layui-btn layui-btn-normal">百搭按钮</button>
<button class="layui-btn layui-btn-warm">暖色按钮</button>
<button class="layui-btn layui-btn-danger">警告按钮</button>
<button class="layui-btn layui-btn-disabled">禁用按钮</button>
```

3.2.1.3. 尺寸

尺寸	组合
大型	class="layui-btn layui-btn-lg"
默认	class="layui-btn"
小型	class="layui-btn layui-btn-sm"
迷你	class="layui-btn layui-btn-xs"

```
<!-- 不同尺寸的按钮 -->
<button class="layui-btn layui-btn-primary layui-btn-lg">大型</button>
<button class="layui-btn">默认</button>
<button class="layui-btn layui-btn-sm layui-btn-danger">小型</button>
<button class="layui-btn layui-btn-xs">迷你</button>
```

3.2.1.4. 圆角

layui-btn-radius

```
<button class="layui-btn layui-btn-radius">默认按钮</button>
<button class="layui-btn layui-btn-primary layui-btn-radius">原始按钮</button>
<button class="layui-btn layui-btn-normal layui-btn-radius">百搭按钮</button>
<button class="layui-btn layui-btn-warm layui-btn-radius">暖色按钮</button>
<button class="layui-btn layui-btn-danger layui-btn-radius">警告按钮</button>
<button class="layui-btn layui-btn-disabled layui-btn-radius">禁用按钮</button>
```

3.2.1.5. 图标

```
<button type="button" class="layui-btn">
    <i class="layui-icon">&#xe608;</i> 添加
</button>

<button type="button" class="layui-btn layui-btn-sm layui-btn-primary">
    <i class="layui-icon">&#x1002;</i> 刷新
</button>
```

3.2.2. 导航

导航一般指页面引导性频道集合，多以菜单的形式呈现，可应用于头部和侧边。面包屑结构简单，支持自定义分隔符。

依赖加载模块：element

实现步骤：

1. 引入的资源

```
<link rel="stylesheet" href="layui/css/layui.css">
<script src="layui/layui.js"></script>
```

2. 依赖加载模块

```
<script type="text/javascript">
    // 注意：导航 依赖 element 模块，否则无法进行功能性操作
    layui.use('element', function(){
        var element = layui.element;

    });
</script>
```

3. 显示指定类型的导航

3.2.2.1. 水平导航

```
<ul class="layui-nav">
  <li class="layui-nav-item"><a href="">最新活动</a></li>
    <li class="layui-nav-item layui-this"><a href="">产品</a></li>
    <li class="layui-nav-item"><a href="">大数据</a></li>
    <li class="layui-nav-item">
        <a href="javascript:;">解决方案</a>
        <dl class="layui-nav-child"> <!-- 二级菜单 -->
            <dd><a href="">移动模块</a></dd>
            <dd><a href="">后台模版</a></dd>
            <dd><a href="">电商平台</a></dd>
        </dl>
    </li>
    <li class="layui-nav-item"><a href="">社区</a></li>
</ul>
```

3.2.2.2. 垂直/侧边导航

```
<ul class="layui-nav layui-nav-tree" >
<!-- 侧边导航: <ul class="layui-nav layui-nav-tree layui-nav-side"> -->
  <li class="layui-nav-item layui-nav-itemed">
    <a href="javascript:;">默认展开</a>
    <dl class="layui-nav-child">
      <dd><a href="javascript:;">选项1</a></dd>
      <dd><a href="javascript:;">选项2</a></dd>
      <dd><a href="">跳转</a></dd>
    </dl>
  </li>
  <li class="layui-nav-item">
    <a href="javascript:;">解决方案</a>
```

```
    <dl class="layui-nav-child">
      <dd><a href="">移动模块</a></dd>
      <dd><a href="">后台模版</a></dd>
      <dd><a href="">电商平台</a></dd>
    </dl>
  </li>
  <li class="layui-nav-item"><a href="">产品</a></li>
  <li class="layui-nav-item"><a href="">大数据</a></li>
</ul>
```

水平、垂直、侧边三个导航的HTML结构是完全一样的，不同的是：

```
水平导航：layui-nav
垂直导航需要追加class：layui-nav-tree
侧边导航需要追加class：layui-nav-tree layui-nav-side
```

3.2.2.3. 导航主题

通过对导航追加CSS背景类，让导航呈现不同的主题色

```
<!-- 如定义一个墨绿背景色的导航 -->
<ul class="layui-nav layui-bg-move" >
  …
</ul>
```

水平导航支持的其它背景主题有：*layui-bg-cyan*（藏青）、*layui-bg-molv*（墨绿）、*layui-bg-blue*（艳蓝）
垂直导航支持的其它背景主题有：*layui-bg-cyan*（藏青）

3.2.2.4. 面包屑

```
<span class="layui-breadcrumb">
  <a href="">首页</a>
  <a href="">国际新闻</a>
  <a href="">亚太地区</a>
  <a><cite>正文</cite></a>
</span>
```

你还可以通过设置属性 *lay-separator=""* 来自定义分隔符。例如：

```
<span class="layui-breadcrumb" lay-separator="-">
  <a href="">首页</a>
  <a href="">国际新闻</a>
  <a href="">亚太地区</a>
  <a><cite>正文</cite></a>
</span>
```

3.2.3. 选项卡

导航菜单可应用于头部和侧边，支持响应式，支持删除选项卡等功能。

依赖加载模块：element

3.2.3.1. 实现步骤

1. 引入的资源

```
<link rel="stylesheet" href="layui/css/layui.css">
<script src="layui/layui.js"></script>
```

2. 依赖加载模块

```
<script type="text/javascript">
    // 注意：选项卡 依赖 element 模块，否则无法进行功能性操作
    layui.use('element', function(){
        var element = layui.element;

    });
</script>
```

3. 加载HTML

```
<div class="layui-tab">
  <ul class="layui-tab-title">
    <li class="layui-this">网站设置</li>
    <li>用户管理</li>
    <li>权限分配</li>
    <li>商品管理</li>
    <li>订单管理</li>
  </ul>
  <div class="layui-tab-content">
    <div class="layui-tab-item layui-show">内容1</div>
    <div class="layui-tab-item">内容2</div>
    <div class="layui-tab-item">内容3</div>
    <div class="layui-tab-item">内容4</div>
    <div class="layui-tab-item">内容5</div>
  </div>
</div>
```

3.2.3.2. 选项卡风格

```
默认风格：layui-tab
简洁风格需要追加class：layui-tab-brief
卡片风格需要追加class：layui-tab-card
```

3.2.3.3. 带删除的选项卡

可以对父层容器设置属性 *lay-allowClose="true"* 来允许Tab选项卡被删除

```
<div class="layui-tab" lay-allowClose="true">
  <ul class="layui-tab-title">
    <li class="layui-this">网站设置</li>
    <li>用户管理</li>
    <li>权限分配</li>
    <li>商品管理</li>
    <li>订单管理</li>
  </ul>
  <div class="layui-tab-content">
    <div class="layui-tab-item layui-show">内容1</div>
    <div class="layui-tab-item">内容2</div>
    <div class="layui-tab-item">内容3</div>
    <div class="layui-tab-item">内容4</div>
    <div class="layui-tab-item">内容5</div>
  </div>
</div>
```

3.2.4. 表格

3.2.4.1. 常规用法

```
<table class="layui-table">
  <colgroup>
    <col width="150">
    <col width="200">
    <col>
  </colgroup>
  <thead>
    <tr>
      <th>昵称</th>
      <th>加入时间</th>
      <th>签名</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>贤心</td>
      <td>2016-11-29</td>
      <td>人生就像是一场修行</td>
    </tr>
    <tr>
      <td>许闲心</td>
      <td>2016-11-28</td>
      <td>于千万人之中遇见你所遇见的人，于千万年之中，时间的无涯的荒野里…</td>
    </tr>
```

```
  </tbody>
</table>
```

3.2.4.2. 基础属性

属性名	属性值	备注
lay-even	无	用于开启 *隔行* 背景，可与其它属性一起使用
lay-skin="属性值"	line （行边框风格） row （列边框风格） nob （无边框风格）	若使用默认风格不设置该属性即可
lay-size="属性值"	sm （小尺寸） lg （大尺寸）	若使用默认尺寸不设置该属性即可

你所需要的基础属性写在table标签上，例如：

```
<!-- 一个带有隔行背景，且行边框风格的大尺寸表格 -->
<table class="layui-table" lay-even lay-size="lg" lay-skin="row">
  <colgroup>
    <col width="150">
    <col width="200">
    <col>
  </colgroup>
  <thead>
    <tr>
      <th>昵称</th>
      <th>加入时间</th>
      <th>签名</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>贤心</td>
      <td>2016-11-29</td>
      <td>人生就像是一场修行</td>
    </tr>
    <tr>
      <td>许闲心</td>
      <td>2016-11-28</td>
      <td>于千万人之中遇见你所遇见的人，于千万年之中，时间的无涯的荒野里…</td>
    </tr>
  </tbody>
</table>
```

3.2.5. 表单

依赖加载模块：form

1. 在一个容器中设定 *class="layui-form"* 来标识一个表单元素块

```
<form class="layui-form" action="">

</form>
```

2. 基本的行区块结构，它提供了响应式的支持。可以换成其他结构，但必须要在外层容器中定义*class="layui-form"*，form模块才能正常工作。

```
<div class="layui-form-item">
    <label class="layui-form-label">标签区域</label>
    <div class="layui-input-block">
      原始表单元素区域
    </div>
</div>
```

3.2.5.1. 输入框

```
<input type="text" name="title" required lay-verify="required" placeholder="请输入标题" autocomplete="off" class="layui-input" />
```

- required：注册浏览器所规定的必填字段
- lay-verify：注册form模块需要验证的类型
- class="layui-input"：layui.css提供的通用CSS类

3.2.5.2. 下拉选择框

```
<select name="city" lay-verify="">
  <option value="">请选择一个城市</option>
  <option value="010">北京</option>
  <option value="021">上海</option>
  <option value="0571">杭州</option>
</select>
```

- 属性*selected*可设定默认项
- 属性*disabled*开启禁用，select和option标签都支持

```
<select name="city" lay-verify="">
  <option value="010">北京</option>
  <option value="021" disabled>上海（禁用效果）</option>
  <option value="0571" selected>杭州</option>
</select>
```

- 可以通过 *optgroup* 标签给select分组

```
<select name="quiz">
  <option value="">请选择</option>
  <optgroup label="城市记忆">
    <option value="你工作的第一个城市">你工作的第一个城市? </option>
  </optgroup>
  <optgroup label="学生时代">
    <option value="你的工号">你的工号? </option>
    <option value="你最喜欢的老师">你最喜欢的老师? </option>
  </optgroup>
</select>
```

- 通过设定属性 *lay-search* 来开启搜索匹配功能

```
<select name="city" lay-verify="" lay-search>
  <option value="010">layer</option>
  <option value="021">form</option>
  <option value="0571" selected>layim</option>
</select>
```

3.2.5.3. 复选框

```
<h2>默认风格：</h2>
<input type="checkbox" name="" title="写作" checked>
<input type="checkbox" name="" title="发呆">
<input type="checkbox" name="" title="禁用" disabled>

<h2>原始风格：</h2>
<input type="checkbox" name="" title="写作" lay-skin="primary" checked>
<input type="checkbox" name="" title="发呆" lay-skin="primary">
<input type="checkbox" name="" title="禁用" lay-skin="primary" disabled>
```

- 属性*title*可自定义文本（温馨提示：如果只想显示复选框，可以不用设置title）
- 属性*checked*可设定默认选中
- 属性*lay-skin*可设置复选框的风格 （原始风格：lay-skin="primary"）
- 设置*value="1"*可自定义值，否则选中时返回的就是默认的on

3.2.5.4. 开关

将复选框checkbox，通过设定 *lay-skin="switch"* 形成了开关风格

```
<input type="checkbox" name="xxx" lay-skin="switch">
<input type="checkbox" name="yyy" lay-skin="switch" lay-text="ON|OFF" checked>
<input type="checkbox" name="zzz" lay-skin="switch" lay-text="开启|关闭">
<input type="checkbox" name="aaa" lay-skin="switch" disabled>
```

- 属性*checked*可设定默认开
- 属性*disabled*开启禁用
- 属性*lay-text*可自定义开关两种状态的文本 （两种文本用 "|" 隔开）

- 设置*value="1"*可自定义值，否则选中时返回的就是默认的on

3.2.5.5. 单选框

```
<input type="radio" name="sex" value="nan" title="男">
<input type="radio" name="sex" value="nv" title="女" checked>
<input type="radio" name="sex" value="" title="中性" disabled>
```

- 属性*title*可自定义文本
- 属性*disabled*开启禁用
- 设置*value="xxx"*可自定义值，否则选中时返回的就是默认的on

3.2.5.6. 文本域

```
<textarea name="remark" required lay-verify="required" placeholder="请输入个人介绍" class="layui-textarea"></textarea>
```

- *class="layui-textarea"*：layui.css提供的通用CSS类

3.2.5.7. 组装行内表单

```
<div class="layui-form-item">
    <div class="layui-inline">
        <label class="layui-form-label">范围</label>
        <div class="layui-input-inline" style="width: 100px;">
          <input type="text" name="price_min" placeholder="￥"
                 autocomplete="off" class="layui-input">
        </div>
        <div class="layui-form-mid">-</div>
        <div class="layui-input-inline" style="width: 100px;">
          <input type="text" name="price_max" placeholder="￥"
                 autocomplete="off" class="layui-input">
        </div>
    </div>
</div>
```

- *class="layui-inline"*：定义外层行内
- *class="layui-input-inline"*：定义内层行内

3.2.5.8. 忽略美化渲染

可以对表单元素增加属性 *lay-ignore* 设置后，将不会对该标签进行美化渲染，即保留系统风格。

```
<input type="radio" name="sex" value="nan" title="男" lay-ignore>
```

3.2.5.9. 表单方框风格

通过追加 *layui-form-pane* 的class，来设定表单的方框风格。

```
<form class="layui-form layui-form-pane" action="">
  <!--
    内部结构都一样，值得注意的是 复选框/开关/单选框 这些组合在该风格下需要额外添加 pane属性（否则  会看起来比较别扭），如：
  -->
  <div class="layui-form-item" pane>
    <label class="layui-form-label">单选框</label>
    <div class="layui-input-block">
      <input type="radio" name="sex" value="男" title="男">
      <input type="radio" name="sex" value="女" title="女" checked>
    </div>
  </div>
</form>
```

3.3. 组件示例

3.3.1. 弹出层

模块加载名称：*layer*，独立版本：layer.layui.com

3.3.1.1. 使用场景

由于layer可以独立使用，也可以通过Layui模块化使用。所以请按照你的实际需求来选择。

场景	用前准备	调用方式
1. 作为独立组件使用	如果你只是单独想使用 layer，你可以去 layer 独立版本官网下载组件包。你需要在你的页面引入*jQuery*1.8以上的任意版本，并引入*layer.js*。	通过script标签引入layer.js后，直接用即可。
2. layui 模块化使用	如果你使用的是 layui，那么你直接在官网下载 layui 框架即可，无需引入 jQuery 和 layer.js，但需要引入*layui.css*和*layui.js*	通过*layui.use('layer', callback)*加载模块

1. 作为独立组件使用 layer

```
<!-- 引入好layer.js后，直接用即可 -->
<script src="layer.js"></script>
<script>
  layer.msg('hello');
</script>
```

2. 在 layui 中使用 layer

```
layui.use('layer', function(){
  var layer = layui.layer;

  layer.msg('hello');
});
```

3.3.1.2. 基础参数

1. type - 基本层类型

```
类型：Number，默认：0

可传入的值有：
  0（信息框，默认）
  1（页面层）
  2（iframe层）
  3（加载层）
  4（tips层）
```

2. title - 标题

```
类型：String/Array/Boolean，默认：'信息'

title支持三种类型的值：
  若传入的是普通的字符串，如 title :'我是标题'，那么只会改变标题文本；
  若需要自定义样式，可以title: ['文本', 'font-size:18px;']，数组第二项可以写任意
css样式；
  若你不想显示标题栏，可以 title: false；
```

3. content - 内容

```
类型：String/DOM/Array，默认：''

content可传入的值是灵活多变的，不仅可以传入普通的html内容，还可以指定DOM。
```

```
 /* 信息框 */
layer.open({
    type:0,
    title:"系统消息",
    // content可以传入任意的文本或html
    content:"Hello"
});

/* 页面层 */
layer.open({
    type:1,
    title:"系统消息",
```

```
    // content可以传入任意的文本或html
    content:"<div style='height:100px;width:200px'>Hello</div>"
});

/* iframe */
layer.open({
    type:2,
    title:"系统消息",
    // content是一个URL，如果你不想让iframe出现滚动条，你还可以content: ['url',
'no']
    content:"http://www.baidu.com"
    // content:["http://www.baidu.com",'no']
});

 /* tips层 */
layer.open({
  type: 4,
  content: ['内容', '#id'] //数组第二项即吸附元素选择器或者DOM
});
```

4. area - 宽高

```
类型：String/Array，默认：'auto'

在默认状态下，layer是宽高都自适应的。
当定义宽度时，你可以area: '500px'，高度仍然是自适应的。
当宽高都要定义时，你可以area: ['500px', '300px']。
```

5. icon - 图标

```
注：信息框和加载层的私有参数。

类型：Number，默认：-1（信息框）/0（加载层）

信息框默认不显示图标。当你想显示图标时，默认层可以传入0-6。如果是加载层，可以传入0-2。
```

```
// eg1
layer.alert('酷毙了', {icon: 1});
// eg2
layer.msg('不开心。。', {icon: 5});
// eg3
layer.load(1); // 风格1的加载
```

6. 示例

```
// eg1
```

```
layer.alert('很高兴见到你😊', {icon: 6});

// eg2
layer.msg('你愿意和我做朋友么？', {
    time: 0, //不自动关闭
    btn: ['当然愿意', '狠心拒绝'], // 按钮
    yes: function(index){
        layer.close(index); // 关闭当前弹出框
        layer.msg('新朋友，你好！', {
            icon: 6, // 图标
            btn: ['开心','快乐']
        });
    }
});

// eg3
layer.msg('这是常用的弹出层');

// eg4
layer.msg('So sad /(ㄒoㄒ)/~~', {icon: 5});

// eg5
layer.msg('玩命加载中...=￣ω￣=', function(){
    // 关闭后的操作
    layer.msg('(⊙o⊙)？');
});
```

3.3.2. 日期与时间选择

模块加载名称：*laydate*，独立版本：http://www.layui.com/laydate/

layDate 包含了大量的更新，其中主要以：*年选择器*、*年月选择器*、*日期选择器*、*时间选择器*、*日期时间选择器* 五种类型的选择方式。

3.3.2.1. 快速使用

和 layer 一样，可以在 layui 中使用 layDate，也可直接使用 layDate 独立版，请按照实际需求来选择。

场景	用前准备	调用方式
1. layui 模块化使用	下载 layui 后，引入*layui.css*和*layui.js*即可	通过*layui.use('laydate', callback)*加载模块后，再调用方法
2. 作为独立组件使用	去 layDate 独立版本官网下载组件包，引入 laydate.js 即可	直接调用方法使用

在layui模块中使用

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>日期与时间选择</title>
    <link rel="stylesheet" href="layui/css/layui.css">
  </head>
  <body>
    <div class="layui-inline">
      <input type="text" class="layui-input" id="date1" />
    </div>
  </body>
  <script src="layui/layui.js"></script>
  <script type="text/javascript">
    layui.use('laydate', function(){
      var laydate = layui.laydate;

      //执行一个laydate实例
      laydate.render({
        elem: '#date1' //指定元素
      });
    });
  </script>
</html>
```

3.3.2.2. 基础参数

1. elem - 绑定元素

   类型：String/DOM，默认值：无

   必填项，用于绑定执行日期渲染的元素，值一般为选择器，或DOM对象

   ```
   laydate.render({
     elem: '#test' //或 elem: document.getElementById('test')、elem: lay('#test') 等
   });
   ```

2. type - 控件选择类型

   类型：String，默认值：date

   用于单独提供不同的选择器类型，可选值如下表：

type可选值	名称	用途
year	年选择器	只提供年列表选择
month	年月选择器	只提供年、月选择
date	日期选择器	可选择：年、月、日。type默认值，一般可不填
time	时间选择器	只提供时、分、秒选择
datetime	日期时间选择器	可选择：年、月、日、时、分、秒

```
//年选择器
laydate.render({
  elem: '#test'
  ,type: 'year'
});

//年月选择器
laydate.render({
  elem: '#test'
  ,type: 'month'
});

//日期选择器
laydate.render({
  elem: '#test'
  //,type: 'date' //默认，可不填
});

//时间选择器
laydate.render({
  elem: '#test'
  ,type: 'time'
});

//日期时间选择器
laydate.render({
  elem: '#test'
  ,type: 'datetime'
});
```

3. format - 自定义格式

类型：String，默认值：yyyy-MM-dd

通过日期时间各自的格式符和长度，来设定一个你所需要的日期格式。layDate 支持的格式如下：

格式符	说明
yyyy	年份，至少四位数。如果不足四位，则前面补零
y	年份，不限制位数，即不管年份多少位，前面均不补零
MM	月份，至少两位数。如果不足两位，则前面补零。
M	月份，允许一位数。
dd	日期，至少两位数。如果不足两位，则前面补零。
d	日期，允许一位数。
HH	小时，至少两位数。如果不足两位，则前面补零。
H	小时，允许一位数。
mm	分钟，至少两位数。如果不足两位，则前面补零。
m	分钟，允许一位数。
ss	秒数，至少两位数。如果不足两位，则前面补零。
s	秒数，允许一位数。

通过上述不同的格式符组合成一段日期时间字符串，可任意排版。

```
//自定义日期格式
laydate.render({
  elem: '#test'
  ,format: 'yyyy年MM月dd日' //可任意组合
});
```

4. value - 初始值

```
类型：String，默认值：new Date()

支持传入符合format参数设定的日期格式字符，或者 new Date()
```

```
// 传入符合format格式的字符给初始值
laydate.render({
  elem: '#test'
  ,value: '2018-08-18' //必须遵循format参数设定的格式
});

// 传入Date对象给初始值
laydate.render({
  elem: '#test'
  ,value: new Date(1534766888000) //参数即为：2018-08-20 20:08:08 的时间戳
});
```

3.3.3. 分页

模块加载名称：*laypage*

3.3.3.1. 快速使用

laypage 的使用非常简单，指向一个用于存放分页的容器，通过服务端得到一些初始值，即可完成分页渲染。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>分页</title>
    <link rel="stylesheet" href="layui/css/layui.css">
  </head>
  <body>
    <div id="test1"></div>
  </body>
  <script src="layui/layui.js"></script>
  <script type="text/javascript">
    layui.use('laypage', function(){
      var laypage = layui.laypage;

      //执行一个laypage实例
      laypage.render({
        elem: 'test1' //注意，这里的 test1 是 ID，不用加 # 号
        ,count: 50 //数据总数，从服务端得到
      });
    });
  </script>
</html>
```

3.3.3.2. 基础参数

通过核心方法：*laypage.render(options)* 来设置基础参数。

参数选项	说明	类型	默认值
elem	指向存放分页的容器，值可以是容器ID、DOM对象。如： 1. elem: 'id' *注意：这里不能加 # 号* 2. elem: document.getElementById('id')	String/Object	-
count	数据总数。一般通过服务端得到	Number	-
limit	每页显示的条数。laypage将会借助 count 和 limit 计算出分页数。	Number	10
limits	每页条数的选择项。如果 layout 参数开启了 limit，则会出现每页条数的select选择框	Array	[10, 20, 30, 40, 50]

curr	起始页。一般用于刷新类型的跳页以及HASH跳页。如：
	// 开启location.hash的记录
	laypage.render({
	elem: 'test1' ,
	count: 500 ,
	// 获取起始页
	curr: location.hash.replace('#!fenye=', '')
	// 自定义hash值
	,hash: 'fenye'
	}); `
		Number	1

groups	连续出现的页码个数	Number	5
prev	自定义“上一页”的内容，支持传入普通文本和HTML	String	上一页
next	自定义“下一页”的内容，同上	String	下一页
first	自定义“首页”的内容，同上	String	1
last	自定义“尾页”的内容，同上	String	总页数值
layout	自定义排版。可选值有：*count*（总条目输区域）、*prev*（上一页区域）、*page*（分页区域）、*next*（下一页区域）、*limit*（条目选项区域）、*refresh*（页面刷新区域。注意：layui 2.3.0 新增） 、*skip*（快捷跳页区域）	Array	['prev', 'page', 'next']
theme	自定义主题。支持传入：*颜色值*，或*任意普通字符*。如： 1. theme: '#c00' 2. theme: 'xxx' //将会生成 class="layui-laypage-xxx" 的CSS类，以便自定义主题	String	-
hash	开启location.hash，并自定义 hash 值。如果开启，在触发分页时，会自动对url追加：*#!hash值={curr}* 利用这个，可以在页面载入时就定位到指定页	String/Boolean	false

3.3.3.3. jump - 切换分页的回调

当分页被切换时触发，函数返回两个参数：*obj*（当前分页的所有选项值）、first（是否首次，一般用于初始加载的判断）

```
laypage.render({
```

```
    elem: 'page'
    ,count: 100 //数据总数，从服务端得到
    ,groups:10 // 连续出现的页码个数
    ,layout:['prev', 'page', 'next','limit','count'] // 自定义排版
    ,limits:[5,10,20] // layout属性设置了limit值，可会出现条数下拉选择框
    ,jump: function(obj, first){
        // obj包含了当前分页的所有参数，比如：
        console.log(obj.curr); //得到当前页，以便向服务端请求对应页的数据。
        console.log(obj.limit); //得到每页显示的条数

        //首次不执行
        if(!first){
            //do something
        }
    }
});
```

3.3.4. 数据表格

模块加载名称：*table*

3.3.4.1. 快速使用

创建一个table实例最简单的方式是，在页面放置一个元素

，然后通过 *table.render()* 方法指定该容器。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>table数据表格</title>
    <link rel="stylesheet" href="layui/css/layui.css">
  </head>
  <body>
    <table id="demo"></table>
  </body>

  <script src="layui/layui.js" type="text/javascript" ></script>
  <script type="text/javascript">
    layui.use('table', function(){
      var table = layui.table;

      // 第一个实例
      table.render({
      elem: '#demo'
      ,url: 'user.json' // 数据接口
      ,cols: [[ // 表头
```

```
        {field: 'id', title: 'ID', width:80, sort: true, fixed: 'left'}
        ,{field: 'username', title: '用户名', width:80}
        ,{field: 'sex', title: '性别', width:80, sort: true}
        ,{field: 'city', title: '城市', width:80}
        ,{field: 'sign', title: '签名', width: 177}
      ]]
      });

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

数据接口 user.json

```
{
  "code": 0,
  "msg": "",
  "count": 50,
  "data": [{
      "id": 10000,
      "username": "user-0",
      "sex": "女",
      "city": "城市-0",
      "sign": "签名-0"
    },
    {
      "id": 10001,
      "username": "user-1",
      "sex": "男",
      "city": "城市-1",
      "sign": "签名-1"
    },
    {
      "id": 10002,
      "username": "user-2",
      "sex": "女",
      "city": "城市-2",
      "sign": "签名-2"
    },
    {
      "id": 10003,
      "username": "user-3",
      "sex": "女",
      "city": "城市-3",
      "sign": "签名-3"
    },
    {
      "id": 10004,
      "username": "user-4",
```

```
      "sex": "男",
      "city": "城市-4",
      "sign": "签名-4"
    }

  ]
}
```

3.3.4.2. 三种初始化渲染方式

机制	适用场景	
方法渲染	用JS方法的配置完成渲染	（推荐）无需写过多的 HTML，在 JS 中指定原始元素，再设定各项参数即可。
自动渲染	HTML配置，自动渲染	无需写过多 JS，可专注于 HTML 表头部分
转换静态表格	转化一段已有的表格元素	无需配置数据接口，在JS中指定表格元素，并简单地给表头加上自定义属性即可

3.3.4.3. 方法渲染

1. 将基础参数的设定放在了JS代码中，且原始的 table 标签只需要一个*选择器*

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

2. 渲染表格

   ```
   layui.use('table', function(){
        var table = layui.table;

        // 执行渲染
        table.render({
           elem: '#demo' // 指定原始表格元素选择器（推荐id选择器）
           ,url: 'user.json' // 数据接口
           ,height: 315 // 容器高度
           ,page:true // 开启分页
           ,cols: [[  // 设置表头
               {field: 'id', title: 'ID'}
               ,{field: 'username', title: '用户名'}
               ,{field: 'sex', title: '性别'}
           ]]
        });
   });
   ```

   注：*table.render()*方法返回一个对象：var tableIns = table.render(options)，可用于对当前表格进行“重载”等操作。

3.3.4.4. 自动渲染

在一段 table 容器中配置好相应的参数，由 table 模块内部自动对其完成渲染，而无需你写初始的渲染方法。

1) 带有 *class="layui-table"* 的

标签。 2) 对标签设置属性 *lay-data=""* 用于配置一些基础参数 3) 在

标签中设置属性*lay-data=""*用于配置表头信息

```
<table class="layui-table" lay-data="{url:'user.json'}">
  <thead>
    <tr>
      <th lay-data="{field:'id'}">ID</th>
      <th lay-data="{field:'username'}">用户名</th>
      <th lay-data="{field:'sex', sort: true}">性别</th>
    </tr>
  </thead>
</table>
```

3.3.4.5. 转换静态表格

页面已经存在了一段有内容的表格，由原始的table标签组成，只需要赋予它一些动态元素。

```
<table lay-filter="demo">
    <thead>
        <tr>
            <th lay-data="{field:'username', width:100}">昵称</th>
            <th lay-data="{field:'experience', width:80, sort:true}">积
分</th>
            <th lay-data="{field:'sign'}">签名</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>贤心1</td>
            <td>66</td>
            <td>人生就像是一场修行a</td>
        </tr>
    </tbody>
</table>
```

执行用于转换表格的JS方法

```
layui.use('table', function(){
    var table = layui.table;

    // 转换静态表格
    table.init('demo', {
        height: 315 // 设置高度
        // 支持所有基础参数
    });

});
```

3.3.4.6. 基础参数使用的场景

```
// 场景一：下述方法中的键值即为基础参数项
table.render({
  height: 300
  ,url: 'user.json'
});

// 场景二：下述 lay-data 里面的内容即为基础参数项，切记：值要用单引号
<table lay-data="{height:300, url:'user.json'}" lay-filter="demo"> ……
</table>

// 更多场景：下述 options 即为含有基础参数项的对象
> table.init('filter', options); // 转化静态表格
> var tableObj = table.render({});
  tableObj.reload(options); // 重载表格
```

3.3.4.7. 开启分页

```
<!-- HTML代码 -->
<table id="demo"></table>
```

```
// JS实现代码
layui.use('table', function(){
    var table = layui.table;

    // 执行渲染
    table.render({
        elem: '#demo' // 指定原始表格元素选择器（推荐id选择器）
        ,url: 'user.json' // 数据接口
        ,height: 315 // 容器高度
        ,page:true // 开启分页
        ,cols: [[  // 设置表头
            {field: 'id', title: 'ID'}
            ,{field: 'username', title: '用户名'}
            ,{field: 'sex', title: '性别'}
```

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

3.3.4.8. 开启头部工具栏

```
<table id="demo" lay-filter="demo"></table>

<!-- 表格工具栏 -->
<script type="text/html" id="toolbarDemo">
  <div class="layui-btn-container">
    <!-- lay-event 给元素绑定事件名 -->
    <button class="layui-btn layui-btn-sm" lay-event="getCheckData">
      获取选中行数据
      </button>
    <button class="layui-btn layui-btn-sm" lay-event="getCheckLength">
      获取选中数目
      </button>
    <button class="layui-btn layui-btn-sm" lay-event="isAll">
      验证是否全选
      </button>
    </div>
</script>

<!-- 表头工具栏 -->
<script type="text/html" id="barDemo">
  <a class="layui-btn layui-btn-xs" lay-event="edit">编辑</a>
  <a class="layui-btn layui-btn-danger layui-btn-xs" lay-event="del">删除
</a>
</script>
```

```
layui.use('table', function(){
    var table = layui.table;

    // 执行渲染
    table.render({
        elem: '#demo' // 指定原始表格元素选择器（推荐id选择器）
        ,url: 'user.json' // 数据接口
        ,height: 315 // 容器高度
        ,page:true // 开启分页
        ,cols: [[  // 设置表头
            {field: 'id', title: 'ID'}
            ,{field: 'username', title: '用户名'}
            ,{field: 'sex', title: '性别'}
            ,{title:'操作', toolbar: '#barDemo'} // 绑定表头工具栏
        ]]
        ,toolbar: '#toolbarDemo' // 开启头部工具栏，并为其绑定左侧模板
```

```
});

/**
* 头工具栏事件
* 语法：
  table.on('toolbar(demo)', function(obj){

  });
  注：demo表示选择器元素上设置的lay-filter属性值
*/
table.on('toolbar(demo)', function(obj){
    // obj.config.id 当前选择器的id属性值，即demo
    // 获取当前表格被选中的记录对象，返回数组
    var checkStatus = table.checkStatus(obj.config.id);
    // obj.event 得到当前点击元素的事件名
    switch(obj.event){
        case 'getCheckData':
            // 获取被选中的记录的数组
            var data = checkStatus.data;
            // 将数组数据解析成字符串
            layer.alert(JSON.stringify(data));
            break;
        case 'getCheckLength':
            var data = checkStatus.data;
            layer.msg('选中了：'+ data.length + ' 个');
            break;
        case 'isAll':
            // checkStatus.isAll 判断记录是否被全选
            layer.msg(checkStatus.isAll ? '全选': '未全选');
            break;
            // 自定义头工具栏右侧图标 - 提示
        case 'LAYTABLE_TIPS':
            layer.alert('这是工具栏右侧自定义的一个图标按钮');
            break;
    };
});

/**
* 监听行工具事件
*/
table.on('tool(demo)', function(obj){
    // 得到当前操作的tr的相关信息
    var data = obj.data;
    if(obj.event === 'del'){
        // 确认框
        layer.confirm('真的删除行么', function(index){
            // 删除指定tr
            obj.del();
            // index 当前弹出层的下标，通过下标关闭弹出层
```

```
                layer.close(index);
            });
        } else if(obj.event === 'edit'){
            // 输入框
            layer.prompt({
                // 表单元素的类型 0=文本框 1=密码框 2=文本域
                formType: 0
                ,value: data.username
            }, function(value, index){
                // 修改指定单元格的值
                // value表示输入的值
                obj.update({
                    username: value
                });
                // 关闭弹出层
                layer.close(index);
            });
        }
    });

});
```

3.3.4.9. 开启单元格编辑

```
<table class="layui-table" lay-data="{url:'user.json', id:'demo'}" lay-
filter="demo">
    <thead>
        <tr>
            <th lay-data="{type:'checkbox'}">ID</th>
            <th lay-data="{field:'id',  sort: true}">ID</th>
            <th lay-data="{field:'username', sort: true, edit: 'text'}">用
户名</th>
            <th lay-data="{field:'sex', edit: 'text'}">性别</th>
            <th lay-data="{field:'city', edit: 'text'}">城市</th>
            <th lay-data="{field:'experience', sort: true, edit: 'text'}">
积分</th>
        </tr>
    </thead>
</table>
```

```
layui.use('table', function(){
    var table = layui.table;

    // 监听单元格编辑
    table.on('edit(demo)', function(obj){
        var value = obj.value // 得到修改后的值
        ,data = obj.data // 得到所在行所有键值
        ,field = obj.field; // 得到字段
        layer.msg('[ID: '+ data.id +'] ' + field + ' 字段更改为: '+ value);
    });

});
```

3.3.4.10. 数据表格的重载

```
<div class="demoTable">
    搜索ID:
    <div class="layui-inline">
        <input class="layui-input" name="id" id="demoReload"
autocomplete="off">
    </div>
    <button class="layui-btn" id="searchBtn">搜索</button>
</div>

<table class="layui-hide" id="demo" lay-filter="demo"></table>
```

```
layui.use('table', function(){
    var table = layui.table;
    var $ = layui.jquery; // 获取jquery对象

    // 执行渲染
    table.render({
        elem: '#demo' // 指定原始表格元素选择器（推荐id选择器）
        ,url: 'user.json' // 数据接口
        ,page:true // 开启分页
        ,cols: [[  // 设置表头
            {type: 'checkbox', fixed: 'left'} // 设置复选框
            ,{field: 'id', title: 'ID'}
            ,{field: 'username', title: '用户名'}
            ,{field: 'sex', title: '性别'}
        ]]
    });

    // 给指定元素绑定事件
    $(document).on('click', '#searchBtn', function(data) {
        // 获取搜索框对象
        var demoReload = $('#demoReload');
        table.reload('demo', {
```

```
            where: { // 设定异步数据接口的额外参数，任意设
                id: demoReload.val()
            }
            ,page: {
                // 让条件查询从第一页开始查询，不写则从当前页开始查询，此页之前的数据
将不纳入条件筛选
                curr: 1 // 重新从第 1 页开始
            }
        }); // 只重载数据
    });
});
```