

# 弹出层

## 弹出层

弹出层官方文档: <http://layui-doc.pearadmin.com/layer/index.html>

layer 可以独立使用，也可以通过Layui模块化使用

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jQuery1.8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

### 作为独立组件使用

1. 首先去 layer 独立版本官网下载组件包。
2. 下载完成后解压，将 layer.js 和 layer.css 拷贝到我们的项目中
3. 引入资源

```
1 <!-- 引入layer.css -->
2 <link rel="stylesheet" href="layer/layer.css" />
3 <!-- 引入jquery.js -->
4 <script src="js/jquery-3.4.1.js" charset="utf-8"></script>
5 <!-- 引入layer.js -->
6 <script src="layer/layer.js"></script>
```

4. 开始使用

```
1 <body>
2   <script type="text/javascript">
3     layer.msg("Hello");
4   </script>
5 </body>
```

### 使用模块化

1. 引入资源

```
1 <!-- layui模块化使用 -->
2 <!-- 引入 layui.css -->
3 <link rel="stylesheet" href="../src/css/layui.css">
```

2. 在 script 中使用 layui.use() 加载模块

1. 依赖模块: layer

```

1 <body>
2 <script src="../../src/layui.js"></script>
3 <script>
4   layui.use(['layer'], function () {
5     var layer = layui.layer
6     layer.msg('Hello')
7   })
8 </script>
9 </body>

```

## 基础参数

### type 基本层类型

- 类型 Number，默认为0
- layer 提供了5种层类型。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
- 若采用 layer.open({type:1}) 方式调用，则 type 为必填项(信息框除外)

### 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；
  - 若需要自定义标题区域样式，title:['文本','font-size: 18px']，数组第二项可以写任意css 样式
  - 若不想显示标题栏，title: false

```

1 <script>
2   layui.use(function () {
3     let $ = layui.$
4     let layer = layui.layer
5
6     /* 信息框 */
7     layer.open({
8       type: 0, // 0为信息框
9       title: '系统消息',
10      // title: false, //不显示标题
11      // title: ['标题', 'color:red;'], // 自定义标题区域样式
12
13      // content可以传入任意的文本或html
14      content: 'Hello',
15    })
16  },
17 )
18 </script>

```

### area宽高

- 类型：String/Array，默认为auto

- 在默认状态下，layer是宽高都自适应的
  - 但当你只想定义宽度时，你可以 area: '500px'，高度仍然是自适应的
  - 当你宽高都要定义时，你可以 area: ['500px', '300px']

## btn按钮

- 类型：String/Array，默认 '确认'
- 信息框(type = 0)模式时，btn默认是一个确认按钮，其它层类型则默认不显示，加载层和tips层则无效。
- 当您只想自定义一个按钮时，你可以 btn: '我知道了'，当你要定义两个按钮时，你可以btn: ['yes', 'no']
- 当然，你也可以定义更多按钮，比如：btn: ['按钮1', '按钮2', '按钮3', ...]，按钮1的回调是yes，而从按钮2开始，则回调为btn2: function(){}，以此类推

```

1 layer.msg('你愿意和我做朋友么？', {
2   time: 0, //不自动关闭
3   btn: ['当然愿意', '狠心拒绝'], // [按钮1,按钮2]
4
5   // 按钮1的回调函数
6   yes: function (index) {
7     layer.close(index) // 关闭当前弹出框
8     layer.msg('新朋友，你好！', {
9       icon: 6, // 图标
10      btn: ['开心', '快乐'],
11    })
12  },
13
14  // 按钮2的回调函数
15  btn2: function (index) {
16    layer.close(index) // 关闭当前弹出框
17    layer.msg('好吧,再见!', {
18      icon: 5,
19      btn: '88',
20    })
21  },
22 })

```

## time自动关闭所需毫秒

- 类型：Number，默认为0
- 默认不会关闭。当你想关闭时，可以 time:5000，即代表 5 s 后自动关闭

## 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，更可以随着 type 的不同而不同
  - 页面层,就是信息提示
  - iframe 弹出来的是页面,例如百度页面
  - tips 就是一个信息提示小框

```

1 /* 页面层 */
2 layer.open({
3   type: 1,

```

```

4     title: "系统消息",
5     // content可以传入任意的文本或html
6     content: "<div style='height:200px;width:400px'>Hello</div>"
7 });
8
9  /* iframe层 */
10 layer.open({
11     type: 2,
12     title: "系统消息",
13     // content是一个URL, 如果你不想让iframe出现滚动条, 你还可以content: ['url',
14     'no']
15     content: "http://www.baidu.com",
16     // content:["http://www.baidu.com",'no'],
17     // area: '500px' ,// 设置宽度, 高度自适应Q
18     area: ['800px', '400px'] // 设置宽高
19 });
20
21 /* tips层 */
22 layer.open({
23     type: 4,
24     content: ['内容', '#btn1'], //数组第二项即吸附元素选择器或者DOM

```

## icon图标

- 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, 默认皮肤可以传入0-6
- 如果是加载层, 可以传入0-2

```

1  //加载层
2  layer.alert('酷毙了', {
3      icon: 1          // 0 ~ 6 均可填
4  });
5
6  layer.msg('不开心。。', {
7      icon: 5,          // -1 ~ 6 均可填
8  });
9
10 // 加载层
11 layer.load(1)        // 0 ~ 2 均可填

```

我们的信息框还可以参与互动响应,会有多个选项可选:

```

1  layer.msg('你愿意和我做朋友么?', {
2      time: 0, //第一个弹出层不自动关闭(因为默认弹出层会5s自动关闭)
3      btn: ['当然愿意', '狠心拒绝'], // 按钮
4      yes: function (index) {
5          layer.close(index) // 关闭当前弹出框
6          layer.msg('新朋友, 你好!', {
7              icon: 6, // 图标
8              btn: ['开心', '快乐'],
9          })
10     },
11 })

```

# 核心方法

## layer.open(options)

- 原始核心方法
- 创建任何类型的弹层都会返回一个当前层索引，上述的 options 即是基础参数

```
1      var index = layer.open({
2          content: 'test',
3      })
4
5      console.log(index)
6      // 拿到的 index 是一个重要的凭据，它是诸如 layer.close(index) 等方法的必传参
      数。
```

## layer.alert()

- 普通信息框

```
1      // layer.alert('只想简单的提示');
2      layer.alert('加了个图标', {
3          icon: 1,
4      })
5
```

## layer.msg()

- 提示框
- 默认是 3s 关闭

```
1      /*默认提示框*/
2      // 案例 1
3      layer.msg('只想弱弱提示')
4      // 案例 2
5      layer.msg('有表情地提示', {
6          icon: 6,
7      })
8      // 案例3
9      layer.msg('关闭后想做什么', function () {
10         console.log('do something')
11     })
12     layer.msg('同上', {
13         icon: 1,
14         time: 2000, //2秒关闭（如果不配置，默认是3秒）
15     }, function () {
16         console.log('do something')
17     })
```

## layer.load()

- 加载层
- 加载层默认是不会自动关闭的

```

1      /*加载层*/
2      var index = layer.load()
3      var index = layer.load(1) //换了种风格
4      var index = layer.load(2, {
5          time: 10 * 1000,
6      }) //又换了种风格，并且设定最长等待10秒
7
8      //关闭
9      layer.close(index)

```

## 时间和日期选择器

日期与时间选择官方文档：<https://www.layui.com/doc/modules/laydate.html>

和 layer 一样，你可以在 layui 中使用 layDate，也可直接使用 layDate 独立版

场景	用前准备	调用方式
1. 在 layui 模块中使用	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

我们使用模块化使用

### 1. 引入资源

```

1      <!-- layui模块化使用 -->
2      <!-- 引入 layui.css -->
3      <link rel="stylesheet" href="layui-v2.5.6/layui/css/layui.css" />
4      <!-- 引入 layui.js -->
5      <script src="layui-v2.5.6/layui/layui.js"></script>

```

### 2. 在 script 中使用 layui.use() 加载模块

- 依赖模块：laydate

```

1      <body>
2      <!-- 用一个容器元素放我们的日期时间选择器 -->
3      <div class="layui-inline">
4          <input type="text" class="layui-input" id="date1" />
5      </div>
6      <script src="../src/layui.js"></script>
7      <script>
8          layui.use('laydate', function () {
9              var laydate = layui.laydate
10             // 加载 laydate 实例
11             laydate.render({
12                 elem: '#date1', //绑定id为date1的元素
13             })
14         })
15     </script>
16 </body>

```

## 基础参数

### 基础参数选项

- 通过核心方法：laydate.render(options) 来设置基础参数，

### elem绑定元素

- 类型：String/DOM，默认值：无
- 必填项，用于执行绑定日期渲染的元素，值一般为选择器，或DOM对象

```
1 <script>
2     layui.use('laydate',function(){
3         var laydate = layui.laydate;
4         laydate.render({
5             elem: '#test' //或 elem: document.getElementById('test')、elem:
lay('#test') 等
6         })
7     })
8 </script>
```

### 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

```
1 <body>
2 <!-- 用一个容器元素放我们的日期时间选择器 -->
3 <div class="layui-inline">
4     <input type="text" class="layui-input" id="date1"/>
5 </div>
6 <div class="layui-inline">
7     <input type="text" class="layui-input" id="date2"/>
8 </div>
9 <div class="layui-inline">
10    <input type="text" class="layui-input" id="date3"/>
11 </div>
12 <div class="layui-inline">
13    <input type="text" class="layui-input" id="date4"/>
14 </div>
15 <div class="layui-inline">
16    <input type="text" class="layui-input" id="date5"/>
17 </div>
18 <div class="layui-inline">
```

```

19     <input type="text" class="layui-input" id="date6"/>
20 </div>
21 <script src="../../src/layui.js"></script>
22 <script>
23     layui.use('laydate', function () {
24         var laydate = layui.laydate
25         // 加载 laydate 实例
26         laydate.render({
27             elem: '#date1', //绑定id为date1的元素
28
29         })
30
31         laydate.render({
32             elem: '#date2', //绑定id为date2的元素
33             type: 'year', // 年选择器
34         })
35
36         laydate.render({
37             elem: '#date3', //绑定id为date3的元素
38             type: 'month', // 年月选择器
39         })
40
41         laydate.render({
42             elem: '#date4', //绑定id为date4的元素
43             type: 'date', // 年月日选择器
44         })
45
46         laydate.render({
47             elem: '#date5', //绑定id为date5的元素
48             type: 'time', // 时间（时分秒）选择器
49         })
50
51         laydate.render({
52             elem: '#date6', //绑定id为date6的元素
53             type: 'datetime', // 年月日时分秒选择器
54         })
55     })
56 </script>
57 </body>

```

## 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，如下所示

格式	示例值
yyyy-MM-dd HH:mm:ss	2017/8/18 20:08
yyyy年MM月dd日 HH时mm分ss秒	2017年08月18日 20时08分08秒
yyyyMMdd	20170818
dd/MM/yyyy	18/08/2017
yyyy年M月	2017年8月
M月d日	8月18日
北京时间：HH点mm分	北京时间：20点08分
yyyy年的M月某天晚上，大概H点	2017年的8月某天晚上，大概20点

```
1 <body>
2 <!--用一个容器元素放我们的日期时间选择器-->
3 <div class="layui-inline">
4   <input type="text" class="layui-input" id="date1"/>
5 </div>
6 <script src="../../src/layui.js"></script>
7 <script>
8   layui.use('laydate', function () {
9     var laydate = layui.laydate
10    //自定义日期格式
11    laydate.render({
12      elem: '#date1',
13      format: 'yyyy/MM月dd日', // yyyy年MM月dd日可任意组合
```

```

14     })
15   })
16 </script>
17 </body>

```

## 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()

```

1 <body>
2 <!--用一个容器元素放我们的日期时间选择器-->
3 <div class="layui-inline">
4   <input type="text" class="layui-input" id="date1"/>
5 </div>
6 <div class="layui-inline">
7   <input type="text" class="layui-input" id="date2"/>
8 </div>
9 <script src="../../src/layui.js"></script>
10 <script>
11   layui.use('laydate', function () {
12     var laydate = layui.laydate
13     // 传入符合format格式的字符给初始值
14     laydate.render({
15       elem: '#date1',
16       value: '2018-08-18', //必须遵循format参数设定的格式
17     })
18     // 传入Date对象给初始值
19     laydate.render({
20       elem: '#date2',
21       value: new Date(1534766888000), //参数即为: 2018-08-20 20:08:08 的时间戳
22     })
23   })
24 </script>
25 </body>
26

```

## lang语言

- 类型: String, 默认: cn
- 内置了两种语言版本: cn (中文版)、en (国际版, 即英文版)。

```

1 // 传入Date对象给初始值
2 laydate.render({
3   elem: '#date2',
4   value: new Date(1534766888000), //参数即为: 2018-08-20 20:08:08 的时间戳
5   lang: 'en',
6 })
7

```

## theme主题

- 类型: String, 默认值: default
- 内置了多种主题, theme的可选值有: default (默认简约)、molv (墨绿背景)、#颜色值 (自定义颜色背景)、grid (格子主题)

```

1 | laydate.render({
2 |   elem: '#date2',
3 |   value: new Date(1534766888000), //参数即为: 2018-08-20 20:08:08 的时间戳
4 |   lang: 'en',
5 |   theme: 'molv'
6 | })
7 |

```

## calendar公历节日

- 类型: Boolean, 默认值: false
- 内置了一些我国通用的公历重要节日, 通过设置 true 来开启。国际版不会显示

```

1 | // 传入Date对象给初始值
2 | laydate.render({
3 |   elem: '#date2',
4 |   value: new Date(1534766888000), //参数即为: 2018-08-20 20:08:08 的时间戳
5 |   // lang: 'en',
6 |   theme: 'molv',
7 |   calendar: true
8 | })

```

## 相关属性

elem:绑定的页面input标签的id

type:指定选择器的类型

|--年year

|--年月month

|--年月日date 默认的

|--时间 time

|--年月日时间 datetime

format 格式化时间

格式	示例值
yyyy-MM-dd HH:mm:ss	2017-08-18 20:08:08
yyyy年MM月dd日 HH时mm分ss秒	2017年08月18日 20时08分08秒
yyyyMMdd	20170818
dd/MM/yyyy	18/08/2017
yyyy年M月	2017年8月
M月d日	8月18日
北京时间：HH点mm分	北京时间：20点08分
yyyy年的M月某天晚上，大概H点	2017年的8月某天晚上，大概20点

value :设置初始值

value:'2019-11-11'指定死的

value:new Date() 指定当前时间

min/max 设置最大时间和最小时间

```
laydate.render({
  elem: '#test3' //指定元素
  //,value:'2019-11-11' 指定具体的时间
  ,value:new Date() //指定当前系统时间
  /* ,min:'2019-01-01'
  ,max:'2025-12-31' */
  /* ,min: -7
  ,max: 7 */
  ,type: 'time'
  ,min: '09:30:00'
  ,max: '17:30:00'
});
```

trigger 触发弹出事件类型

默认为得到焦点 trigger: 'focus'

可以设置成点击事件 trigger:'click'

show:设置是否在页面加载时选择器就弹出来 默认为false

showBottom 是否显示底部栏

btns - 工具按钮

类型: Array, 默认值: ['clear', 'now', 'confirm']

右下角显示的按钮，会按照数组顺序排列，内置可识别的值有：clear、now、confirm

lang - 语言

默认为cn 可以以设置en 英文

theme - 主题

default（默认简约）、molv（墨绿背景）、#颜色值（自定义颜色背景）、grid（格子主题）

calendar - 是否显示公历节日

Boolean，默认值：false

mark - 标注重要日子

```
mark : {  
  '0-0-15': '工资',  
  '0-9-1': '开学'  
}
```

控件初始打开的回调

```
, ready: function(date) {  
  console.log(date); //得到初始的E  
}
```

日期时间被切换后的回调

```
laydate.render({  
  elem: '#test'  
  , change: function(value, date, endDate) {  
    console.log(value); //得到日期生成的值，如：2017-08-  
    console.log(date); //得到日期时间对象：{year: 2017,  
    console.log(endDate); //得结束的日期时间对象，开启范围  
  }  
});
```

控件选择完毕后的回调、

```
001. laydate.render({  
002.   elem: '#test'  
003.   , done: function(value, date, endDate) {  
004.     console.log(value); //得到日期生成的值，如：2017-08-18  
005.     console.log(date); //得到日期时间对象：{year: 2017, month: 8, date: 18, hours: 0, mi  
006.     console.log(endDate); //得结束的日期时间对象，开启范围选择（range: true）才会返回。对象成  
007.   }  
008. });  
009.
```

## 相关代码

```
1 <div class="layui-inline">
2     <!-- 注意：这一层元素并不是必须的 -->
3     <input type="text" class="layui-input" id="test1">
4 </div>
5 <input type="text" class="layui-input" id="test2">
6
7 <div class="layui-form-item">
8     <div class="layui-inline">
9         <label class="layui-form-label">日期范围</label>
10        <div class="layui-inline" id="test-range">
11            <div class="layui-input-inline">
12                <input type="text" id="startDate" class="layui-input"
placeholder="开始日期">
13            </div>
14            <div class="layui-form-mid"></div>
15            <div class="layui-input-inline">
16                <input type="text" id="endDate" class="layui-input"
placeholder="结束日期">
17            </div>
18        </div>
19    </div>
20 </div>
21
22 <script src="static/layui.js"></script>
23 <script>
24     layui.use(function () {
25         let lay_date = layui.laydate
26
27         // lay_date
28         lay_date.render({
29             elem: '#test1' //指定元素 或 elem: document.getElementById('test')、
elem: lay('#test') 等
30             // , type: 'year' // 年份选择器
31             // , type: 'month' // 年月选择器
32             , type: 'date' // 默认, 可不填
33             // , type: 'time' // 时间选择器
34             // , type: 'datetime' // 日期时间选择器
35
36             , value: '2018-08-18' //设置初始值 必须遵循format参数设定的格式
37             , isPreview: false, //禁用面板左下角选择值的预览, 默认 true
38         })
39
40         lay_date.render({
41             elem: '#test2'
42             , range: true, //或 range: '~' 来自定义分割字符
43         })
44
45         // 分开选择范围
46         lay_date.render({
47             elem: '#test-range' //开始时间和结束时间所在 input 框的父选择器
48             //设置开始日期、日期日期的 input 选择器
49             , range: ['#startDate', '#endDate'] //数组格式为 layui 2.6.6 开始新增
50             , format: 'yyyy年MM月dd日', //可任意组合
```

```
51     })
52   })
53 </script>
54
```

## 分页组件

- 分页官方文档: <https://www.layui.com/doc/modules/laypage.html>
- 模块加载名称: laypage
- 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

```
1 <body>
2 <div id="page"></div>
3
4 <script src="../../src/layui.js" charset="utf-8"></script>
5 <script>
6   // 加载laypage模块
7   layui.use(function () {
8     var laypage = layui.laypage
9     , layer = layui.layer
10    // 加载laypage实例
11    laypage.render({
12      elem: 'page', // elem属性绑定的是容器的ID属性值, 不需要加#
13      count: 100, // 总数量, 一般是从服务器获取
14    })
15  })
16 </script>
17 </body>
```

## 基础参数

### 基础参数选项

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

### elem绑定元素

- 类型: String/Object, 必填项
- elem 指向存放分页的容器, 值可以是 容器 ID、DOM对象
  - elem: 'id' 注意: 这里不能加 # 号
  - elem: document.getElementById('id')

### count数据总数

- 类型: Number, 必填项
- 数据总数, 一般通过服务端得到
- count: 100

### limit每页显示条数

- 类型: Number, 默认值 10
- laypage 将会借助 count 和 limit 计算出分页数。

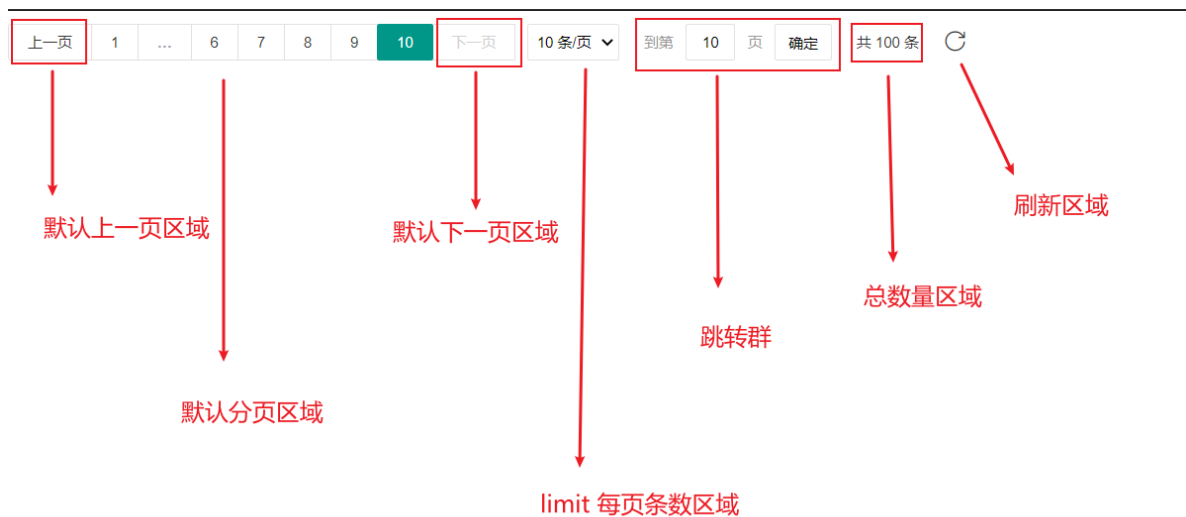
### layout自定义排版

- 类型: Array, 默认值: ['prev', 'page', 'next']
- 自定义排版, 可选值有:
  - count: 总条目输出区域
  - limit: 条目选项区域
  - prev: 上一页区域
  - page: 分页区域
  - next: 下一页区域
  - refresh: 页面刷新区域(layui 2.3.0新增)
  - skip: 快捷跳页区域

```

1 <body>
2 <div id="page"></div>
3
4 <script src="../../src/layui.js" charset="utf-8"></script>
5 <script>
6   layui.use(function () {
7     var laypage = layui.laypage
8       , layer = layui.layer
9     laypage.render({
10      elem: 'page',
11      count: 100,
12      layout: ['prev', 'page', 'next', 'limit', 'skip', 'count', 'refresh'],
13    })
14  })
15 </script>
16 </body>

```



limits每页条数的选择项



- 类型: Array, 默认值: [10,20,30,40,50]
- 如果 layout 参数开启了 limit, 则会出现每页条数的select下拉选择框

```

1 laypage.render({
2   elem: 'page',
3   count: 100,
4   layout: ['prev', 'page', 'next', 'limit', 'skip', 'count', 'refresh'],
5   //limit: 5, // 每页显示的数量
6   limits: [5, 10, 20, 30], // 每页条数的选择项
7 })
8

```

## groups连续出现的页码个数

- 类型: Number, 默认值为5
- 连续出现的页面个数, 就是分页区域省略号...之前显示的页面个数

```

1 laypage.render({
2   elem: 'page',
3   count: 100,
4   layout: ['prev', 'page', 'next', 'limit', 'skip', 'count', 'refresh'],
5   limits: [5, 10, 20, 30],
6   groups: 7, // 连续显示的页码数
7 })

```

## 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)

```

1 <body>
2 <div id="page"></div>
3 <script src="../../src/layui.js" charset="utf-8"></script>
4 <script>
5   layui.use(function () {
6     var laypage = layui.laypage
7       , layer = layui.layer
8
9     laypage.render({
10      elem: 'page',
11      count: 100, //数据总数, 从服务端得到
12      groups: 10, // 连续出现的页码个数
13      layout: ['prev', 'page', 'next', 'limit', 'count'], // 自定义排版
14      limits: [5, 10, 20], // layout属性设置了limit值, 可会出现条数下拉选择框
15      jump: function (obj, first) {
16        // obj包含了当前分页的所有参数, 比如:
17        console.log(obj.curr) //得到当前页, 以便向服务端请求对应页的数据。
18        console.log(obj.limit) //得到每页显示的条数
19        //首次不执行
20        if (!first) {
21          //do something
22        }
23      },
24    })
25  })

```

## 数据表格

- 支持固定表头、固定行、固定列左/列右，支持拖拽改变列宽度，支持排序，支持多级表头，支持单元格的自定义模板，支持对表格重载（比如搜索、条件筛选等），支持复选框，支持分页，支持单元格编辑等等一些列功能。
- 模块加载名称：table

## 三种初始化渲染方式

	方式	机制	适用场景
1	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。
2	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
3	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

## 方法渲染

### 1. 引入资源

```

1 <!-- layui模块化使用 -->
2 <!-- 引入 layui.css -->
3 <link rel="stylesheet" href="../src/css/layui.css">
4 <!-- 引入 layui.js -->
5 <script src="../src/layui.js"></script>
6

```

### 2. 在页面放置一个元素，然后通过 table.render() 方法指定该容器

```

1 <body>
2 <div class="layui-container">
3   <table id="demo" lay-filter="test"></table>
4 </div>
5
6 <script>
7   layui.use(function () {
8     var table = layui.table
9     //第一个实例
10    table.render({
11      elem: '#demo' // 指定原始表格元素选择器（推荐id选择器）
12      , height: 312 // 容器高度
13      , url: 'data.json' //数据接口
14      , page: true //开启分页
15      , cols: [
16        [ //表头
17          { field: 'id', title: 'ID', width: 80, sort: true, fixed:
'left' }

```

```

18         , { field: 'username', title: '用户名', width: 80 }
19         , { field: 'sex', title: '性别', width: 80, sort: true }
20         , { field: 'city', title: '城市', width: 80 }
21         , { field: 'sign', title: '签名', width: 177 }
22         , { field: 'experience', title: '积分', width: 80, sort: true
23     }
24     , { field: 'score', title: '评分', width: 80, sort: true }
25     , { field: 'classify', title: '职业', width: 80 }
26     , { field: 'wealth', title: '财富', width: 135, sort: true },
27     ], //设置表头
28 })
29 </script>
30 </body>

```

注意：上面有一个数据接口 url，通常是从服务器获取。我们这里先本地模拟一些json数据传入

```

1  {"code":0,
2    "msg": "",
3    "count":1000,
4    "data":[
5      {"id":10000,"username":"user-0","sex":"女","city":"城市-0","sign":"签名-0","experience":255,"logins":24,"wealth":82830700,"classify":"作家","score":57},
6      {"id":10001,"username":"user-1","sex":"男","city":"城市-1","sign":"签名-1","experience":884,"logins":58,"wealth":64928690,"classify":"词人","score":27},
7      {"id":10002,"username":"user-2","sex":"女","city":"城市-2","sign":"签名-2","experience":650,"logins":77,"wealth":6298078,"classify":"酱油","score":31},
8      {"id":10003,"username":"user-3","sex":"女","city":"城市-3","sign":"签名-3","experience":362,"logins":157,"wealth":37117017,"classify":"诗人","score":68},
9      {"id":10004,"username":"user-4","sex":"男","city":"城市-4","sign":"签名-4","experience":807,"logins":51,"wealth":76263262,"classify":"作家","score":6},
10     {"id":10005,"username":"user-5","sex":"女","city":"城市-5","sign":"签名-5","experience":173,"logins":68,"wealth":60344147,"classify":"作家","score":87},
11     {"id":10006,"username":"user-6","sex":"女","city":"城市-6","sign":"签名-6","experience":982,"logins":37,"wealth":57768166,"classify":"作家","score":34},
12     {"id":10007,"username":"user-7","sex":"男","city":"城市-7","sign":"签名-7","experience":727,"logins":150,"wealth":82030578,"classify":"作家","score":28},
13     {"id":10008,"username":"user-8","sex":"男","city":"城市-8","sign":"签名-8","experience":951,"logins":133,"wealth":16503371,"classify":"词人","score":14},
14     {"id":10009,"username":"user-9","sex":"女","city":"城市-9","sign":"签名-9","experience":484,"logins":25,"wealth":86801934,"classify":"词人","score":75}
15   ]
16 }

```

- 所谓的自动渲染，即：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，而无需写初始的渲染方法。我们需要关注的是以下三点
  - 带有 class="layui-table" 的 标签
  - 对标签设置属性 lay-data="" 用于配置一些基础参数
  - 在

标签中设置属性 `lay-data=""` 用于配置表头信息。

```
1 <body>
2 <div class="layui-container">
3   <table class="layui-table" lay-data="{height:315,
4     url:'users.json', page:true, id:'test'}" lay-filter="test">
5     <thead>
6       <tr>
7         <th lay-data="{field:'id', width:80, sort:
8           true}">ID</th>
9         <th lay-data="{field:'username', width:80}">用户名
10        </th>
11        <th lay-data="{field:'sex', width:80, sort:
12          true}">性别</th>
13        <th lay-data="{field:'city'}">城市</th>
14        <th lay-data="{field:'sign'}">签名</th>
15        <th lay-data="{field:'experience', sort: true}">积
16        分</th>
17        <th lay-data="{field:'score', sort: true}">评分
18        </th>
19        <th lay-data="{field:'classify'}">职业</th>
20        <th lay-data="{field:'wealth', sort: true}">财富
21        </th>
22      </tr>
23    </thead>
24  </table>
25 </div>
26
27 <script>
28   layui.use('table', function () {
29     var table = layui.table
30   })
31 </script>
32 </body>
```

## 基础参数

- 基础参数官方文档: <http://layui-doc.pearadmin.com/doc/modules/laydate.html#options>
- 官方文档参数很多, 我们不必记忆, 只要会查会用即可

## elem绑定元素

- 类型: String/DOM
- 指定原始 table 容器的选择器或 DOM, 方法渲染方式为必填

## cols设置表头

表头参数官方文档: <https://www.layui.com/doc/modules/table.html#cols>

类型: Array

设置表头, 值是一个二维数组。表头参数设定在数组内, 表头部分参数如下:

参数	类型	说明	示例值
field	String	设定字段名。非常重要，与表格数据列要一一对应	username
title	String	设定标题名称	用户名
width	Number/String	设定列宽，若不填写，则自动分配；若填写，则支持值为：数字、百分比。	200, 30%
type	String	设定列类型。可选值有：normal（常规列，无需设定），checkbox（复选框列），radio（单选框列，layui 2.4.0 新增），numbers（序号列），space（空列）	任意一个可选值
sort	Boolean	是否允许排序（默认：false）。如果设置 true，则在对应的表头显示排序 icon，从而对列开启排序功能。	TRUE
unresize	Boolean	是否禁用拖拽列宽（默认：false）。默认情况下会根据列类型（type）来决定是否禁用，如复选框列，会自动禁用。而其它普通列，默认允许拖拽列宽，当然你也可以设置 true 来禁用该功能。	FALSE
edit	String	单元格编辑类型（默认不开启）目前只支持：text（输入框）	text

### url异步数据参数

官方文档：<https://www.layui.com/doc/modules/table.html#async>

还是一句话，不必记忆，会查会用会修改即可

### page开启分页

- 类型：Boolean/Object，开启分页(默认为 false)
- 支持传入一个对象，里面可包含 laypage 组件所有支持的参数(jump、elem除外)

```
1 <body>
2 <div class="layui-container">
3     <!-- 准备一个容器,设置id属性值 -->
4     <table id="demo"></table>
5 </div>
6 <script>
```

```

7   layui.use('table', function () {
8       var table = layui.table
9       // 加载table 实例
10      table.render({
11          elem: '#demo',          // elem属性用来绑定容器的 id 属性值
12          url: 'users.json', // 数据接口
13          cols: [
14              [
15                  // 设置序号
16                  { field: 'aa', type: 'numbers' },
17                  // 设置复选框列
18                  { field: 'bb', type: 'checkbox' },
19                  { field: 'id', title: '用户编号', sort: true, width:
120 },
20                  { field: 'username', title: '用户姓名', width: 100 },
21                  { field: 'sex', title: '性别', width: 100, sort:
true },
22                  { field: 'city', title: '城市', width: 100 },
23                  { field: 'sign', title: '签名' },
24              ],
25              // 开启分页
26              page: true,
27          })
28      })
29  </script>
30  </body>

```

## toolbar 开启表格头部工具栏

- 类型：String/DOM/Boolean，开启表格头部工具栏，该参数支持四种类型值
  - toolbar: '#toolbarDemo' 指向自定义工具栏模板选择器
  - toolbar: 'xxx' 直接传入工具栏模板字符
  - toolbar: true 仅开启工具栏，不显示左侧模板
  - toolbar: 'default' 让工具栏左侧显示默认的内置模板

```

1  <body>
2  <div class="layui-container">
3      <div id="demo"></div>
4  </div>
5
6  <!-- 表头工具栏 -->
7  <script type="text/html" id="toolbarDemo">
8      <div class="layui-btn-container">
9          <!-- lay-event 给元素绑定事件名 -->
10         <button class="layui-btn layui-btn-sm" lay-
event="getCheckData">
11             获取选中行数据
12         </button>
13         <button class="layui-btn layui-btn-sm" lay-
event="getCheckLength">
14             获取选中数目
15         </button>

```

```
16     <button class="layui-btn layui-btn-sm" lay-
event="isAll">
17         验证是否全选
18     </button>
19 </div>
20 </script>
21
22 <!-- 表格工具栏 -->
23 <script type="text/html" id="barDemo">
24     <a class="layui-btn layui-btn-xs" lay-event="edit">编辑
</a>
25     <a class="layui-btn layui-btn-danger layui-btn-xs" lay-
event="del">删除</a>
26 </script>
27
28 <script>
29     layui.use('table', function () {
30         var table = layui.table
31         table.render({
32             elem: '#demo',
33             url: 'users.json', // 数据接口
34             cols: [
35                 [
36                     // 设置序列号列
37                     { field: 'aa', type: 'numbers' },
38                     // 设置复选框列
39                     { field: 'aa', type: 'checkbox' },
40                     { field: 'id', title: '用户编号', sort: true, width:
120, hide: true },
41                     { field: 'username', title: '用户姓名', width: 100,
edit: 'text' },
42                     { field: 'sex', title: '性别', width: 100, sort:
true },
43                     { field: 'city', title: '城市', width: 100 },
44                     { field: 'sign', title: '签名', edit: 'text' },
45                     // 设置表头工具栏
46                     { field: '操作', toolbar: '#barDemo' },
47                 ],
48                 // 开启分页
49                 page: true,
50                 // 设置表格工具栏
51                 toolbar: '#toolbarDemo',
52             })
53     })
54 </script>
55 </body>
```



```

<!-- 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>

```

表格工具栏

```

// 设置表格工具栏
toolbar: "#toolbarDemo"

```

```

<!-- 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>

```

表头工具栏

```

// 设置表头工具栏
{field: "操作", toolbar: "#barDemo"}

```

	用户编号	用户姓名	性别	城市	签名	
1	10000	user-0	女	城市-0	签名-0	编辑 删除
2	10001	user-1	男	城市-1	签名-1	编辑 删除
3	10002	user-2	女	城市-2	签名-2	编辑 删除
4	10003	user-3	女	城市-3	签名-3	编辑 删除
5	10004	user-4	男	城市-4	签名-4	编辑 删除

表头数据

表格数据

表头右侧数据

## defaultToolbar 头部工具栏右侧图标

- 类型: Array, 默认值: ["filter", "exports", "print"]
- 该参数可自由配置头部工具栏右侧的图标按钮, 值为一个数组, 支持以下可选值:
  - filter: 显示筛选图标
  - exports: 显示导出图标
  - print: 显示打印图标

## 监听头工具栏事件

官方文档: <https://www.layui.com/doc/modules/table.html#on>

点击头部工具栏区域设定了属性为 lay-event="" 的元素时触发

- 语法: table.on('event(filter)', callback)
  - event 为内置事件名, filter 为容器 lay-filter 设定的值
- 回调函数返回一个 object 参数
  - obj.config 对象中可以获取 id 属性值, 即表示当前容器的 ID 属性值
  - obj.event 对象中可以获取 事件名
  - table.checkStatus(obj.config.id) 获取当前表格被选中记录对象, 返回数组

```

{event: "getCheckData", config: {}}
  config: {
    HAS_SET_COLS_PATCH: true
    autoSort: true
    cellMinWidth: 60
    checkName: "LAY_CHECKED"
    clientWidth: 1280
    cols: [Array(8)]
    defaultToolbar: (3) ["filter", "exports", "print"]
    elem: pe.fn.init [table#demo, context: document, selector: "#demo"]
    id: "demo"
    index: 1
    indexName: "LAY_TABLE_INDEX"
    limit: 10
    loading: true
  }

```

13- 数据表格.html:72

obj

obj.config

obj.config.id

obj.event 包含如下

获取选中行数据   获取选中数目   验证是否全选

	<input type="checkbox"/>	用户姓名	性别	城市	签名	
1	<input type="checkbox"/>	user-0	女	城市-0	签名-0	<button>编辑</button> <button>删除</button>
2	<input type="checkbox"/>	user-1	男	城市-1	签名-1	<button>编辑</button> <button>删除</button>
3	<input type="checkbox"/>	user-2	女	城市-2	签名-2	<button>编辑</button> <button>删除</button>
4	<input type="checkbox"/>	user-3	女	城市-3	签名-3	<button>编辑</button> <button>删除</button>
5	<input type="checkbox"/>	user-4	男	城市-4	签名-4	<button>编辑</button> <button>删除</button>

< 1 2 3 ... 5 > 到第 1 页 确定 共 50 条 10 条/页

obj.event 可以得到事件名

```

{event: "getCheckData", config: {~}}
{data: Array(0), isAll: false}
{event: "getCheckLength", config: {~}}
{data: Array(0), isAll: false}
{event: "isAll", config: {~}}
{data: Array(0), isAll: false}

```

table.checkStatus(obj.config.id) 包含如下

获取选中行数据   获取选中数目   验证是否全选

	<input checked="" type="checkbox"/>	用户姓名	性别	城市	签名	
1	<input checked="" type="checkbox"/>	user-0	女	城市-0	签名-0	<button>编辑</button> <button>删除</button>
2	<input checked="" type="checkbox"/>	user-1	男	城市-1	签名-1	<button>编辑</button> <button>删除</button>
3	<input checked="" type="checkbox"/>	user-2	女	城市-2	签名-2	<button>编辑</button> <button>删除</button>
4	<input checked="" type="checkbox"/>	user-3	女	城市-3	签名-3	<button>编辑</button> <button>删除</button>
5	<input checked="" type="checkbox"/>	user-4	男	城市-4	签名-4	<button>编辑</button> <button>删除</button>

< 1 2 3 ... 5 > 到第 1 页 确定 共 50 条 10 条/页

不论点击哪个按钮，都能得到当前勾选了几个数据

```

{data: Array(0), isAll: false}
{data: Array(2), isAll: false}
{data: Array(5), isAll: true}
{data: Array(5), isAll: true}

```

```

1 <body>
2 <div class="layui-container">
3   <div id="demo" lay-filter="test"></div>
4 </div>
5
6 <!-- 表头工具栏 -->
7 <script type="text/html" id="toolbarDemo">
8   <div class="layui-btn-container">
9     <!-- lay-event 给元素绑定事件名 -->
10    <button class="layui-btn layui-btn-sm" lay-
event="getCheckData">
11      获取选中行数据
12    </button>
13    <button class="layui-btn layui-btn-sm" lay-
event="getCheckLength">
14      获取选中数目
15    </button>
16    <button class="layui-btn layui-btn-sm" lay-
event="isAll">

```

```

17         验证是否全选
18     </button>
19 </div>
20 </script>
21
22 <!-- 表格工具栏 -->
23 <script type="text/html" id="barDemo">
24     <a class="layui-btn layui-btn-xs" lay-event="edit">编辑
25 </a>
26     <a class="layui-btn layui-btn-danger layui-btn-xs" lay-
27 event="del">删除</a>
28 </script>
29
30 <script>
31     layui.use('table', function () {
32         var table = layui.table
33         table.render({
34             elem: '#demo',
35             url: 'users.json', // 数据接口
36             cols: [
37                 [
38                     // 设置序列号列
39                     { field: 'aa', type: 'numbers' },
40                     // 设置复选框列
41                     { field: 'aa', type: 'checkbox' },
42                     { field: 'id', title: '用户编号', sort: true, width:
43 120, hide: true },
44                     { field: 'username', title: '用户姓名', width: 100,
45 edit: 'text' },
46                     { field: 'sex', title: '性别', width: 100, sort:
47 true },
48                     { field: 'city', title: '城市', width: 100 },
49                     { field: 'sign', title: '签名', edit: 'text' },
50                     // 设置表头工具栏
51                     { field: '操作', toolbar: '#barDemo' },
52                 ]
53             ],
54             // 开启分页
55             page: true,
56             // 设置表格工具栏
57             toolbar: '#toolbarDemo',
58         })
59
60         /**
61          * 头监听事件
62          * 语法:
63          *      table.on('toolbar(demo)',function(obj){
64
65              });
66          注:demo表示的是容器上设置的lay-filter属性值
67          */
68
69         table.on('toolbar(test)', function (obj) {
70             // console.log(obj);
71             // obj.config对象中可以获取id属性值，即表示当前容器的ID属性值，
72             也就是demo
73             // 获取当前表格被选中记录对象，返回数据
74             var checkStatus = table.checkStatus(obj.config.id)
75             console.log(checkStatus)

```

```

69      // 获取事件名
70      var eventName = obj.event
71      // 判断事件名，执行对应的代码
72      switch (eventName) {
73          case 'getCheckData':
74              // 获取被选中的记录的数组
75              var arr = checkStatus.data
76              // 将数组解析成字符串
77              layer.alert(JSON.stringify(arr))
78              break
79          case 'getCheckLength':
80              // 获取被选中的记录的数组
81              var arr = checkStatus.data
82              layer.msg('选中了' + arr.length + '条记录!')
83              break
84          case 'isAll':
85              // 通过isAll属性判断是否全选
86              var flag = checkStatus.isAll
87              var str = flag ? '全选' : '未全选'
88              layer.msg(str)
89              break
90          case 'LAYTABLE_TIPS':
91              layer.alert('这是一个自定义的图标按钮')
92              break
93      }
94  })
95  })
96 </script>
97 </body>

```



```

<body>
  <div id="demo" lay-filter="test"></div>
</body>

```

```

table.on('toolbar(test)', function(obj) {
  // console.log(obj);
  // obj.config对象中可以获取id属性值，即表示当前容器的ID属性值，也就是demo
  // 获取当前表格被选中记录对象，返回数据

```

## 监听行工具栏事件

官方文档: <https://www.layuion.com/doc/modules/table.html#ontool>

- 语法: table.on('tool(filter)', callback{})
- filter 为容器 lay-filter 设定的值
- 回调函数返回一个 object 参数
  - obj.data 获取当前行数据
  - obj.event 获取 lay-event 对应的值 (也可以是表头 event 参数对应的值)

obj.data 包含如下

	<input type="checkbox"/>	用户姓名	性别	城市	签名	
1	<input type="checkbox"/>	user-0	女	城市-0	签名-0	<input type="checkbox"/> 编辑 删除
2	<input type="checkbox"/>	user-1	男	城市-1	签名-1	<input type="checkbox"/> 编辑 删除
3	<input type="checkbox"/>	user-2	女	城市-2	签名-2	<input type="checkbox"/> 编辑 删除
4	<input type="checkbox"/>	user-3	女	城市-3	签名-3	<input type="checkbox"/> 编辑 删除
5	<input type="checkbox"/>	user-4	男	城市-4	签名-4	<input type="checkbox"/> 编辑 删除

obj.data

```

{id: 10000, username: "user-0", sex: "女", city: "城市-0", sign: "签名-0"}
  id: 10000
  sex: "女"
  sign: "签名-0"
  username: "user-0"
  __proto__: Object
  
```

## 监听单元格编辑

官方文档: <https://www.layui.com/doc/modules/table.html#onedit>

- 监听单元格编辑之前要先打开单元格的编辑
  - edit 类型String, 单元格编辑类型 (默认不开启) 目前只支持: text (输入框)
- 语法: `table.on('edit(filter)', callback)`
  - filter 为容器 lay-filter 设定的值
- 单元格被编辑, 且值发生改变时触发, 回调函数返回一个 object 参数
  - obj.value 获取修改后的值
  - obj.field 获取编辑的字段名
  - obj.data 获取所在行的所有相关数据

```

{tr: pe.fn.init(i), data: {}, value: "user-01", del: f, update: f, _}
  data: {id: 10000, username: "user-01", sex: "女", city: "城市-0", sign: "签名-0"}
  del: f ()
  field: "username"
  tr: pe.fn.init [tr, prevObject: pe.fn.init(i), context: undefined, selector: ".layui-table-body tr[data-index=0]"]
  update: f (e)
  value: "user-01"
  __proto__: Object
  
```

```

1 <body>
2 <div class="layui-container">
3   <div id="demo" lay-filter="test"></div>
4 </div>
5
6 <!-- 表头工具栏 -->
7 <script type="text/html" id="toolbarDemo">
8   <div class="layui-btn-container">
9     <!-- lay-event 给元素绑定事件名 -->
10    <button class="layui-btn layui-btn-sm" lay-
event="getCheckData">
11      获取选中行数据
12    </button>
13    <button class="layui-btn layui-btn-sm" lay-
event="getCheckLength">
14      获取选中数目
15    </button>
16    <button class="layui-btn layui-btn-sm" lay-
event="isAll">
17      验证是否全选
18    </button>
  
```

```

19     </div>
20 </script>
21
22 <!-- 表格工具栏 -->
23 <script type="text/html" id="barDemo">
24     <a class="layui-btn layui-btn-xs" lay-event="edit">编辑
25 </a>
26     <a class="layui-btn layui-btn-danger layui-btn-xs" lay-
27 event="del">删除</a>
28 </script>
29
30 <script>
31     layui.use('table', function () {
32         var table = layui.table
33         table.render({
34             elem: '#demo',
35             url: 'users.json', // 数据接口
36             cols: [
37                 [
38                     // 设置序列号列
39                     { field: 'aa', type: 'numbers' },
40                     // 设置复选框列
41                     { field: 'aa', type: 'checkbox' },
42                     { field: 'id', title: '用户编号', sort: true, width:
43 120, hide: true },
44                     { field: 'username', title: '用户姓名', width: 100,
45 edit: 'text' },
46                     { field: 'sex', title: '性别', width: 100, sort:
47 true },
48                     { field: 'city', title: '城市', width: 100 },
49                     { field: 'sign', title: '签名', edit: 'text' },
50                     // 设置表头工具栏
51                     { field: '操作', toolbar: '#barDemo' },
52                 ],
53                 // 开启分页
54                 page: true,
55                 // 设置表格工具栏
56                 toolbar: '#toolbarDemo',
57             })
58
59 /**
60  * 监听单元格编辑事件
61  * 表头设置edit属性， 单元格编辑类型（默认不开启）目前只支
62 持：text（输入框）
63  */
64 table.on('edit(text)', function (obj) {
65     console.log(obj)
66     // 获取修改后的值
67     var value = obj.value
68     // 得到当前修改的tr对象
69     var data = obj.data
70     // 得到修改的字段名
71     var field = obj.field
72
73     layer.msg('【ID: ' + data.id + '】的' + field + '字段的值修
74 改为: ' + value)
75 })
76 })

```

```
70 </script>
71 </body>
```

## 数据表格重载

官方文档: <https://www.layui.com/doc/modules/table.html#reload>

- 语法: `table.reload(ID,options,deep)`
  - 参数 ID 即为基础参数id对应的值
  - 参数 options 即为各项基础参数
  - 参数 deep: 是否采用深度重载 (即参数深度克隆, 也就是重载时始终携带初始时及上一次重载时的参数), 默认 false。注意: deep 参数为 layui 2.6.0 开始新增。

```
1 <body>
2 <div class="layui-container">
3   <div class="demoTable">
4     搜索ID:
5     <div class="layui-inline">
6       <input class="layui-input" name="id"
7       id="demoReload" autocomplete="off">
8     </div>
9     <button class="layui-btn" id="searchBtn">搜索</button>
10   </div>
11   <div id="demo" lay-filter="test"></div>
12 </div>
13 <!-- 表头工具栏 -->
14 <script type="text/html" id="toolbarDemo">
15   <div class="layui-btn-container">
16     <!-- lay-event 给元素绑定事件名 -->
17     <button class="layui-btn layui-btn-sm" lay-
18     event="getCheckData">
19       获取选中行数据
20     </button>
21     <button class="layui-btn layui-btn-sm" lay-
22     event="getCheckLength">
23       获取选中数目
24     </button>
25     <button class="layui-btn layui-btn-sm" lay-
26     event="isAll">
27       验证是否全选
28     </button>
29   </div>
30 </script>
31 <!-- 表格工具栏 -->
32 <script type="text/html" id="barDemo">
33   <a class="layui-btn layui-btn-xs" lay-event="edit">编辑
34 </a>
35   <a class="layui-btn layui-btn-danger layui-btn-xs" lay-
36   event="del">删除</a>
37 </script>
38 </script>
39
40 <script>
41   layui.use(['table'], function () {
```

```

37     var table = layui.table
38     var $ = layui.$
39     table.render({
40         elem: '#demo',
41         url: 'users.json', // 数据接口
42         cols: [
43             [
44                 // 设置序列号列
45                 { field: 'aa', type: 'numbers' },
46                 // 设置复选框列
47                 { field: 'aa', type: 'checkbox' },
48                 { field: 'id', title: '用户编号', sort: true, width:
120, hide: true },
49                 { field: 'username', title: '用户姓名', width: 100,
edit: 'text' },
50                 { field: 'sex', title: '性别', width: 100, sort:
true },
51                 { field: 'city', title: '城市', width: 100 },
52                 { field: 'sign', title: '签名', edit: 'text' },
53                 // 设置表头工具栏
54                 { field: '操作', toolbar: '#barDemo' },
55             ]],
56             // 开启分页
57             page: true,
58             // 设置表格工具栏
59             toolbar: '#toolbarDemo',
60         })
61
62         /**
63          * 给指定元素绑定事件
64          */
65         $(document).on('click', '#searchBtn', function (data) {
66             // 获取搜索文本框对象
67             var sreach = $('#demoReload')
68             // 调用数据表格的重载方法 table.reload(ID, options)
69             table.reload('demo', {
70                 where: { // 设置需要传递的参数
71                     id: sreach.val(),
72                     name: 'zhangsan',
73                 },
74                 page: {
75                     // 表示让条件查询从第一页开始；如果未设置则从当前页开始查询，
此页前面的所有数据不纳入条件筛选
76                     curr: 1, // 从第一页开始
77                 },
78             })
79         })
80     })
81 </script>
82 </body>

```

## 表单组件

官方文档: <https://www.layui.com/doc/modules/form.html>



## 表单元素

### 输入框

```
1      <div class="layui-form-item">
2          <label class="layui-form-label">用户名</label>
3          <div class="layui-input-block">
4              <!--输入框-->
5              <input type="text" name="" placeholder="请输入用户名" autocomplete="off" class="layui-input">
6          </div>
7      </div>
```

### 密码框

```
1      <div class="layui-form-item">
2          <label class="layui-form-label">密码</label>
3          <div class="layui-input-block">
4              <!--输入框-->
5              <input type="password" name="" placeholder="请输入密码" autocomplete="off" class="layui-input">
6          </div>
7      </div>
```

### 下拉列表【重新渲染】

lay-search="" 是否支持下拉输入过滤

```
1      <div class="layui-form-item">
2          <label class="layui-form-label">请选择性别</label>
3          <div class="layui-input-block">
4              <!--下拉列表-->
5              <select name="interest" lay-filter="gender">
6                  <option value="0">男</option>
7                  <option value="1">女</option>
8                  <option value="3">其他</option>
9              </select>
10         </div>
11     </div>
12
```

### 单选框【重新渲染】

```

1      <div class="layui-form-item">
2          <label class="layui-form-label">是否成年? </label>
3          <div class="layui-input-block">
4              <!--单选框-->
5              <input type="radio" name="sex" value="0"
title="是">
6              <input type="radio" name="sex" value="1" title="否"
checked>
7          </div>
8      </div>

```

## 复选框【重新渲染】

lay-skin="primary" 使用原始的风格

```

1      <div class="layui-form-item">
2          <label class="layui-form-label">复选框</label>
3          <div class="layui-input-block" lay-search="">
4              <!--复选框-->
5              <input type="checkbox" name="like[write]"
title="吃">
6              <input type="checkbox" name="like[read]"
title="喝">
7              <input type="checkbox" name="like[play]"
title="玩">
8              <input type="checkbox" name="like[enjoy]"
title="乐">
9          </div>
10     </div>

```

## 开关【重新渲染】

lay-skin="switch" 使用皮肤

```

1      <div class="layui-form-item">
2          <label class="layui-form-label">同意协议</label>
3          <div class="layui-input-block">
4              <!--开关-->
5              <input type="checkbox" lay-skin="switch">
6              <!--默认选中-->
7              <!--<input type="checkbox" lay-skin="switch"
checked>-->
8          </div>
9      </div>

```

## 文本域

```

1      <div class="layui-form-item layui-form-text">
2          <label class="layui-form-label">请填写描述</label>
3          <div class="layui-input-block">
4              <!--文本域-->
5              <textarea placeholder="请输入内容" class="layui-
textarea" id="lay_editor"></textarea>
6          </div>
7      </div>
8

```

## 表单对象 form

用户名:

用户密码:

用户手机:  用户邮箱:

籍贯:

性别: ☒ 男 ☐ 女 ☐ 禁用

爱好: ☐ 写作 ☒ 阅读 ☐ 游戏

是否在职: ☒ 是 ☐ 否

### 1. 监听提交事件

```

1      // 表单提交
2      form.on('submit(commit)', function (data) {
3          console.log(data)
4          console.log(data.elem) //被执行事件的元素DOM对象，一般为
button对象
5          console.log(data.form) //被执行提交的form对象，一般在存在form
标签时才会返回
6          console.log(data.field) //当前容器的全部表单字段，名值对形式：
{name: value}
7          return false //阻止表单跳转。如果需要表单跳转，去掉这段即可。
8      })

```

### 2. 监听下拉框改变事件

```

1      // 下拉选择事件
2      form.on('select(aihao)', function (data) {
3          console.log(data)
4      })

```

### 3. 监听复选框改变事件

```

1 // 多选框事件
2 form.on('checkbox(eat)', function (data) {
3     console.log(data.elem) //得到checkbox原始DOM对象
4     console.log(data.elem.checked) //是否被选中, true或者false
5     console.log(data.value) //复选框value值, 也可以通过
      data.elem.value得到
6     console.log(data.othis) //得到美化后的DOM对象
7 })

```

#### 4. 监听开关改变事件

```

1 // 开关切换
2 form.on('switch(accept)', function (data) {
3     console.log(data.elem) //得到checkbox原始DOM对象
4     console.log(data.elem.checked) //开关是否开启, true或者
      false
5     console.log(data.value) //开关value值, 也可以通过
      data.elem.value得到
6     console.log(data.othis) //得到美化后的DOM对象
7 })
8

```

#### 5. 监听单选框选中事件

```

1 // 单选框事件
2 form.on('radio(gender)', function (data) {
3     console.log(data.elem) //得到radio原始DOM对象
4     console.log(data.value) //被点击的radio的value值
5 })
6

```

#### 7. 表单初始赋值

```

1 form.val('info', {
2     username: '张大明', // "name": "value"
3     gender: '1',
4     sex: '1',
5     message: 'hello world !',
6     'like[read]': true,
7     accept: true,
8 })

```

### 表单验证

#### 使用方法

lay-verify="required|phone"

#### 相关的值

required 非空验证

phone 手机号验证

number 数值验证

url 链接地址验证 <http://www.baid.com>

## 自定义验证

```
1 // 自定义表单验证
2 form.verify({
3   username: function (value, item) { //value: 表单的值、
    item: 表单的DOM对象
4     if (!new RegExp('^[a-zA-Z0-9_\u4e00-\u9fa5\\s·]+$').test(value)) {
5       return '用户名不能有特殊字符'
6     }
7     if (/^(^_|_|)(_|_|)(_|_|)$/.test(value)) {
8       return '用户名首尾不能出现下划线\'_\'\'
9     }
10    if (/^\d+\d+\d$/.test(value)) {
11      return '用户名不能全为数字'
12    }
13
14    //如果不想自动弹出默认提示框，可以直接返回 true，这时你可以通过其他任意方式提示（v2.5.7 新增）
15    if (value === 'xxx') {
16      alert('用户名不能为敏感词')
17      return true
18    }
19  }
20
21  //我们既支持上述函数式的方式，也支持下述数组的形式
22  //数组的两个值分别代表：[正则匹配、匹配不符时的提示文字]
23  , pass: [
24    /^[s]{6,12}$/
25    , '密码必须6到12位，且不能出现空格',
26  ],
27 })
```

## 数据表格+弹出层的综合案例

### json准备

### 创建页面

查询条件

编号:

用户名:

邮箱:

开始时间:

结束时间:

性别: ☐ 男 ☐ 女

立即查询

重置

增加

批量删除

ID

用户名

邮箱

性别

城市

操作

10001

杜南

xianxin@layui.com

女

浙江杭州

编辑

删除

10002

李白

xianxin@layui.com

女

浙江杭州

编辑

删除

10003

王勃

xianxin@layui.com

女

浙江杭州

编辑

删除

10004

李清照

xianxin@layui.com

女

浙江杭州

编辑

删除

10005

冰心

xianxin@layui.com

女

浙江杭州

编辑

删除

10006

谭心

xianxin@layui.com

女

浙江杭州

编辑

删除

1

2

3

...

11

>

到第

1

页

确定

共 101 条

10 条/页

查询条件

编号:

用户名:

邮箱:

开始时间:

性别: ☒ 男 ☐ 女

立即查询

重置

添加用户

编号:

用户名:

邮箱:

城市:

性别: ☒ 男 ☐ 女

立即查询

重置

ID

用户名

邮箱

性别

城市

操作

10001

杜南

xianxin@layui.com

女

浙江杭州

编辑

删除

10002

李白

xianxin@layui.com

女

浙江杭州

编辑

删除

10003

王勃

xianxin@layui.com

女

浙江杭州

编辑

删除

10004

李清照

xianxin@layui.com

女

浙江杭州

编辑

删除

10005

冰心

xianxin@layui.com

女

浙江杭州

编辑

删除

10006

谭心

xianxin@layui.com

女

浙江杭州

编辑

删除

1

2

3

...

11

>

到第

1

页

确定

共 101 条

10 条/页

table 数据表格文档 - layui

数据表格

李伟超8 - 页面元素 - layui

JSON在线解析及生成工具 - 1

127.0.0.1:8080/bjsoft/22\_table.html

百度

Maven-阿里

JSON.cn

mybatis

docker

前端框架

shimo

首页

redis

查询条件

编号:

用户名:

邮箱:

开始时间:

性别: ☒ 男 ☐ 女

立即查询

重置

修改用户

编号: 10001

用户名: 杜南

邮箱: xianxin@layui.com

城市: 浙江杭州

性别: ☒ 男 ☐ 女

立即查询

重置

ID

用户名

邮箱

性别

城市

操作

10001

杜南

xianxin@layui.com

女

浙江杭州

编辑

删除

10002

李白

xianxin@layui.com

女

浙江杭州

编辑

删除

10003

王勃

xianxin@layui.com

女

浙江杭州

编辑

删除

10004

李清照

xianxin@layui.com

女

浙江杭州

编辑

删除

10005

冰心

xianxin@layui.com

女

浙江杭州

编辑

删除

10006

谭心

xianxin@layui.com

女

浙江杭州

编辑

删除

1

2

3

...

11

>

到第

1

页

确定

共 101 条

10 条/页

富文本

```

4      <label class="layui-form-label">编辑器</label>
5      <div class="layui-input-block">
6          <textarea class="layui-textarea layui-hide" name="content" lay-verify="content" id="LAY_demo_editor"></te
7      </div>
8  </div>
9
10 </form>
11
12
13 <script src="resources/layui/layui.js"></script>
14 <script type="text/javascript">
15     layui.use(['element', 'jquery', 'laydate', 'form', 'layedit'], function() {
16         var $ = layui.jquery;
17         var element = layui.element;
18         var laydate = layui.laydate;
19         var form = layui.form;
20         var layedit = layui.layedit;
21         // 创建一个编辑器
22         var editIndex = layedit.build('LAY_demo_editor');
23     });
24 </script>

```

用户名:

用户密码:

验证手机:

验证邮箱:

单行选择框:

籍贯:









性别: ☒ 男 ☐ 女 ☐ 禁用

复选框: ☐ 写作 ☒ 阅读 ☐ 游戏

原始复选框: ☒ 写作 ☐ 阅读 ☐ 游戏

开关-默认关: ☐

普通文本域:

编辑器: **B** *I* U        

## 文件上传

### 选择文件之后自动上传

```

1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Document</title>
6      <link rel="stylesheet" href="../static/css/layui.css">
7  </head>
8  <body>
9
10 <div class="layui-upload">
11     <button type="button" class="layui-btn" id="test1">上传图片
12 </button>
13     <div class="layui-upload-list">
14         <img class="layui-upload-img" width="60" height="60"
15         id="demo1">

```

```

14     <p id="demoText"></p>
15   </div>
16 </div>
17 <script src="../../static/layui.js"></script>
18 <script>
19   layui.use(function () {
20     var upload = layui.upload
21     var $ = layui.$
22     //普通图片上传
23     var uploadInst = upload.render({
24       elem: '#test1'
25       , url: 'file/upload'
26       , accept: 'images'
27       , acceptMime: 'image/'
28       , auto: true //是否选择文件之后自动上传
29       , field: 'mf' //表单的name值
30       , before: function (obj) {
31         //预读本地文件示例，不支持ie8
32         obj.preview(function (index, file, result) {
33           $('#demo1').attr('src', result) //图片链接（base64）
34         })
35       }, done: function (res) {
36         alert(res)
37         //如果上传失败
38         if (res.code > 0) {
39           return layer.msg('上传失败')
40         }
41         //上传成功
42         layer.msg('上传成功')
43       }, error: function () { //演示失败状态，并实现
重传
44         var demoText = $('#demoText')
45         demoText.html('<span style="color: #FF5722;">上传失败
</span> <a class="layui-btn layui-btn-xs demo-reload">重试
</a>`)
46         demoText.find('.demo-reload').on('click', function ()
{ uploadInst.upload() })
47       },
48     })
49   })
50 </script>
51 </body>
52 </html>

```

## 面板

### 卡片面板

#### 相关样式

layui-row 代表一行

layui-col-space15 space0-space30 代表卡片之间的间距



|-- layui-col-md6 代表一列 md1-md12 代表当前卡片占整行的X/12

|-- layui-card 代表一个卡片

|-- layui-card-header 代表卡片头信息

|-- layui-card-body 代表卡片内容样式

## 普通折叠面板

### 相关样式

layui-collapse 代表一个折叠面板 lay-accordion 开启手风琴

|-- layui-colla-item 代表一个折叠项

|-- layui-colla-title 代表折叠项的显示标题

|-- layui-colla-content 代表折叠项展开的内容

|-- layui-show 是否展开

### 相关事件

<https://www.layui.com/doc/modules/element.html#collapse>

```
1 <script type="text/javascript">
2   layui.use(function () {
3     let element = layui.element
4     let $ = layui.$
5
6     element.on('collapse(test)', function (data) {
7       console.log(data.show) // 得到当前面板的展开状态, true 或者
8       false
9       console.log(data.title) // 得到当前点击面板的标题区域 DOM 对
10      象
11      console.log(data.content) // 得到当前点击面板的内容区域 DOM
12    })
13  })
14 </script>
```

## 手风琴面板

在普通面板上加lay-accordion=""

## 时间线

### 相关样式

layui-timeline 代表一个时间线

```
|-- layui-timeline-item 代表一个时间节点
|-- layui-timeline-axis 代表左边的竖线
    |-- layui-timeline-content 时间线的内容
    |-- layui-text 代表文本
        |-- layui-timeline-title 代表标题
            |--可以分为标题和内容
            |--
```

标题

|--

内容

---