

Element-UI 使用手册 文档 V2.4.6 (Vue版本)

书栈(BookStack.CN)

目 录

致谢

开发指南

安装

快速上手

国际化

自定义主题

内置过渡动画

Basic 基本组件

Layout 布局

Container 布局容器

Color 色彩

Typography 字体

Icon 图标

Button 按钮

Form 表单组件

Radio 单选框

Checkbox 多选框

Input 输入框

InputNumber 计数器

Select 选择器

Cascader 级联选择器

Switch 开关

Slider 滑块

TimePicker 时间选择器

DatePicker 日期选择器

DateTimePicker 日期时间选择器

Upload 上传

Rate 评分

ColorPicker 颜色选择器

Transfer 穿梭框

Form 表单

Data 数据组件

Table 表格

Tag 标签

Progress 进度条

Tree 树形控件

[Pagination 分页](#)

[Badge 标记](#)

[Notice 通知组件](#)

[Alert 警告](#)

[Loading 加载](#)

[Message 消息提示](#)

[MessageBox 弹框](#)

[Notification 通知](#)

[Navigation 导航组件](#)

[NavMenu 导航菜单](#)

[Tabs 标签页](#)

[Breadcrumb 面包屑](#)

[Dropdown 下拉菜单](#)

[Steps 步骤条](#)

[Others 其他组件](#)

[Dialog 对话框](#)

[Tooltip 文字提示](#)

[Popover 弹出框](#)

[Card 卡片](#)

[Carousel 走马灯](#)

[Collapse 折叠面板](#)

致谢

当前文档 《Element-UI 使用手册文档 V2.4.6 (Vue版本)》 由 进击的皇虫 使用 书栈 (BookStack.CN) 进行构建，生成于 2018-08-14。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常工作、生活和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈 (BookStack.CN)，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

文档地址：<http://www.bookstack.cn/books/Element-UI-vue>

书栈官网：<http://www.bookstack.cn>

书栈开源：<https://github.com/TruthHun>

分享，让知识传承更久远！ 感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

开发指南

- 安装
- 快速上手
- 国际化
- 自定义主题
- 内置过渡动画

安装

npm 安装

推荐使用 npm 的方式安装，它能更好地和 webpack 打包工具配合使用。

```
1. npm i element-ui -S
```

CDN

目前可以通过 unpkg.com/element-ui 获取到最新版本的资源，在页面上引入 js 和 css 文件即可开始使用。

```
1. <!-- 引入样式 -->
2. <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
3. <!-- 引入组件库 -->
4. <script src="https://unpkg.com/element-ui/lib/index.js"></script>
```

我们建议使用 CDN 引入 Element 的用户在链接地址上锁定版本，以免将来 Element 升级时受到非兼容性更新的影响。锁定版本的方法请查看 unpkg.com。

Hello world

通过 CDN 的方式我们可以很容易地使用 Element 写出一个 Hello world 页面。[在线演示](#)

如果是通过 npm 安装，并希望配合 webpack 使用，请阅读下一节：[快速上手](#)。

原文：<http://element-cn.eleme.io/#/zh-CN/component/installation>

快速上手

快速上手

本节将介绍如何在项目中使用 Element。

使用 vue-cli@3

我们为新版的 vue-cli 准备了相应的 Element 插件，你可以用它们快速地搭建一个基于 Element 的项目。

使用 Starter Kit

我们提供了通用的[项目模板](#)，你可以直接使用。对于 Laravel 用户，我们也准备了相应的[模板](#)，同样可以直接下载使用。

如果不希望使用我们提供的模板，请继续阅读。

引入 Element

你可以引入整个 Element，或是根据需要仅引入部分组件。我们先介绍如何引入完整的 Element。

完整引入

在 main.js 中写入以下内容：

```
1. import Vue from 'vue';
2. import ElementUI from 'element-ui';
3. import 'element-ui/lib/theme-chalk/index.css';
4. import App from './App.vue';
5.
6. Vue.use(ElementUI);
7.
8. new Vue({
9.   el: '#app',
10.  render: h => h(App)
11. });


```

以上代码便完成了 Element 的引入。需要注意的是，样式文件需要单独引入。

按需引入

借助 [babel-plugin-component](#)，我们可以只引入需要的组件，以达到减小项目体积的目的。

首先，安装 babel-plugin-component：

```
1. npm install babel-plugin-component -D
```

然后，将 .babelrc 修改为：

```
1. {
2.   "presets": [["es2015", { "modules": false }]],
3.   "plugins": [
4.     [
5.       "component",
6.       {
7.         "libraryName": "element-ui",
8.         "styleLibraryName": "theme-chalk"
9.       }
10.    ]
11.  ]
12. }
```

接下来，如果你只希望引入部分组件，比如 Button 和 Select，那么需要在 main.js 中写入以下内容：

```
1. import Vue from 'vue';
2. import { Button, Select } from 'element-ui';
3. import App from './App.vue';
4.
5. Vue.component(Button.name, Button);
6. Vue.component(Select.name, Select);
7. /* 或写为
8.  * Vue.use(Button)
9.  * Vue.use(Select)
10. */
11.
12. new Vue({
13.   el: '#app',
14.   render: h => h(App)
15. });
```

完整组件列表和引入方式（完整组件列表以 [components.json](#) 为准）

```
1. import Vue from 'vue';
2. import {
3.   Pagination,
4.   Dialog,
5.   Autocomplete,
6.   Dropdown,
7.   DropdownMenu,
8.   DropdownItem,
9.   Menu,
10.  Submenu,
11.  MenuItem,
12.  MenuItemGroup,
13.  Input,
14.  InputNumber,
15.  Radio,
16.  RadioGroup,
17.  RadioButton,
18.  Checkbox,
19.  CheckboxButton,
20.  CheckboxGroup,
21.  Switch,
22.  Select,
23.  Option,
24.  OptionGroup,
25.  Button,
26.  ButtonGroup,
27.  Table,
28.  TableColumn,
29.  DatePicker,
30.  TimeSelect,
31.  TimePicker,
32.  Popover,
33.  Tooltip,
34.  Breadcrumb,
35.  BreadcrumbItem,
36.  Form,
37.  FormItem,
38.  Tabs,
39.  TabPane,
40.  Tag,
41.  Tree,
42.  Alert,
```

```
43.   Slider,
44.   Icon,
45.   Row,
46.   Col,
47.   Upload,
48.   Progress,
49.   Badge,
50.   Card,
51.   Rate,
52.   Steps,
53.   Step,
54.   Carousel,
55.   CarouselItem,
56.   Collapse,
57.   CollapseItem,
58.   Cascader,
59.   ColorPicker,
60.   Transfer,
61.   Container,
62.   Header,
63.   Aside,
64.   Main,
65.   Footer,
66.   Loading,
67.   MessageBox,
68.   Message,
69.   Notification
70. } from 'element-ui';
71.
72. Vue.use(Pagination);
73. Vue.use(Dialog);
74. Vue.use(Autocomplete);
75. Vue.use(Dropdown);
76. Vue.use(DropdownMenu);
77. Vue.use(DropdownItem);
78. Vue.use(Menu);
79. Vue.use(Submenu);
80. Vue.use(MenuItem);
81. Vue.use(MenuItemGroup);
82. Vue.use(Input);
83. Vue.use(InputNumber);
84. Vue.use(Radio);
```

```
85. Vue.use(RadioGroup);
86. Vue.use(RadioButton);
87. Vue.use(Checkbox);
88. Vue.use(CheckboxButton);
89. Vue.use(CheckboxGroup);
90. Vue.use(Switch);
91. Vue.use(Select);
92. Vue.use(Option);
93. Vue.use(OptionGroup);
94. Vue.use(Button);
95. Vue.use(ButtonGroup);
96. Vue.use(Table);
97. Vue.use( TableColumn );
98. Vue.use( DatePicker );
99. Vue.use( TimeSelect );
100. Vue.use( TimePicker );
101. Vue.use( Popover );
102. Vue.use( Tooltip );
103. Vue.use( Breadcrumb );
104. Vue.use( BreadcrumbItem );
105. Vue.use( Form );
106. Vue.use( FormItem );
107. Vue.use( Tabs );
108. Vue.use( TabPane );
109. Vue.use( Tag );
110. Vue.use( Tree );
111. Vue.use( Alert );
112. Vue.use( Slider );
113. Vue.use( Icon );
114. Vue.use( Row );
115. Vue.use( Col );
116. Vue.use( Upload );
117. Vue.use( Progress );
118. Vue.use( Badge );
119. Vue.use( Card );
120. Vue.use( Rate );
121. Vue.use( Steps );
122. Vue.use( Step );
123. Vue.use( Carousel );
124. Vue.use( CarouselItem );
125. Vue.use( Collapse );
126. Vue.use( CollapseItem );
```

```

127. Vue.use(Cascader);
128. Vue.use(ColorPicker);
129. Vue.use(Container);
130. Vue.use(Header);
131. Vue.use(Aside);
132. Vue.use(Main);
133. Vue.use(Footer);
134.
135. Vue.use(Loading.directive);
136.
137. Vue.prototype.$loading = Loading.service;
138. Vue.prototype.$msgbox = MessageBox;
139. Vue.prototype.$alert = MessageBox.alert;
140. Vue.prototype.$confirm = MessageBox.confirm;
141. Vue.prototype.$prompt = MessageBox.prompt;
142. Vue.prototype.$notify = Notification;
143. Vue.prototype.$message = Message;

```

全局配置

在引入 Element 时，可以传入一个全局配置对象。该对象目前支持 `size` 与 `zIndex` 字段。`size` 用于改变组件的默认尺寸，`zIndex` 设置弹框的初始 z-index (默认值：2000)。按照引入 Element 的方式，具体操作如下：

完整引入 Element：

```

1. import Vue from 'vue';
2. import Element from 'element-ui';
3. Vue.use(Element, { size: 'small', zIndex: 3000 });

```

按需引入 Element：

```

1. import Vue from 'vue';
2. import { Button } from 'element-ui';
3.
4. Vue.prototype.$ELEMENT = { size: 'small', zIndex: 3000 };
5. Vue.use(Button);

```

按照以上设置，项目中所有拥有 `size` 属性的组件的默认尺寸均为 'small'，弹框的初始 z-index 为 3000。

开始使用

至此，一个基于 Vue 和 Element 的开发环境已经搭建完毕，现在就可以编写代码了。各个组件的使用方法请参阅它们各自的文档。

使用 Nuxt.js

我们还可以使用 [Nuxt.js](#)：

原文：<http://element-cn.eleme.io/#/zh-CN/component/quickstart>

国际化

Element 组件内部默认使用中文，若希望使用其他语言，则需要进行多语言设置。以英文为例，在 main.js 中：

```
1. // 完整引入 Element
2. import Vue from 'vue'
3. import ElementUI from 'element-ui'
4. import locale from 'element-ui/lib/locale/lang/en'
5.
6. Vue.use(ElementUI, { locale })
```

或

```
1. // 按需引入 Element
2. import Vue from 'vue'
3. import { Button, Select } from 'element-ui'
4. import lang from 'element-ui/lib/locale/lang/en'
5. import locale from 'element-ui/lib/locale'
6.
7. // 设置语言
8. locale.use(lang)
9.
10. // 引入组件
11. Vue.component(Button.name, Button)
12. Vue.component(Select.name, Select)
```

如果使用其它语言，默认情况下中文语言包依旧是被引入的，可以使用 webpack 的 NormalModuleReplacementPlugin 替换默认语言包。

webpack.config.js

```
1. {
2.   plugins: [
3.     new webpack.NormalModuleReplacementPlugin(/element-
4.       ui[\.\\\]lib[\.\\\]locale[\.\\\]lang[\.\\\]zh-CN/, 'element-ui/lib/locale/lang/en')
5.   ]
6. }
```

兼容 vue-i18n@5.x

Element 兼容 `vue-i18n@5.x`，搭配使用能更方便地实现多语言切换。

```

1. import Vue from 'vue'
2. import VueI18n from 'vue-i18n'
3. import Element from 'element-ui'
4. import enLocale from 'element-ui/lib/locale/lang/en'
5. import zhLocale from 'element-ui/lib/locale/lang/zh-CN'
6.
7. Vue.use(VueI18n)
8. Vue.use(Element)
9.
10. Vue.config.lang = 'zh-cn'
11. Vue.locale('zh-cn', zhLocale)
12. Vue.locale('en', enLocale)

```

兼容其他 i18n 插件

如果不使用 `vue-i18n@5.x`，而是用其他的 i18n 插件，Element 将无法兼容，但是可以自定义 Element 的 i18n 的处理方法。

```

1. import Vue from 'vue'
2. import Element from 'element-ui'
3. import enLocale from 'element-ui/lib/locale/lang/en'
4. import zhLocale from 'element-ui/lib/locale/lang/zh-CN'
5.
6. Vue.use(Element, {
7.   i18n: function (path, options) {
8.     // ...
9.   }
10. })

```

兼容 vue-i18n@6.x

默认不支持 6.x 的 `vue-i18n`，你需要手动处理。

```
1. import Vue from 'vue'
```

```

2. import Element from 'element-ui'
3. import VueI18n from 'vue-i18n'
4. import enLocale from 'element-ui/lib/locale/lang/en'
5. import zhLocale from 'element-ui/lib/locale/lang/zh-CN'
6.
7. Vue.use(VueI18n)
8.
9. const messages = {
10.   en: {
11.     message: 'hello',
12.     ...enLocale // 或者用 Object.assign({ message: 'hello' }, enLocale)
13.   },
14.   zh: {
15.     message: '你好',
16.     ...zhLocale // 或者用 Object.assign({ message: '你好' }, zhLocale)
17.   }
18. }
19. // Create VueI18n instance with options
20. const i18n = new VueI18n({
21.   locale: 'en', // set locale
22.   messages, // set locale messages
23. })
24.
25. Vue.use(Element, {
26.   i18n: (key, value) => i18n.t(key, value)
27. })
28.
29. new Vue({ i18n }).$mount('#app')

```

按需加载里定制 i18n

```

1. import Vue from 'vue'
2. import DatePicker from 'element/lib/date-picker'
3. import VueI18n from 'vue-i18n'
4.
5. import enLocale from 'element-ui/lib/locale/lang/en'
6. import zhLocale from 'element-ui/lib/locale/lang/zh-CN'
7. import ElementLocale from 'element-ui/lib/locale'
8.
9. Vue.use(VueI18n)
10. Vue.use(DatePicker)

```

```

11.
12. const messages = {
13.   en: {
14.     message: 'hello',
15.     ...enLocale
16.   },
17.   zh: {
18.     message: '你好',
19.     ...zhLocale
20.   }
21. }
22. // Create VueI18n instance with options
23. const i18n = new VueI18n({
24.   locale: 'en', // set locale
25.   messages, // set locale messages
26. })
27.
28. ElementLocale.i18n((key, value) => i18n.t(key, value))

```

通过 CDN 的方式加载语言文件

```

1. <script src="//unpkg.com/vue"></script>
2. <script src="//unpkg.com/element-ui"></script>
3. <script src="//unpkg.com/element-ui/lib/umd/locale/en.js"></script>
4.
5. <script>
6.   ELEMENT.locale(ELEMENT.lang.en)
7. </script>

```

搭配 `vue-i18n` 使用

```

1. <script src="//unpkg.com/vue"></script>
2. <script src="//unpkg.com/vue-i18n/dist/vue-i18n.js"></script>
3. <script src="//unpkg.com/element-ui"></script>
4. <script src="//unpkg.com/element-ui/lib/umd/locale/zh-CN.js"></script>
5. <script src="//unpkg.com/element-ui/lib/umd/locale/en.js"></script>
6.
7. <script>
8.   Vue.locale('en', ELEMENT.lang.en)
9.   Vue.locale('zh-cn', ELEMENT.lang.zhCN)
10. </script>

```

目前 Element 内置了以下语言：

- 简体中文 (zh-CN)
- 英语 (en)
- 德语 (de)
- 葡萄牙语 (pt)
- 西班牙语 (es)
- 丹麦语 (da)
- 法语 (fr)
- 挪威语 (nb-NO)
- 繁体中文 (zh-TW)
- 意大利语 (it)
- 韩语 (ko)
- 日语 (ja)
- 荷兰语 (nl)
- 越南语 (vi)
- 俄语 (ru-RU)
- 土耳其语 (tr-TR)
- 巴西葡萄牙语 (pt-br)
- 波斯语 (fa)
- 泰语 (th)
- 印尼语 (id)
- 保加利亚语 (bg)
- 波兰语 (pl)
- 芬兰语 (fi)
- 瑞典语 (sv-SE)
- 希腊语 (el)
- 斯洛伐克语 (sk)
- 加泰罗尼亚语 (ca)
- 捷克语 (cs-CZ)
- 乌克兰语 (ua)
- 土库曼语 (tk)
- 泰米尔语 (ta)
- 拉脱维亚语 (lv)
- 南非荷兰语 (af-ZA)
- 爱沙尼亚语 (ee)
- 斯洛文尼亚语 (sl)
- 阿拉伯语 (ar)
- 希伯来语 (he)

- 立陶宛语 (lt)
- 蒙古语 (mn)
- 哈萨克斯坦语 (kz)
- 匈牙利语 (hu)
- 罗马尼亚语 (ro)
- 库尔德语 (ku)
- 维吾尔语 (ug-CN)
- 高棉语 (km)

如果你需要使用其他的语言，欢迎贡献 PR：只需在 [这里](#) 添加一个语言配置文件即可。

原文：<http://element-cn.eleme.io/#/zh-CN/component/i18n>

自定义主题

自定义主题

Element 默认提供一套主题，CSS 命名采用 BEM 的风格，方便使用者覆盖样式。我们提供了三种方法，可以进行不同程度的样式自定义。

仅替换主题色

如果仅希望更换 Element 的主题色，推荐使用[在线主题生成工具](#)。Element 默认的主题色是鲜艳、友好的蓝色。通过替换主题色，能够让 Element 的视觉更加符合具体项目的定位。

使用上述工具，可以很方便地实时预览主题色改变之后的视觉，同时它还可以基于新的主题色生成完整的样式文件包，供直接下载使用（关于如何使用下载的主题包，请参考本节「引入自定义主题」和「搭配插件按需引入组件主题」部分）。

在项目中改变 SCSS 变量

Element 的 theme-chalk 使用 SCSS 编写，如果你的项目也使用了 SCSS，那么可以直接在项目中改变 Element 的样式变量。新建一个样式文件，例如 `element-variables.scss`，写入以下内容：

```
1. /* 改变主题色变量 */
2. $--color-primary: teal;
3.
4. /* 改变 icon 字体路径变量，必需 */
5. $--font-path: '~element-ui/lib/theme-chalk/fonts';
6.
7. @import "~element-ui/packages/theme-chalk/src/index";
```

之后，在项目的入口文件中，直接引入以上样式文件即可（无需引入 Element 编译好的 CSS 文件）：

```
1. import Vue from 'vue'
2. import Element from 'element-ui'
3. import './element-variables.scss'
4.
5. Vue.use(Element)
```

需要注意的是，覆盖字体路径变量是必需的，将其赋值为 Element 中 icon 图标所在的相对路径即

可。

命令行主题工具

如果你的项目没有使用 SCSS，那么可以使用命令行主题工具进行深层次的主题定制：

安装工具

首先安装「主题生成工具」，可以全局安装或者安装在当前项目下，推荐安装在项目里，方便别人 clone 项目时能直接安装依赖并启动，这里以全局安装做演示。

```
1. npm i element-theme -g
```

安装白垩主题，可以从 npm 安装或者从 GitHub 拉取最新代码。

```
1. # 从 npm
2. npm i element-theme-chalk -D
3.
4. # 从 GitHub
5. npm i https://github.com/ElementUI/theme-chalk -D
```

初始化变量文件

主题生成工具安装成功后，如果全局安装可以在命令行里通过 `et` 调用工具，如果安装在当前目录下，需要通过 `node_modules/.bin/et` 访问到命令。执行 `-i` 初始化变量文件。默认输出到 `element-variables.scss`，当然你可以传参数指定文件输出目录。

```
1. et -i [可以自定义变量文件]
2.
3. > ✓ Generator variables file
```

如果使用默认配置，执行后当前目录会有一个 `element-variables.scss` 文件。内部包含了主题所用到的所有变量，它们使用 SCSS 的格式定义。大致结构如下：

```
1. $--color-primary: #409EFF !default;
2. $--color-primary-light-1: mix($--color-white, $--color-primary, 10%) !default;
   /* 53a8ff */
3. $--color-primary-light-2: mix($--color-white, $--color-primary, 20%) !default;
   /* 66b1ff */
4. $--color-primary-light-3: mix($--color-white, $--color-primary, 30%) !default;
   /* 79bbff */
5. $--color-primary-light-4: mix($--color-white, $--color-primary, 40%) !default;
```

```

/* 8cc5ff */
6. $--color-primary-light-5: mix($--color-white, $--color-primary, 50%) !default;
/* a0cffff */
7. $--color-primary-light-6: mix($--color-white, $--color-primary, 60%) !default;
/* b3d8fff */
8. $--color-primary-light-7: mix($--color-white, $--color-primary, 70%) !default;
/* c6e2fff */
9. $--color-primary-light-8: mix($--color-white, $--color-primary, 80%) !default;
/* d9ecfff */
10. $--color-primary-light-9: mix($--color-white, $--color-primary, 90%) !default;
/* ecf5fff */
11.
12. $--color-success: #67c23a !default;
13. $--color-warning: #e6a23c !default;
14. $--color-danger: #f56c6c !default;
15. $--color-info: #909399 !default;
16.
17. ...

```

修改变量

直接编辑 `element-variables.scss` 文件，例如修改主题色为红色。

```
1. $--color-primary: red;
```

编译主题

保存文件后，到命令行里执行 `et` 编译主题，如果你想启用 `watch` 模式，实时编译主题，增加 `-w` 参数；如果你在初始化时指定了自定义变量文件，则需要增加 `-c` 参数，并带上你的变量文件名

```
1. et
2.
3. > ✓ build theme font
4. > ✓ build element theme
```

引入自定义主题

默认情况下编译的主目录是放在 `./theme` 下，你可以通过 `-o` 参数指定打包目录。像引入默认主题一样，在代码里直接引用 `theme/index.css` 文件即可。

```
1. import '../theme/index.css'
```

```
2. import ElementUI from 'element-ui'  
3. import Vue from 'vue'  
4.  
5. Vue.use(ElementUI)
```

搭配插件按需引入组件主题

如果是搭配 `babel-plugin-component` 一起使用，只需要修改 `.babelrc` 的配置，指定 `styleLibraryName` 路径为自定义主题相对于 `.babelrc` 的路径，注意要加 `~`。

```
1. {  
2.   "plugins": [[{"component": [  
3.     {  
4.       "libraryName": "element-ui",  
5.       "styleLibraryName": "~theme"  
6.     }  
7.   ]]  
8. }
```

如果不清楚 `babel-plugin-component` 是什么，请阅读 [快速上手](#) 一节。更多 `element-theme` 用法请参考[项目仓库](#)。

原文：<http://element-cn.eleme.io/#/zh-CN/component/custom-theme>

内置过渡动画

内置过渡动画

Element 内应用在部分组件的过渡动画，你也可以直接使用。在使用之前请阅读 [transition 组件文档](#)。

fade 淡入淡出



提供 `el-fade-in-linear` 和 `el-fade-in` 两种效果。

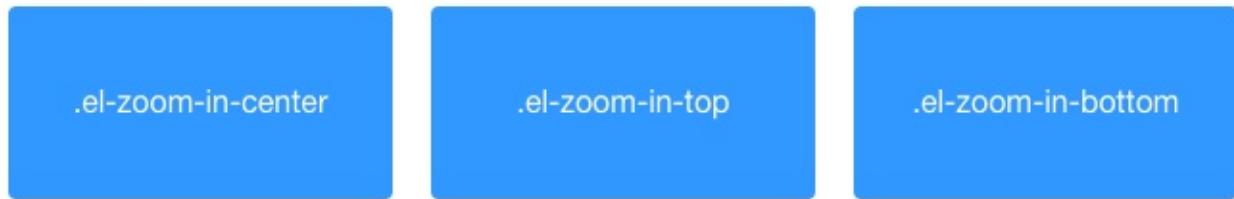
```
1. <template>
2.   <div>
3.     <el-button @click="show = !show">Click Me</el-button>
4.
5.     <div style="display: flex; margin-top: 20px; height: 100px;">
6.       <transition name="el-fade-in-linear">
7.         <div v-show="show" class="transition-box">.el-fade-in-linear</div>
8.       </transition>
9.       <transition name="el-fade-in">
10.         <div v-show="show" class="transition-box">.el-fade-in</div>
11.       </transition>
12.     </div>
13.   </div>
14. </template>
15.
16. <script>
17.   export default {
18.     data: () => ({
19.       show: true
20.     })
21.   }
22. </script>
23.
```

```

24. <style>
25.   .transition-box {
26.     margin-bottom: 10px;
27.     width: 200px;
28.     height: 100px;
29.     border-radius: 4px;
30.     background-color: #409EFF;
31.     text-align: center;
32.     color: #fff;
33.     padding: 40px 20px;
34.     box-sizing: border-box;
35.     margin-right: 20px;
36.   }
37. </style>

```

ZOOM 缩放



提供 `el-zoom-in-center` , `el-zoom-in-top` 和 `el-zoom-in-bottom` 三种效果。

```

1. <template>
2. <div>
3.   <el-button @click="show2 = !show2">Click Me</el-button>
4.
5.   <div style="display: flex; margin-top: 20px; height: 100px;">
6.     <transition name="el-zoom-in-center">
7.       <div v-show="show2" class="transition-box">.el-zoom-in-center</div>
8.     </transition>
9.
10.    <transition name="el-zoom-in-top">
11.      <div v-show="show2" class="transition-box">.el-zoom-in-top</div>
12.    </transition>
13.
14.    <transition name="el-zoom-in-bottom">
15.      <div v-show="show2" class="transition-box">.el-zoom-in-bottom</div>
16.    </transition>
17.  </div>

```

```
18.  </div>
19. </template>
20.
21. <script>
22.   export default {
23.     data: () => ({
24.       show2: true
25.     })
26.   }
27. </script>
28.
29. <style>
30.   .transition-box {
31.     margin-bottom: 10px;
32.     width: 200px;
33.     height: 100px;
34.     border-radius: 4px;
35.     background-color: #409EFF;
36.     text-align: center;
37.     color: #fff;
38.     padding: 40px 20px;
39.     box-sizing: border-box;
40.     margin-right: 20px;
41.   }
42. </style>
```

collapse 展开折叠

使用 `el-collapse-transition` 组件实现折叠展开效果。



```
1. <template>
```

```

2.  <div>
3.      <el-button @click="show3 = !show3">Click Me</el-button>
4.
5.      <div style="margin-top: 20px; height: 200px;">
6.          <el-collapse-transition>
7.              <div v-show="show3">
8.                  <div class="transition-box">el-collapse-transition</div>
9.                  <div class="transition-box">el-collapse-transition</div>
10.             </div>
11.         </el-collapse-transition>
12.     </div>
13. </div>
14. </template>
15.
16. <script>
17.     export default {
18.         data: () => ({
19.             show3: true
20.         })
21.     }
22. </script>
23.
24. <style>
25.     .transition-box {
26.         margin-bottom: 10px;
27.         width: 200px;
28.         height: 100px;
29.         border-radius: 4px;
30.         background-color: #409EFF;
31.         text-align: center;
32.         color: #fff;
33.         padding: 40px 20px;
34.         box-sizing: border-box;
35.         margin-right: 20px;
36.     }
37. </style>

```

按需引入

```

1. // fade/zoom 等
2. import 'element-ui/lib/theme-chalk/base.css';
3. // collapse 展开折叠

```

```
4. import CollapseTransition from 'element-ui/lib/transitions/collapse-  
    transition';  
5. import Vue from 'vue'  
6.  
7. Vue.component(CollapseTransition.name, CollapseTransition)
```

原文: <http://element-cn.eleme.io/#/zh-CN/component/transition>

Basic 基本组件

- [Layout 布局](#)
- [Container 布局容器](#)
- [Color 色彩](#)
- [Typography 字体](#)
- [Icon 图标](#)
- [Button 按钮](#)

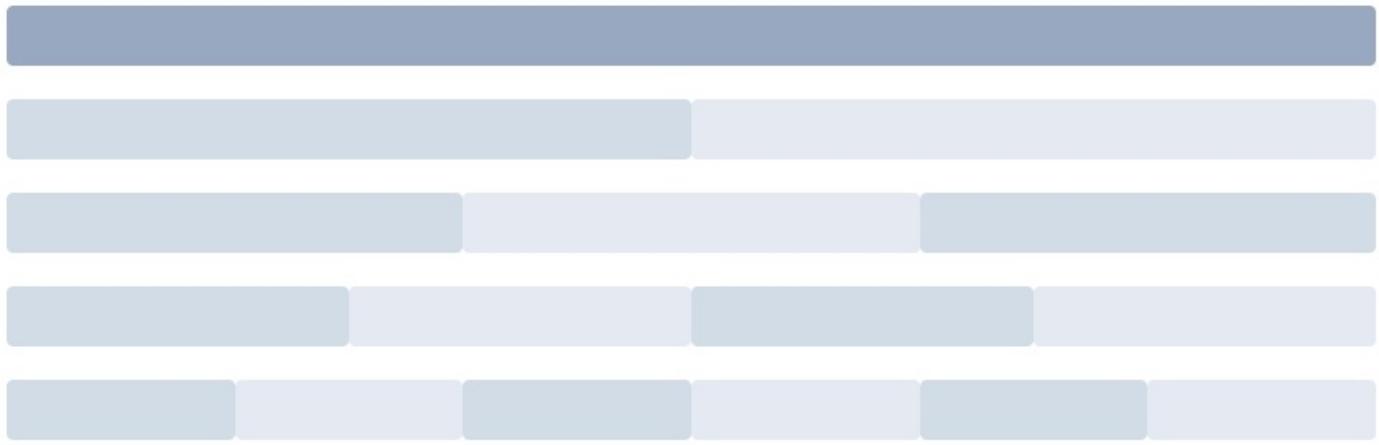
Layout 布局

Layout 布局

通过基础的 24 分栏，迅速简便地创建布局。

基础布局

使用单一分栏创建基础的栅格布局。



通过 `row` 和 `col` 组件，并通过 `col` 组件的 `span` 属性我们就可以自由地组合布局。

```
1. <el-row>
2.   <el-col :span="24"><div class="grid-content bg-purple-dark"></div></el-col>
3. </el-row>
4. <el-row>
5.   <el-col :span="12"><div class="grid-content bg-purple"></div></el-col>
6.   <el-col :span="12"><div class="grid-content bg-purple-light"></div></el-col>
7. </el-row>
8. <el-row>
9.   <el-col :span="8"><div class="grid-content bg-purple"></div></el-col>
10.  <el-col :span="8"><div class="grid-content bg-purple-light"></div></el-col>
11.  <el-col :span="8"><div class="grid-content bg-purple"></div></el-col>
12. </el-row>
13. <el-row>
14.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
15.   <el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>
16.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
17.   <el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>
18. </el-row>
```

```
19. <el-row>
20.   <el-col :span="4"><div class="grid-content bg-purple"></div></el-col>
21.   <el-col :span="4"><div class="grid-content bg-purple-light"></div></el-col>
22.   <el-col :span="4"><div class="grid-content bg-purple"></div></el-col>
23.   <el-col :span="4"><div class="grid-content bg-purple-light"></div></el-col>
24.   <el-col :span="4"><div class="grid-content bg-purple"></div></el-col>
25.   <el-col :span="4"><div class="grid-content bg-purple-light"></div></el-col>
26. </el-row>
27.
28. <style>
29.   .el-row {
30.     margin-bottom: 20px;
31.     &:last-child {
32.       margin-bottom: 0;
33.     }
34.   }
35.   .el-col {
36.     border-radius: 4px;
37.   }
38.   .bg-purple-dark {
39.     background: #99a9bf;
40.   }
41.   .bg-purple {
42.     background: #d3dce6;
43.   }
44.   .bg-purple-light {
45.     background: #e5e9f2;
46.   }
47.   .grid-content {
48.     border-radius: 4px;
49.     min-height: 36px;
50.   }
51.   .row-bg {
52.     padding: 10px 0;
53.     background-color: #f9fafc;
54.   }
55. </style>
```

分栏间隔

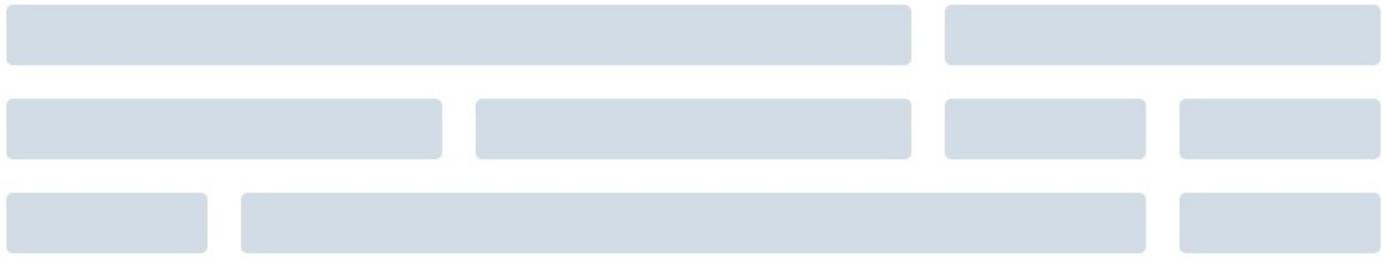
分栏之间存在间隔。

Row 组件 提供 `gutter` 属性来指定每一栏之间的间隔，默认间隔为 0。

```
1. <el-row :gutter="20">
2.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
3.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
4.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
5.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
6. </el-row>
7.
8. <style>
9.   .el-row {
10.     margin-bottom: 20px;
11.     &:last-child {
12.       margin-bottom: 0;
13.     }
14.   }
15.   .el-col {
16.     border-radius: 4px;
17.   }
18.   .bg-purple-dark {
19.     background: #99a9bf;
20.   }
21.   .bg-purple {
22.     background: #d3dce6;
23.   }
24.   .bg-purple-light {
25.     background: #e5e9f2;
26.   }
27.   .grid-content {
28.     border-radius: 4px;
29.     min-height: 36px;
30.   }
31.   .row-bg {
32.     padding: 10px 0;
33.     background-color: #f9fafc;
34.   }
35. </style>
```

混合布局

通过基础的 1/24 分栏任意扩展组合形成较为复杂的混合布局。

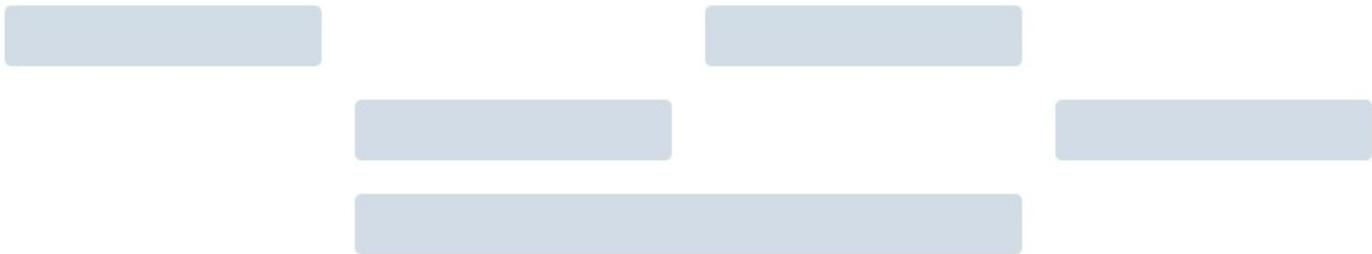


```
1. <el-row :gutter="20">
2.   <el-col :span="16"><div class="grid-content bg-purple"></div></el-col>
3.   <el-col :span="8"><div class="grid-content bg-purple"></div></el-col>
4. </el-row>
5. <el-row :gutter="20">
6.   <el-col :span="8"><div class="grid-content bg-purple"></div></el-col>
7.   <el-col :span="8"><div class="grid-content bg-purple"></div></el-col>
8.   <el-col :span="4"><div class="grid-content bg-purple"></div></el-col>
9.   <el-col :span="4"><div class="grid-content bg-purple"></div></el-col>
10. </el-row>
11. <el-row :gutter="20">
12.   <el-col :span="4"><div class="grid-content bg-purple"></div></el-col>
13.   <el-col :span="16"><div class="grid-content bg-purple"></div></el-col>
14.   <el-col :span="4"><div class="grid-content bg-purple"></div></el-col>
15. </el-row>
16.
17. <style>
18.   .el-row {
19.     margin-bottom: 20px;
20.     &:last-child {
21.       margin-bottom: 0;
22.     }
23.   }
24.   .el-col {
25.     border-radius: 4px;
26.   }
27.   .bg-purple-dark {
28.     background: #99a9bf;
29.   }
30.   .bg-purple {
31.     background: #d3dce6;
```

```
32. }
33. .bg-purple-light {
34.   background: #e5e9f2;
35. }
36. .grid-content {
37.   border-radius: 4px;
38.   min-height: 36px;
39. }
40. .row-bg {
41.   padding: 10px 0;
42.   background-color: #f9fafc;
43. }
44. </style>
```

分栏偏移

支持偏移指定的栏数。



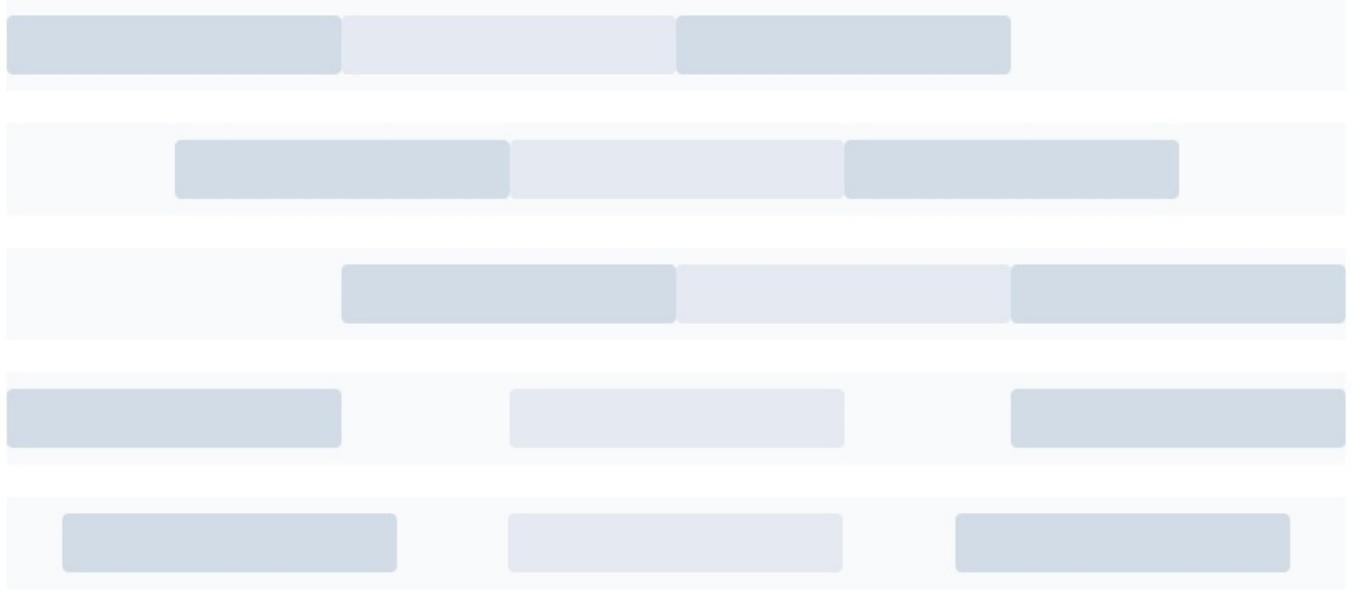
通过制定 col 组件的 `offset` 属性可以指定分栏偏移的栏数。

```
1. <el-row :gutter="20">
2.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
3.   <el-col :span="6" :offset="6"><div class="grid-content bg-purple"></div></el-
   col>
4. </el-row>
5. <el-row :gutter="20">
6.   <el-col :span="6" :offset="6"><div class="grid-content bg-purple"></div></el-
   col>
7.   <el-col :span="6" :offset="6"><div class="grid-content bg-purple"></div></el-
   col>
8. </el-row>
9. <el-row :gutter="20">
10.  <el-col :span="12" :offset="6"><div class="grid-content bg-purple"></div>
11. </el-col>
12. </el-row>
```

```
12.  
13. <style>  
14.   .el-row {  
15.     margin-bottom: 20px;  
16.     &:last-child {  
17.       margin-bottom: 0;  
18.     }  
19.   }  
20.   .el-col {  
21.     border-radius: 4px;  
22.   }  
23.   .bg-purple-dark {  
24.     background: #99a9bf;  
25.   }  
26.   .bg-purple {  
27.     background: #d3dce6;  
28.   }  
29.   .bg-purple-light {  
30.     background: #e5e9f2;  
31.   }  
32.   .grid-content {  
33.     border-radius: 4px;  
34.     min-height: 36px;  
35.   }  
36.   .row-bg {  
37.     padding: 10px 0;  
38.     background-color: #f9fafc;  
39.   }  
40. </style>
```

对齐方式

通过 **flex** 布局来对分栏进行灵活的对齐。



将 `type` 属性赋值为 'flex'，可以启用 flex 布局，并可通过 `justify` 属性来指定 `start`, `center`, `end`, `space-between`, `space-around` 其中的值来定义子元素的排版方式。

```
1. <el-row type="flex" class="row-bg">
2.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
3.   <el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>
4.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
5. </el-row>
6. <el-row type="flex" class="row-bg" justify="center">
7.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
8.   <el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>
9.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
10. </el-row>
11. <el-row type="flex" class="row-bg" justify="end">
12.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
13.   <el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>
14.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
15. </el-row>
16. <el-row type="flex" class="row-bg" justify="space-between">
17.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
18.   <el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>
19.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
20. </el-row>
21. <el-row type="flex" class="row-bg" justify="space-around">
22.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
23.   <el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>
24.   <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
25. </el-row>
```

```

26.
27. <style>
28.   .el-row {
29.     margin-bottom: 20px;
30.     &:last-child {
31.       margin-bottom: 0;
32.     }
33.   }
34.   .el-col {
35.     border-radius: 4px;
36.   }
37.   .bg-purple-dark {
38.     background: #99a9bf;
39.   }
40.   .bg-purple {
41.     background: #d3dce6;
42.   }
43.   .bg-purple-light {
44.     background: #e5e9f2;
45.   }
46.   .grid-content {
47.     border-radius: 4px;
48.     min-height: 36px;
49.   }
50.   .row-bg {
51.     padding: 10px 0;
52.     background-color: #f9fafc;
53.   }
54. </style>

```

[显示代码](#)

响应式布局

参照了 Bootstrap 的 响应式设计，预设了五个响应尺寸：`xs`、`sm`、`md`、`lg` 和 `xl`。



```

1. <el-row :gutter="10">
2.   <el-col :xs="8" :sm="6" :md="4" :lg="3" :xl="1"><div class="grid-content bg-

```

```

    purple"></div></el-col>
3.   <el-col :xs="4" :sm="6" :md="8" :lg="9" :xl="11"><div class="grid-content bg-
purple-light"></div></el-col>
4.   <el-col :xs="4" :sm="6" :md="8" :lg="9" :xl="11"><div class="grid-content bg-
purple"></div></el-col>
5.   <el-col :xs="8" :sm="6" :md="4" :lg="3" :xl="1"><div class="grid-content bg-
purple-light"></div></el-col>
6. </el-row>
7.
8. <style>
9.   .el-col {
10.     border-radius: 4px;
11.   }
12.   .bg-purple-dark {
13.     background: #99a9bf;
14.   }
15.   .bg-purple {
16.     background: #d3dce6;
17.   }
18.   .bg-purple-light {
19.     background: #e5e9f2;
20.   }
21.   .grid-content {
22.     border-radius: 4px;
23.     min-height: 36px;
24.   }
25. </style>

```

[显示代码](#)

基于断点的隐藏类

Element 额外提供了一系列类名，用于在某些条件下隐藏元素。这些类名可以添加在任何 DOM 元素或自定义组件上。如果需要，请自行引入以下文件：

```
1. import 'element-ui/lib/theme-chalk/display.css';
```

包含的类名及其含义为：

- hidden-xs-only - 当视口在 xs 尺寸时隐藏
- hidden-sm-only - 当视口在 sm 尺寸时隐藏
- hidden-sm-and-down - 当视口在 sm 及以下尺寸时隐藏

- hidden-sm-and-up - 当视口在 sm 及以上尺寸时隐藏
- hidden-md-only - 当视口在 md 尺寸时隐藏
- hidden-md-and-down - 当视口在 md 及以下尺寸时隐藏
- hidden-md-and-up - 当视口在 md 及以上尺寸时隐藏
- hidden-lg-only - 当视口在 lg 尺寸时隐藏
- hidden-lg-and-down - 当视口在 lg 及以下尺寸时隐藏
- hidden-lg-and-up - 当视口在 lg 及以上尺寸时隐藏
- hidden-xl-only - 当视口在 xl 尺寸时隐藏

Row Attributes

参数	说明	类型	可选值	默认值
gutter	栅格间隔	number	—	0
type	布局模式, 可选 flex, 现代浏览器下有效	string	—	—
justify	flex 布局下的水平排列方式	string	start/end/center/space-around/space-between	start
align	flex 布局下的垂直排列方式	string	top/middle/bottom	top
tag	自定义元素标签	string	*	div

Col Attributes

参数	说明	类型	可选值	默认值
span	栅格占据的列数	number	—	24
offset	栅格左侧的间隔格数	number	—	0
push	栅格向右移动格数	number	—	0
pull	栅格向左移动格数	number	—	0
xs	<768px 响应式栅格数或者栅格属性对象	number/object (例如: {span: 4, offset: 4})	—	—
sm	≥768px 响应式栅格数或者栅格属性对象	number/object (例如: {span: 4, offset: 4})	—	—
md	≥992px 响应式栅格数或者栅格属性对象	number/object (例如: {span: 4, offset: 4})	—	—
lg	≥1200px 响应式栅格数或者栅格属性对象	number/object (例如: {span: 4, offset: 4})	—	—
xl	≥1920px 响应式栅格数或者栅格属性对象	number/object (例如: {span: 4, offset: 4})	—	—
tag	自定义元素标签	string	*	div

原文: <http://element-cn.eleme.io/#/zh-CN/component/Layout>

Container 布局容器

Container 布局容器

用于布局的容器组件，方便快速搭建页面的基本结构：

`<el-container>`：外层容器。当子元素中包含 `<el-header>` 或 `<el-footer>` 时，全部子元素会垂直上下排列，否则会水平左右排列。

`<el-header>`：顶栏容器。

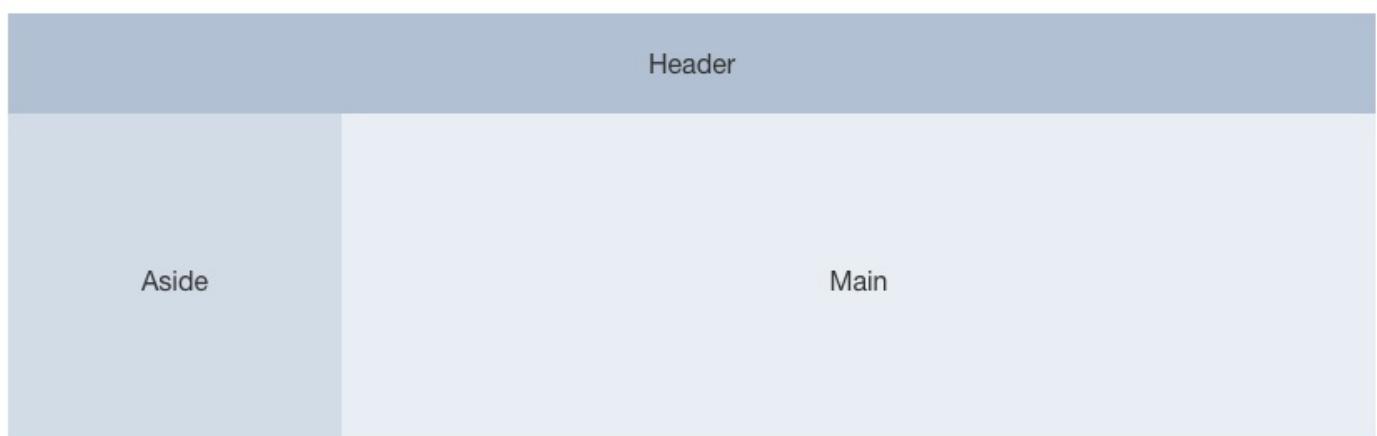
`<el-aside>`：侧边栏容器。

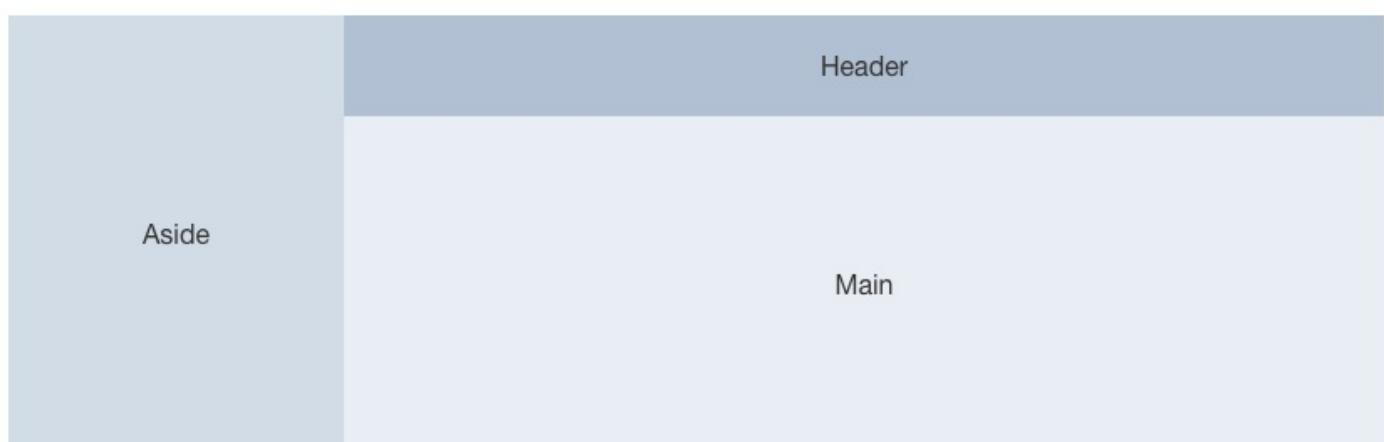
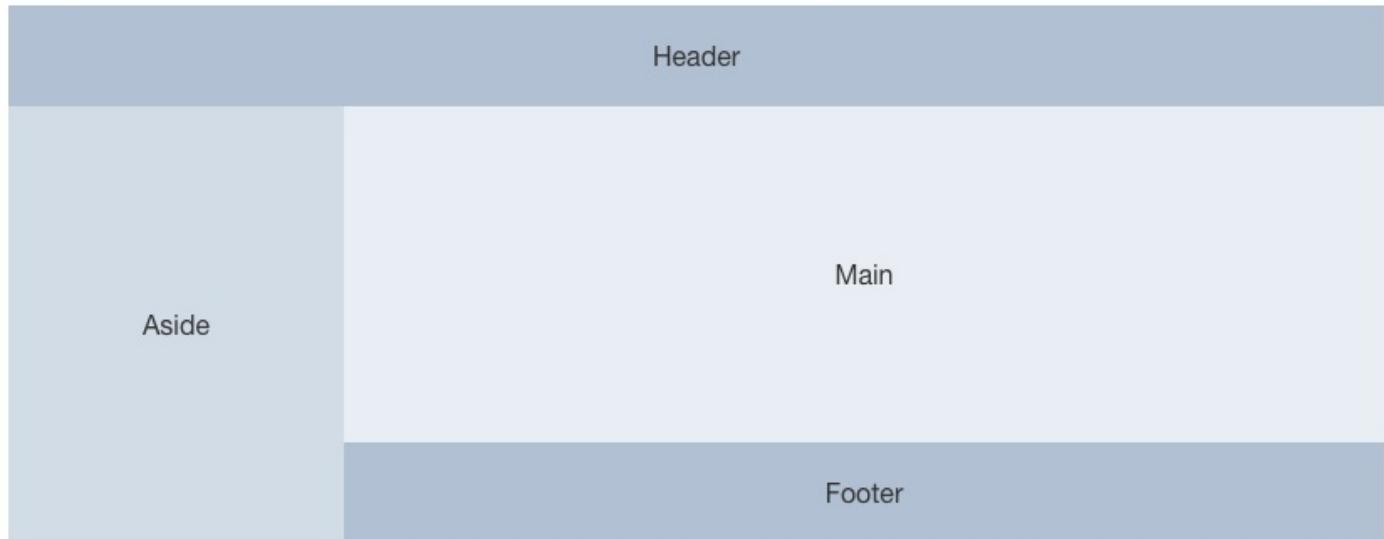
`<el-main>`：主要区域容器。

`<el-footer>`：底栏容器。

以上组件采用了 flex 布局，使用前请确定目标浏览器是否兼容。此外，`<el-container>` 的子元素只能是后四者，后四者的父元素也只能是 `<el-container>`。

常见页面布局





```
1. <el-container>
2.   <el-header>Header</el-header>
3.   <el-main>Main</el-main>
4. </el-container>
5.
6. <el-container>
```

```
7.  <el-header>Header</el-header>
8.  <el-main>Main</el-main>
9.  <el-footer>Footer</el-footer>
10. </el-container>
11.
12. <el-container>
13.   <el-aside width="200px">Aside</el-aside>
14.   <el-main>Main</el-main>
15. </el-container>
16.
17. <el-container>
18.   <el-header>Header</el-header>
19.   <el-container>
20.     <el-aside width="200px">Aside</el-aside>
21.     <el-main>Main</el-main>
22.   </el-container>
23. </el-container>
24.
25. <el-container>
26.   <el-header>Header</el-header>
27.   <el-container>
28.     <el-aside width="200px">Aside</el-aside>
29.     <el-container>
30.       <el-main>Main</el-main>
31.       <el-footer>Footer</el-footer>
32.     </el-container>
33.   </el-container>
34. </el-container>
35.
36. <el-container>
37.   <el-aside width="200px">Aside</el-aside>
38.   <el-container>
39.     <el-header>Header</el-header>
40.     <el-main>Main</el-main>
41.   </el-container>
42. </el-container>
43.
44. <el-container>
45.   <el-aside width="200px">Aside</el-aside>
46.   <el-container>
47.     <el-header>Header</el-header>
48.     <el-main>Main</el-main>
```

```
49.      <el-footer>Footer</el-footer>
50.    </el-container>
51.  </el-container>
52.
53. <style>
54.   .el-header, .el-footer {
55.     background-color: #B3C0D1;
56.     color: #333;
57.     text-align: center;
58.     line-height: 60px;
59.   }
60.
61.   .el-aside {
62.     background-color: #D3DCE6;
63.     color: #333;
64.     text-align: center;
65.     line-height: 200px;
66.   }
67.
68.   .el-main {
69.     background-color: #E9EEF3;
70.     color: #333;
71.     text-align: center;
72.     line-height: 160px;
73.   }
74.
75.   body > .el-container {
76.     margin-bottom: 40px;
77.   }
78.
79.   .el-container:nth-child(5) .el-aside,
80.   .el-container:nth-child(6) .el-aside {
81.     line-height: 260px;
82.   }
83.
84.   .el-container:nth-child(7) .el-aside {
85.     line-height: 320px;
86.   }
87. </style>
```

显示代码

实例

	日期	姓名	地址
选项1	2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
选项2	2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
分组2	2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
选项3	2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
选项4	2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
导航二	2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
分组一	2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
选项1	2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄

```

1. <el-container style="height: 500px; border: 1px solid #eee">
2.   <el-aside width="200px" style="background-color: rgb(238, 241, 246)">
3.     <el-menu :default-openeds="['1', '3']">
4.       <el-submenu index="1">
5.         <template slot="title"><i class="el-icon-message"></i>导航一</template>
6.         <el-menu-item-group>
7.           <template slot="title">分组一</template>
8.           <el-menu-item index="1-1">选项1</el-menu-item>
9.           <el-menu-item index="1-2">选项2</el-menu-item>
10.        </el-menu-item-group>
11.        <el-menu-item-group title="分组2">
12.          <el-menu-item index="1-3">选项3</el-menu-item>
13.        </el-menu-item-group>
14.        <el-submenu index="1-4">
15.          <template slot="title">选项4</template>
16.          <el-menu-item index="1-4-1">选项4-1</el-menu-item>
17.        </el-submenu>
18.      </el-submenu>
19.      <el-submenu index="2">
20.        <template slot="title"><i class="el-icon-menu"></i>导航二</template>
21.        <el-menu-item-group>
```

```
22.          <template slot="title">分组一</template>
23.          <el-menu-item index="2-1">选项1</el-menu-item>
24.          <el-menu-item index="2-2">选项2</el-menu-item>
25.        </el-menu-item-group>
26.        <el-menu-item-group title="分组2">
27.          <el-menu-item index="2-3">选项3</el-menu-item>
28.        </el-menu-item-group>
29.        <el-submenu index="2-4">
30.          <template slot="title">选项4</template>
31.          <el-menu-item index="2-4-1">选项4-1</el-menu-item>
32.        </el-submenu>
33.      </el-submenu>
34.      <el-submenu index="3">
35.        <template slot="title"><i class="el-icon-setting"></i>导航三</template>
36.        <el-menu-item-group>
37.          <template slot="title">分组一</template>
38.          <el-menu-item index="3-1">选项1</el-menu-item>
39.          <el-menu-item index="3-2">选项2</el-menu-item>
40.        </el-menu-item-group>
41.        <el-menu-item-group title="分组2">
42.          <el-menu-item index="3-3">选项3</el-menu-item>
43.        </el-menu-item-group>
44.        <el-submenu index="3-4">
45.          <template slot="title">选项4</template>
46.          <el-menu-item index="3-4-1">选项4-1</el-menu-item>
47.        </el-submenu>
48.      </el-submenu>
49.    </el-menu>
50.  </el-aside>
51.
52.  <el-container>
53.    <el-header style="text-align: right; font-size: 12px">
54.      <el-dropdown>
55.        <i class="el-icon-setting" style="margin-right: 15px"></i>
56.        <el-dropdown-menu slot="dropdown">
57.          <el-dropdown-item>查看</el-dropdown-item>
58.          <el-dropdown-item>新增</el-dropdown-item>
59.          <el-dropdown-item>删除</el-dropdown-item>
60.        </el-dropdown-menu>
61.      </el-dropdown>
62.      <span>王小虎</span>
63.    </el-header>
```

```
64.  
65.      <el-main>  
66.          <el-table :data="tableData">  
67.              <el-table-column prop="date" label="日期" width="140">  
68.                  </el-table-column>  
69.              <el-table-column prop="name" label="姓名" width="120">  
70.                  </el-table-column>  
71.              <el-table-column prop="address" label="地址">  
72.                  </el-table-column>  
73.          </el-table>  
74.      </el-main>  
75.  </el-container>  
76. </el-container>  
77.  
78. <style>  
79.     .el-header {  
80.         background-color: #B3C0D1;  
81.         color: #333;  
82.         line-height: 60px;  
83.     }  
84.  
85.     .el-aside {  
86.         color: #333;  
87.     }  
88. </style>  
89.  
90. <script>  
91.     export default {  
92.         data() {  
93.             const item = {  
94.                 date: '2016-05-02',  
95.                 name: '王小虎',  
96.                 address: '上海市普陀区金沙江路 1518 弄'  
97.             };  
98.             return {  
99.                 tableData: Array(20).fill(item)  
100.            }  
101.        }  
102.    };  
103. </script>
```

显示代码

Container Attributes

参数	说明	类型	可选值	默认值
direction	子元素的排列方向	string	horizontal / vertical	子元素中有 <code>el-header</code> 或 <code>el-footer</code> 时为 vertical, 否则为 horizontal

Header Attributes

参数	说明	类型	可选值	默认值
height	顶栏高度	string	-	60px

Aside Attributes

参数	说明	类型	可选值	默认值
width	侧边栏宽度	string	-	300px

Footer Attributes

参数	说明	类型	可选值	默认值
height	底栏高度	string	-	60px

原文: <http://element-cn.eleme.io/#/zh-CN/component/container>

Color 色彩

Color 色彩

Element 为了避免视觉传达差异，使用一套特定的调色板来规定颜色，为你所搭建的产品提供一致的外观视觉感受。

主色

Element 主要品牌颜色是鲜艳、友好的蓝色。



辅助色

除了主色外的场景色，需要在不同的场景中使用（例如危险色表示危险的操作）。



中性色

中性色用于文本、背景和边框颜色。通过运用不同的中性色，来表现层次结构。

主要文字 #303133	一级边框 #DCDFE6
常规文字 #606266	二级边框 #E4E7ED
次要文字 #909399	三级边框 #EBEEF5
占位文字 #C0C4CC	四级边框 #F2F6FC

原文: <http://element-cn.eleme.io/#/zh-CN/component/color>

Typography 字体

Typography 字体

我们对字体进行统一规范，力求在各个操作系统下都有最佳展示效果。

中文字体



英文 / 数字字体



Font-family 代码

```
1. font-family: "Helvetica Neue", Helvetica, "PingFang SC", "Hiragino Sans  
GB", "Microsoft YaHei", "微软雅黑", Arial, sans-serif;
```

字体使用规范

主标题	用 Element 快速搭建页面	20px Extra large
标题	用 Element 快速搭建页面	18px large

小标题	用 Element 快速搭建页面	16px Medium
正文	用 Element 快速搭建页面	14px Small
正文(小)	用 Element 快速搭建页面	13px Extra Small
辅助文字	用 Element 快速搭建页面	12px Extra Extra Small

原文: <http://element-cn.eleme.io/#/zh-CN/component/typography>

Icon 图标

Icon 图标

提供了一套常用的图标集合。

使用方法

直接通过设置类名为 `el-icon-iconName` 来使用即可。例如：

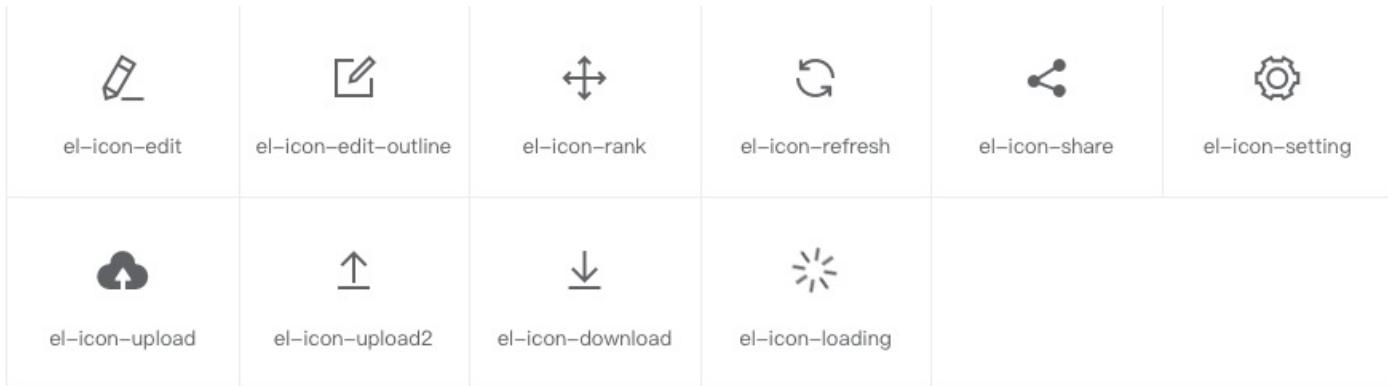


```
1. <i class="el-icon-edit"></i>
2. <i class="el-icon-share"></i>
3. <i class="el-icon-delete"></i>
4. <el-button type="primary" icon="el-icon-search">搜索</el-button>
5.
```

图标集合

el-icon-info	el-icon-error	el-icon-success	el-icon-warning	el-icon-question	el-icon-back
el-icon-arrow-left	el-icon-arrow-down	el-icon-arrow-right	el-icon-arrow-up	el-icon-caret-left	el-icon-caret-bottom
el-icon-caret-top	el-icon-caret-right	el-icon-d-arrow-left	el-icon-d-arrow-right	el-icon-minus	el-icon-plus
el-icon-remove	el-icon-circle-plus	el-icon-remove-outline	el-icon-circle-plus-outline	el-icon-close	el-icon-check
el-icon-circle-close	el-icon-circle-check	el-icon-circle-close-outline	el-icon-circle-check-outline	el-icon-zoom-out	el-icon-zoom-in

el-icon-d-caret	el-icon-sort	el-icon-sort-down	el-icon-sort-up	el-icon-tickets	el-icon-document
el-icon-goods	el-icon-sold-out	el-icon-news	el-icon-message	el-icon-date	el-icon-printer
el-icon-time	el-icon-bell	el-icon-mobile-phone	el-icon-service	el-icon-view	el-icon-menu
el-icon-more	el-icon-more-outline	el-icon-star-on	el-icon-star-off	el-icon-location	el-icon-location-outline
el-icon-phone	el-icon-phone-outline	el-icon-picture	el-icon-picture-outline	el-icon-delete	el-icon-search



- el-icon-info
- el-icon-error
- el-icon-success
- el-icon-warning
- el-icon-question
- el-icon-back
- el-icon-arrow-left
- el-icon-arrow-down
- el-icon-arrow-right
- el-icon-arrow-up
- el-icon-caret-left
- el-icon-caret-bottom
- el-icon-caret-top
- el-icon-caret-right
- el-icon-d-arrow-left
- el-icon-d-arrow-right
- el-icon-minus
- el-icon-plus
- el-icon-remove
- el-icon-circle-plus
- el-icon-remove-outline
- el-icon-circle-plus-outline
- el-icon-close
- el-icon-check
- el-icon-circle-close
- el-icon-circle-check
- el-icon-circle-close-outline
- el-icon-circle-check-outline
- el-icon-zoom-out
- el-icon-zoom-in
- el-icon-d-caret

- el-icon-sort
- el-icon-sort-down
- el-icon-sort-up
- el-icon-tickets
- el-icon-document
- el-icon-goods
- el-icon-sold-out
- el-icon-news
- el-icon-message
- el-icon-date
- el-icon-printer
- el-icon-time
- el-icon-bell
- el-icon-mobile-phone
- el-icon-service
- el-icon-view
- el-icon-menu
- el-icon-more
- el-icon-more-outline
- el-icon-star-on
- el-icon-star-off
- el-icon-location
- el-icon-location-outline
- el-icon-phone
- el-icon-phone-outline
- el-icon-picture
- el-icon-picture-outline
- el-icon-delete
- el-icon-search
- el-icon-edit
- el-icon-edit-outline
- el-icon-rank
- el-icon-refresh
- el-icon-share
- el-icon-setting
- el-icon-upload
- el-icon-upload2
- el-icon-download
- el-icon-loading

原文: <http://element-cn.eleme.io/#/zh-CN/component/icon>

Button 按钮

Button 按钮

常用的操作按钮。

基础用法

基础的按钮用法。



使用 `type`、`plain`、`round` 和 `circle` 属性来定义 Button 的样式。

```

1. <el-row>
2.   <el-button>默认按钮</el-button>
3.   <el-button type="primary">主要按钮</el-button>
4.   <el-button type="success">成功按钮</el-button>
5.   <el-button type="info">信息按钮</el-button>
6.   <el-button type="warning">警告按钮</el-button>
7.   <el-button type="danger">危险按钮</el-button>
8. </el-row>
9.
10. <el-row>
11.   <el-button plain>朴素按钮</el-button>
12.   <el-button type="primary" plain>主要按钮</el-button>
13.   <el-button type="success" plain>成功按钮</el-button>
14.   <el-button type="info" plain>信息按钮</el-button>
15.   <el-button type="warning" plain>警告按钮</el-button>
16.   <el-button type="danger" plain>危险按钮</el-button>
17. </el-row>
18.

```

```

19. <el-row>
20.   <el-button round>圆角按钮</el-button>
21.   <el-button type="primary" round>主要按钮</el-button>
22.   <el-button type="success" round>成功按钮</el-button>
23.   <el-button type="info" round>信息按钮</el-button>
24.   <el-button type="warning" round>警告按钮</el-button>
25.   <el-button type="danger" round>危险按钮</el-button>
26. </el-row>
27.
28. <el-row>
29.   <el-button icon="el-icon-search" circle></el-button>
30.   <el-button type="primary" icon="el-icon-edit" circle></el-button>
31.   <el-button type="success" icon="el-icon-check" circle></el-button>
32.   <el-button type="info" icon="el-icon-message" circle></el-button>
33.   <el-button type="warning" icon="el-icon-star-off" circle></el-button>
34.   <el-button type="danger" icon="el-icon-delete" circle></el-button>
35. </el-row>

```

显示代码

禁用状态

按钮不可用状态。



你可以使用 `disabled` 属性来定义按钮是否可用，它接受一个 `Boolean` 值。

```

1. <el-row>
2.   <el-button disabled>默认按钮</el-button>
3.   <el-button type="primary" disabled>主要按钮</el-button>
4.   <el-button type="success" disabled>成功按钮</el-button>
5.   <el-button type="info" disabled>信息按钮</el-button>
6.   <el-button type="warning" disabled>警告按钮</el-button>
7.   <el-button type="danger" disabled>危险按钮</el-button>
8. </el-row>
9.
10. <el-row>
11.   <el-button plain disabled>朴素按钮</el-button>

```

```

12. <el-button type="primary" plain disabled>主要按钮</el-button>
13. <el-button type="success" plain disabled>成功按钮</el-button>
14. <el-button type="info" plain disabled>信息按钮</el-button>
15. <el-button type="warning" plain disabled>警告按钮</el-button>
16. <el-button type="danger" plain disabled>危险按钮</el-button>
17. </el-row>

```

文字按钮

没有边框和背景色的按钮。

[文字按钮](#) [文字按钮](#)

```

1. <el-button type="text">文字按钮</el-button>
2. <el-button type="text" disabled>文字按钮</el-button>

```

图标按钮

带图标的按钮可增强辨识度（有文字）或节省空间（无文字）。



设置 `icon` 属性即可，`icon` 的列表可以参考 Element 的 `icon` 组件，也可以设置在文字右边的 `icon`，只要使用 `i` 标签即可，可以使用自定义图标。

```

1. <el-button type="primary" icon="el-icon-edit"></el-button>
2. <el-button type="primary" icon="el-icon-share"></el-button>
3. <el-button type="primary" icon="el-icon-delete"></el-button>
4. <el-button type="primary" icon="el-icon-search">搜索</el-button>
5. <el-button type="primary">上传<i class="el-icon-upload el-icon--right"></i></el-button>

```

按钮组

以按钮组的方式出现，常用于多项类似操作。



使用 `<el-button-group>` 标签来嵌套你的按钮。

```

1. <el-button-group>
2.   <el-button type="primary" icon="el-icon-arrow-left">上一页</el-button>
3.   <el-button type="primary">下一页<i class="el-icon-arrow-right el-icon--right">
4.   </i></el-button>
5. </el-button-group>
6.   <el-button type="primary" icon="el-icon-edit"></el-button>
7.   <el-button type="primary" icon="el-icon-share"></el-button>
8.   <el-button type="primary" icon="el-icon-delete"></el-button>
9. </el-button-group>

```

加载中

点击按钮后进行数据加载操作，在按钮上显示加载状态。



要设置为 `loading` 状态，只要设置 `loading` 属性为 `true` 即可。

```
1. <el-button type="primary" :loading="true">加载中</el-button>
```

显示代码

不同尺寸

`Button` 组件提供除了默认值以外的三种尺寸，可以在不同场景下选择合适的按钮尺寸。



额外的尺寸：`medium`、`small`、`mini`，通过设置 `size` 属性来配置它们。

```

1. <el-row>
2.   <el-button>默认按钮</el-button>
3.   <el-button size="medium">中等按钮</el-button>
4.   <el-button size="small">小型按钮</el-button>

```

```

5.   <el-button size="mini">超小按钮</el-button>
6.   </el-row>
7.   <el-row>
8.     <el-button round>默认按钮</el-button>
9.     <el-button size="medium" round>中等按钮</el-button>
10.    <el-button size="small" round>小型按钮</el-button>
11.    <el-button size="mini" round>超小按钮</el-button>
12.  </el-row>

```

Attributes

参数	说明	类型	可选值	默认值
size	尺寸	string	medium / small / mini	—
type	类型	string	primary / success / warning / danger / info / text	—
plain	是否朴素按钮	boolean	—	false
round	是否圆角按钮	boolean	—	false
circle	是否圆形按钮	boolean	—	false
loading	是否加载中状态	boolean	—	false
disabled	是否禁用状态	boolean	—	false
icon	图标类名	string	—	—
autofocus	是否默认聚焦	boolean	—	false
native-type	原生 type 属性	string	button / submit / reset	button

原文: <http://element-cn.eleme.io/#/zh-CN/component/button>

Form 表单组件

- [Radio 单选框](#)
- [Checkbox 多选框](#)
- [Input 输入框](#)
- [InputNumber 计数器](#)
- [Select 选择器](#)
- [Cascader 级联选择器](#)
- [Switch 开关](#)
- [Slider 滑块](#)
- [TimePicker 时间选择器](#)
- [DatePicker 日期选择器](#)
- [DateTimePicker 日期时间选择器](#)
- [Upload 上传](#)
- [Rate 评分](#)
- [ColorPicker 颜色选择器](#)
- [Transfer 穿梭框](#)
- [Form 表单](#)

Radio 单选框

Radio 单选框

在一组备选项中进行单选

基础用法

由于选项默认可见，不宜过多，若选项过多，建议使用 Select 选择器。

备选项 备选项

要使用 Radio 组件，只需要设置 `v-model` 绑定变量，选中意味着变量的值为相应 Radio `label` 属性的值，`label` 可以是 `String`、`Number` 或 `Boolean`。

```

1. <template>
2.   <el-radio v-model="radio" label="1">备选项</el-radio>
3.   <el-radio v-model="radio" label="2">备选项</el-radio>
4. </template>
5.
6. <script>
7.   export default {
8.     data () {
9.       return {
10.         radio: '1'
11.       };
12.     }
13.   }
14. </script>

```

禁用状态

单选框不可用的状态。

备选项 备选项

只要在 `el-radio` 元素中设置 `disabled` 属性即可，它接受一个 `Boolean`，`true` 为禁用。

```

1. <template>
2.   <el-radio disabled v-model="radio1" label="禁用">备选项</el-radio>
3.   <el-radio disabled v-model="radio1" label="选中且禁用">备选项</el-radio>
4. </template>
5.
6. <script>
7.   export default {
8.     data () {
9.       return {
10.         radio1: '选中且禁用'
11.       };
12.     }
13.   }
14. </script>

```

[显示代码](#)

单选框组

适用于在多个互斥的选项中选择的场景



结合 `el-radio-group` 元素和子元素 `el-radio` 可以实现单选组，在 `el-radio-group` 中绑定 `v-model`，在 `el-radio` 中设置好 `label` 即可，无需再给每一个 `el-radio` 绑定变量，另外，还提供了 `change` 事件来响应变化，它会传入一个参数 `value`。

```

1. <template>
2.   <el-radio-group v-model="radio2">
3.     <el-radio :label="3">备选项</el-radio>
4.     <el-radio :label="6">备选项</el-radio>
5.     <el-radio :label="9">备选项</el-radio>
6.   </el-radio-group>
7. </template>
8.
9. <script>
10.  export default {
11.    data () {
12.      return {
13.        radio2: 3

```

```

14.      };
15.    }
16.  }
17. </script>

```

按钮样式

按钮样式的单选组合。



只需要把 `el-radio` 元素换成 `el-radio-button` 元素即可，此外，Element 还提供了 `size` 属性。

```

1.  <template>
2.  <div>
3.    <el-radio-group v-model="radio3">
4.      <el-radio-button label="上海"></el-radio-button>
5.      <el-radio-button label="北京"></el-radio-button>
6.      <el-radio-button label="广州"></el-radio-button>
7.      <el-radio-button label="深圳"></el-radio-button>
8.    </el-radio-group>
9.  </div>
10. <div style="margin-top: 20px">
11.   <el-radio-group v-model="radio4" size="medium">
12.     <el-radio-button label="上海" ></el-radio-button>
13.     <el-radio-button label="北京"></el-radio-button>
14.     <el-radio-button label="广州"></el-radio-button>
15.     <el-radio-button label="深圳"></el-radio-button>
16.   </el-radio-group>
17. </div>
18. <div style="margin-top: 20px">
19.   <el-radio-group v-model="radio5" size="small">
20.     <el-radio-button label="上海"></el-radio-button>

```

```

21.      <el-radio-button label="北京" disabled ></el-radio-button>
22.      <el-radio-button label="广州"></el-radio-button>
23.      <el-radio-button label="深圳"></el-radio-button>
24.    </el-radio-group>
25.  </div>
26.  <div style="margin-top: 20px">
27.    <el-radio-group v-model="radio6" disabled size="mini">
28.      <el-radio-button label="上海"></el-radio-button>
29.      <el-radio-button label="北京"></el-radio-button>
30.      <el-radio-button label="广州"></el-radio-button>
31.      <el-radio-button label="深圳"></el-radio-button>
32.    </el-radio-group>
33.  </div>
34. </template>
35.
36. <script>
37.   export default {
38.     data () {
39.       return {
40.         radio3: '上海',
41.         radio4: '上海',
42.         radio5: '上海',
43.         radio6: '上海'
44.       };
45.     }
46.   }
47. </script>

```

带有边框



设置 `border` 属性可以渲染为带有边框的单选框。

```

1. <template>
2.   <div>
3.     <el-radio v-model="radio7" label="1" border>备选项1</el-radio>
4.     <el-radio v-model="radio7" label="2" border>备选项2</el-radio>
5.   </div>
6.   <div style="margin-top: 20px">
7.     <el-radio v-model="radio8" label="1" border size="medium">备选项1</el-radio>
8.     <el-radio v-model="radio8" label="2" border size="medium">备选项2</el-radio>
9.   </div>
10.  <div style="margin-top: 20px">
11.    <el-radio-group v-model="radio9" size="small">
12.      <el-radio label="1" border>备选项1</el-radio>
13.      <el-radio label="2" border disabled>备选项2</el-radio>
14.    </el-radio-group>
15.  </div>
16.  <div style="margin-top: 20px">
17.    <el-radio-group v-model="radio10" size="mini" disabled>
18.      <el-radio label="1" border>备选项1</el-radio>
19.      <el-radio label="2" border>备选项2</el-radio>
20.    </el-radio-group>
21.  </div>
22. </template>
23.
24. <script>
25.   export default {
26.     data () {
27.       return {
28.         radio7: '1',
29.         radio8: '1',
30.         radio9: '1',
31.         radio10: '1'
32.       };
33.     }
34.   }
35. </script>

```

Radio Attributes

参数	说明	类型	可选值	默认值
label	Radio 的 value	string / number / boolean	-	-
disabled	是否禁用	boolean	-	false

border	是否显示边框	boolean	—	false
size	Radio 的尺寸，仅在 border 为真时有效	string	medium / small / mini	—
name	原生 name 属性	string	—	—

Radio Events

事件名称	说明	回调参数
change	绑定值变化时触发的事件	选中的 Radio label 值

Radio-group Attributes

参数	说明	类型	可选值	默认值
size	单选框组尺寸，仅对按钮形式的 Radio 或带有边框的 Radio 有效	string	medium / small / mini	—
disabled	是否禁用	boolean	—	false
text-color	按钮形式的 Radio 激活时的文本颜色	string	—	#ffffff
fill	按钮形式的 Radio 激活时的填充色和边框色	string	—	#409EFF

Radio-group Events

事件名称	说明	回调参数
change	绑定值变化时触发的事件	选中的 Radio label 值

Radio-button Attributes

参数	说明	类型	可选值	默认值
label	Radio 的 value	string / number	—	—
disabled	是否禁用	boolean	—	false
name	原生 name 属性	string	—	—

原文：<http://element-cn.eleme.io/#/zh-CN/component/radio>

Checkbox 多选框

Checkbox 多选框

一组备选项中进行多选

基础用法

单独使用可以表示两种状态之间的切换，写在标签中的内容为 checkbox 按钮后的介绍。

备选项

在 `el-checkbox` 元素中定义 `v-model` 绑定变量，单一的 `checkbox` 中，默认绑定变量的值会是 `Boolean`，选中为 `true`。

```

1. <template>
2.   <!-- `checked` 为 true 或 false -->
3.   <el-checkbox v-model="checked">备选项</el-checkbox>
4. </template>
5. <script>
6. export default {
7.   data() {
8.     return {
9.       checked: true
10.    };
11.   }
12. }
13. </script>
```

禁用状态

多选框不可用状态。

备选项1 备选项

设置 `disabled` 属性即可。

```

1. <template>
2.   <el-checkbox v-model="checked1" disabled>备选项1</el-checkbox>
3.   <el-checkbox v-model="checked2" disabled>备选项2</el-checkbox>
4. </template>
5. <script>
6.   export default {
7.     data() {
8.       return {
9.         checked1: false,
10.        checked2: true
11.      };
12.    }
13.  };
14. </script>

```

多选框组

适用于多个勾选框绑定到同一个数组的情景，通过是否勾选来表示这一组选项中选中的项。

复选框 A 复选框 B 复选框 C 禁用 选中且禁用

checkbox-group 元素能把多个 checkbox 管理为一组，只需要在 Group 中使用 **v-model** 绑定 **Array** 类型的变量即可。**el-checkbox** 的 **label** 属性是该 checkbox 对应的值，若该标签中无内容，则该属性也充当 checkbox 按钮后的介绍。**label** 与数组中的元素值相对应，如果存在指定的值则为选中状态，否则为不选中。

```

1. <template>
2.   <el-checkbox-group v-model="checkList">
3.     <el-checkbox label="复选框 A"></el-checkbox>
4.     <el-checkbox label="复选框 B"></el-checkbox>
5.     <el-checkbox label="复选框 C"></el-checkbox>
6.     <el-checkbox label="禁用" disabled></el-checkbox>
7.     <el-checkbox label="选中且禁用" disabled></el-checkbox>
8.   </el-checkbox-group>
9. </template>
10.
11. <script>
12.   export default {
13.     data () {
14.       return {

```

```

15.         checkList: ['选中且禁用', '复选框 A']
16.     };
17. }
18. };
19. </script>

```

indeterminate 状态

indeterminate 属性用以表示 checkbox 的不确定状态，一般用于实现全选的效果

全选
 上海 北京 广州 深圳

```

1. <template>
2.   <el-checkbox :indeterminate="isIndeterminate" v-model="checkAll"
3.     @change="handleCheckAllChange">全选</el-checkbox>
4.   <div style="margin: 15px 0;"></div>
5.   <el-checkbox-group v-model="checkedCities"
6.     @change="handleCheckedCitiesChange">
7.     <el-checkbox v-for="city in cities" :label="city" :key="city">{{city}}</el-
8.       checkbox>
9.   </el-checkbox-group>
10.  </template>
11.  <script>
12.    const cityOptions = ['上海', '北京', '广州', '深圳'];
13.    export default {
14.      data() {
15.        return {
16.          checkAll: false,
17.          checkedCities: ['上海', '北京'],
18.          cities: cityOptions,
19.          isIndeterminate: true
20.        };
21.      },
22.      methods: {
23.        handleCheckAllChange(val) {
24.          this.checkedCities = val ? cityOptions : [];
25.          this.isIndeterminate = false;
26.        },
27.        handleCheckedCitiesChange(value) {
28.          if (!value.length) {
29.            this.isIndeterminate = false;
30.          } else if (value.length === this.cities.length) {
31.            this.isIndeterminate = false;
32.          } else {
33.            this.isIndeterminate = true;
34.          }
35.        }
36.      }
37.    }
38.  </script>

```

```

25.     let checkedCount = value.length;
26.     this.checkAll = checkedCount === this.cities.length;
27.     this.isIndeterminate = checkedCount > 0 && checkedCount <
    this.cities.length;
28.   }
29. }
30. );
31. </script>

```

可选项目数量的限制

使用 `min` 和 `max` 属性能够限制可以被勾选的项目的数量。

上海 北京 广州 深圳

```

1.  <template>
2.    <el-checkbox-group
3.      v-model="checkedCities1"
4.      :min="1"
5.      :max="2">
6.        <el-checkbox v-for="city in cities" :label="city" :key="city">{{city}}</el-
    checkbox>
7.    </el-checkbox-group>
8.  </template>
9.  <script>
10.    const cityOptions = ['上海', '北京', '广州', '深圳'];
11.    export default {
12.      data() {
13.        return {
14.          checkedCities1: ['上海', '北京'],
15.          cities: cityOptions
16.        };
17.      }
18.    };
19.  </script>

```

按钮样式

按钮样式的多选组合。



只需要把 `el-checkbox` 元素替换为 `el-checkbox-button` 元素即可。此外，Element 还提供了 `size` 属性。

```

1. <template>
2.   <div>
3.     <el-checkbox-group v-model="checkboxGroup1">
4.       <el-checkbox-button v-for="city in cities" :label="city" :key="city">
5.         {{city}}</el-checkbox-button>
6.     </el-checkbox-group>
7.   </div>
8.   <div style="margin-top: 20px">
9.     <el-checkbox-group v-model="checkboxGroup2" size="medium">
10.      <el-checkbox-button v-for="city in cities" :label="city" :key="city">
11.        {{city}}</el-checkbox-button>
12.      </el-checkbox-group>
13.    </div>
14.    <div style="margin-top: 20px">
15.      <el-checkbox-group v-model="checkboxGroup3" size="small">
16.        <el-checkbox-button v-for="city in cities" :label="city" :disabled="city === '北京'" :key="city">{{city}}</el-checkbox-button>
17.      </el-checkbox-group>
18.    </div>
19.    <div style="margin-top: 20px">
20.      <el-checkbox-group v-model="checkboxGroup4" size="mini" disabled>
21.        <el-checkbox-button v-for="city in cities" :label="city" :key="city">
22.          {{city}}</el-checkbox-button>
23.        </el-checkbox-group>
24.      </div>
25.    </template>
<script>
  const cityOptions = ['上海', '北京', '广州', '深圳'];
  export default {

```

```

26.     data () {
27.       return {
28.         checkboxGroup1: ['上海'],
29.         checkboxGroup2: ['上海'],
30.         checkboxGroup3: ['上海'],
31.         checkboxGroup4: ['上海'],
32.         cities: cityOptions
33.       };
34.     }
35.   }
36. </script>

```

带有边框



设置 `border` 属性可以渲染为带有边框的多选框。

```

1.  <template>
2.    <div>
3.      <el-checkbox v-model="checked3" label="备选项1" border></el-checkbox>
4.      <el-checkbox v-model="checked4" label="备选项2" border></el-checkbox>
5.    </div>
6.    <div style="margin-top: 20px">
7.      <el-checkbox v-model="checked5" label="备选项1" border size="medium"></el-
checkbox>
8.      <el-checkbox v-model="checked6" label="备选项2" border size="medium"></el-
checkbox>
9.    </div>
10.   <div style="margin-top: 20px">
11.     <el-checkbox-group v-model="checkboxGroup5" size="small">
12.       <el-checkbox label="备选项1" border></el-checkbox>
13.       <el-checkbox label="备选项2" border disabled></el-checkbox>
14.     </el-checkbox-group>

```

```

15.   </div>
16.   <div style="margin-top: 20px">
17.     <el-checkbox-group v-model="checkboxGroup6" size="mini" disabled>
18.       <el-checkbox label="备选项1" border></el-checkbox>
19.       <el-checkbox label="备选项2" border></el-checkbox>
20.     </el-checkbox-group>
21.   </div>
22. </template>
23.
24. <script>
25.   export default {
26.     data () {
27.       return {
28.         checked3: true,
29.         checked4: false,
30.         checked5: false,
31.         checked6: true,
32.         checkboxGroup5: [],
33.         checkboxGroup6: []
34.       };
35.     }
36.   }
37. </script>

```

Checkbox Attributes

参数	说明	类型	可选值	默认值
label	选中状态的值（只有在 checkbox-group 或者绑定对象类型为 array 时有效）	string / number / boolean	-	-
true-label	选中时的值	string / number	-	-
false-label	没有选中时的值	string / number	-	-
disabled	是否禁用	boolean	-	false
border	是否显示边框	boolean	-	false
size	Checkbox 的尺寸，仅在 border 为真时有效	string	medium / small / mini	-
name	原生 name 属性	string	-	-
checked	当前是否勾选	boolean	-	false
indeterminate	设置 indeterminate 状态，只负责样式控制	boolean	-	false

Checkbox Events

事件名称	说明	回调参数
change	当绑定值变化时触发的事件	更新后的值

Checkbox-group Attributes

参数	说明	类型	可选值	默认值
size	多选框组尺寸，仅对按钮形式的 Checkbox 或带有边框的 Checkbox 有效	string	medium / small / mini	-
disabled	是否禁用	boolean	-	false
min	可被勾选的 checkbox 的最小数量	number	-	-
max	可被勾选的 checkbox 的最大数量	number	-	-
text-color	按钮形式的 Checkbox 激活时的文本颜色	string	-	#ffffff
fill	按钮形式的 Checkbox 激活时的填充色和边框色	string	-	#409EFF

Checkbox-group Events

事件名称	说明	回调参数
change	当绑定值变化时触发的事件	更新后的值

Checkbox-button Attributes

参数	说明	类型	可选值	默认值
label	选中状态的值 (只有在 <code>checkbox-group</code> 或者绑定对象类型为 <code>array</code> 时有效)	string / number / boolean	-	-
true-label	选中时的值	string / number	-	-
false-label	没有选中时的值	string / number	-	-
disabled	是否禁用	boolean	-	false
name	原生 name 属性	string	-	-
checked	当前是否勾选	boolean	-	false

原文: <http://element-cn.eleme.io/#/zh-CN/component/checkbox>

Input 输入框

Input 输入框

通过鼠标或键盘输入字符

基础用法

```
请输入内容
```

```
1. <el-input v-model="input" placeholder="请输入内容"></el-input>
2.
3. <script>
4. export default {
5.   data() {
6.     return {
7.       input: ''
8.     }
9.   }
10. }
11. </script>
```

禁用状态

```
请输入内容
```

通过 `disabled` 属性指定是否禁用 input 组件

```
1. <el-input
2.   placeholder="请输入内容"
3.   v-model="input1"
4.   :disabled="true">
5. </el-input>
6.
7. <script>
8. export default {
```

```

9.   data() {
10.     return {
11.       input1: ''
12.     }
13.   }
14. }
15. </script>

```

可清空



使用 `clearable` 属性即可得到一个可清空的输入框

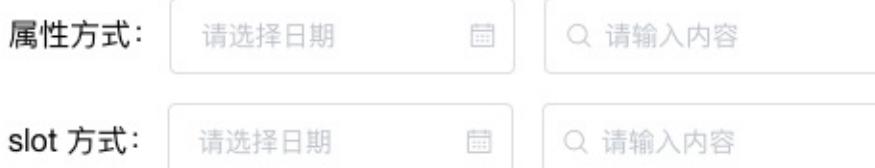
```

1. <el-input
2.   placeholder="请输入内容"
3.   v-model="input10"
4.   clearable>
5. </el-input>
6.
7. <script>
8.   export default {
9.     data() {
10.       return {
11.         input10: ''
12.       }
13.     }
14.   }
15. </script>

```

带 icon 的输入框

带有图标标记输入类型

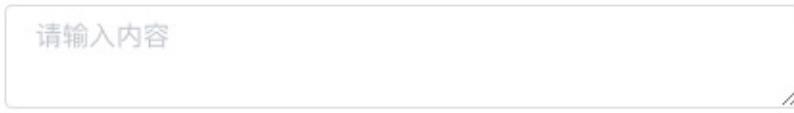


可以通过 `prefix-icon` 和 `suffix-icon` 属性在 `input` 组件首部和尾部增加显示图标，也可以通过 `slot` 来放置图标。

```
1. <div class="demo-input-suffix">
2.   属性方式：
3.   <el-input
4.     placeholder="请选择日期"
5.     suffix-icon="el-icon-date"
6.     v-model="input2">
7.   </el-input>
8.   <el-input
9.     placeholder="请输入内容"
10.    prefix-icon="el-icon-search"
11.    v-model="input21">
12.   </el-input>
13. </div>
14. <div class="demo-input-suffix">
15.   slot 方式：
16.   <el-input
17.     placeholder="请选择日期"
18.     v-model="input22">
19.     <i slot="suffix" class="el-input__icon el-icon-date"></i>
20.   </el-input>
21.   <el-input
22.     placeholder="请输入内容"
23.     v-model="input23">
24.     <i slot="prefix" class="el-input__icon el-icon-search"></i>
25.   </el-input>
26. </div>
27.
28. <script>
29. export default {
30.   data() {
31.     return {
32.       input2: '',
33.       input21: '',
34.       input22: '',
35.       input23: ''
36.     }
37.   }
38. }
39. </script>
```

文本域

用于输入多行文本信息，通过将 `type` 属性的值指定为 `textarea`。



文本域高度可通过 `rows` 属性控制

```

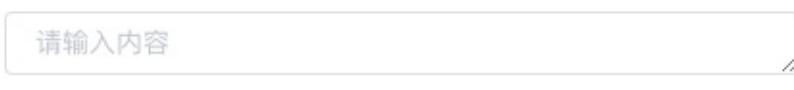
1. <el-input
2.   type="textarea"
3.   :rows="2"
4.   placeholder="请输入内容"
5.   v-model="textarea">
6. </el-input>
7.
8. <script>
9. export default {
10.   data() {
11.     return {
12.       textarea: ''
13.     }
14.   }
15. }
16. </script>

```

[显示代码](#)

可自适应文本高度的文本域

通过设置 `autosize` 属性可以使得文本域的高度能够根据文本内容自动进行调整，并且 `autosize` 还可以设定为一个对象，指定最小行数和最大行数。




```

1. <el-input
2.   type="textarea"

```

```

3.   autosize
4.   placeholder="请输入内容"
5.   v-model="textarea2">
6. </el-input>
7. <div style="margin: 20px 0;"></div>
8. <el-input
9.   type="textarea"
10.  :autosize="{ minRows: 2, maxRows: 4}"
11.  placeholder="请输入内容"
12.  v-model="textarea3">
13. </el-input>
14.
15. <script>
16. export default {
17.   data() {
18.     return {
19.       textarea2: '',
20.       textarea3: ''
21.     }
22.   }
23. }
24. </script>

```

复合型输入框

可前置或后置元素，一般为标签或按钮



可通过 slot 来指定在 input 中前置或者后置内容。

```

1. <div>
2.   <el-input placeholder="请输入内容" v-model="input3">
3.     <template slot="prepend">Http://</template>
4.   </el-input>
5. </div>
6. <div style="margin-top: 15px;">

```

```

7.   <el-input placeholder="请输入内容" v-model="input4">
8.     <template slot="append>.com</template>
9.   </el-input>
10. </div>
11. <div style="margin-top: 15px;">
12.   <el-input placeholder="请输入内容" v-model="input5" class="input-with-select">
13.     <el-select v-model="select" slot="prepend" placeholder="请选择">
14.       <el-option label="餐厅名" value="1"></el-option>
15.       <el-option label="订单号" value="2"></el-option>
16.       <el-option label="用户电话" value="3"></el-option>
17.     </el-select>
18.     <el-button slot="append" icon="el-icon-search"></el-button>
19.   </el-input>
20. </div>
21. <style>
22.   .el-select .el-input {
23.     width: 130px;
24.   }
25.   .input-with-select .el-input-group__prepend {
26.     background-color: #fff;
27.   }
28. </style>
29. <script>
30. export default {
31.   data() {
32.     return {
33.       input3: '',
34.       input4: '',
35.       input5: '',
36.       select: ''
37.     }
38.   }
39. }
40. </script>

```

尺寸



可通过 `size` 属性指定输入框的尺寸，除了默认的大小外，还提供了 `large`、`small` 和 `mini`

三种尺寸。

```
1. <div class="demo-input-size">
2.   <el-input
3.     placeholder="请输入内容"
4.     suffix-icon="el-icon-date"
5.     v-model="input6">
6.   </el-input>
7.   <el-input
8.     size="medium"
9.     placeholder="请输入内容"
10.    suffix-icon="el-icon-date"
11.    v-model="input7">
12.  </el-input>
13.  <el-input
14.    size="small"
15.    placeholder="请输入内容"
16.    suffix-icon="el-icon-date"
17.    v-model="input8">
18.  </el-input>
19.  <el-input
20.    size="mini"
21.    placeholder="请输入内容"
22.    suffix-icon="el-icon-date"
23.    v-model="input9">
24.  </el-input>
25. </div>
26.
27. <script>
28. export default {
29.   data() {
30.     return {
31.       input6: '',
32.       input7: '',
33.       input8: '',
34.       input9: ''
35.     }
36.   }
37. }
38. </script>
```

带输入建议

根据输入内容提供对应的输入建议



autocomplete 是一个可带输入建议的输入框组件，`fetch-suggestions` 是一个返回输入建议的方法属性，如 `querySearch(queryString, cb)`，在该方法中你可以在你的输入建议数据准备好时通过 `cb(data)` 返回到 autocomplete 组件中。

```

1. <el-row class="demo-autocomplete">
2.   <el-col :span="12">
3.     <div class="sub-title">激活即列出输入建议</div>
4.     <el-autocomplete
5.       class="inline-input"
6.       v-model="state1"
7.       :fetch-suggestions="querySearch"
8.       placeholder="请输入内容"
9.       @select="handleSelect"
10.      ></el-autocomplete>
11.    </el-col>
12.    <el-col :span="12">
13.      <div class="sub-title">输入后匹配输入建议</div>
14.      <el-autocomplete
15.        class="inline-input"
16.        v-model="state2"
17.        :fetch-suggestions="querySearch"
18.        placeholder="请输入内容"
19.        :trigger-on-focus="false"
20.        @select="handleSelect"
21.      ></el-autocomplete>
22.    </el-col>
23.  </el-row>
24.  <script>
25.    export default {
26.      data() {
27.        return {
28.          restaurants: [],
29.          state1: '',

```

```

30.         state2: ''
31.     };
32. },
33. methods: {
34.     querySearch(queryString, cb) {
35.         var restaurants = this.restaurants;
36.         var results = queryString ?
37.             restaurants.filter(this.createFilter(queryString)) : restaurants;
38.             // 调用 callback 返回建议列表的数据
39.             cb(results);
40.         },
41.         createFilter(queryString) {
42.             return (restaurant) => {
43.                 return
44.                     (restaurant.value.toLowerCase().indexOf(queryString.toLowerCase()) === 0);
45.             };
46.         },
47.         loadAll() {
48.             return [
49.                 { "value": "三全鲜食（北新泾店）", "address": "长宁区新渔路144号" },
50.                 { "value": "Hot honey 首尔炸鸡（仙霞路）", "address": "上海市长宁区淞虹路
51. 661号" },
52.                 { "value": "新旺角茶餐厅", "address": "上海市普陀区真北路988号创邑金沙谷6号
53. 楼113" },
54.                 { "value": "泷千家(天山西路店)", "address": "天山西路438号" },
55.                 { "value": "胖仙女纸杯蛋糕（上海凌空店）", "address": "上海市长宁区金钟路968
56. 号1幢18号楼一层商铺18-101" },
57.                 { "value": "贡茶", "address": "上海市长宁区金钟路633号" },
58.                 { "value": "豪大大香鸡排超级奶爸", "address": "上海市嘉定区曹安公路曹安路
59. 1685号" },
60.                 { "value": "茶芝兰（奶茶，手抓饼）", "address": "上海市普陀区同普路1435号"
61. },
62.                 { "value": "十二泷町", "address": "上海市北翟路1444弄81号B幢-107" },
63.                 { "value": "星移浓缩咖啡", "address": "上海市嘉定区新郁路817号" },
64.                 { "value": "阿姨奶茶/豪大大", "address": "嘉定区曹安路1611号" },
65.                 { "value": "新麦甜四季甜品炸鸡", "address": "嘉定区曹安公路2383弄55号" },
66.                 { "value": "Monica摩托主题咖啡店", "address": "嘉定区江桥镇曹安公路2409号
67. 1F, 2383弄62号1F" },
68.                 { "value": "浮生若茶（凌空soho店）", "address": "上海长宁区金钟路968号9号楼
69. 地下一层" },
70.                 { "value": "NONO JUICE 鲜榨果汁", "address": "上海市长宁区天山西路119号"
71. },

```

```
62.          { "value": "CoCo都可(北新泾店)", "address": "上海市长宁区仙霞西路" },
63.          { "value": "快乐柠檬(神州智慧店)", "address": "上海市长宁区天山西路567号1
层R117号店铺" },
64.          { "value": "Merci Paul cafe", "address": "上海市普陀区光复西路丹巴路28弄6
号楼819" },
65.          { "value": "猫山王(西郊百联店)", "address": "上海市长宁区仙霞西路88号第一层
G05-F01-1-306" },
66.          { "value": "枪会山", "address": "上海市普陀区棕榈路" },
67.          { "value": "纵食", "address": "元丰天山花园(东门) 双流路267号" },
68.          { "value": "钱记", "address": "上海市长宁区天山西路" },
69.          { "value": "壹杯加", "address": "上海市长宁区通协路" },
70.          { "value": "妙哇滴咖", "address": "上海市长宁区新泾镇金钟路999号2幢(B幢) 第
01层第1-02A单元" },
71.          { "value": "爱茜茜里(西郊百联)", "address": "长宁区仙霞西路88号1305室" },
72.          { "value": "爱茜茜里(近铁广场)", "address": "上海市普陀区真北路818号近铁城市
广场北区地下二楼N-B2-02-C商铺" },
73.          { "value": "鲜果榨汁(金沙江路和美广店)", "address": "普陀区金沙江路2239号
金沙和美广场B1-10-6" },
74.          { "value": "开心丽果(缤谷店)", "address": "上海市长宁区威宁路天山路341号"
},
75.          { "value": "超级鸡车(丰庄路店)", "address": "上海市嘉定区丰庄路240号" },
76.          { "value": "妙生活果园(北新泾店)", "address": "长宁区新渔路144号" },
77.          { "value": "香宜度麻辣香锅", "address": "长宁区淞虹路148号" },
78.          { "value": "凡仔汉堡(老真北路店)", "address": "上海市普陀区老真北路160号"
},
79.          { "value": "港式小铺", "address": "上海市长宁区金钟路968号15楼15-105室" },
80.          { "value": "蜀香源麻辣香锅(剑河路店)", "address": "剑河路443-1" },
81.          { "value": "北京饺子馆", "address": "长宁区北新泾街道天山西路490-1号" },
82.          { "value": "饭典*新简餐(凌空SOHO店)", "address": "上海市长宁区金钟路968号
9号楼地下一层9-83室" },
83.          { "value": "焦耳·川式快餐(金钟路店)", "address": "上海市金钟路633号地下一
层甲部" },
84.          { "value": "动力鸡车", "address": "长宁区仙霞西路299弄3号101B" },
85.          { "value": "浏阳蒸菜", "address": "天山西路430号" },
86.          { "value": "四海游龙(天山西路店)", "address": "上海市长宁区天山西路" },
87.          { "value": "樱花食堂(凌空店)", "address": "上海市长宁区金钟路968号15楼15-
105室" },
88.          { "value": "壹分米客家传统调制米粉(天山店)", "address": "天山西路428号" },
89.          { "value": "福荣祥烧腊(平溪路店)", "address": "上海市长宁区协和路福泉路255
弄57-73号" },
90.          { "value": "速记黄焖鸡米饭", "address": "上海市长宁区北新泾街道金钟路180号1
层01号摊位" },
```

```

91.          { "value": "红辣椒麻辣烫", "address": "上海市长宁区天山西路492号" },
92.          { "value": "(小杨生煎)西郊百联餐厅", "address": "长宁区仙霞西路88号百联2楼"
93.          },
94.          { "value": "阳阳麻辣烫", "address": "天山西路389号" },
95.          { "value": "南拳妈妈龙虾盖浇饭", "address": "普陀区金沙江路1699号鑫乐惠美食
96.          广场A13" }
97.        ],
98.        handleSelect(item) {
99.          console.log(item);
100.        }
101.      },
102.      mounted() {
103.        this.restaurants = this.loadAll();
104.      }
105.    </script>

```

自定义模板

可自定义输入建议的显示



使用 `scoped slot` 自定义输入建议的模板。该 `scope` 的参数为 `item`，表示当前输入建议对象。

```

1.  <el-autocomplete
2.    popper-class="my-autocomplete"

```

```
3.   v-model="state3"
4.   :fetch-suggestions="querySearch"
5.   placeholder="请输入内容"
6.   @select="handleSelect">
7.   <i
8.     class="el-icon-edit el-input__icon"
9.     slot="suffix"
10.    @click="handleIconClick">
11.   </i>
12.   <template slot-scope="{ item }">
13.     <div class="name">{{ item.value }}</div>
14.     <span class="addr">{{ item.address }}</span>
15.   </template>
16. </el-autocomplete>
17.
18. <style>
19. .my-autocomplete {
20.   li {
21.     line-height: normal;
22.     padding: 7px;
23.
24.     .name {
25.       text-overflow: ellipsis;
26.       overflow: hidden;
27.     }
28.     .addr {
29.       font-size: 12px;
30.       color: #b4b4b4;
31.     }
32.
33.     .highlighted .addr {
34.       color: #ddd;
35.     }
36.   }
37. }
38. </style>
39.
40. <script>
41.   export default {
42.     data() {
43.       return {
44.         restaurants: [],
```

```

45.         state3: ''
46.     };
47. },
48. methods: {
49.     querySearch(queryString, cb) {
50.         var restaurants = this.restaurants;
51.         var results = queryString ?
52.             restaurants.filter(this.createFilter(queryString)) : restaurants;
53.             // 调用 callback 返回建议列表的数据
54.             cb(results);
55.         },
56.         createFilter(queryString) {
57.             return (restaurant) => {
58.                 return
59.                     (restaurant.value.toLowerCase().indexOf(queryString.toLowerCase()) === 0);
60.             };
61.         },
62.         loadAll() {
63.             return [
64.                 { "value": "三全鲜食（北新泾店）", "address": "长宁区新渔路144号" },
65.                 { "value": "Hot honey 首尔炸鸡（仙霞路）", "address": "上海市长宁区淞虹路
661号" },
66.                 { "value": "新旺角茶餐厅", "address": "上海市普陀区真北路988号创邑金沙谷6号
67. 楼113" },
68.                 { "value": "泷千家(天山西路店)", "address": "天山西路438号" },
69.                 { "value": "胖仙女纸杯蛋糕（上海凌空店）", "address": "上海市长宁区金钟路968
70. 号1幢18号楼一层商铺18-101" },
71.                 { "value": "贡茶", "address": "上海市长宁区金钟路633号" },
72.                 { "value": "豪大大香鸡排超级奶爸", "address": "上海市嘉定区曹安公路曹安路
73. 1685号" },
74.                 { "value": "茶芝兰（奶茶，手抓饼）", "address": "上海市普陀区同普路1435号"
75. },
76.                 { "value": "十二泷町", "address": "上海市北翟路1444弄81号B幢-107" },
77.                 { "value": "星移浓缩咖啡", "address": "上海市嘉定区新郁路817号" },
78.                 { "value": "阿姨奶茶/豪大大", "address": "嘉定区曹安路1611号" },
79.                 { "value": "新麦甜四季甜品炸鸡", "address": "嘉定区曹安公路2383弄55号" },
80.                 { "value": "Monica摩托主题咖啡店", "address": "嘉定区江桥镇曹安公路2409号
81. 1F, 2383弄62号1F" },
82.                 { "value": "浮生若茶（凌空soho店）", "address": "上海长宁区金钟路968号9号楼
83. 地下一层" },
84.                 { "value": "NONO JUICE 鲜榨果汁", "address": "上海市长宁区天山西路119号"
85. },

```

```
77.      { "value": "CoCo都可(北新泾店)", "address": "上海市长宁区仙霞西路" },
78.      { "value": "快乐柠檬(神州智慧店)", "address": "上海市长宁区天山西路567号1
    层R117号店铺" },
79.      { "value": "Merci Paul cafe", "address": "上海市普陀区光复西路丹巴路28弄6
    号楼819" },
80.      { "value": "猫山王(西郊百联店)", "address": "上海市长宁区仙霞西路88号第一层
    G05-F01-1-306" },
81.      { "value": "枪会山", "address": "上海市普陀区棕榈路" },
82.      { "value": "纵食", "address": "元丰天山花园(东门) 双流路267号" },
83.      { "value": "钱记", "address": "上海市长宁区天山西路" },
84.      { "value": "壹杯加", "address": "上海市长宁区通协路" },
85.      { "value": "妙哇滴咖", "address": "上海市长宁区新泾镇金钟路999号2幢(B幢) 第
    01层第1-02A单元" },
86.      { "value": "爱茜茜里(西郊百联)", "address": "长宁区仙霞西路88号1305室" },
87.      { "value": "爱茜茜里(近铁广场)", "address": "上海市普陀区真北路818号近铁城市
    广场北区地下二楼N-B2-02-C商铺" },
88.      { "value": "鲜果榨汁(金沙江路和美广店)", "address": "普陀区金沙江路2239号
    金沙和美广场B1-10-6" },
89.      { "value": "开心丽果(缤谷店)", "address": "上海市长宁区威宁路天山路341号"
    },
90.      { "value": "超级鸡车(丰庄路店)", "address": "上海市嘉定区丰庄路240号" },
91.      { "value": "妙生活果园(北新泾店)", "address": "长宁区新渔路144号" },
92.      { "value": "香宜度麻辣香锅", "address": "长宁区淞虹路148号" },
93.      { "value": "凡仔汉堡(老真北路店)", "address": "上海市普陀区老真北路160号"
    },
94.      { "value": "港式小铺", "address": "上海市长宁区金钟路968号15楼15-105室" },
95.      { "value": "蜀香源麻辣香锅(剑河路店)", "address": "剑河路443-1" },
96.      { "value": "北京饺子馆", "address": "长宁区北新泾街道天山西路490-1号" },
97.      { "value": "饭典*新简餐(凌空SOHO店)", "address": "上海市长宁区金钟路968号
    9号楼地下一层9-83室" },
98.      { "value": "焦耳·川式快餐(金钟路店)", "address": "上海市金钟路633号地下一
    层甲部" },
99.      { "value": "动力鸡车", "address": "长宁区仙霞西路299弄3号101B" },
100.     { "value": "浏阳蒸菜", "address": "天山西路430号" },
101.     { "value": "四海游龙(天山西路店)", "address": "上海市长宁区天山西路" },
102.     { "value": "樱花食堂(凌空店)", "address": "上海市长宁区金钟路968号15楼15-
    105室" },
103.     { "value": "壹分米客家传统调制米粉(天山店)", "address": "天山西路428号" },
104.     { "value": "福荣祥烧腊(平溪路店)", "address": "上海市长宁区协和路福泉路255
    弄57-73号" },
105.     { "value": "速记黄焖鸡米饭", "address": "上海市长宁区北新泾街道金钟路180号1
    层01号摊位" },
```

```

106.          { "value": "红辣椒麻辣烫", "address": "上海市长宁区天山西路492号" },
107.          { "value": "(小杨生煎)西郊百联餐厅", "address": "长宁区仙霞西路88号百联2楼"
  },
108.          { "value": "阳阳麻辣烫", "address": "天山西路389号" },
109.          { "value": "南拳妈妈龙虾盖浇饭", "address": "普陀区金沙江路1699号鑫乐惠美食
  广场A13" }
110.        ];
111.      },
112.      handleSelect(item) {
113.        console.log(item);
114.      },
115.      handleIconClick(ev) {
116.        console.log(ev);
117.      }
118.    },
119.    mounted() {
120.      this.restaurants = this.loadAll();
121.    }
122.  }
123. </script>

```

[显示代码](#)

远程搜索

从服务端搜索数据

```

1.  <el-autocomplete
2.    v-model="state4"
3.    :fetch-suggestions="querySearchAsync"
4.    placeholder="请输入内容"
5.    @select="handleSelect"
6.  ></el-autocomplete>
7.  <script>
8.    export default {
9.      data() {
10.        return {
11.          restaurants: [],
12.          state4: ''

```

```

13.      timeout: null
14.    };
15.  },
16.  methods: {
17.    loadAll() {
18.      return [
19.        { "value": "三全鲜食（北新泾店）", "address": "长宁区新渔路144号" },
20.        { "value": "Hot honey 首尔炸鸡（仙霞路）", "address": "上海市长宁区淞虹路
661号" },
21.        { "value": "新旺角茶餐厅", "address": "上海市普陀区真北路988号创邑金沙谷6号
楼113" },
22.        { "value": "泷千家(天山西路店)", "address": "天山西路438号" },
23.        { "value": "胖仙女纸杯蛋糕（上海凌空店）", "address": "上海市长宁区金钟路968
号1幢18号楼一层商铺18-101" },
24.        { "value": "贡茶", "address": "上海市长宁区金钟路633号" },
25.        { "value": "豪大大香鸡排超级奶爸", "address": "上海市嘉定区曹安公路曹安路
1685号" },
26.        { "value": "茶芝兰（奶茶，手抓饼）", "address": "上海市普陀区同普路1435号"
},
27.        { "value": "十二泷町", "address": "上海市北翟路1444弄81号B幢-107" },
28.        { "value": "星移浓缩咖啡", "address": "上海市嘉定区新郁路817号" },
29.        { "value": "阿姨奶茶/豪大大", "address": "嘉定区曹安路1611号" },
30.        { "value": "新麦甜四季甜品炸鸡", "address": "嘉定区曹安公路2383弄55号" },
31.        { "value": "Monica摩托主题咖啡店", "address": "嘉定区江桥镇曹安公路2409号
1F, 2383弄62号1F" },
32.        { "value": "浮生若茶（凌空soho店）", "address": "上海长宁区金钟路968号9号楼
地下一层" },
33.        { "value": "NONO JUICE 鲜榨果汁", "address": "上海市长宁区天山西路119号"
},
34.        { "value": "CoCo都可(北新泾店)", "address": "上海市长宁区仙霞西路" },
35.        { "value": "快乐柠檬（神州智慧店）", "address": "上海市长宁区天山西路567号1
层R117号店铺" },
36.        { "value": "Merci Paul cafe", "address": "上海市普陀区光复西路丹巴路28弄6
号楼819" },
37.        { "value": "猫山王（西郊百联店）", "address": "上海市长宁区仙霞西路88号第一层
G05-F01-1-306" },
38.        { "value": "枪会山", "address": "上海市普陀区棕榈路" },
39.        { "value": "纵食", "address": "元丰天山花园(东门) 双流路267号" },
40.        { "value": "钱记", "address": "上海市长宁区天山西路" },
41.        { "value": "壹杯加", "address": "上海市长宁区通协路" },
42.        { "value": "眇哇嘀咖", "address": "上海市长宁区新泾镇金钟路999号2幢（B幢）第
01层第1-02A单元" },

```

```
43.          { "value": "爱茜茜里(西郊百联)", "address": "长宁区仙霞西路88号1305室" },
44.          { "value": "爱茜茜里(近铁广场)", "address": "上海市普陀区真北路818号近铁城市
    广场北区地下二楼N-B2-02-C商铺" },
45.          { "value": "鲜果榨汁(金沙江路和美广店)", "address": "普陀区金沙江路2239号
    金沙和美广场B1-10-6" },
46.          { "value": "开心丽果(缤谷店)", "address": "上海市长宁区威宁路天山路341号"
    },
47.          { "value": "超级鸡车(丰庄路店)", "address": "上海市嘉定区丰庄路240号" },
48.          { "value": "妙生活果园(北新泾店)", "address": "长宁区新渔路144号" },
49.          { "value": "香宜度麻辣香锅", "address": "长宁区淞虹路148号" },
50.          { "value": "凡仔汉堡(老真北路店)", "address": "上海市普陀区老真北路160号"
    },
51.          { "value": "港式小铺", "address": "上海市长宁区金钟路968号15楼15-105室" },
52.          { "value": "蜀香源麻辣香锅(剑河路店)", "address": "剑河路443-1" },
53.          { "value": "北京饺子馆", "address": "长宁区北新泾街道天山西路490-1号" },
54.          { "value": "饭典*新简餐(凌空SOHO店)", "address": "上海市长宁区金钟路968号
    9号楼地下一层9-83室" },
55.          { "value": "焦耳·川式快餐(金钟路店)", "address": "上海市金钟路633号地下一
    层甲部" },
56.          { "value": "动力鸡车", "address": "长宁区仙霞西路299弄3号101B" },
57.          { "value": "浏阳蒸菜", "address": "天山西路430号" },
58.          { "value": "四海游龙(天山西路店)", "address": "上海市长宁区天山西路" },
59.          { "value": "樱花食堂(凌空店)", "address": "上海市长宁区金钟路968号15楼15-
    105室" },
60.          { "value": "壹分米客家传统调制米粉(天山店)", "address": "天山西路428号" },
61.          { "value": "福荣祥烧腊(平溪路店)", "address": "上海市长宁区协和路福泉路255
    弄57-73号" },
62.          { "value": "速记黄焖鸡米饭", "address": "上海市长宁区北新泾街道金钟路180号1
    层01号摊位" },
63.          { "value": "红辣椒麻辣烫", "address": "上海市长宁区天山西路492号" },
64.          { "value": "(小杨生煎)西郊百联餐厅", "address": "长宁区仙霞西路88号百联2楼"
    },
65.          { "value": "阳阳麻辣烫", "address": "天山西路389号" },
66.          { "value": "南拳妈妈龙虾盖浇饭", "address": "普陀区金沙江路1699号鑫乐惠美食
    广场A13" }
67.      ];
68.  },
69.  querySearchAsync(queryString, cb) {
70.      var restaurants = this.restaurants;
71.      var results = queryString ?
    restaurants.filter(this.createStateFilter(queryString)) : restaurants;
72.  }
```

```

73.     clearTimeout(this.timeout);
74.     this.timeout = setTimeout(() => {
75.       cb(results);
76.     }, 3000 * Math.random());
77.   },
78.   createStateFilter(queryString) {
79.     return (state) => {
80.       return (state.value.toLowerCase()).indexOf(queryString.toLowerCase()) === 0;
81.     };
82.   },
83.   handleSelect(item) {
84.     console.log(item);
85.   }
86. },
87. mounted() {
88.   this.restaurants = this.loadAll();
89. }
90. );
91. </script>

```

Input Attributes

参数	说明	类型	可选值	默认值
type	类型	string	text, textarea 和其他 原生 input 的 type 值	text
value	绑定值	string / number	—	—
maxlength	原生属性，最大输入长度	number	—	—
minlength	原生属性，最小输入长度	number	—	—
placeholder	输入框占位文本	string	—	—
clearable	是否可清空	boolean	—	false
disabled	禁用	boolean	—	false
size	输入框尺寸，只在 <code>type!="textarea"</code> 时有效	string	medium / small / mini	—
prefix-icon	输入框头部图标	string	—	—
suffix-icon	输入框尾部图标	string	—	—
rows	输入框行数，只对 <code>type="textarea"</code> 有效	number	—	2
autosize	自适应内容高度。只对 <code>type="textarea"</code> 有效，可传入对象，	boolean /	—	false

	如, { minRows: 2, maxRows: 6 }	object		
auto-complete	原生属性, 自动补全	string	on, off	off
name	原生属性	string	-	-
readonly	原生属性, 是否只读	boolean	-	false
max	原生属性, 设置最大值	-	-	-
min	原生属性, 设置最小值	-	-	-
step	原生属性, 设置输入字段的合法数字间隔	-	-	-
resize	控制是否能被用户缩放	string	none, both, horizontal, vertical	-
autofocus	原生属性, 自动获取焦点	boolean	true, false	false
form	原生属性	string	-	-
label	输入框关联的label文字	string	-	-
tabindex	输入框的tabindex	string	-	-

Input Slots

name	说明
prefix	输入框头部内容, 只对 <code>type="text"</code> 有效
suffix	输入框尾部内容, 只对 <code>type="text"</code> 有效
prepend	输入框前置内容, 只对 <code>type="text"</code> 有效
append	输入框后置内容, 只对 <code>type="text"</code> 有效

Input Events

事件名称	说明	回调参数
blur	在 Input 失去焦点时触发	(event: Event)
focus	在 Input 获得焦点时触发	(event: Event)
change	在 Input 值改变时触发	(value: string)
clear	在点击由 <code>clearable</code> 属性生成的清空按钮时触发	number

Input Methods

方法名	说明	参数
focus	使 input 获取焦点	-
blur	使 input 失去焦点	-
select	选中 input 中的文字	-

Autocomplete Attributes

参数	说明	类型	可选值	默认值
placeholder	输入框占位文本	string	-	-
disabled	禁用	boolean	-	false
value-key	输入建议对象中用于显示的键名	string	-	value
value	必填值，输入绑定值	string	-	-
debounce	获取输入建议的去抖延时	number	-	300
placement	菜单弹出位置	string	top / top-start / top-end / bottom / bottom-start / bottom-end	bottom-start
fetch-suggestions	返回输入建议的方法，仅当你的输入建议数据 resolve 时，通过调用 callback(data:[]) 来返回它	Function(queryString, callback)	-	-
popper-class	Autocomplete 下拉列表的类名	string	-	-
trigger-on-focus	是否在输入框 focus 时显示建议列表	boolean	-	true
name	原生属性	string	-	-
select-when-unmatched	在输入没有任何匹配建议的情况下，按下回车是否触发 select 事件	boolean	-	false
label	输入框关联的label文字	string	-	-
prefix-icon	输入框头部图标	string	-	-
suffix-icon	输入框尾部图标	string	-	-
hide-loading	是否隐藏远程加载时的加载图标	boolean	-	false
popper-append-to-body	是否将下拉列表插入至 body 元素。在下拉列表的定位出现问题时，可将该属性设置为 false	boolean	-	true

Autocomplete Slots

name	说明
prefix	输入框头部内容
suffix	输入框尾部内容
prepend	输入框前置内容
append	输入框后置内容

Autocomplete Scoped Slot

name	说明
-	自定义输入建议，参数为 { item }

Autocomplete Events

事件名称	说明	回调参数
select	点击选中建议项时触发	选中建议项

Autocomplete Methods

方法名	说明	参数
focus	使 input 获得焦点	-

原文: <http://element-cn.eleme.io/#/zh-CN/component/input>



InputNumber 计数器

InputNumber 计数器

仅允许输入标准的数字值，可定义范围

基础用法

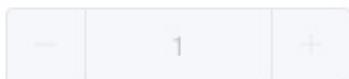


要使用它，只需要在 `el-input-number` 元素中使用 `v-model` 绑定变量即可，变量的初始值即为默认值。

```
1. <template>
2.   <el-input-number v-model="num1" @change="handleChange" :min="1" :max="10"
3.     label="描述文字"></el-input-number>
4. </template>
5. <script>
6.   export default {
7.     data() {
8.       return {
9.         num1: 1
10.      },
11.      methods: {
12.        handleChange(value) {
13.          console.log(value);
14.        }
15.      }
16.    };
17.  </script>
```

显示代码

禁用状态



`disabled` 属性接受一个 `Boolean`，设置为 `true` 即可禁用整个组件，如果你只需要控制数值在某一范围内，可以设置 `min` 属性和 `max` 属性，不设置 `min` 和 `max` 时，最小值为 0。

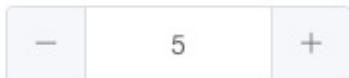
```

1. <template>
2.   <el-input-number v-model="num2" :disabled="true"></el-input-number>
3. </template>
4. <script>
5.   export default {
6.     data() {
7.       return {
8.         num2: 1
9.       }
10.    }
11.  };
12. </script>

```

步数

允许定义递增递减的步数控制



设置 `step` 属性可以控制步长，接受一个 `Number`。

```

1. <template>
2.   <el-input-number v-model="num3" :step="2"></el-input-number>
3. </template>
4. <script>
5.   export default {
6.     data() {
7.       return {
8.         num3: 5
9.       }
10.    }
11.  };
12. </script>

```

精度



设置 `precision` 属性可以控制数值精度，接收一个 `Number`。

```

1. <template>
2.   <el-input-number v-model="num9" :precision="2" :step="0.1" :max="10"></el-
   input-number>
3. </template>
4. <script>
5.   export default {
6.     data() {
7.       return {
8.         num9: 1
9.       }
10.    }
11.  };
12. </script>

```

显示代码

`precision` 的值必须是一个非负整数，并且不能小于 `step` 的小数位数。

尺寸

额外提供了 `medium`、`small`、`mini` 三种尺寸的数字输入框



```

1. <template>
2.   <el-input-number v-model="num4"></el-input-number>
3.   <el-input-number size="medium" v-model="num5"></el-input-number>
4.   <el-input-number size="small" v-model="num6"></el-input-number>
5.   <el-input-number size="mini" v-model="num7"></el-input-number>
6. </template>
7. <script>
8.   export default {
9.     data() {
10.       return {
11.         num4: 1,

```

```

12.         num5: 1,
13.         num6: 1,
14.         num7: 1
15.     }
16. }
17. };
18. </script>

```

[显示代码](#)

按钮位置



设置 `controls-position` 属性可以控制按钮位置。

```

1. <template>
2.   <el-input-number v-model="num8" controls-position="right"
3.     @change="handleChange" :min="1" :max="10"></el-input-number>
4. </template>
5. <script>
6.   export default {
7.     data() {
8.       return {
9.         num8: 1
10.      },
11.      methods: {
12.        handleChange(value) {
13.          console.log(value);
14.        }
15.      }
16.    };
17. </script>

```

[显示代码](#)

Attributes

参数	说明	类型	可选值	默认值
<code>value</code>	绑定值	<code>number</code>	-	-

min	设置计数器允许的最小值	number	-	-Infinity
max	设置计数器允许的最大值	number	-	Infinity
step	计数器步长	number	-	1
precision	数值精度	number	-	-
size	计数器尺寸	string	large, small	-
disabled	是否禁用计数器	boolean	-	false
controls	是否使用控制按钮	boolean	-	true
controls-position	控制按钮位置	string	right	-
name	原生属性	string	-	-
label	输入框关联的label文字	string	-	-

Events

事件名称	说明	回调参数
change	绑定值被改变时触发	最后变更的值
blur	在组件 Input 失去焦点时触发	(event: Event)
focus	在组件 Input 获得焦点时触发	(event: Event)

Methods

方法名	说明	参数
focus	使 input 获得焦点	-

原文: <http://element-cn.eleme.io/#/zh-CN/component/input-number>

Select 选择器

Select 选择器

当选项过多时，使用下拉菜单展示并选择内容。

基础用法

适用广泛的基础单选



`v-model` 的值为当前被选中的 `el-option` 的 `value` 属性值

```
1. <template>
2.   <el-select v-model="value" placeholder="请选择">
3.     <el-option
4.       v-for="item in options"
5.       :key="item.value"
6.       :label="item.label"
7.       :value="item.value">
8.     </el-option>
9.   </el-select>
10. </template>
11.
12. <script>
13.   export default {
14.     data() {
15.       return {
16.         options: [
17.           { value: '选项1', label: '黄金糕' },
18.           { value: '选项2', label: '双皮奶' },
19.           { value: '选项3', label: '蚵仔煎' },
20.           { value: '选项4', label: '龙须面' },
21.           { value: '选项5', label: '北京烤鸭' }
22.         ]
23.       }
24.     }
25.   }
26. </script>
```

```

19.      },
20.      value: '选项2',
21.      label: '双皮奶'
22.    },
23.    value: '选项3',
24.    label: '蚵仔煎'
25.  },
26.  value: '选项4',
27.  label: '龙须面'
28. },
29. value: '选项5',
30. label: '北京烤鸭'
31. ],
32. value: ''
33. }
34. }
35. }
36. </script>

```

[显示代码](#)

有禁用选项



在 `el-option` 中，设定 `disabled` 值为 `true`，即可禁用该选项

```

1. <template>
2.   <el-select v-model="value2" placeholder="请选择">
3.     <el-option
4.       v-for="item in options2"
5.       :key="item.value"
6.       :label="item.label"

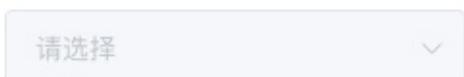
```

```
7.      :value="item.value"
8.      :disabled="item.disabled">
9.    </el-option>
10.   </el-select>
11. </template>
12.
13. <script>
14. export default {
15.   data() {
16.     return {
17.       options2: [
18.         { value: '选项1', label: '黄金糕' },
19.         { value: '选项2', label: '双皮奶', disabled: true },
20.         { value: '选项3', label: '蚵仔煎' },
21.         { value: '选项4', label: '龙须面' },
22.         { value: '选项5', label: '北京烤鸭' }
23.       ],
24.       value2: ''
25.     }
26.   }
27. }
28. 
```

显示代码

禁用状态

选择器不可用状态



为 `el-select` 设置 `disabled` 属性，则整个选择器不可用

```
1. <template>
2.   <el-select v-model="value3" disabled placeholder="请选择">
3.     <el-option
4.       v-for="item in options"
5.         :key="item.value"
6.         :label="item.label"
7.         :value="item.value">
8.       </el-option>
9.     </el-select>
10.   </template>
11.
12. <script>
13.   export default {
14.     data() {
15.       return {
16.         options: [
17.           { value: '选项1', label: '黄金糕' },
18.           { value: '选项2', label: '双皮奶' },
19.           { value: '选项3', label: '蚵仔煎' },
20.           { value: '选项4', label: '龙须面' },
21.           { value: '选项5', label: '北京烤鸭' }
22.         ],
23.         value3: ''
24.       }
25.     }
26.   }
27. }
28. </script>
```

显示代码

可清空单选

包含清空按钮，可将选择器清空为初始状态



为 `el-select` 设置 `clearable` 属性，则可将选择器清空。需要注意的是，`clearable` 属性仅适用于单选。

```
1. <template>
2.   <el-select v-model="value4" clearable placeholder="请选择">
3.     <el-option
4.       v-for="item in options"
5.         :key="item.value"
6.         :label="item.label"
7.         :value="item.value">
8.     </el-option>
9.   </el-select>
10. </template>
11.
12. <script>
13. export default {
14.   data() {
15.     return {
16.       options: [
17.         { value: '选项1', label: '黄金糕' },
18.         { value: '选项2', label: '双皮奶' },
19.         { value: '选项3', label: '蚵仔煎' },
20.         { value: '选项4', label: '龙须面' }
21.       ]
22.     }
23.   }
24. }
```

```

27.         label: '龙须面'
28.     }, {
29.         value: '选项5',
30.         label: '北京烤鸭'
31.     }],
32.         value4: ''
33.     }
34. }
35. }
36. </script>

```

基础多选

适用性较广的基础多选，用 Tag 展示已选项



为 `el-select` 设置 `multiple` 属性即可启用多选，此时 `v-model` 的值为当前选中值所组成的数组。默认情况下选中值会以 Tag 的形式展现，你也可以设置 `collapse-tags` 属性将它们合并为一段文字。

```

1.  <template>
2.    <el-select v-model="value5" multiple placeholder="请选择">
3.      <el-option
4.        v-for="item in options"
5.        :key="item.value"
6.        :label="item.label"
7.        :value="item.value">
8.      </el-option>
9.    </el-select>
10.
11.   <el-select
12.     v-model="value11">

```

```
13.     multiple
14.     collapse-tags
15.     style="margin-left: 20px;"
16.     placeholder="请选择">
17.     <el-option
18.       v-for="item in options"
19.         :key="item.value"
20.         :label="item.label"
21.         :value="item.value">
22.       </el-option>
23.     </el-select>
24.   </template>
25.
26. <script>
27.   export default {
28.     data() {
29.       return {
30.         options: [
31.           { value: '选项1', label: '黄金糕' },
32.           { value: '选项2', label: '双皮奶' },
33.           { value: '选项3', label: '蚵仔煎' },
34.           { value: '选项4', label: '龙须面' },
35.           { value: '选项5', label: '北京烤鸭' }
36.         ],
37.         value5: [],
38.         value11: []
39.       }
40.     },
41.     value5: [],
42.     value11: []
43.   }
44. }
45. }
46. }
47. }
48. }
49. }
50. }
51. </script>
```

自定义模板

可以自定义备选项



将自定义的 HTML 模板插入 `el-option` 的 slot 中即可。

```
1. <template>
2.   <el-select v-model="value6" placeholder="请选择">
3.     <el-option
4.       v-for="item in cities"
5.         :key="item.value"
6.         :label="item.label"
7.         :value="item.value">
8.           <span style="float: left">{{ item.label }}</span>
9.           <span style="float: right; color: #8492a6; font-size: 13px">{{ item.value }}</span>
10.        </el-option>
11.      </el-select>
12.    </template>
13.
14. <script>
15.   export default {
16.     data() {
17.       return {
18.         cities: [
19.           { value: 'Beijing', label: '北京' },
20.           { value: 'Shanghai', label: '上海' },
21.           { value: 'Nanjing', label: '南京' },
22.           { value: 'Chengdu', label: '成都' },
23.           { value: 'Shenzhen', label: '深圳' },
24.           { value: 'Guangzhou', label: '广州' }
25.         ]
26.       }
27.     }
28.   }
29. </script>
```

```

26.           label: '南京'
27.         },
28.         value: 'Chengdu',
29.         label: '成都'
30.       },
31.       value: 'Shenzhen',
32.       label: '深圳'
33.     },
34.     value: 'Guangzhou',
35.     label: '广州'
36.   ],
37.   value6: ''
38. }
39. }
40. }
41. </script>

```

分组

对备选项进行分组展示



使用 `el-option-group` 对备选项进行分组，它的 `label` 属性为分组名

```

1. <template>
2.   <el-select v-model="value7" placeholder="请选择">
3.     <el-option-group
4.       v-for="group in options3"

```

```
5.      :key="group.label"
6.      :label="group.label">
7.      <el-option
8.          v-for="item in group.options"
9.          :key="item.value"
10.         :label="item.label"
11.         :value="item.value">
12.     </el-option>
13.   </el-option-group>
14. </el-select>
15. </template>
16.
17. <script>
18. export default {
19.   data() {
20.     return {
21.       options3: [{{
22.         label: '热门城市',
23.         options: [{{
24.           value: 'Shanghai',
25.           label: '上海'
26.         }, {{
27.           value: 'Beijing',
28.           label: '北京'
29.         }]}
30.       }, {
31.         label: '城市名',
32.         options: [{{
33.           value: 'Chengdu',
34.           label: '成都'
35.         }, {{
36.           value: 'Shenzhen',
37.           label: '深圳'
38.         }, {{
39.           value: 'Guangzhou',
40.           label: '广州'
41.         }, {{
42.           value: 'Dalian',
43.           label: '大连'
44.         }]}
45.       }, ],
46.       value7: ''
```

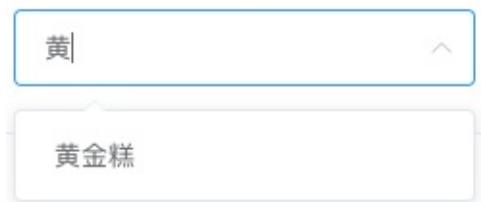
```

47.      }
48.    }
49.  }
50. </script>

```

可搜索

可以利用搜索功能快速查找选项



为 `el-select` 添加 `filterable` 属性即可启用搜索功能。默认情况下，Select 会找出所有 `label` 属性包含输入值的选项。如果希望使用其他的搜索逻辑，可以通过传入一个 `filter-method` 来实现。`filter-method` 为一个 `Function`，它会在输入值发生变化时调用，参数为当前输入值。

```

1. <template>
2.   <el-select v-model="value8" filterable placeholder="请选择">
3.     <el-option
4.       v-for="item in options"
5.         :key="item.value"
6.         :label="item.label"
7.         :value="item.value">
8.     </el-option>
9.   </el-select>
10. </template>
11.
12. <script>
13.   export default {
14.     data() {
15.       return {
16.         options: [
17.           { value: '选项1', label: '黄金糕' },
18.           { value: '选项2', label: '双皮奶' },
19.           { value: '选项3', label: '红豆饼' }
20.         ]
21.       }
22.     }
23.   }

```

```

24.           label: '蚵仔煎'
25.     },
26.     value: '选项4',
27.     label: '龙须面'
28.   },
29.   value: '选项5',
30.   label: '北京烤鸭'
31. ],
32. value8: ''
33. }
34. }
35. }
36. </script>

```

[显示代码](#)

远程搜索

从服务器搜索数据，输入关键字进行查找



请输入关键词

为了启用远程搜索，需要将 `filterable` 和 `remote` 设置为 `true`，同时传入一个 `remote-method`。`remote-method` 为一个 `Function`，它会在输入值发生变化时调用，参数为当前输入值。需要注意的是，如果 `el-option` 是通过 `v-for` 指令渲染出来的，此时需要为 `el-option` 添加 `key` 属性，且其值需具有唯一性，比如此例中的 `item.value`。

```

1. <template>
2.   <el-select
3.     v-model="value9"
4.     multiple
5.     filterable
6.     remote
7.     reserve-keyword
8.     placeholder="请输入关键词"
9.     :remote-method="remoteMethod"
10.    :loading="loading">
11.      <el-option
12.        v-for="item in options4"
13.        :key="item.value"
14.        :label="item.label">

```

```
15.      :value="item.value">
16.    </el-option>
17.  </el-select>
18. </template>
19.
20. <script>
21.   export default {
22.     data() {
23.       return {
24.         options4: [],
25.         value9: [],
26.         list: [],
27.         loading: false,
28.         states: ["Alabama", "Alaska", "Arizona",
29.           "Arkansas", "California", "Colorado",
30.           "Connecticut", "Delaware", "Florida",
31.           "Georgia", "Hawaii", "Idaho", "Illinois",
32.           "Indiana", "Iowa", "Kansas", "Kentucky",
33.           "Louisiana", "Maine", "Maryland",
34.           "Massachusetts", "Michigan", "Minnesota",
35.           "Mississippi", "Missouri", "Montana",
36.           "Nebraska", "Nevada", "New Hampshire",
37.           "New Jersey", "New Mexico", "New York",
38.           "North Carolina", "North Dakota", "Ohio",
39.           "Oklahoma", "Oregon", "Pennsylvania",
40.           "Rhode Island", "South Carolina",
41.           "South Dakota", "Tennessee", "Texas",
42.           "Utah", "Vermont", "Virginia",
43.           "Washington", "West Virginia", "Wisconsin",
44.           "Wyoming"]
45.       }
46.     },
47.     mounted() {
48.       this.list = this.states.map(item => {
49.         return { value: item, label: item };
50.       });
51.     },
52.     methods: {
53.       remoteMethod(query) {
54.         if (query !== '') {
55.           this.loading = true;
56.           setTimeout(() => {
```

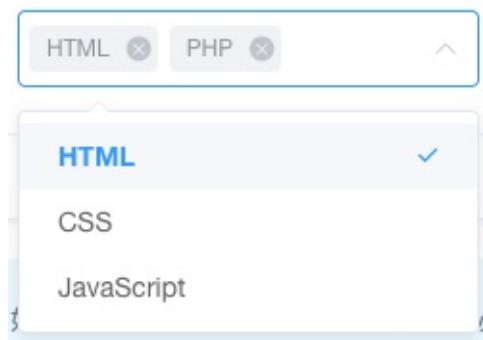
```

57.         this.loading = false;
58.         this.options4 = this.list.filter(item => {
59.             return item.label.toLowerCase()
60.                 .indexOf(query.toLowerCase()) > -1;
61.         });
62.     }, 200);
63. } else {
64.     this.options4 = [];
65. }
66. }
67. }
68. }
69. </script>

```

创建条目

可以创建并选中选项中不存在的条目



使用 `allow-create` 属性即可通过在输入框中输入文字来创建新的条目。注意此时 `filterable` 必须为真。本例还使用了 `default-first-option` 属性，在该属性打开的情况下，按下回车就可以选中当前选项列表中的第一个选项，无需使用鼠标或键盘方向键进行定位。

```

1. <template>
2.   <el-select
3.     v-model="value10"
4.     multiple
5.     filterable
6.     allow-create
7.     default-first-option
8.     placeholder="请选择文章标签">
9.     <el-option
10.       v-for="item in options5"
11.       :key="item.value"

```

```

12.      :label="item.label"
13.      :value="item.value">
14.    </el-option>
15.  </el-select>
16. </template>
17.
18. <script>
19.   export default {
20.     data() {
21.       return {
22.         options5: [
23.           { value: 'HTML', label: 'HTML' },
24.           { value: 'CSS', label: 'CSS' },
25.           { value: 'JavaScript', label: 'JavaScript' }
26.         ],
27.         value10: []
28.       }
29.     }
30.   }
31. }
32. }
33. }
34. }
35. }
36. </script>

```

[显示代码](#)

如果 Select 的绑定值为对象类型，请务必指定 **value-key** 作为它的唯一性标识。

Select Attributes

参数	说明	类型	可选值	默认值
multiple	是否多选	boolean	—	false
disabled	是否禁用	boolean	—	false
value-key	作为 value 唯一标识的键名，绑定值为对象类型时必填	string	—	value
size	输入框尺寸	string	medium/small/mini	—
clearable	单选时是否可以清空选项	boolean	—	false
collapse-tags	多选时是否将选中值按文字的形式展示	boolean	—	false
multiple-limit	多选时用户最多可以选择的项目数，为 0 则不限制	number	—	0

name	select input 的 name 属性	string	—	—
auto-complete	select input 的 autocomplete 属性	string	—	off
placeholder	占位符	string	—	请选择
filterable	是否可搜索	boolean	—	false
allow-create	是否允许用户创建新条目，需配合 filterable 使用	boolean	—	false
filter-method	自定义搜索方法	function	—	—
remote	是否为远程搜索	boolean	—	false
remote-method	远程搜索方法	function	—	—
loading	是否正在从远程获取数据	boolean	—	false
loading-text	远程加载时显示的文字	string	—	加载中
no-match-text	搜索条件无匹配时显示的文字	string	—	无匹配数据
no-data-text	选项为空时显示的文字	string	—	无数据
popper-class	Select 下拉框的类名	string	—	—
reserve-keyword	多选且可搜索时，是否在选中一个选项后保留当前的搜索关键词	boolean	—	false
default-first-option	在输入框按下回车，选择第一个匹配项。需配合 filterable 或 remote 使用	boolean	-	false
popper-append-to-body	是否将弹出框插入至 body 元素。在弹出框的定位出现问题时，可将该属性设置为 false	boolean	-	true
automatic-dropdown	对于不可搜索的 Select，是否在输入框获得焦点后自动弹出选项菜单	boolean	-	false

Select Events

事件名称	说明	回调参数
change	选中值发生变化时触发	目前的选中值
visible-change	下拉框出现/隐藏时触发	出现则为 true，隐藏则为 false
remove-tag	多选模式下移除tag时触发	移除的tag值
clear	可清空的单选模式下用户点击清空按钮时触发	—
blur	当 input 失去焦点时触发	(event: Event)
focus	当 input 获得焦点时触发	(event: Event)

Select Slots

name	说明
-	Option 组件列表
prefix	Select 组件头部内容

Option Group Attributes

参数	说明	类型	可选值	默认值
label	分组的组名	string	-	-
disabled	是否将该分组下所有选项置为禁用	boolean	-	false

Option Attributes

参数	说明	类型	可选值	默认值
value	选项的值	string/number/object	-	-
label	选项的标签，若不设置则默认与 value 相同	string/number	-	-
disabled	是否禁用该选项	boolean	-	false

Methods

方法名	说明	参数
focus	使 input 获取焦点	-
blur	使 input 失去焦点，并隐藏下拉框	-

原文: <http://element-cn.eleme.io/#/zh-CN/component/select>

Cascader 级联选择器

Cascader 级联选择器

当一个数据集合有清晰的层级结构时，可通过级联选择器逐级查看并选择。

基础用法

有两种触发子菜单的方式



只需为 Cascader 的 `options` 属性指定选项数组即可渲染出一个级联选择器。通过 `expand-trigger` 可以定义展开子级菜单的触发方式。本例还展示了 `change` 事件，它的参数为 Cascader 的绑定值：一个由各级菜单的值所组成的数组。

```

1.  <div class="block">
2.    <span class="demonstration">默认 click 触发子菜单</span>
3.    <el-cascader
4.      :options="options"
5.      v-model="selectedOptions"
6.      @change="handleChange">
7.    </el-cascader>
8.  </div>
9.  <div class="block">
10.   <span class="demonstration">hover 触发子菜单</span>
11.   <el-cascader
12.     expand-trigger="hover">

```

```
13.      :options="options"
14.      v-model="selectedOptions2"
15.      @change="handleChange">
16.    </el-cascader>
17.  </div>
18.
19. <script>
20.   export default {
21.     data() {
22.       return {
23.         options: [
24.           {
25.             value: 'zhinan',
26.             label: '指南',
27.             children: [
28.               {
29.                 value: 'shejiyuanze',
30.                 label: '设计原则',
31.                 children: [
32.                   {
33.                     value: 'yizhi',
34.                     label: '一致'
35.                   },
36.                   {
37.                     value: 'fankui',
38.                     label: '反馈'
39.                   },
39.                   {
40.                     value: 'xiaolv',
41.                     label: '效率'
42.                   },
43.                   {
44.                     value: 'kekong',
45.                     label: '可控'
46.                   }
47.                 ]
48.               },
49.               {
50.                 value: 'daohang',
51.                 label: '导航',
52.                 children: [
53.                   {
54.                     value: 'cexiangdaohang',
55.                     label: '侧向导航'
56.                   },
57.                   {
58.                     value: 'dingbudaohang',
59.                     label: '顶部导航'
60.                   }
61.                 ]
62.               }
63.             ],
64.             value: 'zujian',
```

```
55.         label: '组件',
56.         children: [{  
57.             value: 'basic',  
58.             label: 'Basic',  
59.             children: [{  
60.                 value: 'layout',  
61.                 label: 'Layout 布局'  
62.             }, {  
63.                 value: 'color',  
64.                 label: 'Color 色彩'  
65.             }, {  
66.                 value: 'typography',  
67.                 label: 'Typography 字体'  
68.             }, {  
69.                 value: 'icon',  
70.                 label: 'Icon 图标'  
71.             }, {  
72.                 value: 'button',  
73.                 label: 'Button 按钮'  
74.             }]  
75.         }, {  
76.             value: 'form',  
77.             label: 'Form',  
78.             children: [{  
79.                 value: 'radio',  
80.                 label: 'Radio 单选框'  
81.             }, {  
82.                 value: 'checkbox',  
83.                 label: 'Checkbox 多选框'  
84.             }, {  
85.                 value: 'input',  
86.                 label: 'Input 输入框'  
87.             }, {  
88.                 value: 'input-number',  
89.                 label: 'InputNumber 计数器'  
90.             }, {  
91.                 value: 'select',  
92.                 label: 'Select 选择器'  
93.             }, {  
94.                 value: 'cascader',  
95.                 label: 'Cascader 级联选择器'  
96.             }]
```

```
97.          value: 'switch',
98.          label: 'Switch 开关'
99.        },
100.       {
101.         value: 'slider',
102.         label: 'Slider 滑块'
103.       },
104.       {
105.         value: 'time-picker',
106.         label: 'TimePicker 时间选择器'
107.       },
108.       {
109.         value: 'date-picker',
110.         label: 'DatePicker 日期选择器'
111.       },
112.       {
113.         value: 'datetime-picker',
114.         label: 'DateTimePicker 日期时间选择器'
115.       },
116.       {
117.         value: 'upload',
118.         label: 'Upload 上传'
119.       },
120.     ],
121.   },
122.   {
123.     value: 'data',
124.     label: 'Data',
125.     children: [
126.       {
127.         value: 'table',
128.         label: 'Table 表格'
129.       },
130.       {
131.         value: 'tag',
132.         label: 'Tag 标签'
133.       },
134.       {
135.         value: 'progress',
136.         label: 'Progress 进度条'
137.       },
138.       {
139.         value: 'tree',
140.         label: 'Tree 树形控件'
141.       },
142.       {
143.         value: 'pagination',
144.         label: 'Pagination 分页'
145.       }
146.     ]
147.   }
148. }
```

```
139.     }, {
140.         value: 'badge',
141.         label: 'Badge 标记'
142.     }]
143. },
144.     value: 'notice',
145.     label: 'Notice',
146.     children: [{
147.         value: 'alert',
148.         label: 'Alert 警告'
149.     }, {
150.         value: 'loading',
151.         label: 'Loading 加载'
152.     }, {
153.         value: 'message',
154.         label: 'Message 消息提示'
155.     }, {
156.         value: 'message-box',
157.         label: 'MessageBox 弹框'
158.     }, {
159.         value: 'notification',
160.         label: 'Notification 通知'
161.     }]
162. },
163.     value: 'navigation',
164.     label: 'Navigation',
165.     children: [{
166.         value: 'menu',
167.         label: 'NavMenu 导航菜单'
168.     }, {
169.         value: 'tabs',
170.         label: 'Tabs 标签页'
171.     }, {
172.         value: 'breadcrumb',
173.         label: 'Breadcrumb 面包屑'
174.     }, {
175.         value: 'dropdown',
176.         label: 'Dropdown 下拉菜单'
177.     }, {
178.         value: 'steps',
179.         label: 'Steps 步骤条'
180.     }]
}
```

```
181.     },
182.     value: 'others',
183.     label: 'Others',
184.     children: [
185.       {
186.         value: 'dialog',
187.         label: 'Dialog 对话框'
188.       },
189.       {
190.         value: 'tooltip',
191.         label: 'Tooltip 文字提示'
192.       },
193.       {
194.         value: 'popover',
195.         label: 'Popover 弹出框'
196.       },
197.       {
198.         value: 'card',
199.         label: 'Card 卡片'
200.       },
201.       {
202.         value: 'collapse',
203.         label: 'Collapse 折叠面板'
204.       }
205.     ],
206.   },
207.   {
208.     value: 'ziyuan',
209.     label: '资源',
210.     children: [
211.       {
212.         value: 'axure',
213.         label: 'Axure Components'
214.       },
215.       {
216.         value: 'sketch',
217.         label: 'Sketch Templates'
218.       },
219.       {
220.         value: 'jiaohu',
221.         label: '组件交互文档'
222.       }
223.     ],
224.   },
225.   selectedOptions: [],
226.   selectedOptions2: []
227. );
228. },
229. methods: {
```

```

223.     handleChange(value) {
224.       console.log(value);
225.     }
226.   }
227. };
228. </script>

```

禁用选项

通过在数据源中设置 `disabled` 字段来声明该选项是禁用的



本例中，`options` 指定的数组中的第一个元素含有 `disabled: true` 键值对，因此是禁用的。在默认情况下，Cascader 会检查数据中每一项的 `disabled` 字段是否为 `true`，如果你的数据中表示禁用含义的字段名不为 `disabled`，可以通过 `props` 属性来指定（详见下方 API 表格）。当然，`value`、`label` 和 `children` 这三个字段名也可以通过同样的方式指定。

```

1. <el-cascader
2.   :options="optionsWithDisabled"
3. ></el-cascader>
4. <script>
5.   export default {
6.     data() {
7.       return {
8.         optionsWithDisabled: [
9.           {
10.             value: 'zhinan',
11.             label: '指南',
12.             disabled: true,
13.             children: [
14.               {

```

```
15.          children: [{  
16.              value: 'yizhi',  
17.              label: '一致'  
18.          }, {  
19.              value: 'fankui',  
20.              label: '反馈'  
21.          }, {  
22.              value: 'xiaolv',  
23.              label: '效率'  
24.          }, {  
25.              value: 'kekong',  
26.              label: '可控'  
27.          }]  
28.      }, {  
29.          value: 'daohang',  
30.          label: '导航',  
31.          children: [{  
32.              value: 'cexiangdaohang',  
33.              label: '侧向导航'  
34.          }, {  
35.              value: 'dingbudaohang',  
36.              label: '顶部导航'  
37.          }]  
38.      }]  
39.  }, {  
40.      value: 'zujian',  
41.      label: '组件',  
42.      children: [{  
43.          value: 'basic',  
44.          label: 'Basic',  
45.          children: [{  
46.              value: 'layout',  
47.              label: 'Layout 布局'  
48.          }, {  
49.              value: 'color',  
50.              label: 'Color 色彩'  
51.          }, {  
52.              value: 'typography',  
53.              label: 'Typography 字体'  
54.          }, {  
55.              value: 'icon',  
56.              label: 'Icon 图标'
```

```
57.      }, {
58.        value: 'button',
59.        label: 'Button 按钮'
60.      }]
61.    }, {
62.      value: 'form',
63.      label: 'Form',
64.      children: [{
65.        value: 'radio',
66.        label: 'Radio 单选框'
67.      }, {
68.        value: 'checkbox',
69.        label: 'Checkbox 多选框'
70.      }, {
71.        value: 'input',
72.        label: 'Input 输入框'
73.      }, {
74.        value: 'input-number',
75.        label: 'InputNumber 计数器'
76.      }, {
77.        value: 'select',
78.        label: 'Select 选择器'
79.      }, {
80.        value: 'cascader',
81.        label: 'Cascader 级联选择器'
82.      }, {
83.        value: 'switch',
84.        label: 'Switch 开关'
85.      }, {
86.        value: 'slider',
87.        label: 'Slider 滑块'
88.      }, {
89.        value: 'time-picker',
90.        label: 'TimePicker 时间选择器'
91.      }, {
92.        value: 'date-picker',
93.        label: 'DatePicker 日期选择器'
94.      }, {
95.        value: 'datetime-picker',
96.        label: 'DateTimePicker 日期时间选择器'
97.      }, {
98.        value: 'upload',
```

```
99.          label: 'Upload 上传'
100.        },
101.        value: 'rate',
102.        label: 'Rate 评分'
103.      },
104.      value: 'form',
105.      label: 'Form 表单'
106.    }]
107.  },
108.  value: 'data',
109.  label: 'Data',
110.  children: [{{
111.    value: 'table',
112.    label: 'Table 表格'
113.  },
114.  {
115.    value: 'tag',
116.    label: 'Tag 标签'
117.  },
118.  {
119.    value: 'progress',
120.    label: 'Progress 进度条'
121.  },
122.  {
123.    value: 'tree',
124.    label: 'Tree 树形控件'
125.  },
126.  {
127.    value: 'pagination',
128.    label: 'Pagination 分页'
129.  },
130.  {
131.    value: 'notice',
132.    label: 'Notice',
133.    children: [{{
134.      value: 'alert',
135.      label: 'Alert 警告'
136.    },
137.    value: 'loading',
138.    label: 'Loading 加载'
139.  },
140.  {
141.    value: 'message',
142.    label: 'Message 消息提示'
```

```
141.     },
142.     value: 'message-box',
143.     label: 'MessageBox 弹框'
144.   },
145.   value: 'notification',
146.   label: 'Notification 通知'
147. ]
148. },
149. value: 'navigation',
150. label: 'Navigation',
151. children: [
152.   value: 'menu',
153.   label: 'NavMenu 导航菜单'
154. ],
155.   value: 'tabs',
156.   label: 'Tabs 标签页'
157. ],
158.   value: 'breadcrumb',
159.   label: 'Breadcrumb 面包屑'
160. ],
161.   value: 'dropdown',
162.   label: 'Dropdown 下拉菜单'
163. ],
164.   value: 'steps',
165.   label: 'Steps 步骤条'
166. ]
167. },
168. value: 'others',
169. label: 'Others',
170. children: [
171.   value: 'dialog',
172.   label: 'Dialog 对话框'
173. ],
174.   value: 'tooltip',
175.   label: 'Tooltip 文字提示'
176. ],
177.   value: 'popover',
178.   label: 'Popover 弹出框'
179. ],
180.   value: 'card',
181.   label: 'Card 卡片'
182. ]
```

```

183.          value: 'carousel',
184.          label: 'Carousel 走马灯'
185.        },
186.        value: 'collapse',
187.        label: 'Collapse 折叠面板'
188.      ]
189.    ]
190.  },
191.  value: 'ziyuan',
192.  label: '资源',
193.  children: [
194.    value: 'axure',
195.    label: 'Axure Components'
196.  ],
197.  value: 'sketch',
198.  label: 'Sketch Templates'
199. },
200. value: 'jiaohu',
201. label: '组件交互文档'
202. ]
203. ]
204. );
205. }
206. );
207. </script>

```

仅显示最后一级

可以仅在输入框中显示选中项最后一级的标签，而不是选中项所在的完整路径。



属性 `show-all-levels` 定义了是否显示完整的路径，将其赋值为 `false` 则仅显示最后一级

```
1. <el-cascader
2.   :options="options"
3.   :show-all-levels="false"
4. ></el-cascader>
5. <script>
6. export default {
7.   data() {
8.     return {
9.       options: [
10.         {
11.           value: 'zhinan',
12.           label: '指南',
13.           children: [
14.             {
15.               value: 'shejiyuanze',
16.               label: '设计原则',
17.               children: [
18.                 {
19.                   value: 'yizhi',
20.                   label: '一致'
21.                 },
22.                 {
23.                   value: 'fankui',
24.                   label: '反馈'
25.                 },
26.                 {
27.                   value: 'xiaolv',
28.                   label: '效率'
29.                 },
30.                 {
31.                   value: 'kekong',
32.                   label: '可控'
33.                 }
34.               ],
35.             },
36.             {
37.               value: 'daohang',
38.               label: '导航',
39.               children: [
40.                 {
41.                   value: 'zujian',
```

```
41.         label: '组件',
42.         children: [{  
43.             value: 'basic',
44.             label: 'Basic',
45.             children: [{  
46.                 value: 'layout',
47.                 label: 'Layout 布局'
48.             }, {  
49.                 value: 'color',
50.                 label: 'Color 色彩'
51.             }, {  
52.                 value: 'typography',
53.                 label: 'Typography 字体'
54.             }, {  
55.                 value: 'icon',
56.                 label: 'Icon 图标'
57.             }, {  
58.                 value: 'button',
59.                 label: 'Button 按钮'
60.             }]
61.         },
62.         value: 'form',
63.         label: 'Form',
64.         children: [{  
65.             value: 'radio',
66.             label: 'Radio 单选框'
67.         }, {  
68.             value: 'checkbox',
69.             label: 'Checkbox 多选框'
70.         }, {  
71.             value: 'input',
72.             label: 'Input 输入框'
73.         },
74.             value: 'input-number',
75.             label: 'InputNumber 计数器'
76.         },
77.             value: 'select',
78.             label: 'Select 选择器'
79.         },
80.             value: 'cascader',
81.             label: 'Cascader 级联选择器'
82.         }]
```

```
83.          value: 'switch',
84.          label: 'Switch 开关'
85.        },
86.        value: 'slider',
87.        label: 'Slider 滑块'
88.      },
89.      value: 'time-picker',
90.      label: 'TimePicker 时间选择器'
91.    },
92.    value: 'date-picker',
93.    label: 'DatePicker 日期选择器'
94.  },
95.  value: 'datetime-picker',
96.  label: 'DateTimePicker 日期时间选择器'
97. },
98. value: 'upload',
99. label: 'Upload 上传'
100. },
101. value: 'rate',
102. label: 'Rate 评分'
103. },
104. value: 'form',
105. label: 'Form 表单'
106. ]
107. },
108. value: 'data',
109. label: 'Data',
110. children: [
111.   value: 'table',
112.   label: 'Table 表格'
113. },
114.   value: 'tag',
115.   label: 'Tag 标签'
116. },
117.   value: 'progress',
118.   label: 'Progress 进度条'
119. },
120.   value: 'tree',
121.   label: 'Tree 树形控件'
122. },
123.   value: 'pagination',
124.   label: 'Pagination 分页'
```

```
125.      }, {
126.        value: 'badge',
127.        label: 'Badge 标记'
128.      }]
129.    }, {
130.      value: 'notice',
131.      label: 'Notice',
132.      children: [{
133.        value: 'alert',
134.        label: 'Alert 警告'
135.      }, {
136.        value: 'loading',
137.        label: 'Loading 加载'
138.      }, {
139.        value: 'message',
140.        label: 'Message 消息提示'
141.      }, {
142.        value: 'message-box',
143.        label: 'MessageBox 弹框'
144.      }, {
145.        value: 'notification',
146.        label: 'Notification 通知'
147.      }]
148.    }, {
149.      value: 'navigation',
150.      label: 'Navigation',
151.      children: [{
152.        value: 'menu',
153.        label: 'NavMenu 导航菜单'
154.      }, {
155.        value: 'tabs',
156.        label: 'Tabs 标签页'
157.      }, {
158.        value: 'breadcrumb',
159.        label: 'Breadcrumb 面包屑'
160.      }, {
161.        value: 'dropdown',
162.        label: 'Dropdown 下拉菜单'
163.      }, {
164.        value: 'steps',
165.        label: 'Steps 步骤条'
166.      }]
}
```

```
167.     }, {
168.         value: 'others',
169.         label: 'Others',
170.         children: [{
171.             value: 'dialog',
172.             label: 'Dialog 对话框'
173.         }, {
174.             value: 'tooltip',
175.             label: 'Tooltip 文字提示'
176.         }, {
177.             value: 'popover',
178.             label: 'Popover 弹出框'
179.         }, {
180.             value: 'card',
181.             label: 'Card 卡片'
182.         }, {
183.             value: 'carousel',
184.             label: 'Carousel 走马灯'
185.         }, {
186.             value: 'collapse',
187.             label: 'Collapse 折叠面板'
188.         }]
189.     }]
190. },
191.     value: 'ziyuan',
192.     label: '资源',
193.     children: [{
194.         value: 'axure',
195.         label: 'Axure Components'
196.     }, {
197.         value: 'sketch',
198.         label: 'Sketch Templates'
199.     }, {
200.         value: 'jiaohu',
201.         label: '组件交互文档'
202.     }]
203. ],
204. );
205. }
206. };
207. </script>
```

默认值

The screenshot shows a sidebar with the following structure:

- 组件 / Data / Tag 标签** (Selected)
- 指南** > Basic > Table 表格
- 组件** > Form > Tag 标签
- 资源** > Data > Progress 进度条
- Notice > Tree 树形控件
- Navigation > Pagination 分页
- Others > Badge 标记

默认值通过数组的方式指定。

```

1. <el-cascader
2.   :options="options"
3.   v-model="selectedOptions3"
4. ></el-cascader>
5. <script>
6.   export default {
7.     data() {
8.       return {
9.         options: [
10.           {
11.             value: 'zhinan',
12.             label: '指南',
13.             children: [
14.               {
15.                 value: 'shejiyuanze',
16.                 label: '设计原则',
17.                 children: [
18.                   {
19.                     value: 'yizhi',
20.                     label: '一致'
21.                   },
22.                   {
23.                     value: 'fankui',
24.                     label: '反馈'
25.                   },
26.                   {
27.                     value: 'xiaolv',
28.                     label: '效率'
29.                   },
30.                   {
31.                     value: 'kekong',
32.                     label: '可控'
33.                   }
34.                 ]
35.               }
36.             ]
37.           }
38.         ]
39.       }
40.     }
41.   }
42. </script>
43. <template>
44. <el-cascader
45.   :options="options"
46.   v-model="selectedOptions3"
47. ></el-cascader>
48. </div>

```

```
27.          }]
28.      },
29.      value: 'daohang',
30.      label: '导航',
31.      children: [{
32.          value: 'cexiangdaohang',
33.          label: '侧向导航'
34.      }, {
35.          value: 'dingbudaohang',
36.          label: '顶部导航'
37.      }]
38.  }]
39.  },
40.  value: 'zujian',
41.  label: '组件',
42.  children: [{
43.      value: 'basic',
44.      label: 'Basic',
45.      children: [{
46.          value: 'layout',
47.          label: 'Layout 布局'
48.      }, {
49.          value: 'color',
50.          label: 'Color 色彩'
51.      }, {
52.          value: 'typography',
53.          label: 'Typography 字体'
54.      }, {
55.          value: 'icon',
56.          label: 'Icon 图标'
57.      }, {
58.          value: 'button',
59.          label: 'Button 按钮'
60.      }]
61.  },
62.  value: 'form',
63.  label: 'Form',
64.  children: [{
65.      value: 'radio',
66.      label: 'Radio 单选框'
67.  }, {
68.      value: 'checkbox',
```

```
69.          label: 'Checkbox 多选框'
70.        },
71.        value: 'input',
72.        label: 'Input 输入框'
73.      },
74.      value: 'input-number',
75.      label: 'InputNumber 计数器'
76.    },
77.    value: 'select',
78.    label: 'Select 选择器'
79.  },
80.  value: 'cascader',
81.  label: 'Cascader 级联选择器'
82. },
83. value: 'switch',
84. label: 'Switch 开关'
85. },
86. value: 'slider',
87. label: 'Slider 滑块'
88. },
89. value: 'time-picker',
90. label: 'TimePicker 时间选择器'
91. },
92. value: 'date-picker',
93. label: 'DatePicker 日期选择器'
94. },
95. value: 'datetime-picker',
96. label: 'DateTimePicker 日期时间选择器'
97. },
98. value: 'upload',
99. label: 'Upload 上传'
100. },
101. value: 'rate',
102. label: 'Rate 评分'
103. },
104. value: 'form',
105. label: 'Form 表单'
106. }
107. },
108. value: 'data',
109. label: 'Data',
110. children: [{
```

```
111.          value: 'table',
112.          label: 'Table 表格'
113.        },
114.        {
115.          value: 'tag',
116.          label: 'Tag 标签'
117.        },
118.        {
119.          value: 'progress',
120.          label: 'Progress 进度条'
121.        },
122.        {
123.          value: 'tree',
124.          label: 'Tree 树形控件'
125.        },
126.        {
127.          value: 'pagination',
128.          label: 'Pagination 分页'
129.        },
130.        {
131.          value: 'notice',
132.          label: 'Notice',
133.          children: [
134.            {
135.              value: 'alert',
136.              label: 'Alert 警告'
137.            },
138.            {
139.              value: 'loading',
140.              label: 'Loading 加载'
141.            },
142.            {
143.              value: 'message',
144.              label: 'Message 消息提示'
145.            },
146.            {
147.              value: 'message-box',
148.              label: 'MessageBox 弹框'
149.            },
150.            {
151.              value: 'notification',
152.              label: 'Notification 通知'
153.            }
154.          ],
155.          label: 'Notice'
156.        }
157.      ]
158.    }
159.  ]
160.}
161.
```

```
153.         label: 'NavMenu 导航菜单'
154.     },
155.     value: 'tabs',
156.     label: 'Tabs 标签页'
157. },
158.     value: 'breadcrumb',
159.     label: 'Breadcrumb 面包屑'
160. },
161.     value: 'dropdown',
162.     label: 'Dropdown 下拉菜单'
163. },
164.     value: 'steps',
165.     label: 'Steps 步骤条'
166. ]
167. },
168.     value: 'others',
169.     label: 'Others',
170.     children: [
171.         value: 'dialog',
172.         label: 'Dialog 对话框'
173.     },
174.         value: 'tooltip',
175.         label: 'Tooltip 文字提示'
176.     },
177.         value: 'popover',
178.         label: 'Popover 弹出框'
179.     },
180.         value: 'card',
181.         label: 'Card 卡片'
182.     },
183.         value: 'carousel',
184.         label: 'Carousel 走马灯'
185.     },
186.         value: 'collapse',
187.         label: 'Collapse 折叠面板'
188.     ],
189. ],
190. },
191.     value: 'ziyuan',
192.     label: '资源',
193.     children: [
194.         value: 'axure',
```

```
195.         label: 'Axure Components'
196.     }, {
197.         value: 'sketch',
198.         label: 'Sketch Templates'
199.     }, {
200.         value: 'jiaohu',
201.         label: '组件交互文档'
202.     }]
203. ],
204.     selectedOptions3: ['zujian', 'data', 'tag']
205.     );
206. }
207. );
208. </script>
```

选择即改变

点击或移入选项即表示选中该项，可用于选择任意一级菜单的选项。



若需要允许用户选择任意一级选项，则可将 `change-on-select` 赋值为 `true`

```

1. <el-cascader
2.   :options="options"
3.   change-on-select
4. ></el-cascader>
5. <script>
6.   export default {
7.     data() {
8.       return {
9.         options: [
10.           {
11.             value: 'zhinan',
12.             label: '指南',
13.             children: [
14.               {
15.                 value: 'shejiyuanze',
16.                 label: '设计原则',
17.                 children: [
18.                   {
19.                     value: 'yizhi',
20.                     label: '一致'
21.                   },
22.                   {
23.                     value: 'fankui',
24.                     label: '反馈'
25.                   },
26.                   {
27.                     value: 'xiaolv',
28.                     label: '效率'
29.                   }
30.                 ]
31.               }
32.             ]
33.           }
34.         ]
35.       }
36.     }
37.   }
38. </script>

```

```
25.          value: 'kekong',
26.          label: '可控'
27.        }]
28.      },
29.      value: 'daohang',
30.      label: '导航',
31.      children: [
32.        value: 'cexiangdaohang',
33.        label: '侧向导航'
34.      ],
35.      value: 'dingbudaohang',
36.      label: '顶部导航'
37.    ]
38.  ]
39. },
40. value: 'zujian',
41. label: '组件',
42. children: [
43.   value: 'basic',
44.   label: 'Basic',
45.   children: [
46.     value: 'layout',
47.     label: 'Layout 布局'
48.   ],
49.   value: 'color',
50.   label: 'Color 色彩'
51. },
52.   value: 'typography',
53.   label: 'Typography 字体'
54. },
55.   value: 'icon',
56.   label: 'Icon 图标'
57. },
58.   value: 'button',
59.   label: 'Button 按钮'
60. ]
61. },
62. value: 'form',
63. label: 'Form',
64. children: [
65.   value: 'radio',
66.   label: 'Radio 单选框'
```

```
67.      }, {
68.        value: 'checkbox',
69.        label: 'Checkbox 多选框'
70.      }, {
71.        value: 'input',
72.        label: 'Input 输入框'
73.      }, {
74.        value: 'input-number',
75.        label: 'InputNumber 计数器'
76.      }, {
77.        value: 'select',
78.        label: 'Select 选择器'
79.      }, {
80.        value: 'cascader',
81.        label: 'Cascader 级联选择器'
82.      }, {
83.        value: 'switch',
84.        label: 'Switch 开关'
85.      }, {
86.        value: 'slider',
87.        label: 'Slider 滑块'
88.      }, {
89.        value: 'time-picker',
90.        label: 'TimePicker 时间选择器'
91.      }, {
92.        value: 'date-picker',
93.        label: 'DatePicker 日期选择器'
94.      }, {
95.        value: 'datetime-picker',
96.        label: 'DateTimePicker 日期时间选择器'
97.      }, {
98.        value: 'upload',
99.        label: 'Upload 上传'
100.     }, {
101.       value: 'rate',
102.       label: 'Rate 评分'
103.     }, {
104.       value: 'form',
105.       label: 'Form 表单'
106.     }]
107.   }, {
108.     value: 'data',
```

```
109.          label: 'Data',
110.          children: [{  
111.              value: 'table',  
112.              label: 'Table 表格'  
113.          }, {  
114.              value: 'tag',  
115.              label: 'Tag 标签'  
116.          }, {  
117.              value: 'progress',  
118.              label: 'Progress 进度条'  
119.          }, {  
120.              value: 'tree',  
121.              label: 'Tree 树形控件'  
122.          }, {  
123.              value: 'pagination',  
124.              label: 'Pagination 分页'  
125.          }, {  
126.              value: 'badge',  
127.              label: 'Badge 标记'  
128.          }]  
129.      }, {  
130.          value: 'notice',  
131.          label: 'Notice',  
132.          children: [{  
133.              value: 'alert',  
134.              label: 'Alert 警告'  
135.          }, {  
136.              value: 'loading',  
137.              label: 'Loading 加载'  
138.          }, {  
139.              value: 'message',  
140.              label: 'Message 消息提示'  
141.          }, {  
142.              value: 'message-box',  
143.              label: 'MessageBox 弹框'  
144.          }, {  
145.              value: 'notification',  
146.              label: 'Notification 通知'  
147.          }]  
148.      }, {  
149.          value: 'navigation',  
150.          label: 'Navigation',
```

```
151.         children: [{  
152.             value: 'menu',  
153.             label: 'NavMenu 导航菜单'  
154.         }, {  
155.             value: 'tabs',  
156.             label: 'Tabs 标签页'  
157.         }, {  
158.             value: 'breadcrumb',  
159.             label: 'Breadcrumb 面包屑'  
160.         }, {  
161.             value: 'dropdown',  
162.             label: 'Dropdown 下拉菜单'  
163.         }, {  
164.             value: 'steps',  
165.             label: 'Steps 步骤条'  
166.         }]  
167.     }, {  
168.         value: 'others',  
169.         label: 'Others',  
170.         children: [{  
171.             value: 'dialog',  
172.             label: 'Dialog 对话框'  
173.         }, {  
174.             value: 'tooltip',  
175.             label: 'Tooltip 文字提示'  
176.         }, {  
177.             value: 'popover',  
178.             label: 'Popover 弹出框'  
179.         }, {  
180.             value: 'card',  
181.             label: 'Card 卡片'  
182.         }, {  
183.             value: 'carousel',  
184.             label: 'Carousel 走马灯'  
185.         }, {  
186.             value: 'collapse',  
187.             label: 'Collapse 折叠面板'  
188.         }]  
189.     }]  
190. }, {  
191.     value: 'ziyuan',  
192.     label: '资源',
```

```

193.         children: [
194.             {
195.                 value: 'axure',
196.                 label: 'Axure Components'
197.             },
198.             {
199.                 value: 'sketch',
200.                 label: 'Sketch Templates'
201.             },
202.             {
203.                 value: 'jiaohu',
204.                 label: '组件交互文档'
205.             }
206.         ];
207.     </script>

```

动态加载次级选项

当选中某一级时，动态加载该级下的选项。



本例的选项数据源在初始化时不包含城市数据。利用 `active-item-change` 事件，可以在用户点击某个省份时拉取该省份下的城市数据。此外，本例还展示了 `props` 属性的用法。

```

1.  <el-cascader
2.    :options="options2"
3.    @active-item-change="handleItemChange"
4.    :props="props"
5.  ></el-cascader>
6.
7.  <script>

```

```
8.  export default {
9.    data() {
10.      return {
11.        options2: [
12.          { label: '江苏', cities: [] },
13.          { label: '浙江', cities: [] }
14.        ],
15.        props: {
16.          value: 'label',
17.          children: 'cities'
18.        }
19.      }
20.    };
21.  },
22.  methods: {
23.    handleItemChange(val) {
24.      console.log('active item:', val);
25.      setTimeout(_ => {
26.        if (val.indexOf('江苏') > -1 && !this.options2[0].cities.length) {
27.          this.options2[0].cities = [{ label: '南京' }];
28.        } else if (val.indexOf('浙江') > -1 && !this.options2[1].cities.length) {
29.          this.options2[1].cities = [{ label: '杭州' }];
30.        }
31.      }, 300);
32.    }
33.  }
34.};
35.};
36.};
37.};
38.},
39.},
40.},
41.},
42. </script>
```

可搜索

可以快捷地搜索选项并选择。

只可选择最后一级菜单的选项

可选择任意一级菜单的选项

试试搜索：指南

试试搜索：指南

将 `filterable` 赋值为 `true` 即可打开搜索功能。

```

1. <div class="block">
2.   <span class="demonstration">只可选择最后一级菜单的选项</span>
3.   <el-cascader
4.     placeholder="试试搜索：指南"
5.     :options="options"
6.     filterable
7.   ></el-cascader>
8. </div>
9. <div class="block">
10.  <span class="demonstration">可选择任意一级菜单的选项</span>
11.  <el-cascader
12.    placeholder="试试搜索：指南"
13.    :options="options"
14.    filterable
15.    change-on-select
16.  ></el-cascader>
17. </div>
18.
19. <script>
20.   export default {
21.     data() {
22.       return {
23.         options: [
24.           {
25.             value: 'zhinan',
26.             label: '指南',
27.             children: [
28.               {
29.                 value: 'shejiyuanze',
30.                 label: '设计原则',
31.                 children: [
32.                   {
33.                     value: 'yizhi',
34.                     label: '一致'
35.                   },
36.                   {
37.                     value: 'fankui',
38.                     label: '反馈'
39.                   }
40.                 ]
41.               }
42.             ]
43.           }
44.         ]
45.       }
46.     }
47.   }
48. </script>
```

```
36.          value: 'xiaolv',
37.          label: '效率'
38.        },
39.        {
40.          value: 'kekong',
41.          label: '可控'
42.        }
43.      },
44.      {
45.        value: 'daohang',
46.        label: '导航',
47.        children: [
48.          {
49.            value: 'cexiangdaohang',
50.            label: '侧向导航'
51.          }
52.        ]
53.      },
54.      {
55.        value: 'zujian',
56.        label: '组件',
57.        children: [
58.          {
59.            value: 'basic',
60.            label: 'Basic',
61.            children: [
62.              {
63.                value: 'layout',
64.                label: 'Layout 布局'
65.              },
66.              {
67.                value: 'color',
68.                label: 'Color 色彩'
69.              },
70.              {
71.                value: 'typography',
72.                label: 'Typography 字体'
73.              },
74.              {
75.                value: 'icon',
76.                label: 'Icon 图标'
77.              }
78.            ]
79.          },
80.          {
81.            value: 'button',
82.            label: 'Button 按钮'
83.          }
84.        ]
85.      },
86.      {
87.        value: 'form',
88.        label: 'Form',
```

```
78.          children: [{  
79.              value: 'radio',  
80.              label: 'Radio 单选框'  
81.          }, {  
82.              value: 'checkbox',  
83.              label: 'Checkbox 多选框'  
84.          }, {  
85.              value: 'input',  
86.              label: 'Input 输入框'  
87.          }, {  
88.              value: 'input-number',  
89.              label: 'InputNumber 计数器'  
90.          }, {  
91.              value: 'select',  
92.              label: 'Select 选择器'  
93.          }, {  
94.              value: 'cascader',  
95.              label: 'Cascader 级联选择器'  
96.          }, {  
97.              value: 'switch',  
98.              label: 'Switch 开关'  
99.          }, {  
100.             value: 'slider',  
101.             label: 'Slider 滑块'  
102.         }, {  
103.             value: 'time-picker',  
104.             label: 'TimePicker 时间选择器'  
105.         }, {  
106.             value: 'date-picker',  
107.             label: 'DatePicker 日期选择器'  
108.         }, {  
109.             value: 'datetime-picker',  
110.             label: 'DateTimePicker 日期时间选择器'  
111.         }, {  
112.             value: 'upload',  
113.             label: 'Upload 上传'  
114.         }, {  
115.             value: 'rate',  
116.             label: 'Rate 评分'  
117.         }, {  
118.             value: 'form',  
119.             label: 'Form 表单'
```

```
120.      }]
121.    },
122.      value: 'data',
123.      label: 'Data',
124.      children: [{{
125.        value: 'table',
126.        label: 'Table 表格'
127.      }, {
128.        value: 'tag',
129.        label: 'Tag 标签'
130.      }, {
131.        value: 'progress',
132.        label: 'Progress 进度条'
133.      }, {
134.        value: 'tree',
135.        label: 'Tree 树形控件'
136.      }, {
137.        value: 'pagination',
138.        label: 'Pagination 分页'
139.      }, {
140.        value: 'badge',
141.        label: 'Badge 标记'
142.      }]
143.    },
144.      value: 'notice',
145.      label: 'Notice',
146.      children: [{{
147.        value: 'alert',
148.        label: 'Alert 警告'
149.      }, {
150.        value: 'loading',
151.        label: 'Loading 加载'
152.      }, {
153.        value: 'message',
154.        label: 'Message 消息提示'
155.      }, {
156.        value: 'message-box',
157.        label: 'MessageBox 弹框'
158.      }, {
159.        value: 'notification',
160.        label: 'Notification 通知'
161.      }]
}
```

```
162.     },
163.     value: 'navigation',
164.     label: 'Navigation',
165.     children: [
166.       {
167.         value: 'menu',
168.         label: 'NavMenu 导航菜单'
169.       },
170.       {
171.         value: 'tabs',
172.         label: 'Tabs 标签页'
173.       },
174.       {
175.         value: 'breadcrumb',
176.         label: 'Breadcrumb 面包屑'
177.       },
178.       {
179.         value: 'dropdown',
180.         label: 'Dropdown 下拉菜单'
181.       },
182.       {
183.         value: 'others',
184.         label: 'Others',
185.         children: [
186.           {
187.             value: 'dialog',
188.             label: 'Dialog 对话框'
189.           },
190.           {
191.             value: 'tooltip',
192.             label: 'Tooltip 文字提示'
193.           },
194.           {
195.             value: 'popover',
196.             label: 'Popover 弹出框'
197.           },
198.           {
199.             value: 'card',
200.             label: 'Card 卡片'
201.           },
202.           {
203.             value: 'collapse',
204.             label: 'Collapse 折叠面板'
205.           }
206.         ]
207.       }
208.     ]
209.   ]
210. }
```

```

204.     },
205.     value: 'ziyuan',
206.     label: '资源',
207.     children: [
208.       {
209.         value: 'axure',
210.         label: 'Axure Components'
211.       },
212.       {
213.         value: 'sketch',
214.         label: 'Sketch Templates'
215.       },
216.     ]
217.   ];
218. }
219. }
220. }
221. </script>

```

Attributes

参数	说明	类型	可选值	默认值
options	可选项数据源，键名可通过 <code>props</code> 属性配置	array	—	—
props	配置选项，具体见下表	object	—	—
value	选中项绑定值	array	—	—
separator	选项分隔符	string	—	斜杠'/'
popper-class	自定义浮层类名	string	—	—
placeholder	输入框占位文本	string	—	请选择
disabled	是否禁用	boolean	—	false
clearable	是否支持清空选项	boolean	—	false
expand-trigger	次级菜单的展开方式	string	click / hover	click
show-all-levels	输入框中是否显示选中值的完整路径	boolean	—	true
filterable	是否可搜索选项	boolean	—	—
debounce	搜索关键词输入的去抖延迟，毫秒	number	—	300
change-on-select	是否允许选择任意一级的选项	boolean	—	false

size	尺寸	string	medium / small / mini	-
before-filter	筛选之前的钩子，参数为输入的值，若返回 false 或者返回 Promise 且被 reject，则停止筛选	function(value)	-	-

props

参数	说明	类型	可选值	默认值
value	指定选项的值为选项对象的某个属性值	string	-	-
label	指定选项标签为选项对象的某个属性值	string	-	-
children	指定选项的子选项为选项对象的某个属性值	string	-	-
disabled	指定选项的禁用为选项对象的某个属性值	string	-	-

Events

事件名称	说明	回调参数
change	当绑定值变化时触发的事件	当前值
active-item-change	当父级选项变化时触发的事件，仅在 <code>change-on-select</code> 为 <code>false</code> 时可用	各父级选项组成的数组
blur	在 Cascader 失去焦点时触发	(event: Event)
focus	在 Cascader 获得焦点时触发	(event: Event)

原文: <http://element-cn.eleme.io/#/zh-CN/component/cascader>

Switch 开关

Switch 开关

表示两种相互对立的状态间的切换，多用于触发「开/关」。

基本用法



绑定 `v-model` 到一个 `Boolean` 类型的变量。可以使用 `active-color` 属性与 `inactive-color` 属性来设置开关的背景色。

```

1. <el-switch
2.   v-model="value2"
3.   active-color="#13ce66"
4.   inactive-color="#ff4949">
5. </el-switch>
6.
7. <script>
8.   export default {
9.     data() {
10.       return {
11.         value1: true,
12.         value2: true
13.       }
14.     }
15.   };
16. </script>

```

文字描述

按年付费 按月付费

按年付费 按月付费

使用 `active-text` 属性与 `inactive-text` 属性来设置开关的文字描述。

```

1. <el-switch
2.   v-model="value3"
3.   active-text="按月付费"
4.   inactive-text="按年付费">
5. </el-switch>
6. <el-switch
7.   style="display: block"
8.   v-model="value4"
9.   active-color="#13ce66"
10.  inactive-color="#ff4949"
11.  active-text="按月付费"
12.  inactive-text="按年付费">
13. </el-switch>
14.
15. <script>
16.   export default {
17.     data() {
18.       return {
19.         value3: true,
20.         value4: true
21.       }
22.     }
23.   };
24. </script>

```

扩展的 value 类型



设置 `active-value` 和 `inactive-value` 属性，接受 `Boolean` , `String` 或 `Number` 类型的值。

```

1. <el-tooltip :content="'Switch value: ' + value5" placement="top">
2.   <el-switch
3.     v-model="value5"
4.     active-color="#13ce66"
5.     inactive-color="#ff4949"
6.     active-value="100"
7.     inactive-value="0">

```

```

8.  </el-switch>
9.  </el-tooltip>
10.
11. <script>
12.   export default {
13.     data() {
14.       return {
15.         value5: '100'
16.       }
17.     }
18.   };
19. </script>

```

[显示代码](#)

禁用状态



设置 `disabled` 属性，接受一个 `Boolean`，设置 `true` 即可禁用。

```

1.  <el-switch
2.    v-model="value6"
3.    disabled>
4.  </el-switch>
5.  <el-switch
6.    v-model="value7"
7.    disabled>
8.  </el-switch>
9.  <script>
10.   export default {
11.     data() {
12.       return {
13.         value6: true,
14.         value7: false
15.       }
16.     }
17.   };
18. </script>

```

[显示代码](#)

Attributes

参数	说明	类型	可选值	默认值
disabled	是否禁用	boolean	-	false
width	switch 的宽度 (像素)	number	-	40
active-icon-class	switch 打开时所显示图标的类名, 设置此项会忽略 active-text	string	-	-
inactive-icon-class	switch 关闭时所显示图标的类名, 设置此项会忽略 inactive-text	string	-	-
active-text	switch 打开时的文字描述	string	-	-
inactive-text	switch 关闭时的文字描述	string	-	-
active-value	switch 打开时的值	boolean / string / number	-	true
inactive-value	switch 关闭时的值	boolean / string / number	-	false
active-color	switch 打开时的背景色	string	-	#409EFF
inactive-color	switch 关闭时的背景色	string	-	#C0CCDA
name	switch 对应的 name 属性	string	-	-

Events

事件名称	说明	回调参数
change	switch 状态发生变化时的回调函数	新状态的值

Methods

方法名	说明	参数
focus	使 Switch 获取焦点	-

原文: <http://element-cn.eleme.io/#/zh-CN/component/switch>

Slider 滑块

Slider 滑块

通过拖动滑块在一个固定区间内进行选择

基础用法

在拖动滑块时，显示当前值



通过设置绑定值自定义滑块的初始值

```
1. <template>
2.   <div class="block">
3.     <span class="demonstration">默认</span>
4.     <el-slider v-model="value1"></el-slider>
5.   </div>
6.   <div class="block">
7.     <span class="demonstration">自定义初始值</span>
8.     <el-slider v-model="value2"></el-slider>
9.   </div>
10.  <div class="block">
```

```

11.      <span class="demonstration">隐藏 Tooltip</span>
12.      <el-slider v-model="value3" :show-tooltip="false"></el-slider>
13.    </div>
14.    <div class="block">
15.      <span class="demonstration">格式化 Tooltip</span>
16.      <el-slider v-model="value4" :format-tooltip="formatTooltip"></el-slider>
17.    </div>
18.    <div class="block">
19.      <span class="demonstration">禁用</span>
20.      <el-slider v-model="value5" disabled></el-slider>
21.    </div>
22.  </template>
23.
24.  <script>
25.  export default {
26.    data() {
27.      return {
28.        value1: 0,
29.        value2: 50,
30.        value3: 36,
31.        value4: 48,
32.        value5: 42
33.      }
34.    },
35.    methods: {
36.      formatTooltip(val) {
37.        return val / 100;
38.      }
39.    }
40.  }
41. </script>

```

离散值

选项可以是离散的

不显示间断点



显示间断点



改变 `step` 的值可以改变步长，通过设置 `show-step` 属性可以显示间断点

```

1. <template>
2.   <div class="block">
3.     <span class="demonstration">不显示间断点</span>
4.     <el-slider
5.       v-model="value6"
6.       :step="10">
7.     </el-slider>
8.   </div>
9.   <div class="block">
10.    <span class="demonstration">显示间断点</span>
11.    <el-slider
12.      v-model="value7"
13.      :step="10"
14.      show-stops>
15.    </el-slider>
16.  </div>
17. </template>
18.
19. <script>
20. export default {
21.   data() {
22.     return {
23.       value6: 0,
24.       value7: 0
25.     }
26.   }
27. }
28. </script>

```

显示代码

带有输入框

通过输入框设置精确数值



设置 `show-input` 属性会在右侧显示一个输入框

```

1. <template>
2.   <div class="block">
3.     <el-slider
4.       v-model="value8"
5.       show-input>
6.     </el-slider>
7.   </div>
8. </template>
9.
10. <script>
11.   export default {
12.     data() {
13.       return {
14.         value8: 0
15.       }
16.     }
17.   }
18. </script>

```

范围选择

支持选择某一数值范围



设置 `range` 即可开启范围选择，此时绑定值是一个数组，其元素分别为最小边界值和最大边界值

```

1. <template>
2.   <div class="block">
3.     <el-slider
4.       v-model="value9"
5.       range
6.       show-stops
7.       :max="10">
8.     </el-slider>
9.   </div>
10. </template>
11.
12. <script>
13.   export default {
14.     data() {

```

```
15.     return {
16.       value9: [4, 8]
17.     }
18.   }
19. }
20. </script>
```

竖向模式



设置 `vertical` 可使 `Slider` 变成竖向模式，此时必须设置高度 `height` 属性

```
1. <template>
2.   <div class="block">
3.     <el-slider
4.       v-model="value10"
5.       vertical
6.       height="200px">
7.     </el-slider>
8.   </div>
9. </template>
10.
11. <script>
12.   export default {
13.     data() {
14.       return {
15.         value10: 0
16.       }
17.     }
18.   }
19. </script>
```

Attributes

参数	说明	类型	可选值	默认值
min	最小值	number	-	0
max	最大值	number	-	100
disabled	是否禁用	boolean	-	false
step	步长	number	-	1
show-input	是否显示输入框，仅在非范围选择时有效	boolean	-	false
show-input-controls	在显示输入框的情况下，是否显示输入框的控制按钮	boolean	-	true
input-size	输入框的尺寸	string	large / medium / small / mini	small
show-stops	是否显示中断点	boolean	-	false
show-tooltip	是否显示 tooltip	boolean	-	true
format-tooltip	格式化 tooltip message	function(value)	-	-
range	是否为范围选择	boolean	-	false
vertical	是否竖向模式	boolean	-	false
height	Slider 高度，竖向模式时必填	string	-	-
label	屏幕阅读器标签	string	-	-
debounce	输入时的去抖延迟，毫秒，仅在 <code>show-input</code> 等于true时有效	number	-	300
tooltip-class	tooltip 的自定义类名	string	-	-

Events

事件名称	说明	回调参数
change	值改变时触发（使用鼠标拖曳时，只在松开鼠标后触发）	改变后的值

原文：<http://element-cn.eleme.io/#/zh-CN/component/slider>

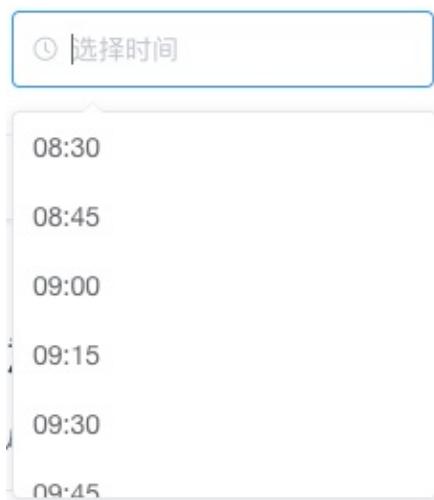
TimePicker 时间选择器

TimePicker 时间选择器

用于选择或输入日期

固定时间点

提供几个固定的时间点供用户选择



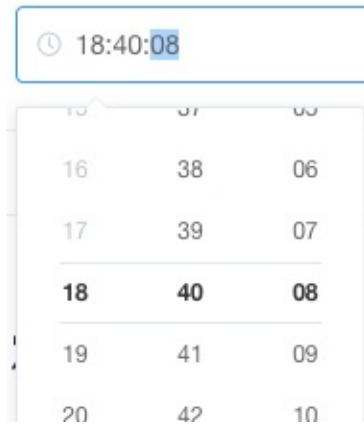
使用 `el-time-select` 标签，分别通过 `start`、`end` 和 `step` 指定可选的起始时间、结束时间和步长

```
1. <el-time-select
2.   v-model="value1"
3.   :picker-options="{
4.     start: '08:30',
5.     step: '00:15',
6.     end: '18:30'
7.   }"
8.   placeholder="选择时间">
9. </el-time-select>
10.
11. <script>
12.   export default {
13.     data() {
14.       return {
15.         value1: ''
16.       };
17.     }
18.   }
19. </script>
```

```
17.      }
18.    }
19.  </script>
```

任意时间点

可以选择任意时间



使用 `el-time-picker` 标签，通过 `selectableRange` 限制可选时间范围。提供了两种交互方式：
默认情况下通过鼠标滚轮进行选择，打开 `arrow-control` 属性则通过界面上的箭头进行选择。

```
1.  <template>
2.    <el-time-picker
3.      v-model="value2"
4.      :picker-options="{
5.        selectableRange: '18:30:00 - 20:30:00'
6.      }"
7.      placeholder="任意时间点">
8.    </el-time-picker>
9.    <el-time-picker
10.       arrow-control
11.       v-model="value3"
12.       :picker-options="{
13.         selectableRange: '18:30:00 - 20:30:00'
14.       }"
15.       placeholder="任意时间点">
16.     </el-time-picker>
17.   </template>
18.
19.  <script>
20.    export default {
21.      data() {
```

```

22.      return {
23.        value2: new Date(2016, 9, 10, 18, 40),
24.        value3: new Date(2016, 9, 10, 18, 40)
25.      };
26.    }
27.  }
28. </script>

```

固定时间范围

若先选择开始时间，则结束时间内备选项的状态会随之改变

① 起始时间 ① 结束时间

```

1.  <template>
2.    <el-time-select
3.      placeholder="起始时间"
4.      v-model="startTime"
5.      :picker-options="{
6.        start: '08:30',
7.        step: '00:15',
8.        end: '18:30'
9.      }">
10.   </el-time-select>
11.   <el-time-select
12.     placeholder="结束时间"
13.     v-model="endTime"
14.     :picker-options="{
15.       start: '08:30',
16.       step: '00:15',
17.       end: '18:30',
18.       minTime: startTime
19.     }">
20.   </el-time-select>
21. </template>
22.
23. <script>
24.   export default {
25.     data() {
26.       return {

```

```

27.         startTime: '',
28.         endTime: ''
29.     };
30. }
31. }
32. </script>

```

任意时间范围

可选择任意的时间范围

添加 `is-range` 属性即可选择时间范围，同样支持 `arrow-control` 属性。

```

1.  <template>
2.    <el-time-picker
3.      is-range
4.      v-model="value4"
5.      range-separator="至"
6.      start-placeholder="开始时间"
7.      end-placeholder="结束时间"
8.      placeholder="选择时间范围">
9.    </el-time-picker>
10.   <el-time-picker
11.     is-range
12.     arrow-control
13.     v-model="value5"
14.     range-separator="至"
15.     start-placeholder="开始时间"
16.     end-placeholder="结束时间"
17.     placeholder="选择时间范围">
18.   </el-time-picker>
19. </template>
20.
21. <script>
22. export default {
23.   data() {
24.     return {
25.       value4: [new Date(2016, 9, 10, 8, 40), new Date(2016, 9, 10, 9, 40)],
26.       value5: [new Date(2016, 9, 10, 8, 40), new Date(2016, 9, 10, 9, 40)]
27.     };
28.   }
29. }

```

30. </script>

Attributes

参数	说明	类型	可选值	默认值
readonly	完全只读	boolean	—	false
disabled	禁用	boolean	—	false
editable	文本框可输入	boolean	—	true
clearable	是否显示清除按钮	boolean	—	true
size	输入框尺寸	string	medium / small / mini	—
placeholder	非范围选择时的占位内容	string	—	—
start-placeholder	范围选择时开始日期的占位内容	string	—	—
end-placeholder	范围选择时结束日期的占位内容	string	—	—
is-range	是否为时间范围选择。仅对 <code><el-time-picker></code> 有效	boolean	—	false
arrow-control	是否使用箭头进行时间选择。仅对 <code><el-time-picker></code> 有效	boolean	—	false
value	绑定值	date(TimePicker) / string(TimeSelect)	—	—
align	对齐方式	string	left / center / right	left
popper-class	TimePicker 下拉框的类名	string	—	—
picker-options	当前时间日期选择器特有的选项参考下表	object	—	{}
range-separator	选择范围时的分隔符	string	-	' - '
value-format	可选，仅TimePicker时可用，绑定值的格式。不指定则绑定值为 Date 对象	string	见日期格式	—
default-value	可选，选择器打开时默认显示的时间	Date(TimePicker) / string(TimeSelect)	可被 <code>new Date()</code> 解析 (TimePicker) / 可选值 (TimeSelect)	—
name	原生属性	string	—	—
prefix-icon	自定义头部图标的类名	string	—	el-icon-time
clear-icon	自定义清空图标的类名	string	—	el-icon-

clear-icon	自定义清空图标的类名	string	-	circle-close
------------	------------	--------	---	--------------

Time Select Options

参数	说明	类型	可选值	默认值
start	开始时间	string	-	09:00
end	结束时间	string	-	18:00
step	间隔时间	string	-	00:30
minTime	最小时间，小于该时间的时间段将被禁用	string	-	00:00
maxTime	最大时间，大于该时间的时间段将被禁用	string	-	-

Time Picker Options

参数	说明	类型	可选值	默认值
selectableRange	可选时间段，例如 '18:30:00 - 20:30:00' 或者传入数组 ['18:30:00 - 19:00:00', '19:00:00 - 20:30:00']	string / array	-	-
format	时间格式化(TimePicker)	string	小时: HH , 分: mm , 秒: ss , AM/PM A	'HH:mm:ss'

Events

事件名	说明	参数
change	用户确认选定的值时触发	组件绑定值
blur	当 input 失去焦点时触发	组件实例
focus	当 input 获得焦点时触发	组件实例

Methods

方法名	说明	参数
focus	使 input 获得焦点	-

原文: <http://element-cn.eleme.io/#/zh-CN/component/time-picker>

DatePicker 日期选择器

DatePicker 日期选择器

用于选择或输入日期

选择日

以「日」为基本单位，基础的日期选择控件



基本单位由 `type` 属性指定。快捷选项需配置 `picker-options` 对象中的 `shortcuts`，禁用日期通过 `disabledDate` 设置，传入函数

```

1.  <template>
2.    <div class="block">
3.      <span class="demonstration">默认</span>
4.      <el-date-picker
5.        v-model="value1"
6.        type="date"
7.        placeholder="选择日期">
8.      </el-date-picker>
9.    </div>
10.   <div class="block">
```

```
11.      <span class="demonstration">带快捷选项</span>
12.      <el-date-picker
13.          v-model="value2"
14.          align="right"
15.          type="date"
16.          placeholder="选择日期"
17.          :picker-options="pickerOptions1">
18.      </el-date-picker>
19.  </div>
20. </template>
21.
22. <script>
23. export default {
24.     data() {
25.         return {
26.             pickerOptions1: {
27.                 disabledDate(time) {
28.                     return time.getTime() > Date.now();
29.                 },
30.                 shortcuts: [
31.                     {
32.                         text: '今天',
33.                         onClick(picker) {
34.                             picker.$emit('pick', new Date());
35.                         }
36.                     },
37.                     {
38.                         text: '昨天',
39.                         onClick(picker) {
40.                             const date = new Date();
41.                             date.setTime(date.getTime() - 3600 * 1000 * 24);
42.                             picker.$emit('pick', date);
43.                         }
44.                     },
45.                     {
46.                         text: '一周前',
47.                         onClick(picker) {
48.                             const date = new Date();
49.                             date.setTime(date.getTime() - 3600 * 1000 * 24 * 7);
50.                             picker.$emit('pick', date);
51.                         }
52.                     }
53.                 ]
54.             },
55.             value1: '',
56.             value2: ''
57.         }
58.     }
59. }
```

```
53.      };
54.    }
55.  };
56. </script>
```

其他日期单位

通过扩展基础的日期选择，可以选择周、月、年或多个日期

周

年

选择周

选择年

月

多个日期

选择月

选择一个或多个日期

```
1.  <div class="container">
2.    <div class="block">
3.      <span class="demonstration">周</span>
4.      <el-date-picker
5.        v-model="value3"
6.        type="week"
7.        format="yyyy 第 ww 周"
8.        placeholder="选择周">
9.      </el-date-picker>
10.     </div>
11.     <div class="block">
12.       <span class="demonstration">月</span>
13.       <el-date-picker
14.         v-model="value4"
15.         type="month"
16.         placeholder="选择月">
17.       </el-date-picker>
18.     </div>
19.   </div>
20.   <div class="container">
21.     <div class="block">
```

```
22.      <span class="demonstration">年</span>
23.      <el-date-picker
24.          v-model="value5"
25.          type="year"
26.          placeholder="选择年">
27.      </el-date-picker>
28.  </div>
29.  <div class="block">
30.      <span class="demonstration">多个日期</span>
31.      <el-date-picker
32.          type="dates"
33.          v-model="value14"
34.          placeholder="选择一个或多个日期">
35.      </el-date-picker>
36.  </div>
37. </div>
38.
39. <script>
40.   export default {
41.     data() {
42.       return {
43.         value3: '',
44.         value4: '',
45.         value5: '',
46.         value14: []
47.       };
48.     }
49.   };
50. </script>
```

选择日期范围

可在任何一个选择器中便捷地选择一个时间范围



在选择日期范围时， 默认情况下左右面板会联动。如果希望两个面板各自独立切换当前月份， 可以使用 `unlink-panels` 属性解除联动。

```

1. <template>
2.   <div class="block">
3.     <span class="demonstration">默认</span>
4.     {{value6}}
5.     <el-date-picker
6.       v-model="value6"
7.       type="daterange"
8.       range-separator="至"
9.       start-placeholder="开始日期"
10.      end-placeholder="结束日期">
11.    </el-date-picker>
12.  </div>
13.  <div class="block">
14.    <span class="demonstration">带快捷选项</span>
15.    <el-date-picker
16.      v-model="value7"
17.      type="daterange"
18.      align="right"
19.      unlink-panels
20.      range-separator="至"
21.      start-placeholder="开始日期"
22.      end-placeholder="结束日期">

```

```
23.      :picker-options="pickerOptions2">
24.    </el-date-picker>
25.  </div>
26. </template>
27.
28. <script>
29. export default {
30.   data() {
31.     return {
32.       pickerOptions2: {
33.         shortcuts: [
34.           {
35.             text: '最近一周',
36.             onClick(picker) {
37.               const end = new Date();
38.               const start = new Date();
39.               start.setTime(start.getTime() - 3600 * 1000 * 24 * 7);
40.               picker.$emit('pick', [start, end]);
41.             }
42.           },
43.           {
44.             text: '最近一个月',
45.             onClick(picker) {
46.               const end = new Date();
47.               const start = new Date();
48.               start.setTime(start.getTime() - 3600 * 1000 * 24 * 30);
49.               picker.$emit('pick', [start, end]);
50.             }
51.           },
52.           {
53.             text: '最近三个月',
54.             onClick(picker) {
55.               const end = new Date();
56.               const start = new Date();
57.               start.setTime(start.getTime() - 3600 * 1000 * 24 * 90);
58.               picker.$emit('pick', [start, end]);
59.             }
60.           }
61.         ],
62.         value6: '',
63.         value7: ''
64.       };
65.     }
66.   }
67. }
```

日期格式

使用 `format` 指定输入框的格式；使用 `value-format` 指定绑定值的格式。

默认情况下，组件接受并返回 `Date` 对象。以下为可用的格式化字串，以 UTC 2017年1月2日 03:04:05 为例：

请注意大小写

格式	含义	备注	举例
<code>yyyy</code>	年		2017
<code>M</code>	月	不补0	1
<code>MM</code>	月		01
<code>W</code>	周	仅周选择器的 <code>format</code> 可用；不补0	1
<code>WW</code>	周	仅周选择器的 <code>format</code> 可用	01
<code>d</code>	日	不补0	2
<code>dd</code>	日		02
<code>H</code>	小时	24小时制；不补0	3
<code>HH</code>	小时	24小时制	03
<code>h</code>	小时	12小时制，须和 <code>A</code> 或 <code>a</code> 使用；不补0	3
<code>hh</code>	小时	12小时制，须和 <code>A</code> 或 <code>a</code> 使用	03
<code>m</code>	分钟	不补0	4
<code>mm</code>	分钟		04
<code>s</code>	秒	不补0	5
<code>ss</code>	秒		05
<code>A</code>	AM/PM	仅 <code>format</code> 可用，大写	AM
<code>a</code>	am/pm	仅 <code>format</code> 可用，小写	am
<code>timestamp</code>	JS时间戳	仅 <code>value-format</code> 可用；组件绑定值为 <code>number</code> 类型	1483326245000

默认为 Date 对象

值：

使用 value-format

值：

时间戳

值：

```

1. <template>
2.   <div class="block">
3.     <span class="demonstration">默认为 Date 对象</span>
4.     <div class="demonstration">值：{{ value10 }}</div>
5.   <el-date-picker>

```

```
6.      v-model="value10"
7.      type="date"
8.      placeholder="选择日期"
9.      format="yyyy 年 MM 月 dd 日">
10.     </el-date-picker>
11.   </div>
12.   <div class="block">
13.     <span class="demonstration">使用 value-format</span>
14.     <div class="demonstration">值 : {{ value11 }}</div>
15.     <el-date-picker
16.       v-model="value11"
17.       type="date"
18.       placeholder="选择日期"
19.       format="yyyy 年 MM 月 dd 日"
20.       value-format="yyyy-MM-dd">
21.     </el-date-picker>
22.   </div>
23.   <div class="block">
24.     <span class="demonstration">时间戳</span>
25.     <div class="demonstration">值 : {{ value12 }}</div>
26.     <el-date-picker
27.       v-model="value12"
28.       type="date"
29.       placeholder="选择日期"
30.       format="yyyy 年 MM 月 dd 日"
31.       value-format="timestamp">
32.     </el-date-picker>
33.   </div>
34. </template>
35.
36. <script>
37.   export default {
38.     data() {
39.       return {
40.         value10: '',
41.         value11: '',
42.         value12: ''
43.       };
44.     }
45.   };
46. </script>
```

默认显示日期

在选择日期范围时，指定起始日期和结束日期的默认时刻。

组件值：["2018-08-01T16:00:00.000Z", "2018-09-06T15:59:59.000Z"]



选择日期范围时，默认情况下，起始日期和结束日期的时间部分均为当天的 0 点 0 分 0 秒。通过 `default-time` 可以分别指定二者的具体时刻。`default-time` 接受一个数组，其中的值为形如 `12:00:00` 的字符串，第一个值控制起始日期的时刻，第二个值控制结束日期的时刻。

```

1. <template>
2.   <div class="block">
3.     <p>组件值：{{ value13 }}</p>
4.     <el-date-picker
5.       v-model="value13"
6.       type="daterange"
7.       start-placeholder="开始日期"
8.       end-placeholder="结束日期"
9.       :default-time="['00:00:00', '23:59:59']">
10.    </el-date-picker>
11.  </div>
12. </template>
13.
14. <script>
15.   export default {
16.     data() {

```

```

17.      return {
18.        value13: []
19.      };
20.    }
21.  };
22. </script>

```

Attributes

参数	说明	类型	可选值	默认值
readonly	完全只读	boolean	—	false
disabled	禁用	boolean	—	false
editable	文本框可输入	boolean	—	true
clearable	是否显示清除按钮	boolean	—	true
size	输入框尺寸	string	large, small, mini	—
placeholder	非范围选择时的占位内容	string	—	—
start-placeholder	范围选择时开始日期的占位内容	string	—	—
end-placeholder	范围选择时结束日期的占位内容	string	—	—
type	显示类型	string	year/month/date/dates/week/datetime/datetimerange/daterange	date
format	显示在输入框中的格式	string	见 日期格式	yyyy-MM-dd
align	对齐方式	string	left, center, right	left
popper-class	DatePicker 下拉框的类名	string	—	—
picker-options	当前时间日期选择器特有的选项参考下表	object	—	{}
range-separator	选择范围时的分隔符	string	—	' - '
default-value	可选，选择器打开时默认显示的时间	Date	可被 <code>new Date()</code> 解析	—
default-time	范围选择时选中日期所使用的当日内具体时刻	string[]	数组，长度为 2，每项值为字符串，形如 <code>12:00:00</code> ，第一项指定开始日期的时刻，第二项指定结束日期的时刻，不指定会使用时刻 <code>00:00:00</code>	—
value-format	可选，绑定值的格式。不指定则绑定值为 Date 对象	string	见 日期格式	—

name	原生属性	string	—	—
unlink-panels	在范围选择器里取消两个日期面板之间的联动	boolean	—	false
prefix-icon	自定义头部图标的类名	string	—	el-icon-date
clear-icon	自定义清空图标的类名	string	—	el-icon-circle-close

Picker Options

参数	说明	类型	可选值	默认值
shortcuts	设置快捷选项，需要传入 { text, onClick } 对象用法参考 demo 或下表	Object[]	—	—
disabledDate	设置禁用状态，参数为当前日期，要求返回 Boolean	Function	—	—
firstDayOfWeek	周起始日	Number	1 到 7	7
onPick	选中日期后会执行的回调。只有当 <code>daterange</code> 或 <code>datetimerange</code> 才生效	Function({ maxDate, minDate })	—	—

Shortcuts

参数	说明	类型	可选值	默认值
text	标题文本	string	—	—
onClick	选中后的回调函数，参数是 vm，可通过触发 'pick' 事件设置选择器的值。例如 <code>vm.\$emit('pick', new Date())</code>	function	—	—

Events

事件名称	说明	回调参数
change	用户确认选定的值时触发	组件绑定值。格式与绑定值一致，可受 <code>value-format</code> 控制
blur	当 input 失去焦点时触发	组件实例
focus	当 input 获得焦点时触发	组件实例

Methods

方法名	说明	参数
focus	使 input 获取焦点	-

原文: <http://element-cn.eleme.io/#/zh-CN/component/date-picker>



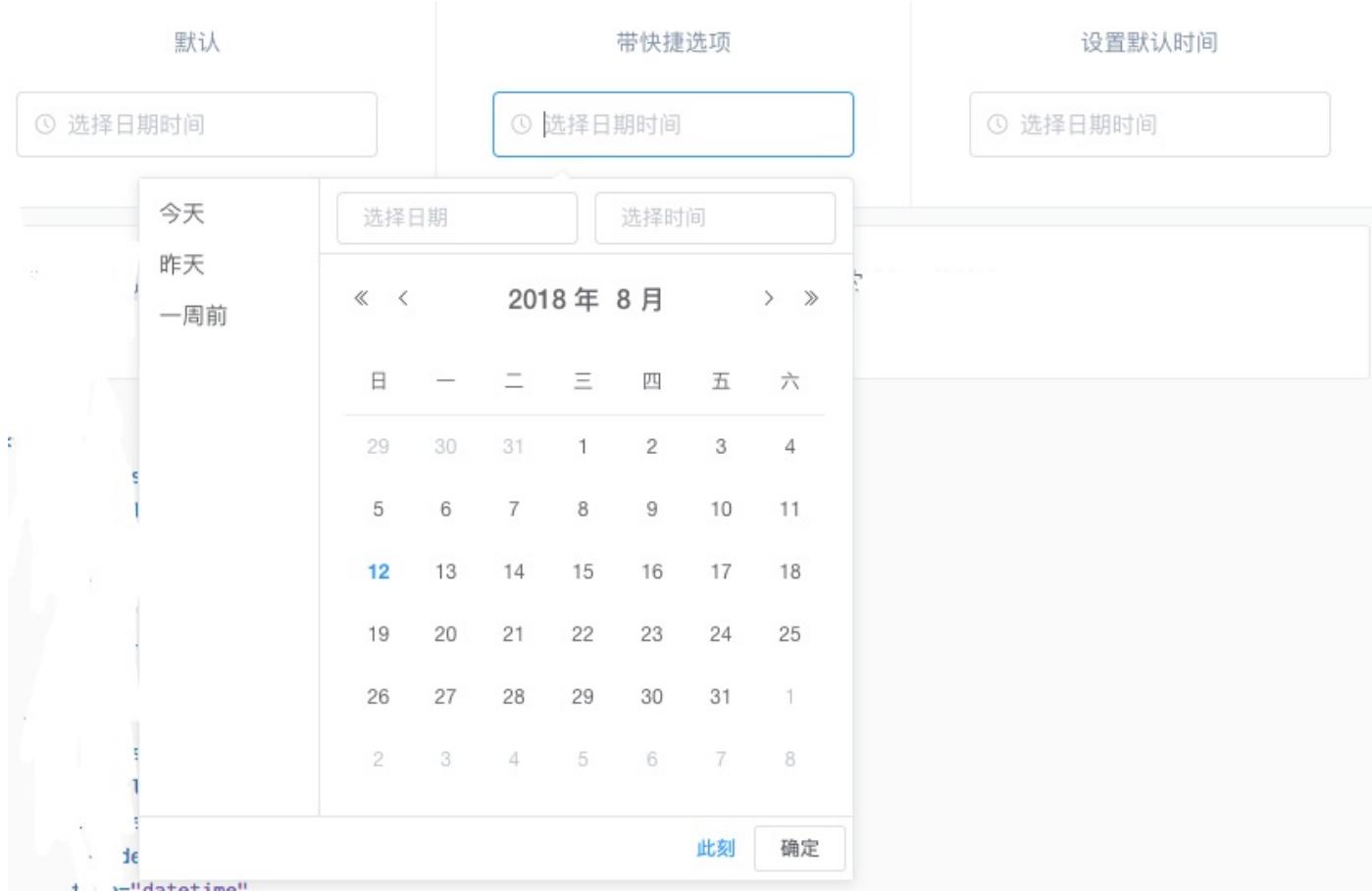
DateTimePicker 日期时间选择器

DateTimePicker 日期时间选择器

在同一个选择器里选择日期和时间

DateTimePicker 由 DatePicker 和 TimePicker 派生，[Picker Options](#) 或者其他选项可以参照 DatePicker 和 TimePicker。

日期和时间点



通过设置 `type` 属性为 `datetime`，即可在同一个选择器里同时进行日期和时间的选择。快捷选项的使用方法与 Date Picker 相同。

```

1. <template>
2.   <div class="block">
3.     <span class="demonstration">默认</span>
4.     <el-date-picker
5.       v-model="value1"
6.       type="datetime">

```

```
7.      placeholder="选择日期时间">
8.      </el-date-picker>
9.    </div>
10.   <div class="block">
11.     <span class="demonstration">带快捷选项</span>
12.     <el-date-picker
13.       v-model="value2"
14.       type="datetime"
15.       placeholder="选择日期时间"
16.       align="right"
17.       :picker-options="pickerOptions1">
18.     </el-date-picker>
19.   </div>
20.   <div class="block">
21.     <span class="demonstration">设置默认时间</span>
22.     <el-date-picker
23.       v-model="value3"
24.       type="datetime"
25.       placeholder="选择日期时间"
26.       default-time="12:00:00">
27.     </el-date-picker>
28.   </div>
29. </template>
30.
31. <script>
32.   export default {
33.     data() {
34.       return {
35.         pickerOptions1: {
36.           shortcuts: [
37.             {
38.               text: '今天',
39.               onClick(picker) {
40.                 picker.$emit('pick', new Date());
41.               }
42.             },
43.             {
44.               text: '昨天',
45.               onClick(picker) {
46.                 const date = new Date();
47.                 date.setTime(date.getTime() - 3600 * 1000 * 24);
48.                 picker.$emit('pick', date);
49.               }
50.             }
51.           }
52.         }
53.       }
54.     }
55.   }
56. </script>
```

```

49.          text: '一周前',
50.          onClick(picker) {
51.            const date = new Date();
52.            date.setTime(date.getTime() - 3600 * 1000 * 24 * 7);
53.            picker.$emit('pick', date);
54.          }
55.        }]
56.      },
57.      value1: '',
58.      value2: '',
59.      value3: ''
60.    );
61.  }
62. }
63. </script>

```

日期和时间范围



设置 `type` 为 `datetimerange` 即可选择日期和时间范围

```

1.  <template>
2.    <div class="block">
3.      <span class="demonstration">默认</span>

```

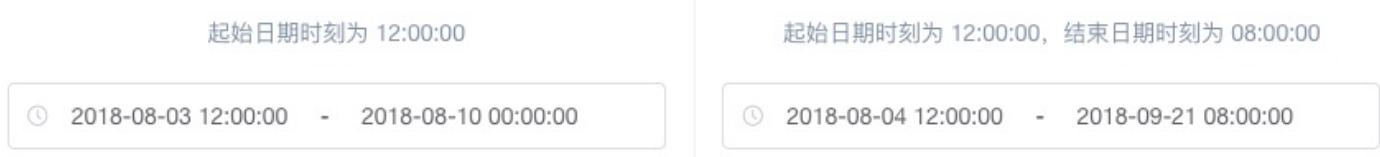
```
4.      <el-date-picker  
5.          v-model="value4"  
6.          type="datetimerange"  
7.          range-separator="至"  
8.          start-placeholder="开始日期"  
9.          end-placeholder="结束日期">  
10.     </el-date-picker>  
11.   </div>  
12.   <div class="block">  
13.     <span class="demonstration">带快捷选项</span>  
14.     <el-date-picker  
15.         v-model="value5"  
16.         type="datetimerange"  
17.         :picker-options="pickerOptions2"  
18.         range-separator="至"  
19.         start-placeholder="开始日期"  
20.         end-placeholder="结束日期"  
21.         align="right">  
22.     </el-date-picker>  
23.   </div>  
24. </template>  
25.  
26. <script>  
27.   export default {  
28.     data() {  
29.       return {  
30.         pickerOptions2: {  
31.           shortcuts: [{  
32.             text: '最近一周',  
33.             onClick(picker) {  
34.               const end = new Date();  
35.               const start = new Date();  
36.               start.setTime(start.getTime() - 3600 * 1000 * 24 * 7);  
37.               picker.$emit('pick', [start, end]);  
38.             }  
39.           }, {  
40.             text: '最近一个月',  
41.             onClick(picker) {  
42.               const end = new Date();  
43.               const start = new Date();  
44.               start.setTime(start.getTime() - 3600 * 1000 * 24 * 30);  
45.               picker.$emit('pick', [start, end]);  
46.             }  
47.           }  
48.         }  
49.       }  
50.     }  
51.   }  
52. </script>
```

```

46.          }
47.      }, {
48.          text: '最近三个月',
49.          onClick(picker) {
50.              const end = new Date();
51.              const start = new Date();
52.              start.setTime(start.getTime() - 3600 * 1000 * 24 * 90);
53.              picker.$emit('pick', [start, end]);
54.          }
55.      }]
56.  },
57. value4: [new Date(2000, 10, 10, 10, 10), new Date(2000, 10, 11, 10,
10)],
58.         value5: ''
59.     });
60. }
61. );
62. </script>

```

默认的起始与结束时刻



使用 `datetimerange` 进行范围选择时，在日期选择面板中选定起始与结束的日期，默认会使用该日期的 `00:00:00` 作为起始与结束的时刻；通过选项 `default-time` 可以控制选中起始与结束日期时所使用的具体时刻。`default-time` 接受一个数组，数组每项值为字符串，形如 `12:00:00`，其中第一项控制起始日期的具体时刻，第二项控制结束日期的具体时刻。

```

1. <template>
2. <div class="block">
3.     <span class="demonstration">起始日期时刻为 12:00:00</span>
4.     <el-date-picker
5.         v-model="value6"
6.         type="datetimerange"
7.         start-placeholder="开始日期"
8.         end-placeholder="结束日期"
9.         :default-time="['12:00:00']">
10.    </el-date-picker>

```

```

11.    </div>
12.    <div class="block">
13.      <span class="demonstration">起始日期时刻为 12:00:00, 结束日期时刻为
14.        08:00:00</span>
15.      <el-date-picker
16.        v-model="value7"
17.        type="datetimerange"
18.        align="right"
19.        start-placeholder="开始日期"
20.        end-placeholder="结束日期"
21.        :default-time="['12:00:00', '08:00:00']">
22.      </el-date-picker>
23.    </div>
24.
25.  <script>
26.    export default {
27.      data() {
28.        return {
29.          value6: '',
30.          value7: ''
31.        };
32.      }
33.    };
34.  </script>

```

[显示代码](#)

Attributes

参数	说明	类型	可选值	默认值
readonly	完全只读	boolean	—	false
disabled	禁用	boolean	—	false
editable	文本框可输入	boolean	—	true
clearable	是否显示清除按钮	boolean	—	true
size	输入框尺寸	string	large, small, mini	—
placeholder	非范围选择时的占位内容	string	—	—
start-placeholder	范围选择时开始日期的占位内容	string	—	—
end-placeholder	范围选择时结束日期的占位内容	string	—	—

time-arrow-control	是否使用箭头进行时间选择	boolean	—	false
type	显示类型	string	year/month/date/week/datetime/datetimerange/daterange	date
format	显示在输入框中的格式	string	见 日期格式	yyyy-MM-dd
align	对齐方式	string	left, center, right	left
popper-class	DateTimePicker 下拉框的类名	string	—	—
picker-options	当前时间日期选择器特有的选项参考下表	object	—	{}
range-separator	选择范围时的分隔符	string	-	' - '
default-value	可选，选择器打开时默认显示的时间	Date	可被 <code>new Date()</code> 解析	—
default-time	选中日期后的默认具体时刻	非范围选择时： string / 范围选择时： string[]	非范围选择时：形如 <code>12:00:00</code> 的字符串；范围选择时：数组，长度为 2，每项值为字符串，形如 <code>12:00:00</code> ，第一项指定开始日期的时刻，第二项指定结束日期的时刻。不指定会使用时刻 <code>00:00:00</code>	—
value-format	可选，绑定值的格式。不指定则绑定值为 Date 对象	string	见 日期格式	—
name	原生属性	string	—	—
unlink-panels	在范围选择器里取消两个日期面板之间的联动	boolean	—	false
prefix-icon	自定义头部图标的类名	string	—	el-icon-date
clear-icon	自定义清空图标的类名	string	—	el-icon-circle-close

Picker Options

参数	说明	类型	可选值	默认值
shortcuts	设置快捷选项，需要传入 { text, onClick } 对象用法参考 demo 或下表	Object[]	—	—
disabledDate	设置禁用状态，参数为当前日期，要求返回 Boolean	Function	—	—
firstDayOfWeek	周起始日	Number	1 到 7	7

Shortcuts

参数	说明	类型	可选值	默认值
text	标题文本	string	—	—
onClick	选中后的回调函数，参数是 vm，可通过触发 'pick' 事件设置选择器的值。例如 vm.\$emit('pick', new Date())	function	—	—

Events

Event Name	Description	Parameters
change	用户确认选定的值时触发	组件绑定值。格式与绑定值一致，可受 value-format 控制
blur	当 input 失去焦点时触发	组件实例
focus	当 input 获得焦点时触发	组件实例

Methods

方法名	说明	参数
focus	使 input 获得焦点	—

原文: <http://element-cn.eleme.io/#/zh-CN/component/datetime-picker>



Upload 上传

Upload 上传

通过点击或者拖拽上传文件

点击上传



通过 slot 你可以传入自定义的上传按钮类型和文字提示。可通过设置 `limit` 和 `on-exceed` 来限制上传文件的个数和定义超出限制时的行为。可通过设置 `before-remove` 来阻止文件移除操作。

```

1. <el-upload
2.   class="upload-demo"
3.   action="https://jsonplaceholder.typicode.com/posts/"
4.   :on-preview="handlePreview"
5.   :on-remove="handleRemove"
6.   :before-remove="beforeRemove"
7.   multiple
8.   :limit="3"
9.   :on-exceed="handleExceed"
10.  :file-list="fileList">
11.    <el-button size="small" type="primary">点击上传</el-button>
12.    <div slot="tip" class="el-upload__tip">只能上传jpg/png文件，且不超过500kb</div>
13.  </el-upload>
14.  <script>
15.    export default {
16.      data() {
17.        return {
18.          fileList: [{name: 'food.jpeg', url:
'https://fuss10.elemecdn.com/3/63/4e7f3a15429bfda99bce42a18cdd1jpeg.jpeg?
imageMogr2/thumbnail/360x360/format/webp/quality/100'}, {name: 'food2.jpeg',
url: 'https://fuss10.elemecdn.com/3/63/4e7f3a15429bfda99bce42a18cdd1jpeg.jpeg?
imageMogr2/thumbnail/360x360/format/webp/quality/100'}]

```

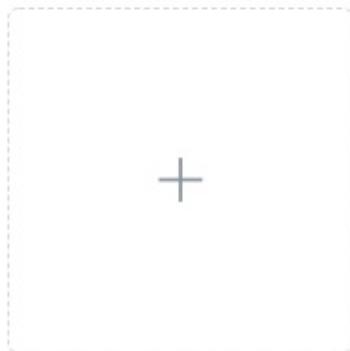
```

19.      };
20.    },
21.    methods: {
22.      handleRemove(file, fileList) {
23.        console.log(file, fileList);
24.      },
25.      handlePreview(file) {
26.        console.log(file);
27.      },
28.      handleExceed(files, fileList) {
29.        this.$message.warning(`当前限制选择 3 个文件，本次选择了 ${files.length} 个
   文件，共选择了 ${files.length + fileList.length} 个文件`);
30.      },
31.      beforeRemove(file, fileList) {
32.        return this.$confirm(`确定移除 ${file.name}?`);
33.      }
34.    }
35.  }
36. </script>

```

用户头像上传

使用 `before-upload` 限制用户上传的图片格式和大小。



```

1. <el-upload
2.   class="avatar-uploader"
3.   action="https://jsonplaceholder.typicode.com/posts/"
4.   :show-file-list="false"
5.   :on-success="handleAvatarSuccess"
6.   :before-upload="beforeAvatarUpload">
7.   
8.   <i v-else class="el-icon-plus avatar-uploader-icon"></i>
9. </el-upload>

```

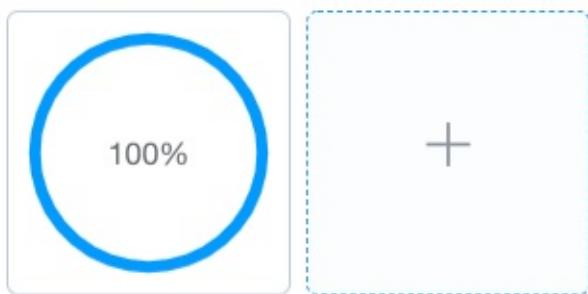
```
10.
11. <style>
12.   .avatar-uploader .el-upload {
13.     border: 1px dashed #d9d9d9;
14.     border-radius: 6px;
15.     cursor: pointer;
16.     position: relative;
17.     overflow: hidden;
18.   }
19.   .avatar-uploader .el-upload:hover {
20.     border-color: #409EFF;
21.   }
22.   .avatar-uploader-icon {
23.     font-size: 28px;
24.     color: #8c939d;
25.     width: 178px;
26.     height: 178px;
27.     line-height: 178px;
28.     text-align: center;
29.   }
30.   .avatar {
31.     width: 178px;
32.     height: 178px;
33.     display: block;
34.   }
35. </style>
36.
37. <script>
38. export default {
39.   data() {
40.     return {
41.       imageUrl: ''
42.     };
43.   },
44.   methods: {
45.     handleAvatarSuccess(res, file) {
46.       this.imageUrl = URL.createObjectURL(file.raw);
47.     },
48.     beforeAvatarUpload(file) {
49.       const isJPEG = file.type === 'image/jpeg';
50.       const isLt2M = file.size / 1024 / 1024 < 2;
51.     }
52.   }
53. }
```

```
52.     if (!isJPG) {
53.         this.$message.error('上传头像图片只能是 JPG 格式！');
54.     }
55.     if (!isLt2M) {
56.         this.$message.error('上传头像图片大小不能超过 2MB！');
57.     }
58.     return isJPG && isLt2M;
59.   }
60. }
61. }
62. </script>
```

显示代码

照片墙

使用 `list-type` 属性来设置文件列表的样式。



```
1. <el-upload
2.   action="https://jsonplaceholder.typicode.com/posts/"
3.   list-type="picture-card"
4.   :on-preview="handlePictureCardPreview"
5.   :on-remove="handleRemove">
6.   <i class="el-icon-plus"></i>
7. </el-upload>
8. <el-dialog :visible.sync="dialogVisible">
9.   
10. </el-dialog>
11. <script>
12.   export default {
13.     data() {
14.       return {
15.         dialogImageUrl: '',
16.         dialogVisible: false
```

```

17.      };
18.    },
19.    methods: {
20.      handleRemove(file, fileList) {
21.        console.log(file, fileList);
22.      },
23.      handlePictureCardPreview(file) {
24.        this.dialogImageUrl = file.url;
25.        this.dialogVisible = true;
26.      }
27.    }
28.  }
29. </script>

```

图片列表缩略图



```

1. <el-upload
2.   class="upload-demo"
3.   action="https://jsonplaceholder.typicode.com/posts/"
4.   :on-preview="handlePreview"
5.   :on-remove="handleRemove"
6.   :file-list="fileList2"
7.   list-type="picture">
8.   <el-button size="small" type="primary">点击上传</el-button>
9.   <div slot="tip" class="el-upload__tip">只能上传jpg/png文件，且不超过500kb</div>
10. </el-upload>
11. <script>
12.   export default {
13.     data() {

```

```

14.     return {
15.       fileList2: [{name: 'food.jpeg', url:
16.         'https://fuss10.elemecdn.com/3/63/4e7f3a15429bfda99bce42a18cdd1jpeg.jpeg?
17.         imageMogr2/thumbnail/360x360/format/webp/quality/100'}, {name: 'food2.jpeg',
18.         url: 'https://fuss10.elemecdn.com/3/63/4e7f3a15429bfda99bce42a18cdd1jpeg.jpeg?
19.         imageMogr2/thumbnail/360x360/format/webp/quality/100'}]
20.       };
21.     },
22.     methods: {
23.       handleRemove(file, fileList) {
24.         console.log(file, fileList);
25.       },
26.       handlePreview(file) {
27.         console.log(file);
28.       }
29.     }
30.   }
31. 
```

上传文件列表控制

通过 `on-change` 钩子函数来对列表进行控制

点击上传

只能上传jpg/png文件，且不超过500kb

food.jpeg	
food2.jpeg	

```

1. <el-upload
2.   class="upload-demo"
3.   action="https://jsonplaceholder.typicode.com/posts/"
4.   :on-change="handleChange"
5.   :file-list="fileList3">
6.   <el-button size="small" type="primary">点击上传</el-button>
7.   <div slot="tip" class="el-upload__tip">只能上传jpg/png文件，且不超过500kb</div>
8. </el-upload>
9. <script>
10.  export default {
11.    data() {
12.      return {
13.        fileList3: []
14.      }
15.    }
16.  }
17. 
```

```

13.     fileList3: [
14.       name: 'food.jpeg',
15.       url:
16.         'https://fuss10.elemecdn.com/3/63/4e7f3a15429bfda99bce42a18cdd1jpeg.jpeg?
17.           imageMogr2/thumbnail/360x360/format/webp/quality/100'
18.     },
19.   ],
20. },
21. },
22. methods: {
23.   handleChange(file, fileList) {
24.     this.fileList3 = fileList.slice(-3);
25.   }
26. }
27. }
28. </script>

```

拖拽上传



将文件拖到此处，或[点击上传](#)

只能上传jpg/png文件，且不超过500kb

```

1. <el-upload
2.   class="upload-demo"
3.   drag
4.   action="https://jsonplaceholder.typicode.com/posts/"
5.   multiple>
6.   <i class="el-icon-upload"></i>
7.   <div class="el-upload__text">将文件拖到此处，或<em>点击上传</em></div>
8.   <div class="el-upload__tip" slot="tip">只能上传jpg/png文件，且不超过500kb</div>

```

9. </el-upload>

手动上传

The screenshot shows a manual file upload interface. At the top, there are two buttons: a blue '选取文件' (Select File) button and a green '上传到服务器' (Upload to Server) button. Below these buttons is a note: '只能上传jpg/png文件, 且不超过500kb'. A file list displays two files: 'food.jpeg' and 'food2.jpeg', each accompanied by a green checkmark icon.

```
1. <el-upload
2.   class="upload-demo"
3.   ref="upload"
4.   action="https://jsonplaceholder.typicode.com/posts/"
5.   :on-preview="handlePreview"
6.   :on-remove="handleRemove"
7.   :file-list="fileList"
8.   :auto-upload="false">
9.   <el-button slot="trigger" size="small" type="primary">选取文件</el-button>
10.  <el-button style="margin-left: 10px;" size="small" type="success"
11.    @click="submitUpload">上传到服务器</el-button>
12.  </el-upload>
13.  <script>
14.    export default {
15.      data() {
16.        return {
17.          fileList: [{name: 'food.jpeg', url:
18.            'https://fuss10.elemecdn.com/3/63/4e7f3a15429bfda99bce42a18cdd1jpeg.jpeg?
19.            imageMogr2/thumbnail/360x360/format/webp/quality/100'}, {name: 'food2.jpeg',
20.            url: 'https://fuss10.elemecdn.com/3/63/4e7f3a15429bfda99bce42a18cdd1jpeg.jpeg?
21.            imageMogr2/thumbnail/360x360/format/webp/quality/100'}]
22.        };
23.      },
24.      methods: {
25.        submitUpload() {
26.          this.$refs.upload.submit();
27.        },
28.        handleRemove(file, fileList) {
29.          console.log(file, fileList);
30.        }
31.      }
32.    }
33.  </script>
```

```

26.      },
27.      handlePreview(file) {
28.        console.log(file);
29.      }
30.    }
31.  }
32. </script>

```

Attribute

参数	说明	类型	可选值
action	必选参数，上传的地址	string	-
headers	设置上传的请求头部	object	-
multiple	是否支持多选文件	boolean	-
data	上传时附带的额外参数	object	-
name	上传的文件字段名	string	-
with-credentials	支持发送 cookie 凭证信息	boolean	-
show-file-list	是否显示已上传文件列表	boolean	-
drag	是否启用拖拽上传	boolean	-
accept	接受上传的文件类型 (thumbnail-mode 模式下此参数无效)	string	-
on-preview	点击文件列表中已上传的文件时的钩子	function(file)	-
on-remove	文件列表移除文件时的钩子	function(file, fileList)	-
on-success	文件上传成功时的钩子	function(response, file, fileList)	-
on-error	文件上传失败时的钩子	function(err, file, fileList)	-
on-progress	文件上传时的钩子	function(event, file, fileList)	-
on-change	文件状态改变时的钩子，添加文件、上传成功和上传失败时都会被调用	function(file, fileList)	-
before-upload	上传文件之前的钩子，参数为上传的文件，若返回 false 或者返回 Promise 且被 reject，则停止上传。	function(file)	-
before-remove	删除文件之前的钩子，参数为上传的文件和文件列表，若返回 false 或者返回 Promise 且被 reject，则停止上传。	function(file, fileList)	-
list-type	文件列表的类型	string	text/picture/picard
auto-upload	是否在选取文件后立即进行上传	boolean	-
	上传的文件列表，例如： [{name:		

file-list	<code>'food.jpg', url: 'https://xxx.cdn.com/xxx.jpg'}]</code>	array	-
http-request	覆盖默认的上传行为，可以自定义上传的实现	function	-
disabled	是否禁用	boolean	-
limit	最大允许上传个数	number	-
on-exceed	文件超出个数限制时的钩子	function(files, fileList)	-

Slot

name	说明
trigger	触发文件选择框的内容
tip	提示说明文字

Methods

方法名	说明	参数
clearFiles	清空已上传的文件列表（该方法不支持在 before-upload 中调用）	-
abort	取消上传请求	(file: fileList 中的 file 对象)

原文: <http://element-cn.eleme.io/#/zh-CN/component/upload>



Rate 评分

Rate 评分

评分组件

基础用法

默认不区分颜色



区分颜色



评分被分为三个等级，可以利用颜色对分数及情感倾向进行分级（默认情况下不区分颜色）。三个等级所对应的颜色用过 `colors` 属性设置，而它们对应的两个阈值则通过 `low-threshold` 和 `high-threshold` 设定。

```

1. <div class="block">
2.   <span class="demonstration">默认不区分颜色</span>
3.   <el-rate v-model="value1"></el-rate>
4. </div>
5. <div class="block">
6.   <span class="demonstration">区分颜色</span>
7.   <el-rate
8.     v-model="value2"
9.     :colors="['#99A9BF', '#F7BA2A', '#FF9900']">
10.  </el-rate>
11. </div>
12.
13. <script>
14.   export default {
15.     data() {
16.       return {
17.         value1: null,
18.         value2: null
19.       }
20.     }
21.   }
22. </script>

```

辅助文字

用辅助文字直接地表达对应分数



为组件设置 `show-text` 属性会在右侧显示辅助文字。通过设置 `texts` 可以为每一个分值指定对应的辅助文字。`texts` 为一个数组，长度应等于最大值 `max`。

```

1. <el-rate
2.   v-model="value3"
3.   show-text>
4. </el-rate>
5.
6. <script>
7.   export default {
8.     data() {
9.       return {
10.         value3: null
11.       }
12.     }
13.   }
14. </script>
```

其它 icon

当有多层评价时，可以用不同类型的 icon 区分评分层级



设置 `icon-classes` 属性可以自定义对应 3 个不同分段的图标。本例还使用 `void-icon-class` 指定了未选中时的图标类名。

```

1. <el-rate
2.   v-model="value4"
3.   :icon-classes="['icon-rate-face-1', 'icon-rate-face-2', 'icon-rate-face-3']"
4.   void-icon-class="icon-rate-face-off"
5.   :colors="['#99A9BF', '#F7BA2A', '#FF9900']">
6. </el-rate>
7.
```

```

8. <script>
9.   export default {
10.     data() {
11.       return {
12.         value4: null
13.       }
14.     }
15.   }
16. </script>

```

只读

只读的评分用来展示分数，允许出现半星



为组件设置 `disabled` 属性表示组件为只读，支持小数分值。此时若设置 `show-score`，则会在右侧显示目前的分值。可以提供 `score-template` 作为显示模板，模板为一个包含了 `{value}` 的字符串，`{value}` 会被解析为分值。

```

1. <el-rate
2.   v-model="value5"
3.   disabled
4.   show-score
5.   text-color="#ff9900"
6.   score-template="{value}">
7. </el-rate>
8.
9. <script>
10. export default {
11.   data() {
12.     return {
13.       value5: 3.7
14.     }
15.   }
16. }
17. </script>

```

[显示代码](#)

Attributes

参数	说明	类型	可选值	默认值
max	最大分值	number	—	5
disabled	是否为只读	boolean	—	false
allow-half	是否允许半选	boolean	—	false
low-threshold	低分和中等分数的界限值，值本身被划分在低分中	number	—	2
high-threshold	高分和中等分数的界限值，值本身被划分在高分中	number	—	4
colors	icon 的颜色数组，共有 3 个元素，为 3 个分段所对应的颜色	array	—	['#F7BA2A', '#F7BA2A', '#F7BA2A']
void-color	未选中 icon 的颜色	string	—	#C6D1DE
disabled-void-color	只读时未选中 icon 的颜色	string	—	#EFF2F7
icon-classes	icon 的类名数组，共有 3 个元素，为 3 个分段所对应的类名	array	—	['el-icon-star-on', 'el-icon-star-on', 'el-icon-star-on']
void-icon-class	未选中 icon 的类名	string	—	el-icon-star-off
disabled-void-icon-class	只读时未选中 icon 的类名	string	—	el-icon-star-on
show-text	是否显示辅助文字，若为真，则会从 texts 数组中选取当前分数对应的文字内容	boolean	—	false
show-score	是否显示当前分数，show-score 和 show-text 不能同时为真	boolean	—	false
text-color	辅助文字的颜色	string	—	#1F2D3D
texts	辅助文字数组	array	—	['极差', '失望', '一般', '满意', '惊喜']
score-template	分数显示模板	string	—	{value}

Events

事件名称	说明	回调参数
change	分值改变时触发	改变后的分值

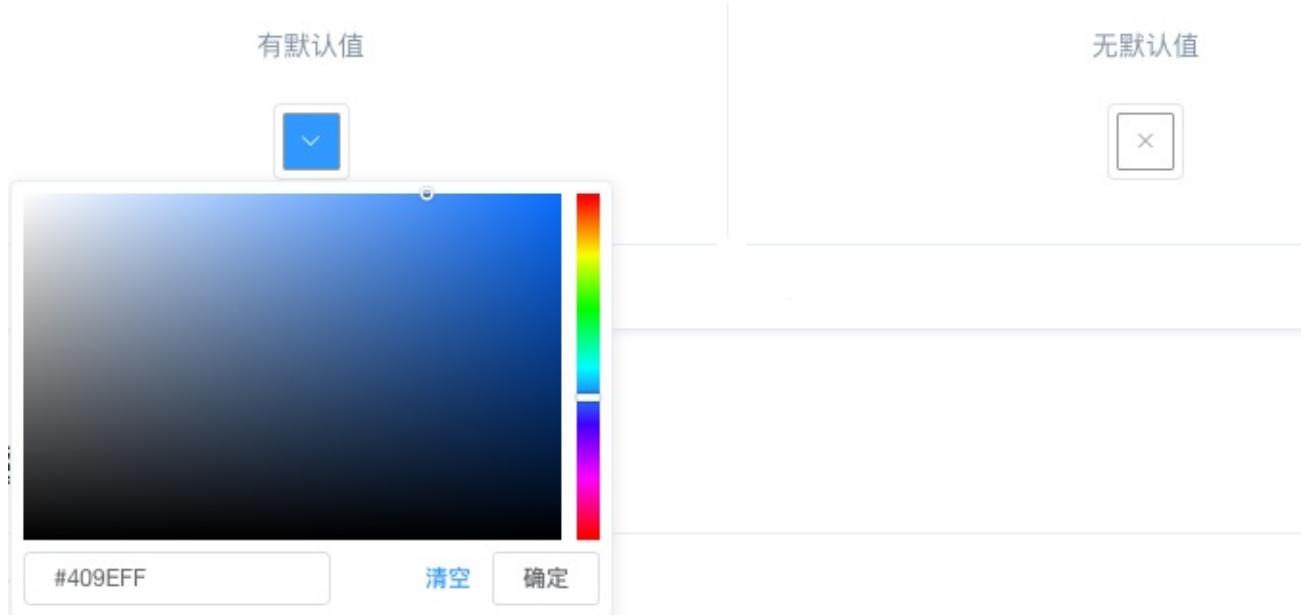
原文：<http://element-cn.eleme.io/#/zh-CN/component/rate>

ColorPicker 颜色选择器

ColorPicker 颜色选择器

用于颜色选择，支持多种格式。

基础用法



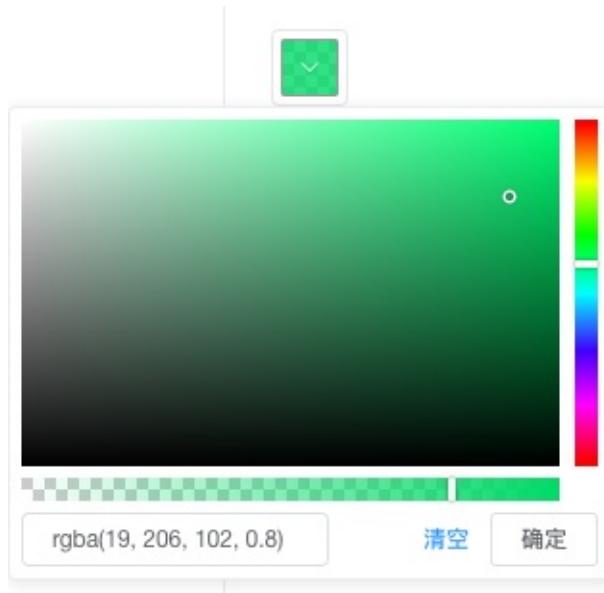
使用 `v-model` 与 Vue 实例中的一个变量进行双向绑定，绑定的变量需要是字符串类型。

```
1. <div class="block">
2.   <span class="demonstration">有默认值</span>
3.   <el-color-picker v-model="color1"></el-color-picker>
4. </div>
5. <div class="block">
6.   <span class="demonstration">无默认值</span>
7.   <el-color-picker v-model="color2"></el-color-picker>
8. </div>
9.
10. <script>
11.   export default {
12.     data() {
13.       return {
14.         color1: '#409EFF',
15.         color2: null
16.       }
17.     }
18.   }
19. </script>
```

```
17.      }
18.    };
19.  </script>
```

显示代码

选择透明度

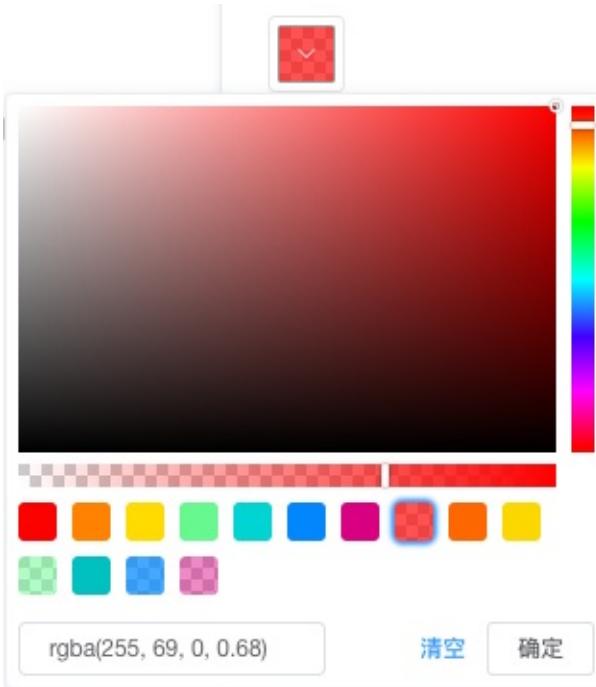


ColorPicker 支持普通颜色，也支持带 Alpha 通道的颜色，通过 `show-alpha` 属性即可控制是否支持透明度的选择。

```
1.  <el-color-picker v-model="color3" show-alpha></el-color-picker>
2.
3. <script>
4.   export default {
5.     data() {
6.       return {
7.         color3: 'rgba(19, 206, 102, 0.8)'
8.       }
9.     }
10.  };
11. </script>
```

显示代码

预定义颜色



ColorPicker 支持预定义颜色

```
1. <el-color-picker  
2.   v-model="color5"  
3.   show-alpha  
4.   :predefine="predefineColors">  
5. </el-color-picker>  
6.  
7. <script>  
8.   export default {  
9.     data() {  
10.       return {  
11.         color5: 'rgba(255, 69, 0, 0.68)',  
12.         predefineColors: [  
13.           '#ff4500',  
14.           '#ff8c00',  
15.           '#ffd700',  
16.           '#90ee90',  
17.           '#00ced1',  
18.           '#1e90ff',  
19.           '#c71585',  
20.           'rgba(255, 69, 0, 0.68)',  
21.           'rgb(255, 120, 0)',  
22.           'hsv(51, 100, 98)',  
23.           'hsva(120, 40, 94, 0.5)',  
24.           'hsl(181, 100%, 37%)',  
25.           'hsla(209, 100%, 56%, 0.73)'  
26.         ]  
27.       }  
28.     }  
29.   }  
30. </script>
```

```

26.           '#c7158577'
27.       ]
28.   }
29. }
30. };
31. </script>

```

不同尺寸



```

1. <el-color-picker v-model="color4"></el-color-picker>
2. <el-color-picker v-model="color4" size="medium"></el-color-picker>
3. <el-color-picker v-model="color4" size="small"></el-color-picker>
4. <el-color-picker v-model="color4" size="mini"></el-color-picker>
5.
6. <script>
7. export default {
8.   data() {
9.     return {
10.       color4: '#409EFF'
11.     }
12.   }
13. }
14. </script>

```

[显示代码](#)

Attributes

参数	说明	类型	可选值	默认值

disabled	是否禁用	boolean	—	false
size	尺寸	string	—	medium / small / mini
show-alpha	是否支持透明度选择	boolean	—	false
color-format	写入 v-model 的颜色的格式	string	hsl / hsv / hex / rgb	hex (show-alpha 为 false) / rgb (show-alpha 为 true)
popper-class	ColorPicker 下拉框的类名	string	—	—
predefine	预定义颜色	array	—	—

Events

事件名称	说明	回调参数
change	当绑定值变化时触发	当前值
active-change	面板中当前显示的颜色发生改变时触发	当前显示的颜色值

原文: <http://element-cn.eleme.io/#/zh-CN/component/color-picker>

Transfer 穿梭框

Transfer 穿梭框

基础用法



Transfer 的数据通过 `data` 属性传入。数据需要是一个对象数组，每个对象有以下属性：`key` 为数据的唯一性标识，`label` 为显示文本，`disabled` 表示该项数据是否禁止转移。目标列表中的数据项会同步到绑定至 `v-model` 的变量，值为数据项的 `key` 所组成的数据。当然，如果希望在初始状态时目标列表不为空，可以像本例一样为 `v-model` 绑定的变量赋予一个初始值。

```

1. <template>
2.   <el-transfer v-model="value1" :data="data"></el-transfer>
3. </template>
4.
5. <script>
6.   export default {
7.     data() {
8.       const generateData = _ => {
9.         const data = [];
10.        for (let i = 1; i <= 15; i++) {
11.          data.push({
12.            key: i,
13.            label: `备选项 ${i}`,
14.            disabled: i % 4 === 0

```

```

15.      });
16.    }
17.    return data;
18.  };
19.  return {
20.    data: generateData(),
21.    value1: [1, 4]
22.  };
23.}
24.};
25.</script>

```

可搜索

在数据很多的情况下，可以对数据进行搜索和过滤。



设置 `filterable` 为 `true` 即可开启搜索模式。默认情况下，若数据项的 `label` 属性包含搜索关键字，则会在搜索结果中显示。你也可以使用 `filter-method` 定义自己的搜索逻辑。`filter-method` 接收一个方法，当搜索关键字变化时，会将当前的关键字和每个数据项传给该方法。若方法返回 `true`，则会在搜索结果中显示对应的数据项。

```

1. <template>
2.   <el-transfer
3.     filterable
4.     :filter-method="filterMethod"
5.     filter-placeholder="请输入城市拼音"
6.     v-model="value2"
7.     :data="data2">

```

```
8.      </el-transfer>
9.    </template>
10.
11.   <script>
12.     export default {
13.       data() {
14.         const generateData2 = _ => {
15.           const data = [];
16.           const cities = ['上海', '北京', '广州', '深圳', '南京', '西安', '成都'];
17.           const pinyin = ['shanghai', 'beijing', 'guangzhou', 'shenzhen',
18.             'nanjing', 'xian', 'chengdu'];
19.             cities.forEach((city, index) => {
20.               data.push({
21.                 label: city,
22.                 key: index,
23.                 pinyin: pinyin[index]
24.               });
25.             });
26.           return data;
27.         };
28.         return {
29.           data2: generateData2(),
30.           value2: [],
31.           filterMethod(query, item) {
32.             return item.pinyin.indexOf(query) > -1;
33.           }
34.         };
35.       };
36.     </script>
```

可自定义

可以对列表标题文案、按钮文案、数据项的渲染函数、列表底部的勾选状态文案、列表底部的内容区等进行自定义。

使用 render-content 自定义数据项

The screenshot shows two panels: Source and Target. The Source panel has a title 'Source' and a count '2/14'. It contains a search bar and a list of items from 4 to 9. The Target panel has a title 'Target' and a count '1/1'. It contains a search bar and a list with one item checked: '1 - 备选项 1'. Below the lists are two buttons: '< 到左边' and '到右边 >'.

Source	Target
4 - 备选项 4 5 - 备选项 5 6 - 备选项 6 7 - 备选项 7 8 - 备选项 8 9 - 备选项 9	1 - 备选项 1

使用 scoped-slot 自定义数据项

This screenshot is similar to the previous one but shows items from 6 to 11. The Source panel has a title 'Source' and a count '2/14'. The Target panel has a title 'Target' and a count '1/1'. The list in the Target panel now includes '1 - 备选项 1'. The buttons are the same: '< 到左边' and '到右边 >'.

Source	Target
6 - 备选项 6 7 - 备选项 7 8 - 备选项 8 9 - 备选项 9 10 - 备选项 10 11 - 备选项 11	1 - 备选项 1

可以使用 `titles`、`button-texts`、`render-content` 和 `format` 属性分别对列表标题文案、按钮文案、数据项的渲染函数和列表顶部的勾选状态文案进行自定义。数据项的渲染还可以使用 `scoped-slot` 进行自定义。对于列表底部的内容区，提供了两个具名 slot：`left-footer` 和 `right-footer`。此外，如果希望某些数据项在初始化时就被勾选，可以使用 `left-default-checked` 和 `right-default-checked` 属性。最后，本例还展示了 `change` 事件的用法。注意：由于 jsfiddle 不支持 JSX 语法，所以使用 `render-content` 自定义数据项的例子在 jsfiddle 中无法运行。但是在实际的项目中，只要正确地配置了相关依赖，就可以正常运行。

```
1. <template>
2.   <p style="text-align: center; margin: 0 0 20px">使用 render-content 自定义数据项</p>
3.   <div style="text-align: center">
4.     <el-transfer
5.       style="text-align: left; display: inline-block"
6.       v-model="value3"
7.       filterable
8.       :left-default-checked="[2, 3]"
9.       :right-default-checked="[1]"
10.      :render-content="renderFunc"
11.      :titles="['Source', 'Target']"
12.      :button-texts="['到左边', '到右边']"
13.      :format={
14.        noChecked: '${total}',
15.        hasChecked: '${checked}/${total}'
16.      }"
17.      @change="handleChange"
18.      :data="data">
19.       <el-button class="transfer-footer" slot="left-footer" size="small">操作
</el-button>
20.       <el-button class="transfer-footer" slot="right-footer" size="small">操作
</el-button>
21.     </el-transfer>
22.   </div>
23.   <p style="text-align: center; margin: 50px 0 20px">使用 scoped-slot 自定义数据项</p>
24.   <div style="text-align: center">
25.     <el-transfer
26.       style="text-align: left; display: inline-block"
27.       v-model="value4"
28.       filterable
29.       :left-default-checked="[2, 3]"
30.       :right-default-checked="[1]"
31.       :titles="['Source', 'Target']"
32.       :button-texts="['到左边', '到右边']"
33.       :format={
34.         noChecked: '${total}',
35.         hasChecked: '${checked}/${total}'
36.       }"
37.       @change="handleChange"
38.       :data="data">
```

```
39.      <span slot-scope="{ option }">{{ option.key }} - {{ option.label }}
```

```
40.      </span>
```

```
41.      <el-button class="transfer-footer" slot="left-footer" size="small">操作
```

```
42.      </el-button>
```

```
43.      <el-button class="transfer-footer" slot="right-footer" size="small">操作
```

```
44.      </el-button>
```

```
45.    </el-transfer>
```

```
46.  </div>
```

```
47.  </template>
```

```
48.
```

```
49.  <style>
```

```
50.    .transfer-footer {
```

```
51.      margin-left: 20px;
```

```
52.      padding: 6px 5px;
```

```
53.    }
```

```
54.  </style>
```

```
55.
```

```
56.  <script>
```

```
57.  export default {
```

```
58.    data() {
```

```
59.      const generateData = _ => {
```

```
60.        const data = [];
```

```
61.        for (let i = 1; i <= 15; i++) {
```

```
62.          data.push({
```

```
63.            key: i,
```

```
64.            label: `备选项 ${ i }`,
```

```
65.            disabled: i % 4 === 0
```

```
66.          });
```

```
67.        }
```

```
68.        return data;
```

```
69.      };
```

```
70.      return {
```

```
71.        data: generateData(),
```

```
72.        value3: [1],
```

```
73.        value4: [1],
```

```
74.        renderFunc(h, option) {
```

```
75.          return <span>{ option.key } - { option.label }</span>;
```

```
76.        }
```

```
77.      },
```

```
78.      methods: {
```

```

78.     handleChange(value, direction, movedKeys) {
79.       console.log(value, direction, movedKeys);
80.     }
81.   }
82. }
83. </script>

```

数据项属性别名

默认情况下，Transfer 仅能识别数据项中的 `key`、`label` 和 `disabled` 字段。如果你的数据的字段名不同，可以使用 `props` 属性为它们设置别名。



本例中的数据源没有 `key` 和 `label` 字段，在功能上与它们相同的字段名为 `value` 和 `desc`。因此可以使用 `props` 属性为 `key` 和 `label` 设置别名。

```

1.  <template>
2.    <el-transfer
3.      v-model="value5"
4.      :props="{
5.        key: 'value',
6.        label: 'desc'
7.      }"
8.      :data="data3">
9.    </el-transfer>
10.   </template>
11.
12.  <script>
13.    export default {

```

```

14.     data() {
15.       const generateData3 = _ => {
16.         const data = [];
17.         for (let i = 1; i <= 15; i++) {
18.           data.push({
19.             value: i,
20.             desc: `备选项 ${i}`,
21.             disabled: i % 4 === 0
22.           });
23.         }
24.       return data;
25.     };
26.     return {
27.       data3: generateData3(),
28.       value5: []
29.     };
30.   }
31. }
32. </script>

```

Attributes

参数	说明	类型	可选值	默认值
data	Transfer 的数据源	array[{ key, label, disabled }]	-	[]
filterable	是否可搜索	boolean	-	false
filter-placeholder	搜索框占位符	string	-	请输入搜索内容
filter-method	自定义搜索方法	function	-	-
target-order	右侧列表元素的排序策略：若为 original，则保持与数据源相同的顺序；若为 push，则新加入的元素排在最后；若为 unshift，则新加入的元素排在最前	string	original / push / unshift	original
titles	自定义列表标题	array	-	['列表 1', '列表 2']
button-texts	自定义按钮文案	array	-	[]
render-content	自定义数据项渲染函数	function(h, option)	-	-

format	列表顶部勾选状态文案	object{noChecked, hasChecked}	-	{ noChecked: '\${checked}/\${total}', hasChecked: '\${checked}/\${total}' }
props	数据源的字段别名	object{key, label, disabled}	-	-
left-default-checked	初始状态下左侧列表的已勾选项的 key 数组	array	-	[]
right-default-checked	初始状态下右侧列表的已勾选项的 key 数组	array	-	[]

Slot

name	说明
left-footer	左侧列表底部的内容
right-footer	右侧列表底部的内容

Scoped Slot

name	说明
-	自定义数据项的内容，参数为 { option }

Methods

方法名	说明	参数
clearQuery	清空某个面板的搜索关键词	'left' / 'right'，指定需要清空的面板

Events

事件名称	说明	回调参数
change	右侧列表元素变化时触发	当前值、数据移动的方向 ('left' / 'right')、发生移动的数据 key 数组
left-check-change	左侧列表元素被用户选中 / 取消选中时触发	当前被选中的元素的 key 数组、选中状态发生变化的元素的 key 数组
right-check-change	右侧列表元素被用户选中 / 取消选中时触发	当前被选中的元素的 key 数组、选中状态发生变化的元素的 key 数组

原文：<http://element-cn.eleme.io/#/zh-CN/component/transfer>

Form 表单

Form 表单

由输入框、选择器、单选框、多选框等控件组成，用以收集、校验、提交数据

典型表单

包括各种表单项，比如输入框、选择器、开关、单选框、多选框等。

活动名称

活动区域

请选择活动区域

活动时间

选择日期

选择时间

即时配送

活动性质

美食/餐厅线上活动 地推活动

线下主题活动 单纯品牌曝光

特殊资源

线上品牌商赞助 线下场地免费

活动形式

立即创建 取消

在 Form 组件中，每一个表单域由一个 Form-Item 组件构成，表单域中可以放置各种类型的表单控件，包括 Input、Select、Checkbox、Radio、Switch、DatePicker、TimePicker

```

1. <el-form ref="form" :model="form" label-width="80px">
2.   <el-form-item label="活动名称">
3.     <el-input v-model="form.name"></el-input>
4.   </el-form-item>

```

```
5.   <el-form-item label="活动区域">
6.     <el-select v-model="form.region" placeholder="请选择活动区域">
7.       <el-option label="区域一" value="shanghai"></el-option>
8.       <el-option label="区域二" value="beijing"></el-option>
9.     </el-select>
10.    </el-form-item>
11.    <el-form-item label="活动时间">
12.      <el-col :span="11">
13.        <el-date-picker type="date" placeholder="选择日期" v-model="form.date1"
14.          style="width: 100%;"></el-date-picker>
15.      </el-col>
16.      <el-col class="line" :span="2">-</el-col>
17.      <el-col :span="11">
18.        <el-time-picker type="fixed-time" placeholder="选择时间" v-
19.          model="form.date2" style="width: 100%;"></el-time-picker>
20.      </el-col>
21.    </el-form-item>
22.    <el-form-item label="即时配送">
23.      <el-switch v-model="form.delivery"></el-switch>
24.    </el-form-item>
25.    <el-form-item label="活动性质">
26.      <el-checkbox-group v-model="form.type">
27.        <el-checkbox label="美食/餐厅线上活动" name="type"></el-checkbox>
28.        <el-checkbox label="地推活动" name="type"></el-checkbox>
29.        <el-checkbox label="线下主题活动" name="type"></el-checkbox>
30.        <el-checkbox label="单纯品牌曝光" name="type"></el-checkbox>
31.      </el-checkbox-group>
32.    </el-form-item>
33.    <el-form-item label="特殊资源">
34.      <el-radio-group v-model="form.resource">
35.        <el-radio label="线上品牌商赞助"></el-radio>
36.        <el-radio label="线下场地免费"></el-radio>
37.      </el-radio-group>
38.    </el-form-item>
39.    <el-form-item label="活动形式">
40.      <el-input type="textarea" v-model="form.desc"></el-input>
41.    </el-form-item>
42.    <el-form-item>
43.      <el-button type="primary" @click="onSubmit">立即创建</el-button>
44.      <el-button>取消</el-button>
45.    </el-form-item>
46.  </el-form>
```

```

45. <script>
46.   export default {
47.     data() {
48.       return {
49.         form: {
50.           name: '',
51.           region: '',
52.           date1: '',
53.           date2: '',
54.           delivery: false,
55.           type: [],
56.           resource: '',
57.           desc: ''
58.         }
59.       }
60.     },
61.     methods: {
62.       onSubmit() {
63.         console.log('submit!');
64.       }
65.     }
66.   }
67. </script>

```

W3C 标准中有如下规定：

When there is only one single-line text input field in a form, the user agent should accept Enter in that field as a request to submit the form.

即：当一个 form 元素中只有一个输入框时，在该输入框中按下回车应提交该表单。如果希望阻止这一默认行为，可以在 `<el-form>` 标签上添加 `@submit.native.prevent`。

行内表单

当垂直方向空间受限且表单较简单时，可以在一行内放置表单。



设置 `inline` 属性可以让表单域变为行内的表单域

```
1. <el-form :inline="true" :model="formInline" class="demo-form-inline">
```

```
2.  <el-form-item label="审批人">
3.    <el-input v-model="formInline.user" placeholder="审批人"></el-input>
4.  </el-form-item>
5.  <el-form-item label="活动区域">
6.    <el-select v-model="formInline.region" placeholder="活动区域">
7.      <el-option label="区域一" value="shanghai"></el-option>
8.      <el-option label="区域二" value="beijing"></el-option>
9.    </el-select>
10.   </el-form-item>
11.   <el-form-item>
12.     <el-button type="primary" @click="onSubmit">查询</el-button>
13.   </el-form-item>
14. </el-form>
15. <script>
16. export default {
17.   data() {
18.     return {
19.       formInline: {
20.         user: '',
21.         region: ''
22.       }
23.     }
24.   },
25.   methods: {
26.     onSubmit() {
27.       console.log('submit!');
28.     }
29.   }
30. }
31. </script>
```

对齐方式

根据具体目标和制约因素，选择最佳的标签对齐方式。

左对齐 右对齐 顶部对齐

名称

活动区域

活动形式

通过设置 `label-position` 属性可以改变表单域标签的位置，可选值为 `top`、`left`，当设为 `top` 时标签会置于表单域的顶部

```

1. <el-radio-group v-model="labelPosition" size="small">
2.   <el-radio-button label="left">左对齐</el-radio-button>
3.   <el-radio-button label="right">右对齐</el-radio-button>
4.   <el-radio-button label="top">顶部对齐</el-radio-button>
5. </el-radio-group>
6. <div style="margin: 20px;"></div>
7. <el-form :label-position="labelPosition" label-width="80px"
   :model="formLabelAlign">
8.   <el-form-item label="名称">
9.     <el-input v-model="formLabelAlign.name"></el-input>
10.  </el-form-item>
11.  <el-form-item label="活动区域">
12.    <el-input v-model="formLabelAlign.region"></el-input>
13.  </el-form-item>
14.  <el-form-item label="活动形式">
15.    <el-input v-model="formLabelAlign.type"></el-input>
16.  </el-form-item>
17. </el-form>
18. <script>
19.   export default {
20.     data() {
21.       return {
22.         labelPosition: 'right',
23.         formLabelAlign: {
24.           name: '',
25.           region: '',
26.           type: ''
27.         }
28.       }
29.     }
30.   }
31. </script>
32. <style>
33.   .el-form-item__label {
34.     width: 80px !important;
35.   }
36. </style>

```

```

28.      };
29.    }
30.  }
31. </script>

```

表单验证

在防止用户犯错的前提下，尽可能让用户更早地发现并纠正错误。

* 活动名称

* 活动区域

请选择活动区域

* 活动时间

即时配送

* 活动性质

美食/餐厅线上活动
 地推活动
 线下主题活动
 单纯品牌曝光

* 特殊资源

线上品牌商赞助
 线下场地免费

* 活动形式

Form 组件提供了表单验证的功能，只需要通过 `rules` 属性传入约定的验证规则，并将 Form-Item 的 `prop` 属性设置为需校验的字段名即可。校验规则参见 [async-validator](#)

```

1.  <el-form :model="ruleForm" :rules="rules" ref="ruleForm" label-width="100px"
2.    class="demo-ruleForm">
3.    <el-form-item label="活动名称" prop="name">
4.      <el-input v-model="ruleForm.name"></el-input>
5.    </el-form-item>
6.    <el-form-item label="活动区域" prop="region">
7.      <el-select v-model="ruleForm.region" placeholder="请选择活动区域">

```

```
7.      <el-option label="区域一" value="shanghai"></el-option>
8.      <el-option label="区域二" value="beijing"></el-option>
9.    </el-select>
10.   </el-form-item>
11.   <el-form-item label="活动时间" required>
12.     <el-col :span="11">
13.       <el-form-item prop="date1">
14.         <el-date-picker type="date" placeholder="选择日期" v-
model="ruleForm.date1" style="width: 100%;"></el-date-picker>
15.       </el-form-item>
16.     </el-col>
17.     <el-col class="line" :span="2">-</el-col>
18.     <el-col :span="11">
19.       <el-form-item prop="date2">
20.         <el-time-picker type="fixed-time" placeholder="选择时间" v-
model="ruleForm.date2" style="width: 100%;"></el-time-picker>
21.       </el-form-item>
22.     </el-col>
23.   </el-form-item>
24.   <el-form-item label="即时配送" prop="delivery">
25.     <el-switch v-model="ruleForm.delivery"></el-switch>
26.   </el-form-item>
27.   <el-form-item label="活动性质" prop="type">
28.     <el-checkbox-group v-model="ruleForm.type">
29.       <el-checkbox label="美食/餐厅线上活动" name="type"></el-checkbox>
30.       <el-checkbox label="地推活动" name="type"></el-checkbox>
31.       <el-checkbox label="线下主题活动" name="type"></el-checkbox>
32.       <el-checkbox label="单纯品牌曝光" name="type"></el-checkbox>
33.     </el-checkbox-group>
34.   </el-form-item>
35.   <el-form-item label="特殊资源" prop="resource">
36.     <el-radio-group v-model="ruleForm.resource">
37.       <el-radio label="线上品牌商赞助"></el-radio>
38.       <el-radio label="线下场地免费"></el-radio>
39.     </el-radio-group>
40.   </el-form-item>
41.   <el-form-item label="活动形式" prop="desc">
42.     <el-input type="textarea" v-model="ruleForm.desc"></el-input>
43.   </el-form-item>
44.   <el-form-item>
45.     <el-button type="primary" @click="submitForm('ruleForm')">立即创建</el-
button>
```

```
46.      <el-button @click="resetForm('ruleForm')">重置</el-button>
47.    </el-form-item>
48.  </el-form>
49.  <script>
50.    export default {
51.      data() {
52.        return {
53.          ruleForm: {
54.            name: '',
55.            region: '',
56.            date1: '',
57.            date2: '',
58.            delivery: false,
59.            type: [],
60.            resource: '',
61.            desc: ''
62.          },
63.          rules: {
64.            name: [
65.              { required: true, message: '请输入活动名称', trigger: 'blur' },
66.              { min: 3, max: 5, message: '长度在 3 到 5 个字符', trigger: 'blur' }
67.            ],
68.            region: [
69.              { required: true, message: '请选择活动区域', trigger: 'change' }
70.            ],
71.            date1: [
72.              { type: 'date', required: true, message: '请选择日期', trigger:
73.                'change' }
74.            ],
75.            date2: [
76.              { type: 'date', required: true, message: '请选择时间', trigger:
77.                'change' }
78.            ],
79.            type: [
80.              { type: 'array', required: true, message: '请至少选择一个活动性质',
81.                trigger: 'change' }
82.            ],
83.            resource: [
84.              { required: true, message: '请选择活动资源', trigger: 'change' }
85.            ],
86.            desc: [
87.              { required: true, message: '请填写活动形式', trigger: 'blur' }
88.            ]
89.          }
90.        }
91.      }
92.    }
93.  
```

```

55.      ]
56.    }
57.  ];
58. },
59. methods: {
60.   submitForm(formName) {
61.     this.$refs[formName].validate(valid) => {
62.       if (valid) {
63.         alert('submit!');
64.       } else {
65.         console.log('error submit!!');
66.         return false;
67.       }
68.     });
69.   },
70.   resetForm(formName) {
71.     this.$refs[formName].resetFields();
72.   }
73. }
74. }
75. </script>

```

自定义校验规则

这个例子中展示了如何使用自定义验证规则来完成密码的二次验证。

本例还使用 `status-icon` 属性为输入框添加了表示校验结果的反馈图标。

```

1.  <el-form :model="ruleForm2" status-icon :rules="rules2" ref="ruleForm2" label-width="100px" class="demo-ruleForm">
2.    <el-form-item label="密码" prop="pass">

```

```
3.      <el-input type="password" v-model="ruleForm2.pass" auto-complete="off">
4.      </el-input>
5.      <el-form-item label="确认密码" prop="checkPass">
6.          <el-input type="password" v-model="ruleForm2.checkPass" auto-
    complete="off"></el-input>
7.      </el-form-item>
8.      <el-form-item label="年龄" prop="age">
9.          <el-input v-model.number="ruleForm2.age"></el-input>
10.     </el-form-item>
11.     <el-form-item>
12.         <el-button type="primary" @click="submitForm('ruleForm2')">提交</el-button>
13.         <el-button @click="resetForm('ruleForm2')">重置</el-button>
14.     </el-form-item>
15. </el-form>
16. <script>
17. export default {
18.     data() {
19.         var checkAge = (rule, value, callback) => {
20.             if (!value) {
21.                 return callback(new Error('年龄不能为空'));
22.             }
23.             setTimeout(() => {
24.                 if (!Number.isInteger(value)) {
25.                     callback(new Error('请输入数字值'));
26.                 } else {
27.                     if (value < 18) {
28.                         callback(new Error('必须年满18岁'));
29.                     } else {
30.                         callback();
31.                     }
32.                 }
33.             }, 1000);
34.         };
35.         var validatePass = (rule, value, callback) => {
36.             if (value === '') {
37.                 callback(new Error('请输入密码'));
38.             } else {
39.                 if (this.ruleForm2.checkPass !== '') {
40.                     this.$refs.ruleForm2.validateField('checkPass');
41.                 }
42.                 callback();
43.             }
44.         };
45.     }
46.     rules: {
47.         pass: [
48.             { required: true, message: '请输入密码' },
49.             { min: 6, max: 12, message: '长度在 6 到 12 个字符' }
50.         ],
51.         checkPass: [
52.             { required: true, message: '请再次输入密码' },
53.             { validator: validatePass, trigger: 'blur' }
54.         ],
55.         age: [
56.             { required: true, message: '年龄不能为空' },
57.             { type: 'number', message: '年龄必须为数字值' },
58.             { validator: checkAge, trigger: 'blur' }
59.         ]
60.     }
61.   };
62. }
63. 
```

```
43.         }
44.     };
45.     var validatePass2 = (rule, value, callback) => {
46.       if (value === '') {
47.         callback(new Error('请再次输入密码'));
48.       } else if (value !== this.ruleForm2.pass) {
49.         callback(new Error('两次输入密码不一致!'));
50.       } else {
51.         callback();
52.       }
53.     };
54.     return {
55.       ruleForm2: {
56.         pass: '',
57.         checkPass: '',
58.         age: ''
59.       },
60.       rules2: {
61.         pass: [
62.           { validator: validatePass, trigger: 'blur' }
63.         ],
64.         checkPass: [
65.           { validator: validatePass2, trigger: 'blur' }
66.         ],
67.         age: [
68.           { validator: checkAge, trigger: 'blur' }
69.         ]
70.       }
71.     };
72.   },
73.   methods: {
74.     submitForm(formName) {
75.       this.$refs[formName].validate(valid) => {
76.         if (valid) {
77.           alert('submit!');
78.         } else {
79.           console.log('error submit!!!');
80.           return false;
81.         }
82.       });
83.     },
84.     resetForm(formName) {
```

```

85.         this.$refs[formName].resetFields();
86.     }
87.   }
88. }
89. </script>

```

动态增减表单项

* 邮箱

* 域名0

删除

提交 新增域名 重置

除了在 Form 组件上一次性传递所有的验证规则外还可以在单个的表单域上传递属性的验证规则

```

<el-form :model="dynamicValidateForm" ref="dynamicValidateForm" label-width="100px" class="demo-dynamic">
  <el-form-item
    prop="email"
    label="邮箱"
    :rules="[
      { required: true, message: '请输入邮箱地址', trigger: 'blur' },
      { type: 'email', message: '请输入正确的邮箱地址', trigger: ['blur', 'change'] }
    ]"
  >
    <el-input v-model="dynamicValidateForm.email"></el-input>
  </el-form-item>
  <el-form-item
    v-for="(domain, index) in dynamicValidateForm.domains"
    :label="'域名' + index"
    :key="domain.key"
    :prop="'domains.' + index + '.value'"
    :rules="{
      required: true, message: '域名不能为空', trigger: 'blur'
    }"
  >
    <el-input v-model="domain.value"></el-input><el-button>

```

```
@click.prevent="removeDomain(domain)">删除</el-button>
  </el-form-item>
  <el-form-item>
    <el-button type="primary" @click="submitForm('dynamicValidateForm')">提交</el-button>
    <el-button @click="addDomain">新增域名</el-button>
    <el-button @click="resetForm('dynamicValidateForm')">重置</el-button>
  </el-form-item>
</el-form>
<script>
  export default {
    data() {
      return {
        dynamicValidateForm: {
          domains: [{ value: '' }],
          email: ''
        }
      };
    },
    methods: {
      submitForm(formName) {
        this.$refs[formName].validate((valid) => {
          if (valid) {
            alert('submit!');
          } else {
            console.log('error submit!!');
            return false;
          }
        });
      },
      resetForm(formName) {
        this.$refs[formName].resetFields();
      },
      removeDomain(item) {
        var index = this.dynamicValidateForm.domains.indexOf(item)
        if (index !== -1) {
          this.dynamicValidateForm.domains.splice(index, 1)
        }
      },
      addDomain() {
```

```

        this.dynamicValidateForm.domains.push({
          value: '',
          key: Date.now()
        });
      }
    }
  }
</script>

```

数字类型验证

* 年龄

提交 重置

数字类型的验证需要在 `v-model` 处加上 `.number` 的修饰符，这是 `Vue` 自身提供的用于将绑定值转化为 `number` 类型的修饰符。

```

<el-form :model="numberValidateForm" ref="numberValidateForm" label-width="100px" class="demo-ruleForm">
  <el-form-item
    label="年龄"
    prop="age"
    :rules="[
      { required: true, message: '年龄不能为空' },
      { type: 'number', message: '年龄必须为数字值' }
    ]"
  >
    <el-input type="age" v-model.number="numberValidateForm.age" auto-complete="off"></el-input>
  </el-form-item>
  <el-form-item>
    <el-button type="primary" @click="submitForm('numberValidateForm')">提交</el-button>
    <el-button @click="resetForm('numberValidateForm')">重置</el-button>
  </el-form-item>
</el-form>
<script>
  export default {
    data() {

```

```
return {
  numberValidateForm: {
    age: ''
  }
},
methods: {
  submitForm(formName) {
    this.$refs[formName].validate((valid) => {
      if (valid) {
        alert('submit!');
      } else {
        console.log('error submit!!');
        return false;
      }
    });
  },
  resetForm(formName) {
    this.$refs[formName].resetFields();
  }
}
</script>
```

嵌套在 `el-form-item` 中的 `el-form-item` 标签宽度默认为零，不会继承 `el-form` 的 `label-width`。如果需要可以为其单独设置 `label-width` 属性。

表单内组件尺寸控制

通过设置 Form 上的 `size` 属性可以使该表单内所有可调节大小的组件继承该尺寸。Form-Item 也具有该属性。

活动名称

活动区域

请选择活动区域

活动时间

选择日期

选择时间

活动性质

美食/餐厅线上活动

地推活动

线下主题活动

特殊资源

线上品牌商赞助

线下场地免费

立即创建

取消

如果希望某个表单项或某个表单组件的尺寸不同于 Form 上的 `size` 属性，直接为这个表单项或表单组件设置自己的 `size` 即可。

```
<el-form ref="form" :model="sizeForm" label-width="80px" size="mini">
  <el-form-item label="活动名称">
    <el-input v-model="sizeForm.name"></el-input>
  </el-form-item>
  <el-form-item label="活动区域">
    <el-select v-model="sizeForm.region" placeholder="请选择活动区域">
      <el-option label="区域一" value="shanghai"></el-option>
      <el-option label="区域二" value="beijing"></el-option>
    </el-select>
  </el-form-item>
  <el-form-item label="活动时间">
    <el-col :span="11">
      <el-date-picker type="date" placeholder="选择日期" v-model="sizeForm.date1" style="width: 100%;"></el-date-picker>
    </el-col>
    <el-col class="line" :span="2">-</el-col>
    <el-col :span="11">
      <el-time-picker type="fixed-time" placeholder="选择时间" v-model="sizeForm.date2" style="width: 100%;"></el-time-picker>
    </el-col>
  </el-form-item>
  <el-form-item label="活动性质">
    <el-checkbox-group v-model="sizeForm.type">
      <el-checkbox-button label="美食/餐厅线上活动" name="type"></el-checkbox-button>
      <el-checkbox-button label="地推活动" name="type"></el-checkbox-button>
    </el-checkbox-group>
  </el-form-item>
</el-form>
```

```
button>
    <el-checkbox-button label="线下主题活动" name="type"></el-checkbox-
button>
</el-checkbox-group>
</el-form-item>
<el-form-item label="特殊资源">
    <el-radio-group v-model="sizeForm.resource" size="medium">
        <el-radio border label="线上品牌商赞助"></el-radio>
        <el-radio border label="线下场地免费"></el-radio>
    </el-radio-group>
</el-form-item>
<el-form-item size="large">
    <el-button type="primary" @click="onSubmit">立即创建</el-button>
    <el-button>取消</el-button>
</el-form-item>
</el-form>

<script>
export default {
  data() {
    return {
      sizeForm: {
        name: '',
        region: '',
        date1: '',
        date2: '',
        delivery: false,
        type: [],
        resource: '',
        desc: ''
      }
    };
  },
  methods: {
    onSubmit() {
      console.log('submit!');
    }
  }
};
</script>
```

Form Attributes

参数	说明	类型	可选值	默认值
model	表单数据对象	object	—	—
rules	表单验证规则	object	—	—
inline	行内表单模式	boolean	—	false
label-position	表单域标签的位置	string	right/left/top	right
label-width	表单域标签的宽度，作为 Form 直接子元素的 form-item 会继承该值	string	—	—
label-suffix	表单域标签的后缀	string	—	—
show-message	是否显示校验错误信息	boolean	—	true
inline-message	是否以行内形式展示校验信息	boolean	—	false
status-icon	是否在输入框中显示校验结果反馈图标	boolean	—	false
validate-on-rule-change	是否在 rules 属性改变后立即触发一次验证	boolean	—	true
size	用于控制该表单内组件的尺寸	string	medium / small / mini	—
disabled	是否禁用该表单内的所有组件。若设置为 true，则表单内组件上的 disabled 属性不再生效	boolean	—	false

Form Methods

方法名	说明	参数
validate	对整个表单进行校验的方法，参数为一个回调函数。该回调函数会在校验结束后被调用，并传入两个参数：是否校验成功和未通过校验的字段。若不传入回调函数，则会返回一个 promise	Function(callback: Function(boolean, object))
validateField	对部分表单字段进行校验的方法	Function(prop: string, callback: Function(errorMessage: string))
resetFields	对整个表单进行重置，将所有字段值重置为初始值并移除校验结果	—
clearValidate	移除表单项的校验结果。传入待移除的表单项的 prop 属性组成的数组，如不传则移除整个表单的校验结果	Function(props: array)

Form Events

事件名称	说明	回调参数
validate	任一表单项被校验后触发	被校验的表单项 prop 值，校验是否通过

Form-Item Attributes

参数	说明	类型	可选值	默认值
prop	表单域 model 字段，在使用 validate、resetFields 方法的情况下，该属性是必填的	string	传入 Form 组件的 model 中的字段	-
label	标签文本	string	-	-
label-width	表单域标签的宽度，例如 '50px'	string	-	-
required	是否必填，如不设置，则会根据校验规则自动生成	boolean	-	false
rules	表单验证规则	object	-	-
error	表单域验证错误信息，设置该值会使表单验证状态变为 error，并显示该错误信息	string	-	-
show-message	是否显示校验错误信息	boolean	-	true
inline-message	以行内形式展示校验信息	boolean	-	false
size	用于控制该表单域下组件的尺寸	string	medium / small / mini	-

Form-Item Slot

name	说明
-	Form Item 的内容
label	标签文本的内容

Form-Item Methods

方法名	说明	参数
resetField	对该表单项进行重置，将其值重置为初始值并移除校验结果	-
clearValidate	移除该表单项的校验结果	-

原文：<http://element-cn.eleme.io/#/zh-CN/component/form>

Data 数据组件

- [Table 表格](#)
- [Tag 标签](#)
- [Progress 进度条](#)
- [Tree 树形控件](#)
- [Pagination 分页](#)
- [Badge 标记](#)

Table 表格

Table 表格

用于展示多条结构类似的数据，可对数据进行排序、筛选、对比或其他自定义操作。

基础表格

基础的表格展示用法。

日期	姓名	地址
2016-05-03	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-04	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-01	王小虎	上海市普陀区金沙江路 1518 弄

当 `el-table` 元素中注入 `data` 对象数组后，在 `el-table-column` 中用 `prop` 属性来对应对象中的键名即可填入数据，用 `label` 属性来定义表格的列名。可以使用 `width` 属性来定义列宽。

```

1.   <template>
2.     <el-table
3.       :data="tableData"
4.       style="width: 100%">
5.         <el-table-column
6.           prop="date"
7.           label="日期"
8.           width="180">
9.         </el-table-column>
10.        <el-table-column
11.          prop="name"
12.          label="姓名"
13.          width="180">
14.        </el-table-column>
15.        <el-table-column
16.          prop="address"

```

```
17.      label="地址">
18.    </el-table-column>
19.  </el-table>
20. </template>
21.
22. <script>
23.   export default {
24.     data() {
25.       return {
26.         tableData: [
27.           date: '2016-05-02',
28.           name: '王小虎',
29.           address: '上海市普陀区金沙江路 1518 弄'
30.         }, {
31.           date: '2016-05-04',
32.           name: '王小虎',
33.           address: '上海市普陀区金沙江路 1517 弄'
34.         }, {
35.           date: '2016-05-01',
36.           name: '王小虎',
37.           address: '上海市普陀区金沙江路 1519 弄'
38.         }, {
39.           date: '2016-05-03',
40.           name: '王小虎',
41.           address: '上海市普陀区金沙江路 1516 弄'
42.         }]
43.       }
44.     }
45.   }
46. </script>
```

带斑马纹表格

使用带斑马纹的表格，可以更容易区分出不同行的数据。

日期	姓名	地址
2016-05-03	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-04	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-01	王小虎	上海市普陀区金沙江路 1518 弄

`stripe` 属性可以创建带斑马纹的表格。它接受一个 `Boolean`，默认为 `false`，设置为 `true` 即为启用。

```

1. <template>
2.   <el-table
3.     :data="tableData"
4.     stripe
5.     style="width: 100%"
6.     <el-table-column
7.       prop="date"
8.       label="日期"
9.       width="180">
10.    </el-table-column>
11.    <el-table-column
12.      prop="name"
13.      label="姓名"
14.      width="180">
15.    </el-table-column>
16.    <el-table-column
17.      prop="address"
18.      label="地址">
19.    </el-table-column>
20.  </el-table>
21. </template>
22.
23. <script>
24.   export default {
25.     data() {
26.       return {
27.         tableData: [
28.           date: '2016-05-02',

```

```

29.         name: '王小虎',
30.         address: '上海市普陀区金沙江路 1518 弄'
31.     },
32.     date: '2016-05-04',
33.     name: '王小虎',
34.     address: '上海市普陀区金沙江路 1517 弄'
35.   },
36.   date: '2016-05-01',
37.   name: '王小虎',
38.   address: '上海市普陀区金沙江路 1519 弄'
39. },
40. date: '2016-05-03',
41. name: '王小虎',
42. address: '上海市普陀区金沙江路 1516 弄'
43. ]
44. }
45. }
46. }
47. </script>

```

带边框表格

日期	姓名	地址
2016-05-03	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-04	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-01	王小虎	上海市普陀区金沙江路 1518 弄

默认情况下，Table 组件是不具有竖直方向的边框的，如果需要，可以使用 `border` 属性，它接受一个 `Boolean`，设置为 `true` 即可启用。

```

1. <template>
2.   <el-table
3.     :data="tableData"
4.     border
5.     style="width: 100%">
6.     <el-table-column
7.       prop="date"

```

```
8.      label="日期"
9.      width="180">
10.     </el-table-column>
11.     <el-table-column
12.       prop="name"
13.       label="姓名"
14.       width="180">
15.     </el-table-column>
16.     <el-table-column
17.       prop="address"
18.       label="地址">
19.     </el-table-column>
20.   </el-table>
21. </template>
22.
23. <script>
24.   export default {
25.     data() {
26.       return {
27.         tableData: [
28.           { date: '2016-05-02',
29.             name: '王小虎',
30.             address: '上海市普陀区金沙江路 1518 弄'
31.           },
32.           { date: '2016-05-04',
33.             name: '王小虎',
34.             address: '上海市普陀区金沙江路 1517 弄'
35.           },
36.           { date: '2016-05-01',
37.             name: '王小虎',
38.             address: '上海市普陀区金沙江路 1519 弄'
39.           },
40.           { date: '2016-05-03',
41.             name: '王小虎',
42.             address: '上海市普陀区金沙江路 1516 弄'
43.           }
44.         ]
45.       }
46.     }
47.   </script>
```

带状态表格

可将表格内容 `highlight` 显示，方便区分「成功、信息、警告、危险」等内容。

日期	姓名	地址
2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-04	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-01	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-03	王小虎	上海市普陀区金沙江路 1518 弄

可以通过指定 `Table` 组件的 `row-class-name` 属性来为 `Table` 中的某一行添加 `class`，表明该行处于某种状态。

```

1. <template>
2.   <el-table
3.     :data="tableData2"
4.     style="width: 100%"
5.     :row-class-name="tableRowClassName">
6.       <el-table-column
7.         prop="date"
8.         label="日期"
9.         width="180">
10.      </el-table-column>
11.      <el-table-column
12.        prop="name"
13.        label="姓名"
14.        width="180">
15.      </el-table-column>
16.      <el-table-column
17.        prop="address"
18.        label="地址">
19.      </el-table-column>
20.    </el-table>
21.  </template>
22.
23. <style>
24.   .el-table .warning-row {
25.     background: oldlace;
26.   }
27.
```

```
28.     .el-table .success-row {
29.       background: #f0f9eb;
30.     }
31.   </style>
32.
33.   <script>
34.     export default {
35.       methods: {
36.         tableRowClassName({row, rowIndex}) {
37.           if (rowIndex === 1) {
38.             return 'warning-row';
39.           } else if (rowIndex === 3) {
40.             return 'success-row';
41.           }
42.           return '';
43.         }
44.       },
45.       data() {
46.         return {
47.           tableData2: [
48.             {
49.               date: '2016-05-02',
50.               name: '王小虎',
51.               address: '上海市普陀区金沙江路 1518 弄',
52.             },
53.             {
54.               date: '2016-05-04',
55.               name: '王小虎',
56.               address: '上海市普陀区金沙江路 1518 弄'
57.             },
58.             {
59.               date: '2016-05-01',
60.               name: '王小虎',
61.               address: '上海市普陀区金沙江路 1518 弄',
62.             },
63.             {
64.               date: '2016-05-03',
65.               name: '王小虎',
66.               address: '上海市普陀区金沙江路 1518 弄'
67.             }
68.           ]
69.         }
70.       }
71.     }
72.   </script>
```

固定表头

纵向内容过多时，可选择固定表头。

日期	姓名	地址
2010-03-04	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-01	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-08	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-06	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-07	王小虎	上海市普陀区金沙江路 1518 弄

只要在 `el-table` 元素中定义了 `height` 属性，即可实现固定表头的表格，而不需要额外的代码。

```

1. <template>
2.   <el-table
3.     :data="tableData3"
4.     height="250"
5.     border
6.     style="width: 100%">
7.       <el-table-column
8.         prop="date"
9.         label="日期"
10.        width="180">
11.      </el-table-column>
12.      <el-table-column
13.        prop="name"
14.        label="姓名"
15.        width="180">
16.      </el-table-column>
17.      <el-table-column
18.        prop="address"
19.        label="地址">
20.      </el-table-column>
21.    </el-table>
22.  </template>
23.
24. <script>
25.   export default {
26.     data() {
27.       return {
28.         tableData3: [

```

```
29.         date: '2016-05-03',
30.         name: '王小虎',
31.         address: '上海市普陀区金沙江路 1518 弄'
32.     }, {
33.         date: '2016-05-02',
34.         name: '王小虎',
35.         address: '上海市普陀区金沙江路 1518 弄'
36.     }, {
37.         date: '2016-05-04',
38.         name: '王小虎',
39.         address: '上海市普陀区金沙江路 1518 弄'
40.     }, {
41.         date: '2016-05-01',
42.         name: '王小虎',
43.         address: '上海市普陀区金沙江路 1518 弄'
44.     }, {
45.         date: '2016-05-08',
46.         name: '王小虎',
47.         address: '上海市普陀区金沙江路 1518 弄'
48.     }, {
49.         date: '2016-05-06',
50.         name: '王小虎',
51.         address: '上海市普陀区金沙江路 1518 弄'
52.     }, {
53.         date: '2016-05-07',
54.         name: '王小虎',
55.         address: '上海市普陀区金沙江路 1518 弄'
56.     }]
57. }
58. }
59. }
60. </script>
```

固定列

横向内容过多时，可选择固定列。

日期	姓名	省份	市区	地址	操作
2016-05-03	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄	查看 编辑
2016-05-02	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄	查看 编辑
2016-05-04	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄	查看 编辑
2016-05-01	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄	查看 编辑

固定列需要使用 `fixed` 属性，它接受 Boolean 值或者 `left` `right`，表示左边固定还是右边固定。

```

1. <template>
2.   <el-table
3.     :data="tableData"
4.     border
5.     style="width: 100%">
6.       <el-table-column
7.         fixed
8.         prop="date"
9.         label="日期"
10.        width="150">
11.      </el-table-column>
12.      <el-table-column
13.        prop="name"
14.        label="姓名"
15.        width="120">
16.      </el-table-column>
17.      <el-table-column
18.        prop="province"
19.        label="省份"
20.        width="120">
21.      </el-table-column>
22.      <el-table-column
23.        prop="city"
24.        label="市区"
25.        width="120">
26.      </el-table-column>
27.      <el-table-column
28.        prop="address"

```

```
29.     label="地址"
30.     width="300">
31.   </el-table-column>
32.   <el-table-column
33.     prop="zip"
34.     label="邮编"
35.     width="120">
36.   </el-table-column>
37.   <el-table-column
38.     fixed="right"
39.     label="操作"
40.     width="100">
41.     <template slot-scope="scope">
42.       <el-button @click="handleClick(scope.row)" type="text" size="small">查
        看</el-button>
43.       <el-button type="text" size="small">编辑</el-button>
44.     </template>
45.   </el-table-column>
46. </el-table>
47. </template>
48.
49. <script>
50.   export default {
51.     methods: {
52.       handleClick(row) {
53.         console.log(row);
54.       }
55.     },
56.
57.     data() {
58.       return {
59.         tableData: [
60.           {
61.             date: '2016-05-03',
62.             name: '王小虎',
63.             province: '上海',
64.             city: '普陀区',
65.             address: '上海市普陀区金沙江路 1518 弄',
66.             zip: 200333
67.           },
68.           {
69.             date: '2016-05-02',
```

```

70.         city: '普陀区',
71.         address: '上海市普陀区金沙江路 1518 弄',
72.         zip: 200333
73.     }, {
74.         date: '2016-05-04',
75.         name: '王小虎',
76.         province: '上海',
77.         city: '普陀区',
78.         address: '上海市普陀区金沙江路 1518 弄',
79.         zip: 200333
80.     }, {
81.         date: '2016-05-01',
82.         name: '王小虎',
83.         province: '上海',
84.         city: '普陀区',
85.         address: '上海市普陀区金沙江路 1518 弄',
86.         zip: 200333
87.     }]
88. }
89. }
90. }
91. </script>

```

固定列和表头

横纵内容过多时，可选择固定列和表头。

日期	姓名	省份	市区	地址
2016-05-02	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄
2016-05-04	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄
2016-05-01	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄
2016-05-08	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄

固定列和表头可以同时使用，只需要将上述两个属性分别设置好即可。

```

1. <template>
2.   <el-table

```

```
3.      :data="tableData3"
4.      style="width: 100%"
5.      height="250">
6.      <el-table-column
7.          fixed
8.          prop="date"
9.          label="日期"
10.         width="150">
11.     </el-table-column>
12.     <el-table-column
13.         prop="name"
14.         label="姓名"
15.         width="120">
16.     </el-table-column>
17.     <el-table-column
18.         prop="province"
19.         label="省份"
20.         width="120">
21.     </el-table-column>
22.     <el-table-column
23.         prop="city"
24.         label="市区"
25.         width="120">
26.     </el-table-column>
27.     <el-table-column
28.         prop="address"
29.         label="地址"
30.         width="300">
31.     </el-table-column>
32.     <el-table-column
33.         prop="zip"
34.         label="邮编"
35.         width="120">
36.     </el-table-column>
37.   </el-table>
38. </template>
39.
40. <script>
41.   export default {
42.     data() {
43.       return {
44.         tableData3: [{
```

```
45.         date: '2016-05-03',
46.         name: '王小虎',
47.         province: '上海',
48.         city: '普陀区',
49.         address: '上海市普陀区金沙江路 1518 弄',
50.         zip: 200333
51.     },
52.     {
53.         date: '2016-05-02',
54.         name: '王小虎',
55.         province: '上海',
56.         city: '普陀区',
57.         address: '上海市普陀区金沙江路 1518 弄',
58.         zip: 200333
59.     },
60.     {
61.         date: '2016-05-04',
62.         name: '王小虎',
63.         province: '上海',
64.         city: '普陀区',
65.         address: '上海市普陀区金沙江路 1518 弄',
66.         zip: 200333
67.     },
68.     {
69.         date: '2016-05-01',
70.         name: '王小虎',
71.         province: '上海',
72.         city: '普陀区',
73.         address: '上海市普陀区金沙江路 1518 弄',
74.         zip: 200333
75.     },
76.     {
77.         date: '2016-05-08',
78.         name: '王小虎',
79.         province: '上海',
80.         city: '普陀区',
81.         address: '上海市普陀区金沙江路 1518 弄',
82.         zip: 200333
83.     },
84.     {
85.         date: '2016-05-06',
86.         name: '王小虎',
```

```

87.         date: '2016-05-07',
88.         name: '王小虎',
89.         province: '上海',
90.         city: '普陀区',
91.         address: '上海市普陀区金沙江路 1518 弄',
92.         zip: 200333
93.     }]
94. }
95. }
96. }
97. </script>

```

流体高度

当数据量动态变化时，可以为 Table 设置一个最大高度。

日期	省份	市区	地址	操作
2016-05-01	上海	普陀区	上海市普陀区金沙江路 1518 弄	移除
2016-05-08	上海	普陀区	上海市普陀区金沙江路 1518 弄	移除
2016-05-06	上海	普陀区	上海市普陀区金沙江路 1518 弄	移除

通过设置 `max-height` 属性为 Table 指定最大高度。此时若表格所需的高度大于最大高度，则会显示一个滚动条。

```

1. <template>
2.   <el-table
3.     :data="tableData4"
4.     style="width: 100%"
5.     max-height="250">
6.     <el-table-column
7.       fixed
8.       prop="date"
9.       label="日期"
10.      width="150">
11.      </el-table-column>
12.      <el-table-column

```

```
13.      prop="name"
14.      label="姓名"
15.      width="120">
16.    </el-table-column>
17.    <el-table-column
18.      prop="province"
19.      label="省份"
20.      width="120">
21.    </el-table-column>
22.    <el-table-column
23.      prop="city"
24.      label="市区"
25.      width="120">
26.    </el-table-column>
27.    <el-table-column
28.      prop="address"
29.      label="地址"
30.      width="300">
31.    </el-table-column>
32.    <el-table-column
33.      prop="zip"
34.      label="邮编"
35.      width="120">
36.    </el-table-column>
37.    <el-table-column
38.      fixed="right"
39.      label="操作"
40.      width="120">
41.      <template slot-scope="scope">
42.        <el-button
43.          @click.native.prevent="deleteRow(scope.$index, tableData4)"
44.          type="text"
45.          size="small">
46.          移除
47.        </el-button>
48.      </template>
49.    </el-table-column>
50.  </el-table>
51. </template>
52.
53. <script>
54.   export default {
```

```
55.     methods: {
56.       deleteRow(index, rows) {
57.         rows.splice(index, 1);
58.       }
59.     },
60.     data() {
61.       return [
62.         tableData4: [
63.           date: '2016-05-03',
64.           name: '王小虎',
65.           province: '上海',
66.           city: '普陀区',
67.           address: '上海市普陀区金沙江路 1518 弄',
68.           zip: 200333
69.         },
70.         {
71.           date: '2016-05-02',
72.           name: '王小虎',
73.           province: '上海',
74.           city: '普陀区',
75.           address: '上海市普陀区金沙江路 1518 弄',
76.           zip: 200333
77.         },
78.         {
79.           date: '2016-05-04',
80.           name: '王小虎',
81.           province: '上海',
82.           city: '普陀区',
83.           address: '上海市普陀区金沙江路 1518 弄',
84.           zip: 200333
85.         },
86.         {
87.           date: '2016-05-01',
88.           name: '王小虎',
89.           province: '上海',
90.           city: '普陀区',
91.           address: '上海市普陀区金沙江路 1518 弄',
92.           zip: 200333
93.         },
94.         {
95.           date: '2016-05-08',
96.           name: '王小虎',
```

```

97.     }, {
98.         date: '2016-05-06',
99.         name: '王小虎',
100.        province: '上海',
101.        city: '普陀区',
102.        address: '上海市普陀区金沙江路 1518 弄',
103.        zip: 200333
104.    }, {
105.        date: '2016-05-07',
106.        name: '王小虎',
107.        province: '上海',
108.        city: '普陀区',
109.        address: '上海市普陀区金沙江路 1518 弄',
110.        zip: 200333
111.    }]
112. }
113. }
114. }
115. </script>

```

多级表头

数据结构比较复杂的时候，可使用多级表头来展现数据的层次关系。

		配送信息		
日期	姓名	地址		
		省份	市区	地址
2016-05-03	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄
2016-05-02	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄
2016-05-04	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄
2016-05-01	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄
2016-05-08	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄
2016-05-06	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄
2016-05-07	王小虎	上海	普陀区	上海市普陀区金沙江路 1518 弄

只需要在 el-table-column 里面嵌套 el-table-column，就可以实现多级表头。

```
1. <template>
2.   <el-table
3.     :data="tableData3"
4.     style="width: 100%">
5.       <el-table-column
6.         prop="date"
7.         label="日期"
8.         width="150">
9.       </el-table-column>
10.      <el-table-column label="配送信息">
11.        <el-table-column
12.          prop="name"
13.          label="姓名"
14.          width="120">
15.        </el-table-column>
16.        <el-table-column label="地址">
17.          <el-table-column
18.            prop="province"
19.            label="省份"
20.            width="120">
21.          </el-table-column>
22.          <el-table-column
23.            prop="city"
24.            label="市区"
25.            width="120">
26.          </el-table-column>
27.          <el-table-column
28.            prop="address"
29.            label="地址"
30.            width="300">
31.          </el-table-column>
32.          <el-table-column
33.            prop="zip"
34.            label="邮编"
35.            width="120">
36.          </el-table-column>
37.        </el-table-column>
38.      </el-table-column>
39.    </el-table>
40.  </template>
```

```
41.  
42. <script>  
43. export default {  
44.   data() {  
45.     return {  
46.       tableData3: [{  
47.         date: '2016-05-03',  
48.         name: '王小虎',  
49.         province: '上海',  
50.         city: '普陀区',  
51.         address: '上海市普陀区金沙江路 1518 弄',  
52.         zip: 200333  
53.       }, {  
54.         date: '2016-05-02',  
55.         name: '王小虎',  
56.         province: '上海',  
57.         city: '普陀区',  
58.         address: '上海市普陀区金沙江路 1518 弄',  
59.         zip: 200333  
60.       }, {  
61.         date: '2016-05-04',  
62.         name: '王小虎',  
63.         province: '上海',  
64.         city: '普陀区',  
65.         address: '上海市普陀区金沙江路 1518 弄',  
66.         zip: 200333  
67.       }, {  
68.         date: '2016-05-01',  
69.         name: '王小虎',  
70.         province: '上海',  
71.         city: '普陀区',  
72.         address: '上海市普陀区金沙江路 1518 弄',  
73.         zip: 200333  
74.       }, {  
75.         date: '2016-05-08',  
76.         name: '王小虎',  
77.         province: '上海',  
78.         city: '普陀区',  
79.         address: '上海市普陀区金沙江路 1518 弄',  
80.         zip: 200333  
81.       }, {  
82.         date: '2016-05-06',
```

```

83.         name: '王小虎',
84.         province: '上海',
85.         city: '普陀区',
86.         address: '上海市普陀区金沙江路 1518 弄',
87.         zip: 200333
88.     }, {
89.         date: '2016-05-07',
90.         name: '王小虎',
91.         province: '上海',
92.         city: '普陀区',
93.         address: '上海市普陀区金沙江路 1518 弄',
94.         zip: 200333
95.     }]
96. }
97. }
98. }
99. </script>

```

单选

选择单行数据时使用色块表示。

#	日期	姓名	地址
1	2016-05-03	王小虎	上海市普陀区金沙江路 1518 弄
2	2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
3	2016-05-04	王小虎	上海市普陀区金沙江路 1518 弄
4	2016-05-01	王小虎	上海市普陀区金沙江路 1518 弄

[选中第二行](#)

[取消选择](#)

Table 组件提供了单选的支持，只需要配置 `highlight-current-row` 属性即可实现单选。之后由 `current-change` 事件来管理选中时触发的事件，它会传入 `currentRow`，`oldCurrentRow`。如果需要显示索引，可以增加一列 `el-table-column`，设置 `type` 属性为 `index` 即可显示从 1 开始的索引号。

```

1.  <template>
2.    <el-table>

```

```
3.      ref="singleTable"
4.      :data="tableData"
5.      highlight-current-row
6.      @current-change="handleCurrentChange"
7.      style="width: 100%">
8.      <el-table-column
9.          type="index"
10.         width="50">
11.      </el-table-column>
12.      <el-table-column
13.          property="date"
14.          label="日期"
15.          width="120">
16.      </el-table-column>
17.      <el-table-column
18.          property="name"
19.          label="姓名"
20.          width="120">
21.      </el-table-column>
22.      <el-table-column
23.          property="address"
24.          label="地址">
25.      </el-table-column>
26.  </el-table>
27.  <div style="margin-top: 20px">
28.      <el-button @click="setCurrent(tableData[1])">选中第二行</el-button>
29.      <el-button @click="setCurrent()">取消选择</el-button>
30.  </div>
31. </template>
32.
33. <script>
34.   export default {
35.     data() {
36.       return {
37.         tableData: [
38.           {
39.             date: '2016-05-02',
40.             name: '王小虎',
41.             address: '上海市普陀区金沙江路 1518 弄'
42.           },
43.           {
44.             date: '2016-05-04',
45.             name: '王小虎',
46.             address: '上海市普陀区金沙江路 1517 弄'
```

```
45.     }, {
46.         date: '2016-05-01',
47.         name: '王小虎',
48.         address: '上海市普陀区金沙江路 1519 弄'
49.     }, {
50.         date: '2016-05-03',
51.         name: '王小虎',
52.         address: '上海市普陀区金沙江路 1516 弄'
53.     }],
54.     currentRow: null
55.   }
56. },
57.
58. methods: {
59.   setCurrent(row) {
60.     this.$refs.singleTable.setCurrentRow(row);
61.   },
62.   handleCurrentChange(val) {
63.     this.currentRow = val;
64.   }
65. }
66. }
67. </script>
```

多选

选择多行数据时使用 Checkbox。

<input checked="" type="checkbox"/>	日期	姓名	地址
<input checked="" type="checkbox"/>	2016-05-03	王小虎	上海市普陀区金沙江路 1518 弄
<input checked="" type="checkbox"/>	2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
<input checked="" type="checkbox"/>	2016-05-04	王小虎	上海市普陀区金沙江路 1518 弄
<input checked="" type="checkbox"/>	2016-05-01	王小虎	上海市普陀区金沙江路 1518 弄
<input checked="" type="checkbox"/>	2016-05-08	王小虎	上海市普陀区金沙江路 1518 弄
<input checked="" type="checkbox"/>	2016-05-06	王小虎	上海市普陀区金沙江路 1518 弄
<input checked="" type="checkbox"/>	2016-05-07	王小虎	上海市普陀区金沙江路 1518 弄

切换第二、第三行的选中状态取消选择

实现多选非常简单：手动添加一个 `el-table-column`，设 `type` 属性为 `selection` 即可；默认情况下若内容过多会折行显示，若需要单行显示可以使用 `show-overflow-tooltip` 属性，它接受一个 `Boolean`，为 `true` 时多余的内容会在 `hover` 时以 `tooltip` 的形式显示出来。

```

1.  <template>
2.    <el-table
3.      ref="multipleTable"
4.      :data="tableData3"
5.      tooltip-effect="dark"
6.      style="width: 100%"
7.      @selection-change="handleSelectionChange">
8.        <el-table-column
9.          type="selection"
10.         width="55">
11.        </el-table-column>
12.        <el-table-column
13.          label="日期"
14.          width="120">
15.          <template slot-scope="scope">{{ scope.row.date }}</template>
16.        </el-table-column>
17.        <el-table-column
18.          prop="name">
```

```
19.      label="姓名"
20.      width="120">
21.    </el-table-column>
22.    <el-table-column
23.      prop="address"
24.      label="地址"
25.      show-overflow-tooltip>
26.    </el-table-column>
27.  </el-table>
28.  <div style="margin-top: 20px">
29.    <el-button @click="toggleSelection([tableData3[1], tableData3[2]])">切换第
二、第三行的选中状态</el-button>
30.    <el-button @click="toggleSelection()">取消选择</el-button>
31.  </div>
32. </template>
33.
34. <script>
35.   export default {
36.     data() {
37.       return {
38.         tableData3: [
39.           {
40.             date: '2016-05-03',
41.             name: '王小虎',
42.             address: '上海市普陀区金沙江路 1518 弄'
43.           },
44.           {
45.             date: '2016-05-02',
46.             name: '王小虎',
47.             address: '上海市普陀区金沙江路 1518 弄'
48.           },
49.           {
50.             date: '2016-05-04',
51.             name: '王小虎',
52.             address: '上海市普陀区金沙江路 1518 弄'
53.           },
54.           {
55.             date: '2016-05-01',
56.             name: '王小虎',
57.             address: '上海市普陀区金沙江路 1518 弄'
58.           },
59.           {
60.             date: '2016-05-08',
61.             name: '王小虎',
62.             address: '上海市普陀区金沙江路 1518 弄'
63.           },
64.           {
65.             date: '2016-05-06',
66.             name: '王小虎',
67.             address: '上海市普陀区金沙江路 1518 弄'
68.           }
69.         ]
70.       }
71.     }
72.   }
73. 
```

```
60.         name: '王小虎',
61.         address: '上海市普陀区金沙江路 1518 弄'
62.     }, {
63.         date: '2016-05-07',
64.         name: '王小虎',
65.         address: '上海市普陀区金沙江路 1518 弄'
66.     ],
67.     multipleSelection: []
68.   }
69. },
70.
71. methods: {
72.   toggleSelection(rows) {
73.     if (rows) {
74.       rows.forEach(row => {
75.         this.$refs.multipleTable.toggleRowSelection(row);
76.       });
77.     } else {
78.       this.$refs.multipleTable.clearSelection();
79.     }
80.   },
81.   handleSelectionChange(val) {
82.     this.multipleSelection = val;
83.   }
84. }
85. }
86. </script>
```

显示代码

排序

对表格进行排序，可快速查找或对比数据。

日期	姓名	地址
2016-05-04	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-03	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
2016-05-01	王小虎	上海市普陀区金沙江路 1518 弄

在列中设置 `sortable` 属性即可实现以该列为基准的排序，接受一个 `Boolean`，默认为 `false`。可以通过 `Table` 的 `default-sort` 属性设置默认的排序列和排序顺序。可以使用 `sort-method` 或者 `sort-by` 使用自定义的排序规则。如果需要后端排序，需将 `sortable` 设置为 `custom`，同时在 `Table` 上监听 `sort-change` 事件，在事件回调中可以获取当前排序的字段名和排序顺序，从而向接口请求排序后的表格数据。在本例中，我们还使用了 `formatter` 属性，它用于格式化指定列的值，接受一个 `Function`，会传入两个参数：`row` 和 `column`，可以根据自己的需求进行处理。

```

1.  <template>
2.    <el-table
3.      :data="tableData"
4.      style="width: 100%"
5.      :default-sort = "{prop: 'date', order: 'descending'}"
6.    >
7.      <el-table-column
8.        prop="date"
9.        label="日期"
10.       sortable
11.       width="180">
12.     </el-table-column>
13.     <el-table-column
14.       prop="name"
15.       label="姓名"
16.       sortable
17.       width="180">
```

```
18.      </el-table-column>
19.      <el-table-column
20.          prop="address"
21.          label="地址"
22.          :formatter="formatter">
23.      </el-table-column>
24.  </el-table>
25. </template>
26.
27. <script>
28.   export default {
29.     data() {
30.       return {
31.         tableData: [
32.           {
33.             date: '2016-05-02',
34.             name: '王小虎',
35.             address: '上海市普陀区金沙江路 1518 弄'
36.           },
37.           {
38.             date: '2016-05-04',
39.             name: '王小虎',
40.             address: '上海市普陀区金沙江路 1517 弄'
41.           },
42.           {
43.             date: '2016-05-01',
44.             name: '王小虎',
45.             address: '上海市普陀区金沙江路 1519 弄'
46.           },
47.           {
48.             date: '2016-05-03',
49.             name: '王小虎',
50.             address: '上海市普陀区金沙江路 1516 弄'
51.           }
52.         ]
53.       }
54.     },
55.     methods: {
56.       formatter(row, column) {
57.         return row.address;
58.       }
59.     }
60.   }
61. </script>
```

筛选

对表格进行筛选，可快速查找到自己想看的数据。

日期	姓名	地址	标签
2016-05-03	王小虎	上海市普陀区金沙江路 1518 弄	<input type="checkbox"/> 家 <input type="checkbox"/> 公司
2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄	筛选 重置
2016-05-04	王小虎	上海市普陀区金沙江路 1518 弄	家
2016-05-01	王小虎	上海市普陀区金沙江路 1518 弄	公司

在列中设置 `filters` `filter-method` 属性即可开启该列的筛选，`filters` 是一个数组，`filter-method` 是一个方法，它用于决定某些数据是否显示，会传入三个参数：`value`，`row` 和 `column`。

```

1. <template>
2.   <el-table
3.     :data="tableData"
4.     style="width: 100%">
5.       <el-table-column
6.         prop="date"
7.         label="日期"
8.         sortable
9.         width="180"
10.        :filters="[{text: '2016-05-01', value: '2016-05-01'}, {text: '2016-05-02', value: '2016-05-02'}, {text: '2016-05-03', value: '2016-05-03'}, {text: '2016-05-04', value: '2016-05-04'}]"
11.        :filter-method="filterHandler"
12.      >
13.        </el-table-column>
14.        <el-table-column
15.          prop="name"
16.          label="姓名"
17.          width="180">
18.        </el-table-column>
19.        <el-table-column
20.          prop="address"
21.          label="地址"
22.          :formatter="formatter">
```

```
23.      </el-table-column>
24.      <el-table-column
25.          prop="tag"
26.          label="标签"
27.          width="100"
28.          :filters="[{ text: '家', value: '家' }, { text: '公司', value: '公司' }]"
29.          :filter-method="filterTag"
30.          filter-placement="bottom-end">
31.          <template slot-scope="scope">
32.              <el-tag
33.                  :type="scope.row.tag === '家' ? 'primary' : 'success'">
34.                  disable-transitions>{{scope.row.tag}}</el-tag>
35.          </template>
36.      </el-table-column>
37.  </el-table>
38. </template>
39.
40. <script>
41.   export default {
42.     data() {
43.       return {
44.         tableData: [
45.           {
46.             date: '2016-05-02',
47.             name: '王小虎',
48.             address: '上海市普陀区金沙江路 1518 弄',
49.             tag: '家'
50.           },
51.           {
52.             date: '2016-05-04',
53.             name: '王小虎',
54.             address: '上海市普陀区金沙江路 1517 弄',
55.             tag: '公司'
56.           },
57.           {
58.             date: '2016-05-01',
59.             name: '王小虎',
60.             address: '上海市普陀区金沙江路 1519 弄',
61.             tag: '家'
62.           },
63.           {
64.             date: '2016-05-03',
65.             name: '王小虎',
66.             address: '上海市普陀区金沙江路 1516 弄',
67.             tag: '公司'
68.           }
69.         ]
70.       }
71.     }
72.   }
73. 
```

```

65.      }
66.    },
67.  methods: {
68.    formatter(row, column) {
69.      return row.address;
70.    },
71.    filterTag(value, row) {
72.      return row.tag === value;
73.    },
74.    filterHandler(value, row, column) {
75.      const property = column['property'];
76.      return row[property] === value;
77.    }
78.  }
79. }
80. </script>

```

自定义列模板

自定义列的显示内容，可组合其他组件使用。

日期	姓名	操作	
① 2016-05-03	王小虎	编辑	删除
① 2016-05-02	王小虎	编辑	删除
① 2016-05-04	王小虎	编辑	删除
① 2016-05-01	王小虎	编辑	删除

通过 `Scoped slot` 可以获取到 `row`, `column`, `$index` 和 `store`(`table` 内部的状态管理) 的数据，用法参考 `demo`。

```

1.  <template>
2.    <el-table
3.      :data="tableData"
4.      style="width: 100%">
5.        <el-table-column
6.          label="日期"
7.          width="180">

```

```
8.      <template slot-scope="scope">
9.          <i class="el-icon-time"></i>
10.         <span style="margin-left: 10px">{{ scope.row.date }}</span>
11.     </template>
12. </el-table-column>
13. <el-table-column
14.     label="姓名"
15.     width="180">
16.     <template slot-scope="scope">
17.         <el-popover trigger="hover" placement="top">
18.             <p>姓名: {{ scope.row.name }}</p>
19.             <p>住址: {{ scope.row.address }}</p>
20.             <div slot="reference" class="name-wrapper">
21.                 <el-tag size="medium">{{ scope.row.name }}</el-tag>
22.             </div>
23.         </el-popover>
24.     </template>
25. </el-table-column>
26. <el-table-column label="操作">
27.     <template slot-scope="scope">
28.         <el-button
29.             size="mini"
30.             @click="handleEdit(scope.$index, scope.row)">编辑</el-button>
31.         <el-button
32.             size="mini"
33.             type="danger"
34.             @click="handleDelete(scope.$index, scope.row)">删除</el-button>
35.     </template>
36. </el-table-column>
37. </el-table>
38. </template>
39.
40. <script>
41. export default {
42.     data() {
43.         return {
44.             tableData: [
45.                 date: '2016-05-02',
46.                 name: '王小虎',
47.                 address: '上海市普陀区金沙江路 1518 弄'
48.             ],
49.             date: '2016-05-04',
```

```
50.         name: '王小虎',
51.         address: '上海市普陀区金沙江路 1517 弄'
52.     },
53.     {
54.         date: '2016-05-01',
55.         name: '王小虎',
56.         address: '上海市普陀区金沙江路 1519 弄'
57.     },
58.     {
59.         date: '2016-05-03',
60.         name: '王小虎',
61.         address: '上海市普陀区金沙江路 1516 弄'
62.     }
63. },
64. methods: {
65.     handleEdit(index, row) {
66.         console.log(index, row);
67.     },
68.     handleDelete(index, row) {
69.         console.log(index, row);
70.     }
71. }
```

展开行

当行内容过多并且不想显示横向滚动条时，可以使用 Table 展开行功能。

商品 ID	商品名称	描述	
▼ 12987122	好滋好味鸡蛋仔	荷兰优质淡奶，奶香浓而不腻	
商品名称	好滋好味鸡蛋仔	所属店铺	王小虎夫妻店
商品 ID	12987122	店铺 ID	10333
商品分类	江浙小吃、小吃零食	店铺地址	上海市普陀区真北路
商品描述	荷兰优质淡奶，奶香浓而不腻		
▶ 12987123	好滋好味鸡蛋仔	荷兰优质淡奶，奶香浓而不腻	
▶ 12987125	好滋好味鸡蛋仔	荷兰优质淡奶，奶香浓而不腻	
▶ 12987126	好滋好味鸡蛋仔	荷兰优质淡奶，奶香浓而不腻	

通过设置 `type="expand"` 和 `Scoped slot` 可以开启展开行功能，`el-table-column` 的模板会被渲染成为展开行的内容，展开行可访问的属性与使用自定义列模板时的 `Scoped slot` 相同。

```

1. <template>
2.   <el-table
3.     :data="tableData5"
4.     style="width: 100%">
5.       <el-table-column type="expand">
6.         <template slot-scope="props">
7.           <el-form label-position="left" inline class="demo-table-expand">
8.             <el-form-item label="商品名称">
9.               <span>{{ props.row.name }}</span>
10.            </el-form-item>
11.            <el-form-item label="所属店铺">
12.              <span>{{ props.row.shop }}</span>
13.            </el-form-item>
14.            <el-form-item label="商品 ID">
15.              <span>{{ props.row.id }}</span>
16.            </el-form-item>
17.            <el-form-item label="店铺 ID">
18.              <span>{{ props.row.shopId }}</span>
19.            </el-form-item>
20.            <el-form-item label="商品分类">
```

```
21.          <span>{{ props.row.category }}</span>
22.        </el-form-item>
23.        <el-form-item label="店铺地址">
24.          <span>{{ props.row.address }}</span>
25.        </el-form-item>
26.        <el-form-item label="商品描述">
27.          <span>{{ props.row.desc }}</span>
28.        </el-form-item>
29.      </el-form>
30.    </template>
31.  </el-table-column>
32. <el-table-column
33.   label="商品 ID"
34.   prop="id">
35. </el-table-column>
36. <el-table-column
37.   label="商品名称"
38.   prop="name">
39. </el-table-column>
40. <el-table-column
41.   label="描述"
42.   prop="desc">
43. </el-table-column>
44. </el-table>
45. </template>
46.
47. <style>
48.   .demo-table-expand {
49.     font-size: 0;
50.   }
51.   .demo-table-expand label {
52.     width: 90px;
53.     color: #99a9bf;
54.   }
55.   .demo-table-expand .el-form-item {
56.     margin-right: 0;
57.     margin-bottom: 0;
58.     width: 50%;
59.   }
60. </style>
61.
62. <script>
```

```
63.  export default {
64.    data() {
65.      return {
66.        tableData5: [
67.          id: '12987122',
68.          name: '好滋好味鸡蛋仔',
69.          category: '江浙小吃、小吃零食',
70.          desc: '荷兰优质淡奶，奶香浓而不腻',
71.          address: '上海市普陀区真北路',
72.          shop: '王小虎夫妻店',
73.          shopId: '10333'
74.        },
75.        id: '12987123',
76.        name: '好滋好味鸡蛋仔',
77.        category: '江浙小吃、小吃零食',
78.        desc: '荷兰优质淡奶，奶香浓而不腻',
79.        address: '上海市普陀区真北路',
80.        shop: '王小虎夫妻店',
81.        shopId: '10333'
82.      },
83.      id: '12987125',
84.      name: '好滋好味鸡蛋仔',
85.      category: '江浙小吃、小吃零食',
86.      desc: '荷兰优质淡奶，奶香浓而不腻',
87.      address: '上海市普陀区真北路',
88.      shop: '王小虎夫妻店',
89.      shopId: '10333'
90.    },
91.    id: '12987126',
92.    name: '好滋好味鸡蛋仔',
93.    category: '江浙小吃、小吃零食',
94.    desc: '荷兰优质淡奶，奶香浓而不腻',
95.    address: '上海市普陀区真北路',
96.    shop: '王小虎夫妻店',
97.    shopId: '10333'
98.  ],
99. }
100. }
101. }
102. </script>
```

表尾合计行

若表格展示的是各类数字，可以在表尾显示各列的合计。

ID	姓名	数值 1	数值 2	数值 3
12987122	王小虎	234	3.2	10
12987123	王小虎	165	4.43	12
12987124	王小虎	324	1.9	9
12987125	王小虎	621	2.2	17
12987126	王小虎	539	4.1	15
合计		1883	15.83	63

ID	姓名	数值 1 (元)	数值 2 (元)	数值 3 (元)
12987122	王小虎	234	3.2	10
12987123	王小虎	165	4.43	12
总价	N/A	1883 元	15.83 元	63 元

将 `show-summary` 设置为 `true` 就会在表格尾部展示合计行。默认情况下，对于合计行，第一列不进行数据求合操作，而是显示「合计」二字（可通过 `sum-text` 配置），其余列会将本列所有数值进行求合操作，并显示出来。当然，你也可以定义自己的合计逻辑。使用 `summary-method` 并传入一个方法，返回一个数组，这个数组中的各项就会显示在合计行的各列中，具体可以参考本例中的第二个表格。

```

1. <template>
2.   <el-table
3.     :data="tableData6"
4.     border
5.     show-summary
6.     style="width: 100%">
7.       <el-table-column
8.         prop="id"
9.         label="ID"
10.        width="180">
11.      </el-table-column>
12.      <el-table-column
13.        prop="name"

```

```
14.      label="姓名">
15.    </el-table-column>
16.    <el-table-column
17.      prop="amount1"
18.      sortable
19.      label="数值 1">
20.    </el-table-column>
21.    <el-table-column
22.      prop="amount2"
23.      sortable
24.      label="数值 2">
25.    </el-table-column>
26.    <el-table-column
27.      prop="amount3"
28.      sortable
29.      label="数值 3">
30.    </el-table-column>
31.  </el-table>
32.
33.  <el-table
34.    :data="tableData6"
35.    border
36.    height="200"
37.    :summary-method="getSummaries"
38.    show-summary
39.    style="width: 100%; margin-top: 20px">
40.    <el-table-column
41.      prop="id"
42.      label="ID"
43.      width="180">
44.    </el-table-column>
45.    <el-table-column
46.      prop="name"
47.      label="姓名">
48.    </el-table-column>
49.    <el-table-column
50.      prop="amount1"
51.      label="数值 1 (元) ">
52.    </el-table-column>
53.    <el-table-column
54.      prop="amount2"
55.      label="数值 2 (元) ">
```

```
56.      </el-table-column>
57.      <el-table-column
58.          prop="amount3"
59.          label="数值 3 (元) ">
60.      </el-table-column>
61.  </el-table>
62. </template>
63.
64. <script>
65. export default {
66.   data() {
67.     return {
68.       tableData6: [
69.         { id: '12987122',
70.           name: '王小虎',
71.           amount1: '234',
72.           amount2: '3.2',
73.           amount3: 10
74.         },
75.         { id: '12987123',
76.           name: '王小虎',
77.           amount1: '165',
78.           amount2: '4.43',
79.           amount3: 12
80.         },
81.         { id: '12987124',
82.           name: '王小虎',
83.           amount1: '324',
84.           amount2: '1.9',
85.           amount3: 9
86.         },
87.         { id: '12987125',
88.           name: '王小虎',
89.           amount1: '621',
90.           amount2: '2.2',
91.           amount3: 17
92.         },
93.         { id: '12987126',
94.           name: '王小虎',
95.           amount1: '539',
96.           amount2: '4.1',
97.           amount3: 15

```

```
98.      }]
99.    );
100.   },
101.   methods: {
102.     getSummaries(param) {
103.       const { columns, data } = param;
104.       const sums = [];
105.       columns.forEach((column, index) => {
106.         if (index === 0) {
107.           sums[index] = '总价';
108.           return;
109.         }
110.         const values = data.map(item => Number(item[column.property]));
111.         if (!values.every(value => isNaN(value))) {
112.           sums[index] = values.reduce((prev, curr) => {
113.             const value = Number(curr);
114.             if (!isNaN(value)) {
115.               return prev + curr;
116.             } else {
117.               return prev;
118.             }
119.           }, 0);
120.           sums[index] += ' 元';
121.         } else {
122.           sums[index] = 'N/A';
123.         }
124.       });
125.
126.       return sums;
127.     }
128.   }
129. };
130. </script>
```

合并行或列

多行或多列共用一个数据时，可以合并行或列。

ID	姓名	数值 1	数值 2	数值 3
12987122		234	3.2	10
12987123	王小虎	165	4.43	12
12987124		324	1.9	9
12987125	王小虎	621	2.2	17
12987126		539	4.1	15

ID	姓名	数值 1 (元)	数值 2 (元)	数值 3 (元)
12987122	王小虎	234	3.2	10
	王小虎	165	4.43	12
12987124	王小虎	324	1.9	9
	王小虎	621	2.2	17
12987126	王小虎	539	4.1	15

通过给 `table` 传入 `span-method` 方法可以实现合并行或列，方法的参数是一个对象，里面包含当前行 `row`、当前列 `column`、当前行号 `rowIndex`、当前列号 `columnIndex` 四个属性。该函数可以返回一个包含两个元素的数组，第一个元素代表 `rowspan`，第二个元素代表 `colspan`。也可以返回一个键名为 `rowspan` 和 `colspan` 的对象。

```

1. <template>
2.   <div>
3.     <el-table
4.       :data="tableData6"
5.       :span-method="arraySpanMethod"
6.       border
7.       style="width: 100%">
8.       <el-table-column
9.         prop="id"
10.        label="ID"
11.        width="180">
12.       </el-table-column>
13.       <el-table-column
14.         prop="name">
```

```
15.      label="姓名">
16.    </el-table-column>
17.    <el-table-column
18.      prop="amount1"
19.      sortable
20.      label="数值 1">
21.    </el-table-column>
22.    <el-table-column
23.      prop="amount2"
24.      sortable
25.      label="数值 2">
26.    </el-table-column>
27.    <el-table-column
28.      prop="amount3"
29.      sortable
30.      label="数值 3">
31.    </el-table-column>
32.  </el-table>
33.
34.  <el-table
35.    :data="tableData6"
36.    :span-method="objectSpanMethod"
37.    border
38.    style="width: 100%; margin-top: 20px">
39.    <el-table-column
40.      prop="id"
41.      label="ID"
42.      width="180">
43.    </el-table-column>
44.    <el-table-column
45.      prop="name"
46.      label="姓名">
47.    </el-table-column>
48.    <el-table-column
49.      prop="amount1"
50.      label="数值 1 (元) ">
51.    </el-table-column>
52.    <el-table-column
53.      prop="amount2"
54.      label="数值 2 (元) ">
55.    </el-table-column>
56.    <el-table-column
```

```
57.         prop="amount3"
58.         label="数值 3 (元) "
59.     
```

```
60.     </el-table-column>
61.   
```

```
62. </div>
63.
64. <script>
65.   export default {
66.     data() {
67.       return {
68.         tableData6: [
69.           { id: '12987122', name: '王小虎', amount1: '234', amount2: '3.2', amount3: 10 },
70.           { id: '12987123', name: '王小虎', amount1: '165', amount2: '4.43', amount3: 12 },
71.           { id: '12987124', name: '王小虎', amount1: '324', amount2: '1.9', amount3: 9 },
72.           { id: '12987125', name: '王小虎', amount1: '621', amount2: '2.2', amount3: 17 },
73.           { id: '12987126', name: '王小虎', amount1: '539', amount2: '4.1', amount3: 15 }
74.         ]
75.       }
76.     }
77.   }
78. }
```

```
99.      };
100.     },
101.     methods: {
102.       arraySpanMethod({ row, column, rowIndex, columnIndex }) {
103.         if (rowIndex % 2 === 0) {
104.           if (columnIndex === 0) {
105.             return [1, 2];
106.           } else if (columnIndex === 1) {
107.             return [0, 0];
108.           }
109.         }
110.       },
111.
112.       objectSpanMethod({ row, column, rowIndex, columnIndex }) {
113.         if (columnIndex === 0) {
114.           if (rowIndex % 2 === 0) {
115.             return {
116.               rowspan: 2,
117.               colspan: 1
118.             };
119.           } else {
120.             return {
121.               rowspan: 0,
122.               colspan: 0
123.             };
124.           }
125.         }
126.       }
127.     }
128.   };
129. </script>
```

自定义索引

自定义 `type=index` 列的行号。

#	日期	姓名	地址
0	2016-05-03	王小虎	上海市普陀区金沙江路 1518 弄
2	2016-05-02	王小虎	上海市普陀区金沙江路 1518 弄
4	2016-05-04	王小虎	上海市普陀区金沙江路 1518 弄
6	2016-05-01	王小虎	上海市普陀区金沙江路 1518 弄

通过给 `type=index` 的列传入 `index` 属性，可以自定义索引。该属性传入数字时，将作为索引的起始值。也可以传入一个方法，它提供当前行的行号（从 `0` 开始）作为参数，返回值将作为索引展示。

```

1. <template>
2.   <el-table
3.     :data="tableData"
4.     style="width: 100%">
5.       <el-table-column
6.         type="index"
7.         :index="indexMethod">
8.       </el-table-column>
9.       <el-table-column
10.        prop="date"
11.        label="日期"
12.        width="180">
13.      </el-table-column>
14.      <el-table-column
15.        prop="name"
16.        label="姓名"
17.        width="180">
18.      </el-table-column>
19.      <el-table-column
20.        prop="address"
21.        label="地址">
22.      </el-table-column>
23.    </el-table>
24.  </template>
25.
26.  <script>
27.    export default {

```

```
28.     data() {
29.       return {
30.         tableData: [
31.           date: '2016-05-03',
32.           name: '王小虎',
33.           province: '上海',
34.           city: '普陀区',
35.           address: '上海市普陀区金沙江路 1518 弄',
36.           zip: 200333,
37.           tag: '家'
38.         },
39.         {
40.           date: '2016-05-02',
41.           name: '王小虎',
42.           province: '上海',
43.           city: '普陀区',
44.           address: '上海市普陀区金沙江路 1518 弄',
45.           zip: 200333,
46.           tag: '公司'
47.         },
48.         {
49.           date: '2016-05-04',
50.           name: '王小虎',
51.           province: '上海',
52.           city: '普陀区',
53.           address: '上海市普陀区金沙江路 1518 弄',
54.           zip: 200333,
55.           tag: '家'
56.         },
57.         {
58.           date: '2016-05-01',
59.           name: '王小虎',
60.           province: '上海',
61.           city: '普陀区',
62.           address: '上海市普陀区金沙江路 1518 弄',
63.           zip: 200333,
64.           tag: '公司'
65.         }
66.       ],
67.       methods: {
68.         indexMethod(index) {
69.           return index * 2;
70.         }
71.       }
72.     }
73.   }
74. 
```

```

70.    };
71.  </script>

```

显示代码

Table Attributes

参数	说明	类型	可选值	默认值
data	显示的数据	array	—	—
height	Table 的高度，默认为自动高度。如果 height 为 number 类型，单位 px；如果 height 为 string 类型，则这个高度会设置为 Table 的 style.height 的值，Table 的高度会受控于外部样式。	string/number	—	—
max-height	Table 的最大高度	string/number	—	—
stripe	是否为斑马纹 table	boolean	—	false
border	是否带有纵向边框	boolean	—	false
size	Table 的尺寸	string	medium / small / mini	—
fit	列的宽度是否自撑开	boolean	—	true
show-header	是否显示表头	boolean	—	true
highlight-current-row	是否要高亮当前行	boolean	—	false
current-row-key	当前行的 key，只写属性	String, Number	—	—
row-class-name	行的 className 的回调方法，也可以使用字符串为所有行设置一个固定的 className。	Function({row, rowIndex})/String	—	—
row-style	行的 style 的回调方法，也可以使用一个固定的 Object 为所有行设置一样的 Style。	Function({row, rowIndex})/Object	—	—
cell-class-name	单元格的 className 的回调方法，也可以使用字符串为所有单元格设置一个固定的 className。	Function({row, column, rowIndex, columnIndex})/String	—	—
cell-style	单元格的 style 的回调方法，也可以使用一个固定的 Object 为所有单元格设置一样的 Style。	Function({row, column, rowIndex, columnIndex})/Object	—	—
	表头行的 className			

Table 表格

header-row-class-name	的回调方法，也可以使用字符串为所有表头行设置一个固定的 className。	Function({row, rowIndex})/String	—	—
header-row-style	表头行的 style 的回调方法，也可以使用一个固定的 Object 为所有表头行设置一样的 Style。	Function({row, rowIndex})/Object	—	—
header-cell-class-name	表头单元格的 className 的回调方法，也可以使用字符串为所有表头单元格设置一个固定的 className。	Function({row, column, rowIndex, columnIndex})/String	—	—
header-cell-style	表头单元格的 style 的回调方法，也可以使用一个固定的 Object 为所有表头单元格设置一样的 Style。	Function({row, column, rowIndex, columnIndex})/Object	—	—
row-key	行数据的 Key，用来优化 Table 的渲染；在使用 reserve-selection 功能的情况下，该属性是必填的。类型为 String 时，支持多层访 问： <code>user.info.id</code> ，但不支持 <code>user.info[0].id</code> ，此种情况请使用 <code>Function</code> 。	Function(row)/String	—	—
empty-text	空数据时显示的文本内容，也可以通过 <code>slot="empty"</code> 设置	String	—	暂无数据
default-expand-all	是否默认展开所有行，当 Table 中存在 type="expand" 的 Column 的时候有效	Boolean	—	false
expand-row-keys	可以通过该属性设置 Table 目前的展开行，需要设置 row-key 属性才能使用，该属性为展开行的 keys 数组。	Array	—	
default-sort	默认的排序列的 prop 和顺序。它的 prop 属性指定默认的排序的列，order 指定默认排序的顺序	Object	<code>order : ascending, descending</code>	如果只指定了 prop，没有指定 order，则默认顺序是 ascending
tooltip-effect	tooltip effect 属性	String	dark/light	
show-summary	是否在表尾显示合计行	Boolean	—	false
sum-text	合计行第一列的文本	String	—	合计
summary-method	自定义的合计计算方法	Function({ columns, data })	—	—

span-method	合并行或列的计算方法	Function({ row, column, rowIndex, columnIndex })	-	-
select-on-indeterminate	在多选表格中，当仅有部分行被选中时，点击表头的多选框时的行为。若为 true，则选中所有行；若为 false，则取消选择所有行	Boolean	-	true

Table Events

事件名	说明	参数
select	当用户手动勾选数据行的 Checkbox 时触发的事件	selection, row
select-all	当用户手动勾选全选 Checkbox 时触发的事件	selection
selection-change	当选择项发生变化时会触发该事件	selection
cell-mouse-enter	当单元格 hover 进入时会触发该事件	row, column, cell, event
cell-mouse-leave	当单元格 hover 退出时会触发该事件	row, column, cell, event
cell-click	当某个单元格被点击时会触发该事件	row, column, cell, event
cell-dblclick	当某个单元格被双击时会触发该事件	row, column, cell, event
row-click	当某一行被点击时会触发该事件	row, event, column
row-contextmenu	当某一行被鼠标右键点击时会触发该事件	row, event
row-dblclick	当某一行被双击时会触发该事件	row, event
header-click	当某一列的表头被点击时会触发该事件	column, event
header-contextmenu	当某一列的表头被鼠标右键点击时会触发该事件	column, event
sort-change	当表格的排序条件发生变化的时候会触发该事件	{ column, prop, order }
filter-change	当表格的筛选条件发生变化的时候会触发该事件，参数的值是一个对象，对象的 key 是 column 的 columnKey，对应的 value 为用户选择的筛选条件的数组。	filters
current-change	当表格的当前行发生变化的时候会触发该事件，如果要高亮当前行，请打开表格的 highlight-current-row 属性	currentRow, oldCurrentRow
header-dragend	当拖动表头改变了列的宽度的时候会触发该事件	newWidth, oldWidth, column, event
expand-change	当用户对某一行展开或者关闭的时候会触发该事件	row, expandedRows

Table Methods

方法名	说明	参数
clearSelection	用于多选表格，清空用户的选择	-
toggleRowSelection	用于多选表格，切换某一行的选中状态，如果使用了第二个参数，则是设置这一行选中与否 (selected 为 true 则选中)	row, selected
toggleAllSelection	用于多选表格，切换所有行的选中状态	-
toggleRowExpansion	用于可展开表格，切换某一行的展开状态，如果使用了第二个参数，则是设置这一行展开与否 (expanded 为 true 则展开)	row, expanded
setCurrentRow	用于单选表格，设定某一行作为选中行，如果调用时不加参数，则会取消目前高亮行的选中状态。	row
clearSort	用于清空排序条件，数据会恢复成未排序的状态	-
clearFilter	用于清空过滤条件，数据会恢复成未过滤的状态	-
doLayout	对 Table 进行重新布局。当 Table 或其祖先元素由隐藏切换为显示时，可能需要调用此方法	-
sort	手动对 Table 进行排序。参数 prop 属性指定排序列，order 指定排序顺序。	prop: string, order: string

Table Slot

name	说明
append	插入至表格最后一行之后的内容，如果需要对表格的内容进行无限滚动操作，可能需要用到这个 slot。若表格有合计行，该 slot 会位于合计行之上。

Table-column Attributes

参数	说明	类型	可选值
type	对应列的类型。如果设置了 selection，则显示多选框；如果设置了 index，则显示该行的索引（从 1 开始计算）；如果设置了 expand，则显示为一个可展开的按钮	string	selection/index/expand
index	如果设置了 type=index，可以通过传递 index 属性来自定义索引	string, Function(index)	-
	column 的		

column-key	key, 如果需要使用 filter-change 事件, 则需要此属性标识是哪个 column 的筛选条件	string	—
label	显示的标题	string	—
prop	对应列内容的字段名, 也可以使用 property 属性	string	—
width	对应列的宽度	string	—
min-width	对应列的最小宽度, 与 width 的区别是 width 是固定的, min-width 会把剩余宽度按比例分配给设置了 min-width 的列	string	—
fixed	列是否固定在左侧或者右侧, true 表示固定在左侧	string, boolean	true, left, right
render-header	列标题 Label 区域渲染使用的 Function	Function(h, { column, \$index })	—
sortable	对应列是否可以排序, 如果设置为 'custom', 则代表用户希望远程排序, 需要监听 Table 的 sort-change 事件	boolean, string	true, false, 'custom'
sort-method	对数据进行排序的时候使用的方法, 仅当 sortable 设置为 true 的时候有效, 需返回一个数字, 和 Array.sort 表现一致	Function(a, b)	—
sort-by	指定数据按照哪个属性进行排序, 仅当 sortable 设置为 true 且没有设置 sort-method 的时候有效。如果 sort-by 为数组, 则先按照第 1 个属性排序, 如果第 1 个相等, 再按照第 2 个排序, 以此类推	String/Array/Function(row, index)	—
sort-	数据在排序时所使用排序策略的轮转顺序, 仅当 sortable 为 true 时有效。需	数组中的元素需为以下三者之一: —: ascending 表示升序, descending 表示降序	

Table 表格

orders	传入一个数组，随着用户点击表头，该列依次按照数组中元素的顺序进行排序		序, null 表示还原为原始顺序
resizable	对应列是否可以通过拖动改变宽度（需要在 el-table 上设置 border 属性为真）	boolean	—
formatter	用来格式化内容	Function(row, column, cellValue, index)	—
show-overflow-tooltip	当内容过长被隐藏时显示 tooltip	Boolean	—
align	对齐方式	String	left/center/right
header-align	表头对齐方式，若不设置该项，则使用表格的对齐方式	String	left/center/right
class-name	列的 className	string	—
label-class-name	当前列标题的自定义类名	string	—
selectable	仅对 type=selection 的列有效，类型为 Function, Function 的返回值用来决定这一行的 CheckBox 是否可以勾选	Function(row, index)	—
reserve-selection	仅对 type=selection 的列有效，类型为 Boolean，为 true 则会在数据更新之后保留之前选中的数据（需指定 row-key ）	Boolean	—
filters	数据过滤的选项，数组格式，数组中的元素需要有 text 和 value 属性。	Array[{ text, value }]	—
filter-placement	过滤弹出框的定位	String	与 Tooltip 的 placement 属性相同
filter-multiple	数据过滤的选项是否多选	Boolean	—
filter-method	数据过滤使用的方法，如果是多选的筛选项，对每一条数据会执行多次，任意一次返回 true 就会显示。	Function(value, row, column)	—

filtered-value	选中的数据过滤项，如果需要自定义表头过滤的渲染方式，可能会需要此属性。	Array	-
----------------	-------------------------------------	-------	---

Table-column Scoped Slot

name	说明
-	自定义列的内容，参数为 { row, column, \$index }

原文: <http://element-cn.eleme.io/#/zh-CN/component/table>



Tag 标签

Tag 标签

用于标记和选择。

基础用法



由 `type` 属性来选择tag的类型，也可以通过 `color` 属性来自定义背景色。

```
1. <el-tag>标签一</el-tag>
2. <el-tag type="success">标签二</el-tag>
3. <el-tag type="info">标签三</el-tag>
4. <el-tag type="warning">标签四</el-tag>
5. <el-tag type="danger">标签五</el-tag>
```

可移除标签



设置 `closable` 属性可以定义一个标签是否可移除。默认的标签移除时会附带渐变动画，如果不想使用，可以设置 `disable-transitions` 属性，它接受一个 `Boolean`，`true` 为关闭。

```
1. <el-tag
2.   v-for="tag in tags"
3.   :key="tag.name"
4.   closable
5.   :type="tag.type">
6.   {{tag.name}}
7. </el-tag>
8.
9. <script>
10. export default {
11.   data() {
12.     return {
13.       tags: [
```

```

14.      { name: '标签一', type: '' },
15.      { name: '标签二', type: 'success' },
16.      { name: '标签三', type: 'info' },
17.      { name: '标签四', type: 'warning' },
18.      { name: '标签五', type: 'danger' }
19.    ]
20.  );
21. }
22. }
23. </script>

```

动态编辑标签

动态编辑标签可以通过点击标签关闭按钮后触发的 `close` 事件来实现



```

1. <el-tag
2.   :key="tag"
3.   v-for="tag in dynamicTags"
4.   closable
5.   :disable-transitions="false"
6.   @close="handleClose(tag)"
7.   {{tag}}
8. </el-tag>
9. <el-input
10.  class="input-new-tag"
11.  v-if="inputVisible"
12.  v-model="inputValue"
13.  ref="saveTagInput"
14.  size="small"
15.  @keyup.enter.native="handleInputConfirm"
16.  @blur="handleInputConfirm"
17. >
18. </el-input>
19. <el-button v-else class="button-new-tag" size="small" @click="showInput">+ New
  Tag</el-button>
20.
21. <style>
22.   .el-tag + .el-tag {
23.     margin-left: 10px;

```

```
24. }
25. .button-new-tag {
26.   margin-left: 10px;
27.   height: 32px;
28.   line-height: 30px;
29.   padding-top: 0;
30.   padding-bottom: 0;
31. }
32. .input-new-tag {
33.   width: 90px;
34.   margin-left: 10px;
35.   vertical-align: bottom;
36. }
37. </style>
38.
39. <script>
40. export default {
41.   data() {
42.     return {
43.       dynamicTags: ['标签一', '标签二', '标签三'],
44.       inputVisible: false,
45.       inputValue: ''
46.     };
47.   },
48.   methods: {
49.     handleClose(tag) {
50.       this.dynamicTags.splice(this.dynamicTags.indexOf(tag), 1);
51.     },
52.
53.     showInput() {
54.       this.inputVisible = true;
55.       this.$nextTick(_ => {
56.         this.$refs.saveTagInput.$refs.input.focus();
57.       });
58.     },
59.
60.     handleInputConfirm() {
61.       let inputValue = this.inputValue;
62.       if (inputValue) {
63.         this.dynamicTags.push(inputValue);
64.       }
65.       this.inputVisible = false;
```

```

66.         this inputValue = '';
67.     }
68. }
69. }
70. </script>

```

不同尺寸

Tag 组件提供除了默认值以外的三种尺寸，可以在不同场景下选择合适的按钮尺寸。

[默认标签](#) [中等标签](#) [小型标签](#) [超小标签](#)

额外的尺寸：[medium](#)、[small](#)、[mini](#)，通过设置 `size` 属性来配置它们。

1. `<el-tag closable>默认标签</el-tag>`
2. `<el-tag size="medium" closable>中等标签</el-tag>`
3. `<el-tag size="small" closable>小型标签</el-tag>`
4. `<el-tag size="mini" closable>超小标签</el-tag>`

[显示代码](#)

Attributes

参数	说明	类型	可选值	默认值
<code>type</code>	主题	<code>string</code>	<code>success/info/warning/danger</code>	—
<code>closable</code>	是否可关闭	<code>boolean</code>	—	<code>false</code>
<code>disable-transitions</code>	是否禁用渐变动画	<code>boolean</code>	—	<code>false</code>
<code>hit</code>	是否有边框描边	<code>boolean</code>	—	<code>false</code>
<code>color</code>	背景色	<code>string</code>	—	—
<code>size</code>	尺寸	<code>string</code>	<code>medium / small / mini</code>	—

Events

事件名称	说明	回调参数
<code>close</code>	关闭 Tag 时触发的事件	—

原文：<http://element-cn.eleme.io/#/zh-CN/component/tag>

Progress 进度条

Progress 进度条

用于展示操作进度，告知用户当前状态和预期。

线形进度条 – 百分比外显



Progress 组件设置 `percentage` 属性即可，表示进度条对应的百分比，必填，必须在 0-100。

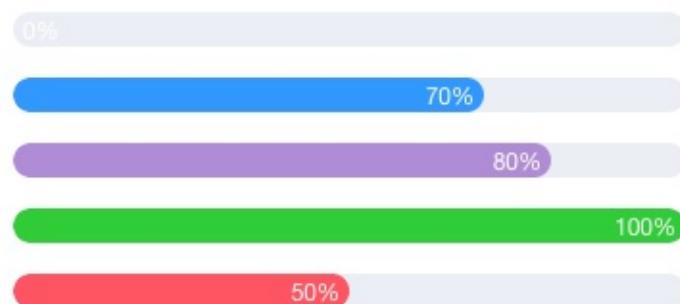
- ```

1. <el-progress :percentage="0"></el-progress>
2. <el-progress :percentage="70"></el-progress>
3. <el-progress :percentage="80" color="#8e71c7"></el-progress>
4. <el-progress :percentage="100" status="success"></el-progress>
5. <el-progress :percentage="50" status="exception"></el-progress>

```

### 线形进度条 – 百分比内显

百分比不占用额外控件，适用于文件上传等场景。



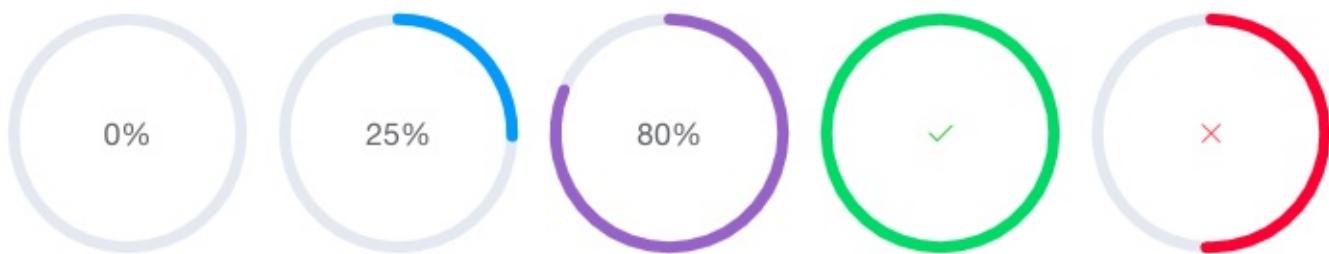
Progress 组件可通过 `stroke-width` 属性更改进度条的高度，并可通过 `text-inside` 属性来将进度条描述置于进度条内部。

```

1. <el-progress :text-inside="true" :stroke-width="18" :percentage="0"></el-progress>
2. <el-progress :text-inside="true" :stroke-width="18" :percentage="70"></el-progress>
3. <el-progress :text-inside="true" :stroke-width="18" :percentage="80" color="rgba(142, 113, 199, 0.7)"></el-progress>
4. <el-progress :text-inside="true" :stroke-width="18" :percentage="100" status="success"></el-progress>
5. <el-progress :text-inside="true" :stroke-width="18" :percentage="50" status="exception"></el-progress>

```

## 环形进度条



Progress 组件可通过 `type` 属性来指定使用环形进度条，在环形进度条中，还可以通过 `width` 属性来设置其大小。

```

1. <el-progress type="circle" :percentage="0"></el-progress>
2. <el-progress type="circle" :percentage="25"></el-progress>
3. <el-progress type="circle" :percentage="80" color="#8e71c7"></el-progress>
4. <el-progress type="circle" :percentage="100" status="success"></el-progress>
5. <el-progress type="circle" :percentage="50" status="exception"></el-progress>

```

## Attributes

| 参数                        | 说明                                            | 类型      | 可选值               | 默认值   |
|---------------------------|-----------------------------------------------|---------|-------------------|-------|
| <code>percentage</code>   | 百分比 (必填)                                      | number  | 0-100             | 0     |
| <code>type</code>         | 进度条类型                                         | string  | line/circle       | line  |
| <code>stroke-width</code> | 进度条的宽度，单位 px                                  | number  | -                 | 6     |
| <code>text-inside</code>  | 进度条显示文字内置在进度条内（只在 <code>type=line</code> 时可用） | boolean | -                 | false |
| <code>status</code>       | 进度条当前状态                                       | string  | success/exception | -     |
| <code>color</code>        | 进度条背景色（会覆盖 <code>status</code> 状态颜色）          | string  | -                 | -     |

|           |                               |         |   |      |
|-----------|-------------------------------|---------|---|------|
| width     | 环形进度条画布宽度（只在 type=circle 时可用） | number  |   | 126  |
| show-text | 是否显示进度条文字内容                   | boolean | - | true |

原文: <http://element-cn.eleme.io/#/zh-CN/component/progress>

# Tree 树形控件

## Tree 树形控件

用清晰的层级结构展示信息，可展开或折叠。

### 基础用法

基础的树形结构展示。

```
▼ 一级 1
 ▼ 二级 1-1
 三级 1-1-1
▶ 一级 2
▶ 一级 3
```

```
1. <el-tree :data="data" :props="defaultProps" @node-click="handleNodeClick"></el-
tree>
2.
3. <script>
4. export default {
5. data() {
6. return {
7. data: [
8. {label: '一级 1',
9. children: [
10. {label: '二级 1-1',
11. children: [
12. {label: '三级 1-1-1'
13. }]
14. }]
15. },
16. {label: '一级 2',
17. children: [
18. {label: '二级 2-1',
19. children: [
20. {label: '三级 2-1-1'
21. }]
22. },
23. {label: '二级 2-2',
```

```

24. children: [{

25. label: '三级 2-2-1'

26. }]

27. }]

28. }, {

29. label: '一级 3',

30. children: [{

31. label: '二级 3-1',

32. children: [{

33. label: '三级 3-1-1'

34. }]

35. }, {

36. label: '二级 3-2',

37. children: [{

38. label: '三级 3-2-1'

39. }]

40. }]

41. },

42. defaultProps: {

43. children: 'children',

44. label: 'label'

45. }

46. };

47. },

48. methods: {

49. handleNodeClick(data) {

50. console.log(data);

51. }

52. }

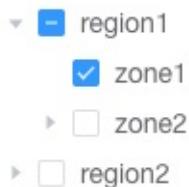
53. };

54. </script>

```

## 可选择

适用于需要选择层级时使用。



本例还展示了动态加载节点数据的方法。

```
1. <el-tree
2. :props="props"
3. :load="loadNode"
4. lazy
5. show-checkbox
6. @check-change="handleCheckChange">
7. </el-tree>
8.
9. <script>
10. export default {
11. data() {
12. return {
13. props: {
14. label: 'name',
15. children: 'zones'
16. },
17. count: 1
18. };
19. },
20. methods: {
21. handleCheckChange(data, checked, indeterminate) {
22. console.log(data, checked, indeterminate);
23. },
24. handleNodeClick(data) {
25. console.log(data);
26. },
27. loadNode(node, resolve) {
28. if (node.level === 0) {
29. return resolve([{ name: 'region1' }, { name: 'region2' }]);
30. }
31. if (node.level > 3) return resolve([]);
32.
33. var hasChild;
34. if (node.data.name === 'region1') {
35. hasChild = true;
36. } else if (node.data.name === 'region2') {
37. hasChild = false;
38. } else {
39. hasChild = Math.random() > 0.5;
40. }
41. }
42. }
43. }
```

```

41.
42. setTimeout(() => {
43. var data;
44. if (hasChild) {
45. data = [{{
46. name: 'zone' + this.count++
47. }, {
48. name: 'zone' + this.count++
49. }];
50. } else {
51. data = [];
52. }
53.
54. resolve(data);
55. }, 500);
56. }
57. }
58. }
59. </script>

```

## 懒加载自定义叶子节点



由于在点击节点时才进行该层数据的获取，默认情况下 Tree 无法预知某个节点是否为叶子节点，所以会为每个节点添加一个下拉按钮，如果节点没有下层数据，则点击后下拉按钮会消失。同时，你也可以提前告知 Tree 某个节点是否为叶子节点，从而避免在叶子节点前渲染下拉按钮。

```

1. <el-tree
2. :props="props1"
3. :load="loadNode1"
4. lazy
5. show-checkbox>
6. </el-tree>
7.
8. <script>
9. export default {
10. data() {
11. return {

```

```

12. props1: {
13. label: 'name',
14. children: 'zones',
15. isLeaf: 'leaf'
16. },
17. },
18. },
19. methods: {
20. loadNode1(node, resolve) {
21. if (node.level === 0) {
22. return resolve([{ name: 'region' }]);
23. }
24. if (node.level > 1) return resolve([]);
25.
26. setTimeout(() => {
27. const data = [
28. {
29. name: 'leaf',
30. leaf: true
31. },
32. {
33. name: 'zone'
34. }];
35.
36. resolve(data);
37. }, 500);
38. }
39. }

```

## 默认展开和默认选中

可将 Tree 的某些节点设置为默认展开或默认选中

- ▶  一级 1
- ▼  一级 2
  - 二级 2-1
  - 二级 2-2
- ▼  一级 3
  - 二级 3-1
  - 二级 3-2

分别通过 `default-expanded-keys` 和 `default-checked-keys` 设置默认展开和默认选中的节点。需要注意的是，此时必须设置 `node-key`，其值为节点数据中的一个字段名，该字段在整棵树中是唯一的。

```
1. <el-tree
2. :data="data2"
3. show-checkbox
4. node-key="id"
5. :default-expanded-keys="[2, 3]"
6. :default-checked-keys="[5]"
7. :props="defaultProps">
8. </el-tree>
9.
10. <script>
11. export default {
12. data() {
13. return {
14. data2: [
15. { id: 1,
16. label: '一级 1',
17. children: [
18. { id: 4,
19. label: '二级 1-1',
20. children: [
21. { id: 9,
22. label: '三级 1-1-1'
23. },
24. { id: 10,
25. label: '三级 1-1-2'
26. }
27.]
28. },
29. { id: 2,
30. label: '一级 2',
31. children: [
32. { id: 5,
33. label: '二级 2-1'
34. },
35. { id: 6,
36. label: '二级 2-2'
37. }
38.],
39. }
40.]
41. }
42.]
43. }
44. }
45. }
46.
```

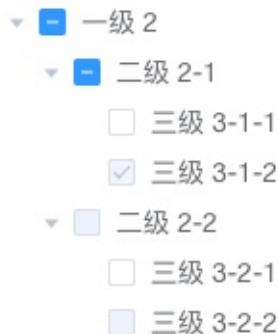
```

39. id: 3,
40. label: '一级 3',
41. children: [{
42. id: 7,
43. label: '二级 3-1'
44. }, {
45. id: 8,
46. label: '二级 3-2'
47. }]
48.],
49. defaultProps: {
50. children: 'children',
51. label: 'label'
52. }
53. };
54. }
55. };
56. </script>

```

## 禁用状态

可将 Tree 的某些节点设置为禁用状态



通过 `disabled` 设置禁用状态。

```

1. <el-tree
2. :data="data3"
3. show-checkbox
4. node-key="id"
5. :default-expanded-keys="[2, 3]"
6. :default-checked-keys="[5]>
7. </el-tree>
8.

```

```
9. <script>
10. export default {
11. data() {
12. return {
13. data3: [{{
14. id: 1,
15. label: '一级 2',
16. children: [{{
17. id: 3,
18. label: '二级 2-1',
19. children: [{{
20. id: 4,
21. label: '三级 3-1-1'
22. }, {{
23. id: 5,
24. label: '三级 3-1-2',
25. disabled: true
26. }]
27. }, {{
28. id: 2,
29. label: '二级 2-2',
30. disabled: true,
31. children: [{{
32. id: 6,
33. label: '三级 3-2-1'
34. }, {{
35. id: 7,
36. label: '三级 3-2-2',
37. disabled: true
38. }]
39. },]
40. },],
41. defaultProps: {
42. children: 'children',
43. label: 'label'
44. }
45. };
46. }
47. };
48. </script>
```

## 树节点的选择

通过 node 获取通过 key 获取通过 node 设置通过 key 设置清空

本例展示如何获取和设置选中节点。获取和设置各有两种方式：通过 node 或通过 key。如果需要通过 key 来获取或设置，则必须设置 `node-key`。

```

1. <el-tree
2. :data="data2"
3. show-checkbox
4. default-expand-all
5. node-key="id"
6. ref="tree"
7. highlight-current
8. :props="defaultProps">
9. </el-tree>
10.
11. <div class="buttons">
12. <el-button @click="getCheckedNodes">通过 node 获取</el-button>
13. <el-button @click="getCheckedKeys">通过 key 获取</el-button>
14. <el-button @click="setCheckedNodes">通过 node 设置</el-button>
15. <el-button @click="setCheckedKeys">通过 key 设置</el-button>
16. <el-button @click="resetChecked">清空</el-button>
17. </div>
18.
19. <script>
20. export default {
21. methods: {
22. getCheckedNodes() {
23. console.log(this.$refs.tree.getCheckedNodes());
24. },
25. getCheckedKeys() {

```

```
26. console.log(this.$refs.tree.getCheckedKeys());
27. },
28. setCheckedNodes() {
29. this.$refs.tree.setCheckedNodes([
30. {
31. id: 5,
32. label: '二级 2-1'
33. },
34. {
35. id: 9,
36. label: '三级 1-1-1'
37. }
38.]);
39. },
40. setCheckedKeys() {
41. this.$refs.tree.setCheckedKeys([3]);
42. },
43. resetChecked() {
44. this.$refs.tree.setCheckedKeys([]);
45. },
46. data() {
47. return {
48. data2: [
49. {
50. id: 1,
51. label: '一级 1',
52. children: [
53. {
54. id: 4,
55. label: '二级 1-1',
56. children: [
57. {
58. id: 9,
59. label: '三级 1-1-1'
60. },
61. {
62. id: 10,
63. label: '三级 1-1-2'
64. }
65.]
66. }
67.],
68. }
69.],
70. data3: [
71. {
72. id: 2,
73. label: '一级 2',
74. children: [
75. {
76. id: 5,
77. label: '二级 2-1'
78. }
79.],
80. }
81.]
82. };
83. }
84. }
```

```

68. id: 6,
69. label: '二级 2-2'
70. }]
71. },
72. id: 3,
73. label: '一级 3',
74. children: [{
75. id: 7,
76. label: '二级 3-1'
77. }, {
78. id: 8,
79. label: '二级 3-2'
80. }]
81.],
82. defaultProps: {
83. children: 'children',
84. label: 'label'
85. }
86.);
87. }
88.);
89. </script>

```

## 自定义节点内容

节点的内容支持自定义，可以在节点区添加按钮或图标等内容

| 使用 render-content                 | 使用 scoped slot                                |
|-----------------------------------|-----------------------------------------------|
| ▼ <input type="checkbox"/> 一级 1   | <a href="#">Append</a> <a href="#">Delete</a> |
| ▼ <input type="checkbox"/> 二级 1-1 | <a href="#">Append</a> <a href="#">Delete</a> |
| <input type="checkbox"/> 三级 1-1-1 | <a href="#">Append</a> <a href="#">Delete</a> |
| <input type="checkbox"/> 三级 1-1-2 | <a href="#">Append</a> <a href="#">Delete</a> |
| ▼ <input type="checkbox"/> 一级 2   | <a href="#">Append</a> <a href="#">Delete</a> |
| <input type="checkbox"/> 二级 2-1   | <a href="#">Append</a> <a href="#">Delete</a> |
| <input type="checkbox"/> 二级 2-2   | <a href="#">Append</a> <a href="#">Delete</a> |
| ▼ <input type="checkbox"/> 一级 3   | <a href="#">Append</a> <a href="#">Delete</a> |
| <input type="checkbox"/> 二级 3-1   | <a href="#">Append</a> <a href="#">Delete</a> |
| <input type="checkbox"/> 二级 3-2   | <a href="#">Append</a> <a href="#">Delete</a> |
| ▼ <input type="checkbox"/> 一级 1   |                                               |
| ▼ <input type="checkbox"/> 二级 1-1 |                                               |
| <input type="checkbox"/> 三级 1-1-1 |                                               |
| <input type="checkbox"/> 三级 1-1-2 |                                               |
| ▼ <input type="checkbox"/> 一级 2   |                                               |
| <input type="checkbox"/> 二级 2-1   |                                               |
| <input type="checkbox"/> 二级 2-2   |                                               |
| ▼ <input type="checkbox"/> 一级 3   |                                               |
| <input type="checkbox"/> 二级 3-1   |                                               |
| <input type="checkbox"/> 二级 3-2   |                                               |

可以通过两种方法进行树节点内容的自定义：`render-content` 和 `scoped slot`。使用 `render-content` 指定渲染函数，该函数返回需要的节点区内容即可。渲染函数的用法请参考 Vue 文档。使

用 scoped slot 会传入两个参数 `node` 和 `data`，分别表示当前节点的 Node 对象和当前节点的数据。注意：由于 jsfiddle 不支持 JSX 语法，所以 `render-content` 示例在 jsfiddle 中无法运行。但是在实际的项目中，只要正确地配置了相关依赖，就可以正常运行。

```
1. <div class="custom-tree-container">
2. <div class="block">
3. <p>使用 render-content</p>
4. <el-tree
5. :data="data4"
6. show-checkbox
7. node-key="id"
8. default-expand-all
9. :expand-on-click-node="false"
10. :render-content="renderContent">
11. </el-tree>
12. </div>
13. <div class="block">
14. <p>使用 scoped slot</p>
15. <el-tree
16. :data="data5"
17. show-checkbox
18. node-key="id"
19. default-expand-all
20. :expand-on-click-node="false">
21.
22. {{ node.label }}
23.
24. <el-button
25. type="text"
26. size="mini"
27. @click="() => append(data)">
28. Append
29. </el-button>
30. <el-button
31. type="text"
32. size="mini"
33. @click="() => remove(node, data)">
34. Delete
35. </el-button>
36.
37.
38. </el-tree>
```

```
39. </div>
40. </div>
41.
42. <script>
43. let id = 1000;
44.
45. export default {
46. data() {
47. const data = [{
48. id: 1,
49. label: '一级 1',
50. children: [{
51. id: 4,
52. label: '二级 1-1',
53. children: [{
54. id: 9,
55. label: '三级 1-1-1'
56. }, {
57. id: 10,
58. label: '三级 1-1-2'
59. }]
60. }]
61. }, {
62. id: 2,
63. label: '一级 2',
64. children: [{
65. id: 5,
66. label: '二级 2-1'
67. }, {
68. id: 6,
69. label: '二级 2-2'
70. }]
71. }, {
72. id: 3,
73. label: '一级 3',
74. children: [{
75. id: 7,
76. label: '二级 3-1'
77. }, {
78. id: 8,
79. label: '二级 3-2'
80. }]
81. }
82. }
83.
```

```
81. }];
82. return {
83. data4: JSON.parse(JSON.stringify(data)),
84. data5: JSON.parse(JSON.stringify(data))
85. }
86. },
87.
88. methods: {
89. append(data) {
90. const newChild = { id: id++, label: 'testtest', children: [] };
91. if (!data.children) {
92. this.$set(data, 'children', []);
93. }
94. data.children.push(newChild);
95. },
96.
97. remove(node, data) {
98. const parent = node.parent;
99. const children = parent.data.children || parent.data;
100. const index = children.findIndex(d => d.id === data.id);
101. children.splice(index, 1);
102. },
103.
104. renderContent(h, { node, data, store }) {
105. return (
106.
107. {node.label}
108.
109. <el-button size="mini" type="text" on-click={()=>
110. this.append(data)}>Append</el-button>
111. <el-button size="mini" type="text" on-click={()=>
112. this.remove(node, data)}>Delete</el-button>
113.
114.
115.);
116. </script>
117.
118. <style>
119. .custom-tree-node {
120. flex: 1;
```

```

121. display: flex;
122. align-items: center;
123. justify-content: space-between;
124. font-size: 14px;
125. padding-right: 8px;
126. }
127. </style>

```

## 节点过滤

通过关键字过滤树节点

输入关键字进行过滤

- ▼ 一级 1
  - ▼ 二级 1-1
    - 三级 1-1-1
    - 三级 1-1-2
- ▼ 一级 2
  - 二级 2-1
  - 二级 2-2
- ▼ 一级 3
  - 二级 3-1
  - 二级 3-2

在需要对节点进行过滤时，调用 Tree 实例的 `filter` 方法，参数为关键字。需要注意的是，此时需要设置 `filter-node-method`，值为过滤函数。

```

1. <el-input
2. placeholder="输入关键字进行过滤"
3. v-model="filterText">
4. </el-input>
5.
6. <el-tree
7. class="filter-tree"
8. :data="data2"
9. :props="defaultProps"
10. default-expand-all
11. :filter-node-method="filterNode"
12. ref="tree2">
13. </el-tree>
14.

```

```
15. <script>
16. export default {
17. watch: {
18. filterText(val) {
19. this.$refs.tree2.filter(val);
20. }
21. },
22.
23. methods: {
24. filterNode(value, data) {
25. if (!value) return true;
26. return data.label.indexOf(value) !== -1;
27. }
28. },
29.
30. data() {
31. return {
32. filterText: '',
33. data2: [
34. {
35. id: 1,
36. label: '一级 1',
37. children: [
38. {
39. id: 4,
40. label: '二级 1-1',
41. children: [
42. {
43. id: 9,
44. label: '三级 1-1-1'
45. },
46. {
47. id: 10,
48. label: '三级 1-1-2'
49. }
50.]
51. }
52.],
53. {
54. id: 2,
55. label: '一级 2',
56. children: [
57. {
58. id: 5,
59. label: '二级 2-1'
60. },
61. {
62. id: 6,
63. label: '二级 2-2'
64. }
65.]
66. }
67.],
68. }
69. }
70. }
71.
```

```

57. },
58. id: 3,
59. label: '一级 3',
60. children: [
61. {
62. id: 7,
63. label: '二级 3-1'
64. },
65. {
66. id: 8,
67. label: '二级 3-2'
68. }
69.],
70. defaultProps: {
71. children: 'children',
72. label: 'label'
73. }
74. };
75. </script>

```

## 手风琴模式

对于同级别的节点，每次只能展开一个

- ▶ 一级 1
- ▼ 一级 2
  - ▼ 二级 2-1
    - 三级 2-1-1
  - ▶ 二级 2-2
- ▶ 一级 3

```

1. <el-tree
2. :data="data"
3. :props="defaultProps"
4. accordion
5. @node-click="handleNodeClick">
6. </el-tree>
7.
8. <script>
9. export default {
10. data() {

```

```
11. return {
12. data: [
13. {
14. label: '一级 1',
15. children: [
16. {
17. label: '二级 1-1',
18. children: [
19. {
20. label: '三级 1-1-1'
21. }
22.]
23. }
24.],
25. label: '一级 2',
26. children: [
27. {
28. label: '二级 2-1',
29. children: [
30. {
31. label: '三级 2-1-1'
32. }
33.]
34. }
35.],
36. label: '一级 3',
37. children: [
38. {
39. label: '二级 3-1',
40. children: [
41. {
42. label: '三级 3-1-1'
43. }
44.]
45. }
46.],
47.],
48. defaultProps: {
49. children: 'children',
50. label: 'label'
51. }
52. },
53. },
```

```

53. methods: {
54. handleNodeClick(data) {
55. console.log(data);
56. }
57. }
58. };
59. </script>

```

## 可拖拽节点

通过 `draggable` 属性可让节点变为可拖拽。

- ▼ 一级 1
  - ▼ 二级 1-1
    - 三级 1-1-2
    - 三级 1-1-1
- ▼ 一级 2
  - 二级 2-1
  - 二级 2-2
- ▼ 一级 3
  - 二级 3-1
  - ▼ 二级 3-2
    - 三级 3-2-1
    - 三级 3-2-2
    - 三级 3-2-3

```

1. <el-tree
2. :data="data6"
3. node-key="id"
4. default-expand-all
5. @node-drag-start="handleDragStart"
6. @node-drag-enter="handleDragEnter"
7. @node-drag-leave="handleDragLeave"
8. @node-drag-over="handleDragOver"
9. @node-drag-end="handleDragEnd"
10. @node-drop="handleDrop"
11. draggable
12. :allow-drop="allowDrop"
13. :allow-drag="allowDrag">
14. </el-tree>

```

```
15.
16. <script>
17. export default {
18. data() {
19. return {
20. data6: [{
21. id: 1,
22. label: '一级 1',
23. children: [{
24. id: 4,
25. label: '二级 1-1',
26. children: [{
27. id: 9,
28. label: '三级 1-1-1'
29. }, {
30. id: 10,
31. label: '三级 1-1-2'
32. }]
33. }]
34. }, {
35. id: 2,
36. label: '一级 2',
37. children: [{
38. id: 5,
39. label: '二级 2-1'
40. }, {
41. id: 6,
42. label: '二级 2-2'
43. }]
44. }, {
45. id: 3,
46. label: '一级 3',
47. children: [{
48. id: 7,
49. label: '二级 3-1'
50. }, {
51. id: 8,
52. label: '二级 3-2',
53. children: [{
54. id: 11,
55. label: '三级 3-2-1'
56. }],
57. }],
58. }],
59. }]
60. }]
61. }]
62. }
```

```
57. id: 12,
58. label: '三级 3-2-2'
59. }, {
60. id: 13,
61. label: '三级 3-2-3'
62. }]
63.]
64.],
65. defaultProps: {
66. children: 'children',
67. label: 'label'
68. }
69.);
70. },
71. methods: {
72. handleDragStart(node, ev) {
73. console.log('drag start', node);
74. },
75. handleDragEnter(draggingNode, dropNode, ev) {
76. console.log('tree drag enter: ', dropNode.label);
77. },
78. handleDragLeave(draggingNode, dropNode, ev) {
79. console.log('tree drag leave: ', dropNode.label);
80. },
81. handleDragOver(draggingNode, dropNode, ev) {
82. console.log('tree drag over: ', dropNode.label);
83. },
84. handleDragEnd(draggingNode, dropNode, dropType, ev) {
85. console.log('tree drag end: ', dropNode && dropNode.label, dropType);
86. },
87. handleDrop(draggingNode, dropNode, dropType, ev) {
88. console.log('tree drop: ', dropNode.label, dropType);
89. },
90. allowDrop(draggingNode, dropNode, type) {
91. if (dropNode.data.label === '二级 3-1') {
92. return type !== 'inner';
93. } else {
94. return true;
95. }
96. },
97. allowDrag(draggingNode) {
98. return draggingNode.data.label.indexOf('三级 3-2-2') === -1;
```

```

99. }
100. }
101. };
102. </script>

```

## Attributes

| 参数                    | 说明                                                             | 类型                                 | 可选值 | 默认值   |
|-----------------------|----------------------------------------------------------------|------------------------------------|-----|-------|
| data                  | 展示数据                                                           | array                              | —   | —     |
| empty-text            | 内容为空的时候展示的文本                                                   | String                             | —   | —     |
| node-key              | 每个树节点用来作为唯一标识的属性，整棵树应该是唯一的                                     | String                             | —   | —     |
| props                 | 配置选项，具体看下表                                                     | object                             | —   | —     |
| render-after-expand   | 是否在第一次展开某个树节点后才渲染其子节点                                          | boolean                            | —   | true  |
| load                  | 加载子树数据的方法，仅当 lazy 属性为 true 时生效                                 | function(node, resolve)            | —   | —     |
| render-content        | 树节点的内容区的渲染 Function                                            | Function(h, { node, data, store }) | —   | —     |
| highlight-current     | 是否高亮当前选中节点，默认值是 false。                                         | boolean                            | —   | false |
| default-expand-all    | 是否默认展开所有节点                                                     | boolean                            | —   | false |
| expand-on-click-node  | 是否在点击节点的时候展开或者收缩节点， 默认值为 true，如果为 false，则只有点箭头图标的时候才会展开或者收缩节点。 | boolean                            | —   | true  |
| check-on-click-node   | 是否在点击节点的时候选中节点， 默认值为 false，即只有在点击复选框时才会选中节点。                   | boolean                            | —   | false |
| auto-expand-parent    | 展开子节点的时候是否自动展开父节点                                              | boolean                            | —   | true  |
| default-expanded-keys | 默认展开的节点的 key 的数组                                               | array                              | —   | —     |
| show-checkbox         | 节点是否可被选择                                                       | boolean                            | —   | false |
| check-stricly         | 在显示复选框的情况下，是否严格的遵循父子不互相关联的做法， 默认为 false                        | boolean                            | —   | false |
| default-checked-keys  | 默认勾选的节点的 key 的数组                                               | array                              | —   | —     |
| filter-node-method    | 对树节点进行筛选时执行的方法，返回 true 表示这个节点可以显示，返回 false 则表示这个节点会被隐藏         | Function(value, data, node)        | —   | —     |

|            |                                                                                  |                                        |   |       |
|------------|----------------------------------------------------------------------------------|----------------------------------------|---|-------|
| accordion  | 是否每次只打开一个同级树节点展开                                                                 | boolean                                | - | false |
| indent     | 相邻级节点间的水平缩进，单位为像素                                                                | number                                 | - | 16    |
| lazy       | 是否懒加载子节点，需与 load 方法结合使用                                                          | boolean                                | - | false |
| draggable  | 是否开启拖拽节点功能                                                                       | boolean                                | - | false |
| allow-drag | 判断节点能否被拖拽                                                                        | Function(node)                         | - | -     |
| allow-drop | 拖拽时判定目标节点能否被放置。<br>参数有三种情况：'prev'、'inner' 和 'next'，分别表示放置在目标节点前、插入至目标节点和放置在目标节点后 | Function(draggingNode, dropNode, type) | - | -     |

## props

| 参数       | 说明                              | 类型                            | 可选值 | 默认值 |
|----------|---------------------------------|-------------------------------|-----|-----|
| label    | 指定节点标签为节点对象的某个属性值               | string, function(data, node)  | -   | -   |
| children | 指定子树为节点对象的某个属性值                 | string                        | -   | -   |
| disabled | 指定节点选择框是否禁用为节点对象的某个属性值          | boolean, function(data, node) | -   | -   |
| isLeaf   | 指定节点是否为叶子节点，仅在指定了 lazy 属性的情况下生效 | boolean, function(data, node) | -   | -   |

## 方法

**Tree** 内部使用了 Node 类型的对象来包装用户传入的数据，用来保存目前节点的状态。 **Tree** 拥有如下方法：

| 方法名               | 说明                                                      | 参数                                                                                              |
|-------------------|---------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| filter            | 对树节点进行筛选操作                                              | 接收一个任意类型的参数，该参数会在 filter-node-method 中作为第一个参数                                                   |
| updateKeyChildren | 通过 keys 设置节点子元素，使用此方法必须设置 node-key 属性                   | (key, data) 接收两个参数，1. 节点 key 2. 节点数据的数组                                                         |
| getCheckedNodes   | 若节点可被选择（即 show-checkbox 为 true），则返回目前被选中的节点所组成的数组       | (leafOnly, includeHalfChecked) 接收两个 boolean 类型的参数，1. 是否只是叶子节点，默认值为 false 2. 是否包含半选节点，默认值为 false |
| setCheckedNodes   | 设置目前勾选的节点，使用此方法必须设置 node-key 属性                         | (nodes) 接收勾选节点数据的数组                                                                             |
| getCheckedKeys    | 若节点可被选择（即 show-checkbox 为 true），则返回目前被选中的节点的 key 所组成的数组 | (leafOnly) 接收一个 boolean 类型的参数，若为 true 则仅返回被选中的叶子节点的 keys，默认值为 false                             |
|                   | 通过 keys 设置目前勾选                                          | (keys, leafOnly) 接收两个参数，1. 勾选节                                                                  |

|                     |                                                         |                                                                                                             |
|---------------------|---------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| setCheckedKeys      | 通过 keys 设置目前勾选的节点，使用此方法必须设置 node-key 属性                 | 点的 key 的数组 2. boolean 类型的参数，若为 true 则仅设置叶子节点的选中状态，默认值为 false                                                |
| setChecked          | 通过 key / data 设置某个节点的勾选状态，使用此方法必须设置 node-key 属性         | (key/data, checked, deep) 接收三个参数，1. 勾选节点的 key 或者 data 2. boolean 类型，节点是否选中 3. boolean 类型，是否设置子节点，，默认为 false |
| getHalfCheckedNodes | 若节点可被选择（即 show-checkbox 为 true），则返回目前半选中的节点所组成的数组       | -                                                                                                           |
| getHalfCheckedKeys  | 若节点可被选择（即 show-checkbox 为 true），则返回目前半选中的节点的 key 所组成的数组 | -                                                                                                           |
| getCurrentKey       | 获取当前被选中节点的 key，使用此方法必须设置 node-key 属性，若没有节点被选中则返回 null   | -                                                                                                           |
| getCurrentNode      | 获取当前被选中节点的 node，若没有节点被选中则返回 null                        | -                                                                                                           |
| setCurrentKey       | 通过 key 设置某个节点的当前选中状态，使用此方法必须设置 node-key 属性              | (key) 待被选节点的 key，若为 null 则取消当前高亮的节点                                                                         |
| setCurrentNode      | 通过 node 设置某个节点的当前选中状态，使用此方法必须设置 node-key 属性             | (node) 待被选节点的 node                                                                                          |
| getNode             | 根据 data 或者 key 拿到 Tree 组件中的 node                        | (data) 要获得 node 的 key 或者 data                                                                               |
| remove              | 删除 Tree 中的一个节点，使用此方法必须设置 node-key 属性                    | (data) 要删除的节点的 data 或者 node                                                                                 |
| append              | 为 Tree 中的一个节点追加一个子节点                                    | (data, parentNode) 接收两个参数，1. 要追加的子节点的 data 2. 子节点的 parent 的 data、key 或者 node                                |
| insertBefore        | 为 Tree 的一个节点的前面增加一个节点                                   | (data, refNode) 接收两个参数，1. 要增加的节点的 data 2. 要增加的节点的后一个节点的 data、key 或者 node                                    |
| insertAfter         | 为 Tree 的一个节点的后面增加一个节点                                   | (data, refNode) 接收两个参数，1. 要增加的节点的 data 2. 要增加的节点的前一个节点的 data、key 或者 node                                    |

## Events

| 事件名称       | 说明               | 回调参数                                                  |
|------------|------------------|-------------------------------------------------------|
| node-click | 节点被点击时的回调        | 共三个参数，依次为：传递给 data 属性的数组中该节点所对应的对象、节点对应的 Node、节点组件本身。 |
| node-      | 当某一节点被鼠标右键点击时会触发 | 共四个参数，依次为：event、传递给 data 属性的数组中该节点                    |

|                 |                                  |                                                                                                                              |
|-----------------|----------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| contextmenu     | 该事件                              | 所对应的对象、节点对应的 Node、节点组件本身。                                                                                                    |
| check-change    | 节点选中状态发生变化时的回调                   | 共三个参数，依次为：传递给 <code>data</code> 属性的数组中该节点所对应的对象、节点本身是否被选中、节点的子树中是否有被选中的节点                                                    |
| check           | 当复选框被点击的时候触发                     | 共两个参数，依次为：传递给 <code>data</code> 属性的数组中该节点所对应的对象、树目前的选中状态对象，包含 checkedNodes、checkedKeys、halfCheckedNodes、halfCheckedKeys 四个属性 |
| current-change  | 当前选中节点变化时触发的事件                   | 共两个参数，依次为：当前节点的数据，当前节点的 Node 对象                                                                                              |
| node-expand     | 节点被展开时触发的事件                      | 共三个参数，依次为：传递给 <code>data</code> 属性的数组中该节点所对应的对象、节点对应的 Node、节点组件本身                                                            |
| node-collapse   | 节点被关闭时触发的事件                      | 共三个参数，依次为：传递给 <code>data</code> 属性的数组中该节点所对应的对象、节点对应的 Node、节点组件本身                                                            |
| node-drag-start | 节点开始拖拽时触发的事件                     | 共两个参数，依次为：被拖拽节点对应的 Node、event                                                                                                |
| node-drag-enter | 拖拽进入其他节点时触发的事件                   | 共三个参数，依次为：被拖拽节点对应的 Node、所进入节点对应的 Node、event                                                                                  |
| node-drag-leave | 拖拽离开某个节点时触发的事件                   | 共三个参数，依次为：被拖拽节点对应的 Node、所离开节点对应的 Node、event                                                                                  |
| node-drag-over  | 在拖拽节点时触发的事件（类似浏览器的 mouseover 事件） | 共三个参数，依次为：被拖拽节点对应的 Node、当前进入节点对应的 Node、event                                                                                 |
| node-drag-end   | 拖拽结束时（可能未成功）触发的事件                | 共四个参数，依次为：被拖拽节点对应的 Node、结束拖拽时最后进入的节点（可能为空）、被拖拽节点的放置位置（before、after、inner）、event                                              |
| node-drop       | 拖拽成功完成时触发的事件                     | 共四个参数，依次为：被拖拽节点对应的 Node、结束拖拽时最后进入的节点、被拖拽节点的放置位置（before、after、inner）、event                                                    |

## Scoped Slot

| name | 说明                           |
|------|------------------------------|
| -    | 自定义树节点的内容，参数为 { node, data } |

原文：<http://element-cn.eleme.io/#/zh-CN/component/tree>

# Pagination 分页

## Pagination 分页

当数据量过多时，使用分页分解数据。

### 基础用法

页数较少时的效果

< 1 2 3 4 5 >

大于 7 页时的效果

< 1 2 3 4 5 6 ... 100 >

设置 `layout`，表示需要显示的内容，用逗号分隔，布局元素会依次显示。`prev` 表示上一页，`next` 为下一页，`pager` 表示页码列表，除此以外还提供了 `jumper` 和 `total`，`size` 和特殊的布局符号 `->`，`->` 后的元素会靠右显示，`jumper` 表示跳页元素，`total` 表示显示页码总数，`size` 用于设置每页显示的页码数量。

```

1. <div class="block">
2. 页数较少时的效果
3. <el-pagination
4. layout="prev, pager, next"
5. :total="50">
6. </el-pagination>
7. </div>
8. <div class="block">
9. 大于 7 页时的效果
10. <el-pagination
11. layout="prev, pager, next"
12. :total="1000">
13. </el-pagination>
14. </div>

```

### 设置最大页码按钮数

< 1 2 3 4 5 6 7 8 9 10 ... 50 >

默认情况下，当总页数超过 7 页时，`Pagination` 会折叠多余的页码按钮。通过 `pager-count` 属性

性可以设置最大页码按钮数。

```

1. <el-pagination
2. :page-size="20"
3. :pager-count="11"
4. layout="prev, pager, next"
5. :total="1000">
6. </el-pagination>
```

## 带有背景色的分页



设置 `background` 属性可以为分页按钮添加背景色。

```

1. <el-pagination
2. background
3. layout="prev, pager, next"
4. :total="1000">
5. </el-pagination>
```

## 小型分页

在空间有限的情况下，可以使用简单的小型分页。



只需要一个 `small` 属性，它接受一个 `Boolean`，默认为 `false`，设为 `true` 即可启用。

```

1. <el-pagination
2. small
3. layout="prev, pager, next"
4. :total="50">
5. </el-pagination>
```

## 附加功能

根据场景需要，可以添加其他功能模块。

The image shows four examples of the El-Pagination component:

- 显示总数**: Shows a total of 1000 items with page numbers 1 through 10.
- 调整每页显示条数**: Allows changing the number of items per page to 100, with page numbers 1 through 10.
- 直接前往**: Direct navigation to page 5 of 10.
- 完整功能**: Shows a total of 400 items with page numbers 1 through 4.

此例是一个完整的用例，使用了 `size-change` 和 `current-change` 事件来处理页码大小和当前页变动时候触发的事件。`page-sizes` 接受一个整型数组，数组元素为展示的选择每页显示个数的选项，`[100, 200, 300, 400]` 表示四个选项，每页显示 100 个，200 个，300 个或者 400 个。

```

1. <template>
2. <div class="block">
3. 显示总数
4. <el-pagination
5. @size-change="handleSizeChange"
6. @current-change="handleCurrentChange"
7. :current-page.sync="currentPage1"
8. :page-size="100"
9. layout="total, prev, pager, next"
10. :total="1000">
11. </el-pagination>
12. </div>
13. <div class="block">
14. 调整每页显示条数
15. <el-pagination
16. @size-change="handleSizeChange"
17. @current-change="handleCurrentChange"
18. :current-page.sync="currentPage2"
19. :page-sizes="[100, 200, 300, 400]"
20. :page-size="100"
21. layout="sizes, prev, pager, next"
22. :total="1000">

```

```
23. </el-pagination>
24. </div>
25. <div class="block">
26. 直接前往
27. <el-pagination
28. @size-change="handleSizeChange"
29. @current-change="handleCurrentChange"
30. :current-page.sync="currentPage3"
31. :page-size="100"
32. layout="prev, pager, next, jumper"
33. :total="1000">
34. </el-pagination>
35. </div>
36. <div class="block">
37. 完整功能
38. <el-pagination
39. @size-change="handleSizeChange"
40. @current-change="handleCurrentChange"
41. :current-page="currentPage4"
42. :page-sizes="[100, 200, 300, 400]"
43. :page-size="100"
44. layout="total, sizes, prev, pager, next, jumper"
45. :total="400">
46. </el-pagination>
47. </div>
48. </template>
49. <script>
50. export default {
51. methods: {
52. handleSizeChange(val) {
53. console.log(`每页 ${val} 条`);
54. },
55. handleCurrentChange(val) {
56. console.log(`当前页: ${val}`);
57. }
58. },
59. data() {
60. return {
61. currentPage1: 5,
62. currentPage2: 5,
63. currentPage3: 5,
64. currentPage4: 4
```

```

65. };
66. }
67. }
68. </script>

```

[显示代码](#)

## Attributes

| 参数           | 说明                                                                            | 类型       | 可选值                                                | 默认值                                    |
|--------------|-------------------------------------------------------------------------------|----------|----------------------------------------------------|----------------------------------------|
| small        | 是否使用小型分页样式                                                                    | boolean  | —                                                  | false                                  |
| background   | 是否为分页按钮添加背景色                                                                  | boolean  | —                                                  | false                                  |
| page-size    | 每页显示条目个数，支持 .sync 修饰符                                                         | number   | —                                                  | 10                                     |
| total        | 总条目数                                                                          | number   | —                                                  | —                                      |
| page-count   | 总页数，total 和 page-count 设置任意一个就可以达到显示页码的功能；如果要支持 page-sizes 的更改，则需要使用 total 属性 | Number   | —                                                  | —                                      |
| page-count   | 页码按钮的数量，当总页数超过该值时会折叠                                                          | number   | 大于等于 5 且小于等于 21 的奇数                                | 7                                      |
| current-page | 当前页数，支持 .sync 修饰符                                                             | number   | —                                                  | 1                                      |
| layout       | 组件布局，子组件名用逗号分隔                                                                | String   | sizes, nprev, nager, next, jumper, ->, total, slot | 'prev, pager, next, jumper, ->, total' |
| page-sizes   | 每页显示个数选择器的选项设置                                                                | number[] | —                                                  | [10, 20, 30, 40, 50, 100]              |
| popper-class | 每页显示个数选择器的下拉框类名                                                               | string   | —                                                  | —                                      |
| prev-text    | 替代图标显示的上一页文字                                                                  | string   | —                                                  | —                                      |
| next-text    | 替代图标显示的下一页文字                                                                  | string   | —                                                  | —                                      |
| disabled     | 是否禁用                                                                          | boolean  | —                                                  | false                                  |

## Events

| 事件名称           | 说明                 | 回调参数 |
|----------------|--------------------|------|
| size-change    | pageSize 改变时会触发    | 每页条数 |
| current-change | currentPage 改变时会触发 | 当前页  |
| prev-click     | 用户点击上一页按钮改变当前页后触发  | 当前页  |

|            |                   |     |
|------------|-------------------|-----|
| next-click | 用户点击下一页按钮改变当前页后触发 | 当前页 |
|------------|-------------------|-----|

## Slot

| name | 说明                                                  |
|------|-----------------------------------------------------|
| -    | 自定义内容，需要在 <code>layout</code> 中列出 <code>slot</code> |

原文: <http://element-cn.eleme.io/#/zh-CN/component/pagination>

# Badge 标记

## Badge 标记

出现在按钮、图标旁的数字或状态标记。

### 基础用法

展示新消息数量。



定义 `value` 属性，它接受 `Number` 或者 `String`。

```
1. <el-badge :value="12" class="item">
2. <el-button size="small">评论</el-button>
3. </el-badge>
4. <el-badge :value="3" class="item">
5. <el-button size="small">回复</el-button>
6. </el-badge>
7.
8. <el-dropdown trigger="click">
9.
10. 点我查看<i class="el-icon-caret-bottom el-icon--right"></i>
11.
12. <el-dropdown-menu slot="dropdown">
13. <el-dropdown-item class="clearfix">
14. 评论
15. <el-badge class="mark" :value="12" />
16. </el-dropdown-item>
17. <el-dropdown-item class="clearfix">
18. 回复
19. <el-badge class="mark" :value="3" />
20. </el-dropdown-item>
```

```

21. </el-dropdown-menu>
22. </el-dropdown>
23.
24. <style>
25. .item {
26. margin-top: 10px;
27. margin-right: 40px;
28. }
29. </style>

```

## 最大值

可自定义最大值。



由 `max` 属性定义，它接受一个 `Number`，需要注意的是，只有当 `value` 为 `Number` 时，它才会生效。

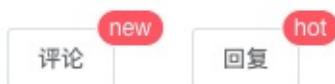
```

1. <el-badge :value="200" :max="99" class="item">
2. <el-button size="small">评论</el-button>
3. </el-badge>
4. <el-badge :value="100" :max="10" class="item">
5. <el-button size="small">回复</el-button>
6. </el-badge>
7.
8. <style>
9. .item {
10. margin-top: 10px;
11. margin-right: 40px;
12. }
13. </style>

```

## 自定义内容

可以显示数字以外的文本内容。



定义 `value` 为 `String` 类型时可以用于显示自定义文本。

```

1. <el-badge value="new" class="item">
2. <el-button size="small">评论</el-button>
3. </el-badge>
4. <el-badge value="hot" class="item">
5. <el-button size="small">回复</el-button>
6. </el-badge>
7.
8. <style>
9. .item {
10. margin-top: 10px;
11. margin-right: 40px;
12. }
13. </style>
```

## 小红点

以红点的形式标注需要关注的内容。



除了数字外，设置 `is-dot` 属性，它接受一个 `Boolean`。

```

1. <el-badge is-dot class="item">数据查询</el-badge>
2. <el-badge is-dot class="item">
3. <el-button class="share-button" icon="el-icon-share" type="primary"></el-
4. button>
5. </el-badge>
6.
7. <style>
8. .item {
9. margin-top: 10px;
10. margin-right: 40px;
11. }
```

[显示代码](#)

## Attributes

| 参数     | 说明                                         | 类型             | 可选值 | 默认值   |
|--------|--------------------------------------------|----------------|-----|-------|
| value  | 显示值                                        | string, number | —   | —     |
| max    | 最大值，超过最大值会显示 '{max}+'，要求 value 是 Number 类型 | number         | —   | —     |
| is-dot | 小圆点                                        | boolean        | —   | false |
| hidden | 隐藏 badge                                   | boolean        | —   | false |

原文: <http://element-cn.eleme.io/#/zh-CN/component/badge>

# Notice 通知组件

- [Alert 警告](#)
- [Loading 加载](#)
- [Message 消息提示](#)
- [MessageBox 弹框](#)
- [Notification 通知](#)

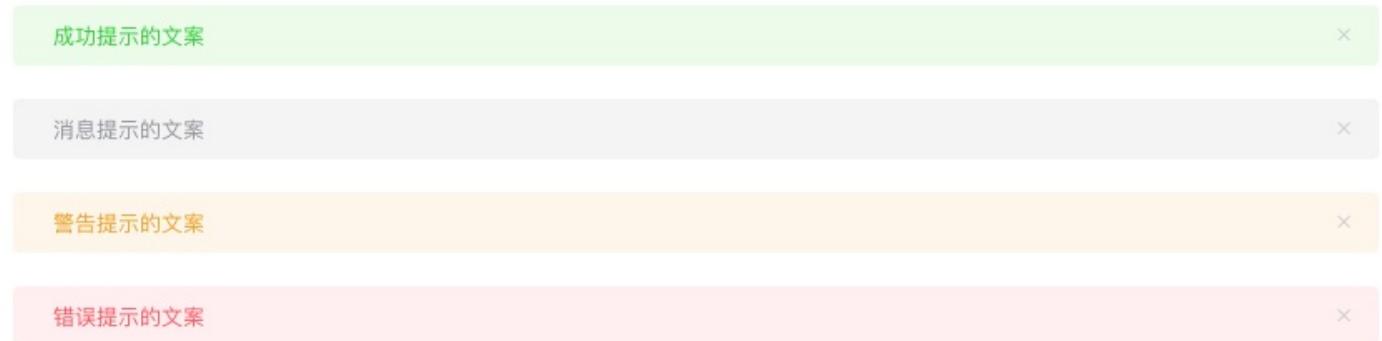
# Alert 警告

## Alert 警告

用于页面中展示重要的提示信息。

### 基本用法

页面中的非浮层元素，不会自动消失。



Alert 组件提供四种主题，由 `type` 属性指定，默认值为 `info`。

```
1. <template>
2. <el-alert
3. title="成功提示的文案"
4. type="success">
5. </el-alert>
6. <el-alert
7. title="消息提示的文案"
8. type="info">
9. </el-alert>
10. <el-alert
11. title="警告提示的文案"
12. type="warning">
13. </el-alert>
14. <el-alert
15. title="错误提示的文案"
16. type="error">
17. </el-alert>
18. </template>
```

## 自定义关闭按钮

自定义关闭按钮为文字或其他符号。

不可关闭的 alert

自定义 close-text

知道了

设置了回调的 alert

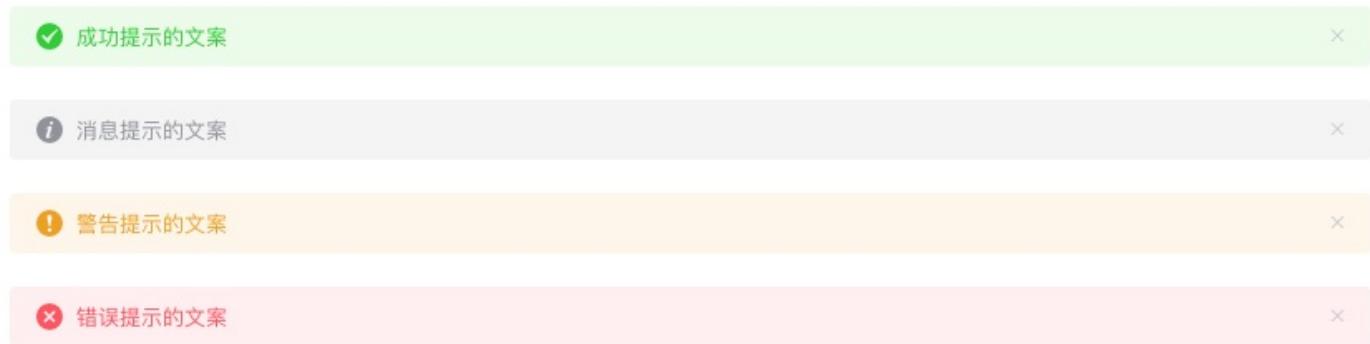
X

在 Alert 组件中，你可以设置是否可关闭，关闭按钮的文本以及关闭时的回调函数。`closable` 属性决定是否可关闭，接受 `boolean`，默认为 `true`。你可以设置 `close-text` 属性来代替右侧的关闭图标，注意：`close-text` 必须为文本。设置 `close` 事件来设置关闭时的回调。

```
1. <template>
2. <el-alert
3. title="不可关闭的 alert"
4. type="success"
5. :closable="false">
6. </el-alert>
7. <el-alert
8. title="自定义 close-text"
9. type="info"
10. close-text="知道了">
11. </el-alert>
12. <el-alert
13. title="设置了回调的 alert"
14. type="warning"
15. @close="hello">
16. </el-alert>
17. </template>
18.
19. <script>
20. export default {
21. methods: {
22. hello() {
23. alert('Hello World!');
24. }
25. }
26. }
27. </script>
```

## 带有 icon

表示某种状态时提升可读性。



通过设置 `show-icon` 属性来显示 Alert 的 icon，这能更有效地向用户展示你的显示意图。

```
1. <template>
2. <el-alert
3. title="成功提示的文案"
4. type="success"
5. show-icon>
6. </el-alert>
7. <el-alert
8. title="消息提示的文案"
9. type="info"
10. show-icon>
11. </el-alert>
12. <el-alert
13. title="警告提示的文案"
14. type="warning"
15. show-icon>
16. </el-alert>
17. <el-alert
18. title="错误提示的文案"
19. type="error"
20. show-icon>
21. </el-alert>
22. </template>
```

## 文字居中

使用 `center` 属性让文字水平居中。



The screenshot displays four distinct alert components, each with a unique background color and icon:

- 成功提示的文案** (Success): Green background, checkmark icon.
- 消息提示的文案** (Info): Light gray background, information icon.
- 警告提示的文案** (Warning): Orange background, exclamation mark icon.
- 错误提示的文案** (Error): Red background, error icon.

Below the alerts, the corresponding Vue.js template code is shown:

```
1. <template>
2. <el-alert
3. title="成功提示的文案"
4. type="success"
5. center
6. show-icon>
7. </el-alert>
8. <el-alert
9. title="消息提示的文案"
10. type="info"
11. center
12. show-icon>
13. </el-alert>
14. <el-alert
15. title="警告提示的文案"
16. type="warning"
17. center
18. show-icon>
19. </el-alert>
20. <el-alert
21. title="错误提示的文案"
22. type="error"
23. center
24. show-icon>
25. </el-alert>
26. </template>
```

## 带有辅助性文字介绍

包含标题和内容，解释更详细的警告。

## 带辅助性文字介绍

这是一句绕口令：黑灰化肥会挥发发灰黑化肥挥发；灰黑化肥会挥发发黑灰化肥挥发。黑灰化肥会挥发发灰黑化肥黑灰挥发化为灰.....

除了必填的 `title` 属性外，你可以设置 `description` 属性来帮助你更好地介绍，我们称之为辅助性文字。辅助性文字只能存放单行文本，会自动换行显示。

```

1. <template>
2. <el-alert
3. title="带辅助性文字介绍"
4. type="success"
5. description="这是一句绕口令：黑灰化肥会挥发发灰黑化肥挥发；灰黑化肥会挥发发黑灰化肥发
 挥。黑灰化肥会挥发发灰黑化肥黑灰挥发化为灰.....">
6. </el-alert>
7. </template>
```

## 带有 icon 和辅助性文字介绍



## 成功提示的文案

文字说明文字说明文字说明文字说明文字说明文字说明



## 消息提示的文案

文字说明文字说明文字说明文字说明文字说明文字说明



## 警告提示的文案

文字说明文字说明文字说明文字说明文字说明文字说明



## 错误提示的文案

文字说明文字说明文字说明文字说明文字说明文字说明

最后，这是一个同时具有 icon 和辅助性文字的样例。

```

1. <template>
2. <el-alert
3. title="成功提示的文案"
4. type="success"
5. description="文字说明文字说明文字说明文字说明文字说明文字说明"
6. show-icon>
7. </el-alert>
8. <el-alert
9. title="消息提示的文案"
```

```

10. type="info"
11. description="文字说明文字说明文字说明文字说明文字说明"
12. show-icon>
13. </el-alert>
14. <el-alert
15. title="警告提示的文案"
16. type="warning"
17. description="文字说明文字说明文字说明文字说明文字说明"
18. show-icon>
19. </el-alert>
20. <el-alert
21. title="错误提示的文案"
22. type="error"
23. description="文字说明文字说明文字说明文字说明文字说明"
24. show-icon>
25. </el-alert>
26. </template>

```

## Attributes

| 参数           | 说明                   | 类型      | 可选值                        | 默认值   |
|--------------|----------------------|---------|----------------------------|-------|
| <b>title</b> | 标题, 必选参数             | string  | —                          | —     |
| type         | 主题                   | string  | success/warning/info/error | info  |
| description  | 辅助性文字。也可通过默认 slot 传入 | string  | —                          | —     |
| closable     | 是否可关闭                | boolean | —                          | true  |
| center       | 文字是否居中               | boolean | —                          | true  |
| close-text   | 关闭按钮自定义文本            | string  | —                          | —     |
| show-icon    | 是否显示图标               | boolean | —                          | false |

## Events

| 事件名称  | 说明            | 回调参数 |
|-------|---------------|------|
| close | 关闭alert时触发的事件 | —    |

原文: <http://element-cn.eleme.io/#/zh-CN/component/alert>

# Loading 加载

## Loading 加载

加载数据时显示动效。

### 区域加载

在表格等容器中加载数据时显示。

|        | 姓名  | 地址                                                                                                  |
|--------|-----|-----------------------------------------------------------------------------------------------------|
| -05-03 | 王小虎 | 上海市普陀区金沙江路 1518 弄                                                                                   |
| -05-02 | 王小虎 |  上海市普陀区金沙江路 1518 弄 |
| -05-04 | 王小虎 | 上海市普陀区金沙江路 1518 弄                                                                                   |

Element 提供了两种调用 Loading 的方法：指令和服务。对于自定义指令 `v-loading`，只需要绑定 `Boolean` 即可。默认状况下，Loading 遮罩会插入到绑定元素的子节点，通过添加 `body` 修饰符，可以使遮罩插入至 DOM 中的 body 上。

```
1. <template>
2. <el-table
3. v-loading="loading"
4. :data="tableData"
5. style="width: 100%">
6. <el-table-column
7. prop="date"
8. label="日期"
9. width="180">
10. </el-table-column>
11. <el-table-column
12. prop="name"
13. label="姓名"
14. width="180">
15. </el-table-column>
16. <el-table-column
17. prop="address"
```

```
18. label="地址">
19. </el-table-column>
20. </el-table>
21. </template>
22.
23. <style>
24. body {
25. margin: 0;
26. }
27. </style>
28.
29. <script>
30. export default {
31. data() {
32. return {
33. tableData: [
34. {
35. date: '2016-05-03',
36. name: '王小虎',
37. address: '上海市普陀区金沙江路 1518 弄'
38. },
39. {
40. date: '2016-05-02',
41. name: '王小虎',
42. address: '上海市普陀区金沙江路 1518 弄'
43. },
44. {
45. date: '2016-05-04',
46. name: '王小虎',
47. address: '上海市普陀区金沙江路 1518 弄'
48. }
49.],
50. loading: true
51. };
52. }
53. };
54. </script>
```

显示代码

## 自定义

可自定义加载文案、图标和背景色。

| 日期         | 姓名  | 地址                         |
|------------|-----|----------------------------|
| 2016-05-03 | 王小虎 | 上海市普陀区金沙江路 1518 弄<br>✿     |
| 2016-05-02 | 王小虎 | 拼命加载中<br>上海市普陀区金沙江路 1518 弄 |
| 2016-05-04 | 王小虎 | 上海市普陀区金沙江路 1518 弄          |

在绑定了 `v-loading` 指令的元素上添加 `element-loading-text` 属性，其值会被渲染为加载文案，并显示在加载图标的下方。类似地，`element-loading-spinner` 和 `element-loading-background` 属性分别用来设定图标类名和背景色值。

```

1. <template>
2. <el-table
3. v-loading="loading2"
4. element-loading-text="拼命加载中"
5. element-loading-spinner="el-icon-loading"
6. element-loading-background="rgba(0, 0, 0, 0.8)"
7. :data="tableData"
8. style="width: 100%">
9. <el-table-column
10. prop="date"
11. label="日期"
12. width="180">
13. </el-table-column>
14. <el-table-column
15. prop="name"
16. label="姓名"
17. width="180">
18. </el-table-column>
19. <el-table-column
20. prop="address"
21. label="地址">
22. </el-table-column>
23. </el-table>
24. </template>
25.
26. <script>
27. export default {
28. data() {
29. return {
30. tableData: [{


```

```

31. date: '2016-05-03',
32. name: '王小虎',
33. address: '上海市普陀区金沙江路 1518 弄'
34. },
35. {
36. date: '2016-05-02',
37. name: '王小虎',
38. address: '上海市普陀区金沙江路 1518 弄'
39. },
40. {
41. date: '2016-05-04',
42. name: '王小虎',
43. address: '上海市普陀区金沙江路 1518 弄'
44. }],
45. loading2: true
46.);
47. </script>

```

## 整页加载

页面数据加载时显示。

指令方式

服务方式

当使用指令方式时，全屏遮罩需要添加 `fullscreen` 修饰符（遮罩会插入至 `body` 上），此时若需要锁定屏幕的滚动，可以使用 `lock` 修饰符；当使用服务方式时，遮罩默认即为全屏，无需额外设置。

```

1. <template>
2. <el-button
3. type="primary"
4. @click="openFullScreen"
5. v-loading.fullscreen.lock="fullscreenLoading">
6. 指令方式
7. </el-button>
8. <el-button
9. type="primary"
10. @click="openFullScreen2">
11. 服务方式
12. </el-button>

```

```
13. </template>
14.
15. <script>
16. export default {
17. data() {
18. return {
19. fullscreenLoading: false
20. }
21. },
22. methods: {
23. openFullScreen() {
24. this.fullscreenLoading = true;
25. setTimeout(() => {
26. this.fullscreenLoading = false;
27. }, 2000);
28. },
29. openFullScreen2() {
30. const loading = this.$loading({
31. lock: true,
32. text: 'Loading',
33. spinner: 'el-icon-loading',
34. background: 'rgba(0, 0, 0, 0.7)'
35. });
36. setTimeout(() => {
37. loading.close();
38. }, 2000);
39. }
40. }
41. }
42. </script>
```

## 服务

Loading 还可以以服务的方式调用。引入 Loading 服务：

```
1. import { Loading } from 'element-ui';
```

在需要调用时：

```
1. Loading.service(options);
```

其中 `options` 参数为 Loading 的配置项，具体见下表。`LoadingService` 会返回一个 Loading 实例，可通过调用该实例的 `close` 方法来关闭它：

```
1. let loadingInstance = Loading.service(options);
2. this.$nextTick(() => { // 以服务的方式调用的 Loading 需要异步关闭
3. loadingInstance.close();
4. });
```

需要注意的是，以服务的方式调用的全屏 Loading 是单例的：若在前一个全屏 Loading 关闭前再次调用全屏 Loading，并不会创建一个新的 Loading 实例，而是返回现有全屏 Loading 的实例：

```
1. let loadingInstance1 = Loading.service({ fullscreen: true });
2. let loadingInstance2 = Loading.service({ fullscreen: true });
3. console.log(loadingInstance1 === loadingInstance2); // true
```

此时调用它们中任意一个的 `close` 方法都能关闭这个全屏 Loading。

如果完整引入了 Element，那么 `Vue.prototype` 上会有一个全局方法 `$loading`，它的调用方式为：`this.$loading(options)`，同样会返回一个 Loading 实例。

## Options

| 参数          | 说明                                                                                                        | 类型            | 可选值 | 默认值           |
|-------------|-----------------------------------------------------------------------------------------------------------|---------------|-----|---------------|
| target      | Loading 需要覆盖的 DOM 节点。可传入一个 DOM 对象或字符串；若传入字符串，则会将其作为参数传入 <code>document.querySelector</code> 以获取到对应 DOM 节点 | object/string | -   | document.body |
| body        | 同 <code>v-loading</code> 指令中的 <code>body</code> 修饰符                                                       | boolean       | -   | false         |
| fullscreen  | 同 <code>v-loading</code> 指令中的 <code>fullscreen</code> 修饰符                                                 | boolean       | -   | true          |
| lock        | 同 <code>v-loading</code> 指令中的 <code>lock</code> 修饰符                                                       | boolean       | -   | false         |
| text        | 显示在加载图标下方的加载文案                                                                                            | string        | -   | -             |
| spinner     | 自定义加载图标类名                                                                                                 | string        | -   | -             |
| background  | 遮罩背景色                                                                                                     | string        | -   | -             |
| customClass | Loading 的自定义类名                                                                                            | string        | -   | -             |

原文：<http://element-cn.eleme.io/#/zh-CN/component/loading>

# Message 消息提示

## Message 消息提示

常用于主动操作后的反馈提示。与 `Notification` 的区别是后者更多用于系统级通知的被动提醒。

### 基础用法

从顶部出现，3 秒后自动消失。

 内容可以是 `VNode`

[打开消息提示](#)

[VNode](#)

Message 在配置上与 `Notification` 非常类似，所以部分 `options` 在此不做详尽解释，文末有 `options` 列表，可以结合 `Notification` 的文档理解它们。Element 注册了一个 `$message` 方法用于调用，Message 可以接收一个字符串或一个 `VNode` 作为参数，它会被显示为正文内容。

```
1. <template>
2. <el-button :plain="true" @click="open">打开消息提示</el-button>
3. <el-button :plain="true" @click="openVn">VNode</el-button>
4. </template>
5.
6. <script>
7. export default {
8. methods: {
9. open() {
10. this.$message('这是一条消息提示');
11. },
12.
13. openVn() {
14. const h = this.$createElement;
15. this.$message({
16. message: h('p', null, [
17. h('span', null, '内容可以是 '),
18. h('i', { style: 'color: teal' }, 'VNode')
19.])
20. });
21. }
22. }
23. }
24.
```

```

21. }
22. }
23. }
24. </script>

```

## 不同状态

用来显示「成功、警告、消息、错误」类的操作反馈。



恭喜你，这是一条成功消息

成功

警告

消息

错误

当需要自定义更多属性时，Message 也可以接收一个对象为参数。比如，设置 `type` 字段可以定义不同的状态，默认为 `info`。此时正文内容以 `message` 的值传入。同时，我们也为 Message 的各种 `type` 注册了方法，可以在不传入 `type` 字段的情况下像 `open4` 那样直接调用。

```

1. <template>
2. <el-button :plain="true" @click="open2">成功</el-button>
3. <el-button :plain="true" @click="open3">警告</el-button>
4. <el-button :plain="true" @click="open">消息</el-button>
5. <el-button :plain="true" @click="open4">错误</el-button>
6. </template>
7.
8. <script>
9. export default {
10. methods: {
11. open() {
12. this.$message('这是一条消息提示');
13. },
14. open2() {
15. this.$message({
16. message: '恭喜你，这是一条成功消息',
17. type: 'success'
18. });
19. },
20.
21. open3() {
22. this.$message({
23. message: '警告哦，这是一条警告消息',

```

```

24. type: 'warning'
25. });
26. },
27.
28. open4() {
29. this.$message.error('错了哦，这是一条错误消息');
30. }
31. }
32. }
33. </script>

```

## 可关闭

可以添加关闭按钮。



默认的 Message 是不可以被人工关闭的，如果需要可手动关闭的 Message，可以用 `showClose` 字段。此外，和 Notification 一样，Message 拥有可控的 `duration`，设置 `0` 为不会被自动关闭，默认为 3000 毫秒。

```

1. <template>
2. <el-button :plain="true" @click="open5">消息</el-button>
3. <el-button :plain="true" @click="open6">成功</el-button>
4. <el-button :plain="true" @click="open7">警告</el-button>
5. <el-button :plain="true" @click="open8">错误</el-button>
6. </template>
7.
8. <script>
9. export default {
10. methods: {
11. open5() {
12. this.$message({
13. showClose: true,
14. message: '这是一条消息提示'
15. });
16. },
17. }
18. }

```

```

18. open6() {
19. this.$message({
20. showClose: true,
21. message: '恭喜你，这是一条成功消息',
22. type: 'success'
23. });
24. },
25.
26. open7() {
27. this.$message({
28. showClose: true,
29. message: '警告哦，这是一条警告消息',
30. type: 'warning'
31. });
32. },
33.
34. open8() {
35. this.$message({
36. showClose: true,
37. message: '错了哦，这是一条错误消息',
38. type: 'error'
39. });
40. }
41. }
42. }
43. </script>

```

## 文字居中

使用 `center` 属性让文字水平居中。

 居中的文字

文字居中

```

1. <template>
2. <el-button :plain="true" @click="openCenter">文字居中</el-button>
3. </template>
4.

```

```

5. <script>
6. export default {
7. methods: {
8. openCenter() {
9. this.$message({
10. message: '居中的文字',
11. center: true
12. });
13. }
14. }
15. }
16. </script>

```

## 使用 HTML 片段

`message` 属性支持传入 HTML 片段

 这是 `HTML` 片段

使用 HTML 片段

将 `dangerouslyUseHTMLString` 属性设置为 `true`, `message` 就会被当作 HTML 片段处理。

```

1. <template>
2. <el-button :plain="true" @click="openHTML">使用 HTML 片段</el-button>
3. </template>
4.
5. <script>
6. export default {
7. methods: {
8. openHTML() {
9. this.$message({
10. dangerouslyUseHTMLString: true,
11. message: '这是 <i>HTML</i> 片段'
12. });
13. }
14. }
15. }
16. </script>

```

`message` 属性虽然支持传入 HTML 片段，但是在网站上动态渲染任意 HTML 是非常危险的，因为容易导致 XSS 攻击。因此在 `dangerouslyUseHTMLString` 打开的情况下，请确保 `message` 的内容是可信的，永远不要将用户提交的内容赋值给 `message` 属性。

## 全局方法

Element 为 `Vue.prototype` 添加了全局方法 `$message`。因此在 `vue instance` 中可以采用本页面中的方式调用 `Message`。

### 单独引用

单独引入 `Message`：

```
1. import { Message } from 'element-ui';
```

此时调用方法为 `Message(options)`。我们也为每个 `type` 定义了各自的方法，如 `Message.success(options)`。并且可以调用 `Message.closeAll()` 手动关闭所有实例。

### Options

| 参数                                    | 说明                                      | 类型                          | 可选值                                     | 默认值                |
|---------------------------------------|-----------------------------------------|-----------------------------|-----------------------------------------|--------------------|
| <code>message</code>                  | 消息文字                                    | <code>string / VNode</code> | -                                       | -                  |
| <code>type</code>                     | 主题                                      | <code>string</code>         | <code>success/warning/info/error</code> | <code>info</code>  |
| <code>iconClass</code>                | 自定义图标的类名，会覆盖 <code>type</code>          | <code>string</code>         | -                                       | -                  |
| <code>dangerouslyUseHTMLString</code> | 是否将 <code>message</code> 属性作为 HTML 片段处理 | <code>boolean</code>        | -                                       | <code>false</code> |
| <code>customClass</code>              | 自定义类名                                   | <code>string</code>         | -                                       | -                  |
| <code>duration</code>                 | 显示时间，毫秒。设为 0 则不会自动关闭                    | <code>number</code>         | -                                       | <code>3000</code>  |
| <code>showClose</code>                | 是否显示关闭按钮                                | <code>boolean</code>        | -                                       | <code>false</code> |
| <code>center</code>                   | 文字是否居中                                  | <code>boolean</code>        | -                                       | <code>false</code> |
|                                       | 关闭时的回调函                                 |                             |                                         |                    |

|         |                       |          |   |   |
|---------|-----------------------|----------|---|---|
| onClose | 参数，参数为被关闭的 message 实例 | function | - | - |
|---------|-----------------------|----------|---|---|

## 方法

调用 `Message` 或 `this.$message` 会返回当前 `Message` 的实例。如果需要手动关闭实例，可以调用它的 `close` 方法。

| 方法名                | 说明                         |
|--------------------|----------------------------|
| <code>close</code> | 关闭当前的 <code>Message</code> |

原文: <http://element-cn.eleme.io/#/zh-CN/component/message>

# MessageBox 弹框

## MessageBox 弹框

模拟系统的消息提示框而实现的一套模态对话框组件，用于消息提示、确认消息和提交内容。

从场景上说，MessageBox 的作用是美化系统自带的 `alert`、`confirm` 和 `prompt`，因此适合展示较为简单的内容。如果需要弹出较为复杂的内容，请使用 Dialog。

### 消息提示

当用户进行操作时会被触发，该对话框中断用户操作，直到用户确认知晓后才可关闭。



调用 `$alert` 方法即可打开消息提示，它模拟了系统的 `alert`，无法通过按下 ESC 或点击框外关闭。此例中接收了两个参数，`message` 和 `title`。值得一提的是，窗口被关闭后，它默认会返回一个 `Promise` 对象便于进行后续操作的处理。若不确定浏览器是否支持 `Promise`，可自行引入第三方 polyfill 或像本例一样使用回调进行后续处理。

```

1. <template>
2. <el-button type="text" @click="open">点击打开 Message Box</el-button>
3. </template>
4.
5. <script>
6. export default {
7. methods: {
8. open() {
9. this.$alert('这是一段内容', '标题名称', {
10. confirmButtonText: '确定',
11. callback: action => {
12. this.$message({
13. type: 'info',

```

```

14. message: `action: ${ action }`

15.);

16. }

17.);

18. }

19. }

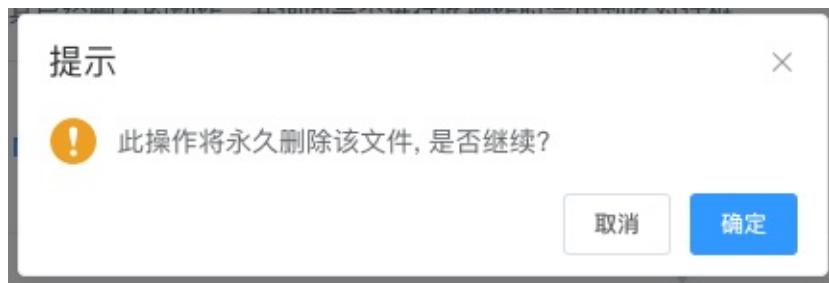
20. }

21. </script>

```

## 确认消息

提示用户确认其已经触发的动作，并询问是否进行此操作时会用到此对话框。



调用 `$confirm` 方法即可打开消息提示，它模拟了系统的 `confirm`。Message Box 组件也拥有极高的定制性，我们可以传入 `options` 作为第三个参数，它是一个字面量对象。`type` 字段表明消息类型，可以为 `success`，`error`，`info` 和 `warning`，无效的设置将会被忽略。注意，第二个参数 `title` 必须定义为 `String` 类型，如果是 `Object`，会被理解为 `options`。在这里我们用了 Promise 来处理后续响应。

```

1. <template>

2. <el-button type="text" @click="open2">点击打开 Message Box</el-button>

3. </template>

4.

5. <script>

6. export default {

7. methods: {

8. open2() {

9. this.$confirm('此操作将永久删除该文件, 是否继续?', '提示', {

10. confirmButtonText: '确定',

11. cancelButtonText: '取消',

12. type: 'warning'

13. }).then(() => {

14. this.$message({

15. type: 'success',

16. message: '删除成功!'

```

```

17. });
18. }).catch(() => {
19. this.$message({
20. type: 'info',
21. message: '已取消删除'
22. });
23. });
24. }
25. }
26. }
27. </script>

```

## 提交内容

当用户进行操作时会被触发，中断用户操作，提示用户进行输入的对话框。



调用 `$prompt` 方法即可打开消息提示，它模拟了系统的 `prompt`。可以用 `inputPattern` 字段自己规定匹配模式，或者用 `inputValidator` 规定校验函数，可以返回 `Boolean` 或 `String`，返回 `false` 或字符串时均表示校验未通过，同时返回的字符串相当于定义了 `inputErrorMessage` 字段。此外，可以用 `inputPlaceholder` 字段来定义输入框的占位符。

```

1. <template>
2. <el-button type="text" @click="open3">点击打开 Message Box</el-button>
3. </template>
4.
5. <script>
6. export default {
7. methods: {
8. open3() {
9. this.$prompt('请输入邮箱', '提示', {
10. confirmButtonText: '确定',
11. cancelButtonText: '取消',

```

```

12. inputPattern: /[\w!#$%&!*+/?^_`{|}~-]+(?:\.[\w!#$%&!*+/?^_`{|}~-]+)*@(?:[\w](?:[\w-]*[\w])?\.)+[\w](?:[\w-]*[\w])?/,
13. inputErrorMessage: '邮箱格式不正确'
14. }).then(({ value }) => {
15. this.$message({
16. type: 'success',
17. message: '你的邮箱是: ' + value
18. });
19. }).catch(() => {
20. this.$message({
21. type: 'info',
22. message: '取消输入'
23. });
24. });
25. }
26. }
27. }
28. </script>

```

## 自定义

可自定义配置不同内容。



以上三个方法都是对 `$msgbox` 方法的再包装。本例直接调用 `$msgbox` 方法，使用了 `showCancelButton` 字段，用于显示取消按钮。另外可使用 `cancelButtonClass` 为其添加自定义样式，使用 `cancelButtonText` 来自定义按钮文本（Confirm 按钮也具有相同的字段，在文末的字段说明中有完整的字段列表）。此例还使用了 `beforeClose` 属性，它的值是一个方法，会在 MessageBox 的实例关闭前被调用，同时暂停实例的关闭。它有三个参数：`action`、实例本身和 `done` 方法。使用它能够在关闭前对实例进行一些操作，比如为确定按钮添加 `loading` 状态等；此时若需要关闭实例，可以调用 `done` 方法（若在 `beforeClose` 中没有调用 `done`，则实例不会关闭）。

```

1. <template>
2. <el-button type="text" @click="open4">点击打开 Message Box</el-button>
3. </template>

```

```

4.
5. <script>
6. export default {
7. methods: {
8. open4() {
9. const h = this.$createElement;
10. this.$msgbox({
11. title: '消息',
12. message: h('p', null, [
13. h('span', null, '内容可以是 '),
14. h('i', { style: 'color: teal' }, 'VNode')
15.]),
16. showCancelButton: true,
17. confirmButtonText: '确定',
18. cancelButtonText: '取消',
19. beforeClose: (action, instance, done) => {
20. if (action === 'confirm') {
21. instance.confirmButtonLoading = true;
22. instance.confirmButtonText = '执行中...';
23. setTimeout(() => {
24. done();
25. setTimeout(() => {
26. instance.confirmButtonLoading = false;
27. }, 300);
28. }, 3000);
29. } else {
30. done();
31. }
32. }
33. }).then(action => {
34. this.$message({
35. type: 'info',
36. message: `action: ${action}`
37. });
38. });
39. }
40. }
41. }
42. </script>

```

弹出层的内容可以是 `VNode`，所以我们可以把一些自定义组件传入其中。每次弹出层打开后，Vue 会对新老 `VNode` 节点进行比对，然后将根据比较结果进行最小单位地修改视图。这也许会造成弹出

层内容区域的组件没有重新渲染，例如 #8931。当这类问题出现时，解决方案是给 `VNode` 加上一个不相同的 key，参考[这里](#)。

## 使用 HTML 片段

`message` 属性支持传入 HTML 片段。



将 `dangerouslyUseHTMLString` 属性设置为 `true`，`message` 就会被当作 HTML 片段处理。

```

1. <template>
2. <el-button type="text" @click="open5">点击打开 Message Box</el-button>
3. </template>
4.
5. <script>
6. export default {
7. methods: {
8. open5() {
9. this.$alert('这是 <i>HTML</i> 片段', 'HTML 片段', {
10. dangerouslyUseHTMLString: true
11. });
12. }
13. }
14. }
15. </script>

```

`message` 属性虽然支持传入 HTML 片段，但是在网站上动态渲染任意 HTML 是非常危险的，因为容易导致 [XSS 攻击](#)。因此在 `dangerouslyUseHTMLString` 打开的情况下，请确保 `message` 的内容是可信的，永远不要将用户提交的内容赋值给 `message` 属性。

## 区分取消与关闭

有些场景下，点击取消按钮与点击关闭按钮有着不同的含义。



默认情况下，当用户触发取消（点击取消按钮）和触发关闭（点击关闭按钮或遮罩层、按下 ESC 键）时，Promise 的 reject 回调和 `callback` 回调的参数均为 'cancel'。如果将 `distinguishCancelAndClose` 属性设置为 true，则上述两种行为的参数分别为 'cancel' 和 'close'。

```

1. <template>
2. <el-button type="text" @click="open6">点击打开 Message Box</el-button>
3. </template>
4.
5. <script>
6. export default {
7. methods: {
8. open6() {
9. this.$confirm('检测到未保存的内容，是否在离开页面前保存修改？', '确认信息', {
10. distinguishCancelAndClose: true,
11. confirmButtonText: '保存',
12. cancelButtonText: '放弃修改'
13. })
14. .then(() => {
15. this.$message({
16. type: 'info',
17. message: '保存修改'
18. });
19. })
20. .catch(action => {
21. this.$message({
22. type: 'info',
23. message: action === 'cancel'
24. ? '放弃保存并离开页面'
25. : '停留在当前页面'
26. })
27. });
28. }
29. }
30. }

```

31. &lt;/script&gt;

## 居中布局

内容支持居中布局

将 `center` 设置为 `true` 即可开启居中布局

```

1. <template>
2. <el-button type="text" @click="open7">点击打开 Message Box</el-button>
3. </template>
4.
5. <script>
6. export default {
7. methods: {
8. open7() {
9. this.$confirm('此操作将永久删除该文件, 是否继续?', '提示', {
10. confirmButtonText: '确定',
11. cancelButtonText: '取消',
12. type: 'warning',
13. center: true
14. }).then(() => {
15. this.$message({
16. type: 'success',
17. message: '删除成功!'
18. });
19. }).catch(() => {
20. this.$message({
21. type: 'info',
22. message: '已取消删除'
23. });
24. });
25. }
26. }
27. }
28. </script>

```

## 全局方法

如果你完整引入了 Element，它会为 `Vue.prototype` 添加如下全局方法：`$msgbox`, `$alert`, `$confirm` 和 `$prompt`。因此在 `Vue instance` 中可以采用本页面中的方式调用 `MessageBox`。调用参数为：

- \$msgbox(options)
- \$alert(message, title, options) 或 \$alert(message, options)
- \$confirm(message, title, options) 或 \$confirm(message, options)
- \$prompt(message, title, options) 或 \$prompt(message, options)

## 单独引用

如果单独引入 `MessageBox` :

```
1. import { MessageBox } from 'element-ui';
```

那么对应于上述四个全局方法的调用方法依次为：`MessageBox`, `MessageBox.alert`, `MessageBox.confirm` 和 `MessageBox.prompt`, 调用参数与全局方法相同。

## Options

| 参数                                    | 说明                                                                   | 类型                                                                                                                                                                                                                 | 可选值                                           | 默认                 |
|---------------------------------------|----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|--------------------|
| <code>title</code>                    | <code>MessageBox</code> 标题                                           | <code>string</code>                                                                                                                                                                                                | —                                             | —                  |
| <code>message</code>                  | <code>MessageBox</code> 消息正文内容                                       | <code>string / VNode</code>                                                                                                                                                                                        | —                                             | —                  |
| <code>dangerouslyUseHTMLString</code> | 是否将 <code>message</code> 属性作为 HTML 片段处理                              | <code>boolean</code>                                                                                                                                                                                               | —                                             | <code>false</code> |
| <code>type</code>                     | 消息类型，用于显示图标                                                          | <code>string</code>                                                                                                                                                                                                | <code>success / info / warning / error</code> | —                  |
| <code>iconClass</code>                | 自定义图标的类名，会覆盖 <code>type</code>                                       | <code>string</code>                                                                                                                                                                                                | —                                             | —                  |
| <code>customClass</code>              | <code>MessageBox</code> 的自定义类名                                       | <code>string</code>                                                                                                                                                                                                | —                                             | —                  |
| <code>callback</code>                 | 若不使用 <code>Promise</code> , 可以使用此参数指定 <code>MessageBox</code> 关闭后的回调 | <code>function(action, instance)</code> , <code>action</code> 的值为 ' <code>confirm</code> ', ' <code>cancel</code> ' 或 ' <code>close</code> ', <code>instance</code> 为 <code>MessageBox</code> 实例, 可以通过它访问实例上的属性和方法 | —                                             | —                  |
| <code>showClose</code>                | <code>MessageBox</code> 是否显示右上角关闭按钮                                  | <code>boolean</code>                                                                                                                                                                                               | —                                             | <code>true</code>  |
| <code>beforeClose</code>              | <code>MessageBox</code> 关闭前的回调, 会暂停实例的关闭                             | <code>function(action, instance, done)</code> , <code>action</code> 的值为 ' <code>confirm</code> ', ' <code>cancel</code> ' 或 ' <code>close</code> '; <code>instance</code> 为 <code>MessageBox</code> 实例, 可          | —                                             | —                  |

|                           |                                                              |                                          |   |                                     |
|---------------------------|--------------------------------------------------------------|------------------------------------------|---|-------------------------------------|
|                           |                                                              | 以通过它访问实例上的属性和方法; done 用于关闭 MessageBox 实例 |   |                                     |
| distinguishCancelAndClose | 是否将取消(点击取消按钮)与关闭(点击关闭按钮或遮罩层、按下 ESC 键)进行区分                    | boolean                                  | - | false                               |
| lockScroll                | 是否在 MessageBox 出现时将 body 滚动锁定                                | boolean                                  | - | true                                |
| showCancelButton          | 是否显示取消按钮                                                     | boolean                                  | - | false<br>config 和 prompt 方式调用为 true |
| showConfirmButton         | 是否显示确定按钮                                                     | boolean                                  | - | true                                |
| cancelButtonText          | 取消按钮的文本内容                                                    | string                                   | - | 取消                                  |
| confirmButtonText         | 确定按钮的文本内容                                                    | string                                   | - | 确定                                  |
| cancelButtonClass         | 取消按钮的自定义类名                                                   | string                                   | - | -                                   |
| confirmButtonClass        | 确定按钮的自定义类名                                                   | string                                   | - | -                                   |
| closeOnClickModal         | 是否可通过点击遮罩关闭 MessageBox                                       | boolean                                  | - | true<br>alert 式调用 false             |
| closeOnPressEscape        | 是否可通过按下 ESC 键关闭 MessageBox                                   | boolean                                  | - | true<br>alert 式调用 false             |
| closeOnHashChange         | 是否在 hashchange 时关闭 MessageBox                                | boolean                                  | - | true                                |
| showInput                 | 是否显示输入框                                                      | boolean                                  | - | false<br>prompt 方式调用为 true          |
| inputPlaceholder          | 输入框的占位符                                                      | string                                   | - | -                                   |
| inputType                 | 输入框的类型                                                       | string                                   | - | text                                |
| inputValue                | 输入框的初始文本                                                     | string                                   | - | -                                   |
| inputPattern              | 输入框的校验表达式                                                    | regexp                                   | - | -                                   |
| inputValidator            | 输入框的校验函数。可以返回布尔值或字符串, 若返回一个字符串, 则返回结果会被赋值给 inputErrorMessage | function                                 | - | -                                   |
| inputErrorMessage         | 校验未通过时的提示文本                                                  | string                                   | - | 输入的不合法                              |
| center                    | 是否居中布局                                                       | boolean                                  | - | false                               |
| roundButton               | 是否使用圆角按钮                                                     | boolean                                  | - | false                               |

原文: <http://element-cn.eleme.io/#/zh-CN/component/message-box>



# Notification 通知

## Notification 通知

悬浮出现在页面角落，显示全局的通知提醒消息。

### 基本用法

适用性广泛的通知栏



Notification 组件提供通知功能，Element 注册了 `$notify` 方法，接收一个 `options` 字面量参数，在最简单的情况下，你可以设置 `title` 字段和 `message` 字段，用于设置通知的标题和正文。默认情况下，经过一段时间后 Notification 组件会自动关闭，但是通过设置 `duration`，可以控制关闭的时间间隔，特别的是，如果设置为 `0`，则不会自动关闭。注意：`duration` 接收一个 `Number`，单位为毫秒，默认为 `4500`。

```

1. <template>
2. <el-button
3. plain
4. @click="open"
5. 可自动关闭
6. </el-button>
7. <el-button
8. plain
9. @click="open2"
10. 不会自动关闭
11. </el-button>
12. </template>
13.

```

```
14. <script>
15. export default {
16. methods: {
17. open() {
18. const h = this.$createElement;
19.
20. this.$notify({
21. title: '标题名称',
22. message: h('i', { style: 'color: teal' }), '这是提示文案这是提示
 文案这是提示文案这是提示文案这是提示文案这是提示文案这是提示文案'
23. });
24. },
25.
26. open2() {
27. this.$notify({
28. title: '提示',
29. message: '这是一条不会自动关闭的消息',
30. duration: 0
31. });
32. }
33. }
34. }
35. </script>
```

## 带有倾向性

带有 icon，常用来显示「成功、警告、消息、错误」类的系统消息



Element 为 Notification 组件准备了四种通知类型：`success`，`warning`，`info`，`error`。通过 `type` 字段来设置，除此以外的值将被忽略。同时，我们也为 Notification 的各种 `type` 注册了方法，可以在不传入 `type` 字段的情况下像 `open5` 和 `open6` 那样直接调用。

```

1. <template>
2. <el-button
3. plain
4. @click="open3">
5. 成功
6. </el-button>
7. <el-button
8. plain
9. @click="open4">
10. 警告
11. </el-button>
12. <el-button
13. plain
14. @click="open5">
15. 消息
16. </el-button>
17. <el-button
18. plain
19. @click="open6">
20. 错误
21. </el-button>
22. </template>

```

```
23.
24. <script>
25. export default {
26. methods: {
27. open3() {
28. this.$notify({
29. title: '成功',
30. message: '这是一条成功的提示消息',
31. type: 'success'
32. });
33. },
34.
35. open4() {
36. this.$notify({
37. title: '警告',
38. message: '这是一条警告的提示消息',
39. type: 'warning'
40. });
41. },
42.
43. open5() {
44. this.$notify.info({
45. title: '消息',
46. message: '这是一条消息的提示消息'
47. });
48. },
49.
50. open6() {
51. this.$notify.error({
52. title: '错误',
53. message: '这是一条错误的提示消息'
54. });
55. }
56. }
57. }
58. </script>
```

显示代码

## 自定义弹出位置

可以让 Notification 从屏幕四角中的任意一角弹出

使用 `position` 属性定义 Notification 的弹出位置，支持四个选项：`top-right`、`top-left`、`bottom-right`、`bottom-left`，默认为 `top-right`。

```
1. <template>
2. <el-button
3. plain
4. @click="open7">
5. 右上角
6. </el-button>
7. <el-button
8. plain
9. @click="open8">
10. 右下角
11. </el-button>
12. <el-button
13. plain
14. @click="open9">
15. 左下角
16. </el-button>
17. <el-button
18. plain
19. @click="open10">
20. 左上角
21. </el-button>
22. </template>
23.
24. <script>
25. export default {
26. methods: {
27. open7() {
28. this.$notify({
29. title: '自定义位置',
30. message: '右上角弹出的消息'
31. });
32. },
33.
34. open8() {
35. this.$notify({
36. title: '自定义位置',
37. message: '右下角弹出的消息',
38. position: 'bottom-right'
39. });
40. }
41. }
42. </script>
```

```

40. },
41.
42. open9() {
43. this.$notify({
44. title: '自定义位置',
45. message: '左下角弹出的消息',
46. position: 'bottom-left'
47. });
48. },
49.
50. open10() {
51. this.$notify({
52. title: '自定义位置',
53. message: '左上角弹出的消息',
54. position: 'top-left'
55. });
56. }
57. }
58. }
59. </script>

```

## 带有偏移

让 Notification 偏移一些位置

Notification 提供设置偏移量的功能，通过设置 `offset` 字段，可以使弹出的消息距屏幕边缘偏移一段距离。注意在同一时刻，所有的 Notification 实例应当具有一个相同的偏移量。

```

1. <template>
2. <el-button
3. plain
4. @click="open11">
5. 偏移的消息
6. </el-button>
7. </template>
8.
9. <script>
10. export default {
11. methods: {
12. open11() {
13. this.$notify({
14. title: '偏移',

```

```

15. message: '这是一条带有偏移的提示消息',
16. offset: 100
17. });
18. }
19. }
20. }
21. </script>

```

## 使用 HTML 片段

`message` 属性支持传入 HTML 片段



将 `dangerouslyUseHTMLString` 属性设置为 `true`, `message` 就会被当作 HTML 片段处理。

```

1. <template>
2. <el-button
3. plain
4. @click="open12">
5. 使用 HTML 片段
6. </el-button>
7. </template>
8.
9. <script>
10. export default {
11. methods: {
12. open12() {
13. this.$notify({
14. title: 'HTML 片段',
15. dangerouslyUseHTMLString: true,
16. message: '这是 <i>HTML</i> 片段'
17. });
18. }
19. }
20. }
21. </script>

```

`message` 属性虽然支持传入 HTML 片段,但是在网站上动态渲染任意 HTML 是非常危险的,因为

容易导致 XSS 攻击。因此在 `dangerouslyUseHTMLString` 打开的情况下，请确保 `message` 的内容是可信的，永远不要将用户提交的内容赋值给 `message` 属性。

## 隐藏关闭按钮

可以不显示关闭按钮



将 `showClose` 属性设置为 `false` 即可隐藏关闭按钮。

```

1. <template>
2. <el-button
3. plain
4. @click="open13">
5. 隐藏关闭按钮
6. </el-button>
7. </template>
8.
9. <script>
10. export default {
11. methods: {
12. open13() {
13. this.$notify.success({
14. title: 'Info',
15. message: '这是一条没有关闭按钮的消息',
16. showClose: false
17. });
18. }
19. }
20. }
21. </script>

```

## 全局方法

Element 为 `Vue.prototype` 添加了全局方法 `$notify`。因此在 `vue instance` 中可以采用本页面中的方式调用 `Notification`。

## 单独引用

单独引入 Notification:

```
1. import { Notification } from 'element-ui';
```

此时调用方法为 `Notification(options)`。我们也为每个 type 定义了各自的方法，如 `Notification.success(options)`。并且可以调用 `Notification.closeAll()` 手动关闭所有实例。

## Options

| 参数                                    | 说明                                                            | 类型                            | 可选值                                                      |
|---------------------------------------|---------------------------------------------------------------|-------------------------------|----------------------------------------------------------|
| <code>title</code>                    | 标题                                                            | <code>string</code>           | —                                                        |
| <code>message</code>                  | 说明文字                                                          | <code>string/Vue.VNode</code> | —                                                        |
| <code>dangerouslyUseHTMLString</code> | 是否将 message 属性作为 HTML 片段处理                                    | <code>boolean</code>          | —                                                        |
| <code>type</code>                     | 主题样式，如果不在可选值内将被忽略                                             | <code>string</code>           | <code>success/warning/info/error</code>                  |
| <code>iconClass</code>                | 自定义图标的类名。若设置了 <code>type</code> 则 <code>iconClass</code> 会被覆盖 | <code>string</code>           | —                                                        |
| <code>customClass</code>              | 自定义类名                                                         | <code>string</code>           | —                                                        |
| <code>duration</code>                 | 显示时间，毫秒。设为 0 则不会自动关闭                                          | <code>number</code>           | —                                                        |
| <code>position</code>                 | 自定义弹出位置                                                       | <code>string</code>           | <code>top-right/top-left/bottom-right/bottom-left</code> |
| <code>showClose</code>                | 是否显示关闭按钮                                                      | <code>boolean</code>          | —                                                        |
| <code>onClose</code>                  | 关闭时的回调函数                                                      | <code>function</code>         | —                                                        |
| <code>onClick</code>                  | 点击 Notification 时的回调函数                                        | <code>function</code>         | —                                                        |
| <code>offset</code>                   | 偏移的距离，在同一时刻，所有的 Notification 实例应当具有一个相同的偏移量                   | <code>number</code>           | —                                                        |

## 方法

调用 `Notification` 或 `this.$notify` 会返回当前 Notification 的实例。如果需要手动

关闭实例，可以调用它的 `close` 方法。

| 方法名                | 说明                 |
|--------------------|--------------------|
| <code>close</code> | 关闭当前的 Notification |

原文: <http://element-cn.eleme.io/#/zh-CN/component/notification>



# Navigation 导航组件

- [NavMenu 导航菜单](#)
- [Tabs 标签页](#)
- [Breadcrumb 面包屑](#)
- [Dropdown 下拉菜单](#)
- [Steps 步骤条](#)

# NavMenu 导航菜单

## NavMenu 导航菜单

为网站提供导航功能的菜单。

### 顶栏

适用广泛的基础用法。



导航菜单默认为垂直模式，通过 `mode` 属性可以使导航菜单变更为水平模式。另外，在菜单中通过 `submenu` 组件可以生成二级菜单。Menu 还提供了 `background-color`、`text-color` 和 `active-text-color`，分别用于设置菜单的背景色、菜单的文字颜色和当前激活菜单的文字颜色。

```
1. <el-menu :default-active="activeIndex" class="el-menu-demo" mode="horizontal">
 @select="handleSelect"
2. <el-menu-item index="1">处理中心</el-menu-item>
3. <el-submenu index="2">
4. <template slot="title">我的工作台</template>
5. <el-menu-item index="2-1">选项1</el-menu-item>
6. <el-menu-item index="2-2">选项2</el-menu-item>
7. <el-menu-item index="2-3">选项3</el-menu-item>
8. <el-submenu index="2-4">
9. <template slot="title">选项4</template>
10. <el-menu-item index="2-4-1">选项1</el-menu-item>
11. <el-menu-item index="2-4-2">选项2</el-menu-item>
12. <el-menu-item index="2-4-3">选项3</el-menu-item>
13. </el-submenu>
14. </el-submenu>
15. <el-menu-item index="3" disabled>消息中心</el-menu-item>
```

```
16. <el-menu-item index="4">订单管
 理</el-menu-item>
17. </el-menu>
18. <div class="line"></div>
19. <el-menu
20. :default-active="activeIndex2"
21. class="el-menu-demo"
22. mode="horizontal"
23. @select="handleSelect"
24. background-color="#545c64"
25. text-color="#fff"
26. active-text-color="#ffd04b">
27. <el-menu-item index="1">处理中心</el-menu-item>
28. <el-submenu index="2">
29. <template slot="title">我的工作台</template>
30. <el-menu-item index="2-1">选项1</el-menu-item>
31. <el-menu-item index="2-2">选项2</el-menu-item>
32. <el-menu-item index="2-3">选项3</el-menu-item>
33. <el-submenu index="2-4">
34. <template slot="title">选项4</template>
35. <el-menu-item index="2-4-1">选项1</el-menu-item>
36. <el-menu-item index="2-4-2">选项2</el-menu-item>
37. <el-menu-item index="2-4-3">选项3</el-menu-item>
38. </el-submenu>
39. </el-submenu>
40. <el-menu-item index="3" disabled>消息中心</el-menu-item>
41. <el-menu-item index="4">订单管
 理</el-menu-item>
42. </el-menu>
43.
44. <script>
45. export default {
46. data() {
47. return {
48. activeIndex: '1',
49. activeIndex2: '1'
50. };
51. },
52. methods: {
53. handleSelect(key, keyPath) {
54. console.log(key, keyPath);
55. }
56. }
57. }
58.
```

```

56. }
57. }
58. </script>

```

## 侧栏

垂直菜单，可内嵌子菜单。



通过 `el-menu-item-group` 组件可以实现菜单进行分组，分组名可以通过 `title` 属性直接设定，也可以通过具名 slot 来设定。

```

1. <el-row class="tac">
2. <el-col :span="12">
3. <h5>默认颜色</h5>
4. <el-menu
5. default-active="2"
6. class="el-menu-vertical-demo"
7. @open="handleOpen"
8. @close="handleClose">
9. <el-submenu index="1">
10. <template slot="title">

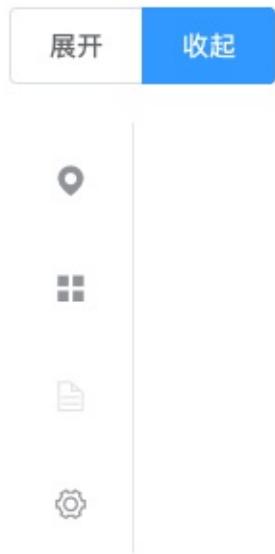
```

```
11. <i class="el-icon-location"></i>
12. 导航一
13. </template>
14. <el-menu-item-group>
15. <template slot="title">分组一</template>
16. <el-menu-item index="1-1">选项1</el-menu-item>
17. <el-menu-item index="1-2">选项2</el-menu-item>
18. </el-menu-item-group>
19. <el-menu-item-group title="分组2">
20. <el-menu-item index="1-3">选项3</el-menu-item>
21. </el-menu-item-group>
22. <el-submenu index="1-4">
23. <template slot="title">选项4</template>
24. <el-menu-item index="1-4-1">选项1</el-menu-item>
25. </el-submenu>
26. </el-submenu>
27. <el-menu-item index="2">
28. <i class="el-icon-menu"></i>
29. 导航二
30. </el-menu-item>
31. <el-menu-item index="3" disabled>
32. <i class="el-icon-document"></i>
33. 导航三
34. </el-menu-item>
35. <el-menu-item index="4">
36. <i class="el-icon-setting"></i>
37. 导航四
38. </el-menu-item>
39. </el-menu>
40. </el-col>
41. <el-col :span="12">
42. <h5>自定义颜色</h5>
43. <el-menu
44. default-active="2"
45. class="el-menu-vertical-demo"
46. @open="handleOpen"
47. @close="handleClose"
48. background-color="#545c64"
49. text-color="#fff"
50. active-text-color="#ffd04b">
51. <el-submenu index="1">
52. <template slot="title">
```

```
53. <i class="el-icon-location"></i>
54. 导航一
55. </template>
56. <el-menu-item-group>
57. <template slot="title">分组一</template>
58. <el-menu-item index="1-1">选项1</el-menu-item>
59. <el-menu-item index="1-2">选项2</el-menu-item>
60. </el-menu-item-group>
61. <el-menu-item-group title="分组2">
62. <el-menu-item index="1-3">选项3</el-menu-item>
63. </el-menu-item-group>
64. <el-submenu index="1-4">
65. <template slot="title">选项4</template>
66. <el-menu-item index="1-4-1">选项1</el-menu-item>
67. </el-submenu>
68. </el-submenu>
69. <el-menu-item index="2">
70. <i class="el-icon-menu"></i>
71. 导航二
72. </el-menu-item>
73. <el-menu-item index="3" disabled>
74. <i class="el-icon-document"></i>
75. 导航三
76. </el-menu-item>
77. <el-menu-item index="4">
78. <i class="el-icon-setting"></i>
79. 导航四
80. </el-menu-item>
81. </el-menu>
82. </el-col>
83. </el-row>
84.
85. <script>
86. export default {
87. methods: {
88. handleOpen(key, keyPath) {
89. console.log(key, keyPath);
90. },
91. handleClose(key, keyPath) {
92. console.log(key, keyPath);
93. }
94. }
95. }
96. </script>
```

```
95. }
96. </script>
```

## 折叠



```
1. <el-radio-group v-model="isCollapse" style="margin-bottom: 20px;">
2. <el-radio-button :label="false">展开</el-radio-button>
3. <el-radio-button :label="true">收起</el-radio-button>
4. </el-radio-group>
5. <el-menu default-active="1-4-1" class="el-menu-vertical-demo"
 @open="handleOpen" @close="handleClose" :collapse="isCollapse">
6. <el-submenu index="1">
7. <template slot="title">
8. <i class="el-icon-location"></i>
9. 导航一
10. </template>
11. <el-menu-item-group>
12. 分组一
13. <el-menu-item index="1-1">选项1</el-menu-item>
14. <el-menu-item index="1-2">选项2</el-menu-item>
15. </el-menu-item-group>
16. <el-menu-item-group title="分组2">
17. <el-menu-item index="1-3">选项3</el-menu-item>
18. </el-menu-item-group>
19. <el-submenu index="1-4">
20. 选项4
21. <el-menu-item index="1-4-1">选项1</el-menu-item>
22. </el-submenu>
```

```
23. </el-submenu>
24. <el-menu-item index="2">
25. <i class="el-icon-menu"></i>
26. 导航二
27. </el-menu-item>
28. <el-menu-item index="3" disabled>
29. <i class="el-icon-document"></i>
30. 导航三
31. </el-menu-item>
32. <el-menu-item index="4">
33. <i class="el-icon-setting"></i>
34. 导航四
35. </el-menu-item>
36. </el-menu>
37.
38. <style>
39. .el-menu-vertical-demo:not(.el-menu--collapse) {
40. width: 200px;
41. min-height: 400px;
42. }
43. </style>
44.
45. <script>
46. export default {
47. data() {
48. return {
49. isCollapse: true
50. };
51. },
52. methods: {
53. handleOpen(key, keyPath) {
54. console.log(key, keyPath);
55. },
56. handleClose(key, keyPath) {
57. console.log(key, keyPath);
58. }
59. }
60. }
61. </script>
```

显示代码

## Menu Attribute

| 参数                  | 说明                                                      | 类型      | 可选值                   | 默认值      |
|---------------------|---------------------------------------------------------|---------|-----------------------|----------|
| mode                | 模式                                                      | string  | horizontal / vertical | vertical |
| collapse            | 是否水平折叠收起菜单 (仅在 mode 为 vertical 时可用)                     | boolean | -                     | false    |
| background-color    | 菜单的背景色 (仅支持 hex 格式)                                     | string  | -                     | #ffffff  |
| text-color          | 菜单的文字颜色 (仅支持 hex 格式)                                    | string  | -                     | #303133  |
| active-text-color   | 当前激活菜单的文字颜色 (仅支持 hex 格式)                                | string  | -                     | #409EFF  |
| default-active      | 当前激活菜单的 index                                           | string  | -                     | -        |
| default-openeds     | 当前打开的 sub-menu 的 index 的数组                              | Array   | -                     | -        |
| unique-opened       | 是否只保持一个子菜单的展开                                           | boolean | -                     | false    |
| menu-trigger        | 子菜单打开的触发方式(只在 mode 为 horizontal 时有效)                    | string  | hover / click         | hover    |
| router              | 是否使用 vue-router 的模式, 启用该模式会在激活导航时以 index 作为 path 进行路由跳转 | boolean | -                     | false    |
| collapse-transition | 是否开启折叠动画                                                | boolean | -                     | true     |

## Menu Methods

| 事件名称  | 说明             | 参数                            |
|-------|----------------|-------------------------------|
| open  | 展开指定的 sub-menu | index: 需要打开的 sub-menu 的 index |
| close | 收起指定的 sub-menu | index: 需要收起的 sub-menu 的 index |

## Menu Events

| 事件名称   | 说明             | 回调参数                                                              |
|--------|----------------|-------------------------------------------------------------------|
| select | 菜单激活回调         | index: 选中菜单项的 index, indexPath: 选中菜单项的 index path                 |
| open   | sub-menu 展开的回调 | index: 打开的 sub-menu 的 index, indexPath: 打开的 sub-menu 的 index path |
| close  | sub-menu 收起的回调 | index: 收起的 sub-menu 的 index, indexPath: 收起的 sub-menu 的 index path |

## SubMenu Attribute

|  |  |  |   |  |
|--|--|--|---|--|
|  |  |  | 可 |  |
|--|--|--|---|--|

| 参数                    | 说明                                      | 类型      | 选值 | 默认值                            |
|-----------------------|-----------------------------------------|---------|----|--------------------------------|
| index                 | 唯一标志                                    | string  | -  | -                              |
| popper-class          | 弹出菜单的自定义类名                              | string  | -  | -                              |
| show-timeout          | 展开 sub-menu 的延时                         | number  | -  | 300                            |
| hide-timeout          | 收起 sub-menu 的延时                         | number  | -  | 300                            |
| disabled              | 是否禁用                                    | boolean | -  | false                          |
| popper-append-to-body | 是否将弹出菜单插入至 body 元素。在菜单的定位出现问题时，可尝试修改该属性 | boolean | -  | 一级子菜单: true<br>/ 非一级子菜单: false |

## Menu-Item Attribute

| 参数       | 说明              | 类型      | 可选值 | 默认值   |
|----------|-----------------|---------|-----|-------|
| index    | 唯一标志            | string  | -   | -     |
| route    | Vue Router 路径对象 | Object  | -   | -     |
| disabled | 是否禁用            | boolean | -   | false |

## Menu-Group Attribute

| 参数    | 说明   | 类型     | 可选值 | 默认值 |
|-------|------|--------|-----|-----|
| title | 分组标题 | string | -   | -   |

原文: <http://element-cn.eleme.io/#/zh-CN/component/menu>

# Tabs 标签页

## Tabs 标签页

分隔内容上有关联但属于不同类别的数据集合。

### 基础用法

基础的、简洁的标签页。

用户管理    配置管理    角色管理    定时任务补偿

配置管理

Tabs 组件提供了选项卡功能，默认选中第一个标签页，你也可以通过 `value` 属性来指定当前选中的标签页。

```
1. <template>
2. <el-tabs v-model="activeName" @tab-click="handleClick">
3. <el-tab-pane label="用户管理" name="first">用户管理</el-tab-pane>
4. <el-tab-pane label="配置管理" name="second">配置管理</el-tab-pane>
5. <el-tab-pane label="角色管理" name="third">角色管理</el-tab-pane>
6. <el-tab-pane label="定时任务补偿" name="fourth">定时任务补偿</el-tab-pane>
7. </el-tabs>
8. </template>
9. <script>
10. export default {
11. data() {
12. return {
13. activeName: 'second'
14. };
15. },
16. methods: {
17. handleClick(tab, event) {
18. console.log(tab, event);
19. }
20. }
21. };
22. </script>
```

## 选项卡样式

选项卡样式的标签页。



### 用户管理

只需要设置 `type` 属性为 `card` 就可以使选项卡改变为标签风格。

```

1. <template>
2. <el-tabs v-model="activeName2" type="card" @tab-click="handleClick">
3. <el-tab-pane label="用户管理" name="first">用户管理</el-tab-pane>
4. <el-tab-pane label="配置管理" name="second">配置管理</el-tab-pane>
5. <el-tab-pane label="角色管理" name="third">角色管理</el-tab-pane>
6. <el-tab-pane label="定时任务补偿" name="fourth">定时任务补偿</el-tab-pane>
7. </el-tabs>
8. </template>
9. <script>
10. export default {
11. data() {
12. return {
13. activeName2: 'first'
14. };
15. },
16. methods: {
17. handleClick(tab, event) {
18. console.log(tab, event);
19. }
20. }
21. };
22. </script>

```

## 卡片化

卡片化的标签页。



将 `type` 设置为 `border-card`。

```

1. <el-tabs type="border-card">
2. <el-tab-pane label="用户管理">用户管理</el-tab-pane>
3. <el-tab-pane label="配置管理">配置管理</el-tab-pane>
4. <el-tab-pane label="角色管理">角色管理</el-tab-pane>
5. <el-tab-pane label="定时任务补偿">定时任务补偿</el-tab-pane>
6. </el-tabs>

```

## 位置

可以通过 `tab-position` 设置标签的位置

|     |       |        |      |
|-----|-------|--------|------|
| top | right | bottom | left |
|-----|-------|--------|------|



标签一共有四个方向的设置 `tabPosition="left|right|top|bottom"`

```

1. <template>
2. <el-radio-group v-model="tabPosition" style="margin-bottom: 30px;">
3. <el-radio-button label="top">top</el-radio-button>
4. <el-radio-button label="right">right</el-radio-button>
5. <el-radio-button label="bottom">bottom</el-radio-button>
6. <el-radio-button label="left">left</el-radio-button>
7. </el-radio-group>
8.
9. <el-tabs :tab-position="tabPosition" style="height: 200px;">
10. <el-tab-pane label="用户管理">用户管理</el-tab-pane>
11. <el-tab-pane label="配置管理">配置管理</el-tab-pane>
12. <el-tab-pane label="角色管理">角色管理</el-tab-pane>
13. <el-tab-pane label="定时任务补偿">定时任务补偿</el-tab-pane>
14. </el-tabs>

```

```

15. </template>
16. <script>
17. export default {
18. data() {
19. return {
20. tabPosition: 'top'
21. };
22. }
23. };
24. </script>

```

## 自定义标签页

可以通过具名 `slot` 来实现自定义标签页的内容



```

1. <el-tabs type="border-card">
2. <el-tab-pane>
3. <i class="el-icon-date"></i> 我的行程
4. 我的行程
5. </el-tab-pane>
6. <el-tab-pane label="消息中心">消息中心</el-tab-pane>
7. <el-tab-pane label="角色管理">角色管理</el-tab-pane>
8. <el-tab-pane label="定时任务补偿">定时任务补偿</el-tab-pane>
9. </el-tabs>

```

## 动态增减标签页

增减标签页按钮只能在选项卡样式的标签页下使用



Tab 2 content

```

1. <el-tabs v-model="editableTabsValue" type="card" editable
@edit="handleTabsEdit">

```

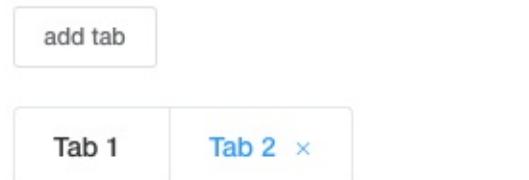
```
2. <el-tab-pane
3. :key="item.name"
4. v-for="(item, index) in editableTabs"
5. :label="item.title"
6. :name="item.name"
7. >
8. {{item.content}}
9. </el-tab-pane>
10. </el-tabs>
11. <script>
12. export default {
13. data() {
14. return {
15. editableTabsValue: '2',
16. editableTabs: [{{
17. title: 'Tab 1',
18. name: '1',
19. content: 'Tab 1 content'
20. }, {{
21. title: 'Tab 2',
22. name: '2',
23. content: 'Tab 2 content'
24. }],
25. tabIndex: 2
26. }
27. },
28. methods: {
29. handleTabsEdit(targetName, action) {
30. if (action === 'add') {
31. let newTabName = ++this.tabIndex + '';
32. this.editableTabs.push({
33. title: 'New Tab',
34. name: newTabName,
35. content: 'New Tab content'
36. });
37. this.editableTabsValue = newTabName;
38. }
39. if (action === 'remove') {
40. let tabs = this.editableTabs;
41. let activeName = this.editableTabsValue;
42. if (activeName === targetName) {
43. tabs.forEach((tab, index) => {
```

```

44. if (tab.name === targetName) {
45. let nextTab = tabs[index + 1] || tabs[index - 1];
46. if (nextTab) {
47. activeName = nextTab.name;
48. }
49. }
50. });
51. }
52.
53. this.editableTabsValue = activeName;
54. this.editableTabs = tabs.filter(tab => tab.name !== targetName);
55. }
56. }
57. }
58. }
59. </script>

```

## 自定义增加标签页触发器



Tab 2 content

```

1. <div style="margin-bottom: 20px;">
2. <el-button
3. size="small"
4. @click="addTab(editableTabsValue2)"
5. >
6. add tab
7. </el-button>
8. </div>
9. <el-tabs v-model="editableTabsValue2" type="card" closable @tab-
remove="removeTab">
10. <el-tab-pane
11. v-for="(item, index) in editableTabs2"
12. :key="item.name"
13. :label="item.title"
14. :name="item.name"

```

```
15. >
16. {{item.content}}
17. </el-tab-pane>
18. </el-tabs>
19. <script>
20. export default {
21. data() {
22. return {
23. editableTabsValue2: '2',
24. editableTabs2: [{{
25. title: 'Tab 1',
26. name: '1',
27. content: 'Tab 1 content'
28. }, {{
29. title: 'Tab 2',
30. name: '2',
31. content: 'Tab 2 content'
32. }],
33. tabIndex: 2
34. }
35. },
36. methods: {
37. addTab(targetName) {
38. let newTabName = ++this.tabIndex + '';
39. this.editableTabs2.push({
40. title: 'New Tab',
41. name: newTabName,
42. content: 'New Tab content'
43. });
44. this.editableTabsValue2 = newTabName;
45. },
46. removeTab(targetName) {
47. let tabs = this.editableTabs2;
48. let activeName = this.editableTabsValue2;
49. if (activeName === targetName) {
50. tabs.forEach((tab, index) => {
51. if (tab.name === targetName) {
52. let nextTab = tabs[index + 1] || tabs[index - 1];
53. if (nextTab) {
54. activeName = nextTab.name;
55. }
56. }
57. })
58. }
59. }
60. }
61. }
62.
```

```

57. });
58. }
59.
60. this.editableTabsValue2 = activeName;
61. this.editableTabs2 = tabs.filter(tab => tab.name !== targetName);
62. }
63. }
64. }
65. </script>

```

## Tabs Attributes

| 参数           | 说明                                                | 类型                                  | 可选值                   | 默认值          |
|--------------|---------------------------------------------------|-------------------------------------|-----------------------|--------------|
| type         | 风格类型                                              | string                              | card/border-card      | -            |
| closable     | 标签是否可关闭                                           | boolean                             | -                     | false        |
| addable      | 标签是否可增加                                           | boolean                             | -                     | false        |
| editable     | 标签是否同时可增加和关闭                                      | boolean                             | -                     | false        |
| value        | 绑定值，选中选项卡的 name                                   | string                              | -                     | 第一个选项卡的 name |
| tab-position | 选项卡所在位置                                           | string                              | top/right/bottom/left | top          |
| stretch      | 标签的宽度是否自撑开                                        | boolean                             | -                     | false        |
| before-leave | 切换标签之前的钩子，若返回 false 或者返回 Promise 且被 reject，则阻止切换。 | Function(activeName, oldActiveName) | -                     | -            |

## Tabs Events

| 事件名称       | 说明                        | 回调参数                 |
|------------|---------------------------|----------------------|
| tab-click  | tab 被选中时触发                | 被选中的标签 tab 实例        |
| tab-remove | 点击 tab 移除按钮后触发            | 被删除的标签的 name         |
| tab-add    | 点击 tabs 的新增按钮后触发          | -                    |
| edit       | 点击 tabs 的新增按钮或 tab 被关闭后触发 | (targetName, action) |

## Tab-pane Attributes

| 参数 | 说明 | 类型 | 可选 | 默认值 |
|----|----|----|----|-----|
|    |    |    |    |     |

|          |                                |         | 值 |                              |
|----------|--------------------------------|---------|---|------------------------------|
| label    | 选项卡标题                          | string  | — | —                            |
| disabled | 是否禁用                           | boolean | — | false                        |
| name     | 与选项卡 activeName 对应的标识符，表示选项卡别名 | string  | — | 该选项卡在选项卡列表中的顺序值，如第一个选项卡则为'1' |
| closable | 标签是否可关闭                        | boolean | — | false                        |
| lazy     | 标签是否延迟渲染                       | boolean | — | false                        |

原文: <http://element-cn.eleme.io/#/zh-CN/component/tabs>

# Breadcrumb 面包屑

## Breadcrumb 面包屑

显示当前页面的路径，快速返回之前的任意页面。

### 基础用法

适用广泛的基础用法。

首页 / 活动管理 / 活动列表 / 活动详情

在 `el-breadcrumb` 中使用 `el-breadcrumb-item` 标签表示从首页开始的每一级。Element 提供了一个 `separator` 属性，在 `el-breadcrumb` 标签中设置它来决定分隔符，它只能是字符串，默认为斜杠 `/`。

```
1. <el-breadcrumb separator="/">
2. <el-breadcrumb-item :to="{ path: '/' }">首页</el-breadcrumb-item>
3. <el-breadcrumb-item>活动管理</el-breadcrumb-item>
4. <el-breadcrumb-item>活动列表</el-breadcrumb-item>
5. <el-breadcrumb-item>活动详情</el-breadcrumb-item>
6. </el-breadcrumb>
```

### 图标分隔符

首页 > 活动管理 > 活动列表 > 活动详情

通过设置 `separator-class` 可使用相应的 `iconfont` 作为分隔符，注意这将使 `separator` 设置失效

```
1. <el-breadcrumb separator-class="el-icon-arrow-right">
2. <el-breadcrumb-item :to="{ path: '/' }">首页</el-breadcrumb-item>
3. <el-breadcrumb-item>活动管理</el-breadcrumb-item>
4. <el-breadcrumb-item>活动列表</el-breadcrumb-item>
5. <el-breadcrumb-item>活动详情</el-breadcrumb-item>
6. </el-breadcrumb>
```

## Breadcrumb Attributes

| 参数              | 说明          | 类型     | 可选值 | 默认值   |
|-----------------|-------------|--------|-----|-------|
| separator       | 分隔符         | string | -   | 斜杠'/' |
| separator-class | 图标分隔符 class | string | -   | -     |

## Breadcrumb Item Attributes

| 参数      | 说明                                                                                  | 类型            | 可选值 | 默认值   |
|---------|-------------------------------------------------------------------------------------|---------------|-----|-------|
| to      | 路由跳转对象，同 <code>vue-router</code> 的 <code>to</code>                                  | string/object | -   | -     |
| replace | 在使用 <code>to</code> 进行路由跳转时，启用 <code>replace</code> 将不会向 <code>history</code> 添加新记录 | boolean       | -   | false |

原文: <http://element-cn.eleme.io/#/zh-CN/component/breadcrumb>

# Dropdown 下拉菜单

## Dropdown 下拉菜单

将动作或菜单折叠到下拉菜单中。

### 基础用法

移动到下拉菜单上，展开更多操作。

下拉菜单 ▾



通过组件 `slot` 来设置下拉触发的元素以及需要通过具名 `slot` 为 `dropdown` 来设置下拉菜单。默认情况下，下拉按钮只要 `hover` 即可，无需点击也会显示下拉菜单。

```

1. <el-dropdown>
2.
3. 下拉菜单<i class="el-icon-arrow-down el-icon--right"></i>
4.
5. <el-dropdown-menu slot="dropdown">
6. <el-dropdown-item>黄金糕</el-dropdown-item>
7. <el-dropdown-item>狮子头</el-dropdown-item>
8. <el-dropdown-item>螺蛳粉</el-dropdown-item>
9. <el-dropdown-item disabled>双皮奶</el-dropdown-item>
10. <el-dropdown-item divided>蚵仔煎</el-dropdown-item>
11. </el-dropdown-menu>
12. </el-dropdown>

```

### 触发对象

可使用按钮触发下拉菜单。



设置 `split-button` 属性来让触发下拉元素呈现为按钮组，左边是功能按钮，右边是触发下拉菜单的按钮，设置为 `true` 即可。

```

1. <el-dropdown>
2. <el-button type="primary">
3. 更多菜单<i class="el-icon-arrow-down el-icon--right"></i>
4. </el-button>
5. <el-dropdown-menu slot="dropdown">
6. <el-dropdown-item>黄金糕</el-dropdown-item>
7. <el-dropdown-item>狮子头</el-dropdown-item>
8. <el-dropdown-item>螺蛳粉</el-dropdown-item>
9. <el-dropdown-item>双皮奶</el-dropdown-item>
10. <el-dropdown-item>蚵仔煎</el-dropdown-item>
11. </el-dropdown-menu>
12. </el-dropdown>
13. <el-dropdown split-button type="primary" @click="handleClick">
14. 更多菜单
15. <el-dropdown-menu slot="dropdown">
16. <el-dropdown-item>黄金糕</el-dropdown-item>
17. <el-dropdown-item>狮子头</el-dropdown-item>
18. <el-dropdown-item>螺蛳粉</el-dropdown-item>
19. <el-dropdown-item>双皮奶</el-dropdown-item>
20. <el-dropdown-item>蚵仔煎</el-dropdown-item>
21. </el-dropdown-menu>
22. </el-dropdown>
23.
24. <style>
25. .el-dropdown {
26. vertical-align: top;
27. }
28. .el-dropdown + .el-dropdown {
29. margin-left: 15px;
30. }
31. .el-icon-arrow-down {
```

```

32. font-size: 12px;
33. }
34. </style>
35.
36. <script>
37. export default {
38. methods: {
39. handleClick() {
40. alert('button click');
41. }
42. }
43. }
44. </script>
45.

```

[显示代码](#)

## 触发方式

可以配置 click 激活或者 hover 激活。

在 `trigger` 属性设置为 `click` 即可。

```

1. <el-row class="block-col-2">
2. <el-col :span="12">
3. hover 激活
4. <el-dropdown>
5.
6. 下拉菜单<i class="el-icon-arrow-down el-icon--right"></i>
7.
8. <el-dropdown-menu slot="dropdown">
9. <el-dropdown-item>黄金糕</el-dropdown-item>
10. <el-dropdown-item>狮子头</el-dropdown-item>
11. <el-dropdown-item>螺蛳粉</el-dropdown-item>
12. <el-dropdown-item>双皮奶</el-dropdown-item>
13. <el-dropdown-item>蚵仔煎</el-dropdown-item>
14. </el-dropdown-menu>
15. </el-dropdown>
16. </el-col>
17. <el-col :span="12">
18. click 激活
19. <el-dropdown trigger="click">

```

```

20.
21. 下拉菜单<i class="el-icon-arrow-down el-icon--right"></i>
22.
23. <el-dropdown-menu slot="dropdown">
24. <el-dropdown-item>黄金糕</el-dropdown-item>
25. <el-dropdown-item>狮子头</el-dropdown-item>
26. <el-dropdown-item>螺蛳粉</el-dropdown-item>
27. <el-dropdown-item>双皮奶</el-dropdown-item>
28. <el-dropdown-item>蚵仔煎</el-dropdown-item>
29. </el-dropdown-menu>
30. </el-dropdown>
31. </el-col>
32. </el-row>

```

## 菜单隐藏方式

可以 `hide-on-click` 属性来配置。



下拉菜单默认在点击菜单项后会被隐藏，将 `hide-on-click` 属性默认为 `false` 可以关闭此功能。

```

1. <el-dropdown :hide-on-click="false">
2.
3. 下拉菜单<i class="el-icon-arrow-down el-icon--right"></i>
4.
5. <el-dropdown-menu slot="dropdown">
6. <el-dropdown-item>黄金糕</el-dropdown-item>
7. <el-dropdown-item>狮子头</el-dropdown-item>
8. <el-dropdown-item>螺蛳粉</el-dropdown-item>
9. <el-dropdown-item disabled>双皮奶</el-dropdown-item>
10. <el-dropdown-item divided>蚵仔煎</el-dropdown-item>
11. </el-dropdown-menu>
12. </el-dropdown>
13.
14. <style>
15. .el-dropdown-link {

```

```

16. cursor: pointer;
17. color: #409EFF;
18. }
19. .el-icon-arrow-down {
20. font-size: 12px;
21. }
22. </style>

```

## 指令事件

点击菜单项后会触发事件，用户可以通过相应的菜单项 key 进行不同的操作



i click on item a

下拉菜单 ▾

```

1. <el-dropdown @command="handleCommand">
2.
3. 下拉菜单<i class="el-icon-arrow-down el-icon--right"></i>
4.
5. <el-dropdown-menu slot="dropdown">
6. <el-dropdown-item command="a">黄金糕</el-dropdown-item>
7. <el-dropdown-item command="b">狮子头</el-dropdown-item>
8. <el-dropdown-item command="c">螺蛳粉</el-dropdown-item>
9. <el-dropdown-item command="d" disabled>双皮奶</el-dropdown-item>
10. <el-dropdown-item command="e" divided>蚵仔煎</el-dropdown-item>
11. </el-dropdown-menu>
12. </el-dropdown>
13.
14. <style>
15. .el-dropdown-link {
16. cursor: pointer;
17. color: #409EFF;
18. }
19. .el-icon-arrow-down {
20. font-size: 12px;
21. }
22. </style>
23.

```

```

24. <script>
25. export default {
26. methods: {
27. handleCommand(command) {
28. this.$message('click on item ' + command);
29. }
30. }
31. }
32. </script>

```

## 不同尺寸

Dropdown 组件提供除了默认值以外的三种尺寸，可以在不同场景下选择合适的尺寸。



额外的尺寸：`medium`、`small`、`mini`，通过设置 `size` 属性来配置它们。

```

1. <el-dropdown split-button type="primary">
2. 默认尺寸
3. <el-dropdown-menu slot="dropdown">
4. <el-dropdown-item>黄金糕</el-dropdown-item>
5. <el-dropdown-item>狮子头</el-dropdown-item>
6. <el-dropdown-item>螺蛳粉</el-dropdown-item>
7. <el-dropdown-item>双皮奶</el-dropdown-item>
8. <el-dropdown-item>蚵仔煎</el-dropdown-item>
9. </el-dropdown-menu>
10. </el-dropdown>
11.
12. <el-dropdown size="medium" split-button type="primary">
13. 中等尺寸
14. <el-dropdown-menu slot="dropdown">
15. <el-dropdown-item>黄金糕</el-dropdown-item>
16. <el-dropdown-item>狮子头</el-dropdown-item>
17. <el-dropdown-item>螺蛳粉</el-dropdown-item>
18. <el-dropdown-item>双皮奶</el-dropdown-item>
19. <el-dropdown-item>蚵仔煎</el-dropdown-item>
20. </el-dropdown-menu>
21. </el-dropdown>
22.
23. <el-dropdown size="small" split-button type="primary">

```

```

24. 小型尺寸
25. <el-dropdown-menu slot="dropdown">
26. <el-dropdown-item>黄金糕</el-dropdown-item>
27. <el-dropdown-item>狮子头</el-dropdown-item>
28. <el-dropdown-item>螺蛳粉</el-dropdown-item>
29. <el-dropdown-item>双皮奶</el-dropdown-item>
30. <el-dropdown-item>蚵仔煎</el-dropdown-item>
31. </el-dropdown-menu>
32. </el-dropdown>
33.
34. <el-dropdown size="mini" split-button type="primary">
35. 超小尺寸
36. <el-dropdown-menu slot="dropdown">
37. <el-dropdown-item>黄金糕</el-dropdown-item>
38. <el-dropdown-item>狮子头</el-dropdown-item>
39. <el-dropdown-item>螺蛳粉</el-dropdown-item>
40. <el-dropdown-item>双皮奶</el-dropdown-item>
41. <el-dropdown-item>蚵仔煎</el-dropdown-item>
42. </el-dropdown-menu>
43. </el-dropdown>

```

## Dropdown Attributes

| 参数            | 说明                                                                                           | 类型      | 可选值                                                  | 默认值        |
|---------------|----------------------------------------------------------------------------------------------|---------|------------------------------------------------------|------------|
| type          | 菜单按钮类型，同 <code>Button</code> 组件<br>(只在 <code>split-button</code> 为 <code>true</code> 的情况下有效) | string  | —                                                    | —          |
| size          | 菜单尺寸，在 <code>split-button</code> 为 <code>true</code> 的情况下也对触发按钮生效                            | string  | medium / small / mini                                | —          |
| split-button  | 下拉触发元素呈现为按钮组                                                                                 | boolean | —                                                    | false      |
| placement     | 菜单弹出位置                                                                                       | string  | top/top-start/top-end/bottom/bottom-start/bottom-end | bottom-end |
| trigger       | 触发下拉的行为                                                                                      | string  | hover, click                                         | hover      |
| hide-on-click | 是否在点击菜单项后隐藏菜单                                                                                | boolean | —                                                    | true       |
| show-timeout  | 展开下拉菜单的延时 (仅在 trigger 为 hover 时有效)                                                           | number  | —                                                    | 250        |
| hide-timeout  | 收起下拉菜单的延时 (仅在 trigger 为 hover 时有效)                                                           | number  | —                                                    | 150        |

## Dropdown Events

| 事件名称           | 说明                                           | 回调参数                 |
|----------------|----------------------------------------------|----------------------|
| click          | <code>split-button</code> 为 true 时，点击左侧按钮的回调 | —                    |
| command        | 点击菜单项触发的事件回调                                 | dropdown-item 的指令    |
| visible-change | 下拉框出现/隐藏时触发                                  | 出现则为 true，隐藏则为 false |

## Dropdown Menu Item Attributes

| 参数       | 说明    | 类型                   | 可选值 | 默认值   |
|----------|-------|----------------------|-----|-------|
| command  | 指令    | string/number/object | —   | —     |
| disabled | 禁用    | boolean              | —   | false |
| divided  | 显示分割线 | boolean              | —   | false |

原文: <http://element-cn.eleme.io/#/zh-CN/component/dropdown>

# Steps 步骤条

## Steps 步骤条

引导用户按照流程完成任务的分步导航条，可根据实际应用场景设定步骤，步骤不得少于 2 步。

### 基础用法

简单的步骤条。



设置 `active` 属性，接受一个 `Number`，表明步骤的 `index`，从 0 开始。需要定宽的步骤条时，设置 `space` 属性即可，它接受 `Boolean`，单位为 `px`，如果不设置，则为自适应。设置 `finish-status` 属性可以改变已经完成的步骤的状态。

```

1. <el-steps :active="active" finish-status="success">
2. <el-step title="步骤 1"></el-step>
3. <el-step title="步骤 2"></el-step>
4. <el-step title="步骤 3"></el-step>
5. </el-steps>
6.
7. <el-button style="margin-top: 12px;" @click="next">下一步</el-button>
8.
9. <script>
10. export default {
11. data() {
12. return {
13. active: 0
14. };
15. },
16.
17. methods: {
18. next() {
19. if (this.active++ > 2) this.active = 0;
20. }
21. }
22. }

```

```

21. }
22. }
23. </script>

```

## 含状态步骤条

每一步骤显示出该步骤的状态。



也可以使用 `title` 具名分发，可以用 `slot` 的方式来取代属性的设置，在本文档最后的列表中有所有的 `slot name` 可供参考。

```

1. <el-steps :space="200" :active="1" finish-status="success">
2. <el-step title="已完成"></el-step>
3. <el-step title="进行中"></el-step>
4. <el-step title="步骤 3"></el-step>
5. </el-steps>

```

## 有描述的步骤条

每个步骤有其对应的步骤状态描述。



```

1. <el-steps :active="1">
2. <el-step title="步骤 1" description="这是一段很长很长很长的描述性文字"></el-step>
3. <el-step title="步骤 2" description="这是一段很长很长很长的描述性文字"></el-step>
4. <el-step title="步骤 3" description="这段就没那么长了"></el-step>
5. </el-steps>

```

## 居中的步骤条

标题和描述都将居中。



```

1. <el-steps :active="2" align-center>
2. <el-step title="步骤1" description="这是一段很长很长很长的描述性文字"></el-step>
3. <el-step title="步骤2" description="这是一段很长很长很长的描述性文字"></el-step>
4. <el-step title="步骤3" description="这是一段很长很长很长的描述性文字"></el-step>
5. <el-step title="步骤4" description="这是一段很长很长很长的描述性文字"></el-step>
6. </el-steps>

```

## 带图标的步骤条

步骤条内可以启用各种自定义的图标。



通过 `icon` 属性来设置图标，图标的类型可以参考 `Icon` 组件的文档，除此以外，还能通过具名 `slot` 来使用自定义的图标。

```

1.
2. <el-steps :active="1">
3. <el-step title="步骤 1" icon="el-icon-edit"></el-step>
4. <el-step title="步骤 2" icon="el-icon-upload"></el-step>
5. <el-step title="步骤 3" icon="el-icon-picture"></el-step>
6. </el-steps>

```

## 竖式步骤条

竖直方向的步骤条。

① 步骤 1

② 步骤 2

③ 步骤 3

这是一段很长很长很长的描述性文字

只需要在 `el-steps` 元素中设置 `direction` 属性为 `vertical` 即可。

```

1. <div style="height: 300px;">
2. <el-steps direction="vertical" :active="1">
3. <el-step title="步骤 1"></el-step>
4. <el-step title="步骤 2"></el-step>
5. <el-step title="步骤 3" description="这是一段很长很长很长的描述性文字"></el-step>
6. </el-steps>
7. </div>

```

## 简洁风格的步骤条

设置 `simple` 可应用简洁风格，该条件下 `align-center` / `description` / `direction` / `space` 都将失效。

步骤 1 > 步骤 2 > 步骤 3

步骤 1 > 步骤 2 > 步骤 3

```

1.
2. <el-steps :active="1" simple>
3. <el-step title="步骤 1" icon="el-icon-edit"></el-step>
4. <el-step title="步骤 2" icon="el-icon-upload"></el-step>
5. <el-step title="步骤 3" icon="el-icon-picture"></el-step>
6. </el-steps>

```

```

7.
8. <el-steps :active="1" finish-status="success" simple style="margin-top: 20px">
9. <el-step title="步骤 1" ></el-step>
10. <el-step title="步骤 2" ></el-step>
11. <el-step title="步骤 3" ></el-step>
12. </el-steps>

```

## Steps Attributes

| 参数             | 说明                           | 类型              | 可选值                                       | 默认值        |
|----------------|------------------------------|-----------------|-------------------------------------------|------------|
| space          | 每个 step 的间距，不填写将自适应间距。支持百分比。 | number / string | —                                         | —          |
| direction      | 显示方向                         | string          | vertical/horizontal                       | horizontal |
| active         | 设置当前激活步骤                     | number          | —                                         | 0          |
| process-status | 设置当前步骤的状态                    | string          | wait / process / finish / error / success | process    |
| finish-status  | 设置结束步骤的状态                    | string          | wait / process / finish / error / success | finish     |
| align-center   | 进行居中对齐                       | boolean         | -                                         | false      |
| simple         | 是否应用简洁风格                     | boolean         | -                                         | false      |

## Step Attributes

| 参数          | 说明                          | 类型                                        | 可选值    | 默认值 |
|-------------|-----------------------------|-------------------------------------------|--------|-----|
| title       | 标题                          | string                                    | —      | —   |
| description | 描述性文字                       | string                                    | —      | —   |
| icon        | 图标                          | 传入 icon 的 class 全名来自定义 icon，也支持 slot 方式写入 | string | —   |
| status      | 设置当前步骤的状态，不设置则根据 steps 确定状态 | wait / process / finish / error / success | -      |     |

## Step Slot

| name        | 说明    |
|-------------|-------|
| icon        | 图标    |
| title       | 标题    |
| description | 描述性文字 |

原文: <http://element-cn.eleme.io/#/zh-CN/component/steps>

# Others 其他组件

- [Dialog 对话框](#)
- [Tooltip 文字提示](#)
- [Popover 弹出框](#)
- [Card 卡片](#)
- [Carousel 走马灯](#)
- [Collapse 折叠面板](#)

# Dialog 对话框

## Dialog 对话框

在保留当前页面状态的情况下，告知用户并承载相关操作。

### 基本用法

Dialog 弹出一个对话框，适合需要定制性更大的场景。



需要设置 `visible` 属性，它接收 `Boolean`，当为 `true` 时显示 Dialog。Dialog 分为两个部分：`body` 和 `footer`，`footer` 需要具名为 `footer` 的 `slot`。`title` 属性用于定义标题，它是可选的，默认值为空。最后，本例还展示了 `before-close` 的用法。

```
1. <el-button type="text" @click="dialogVisible = true">点击打开 Dialog</el-button>
2.
3. <el-dialog
4. title="提示"
5. :visible.sync="dialogVisible"
6. width="30%"
7. :before-close="handleClose">
8. 这是一段信息
9.
10. <el-button @click="dialogVisible = false">取 消</el-button>
11. <el-button type="primary" @click="dialogVisible = false">确 定</el-button>
12.
13. </el-dialog>
14.
15. <script>
16. export default {
```

```

17. data() {
18. return {
19. dialogVisible: false
20. };
21. },
22. methods: {
23. handleClose(done) {
24. this.$confirm('确认关闭？')
25. .then(_ => {
26. done();
27. })
28. .catch(_ => {});
29. }
30. }
31. };
32. </script>

```

`before-close` 仅当用户通过点击关闭图标或遮罩关闭 Dialog 时起效。如果你在具名 slot 里添加了用于关闭 Dialog 的按钮，那么可以在按钮的点击回调函数里加入 `before-close` 的相关逻辑。

## 自定义内容

Dialog 组件的内容可以是任意的，甚至可以是表格或表单，下面是应用了 Element Table 和 Form 组件的两个样例。

| 日期         | 姓名  | 地址                |
|------------|-----|-------------------|
| 2016-05-02 | 王小虎 | 上海市普陀区金沙江路 1518 弄 |
| 2016-05-04 | 王小虎 | 上海市普陀区金沙江路 1518 弄 |
| 2016-05-01 | 王小虎 | 上海市普陀区金沙江路 1518 弄 |
| 2016-05-03 | 王小虎 | 上海市普陀区金沙江路 1518 弄 |



```
1. <!-- Table -->
2. <el-button type="text" @click="dialogTableVisible = true">打开嵌套表格的
 Dialog</el-button>
3.
4. <el-dialog title="收货地址" :visible.sync="dialogTableVisible">
5. <el-table :data="gridData">
6. <el-table-column property="date" label="日期" width="150"></el-table-
 column>
7. <el-table-column property="name" label="姓名" width="200"></el-table-
 column>
8. <el-table-column property="address" label="地址"></el-table-column>
9. </el-table>
10. </el-dialog>
11.
12. <!-- Form -->
13. <el-button type="text" @click="dialogFormVisible = true">打开嵌套表单的
 Dialog</el-button>
14.
15. <el-dialog title="收货地址" :visible.sync="dialogFormVisible">
16. <el-form :model="form">
17. <el-form-item label="活动名称" :label-width="formLabelWidth">
18. <el-input v-model="form.name" auto-complete="off"></el-input>
19. </el-form-item>
20. <el-form-item label="活动区域" :label-width="formLabelWidth">
21. <el-select v-model="form.region" placeholder="请选择活动区域">
22. <el-option label="区域一" value="shanghai"></el-option>
23. <el-option label="区域二" value="beijing"></el-option>
24. </el-select>
```

```
25. </el-form-item>
26. </el-form>
27. <div slot="footer" class="dialog-footer">
28. <el-button @click="dialogFormVisible = false">取 消</el-button>
29. <el-button type="primary" @click="dialogFormVisible = false">确 定</el-
 button>
30. </div>
31. </el-dialog>
32.
33. <script>
34. export default {
35. data() {
36. return {
37. gridData: [
38. { date: '2016-05-02', name: '王小虎', address: '上海市普陀区金沙江路 1518 弄' },
39. { date: '2016-05-04', name: '王小虎', address: '上海市普陀区金沙江路 1518 弄' },
40. { date: '2016-05-01', name: '王小虎', address: '上海市普陀区金沙江路 1518 弄' },
41. { date: '2016-05-03', name: '王小虎', address: '上海市普陀区金沙江路 1518 弄' }
42.],
43. dialogTableVisible: false,
44. dialogFormVisible: false,
45. form: {
46. name: '',
47. region: '',
48. date1: '',
49. date2: '',
50. delivery: false,
51. type: [],
52. resource: '',
53. desc: ''
54. },
55. }
56. }
57. }
58.
```

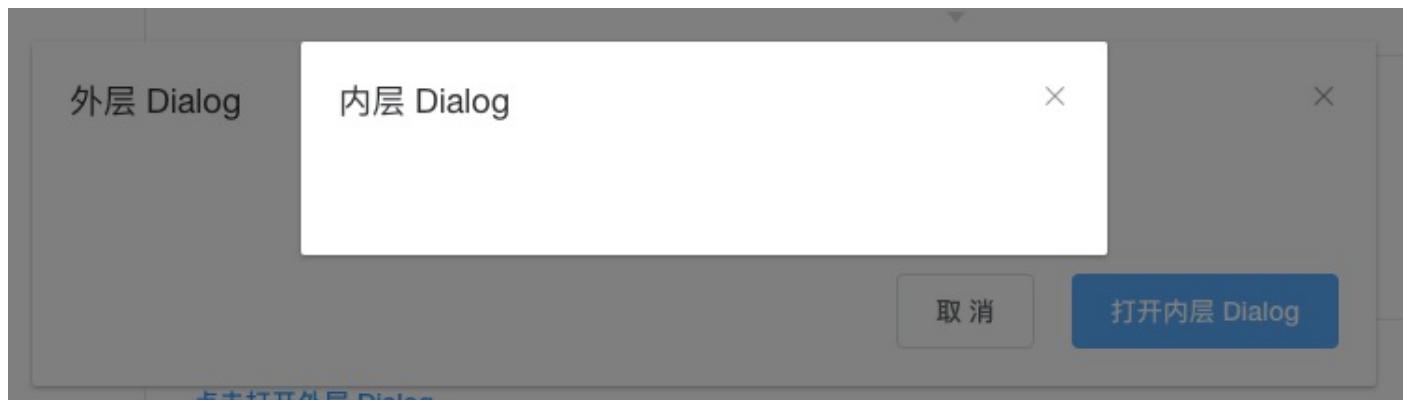
```

66. formLabelWidth: '120px'
67. };
68. }
69. };
70. </script>

```

## 嵌套的 Dialog

如果需要在一个 Dialog 内部嵌套另一个 Dialog，需要使用 `append-to-body` 属性。



正常情况下，我们不建议使用嵌套的 Dialog，如果需要在页面上同时显示多个 Dialog，可以将它们平级放置。对于确实需要嵌套 Dialog 的场景，我们提供了 `append-to-body` 属性。将内层 Dialog 的该属性设置为 `true`，它就会插入至 `body` 元素上，从而保证内外层 Dialog 和遮罩层级关系的正确。

```

1. <template>
2. <el-button type="text" @click="outerVisible = true">点击打开外层 Dialog</el-
 button>
3.
4. <el-dialog title="外层 Dialog" :visible.sync="outerVisible">
5. <el-dialog
6. width="30%"
7. title="内层 Dialog"
8. :visible.sync="innerVisible"
9. append-to-body>
10. </el-dialog>
11. <div slot="footer" class="dialog-footer">
12. <el-button @click="outerVisible = false">取 消</el-button>
13. <el-button type="primary" @click="innerVisible = true">打开内层
 Dialog</el-button>
14. </div>
15. </el-dialog>

```

```

16. </template>
17.
18. <script>
19. export default {
20. data() {
21. return {
22. outerVisible: false,
23. innerVisible: false
24. };
25. }
26. }
27. </script>

```

## 居中布局

标题和底部可水平居中



将 `center` 设置为 `true` 即可使标题和底部居中。`center` 仅影响标题和底部区域。Dialog 的内容是任意的，在一些情况下，内容并不适合居中布局。如果需要内容也水平居中，请自行为其添加 CSS。

```

1. <el-button type="text" @click="centerDialogVisible = true">点击打开 Dialog</el-
button>
2.
3. <el-dialog
4. title="提示"
5. :visible.sync="centerDialogVisible"
6. width="30%"
7. center>
8. 需要注意的是内容是默认不居中的
9.
10. <el-button @click="centerDialogVisible = false">取 消</el-button>

```

```

11. <el-button type="primary" @click="centerDialogVisible = false">确定</el-
button>
12.
13. </el-dialog>
14.
15. <script>
16. export default {
17. data() {
18. return {
19. centerDialogVisible: false
20. };
21. }
22. };
23. </script>

```

Dialog 的内容是懒渲染的，即在第一次被打开之前，传入的默认 slot 不会被渲染到 DOM 上。因此，如果需要执行 DOM 操作，或通过 `ref` 获取相应组件，请在 `open` 事件回调中进行。

如果 `visible` 属性绑定的变量位于 Vuex 的 store 内，那么 `.sync` 不会正常工作。此时需要去除 `.sync` 修饰符，同时监听 Dialog 的 `open` 和 `close` 事件，在事件回调中执行 Vuex 中对应的 mutation 更新 `visible` 属性绑定的变量的值。

## Attributes

| 参数                                | 说明                                                                            | 类型                   | 可选值 | 默认值                |
|-----------------------------------|-------------------------------------------------------------------------------|----------------------|-----|--------------------|
| <code>visible</code>              | 是否显示 Dialog，支持 <code>.sync</code> 修饰符                                         | <code>boolean</code> | —   | <code>false</code> |
| <code>title</code>                | Dialog 的标题，也可通过具名 slot（见下表）传入                                                 | <code>string</code>  | —   | —                  |
| <code>width</code>                | Dialog 的宽度                                                                    | <code>string</code>  | —   | <code>50%</code>   |
| <code>fullscreen</code>           | 是否为全屏 Dialog                                                                  | <code>boolean</code> | —   | <code>false</code> |
| <code>top</code>                  | Dialog CSS 中的 <code>margin-top</code> 值                                       | <code>string</code>  | —   | <code>15vh</code>  |
| <code>modal</code>                | 是否需要遮罩层                                                                       | <code>boolean</code> | —   | <code>true</code>  |
| <code>modal-append-to-body</code> | 遮罩层是否插入至 <code>body</code> 元素上，若为 <code>false</code> ，则遮罩层会插入至 Dialog 的父元素上   | <code>boolean</code> | —   | <code>true</code>  |
| <code>append-to-body</code>       | Dialog 自身是否插入至 <code>body</code> 元素上。嵌套的 Dialog 必须指定该属性并赋值为 <code>true</code> | <code>boolean</code> | —   | <code>false</code> |
| <code>lock-scroll</code>          | 是否在 Dialog 出现时将 <code>body</code> 滚动锁定                                        | <code>boolean</code> | —   | <code>true</code>  |
| <code>custom-class</code>         | Dialog 的自定义类名                                                                 | <code>string</code>  | —   | —                  |

|                       |                          |                                     |   |       |
|-----------------------|--------------------------|-------------------------------------|---|-------|
| close-on-click-modal  | 是否可以通过点击 modal 关闭 Dialog | boolean                             | — | true  |
| close-on-press-escape | 是否可以通过按下 ESC 关闭 Dialog   | boolean                             | — | true  |
| show-close            | 是否显示关闭按钮                 | boolean                             | — | true  |
| before-close          | 关闭前的回调，会暂停 Dialog 的关闭    | function(done),<br>done 用于关闭 Dialog | — | —     |
| center                | 是否对头部和底部采用居中布局           | boolean                             | — | false |

## Slot

| name   | 说明              |
|--------|-----------------|
| —      | Dialog 的内容      |
| title  | Dialog 标题区的内容   |
| footer | Dialog 按钮操作区的内容 |

## Events

| 事件名称   | 说明                | 回调参数 |
|--------|-------------------|------|
| open   | Dialog 打开的回调      | —    |
| close  | Dialog 关闭的回调      | —    |
| closed | Dialog 关闭动画结束时的回调 | —    |

原文: <http://element-cn.eleme.io/#/zh-CN/component/dialog>

# Tooltip 文字提示

## Tooltip 文字提示

常用于展示鼠标 hover 时的提示信息。

### 基础用法

在这里我们提供 9 种不同方向的展示方式，可以通过以下完整示例来理解，选择你要的效果。



使用 `content` 属性来决定 `hover` 时的提示信息。由 `placement` 属性决定展示效果：  
 果：`placement` 属性值为： 方向-对齐位置；四个方  
 向：`top`、`left`、`right`、`bottom`；三种对齐位置：`start`，`end`，默认为空。  
 如 `placement="left-end"`，则提示信息出现在目标元素的左侧，且提示信息的底部与目标元素的底部对齐。

```

1. <div class="box">
2. <div class="top">
3. <el-tooltip class="item" effect="dark" content="Top Left 提示文字"
 placement="top-start">
4. <el-button>上左</el-button>
5. </el-tooltip>
6. <el-tooltip class="item" effect="dark" content="Top Center 提示文字"
 placement="top">
7. <el-button>上边</el-button>
8. </el-tooltip>
9. <el-tooltip class="item" effect="dark" content="Top Right 提示文字"
 placement="top-end">
10. <el-button>上右</el-button>

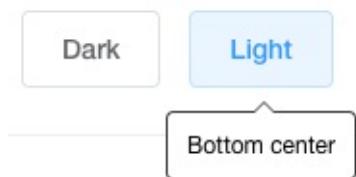
```

```
11. </el-tooltip>
12. </div>
13. <div class="left">
14. <el-tooltip class="item" effect="dark" content="Left Top 提示文字"
15. placement="left-start">
16. <el-button>左上</el-button>
17. </el-tooltip>
18. <el-tooltip class="item" effect="dark" content="Left Center 提示文字"
19. placement="left">
20. <el-button>左边</el-button>
21. </el-tooltip>
22. <el-tooltip class="item" effect="dark" content="Left Bottom 提示文字"
23. placement="left-end">
24. <el-button>左下</el-button>
25. </el-tooltip>
26. </div>
27. <div class="right">
28. <el-tooltip class="item" effect="dark" content="Right Top 提示文字"
29. placement="right-start">
30. <el-button>右上</el-button>
31. </el-tooltip>
32. <el-tooltip class="item" effect="dark" content="Right Center 提示文字"
33. placement="right">
34. <el-button>右边</el-button>
35. </el-tooltip>
36. <el-tooltip class="item" effect="dark" content="Right Bottom 提示文字"
37. placement="right-end">
38. <el-button>右下</el-button>
39. </el-tooltip>
40. </div>
41. <div class="bottom">
42. <el-tooltip class="item" effect="dark" content="Bottom Left 提示文字"
43. placement="bottom-start">
44. <el-button>下左</el-button>
45. </el-tooltip>
46. <el-tooltip class="item" effect="dark" content="Bottom Center 提示文字"
47. placement="bottom">
48. <el-button>下边</el-button>
49. </el-tooltip>
50. <el-tooltip class="item" effect="dark" content="Bottom Right 提示文字"
51. placement="bottom-end">
```

```
44. <el-button>下右</el-button>
45. </el-tooltip>
46. </div>
47. </div>
48.
49. <style>
50. .box {
51. width: 400px;
52.
53. .top {
54. text-align: center;
55. }
56.
57. .left {
58. float: left;
59. width: 60px;
60. }
61.
62. .right {
63. float: right;
64. width: 60px;
65. }
66.
67. .bottom {
68. clear: both;
69. text-align: center;
70. }
71.
72. .item {
73. margin: 4px;
74. }
75.
76. .left .el-tooltip__popper,
77. .right .el-tooltip__popper {
78. padding: 8px 10px;
79. }
80. }
81. </style>
```

## 主题

Tooltip 组件提供了两个不同的主题：`dark` 和 `light`。



通过设置 `effect` 属性来改变主题，默認為 `dark`。

```

1. <el-tooltip content="Top center" placement="top">
2. <el-button>Dark</el-button>
3. </el-tooltip>
4. <el-tooltip content="Bottom center" placement="bottom" effect="light">
5. <el-button>Light</el-button>
6. </el-tooltip>

```

## 更多 Content

展示多行文本或者是设置文本内容的格式



用具名 slot 分发 `content`，替代 `tooltip` 中的 `content` 属性。

```

1. <el-tooltip placement="top">
2. <div slot="content">多行信息
第二行信息</div>
3. <el-button>Top center</el-button>
4. </el-tooltip>

```

## 高级扩展

除了这些基本设置外，还有一些属性可以让使用者更好的定制自己的效果：

`transition` 属性可以定制显隐的动画效果，默認為 `fade-in-linear`。如果需要关闭 `tooltip` 功能，`disabled` 属性可以满足这个需求，它接受一个 `Boolean`，设置为 `true` 即可。

事实上，这是基于 `Vue-popper` 的扩展，你可以自定义任意 `Vue-popper` 中允许定义的字段。当然 `Tooltip` 组件实际上十分强大，文末的API文档会做一一说明。

点击关闭 tooltip 功能

点击关闭 tooltip 功能

```

1. <template>
2. <el-tooltip :disabled="disabled" content="点击关闭 tooltip 功能"
 placement="bottom" effect="light">
3. <el-button @click="disabled = !disabled">点击{{disabled ? '开启' : '关闭'}}
 tooltip 功能</el-button>
4. </el-tooltip>
5. </template>

```

## 显示代码

tooltip 内不支持 `router-link` 组件，请使用 `vm.$router.push` 代替。

tooltip 内不支持 disabled form 元素，参考[MDN](#)，请在 disabled form 元素外层添加一层包裹元素。

## Attributes

| 参数                                             | 说明                                                            | 类型      | 可选值                                                                                                                               | 默认值               |
|------------------------------------------------|---------------------------------------------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------|-------------------|
| effect                                         | 默认提供的主题                                                       | String  | dark/light                                                                                                                        | dark              |
| content                                        | 显示的内容，也可<br>以通过<br><code>slot#content</code><br>传入 DOM        | String  | —                                                                                                                                 | —                 |
| placement                                      | Tooltip 的出现<br>位置                                             | String  | top/top-start/top-<br>end/bottom/bottom-<br>start/bottom-<br>end/left/left-<br>start/left-<br>end/right/right-<br>start/right-end | bottom            |
| value( <code>v-</code><br><code>model</code> ) | 状态是否可见                                                        | Boolean | —                                                                                                                                 | false             |
| disabled                                       | Tooltip 是否可<br>用                                              | Boolean | —                                                                                                                                 | false             |
| offset                                         | 出现位置的偏移量                                                      | Number  | —                                                                                                                                 | 0                 |
| transition                                     | 定义渐变动画                                                        | String  | —                                                                                                                                 | el-fade-in-linear |
| visible-<br>arrow                              | 是否显示<br>Tooltip 箭头，<br>更多参数可见 <a href="#">Vue-<br/>popper</a> | Boolean | —                                                                                                                                 | true              |
|                                                |                                                               |         |                                                                                                                                   | {                 |

|                |                                                     |         |                              |                                                     |
|----------------|-----------------------------------------------------|---------|------------------------------|-----------------------------------------------------|
| popper-options | <code>popper.js</code> 的参数                          | Object  | 参考 <code>popper.js</code> 文档 | boundariesElement: 'body', gpuAcceleration: false } |
| open-delay     | 延迟出现, 单位毫秒                                          | Number  | -                            | 0                                                   |
| manual         | 手动控制模式, 设置为 true 后, mouseenter 和 mouseleave 事件将不会生效 | Boolean | -                            | false                                               |
| popper-class   | 为 Tooltip 的 popper 添加类名                             | String  | -                            | -                                                   |
| enterable      | 鼠标是否可进入到 tooltip 中                                  | Boolean | -                            | true                                                |
| hide-after     | Tooltip 出现后自动隐藏延时, 单位毫秒, 为 0 则不会自动隐藏                | number  | -                            | 0                                                   |

原文: <http://element-cn.eleme.io/#/zh-CN/component/tooltip>

# Popover 弹出框

## Popover 弹出框

### 基础用法

Popover 的属性与 Tooltip 很类似，它们都是基于 [Vue-popper](#) 开发的，因此对于重复属性，请参考 Tooltip 的文档，在此文档中不做详尽解释。



`trigger` 属性用于设置何时触发 Popover，支持四种触发方式：`hover`，`click`，`focus` 和 `manual`。对于触发 Popover 的元素，有两种写法：使用 `slot="reference"` 的具名插槽，或使用自定义指令 `v-popover` 指向 Popover 的索引 `ref`。

```

1. <template>
2. <el-popover
3. placement="top-start"
4. title="标题"
5. width="200"
6. trigger="hover"
7. content="这是一段内容, 这是一段内容, 这是一段内容, 这是一段内容。">
8. <el-button slot="reference">hover 激活</el-button>
9. </el-popover>
10.
11. <el-popover
12. placement="bottom"
13. title="标题"
14. width="200"
15. trigger="click"
16. content="这是一段内容, 这是一段内容, 这是一段内容, 这是一段内容。">
17. <el-button slot="reference">click 激活</el-button>
18. </el-popover>
19.
```

```
20. <el-popover
21. ref="popover"
22. placement="right"
23. title="标题"
24. width="200"
25. trigger="focus"
26. content="这是一段内容,这是一段内容,这是一段内容,这是一段内容。">
27. </el-popover>
28. <el-button v-popover:popover>focus 激活</el-button>
29.
30. <el-popover
31. placement="bottom"
32. title="标题"
33. width="200"
34. trigger="manual"
35. content="这是一段内容,这是一段内容,这是一段内容,这是一段内容。"
36. v-model="visible">
37. <el-button slot="reference" @click="visible = !visible">手动激活</el-button>
38. </el-popover>
39. </template>
40.
41. <script>
42. export default {
43. data() {
44. return {
45. visible: false
46. };
47. }
48. };
49. </script>
```

## 嵌套信息

可以在 Popover 中嵌套多种类型信息，以下为嵌套表格的例子。



## 利用分发取代 `content` 属性

```

1. <el-popover
2. placement="right"
3. width="400"
4. trigger="click">
5. <el-table :data="gridData">
6. <el-table-column width="150" property="date" label="日期"></el-table-
 column>
7. <el-table-column width="100" property="name" label="姓名"></el-table-
 column>
8. <el-table-column width="300" property="address" label="地址"></el-table-
 column>
9. </el-table>
10. <el-button slot="reference">click 激活</el-button>
11. </el-popover>
12.
13. <script>
14. export default {
15. data() {
16. return {
17. gridData: [
18. date: '2016-05-02',
19. name: '王小虎',
20. address: '上海市普陀区金沙江路 1518 弄'
21.],
22. date: '2016-05-04',
23. name: '王小虎',
24. address: '上海市普陀区金沙江路 1518 弄'
25.],
26. }
27. }
28. }
29. </script>
30.
```

```

26. date: '2016-05-01',
27. name: '王小虎',
28. address: '上海市普陀区金沙江路 1518 弄'
29. }, {
30. date: '2016-05-03',
31. name: '王小虎',
32. address: '上海市普陀区金沙江路 1518 弄'
33. }]
34.];
35. }
36. };
37. </script>

```

## 嵌套操作

当然，你还可以嵌套操作，这相比 Dialog 更为轻量：



```

1. <el-popover
2. placement="top"
3. width="160"
4. v-model="visible2">
5. <p>这是一段内容这是一段内容确定删除吗？</p>
6. <div style="text-align: right; margin: 0">
7. <el-button size="mini" type="text" @click="visible2 = false">取消</el-
8. button>
9. <el-button type="primary" size="mini" @click="visible2 = false">确定</el-
10. button>
11. </div>
12. <el-button slot="reference">删除</el-button>
13. </el-popover>
14. <script>
15. export default {

```

```

15. data() {
16. return {
17. visible2: false,
18. };
19. }
20. }
21. </script>

```

## Attributes

| 参数             | 说明                                                | 类型             | 可选值                                                                                                       | 默认值                                                                            |
|----------------|---------------------------------------------------|----------------|-----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| trigger        | 触发方式                                              | String         | click/focus-hover/manual                                                                                  | click                                                                          |
| title          | 标题                                                | String         | —                                                                                                         | —                                                                              |
| content        | 显示的内容，也可以通过 slot 传入 DOM                           | String         | —                                                                                                         | —                                                                              |
| width          | 宽度                                                | String, Number | —                                                                                                         | 最小宽度 150px                                                                     |
| placement      | 出现位置                                              | String         | top/top-start/top-end/bottom/bottom-start/bottom-end/left/left-start/left-end/right/right-start/right-end | bottom                                                                         |
| disabled       | Popover 是否可用                                      | Boolean        | —                                                                                                         | false                                                                          |
| value(v-model) | 状态是否可见                                            | Boolean        | —                                                                                                         | false                                                                          |
| offset         | 出现位置的偏移量                                          | Number         | —                                                                                                         | 0                                                                              |
| transition     | 定义渐变动画                                            | String         | —                                                                                                         | fade-in-linear                                                                 |
| visible-arrow  | 是否显示 Tooltip 箭头，更多参数可见 <a href="#">Vue-popper</a> | Boolean        | —                                                                                                         | true                                                                           |
| popper-options | popper.js 的参数                                     | Object         | 参考 <a href="#">popper.js 文档</a>                                                                           | <pre>       boundariesElement: 'body',       gpuAcceleration: false     </pre> |
| popper-class   | 为 popper 添加类名                                     | String         | —                                                                                                         | —                                                                              |
| open-delay     | 触发方式为 hover 时的显示延迟，单位为毫秒                          | Number         | —                                                                                                         | —                                                                              |

## Slot

| 参数        | 说明                     |
|-----------|------------------------|
| -         | Popover 内嵌 HTML 文本     |
| reference | 触发 Popover 显示的 HTML 元素 |

## Events

| 事件名称        | 说明          | 回调参数 |
|-------------|-------------|------|
| show        | 显示时触发       | -    |
| after-enter | 显示动画播放完毕后触发 | -    |
| hide        | 隐藏时触发       | -    |
| after-leave | 隐藏动画播放完毕后触发 | -    |

原文: <http://element-cn.eleme.io/#/zh-CN/component/popover>

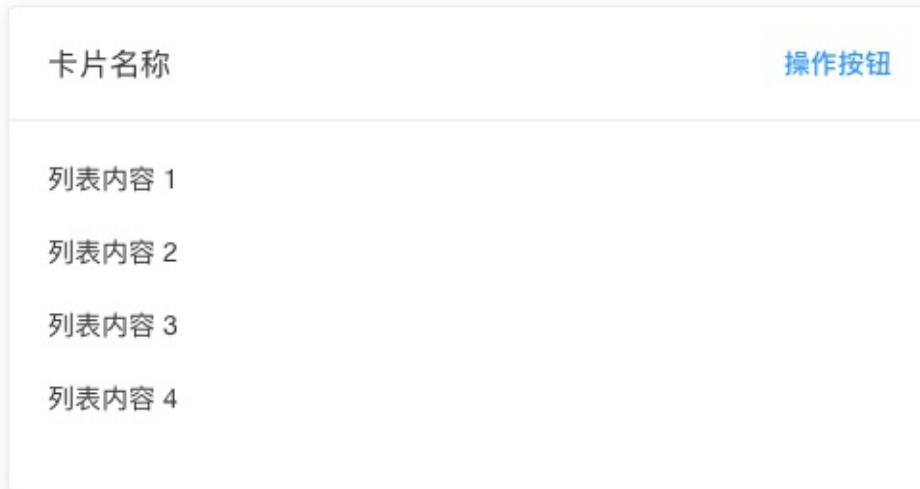
# Card 卡片

## Card 卡片

将信息聚合在卡片容器中展示。

### 基础用法

包含标题，内容和操作。



Card 组件包括 `header` 和 `body` 部分，`header` 部分需要有显式具名 slot 分发，同时也是可选的。

```

1. <el-card class="box-card">
2. <div slot="header" class="clearfix">
3. 卡片名称
4. <el-button style="float: right; padding: 3px 0" type="text">操作按钮</el-
 button>
5. </div>
6. <div v-for="o in 4" :key="o" class="text item">
7. {{'列表内容 ' + o }}
8. </div>
9. </el-card>
10.
11. <style>
12. .text {
13. font-size: 14px;
14. }

```

```
15. .item {
16. margin-bottom: 18px;
17. }
18.
19. .clearfix:before,
20. .clearfix:after {
21. display: table;
22. content: "";
23. }
24. .clearfix:after {
25. clear: both
26. }
27.
28.
29. .box-card {
30. width: 480px;
31. }
32. </style>
```

## 简单卡片

卡片可以只有内容区域。

列表内容 1

列表内容 2

列表内容 3

列表内容 4

```
1. <el-card class="box-card">
2. <div v-for="o in 4" :key="o" class="text item">
3. {{'列表内容 ' + o }}
4. </div>
5. </el-card>
6.
7. <style>
8. .text {
9. font-size: 14px;
```

```

10. }
11.
12. .item {
13. padding: 18px 0;
14. }
15.
16. .box-card {
17. width: 480px;
18. }
19. </style>

```

## 带图片

可配置定义更丰富的内容展示。



好吃的汉堡

2018-08-14 07:20

[操作按钮](#)



好吃的汉堡

2018-08-14 07:20

[操作按钮](#)

配置 `body-style` 属性来自定义 `body` 部分的 `style`，我们还使用了布局组件。

```

1. <el-row>
2. <el-col :span="8" v-for="(o, index) in 2" :key="o" :offset="index > 0 ? 2 :
 0">
3. <el-card :body-style="{ padding: '0px' }">
4.
5. <div style="padding: 14px;">
6. 好吃的汉堡
7. <div class="bottom clearfix">
8. <time class="time">{{ currentDate }}</time>

```

```
9. <el-button type="text" class="button">操作按钮</el-button>
10. </div>
11. </div>
12. </el-card>
13. </el-col>
14. </el-row>
15.
16. <style>
17. .time {
18. font-size: 13px;
19. color: #999;
20. }
21.
22. .bottom {
23. margin-top: 13px;
24. line-height: 12px;
25. }
26.
27. .button {
28. padding: 0;
29. float: right;
30. }
31.
32. .image {
33. width: 100%;
34. display: block;
35. }
36.
37. .clearfix:before,
38. .clearfix:after {
39. display: table;
40. content: "";
41. }
42.
43. .clearfix:after {
44. clear: both
45. }
46. </style>
47.
48. <script>
49. export default {
50. data() {
```

```

51. return {
52. currentDate: new Date()
53. };
54. }
55. }
56. </script>

```

## 卡片阴影

可对阴影的显示进行配置。

总是显示

鼠标悬浮时显示

从不显示

通过 `shadow` 属性设置卡片阴影出现的时机： `always`、`hover` 或 `never`。

```

1. <el-row :gutter="12">
2. <el-col :span="8">
3. <el-card shadow="always">
4. 总是显示
5. </el-card>
6. </el-col>
7. <el-col :span="8">
8. <el-card shadow="hover">
9. 鼠标悬浮时显示
10. </el-card>
11. </el-col>
12. <el-col :span="8">
13. <el-card shadow="never">
14. 从不显示
15. </el-card>
16. </el-col>
17. </el-row>

```

## Attributes

| 参数         | 说明                                                            | 类型     | 可选值 | 默认值                 |
|------------|---------------------------------------------------------------|--------|-----|---------------------|
| header     | 设置 <code>header</code> ，也可以通过 <code>slot#header</code> 传入 DOM | string | -   | -                   |
| body-style | 设置 <code>body</code> 的样式                                      | object | -   | { padding: '20px' } |

|        |          |        |                           |        |
|--------|----------|--------|---------------------------|--------|
| shadow | 设置阴影显示时机 | string | always / hover<br>/ never | always |
|--------|----------|--------|---------------------------|--------|

原文: <http://element-cn.eleme.io/#/zh-CN/component/card>

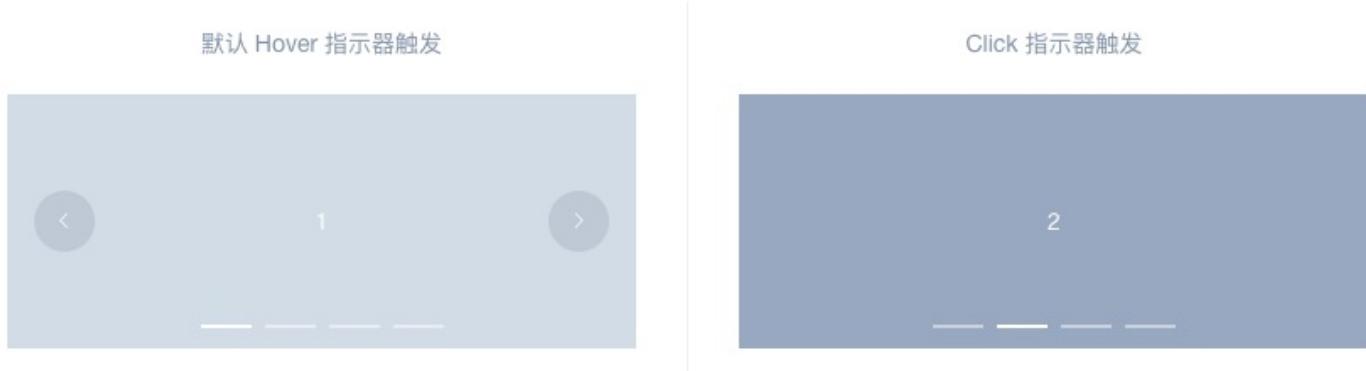
# Carousel 走马灯

## Carousel 走马灯

在有限空间内，循环播放同一类型的图片、文字等内容

### 基础用法

适用广泛的基础用法



结合使用 `el-carousel` 和 `el-carousel-item` 标签就得到了一个走马灯。幻灯片的内容是任意的，需要放在 `el-carousel-item` 标签中。默认情况下，在鼠标 hover 底部的指示器时就会触发切换。通过设置 `trigger` 属性为 `click`，可以达到点击触发的效果。

```

1. <template>
2. <div class="block">
3. 默认 Hover 指示器触发
4. <el-carousel height="150px">
5. <el-carousel-item v-for="item in 4" :key="item">
6. <h3>{{ item }}</h3>
7. </el-carousel-item>
8. </el-carousel>
9. </div>
10. <div class="block">
11. Click 指示器触发
12. <el-carousel trigger="click" height="150px">
13. <el-carousel-item v-for="item in 4" :key="item">
14. <h3>{{ item }}</h3>
15. </el-carousel-item>
16. </el-carousel>
17. </div>

```

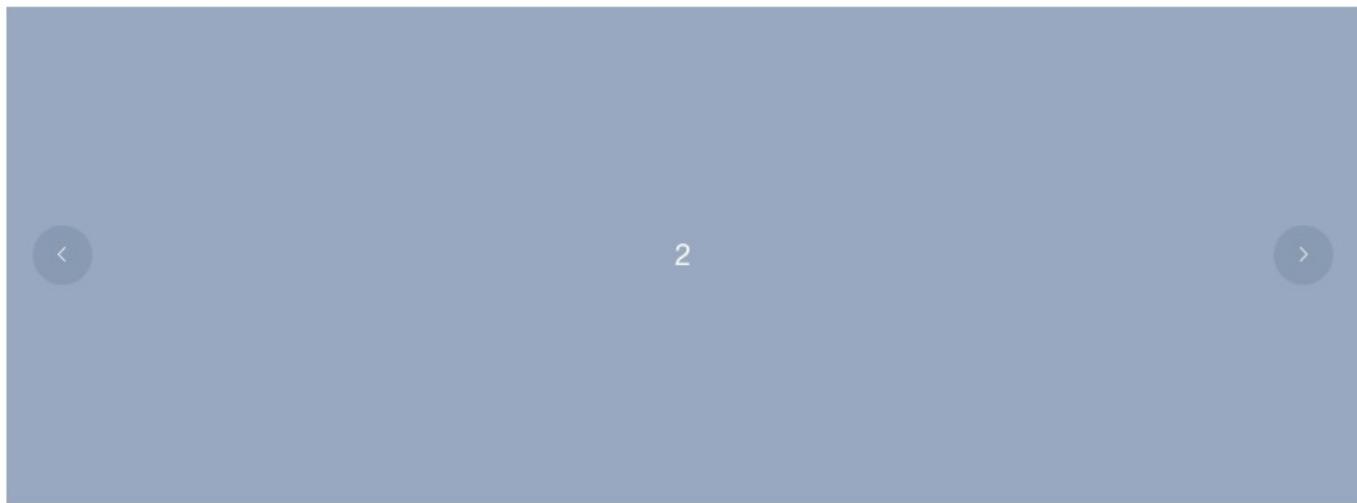
```

18. </template>
19.
20. <style>
21. .el-carousel__item h3 {
22. color: #475669;
23. font-size: 14px;
24. opacity: 0.75;
25. line-height: 150px;
26. margin: 0;
27. }
28.
29. .el-carousel__item:nth-child(2n) {
30. background-color: #99a9bf;
31. }
32.
33. .el-carousel__item:nth-child(2n+1) {
34. background-color: #d3dce6;
35. }
36. </style>

```

## 指示器

可以将指示器的显示位置设置在容器外部



`indicator-position` 属性定义了指示器的位置。默认情况下，它会显示在走马灯内部，设置为 `outside` 则会显示在外部；设置为 `none` 则不会显示指示器。

```

1. <template>
2. <el-carousel indicator-position="outside">

```

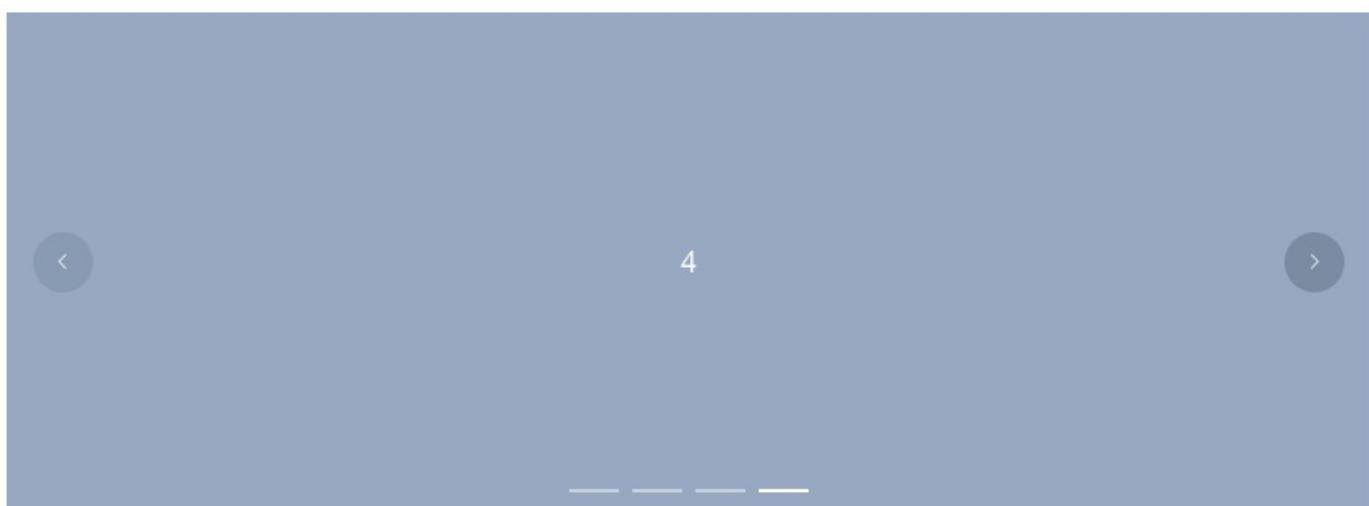
```

3. <el-carousel-item v-for="item in 4" :key="item">
4. <h3>{{ item }}</h3>
5. </el-carousel-item>
6. </el-carousel>
7. </template>
8.
9. <style>
10. .el-carousel__item h3 {
11. color: #475669;
12. font-size: 18px;
13. opacity: 0.75;
14. line-height: 300px;
15. margin: 0;
16. }
17.
18. .el-carousel__item:nth-child(2n) {
19. background-color: #99a9bf;
20. }
21.
22. .el-carousel__item:nth-child(2n+1) {
23. background-color: #d3dce6;
24. }
25. </style>

```

## 切换箭头

可以设置切换箭头的显示时机



`arrow` 属性定义了切换箭头的显示时机。默认情况下，切换箭头只有在鼠标 hover 到走马灯上时才会显示；若将 `arrow` 设置为 `always`，则会一直显示；设置为 `never`，则会一直隐藏。

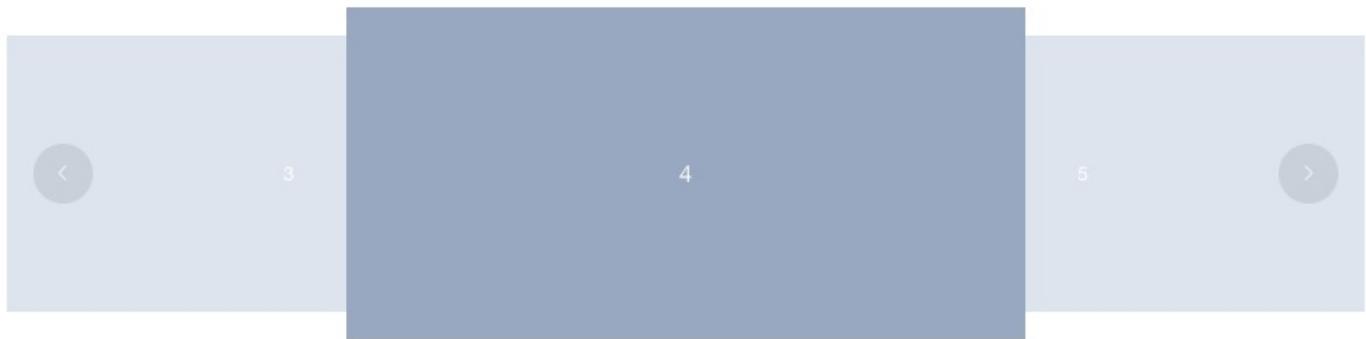
```

1. <template>
2. <el-carousel :interval="5000" arrow="always">
3. <el-carousel-item v-for="item in 4" :key="item">
4. <h3>{{ item }}</h3>
5. </el-carousel-item>
6. </el-carousel>
7. </template>
8.
9. <style>
10. .el-carousel__item h3 {
11. color: #475669;
12. font-size: 18px;
13. opacity: 0.75;
14. line-height: 300px;
15. margin: 0;
16. }
17.
18. .el-carousel__item:nth-child(2n) {
19. background-color: #99a9bf;
20. }
21.
22. .el-carousel__item:nth-child(2n+1) {
23. background-color: #d3dce6;
24. }
25. </style>

```

## 卡片化

当页面宽度方向空间空余，但高度方向空间匮乏时，可使用卡片风格



将 `type` 属性设置为 `card` 即可启用卡片模式。从交互上来说，卡片模式和一般模式的最大区别在于，可以通过直接点击两侧的幻灯片进行切换。

```

1. <template>
2. <el-carousel :interval="4000" type="card" height="200px">
3. <el-carousel-item v-for="item in 6" :key="item">
4. <h3>{{ item }}</h3>
5. </el-carousel-item>
6. </el-carousel>
7. </template>
8.
9. <style>
10. .el-carousel__item h3 {
11. color: #475669;
12. font-size: 14px;
13. opacity: 0.75;
14. line-height: 200px;
15. margin: 0;
16. }
17.
18. .el-carousel__item:nth-child(2n) {
19. background-color: #99a9bf;
20. }
21.
22. .el-carousel__item:nth-child(2n+1) {
23. background-color: #d3dce6;
24. }
25. </style>

```

## Carousel Attributes

| 参数                 | 说明                   | 类型      | 可选值                | 默认值   |
|--------------------|----------------------|---------|--------------------|-------|
| height             | 走马灯的高度               | string  | —                  | —     |
| initial-index      | 初始状态激活的幻灯片的索引，从 0 开始 | number  | —                  | 0     |
| trigger            | 指示器的触发方式             | string  | click              | —     |
| autoplay           | 是否自动切换               | boolean | —                  | true  |
| interval           | 自动切换的时间间隔，单位为毫秒      | number  | —                  | 3000  |
| indicator-position | 指示器的位置               | string  | outside/none       | —     |
| arrow              | 切换箭头的显示时机            | string  | always-hover/never | hover |
| type               | 走马灯的类型               | string  | card               | —     |

## Carousel Events

| 事件名称   | 说明       | 回调参数                |
|--------|----------|---------------------|
| change | 幻灯片切换时触发 | 目前激活的幻灯片的索引，原幻灯片的索引 |

## Carousel Methods

| 方法名           | 说明        | 参数                                            |
|---------------|-----------|-----------------------------------------------|
| setActiveItem | 手动切换幻灯片   | 需要切换的幻灯片的索引，从 0 开始；或相应的 <code>name</code> 属性值 |
| prev          | 切换至上一张幻灯片 | —                                             |
| next          | 切换至下一张幻灯片 | —                                             |

## Carousel-Item Attributes

| 参数    | 说明                                        | 类型     | 可选值 | 默认值 |
|-------|-------------------------------------------|--------|-----|-----|
| name  | 幻灯片的名字，可用作 <code>setActiveItem</code> 的参数 | string | —   | —   |
| label | 该幻灯片所对应指示器的文本                             | string | —   | —   |

原文：<http://element-cn.eleme.io/#/zh-CN/component/carousel>

# Collapse 折叠面板

## Collapse 折叠面板

通过折叠面板收纳内容区域

### 基础用法

可同时展开多个面板，面板之间不影响

#### 一致性 Consistency

与现实生活一致：与现实生活的流程、逻辑保持一致，遵循用户习惯的语言和概念；  
在界面中一致：所有的元素和结构需保持一致，比如：设计样式、图标和文本、元素的位置等。

#### 反馈 Feedback

#### 效率 Efficiency

#### 可控 Controllability

```

1. <el-collapse v-model="activeNames" @change="handleChange">
2. <el-collapse-item title="一致性 Consistency" name="1">
3. <div>与现实生活一致：与现实生活的流程、逻辑保持一致，遵循用户习惯的语言和概念；</div>
4. <div>在界面中一致：所有的元素和结构需保持一致，比如：设计样式、图标和文本、元素的位置
等。</div>
5. </el-collapse-item>
6. <el-collapse-item title="反馈 Feedback" name="2">
7. <div>控制反馈：通过界面样式和交互动效让用户可以清晰的感知自己的操作；</div>
8. <div>页面反馈：操作后，通过页面元素的变化清晰地展现当前状态。</div>
9. </el-collapse-item>
10. <el-collapse-item title="效率 Efficiency" name="3">
11. <div>简化流程：设计简洁直观的操作流程；</div>
12. <div>清晰明确：语言表达清晰且表意明确，让用户快速理解进而作出决策；</div>
13. <div>帮助用户识别：界面简单直白，让用户快速识别而非回忆，减少用户记忆负担。</div>
14. </el-collapse-item>
15. <el-collapse-item title="可控 Controllability" name="4">
16. <div>用户决策：根据场景可给予用户操作建议或安全提示，但不能代替用户进行决策；</div>
17. <div>结果可控：用户可以自由的进行操作，包括撤销、回退和终止当前操作等。</div>
18. </el-collapse-item>
```

```

19. </el-collapse>
20. <script>
21. export default {
22. data() {
23. return {
24. activeNames: ['1']
25. };
26. },
27. methods: {
28. handleChange(val) {
29. console.log(val);
30. }
31. }
32. }
33. </script>

```

## 手风琴效果

每次只能展开一个面板

### 一致性 Consistency

与现实生活一致：与现实生活的流程、逻辑保持一致，遵循用户习惯的语言和概念；  
在界面中一致：所有的元素和结构需保持一致，比如：设计样式、图标和文本、元素的位置等。

### 反馈 Feedback

### 效率 Efficiency

### 可控 Controllability

通过 `accordion` 属性来设置是否以手风琴模式显示。

```

1. <el-collapse v-model="activeName" accordion>
2. <el-collapse-item title="一致性 Consistency" name="1">
3. <div>与现实生活一致：与现实生活的流程、逻辑保持一致，遵循用户习惯的语言和概念；</div>
4. <div>在界面中一致：所有的元素和结构需保持一致，比如：设计样式、图标和文本、元素的位置
 等。</div>
5. </el-collapse-item>
6. <el-collapse-item title="反馈 Feedback" name="2">
7. <div>控制反馈：通过界面样式和交互动效让用户可以清晰的感知自己的操作；</div>
8. <div>页面反馈：操作后，通过页面元素的变化清晰地展现当前状态。</div>

```

```

9. </el-collapse-item>
10. <el-collapse-item title="效率 Efficiency" name="3">
11. <div>简化流程：设计简洁直观的操作流程；</div>
12. <div>清晰明确：语言表达清晰且表意明确，让用户快速理解进而作出决策；</div>
13. <div>帮助用户识别：界面简单直白，让用户快速识别而非回忆，减少用户记忆负担。</div>
14. </el-collapse-item>
15. <el-collapse-item title="可控 Controllability" name="4">
16. <div>用户决策：根据场景可给予用户操作建议或安全提示，但不能代替用户进行决策；</div>
17. <div>结果可控：用户可以自由的进行操作，包括撤销、回退和终止当前操作等。</div>
18. </el-collapse-item>
19. </el-collapse>
20. <script>
21. export default {
22. data() {
23. return {
24. activeName: '1'
25. };
26. }
27. }
28. </script>

```

## 自定义面板标题

除了可以通过 `title` 属性以外，还可以通过具名 `slot` 来实现自定义面板的标题内容，以实现增加图标等效果。

一致性 Consistency 

反馈 Feedback 

效率 Efficiency 

可控 Controllability 

```

1. <el-collapse accordion>
2. <el-collapse-item>
3. <template slot="title">
4. 一致性 Consistency<i class="header-icon el-icon-info"></i>
5. </template>
6. <div>与现实生活一致：与现实生活的流程、逻辑保持一致，遵循用户习惯的语言和概念；</div>
7. <div>在界面中一致：所有的元素和结构需保持一致，比如：设计样式、图标和文本、元素的位置

```

```

等。</div>
8. </el-collapse-item>
9. <el-collapse-item title="反馈 Feedback">
10. <div>控制反馈：通过界面样式和交互动效让用户可以清晰的感知自己的操作；</div>
11. <div>页面反馈：操作后，通过页面元素的变化清晰地展现当前状态。</div>
12. </el-collapse-item>
13. <el-collapse-item title="效率 Efficiency">
14. <div>简化流程：设计简洁直观的操作流程；</div>
15. <div>清晰明确：语言表达清晰且表意明确，让用户快速理解进而作出决策；</div>
16. <div>帮助用户识别：界面简单直白，让用户快速识别而非回忆，减少用户记忆负担。</div>
17. </el-collapse-item>
18. <el-collapse-item title="可控 Controllability">
19. <div>用户决策：根据场景可给予用户操作建议或安全提示，但不能代替用户进行决策；</div>
20. <div>结果可控：用户可以自由的进行操作，包括撤销、回退和终止当前操作等。</div>
21. </el-collapse-item>
22. </el-collapse>

```

## Collapse Attributes

| 参数        | 说明                                                                      | 类型           | 可选值 | 默认值   |
|-----------|-------------------------------------------------------------------------|--------------|-----|-------|
| accordion | 是否手风琴模式                                                                 | boolean      | -   | false |
| value     | 当前激活的面板(如果是手风琴模式，绑定值类型需要为 <code>string</code> ，否则为 <code>array</code> ) | string/array | -   | -     |

## Collapse Events

| 事件名称   | 说明                                                                                                 | 回调参数                                |
|--------|----------------------------------------------------------------------------------------------------|-------------------------------------|
| change | 当前激活面板改变时触发(如果是手风琴模式，参数 <code>activeNames</code> 类型为 <code>string</code> ，否则为 <code>array</code> ) | ( <code>activeNames: array</code> ) |

## Collapse Item Attributes

| 参数    | 说明    | 类型            | 可选值 | 默认值 |
|-------|-------|---------------|-----|-----|
| name  | 唯一标志符 | string/number | -   | -   |
| title | 面板标题  | string        | -   | -   |

原文：<http://element-cn.eleme.io/#/zh-CN/component/collapse>

