

10-ElementPlus组件库

一、基本介绍

官方文档: <https://element-plus.org/zh-CN/>

1. 什么是组件

一个可复用的Vue实例、一段独立的UI界面, 有自己的 HTML+CSS+JavaScript, 代码上体现在一个独立的 vue 文件

```
1 <script setup></script>
2 <template></template>
3 <style lang="scss" scoped></style>
```

2. 什么是组件库

很多个组件的集合

3. 什么是ElementPlus组件库

ElementPlus是由 饿了么 团队开发的。是一个基于Vue3开发PC端开源UI组件库, 专为开发者提供了一套优雅而灵活的UI组件。它包含了各种常用的Web组件, 如按钮、表单、导航等; 以及高级组件, 如日期选择器、对话框等。ElementPlus默认语言环境是中文, 但可以通过指定locale属性来实现其他语言环境, 如英文。总之, ElementPlus提供了丰富的PC组件, 有效地降低了开发难度。

- 组件分类
 - 基础组件
 - 表单组件
 - 数据展示
 - 导航组件
 - 反馈组件
 - 其他组件

4. 说明

ElementPlus的组件很多, 我们会重点学其中的典型和常用组件, 掌握组件的使用方法和文档的阅读技巧, 举一反三、触类旁通

5. 学习目标

掌握ElementPlus常用组件的使用, 达到学"1"个组件通N个组件, 学"1"个组件库通N个组件库。因为所有组件使用时关注点和组件库的使用方法都大同小异。

二、快速开始

1. 安装

```
1 # npm
2 npm i element-plus
3
4 # yarn
5 yarn add element-plus
6
7 # pnpm
8 pnpm i element-plus
```

2. 完整引入

```
1 // 导入 ElementPlus 组件
2 import ElementPlus from 'element-plus'
3 // 导入组件样式
4 import 'element-plus/dist/index.css'
5
6 // 全局注册所有组件
7 app.use(ElementPlus)
```

3. 测试

以 Button 组件为例, 在 App.vue 测试

```
1 <el-button type="primary">Primary</el-button>
2 <el-button type="success">Success</el-button>
```

三、学习ElementPlus常用组件

1. Form表单组件

<https://element-plus.org/zh-CN/component/form.html>

1.1 介绍

表单包含 输入框，单选框，下拉选择，多选框 等用户输入的组件。使用表单可以收集、验证和提交数据。

Form 组件已经从 2.x 的 Float 布局升级为 Flex 布局。

Activity name

Activity zone

Activity time -

Instant delivery ☐

Activity type ☐ Online activities ☐ Promotion activities
☐ Offline activities ☐ Simple brand exposure

Resources ☐ Sponsor ☐ Venue

Activity form

Create Cancel

1.2 基本使用

```
1 <script setup>
2   import { reactive } from 'vue'
3
4   // 表单对象
5   const form = reactive({
6     name: '',
7     region: '',
8     date1: '',
9     date2: '',
10    delivery: false,
11    type: [],
12    resource: '',
13    desc: ''
14  })
15
16  // 提交
17  const onSubmit = () => {
18    console.log('submit!')
19  }
20 </script>
21 <template>
22   <!--
23     model: 表单数据对象
24     label-width: 标签的宽度, 可使用 auto
25     style: 行内样式
26   -->
27   <el-form
28     :model="form"
```

```
29     label-width="auto"
30     style="max-width: 600px">
31     <!--
32         label: 标签文本
33     -->
34     <el-form-item label="Activity name">
35         <!--
36             v-model: 双向绑定, 收集或设置表单数据
37         -->
38         <el-input v-model="form.name" />
39     </el-form-item>
40     <el-form-item label="Activity zone">
41         <!--
42             下拉列表
43             placeholder: 占位提示文本
44         -->
45         <el-select
46             v-model="form.region"
47             placeholder="please select your zone">
48             <!--
49                 选项
50                 label: 展示的文本
51                 value: 同步到v-model的值
52             -->
53             <el-option
54                 label="Zone one"
55                 value="shanghai" />
56             <el-option
57                 label="Zone two"
58                 value="beijing" />
59         </el-select>
60     </el-form-item>
61     <el-form-item label="Activity time">
62         <!--
63             列
64             span: 占据的宽度份数, 一行分为24份
65         -->
66         <el-col :span="11">
67             <!-- 日期选择器 -->
68             <el-date-picker
69                 v-model="form.date1"
70                 type="date"
71                 placeholder="Pick a date"
72                 style="width: 100%" />
73         </el-col>
```

```
76     <el-col
77       :span="2"
78       class="text-center">
79       <span class="text-gray-500">-</span>
80     </el-col>
81     <el-col :span="11">
82       <!-- 时间选择器 -->
83       <el-time-picker
84         v-model="form.date2"
85         placeholder="Pick a time"
86         style="width: 100%" />
87     </el-col>
88   </el-form-item>
89   <el-form-item label="Instant delivery">
90     <!-- 开关 -->
91     <el-switch v-model="form.delivery" />
92   </el-form-item>
93   <el-form-item label="Activity type">
94     <!-- 复选框组 -->
95     <el-checkbox-group v-model="form.type">
96       <el-checkbox
97         value="Online activities"
98         name="type">
99         Online activities
100     </el-checkbox>
101     <el-checkbox
102       value="Promotion activities"
103       name="type">
104       Promotion activities
105     </el-checkbox>
106     <el-checkbox
107       value="Offline activities"
108       name="type">
109       Offline activities
110     </el-checkbox>
111     <el-checkbox
112       value="Simple brand exposure"
113       name="type">
114       Simple brand exposure
115     </el-checkbox>
116   </el-checkbox-group>
117 </el-form-item>
118 <el-form-item label="Resources">
119   <!-- 单选框组 -->
120   <el-radio-group v-model="form.resource">
121     <el-radio value="Sponsor">Sponsor</el-radio>
122     <el-radio value="Venue">Venue</el-radio>
```

```

123     </el-radio-group>
124   </el-form-item>
125   <el-form-item label="Activity form">
126     <el-input
127       v-model="form.desc"
128       type="textarea" />
129   </el-form-item>
130   <el-form-item>
131     <!-- 提交按钮 -->
132     <el-button
133       type="primary"
134       @click="onSubmit"
135       >Create</el-button
136   >
137     <!-- 取消按钮 -->
138     <el-button>Cancel</el-button>
139   </el-form-item>
140 </el-form>
141 </template>

```

1.3 表单校验

Form 组件允许你验证用户的输入是否符合规范, 来帮助找到和纠正错误。

Form 组件提供了表单验证的功能, 只需为 `rules` 属性传入约定的验证规则, 并将 `form-item` 的 `prop` 属性设置为需要验证的特殊键值即可。

```

1 <script setup>
2   import { reactive, ref } from 'vue'
3
4   const formRef = ref(null)
5
6   // 表单对象
7   const form = reactive({
8     name: '',
9     region: '',
10    date1: '',
11    date2: '',
12    delivery: false,
13    type: [],
14    resource: '',
15    desc: ''
16  })
17
18  // 校验规则
19  const rules = {

```

```
20     name: [
21         {
22             required: true,
23             message: 'Please input Activity name',
24             trigger: 'blur'
25         },
26         { min: 3, max: 5, message: 'Length should be 3 to 5', trigger: 'blur' }
27     ],
28     region: [
29         {
30             required: true,
31             message: 'Please select Activity zone',
32             trigger: 'change'
33         }
34     ],
35     count: [
36         {
37             required: true,
38             message: 'Please select Activity count',
39             trigger: 'change'
40         }
41     ],
42     date1: [
43         {
44             type: 'date',
45             required: true,
46             message: 'Please pick a date',
47             trigger: 'change'
48         }
49     ],
50     date2: [
51         {
52             type: 'date',
53             required: true,
54             message: 'Please pick a time',
55             trigger: 'change'
56         }
57     ],
58     location: [
59         {
60             required: true,
61             message: 'Please select a location',
62             trigger: 'change'
63         }
64     ],
65     type: [
66         {
```

```

67         type: 'array',
68         required: true,
69         message: 'Please select at least one activity type',
70         trigger: 'change'
71     }
72 ],
73     resource: [
74         {
75             required: true,
76             message: 'Please select activity resource',
77             trigger: 'change'
78         }
79     ],
80     desc: [
81         { required: true, message: 'Please input activity form', trigger: 'blur'
82     }
83 }
84
85 // 提交
86 const submitForm = async () => {
87     // 对整个表单进行验证。
88     formRef.value.validate((valid) => {
89         if (valid) {
90             console.log('submit!')
91         } else {
92             console.log('error submit!')
93         }
94     })
95 }
96
97 // 重置
98 const resetForm = () => {
99     // 重置该表单项，将其值重置为初始值，并移除校验结果
100     formRef.value.resetFields()
101 }
102 </script>
103
104 <template>
105     <!--
106         rules: 校验规则
107         ref: 用来调用 Form 提供的方法
108     -->
109     <el-form
110
111         style="max-width: 600px"
112         :model="form"

```



```
113     :rules="rules"
114     ref="formRef"
115     label-width="auto"
116     status-icon>
117     <!--
118       prop: 校验的字段
119     -->
120     <el-form-item
121       label="Activity name"
122       prop="name">
123       <el-input v-model="form.name" />
124     </el-form-item>
125     <el-form-item
126       label="Activity zone"
127       prop="region">
128       <el-select
129         v-model="form.region"
130         placeholder="Activity zone">
131         <el-option
132           label="Zone one"
133           value="shanghai" />
134         <el-option
135           label="Zone two"
136           value="beijing" />
137       </el-select>
138     </el-form-item>
139     <el-form-item
140       label="Activity time"
141       required>
142     <el-col :span="11">
143       <el-form-item prop="date1">
144         <el-date-picker
145           v-model="form.date1"
146           type="date"
147           aria-label="Pick a date"
148           placeholder="Pick a date"
149           style="width: 100%" />
150       </el-form-item>
151     </el-col>
152     <el-col
153       class="text-center"
154       :span="2">
155       <span class="text-gray-500"></span>
156     </el-col>
157     <el-col :span="11">
158       <el-form-item prop="date2">
159         <el-time-picker
```

```
160         v-model="form.date2"
161         aria-label="Pick a time"
162         placeholder="Pick a time"
163         style="width: 100%" />
164     </el-form-item>
165 </el-col>
166 </el-form-item>
167 <el-form-item
168     label="Instant delivery"
169     prop="delivery">
170     <el-switch v-model="form.delivery" />
171 </el-form-item>
172 <el-form-item
173     label="Activity type"
174     prop="type">
175     <el-checkbox-group v-model="form.type">
176         <el-checkbox
177             value="Online activities"
178             name="type">
179             Online activities
180         </el-checkbox>
181         <el-checkbox
182             value="Promotion activities"
183             name="type">
184             Promotion activities
185         </el-checkbox>
186         <el-checkbox
187             value="Offline activities"
188             name="type">
189             Offline activities
190         </el-checkbox>
191         <el-checkbox
192             value="Simple brand exposure"
193             name="type">
194             Simple brand exposure
195         </el-checkbox>
196     </el-checkbox-group>
197 </el-form-item>
198 <el-form-item
199     label="Resources"
200     prop="resource">
201     <el-radio-group v-model="form.resource">
202         <el-radio value="Sponsorship">Sponsorship</el-radio>
203         <el-radio value="Venue">Venue</el-radio>
204     </el-radio-group>
205 </el-form-item>
206 <el-form-item
```

```

207     label="Activity form"
208     prop="desc">
209     <el-input
210       v-model="form.desc"
211       type="textarea" />
212   </el-form-item>
213   <el-form-item>
214     <el-button
215       type="primary"
216       @click="submitForm">
217       Create
218     </el-button>
219     <el-button @click="resetForm">Reset</el-button>
220   </el-form-item>
221 </el-form>
222 </template>

```

1.4 完成如下布局并校验

校验规则

- 用户名: 必填项、长度 3 - 10 个字符、失焦做校验
- 密码: 必填项、长度 6 - 15 个字符、失焦做校验
- 点击登录时做整体校验
- 点击取消按钮进行重置

完整代码

```

1 <script setup>
2   import { reactive, ref } from 'vue'
3
4   import { ElMessage } from 'element-plus'
5
6   // 登录表单
7   const loginForm = reactive({
8     username: '', // 用户名
9     password: '' // 密码

```

```
10  })
11
12  const formRef = ref(null)
13
14  // 校验规则
15  const loginFormRules = {
16    // 用户名
17    username: [
18      {
19        required: true,
20        message: '用户名不能为空',
21        trigger: 'blur'
22      },
23      {
24        min: 3,
25        max: 6,
26        message: '用户名长度在3-6之间',
27        trigger: 'blur'
28      }
29    ],
30    // 密码
31    password: [
32      {
33        required: true,
34        message: '密码不能为空',
35        trigger: 'blur'
36      },
37      {
38        min: 6,
39        max: 15,
40        message: '密码长度在6-15之间',
41        trigger: 'blur'
42      }
43    ]
44  }
45
46  // 登录
47  const onLogin = () => {
48    // 整体校验
49    formRef.value.validate((valid) => {
50      // 如果没通过校验, 给出错题提示并提前结束代码的执行
51      if (!valid) return ElMessage.error('用户名或密码不对')
52      console.log('ok')
53      // ...
54    })
55  }
56
```

```
57 // 取消
58 const onCancel = () => {
59   // 进行表单重置
60   formRef.value.resetFields()
61 }
62 </script>
63
64 <template>
65   <!-- 登录外层盒子 -->
66   <div class="login-box">
67     <!-- 表单组件 -->
68     <el-form
69       ref="formRef"
70       :model="loginForm"
71       :rules="loginFormRules">
72       <el-form-item prop="username">
73         <!--
74           prefix-icon: 在输入框的前面插入字体图标
75         -->
76         <el-input
77           v-model="loginForm.username"
78           placeholder="请填写用户名"
79           prefix-icon="User" />
80       </el-form-item>
81       <el-form-item prop="password">
82         <el-input
83           v-model="loginForm.password"
84           placeholder="请填写密码"
85           prefix-icon="Lock"
86           type="password"
87           show-password />
88       </el-form-item>
89       <el-form-item>
90         <el-button
91           type="primary"
92           @click="onLogin"
93           >登 录</el-button>
94         >
95         <el-button @click="onCancel">取 消</el-button>
96       </el-form-item>
97     </el-form>
98   </div>
99 </template>
100
101 <style lang="scss" scoped>
102   * {
103     margin: 0;
```

```

104   }
105   body {
106     background: #ddd;
107   }
108
109   .login-box {
110     display: flex;
111     justify-content: center;
112     align-items: center;
113     height: 100vh;
114     .el-form {
115       width: 500px;
116       padding: 40px 50px;
117       background: #fff;
118       .el-form-item {
119         margin: 20px 0;
120       }
121       .el-input {
122         height: 40px;
123       }
124     }
125   }
126 </style>

```

2. Card卡片组件

2.1 介绍

将信息聚合在卡片容器中展示

2.2 基础用法

卡片包含标题，内容以及操作区域。

Card 组件由 `header` `body` 和 `footer` 组成。`header` 和 `footer` 是可选的, 其内容取决于一个默认的 slot。

Card name
List item 1
List item 2
List item 3
List item 4
Footer content

2.3 示例代码

```

1
2 <template>
3   <el-card style="max-width: 480px">
4     <template #header>
5       <div class="card-header">
6         <span>Card name</span>
7       </div>
8     </template>
9     <template #default>
10      <p
11        v-for="i in 4"
12        :key="i"
13        class="text item">
14        {{ 'List item ' + i }}
15      </p>
16    </template>
17    <template #footer>Footer content</template>
18  </el-card>
19 </template>

```

3. Table表格组件

3.1 介绍

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。

3.2 基础用法

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。

Date	Name	Address
2016-05-03	Tom	No. 189, Grove St, Los Angeles
2016-05-02	Tom	No. 189, Grove St, Los Angeles
2016-05-04	Tom	No. 189, Grove St, Los Angeles
2016-05-01	Tom	No. 189, Grove St, Los Angeles

3.3 示例代码

```

1 <script setup>
2   const tableData = [
3     {
4       date: '2016-05-03',
5       name: 'Tom',

```

```
6     address: 'No. 189, Grove St, Los Angeles'
7   },
8   {
9     date: '2016-05-02',
10    name: 'Tom',
11    address: 'No. 189, Grove St, Los Angeles'
12  },
13  {
14    date: '2016-05-04',
15    name: 'Tom',
16    address: 'No. 189, Grove St, Los Angeles'
17  },
18  {
19    date: '2016-05-01',
20    name: 'Tom',
21    address: 'No. 189, Grove St, Los Angeles'
22  }
23 ]
```

```
24 </script>
```

```
25
26 <template>
```

```
27   <!--
```

```
28     data: 数据源
```

```
29     stripe: 带有斑马纹
```

```
30     border: 带边框
```

```
31   -->
```

```
32 <el-table
```

```
33   :data="tableData"
```

```
34   stripe
```

```
35   border
```

```
36   style="width: 100%">
```

```
37   <!--
```

```
38     table-column: 列
```

```
39     label: 列名
```

```
40     prop: 渲染的字段名
```

```
41     width: 列宽
```

```
42   -->
```

```
43 <el-table-column
```

```
44   prop="date"
```

```
45   label="Date"
```

```
46   width="180" />
```

```
47 <el-table-column
```

```
48   prop="name"
```

```
49   label="Name"
```

```
50   width="180" />
```

```
51 <el-table-column
```

```
52   prop="address"
```



```
53     label="Address" />
54   </el-table>
55 </template>
```

3.4 自定义列模版

自定义列的显示内容, 可组合其他组件使用。

通过 `slot` 可以获取到 row, column, \$index 数据

自定义操作列

3.4.1 安装图标模块

```
1 # npm
2 npm install @element-plus/icons-vue
3
4 # yarn
5 yarn add @element-plus/icons-vue
6
7 # pnpm
8 pnpm install @element-plus/icons-vue
```

3.4.2 导入并注册图标

```
1 // main.js
2 import { Delete, Edit } from '@element-plus/icons-vue'
3
4 app.component(Delete.name, Delete)
5 app.component(Edit.name, Edit)
```

3.4.3 插槽自定义列模版

```
1 <script setup>
2   const tableData = [
3     {
4       date: '2016-05-03',
5       name: 'Tom',
6       address: 'No. 189, Grove St, Los Angeles'
7     },
8     {
```

```
9     date: '2016-05-02',
10     name: 'Tom',
11     address: 'No. 189, Grove St, Los Angeles'
12 },
13 {
14     date: '2016-05-04',
15     name: 'Tom',
16     address: 'No. 189, Grove St, Los Angeles'
17 },
18 {
19     date: '2016-05-01',
20     name: 'Tom',
21     address: 'No. 189, Grove St, Los Angeles'
22 }
23 ]
```

```
24 </script>
```

```
25
26 <template>
27   <el-table
28     stripe
29     border
30     :data="tableData"
31     style="width: 60%">
32     <el-table-column
33       prop="date"
34       label="Date"
35       width="180" />
36     <el-table-column
37       label="Name"
38       prop="name"
39       width="180" />
40
41     <el-table-column
42       prop="address"
43       label="Address" />
44     <el-table-column label="操作">
45       <template #default>
46         <el-icon :size="18">
47           <Delete />
48         </el-icon>
49         <el-icon :size="18">
50           <Edit />
51         </el-icon>
52       </template>
53     </el-table-column>
54   </el-table>
55 </template>
```

4. Tree树形组件

4.1 介绍

用清晰的层级结构展示信息，可展开或折叠。

▼ ☐ Level one 1

▼ ☐ Level two 1-1

☐ Level three 1-1-1

☐ Level three 1-1-2

☐ Level one 2

☐ Level two 2-1

☐ Level two 2-2

▼ ☐ Level one 3

☐ Level two 3-1

☐ Level two 3-2

Append Delete

Append Delete

Append Delete

Append Delete

Append Delete

Append Delete

Append Delete

Append Delete

Append Delete

Append Delete

4.2 基础用法及示例代码

```
1 <script setup>
2   // 节点 点击时
3   const handleClick = (data) => {
4     // data: 当前点击的节点
5     console.log(data)
6   }
7
8   // 树形数据源
9   const data = [
10    {
11      label: 'Level one 1',
12      children: [
13        {
14          label: 'Level two 1-1',
15          children: [
16            {
17              label: 'Level three 1-1-1'
18            }
19          ]
20        }
21      ]
22    },
23    {
24      label: 'Level one 2',
25      children: [
26        {
27          label: 'Level two 2-1',
28          children: [
29            {
```

```
30         label: 'Level three 2-1-1'
31     }
32 ]
33 },
34 {
35     label: 'Level two 2-2',
36     children: [
37         {
38             label: 'Level three 2-2-1'
39         }
40     ]
41 }
42 ]
43 },
44 {
45     label: 'Level one 3',
46     children: [
47         {
48             label: 'Level two 3-1',
49             children: [
50                 {
51                     label: 'Level three 3-1-1'
52                 }
53             ]
54         },
55         {
56             label: 'Level two 3-2',
57             children: [
58                 {
59                     label: 'Level three 3-2-1'
60                 }
61             ]
62         }
63     ]
64 }
65 ]
66
67 // 默认渲染的属性名
68 const defaultProps = {
69     children: 'children',
70     label: 'label'
71 }
72 </script>
73
74 <template>
75     <!--
76     style: 样内样式
```

```

77     data: 数据源
78     props: 默认渲染的属性名 为 label 和 children
79     show-checkbox: 节点是否可被选择
80     accordion: 是否每次只打开一个同级树节点展开
81     @node-click: 节点点击事件
82 -->
83 <el-tree
84     style="max-width: 600px"
85     :data="data"
86     :props="defaultProps"
87     show-checkbox
88     accordion
89     @node-click="handleNodeClick" />
90 </template>

```

4.3 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

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。

```

1 <script setup>
2   import { ref } from 'vue'
3
4   // 树形数据
5   const data = ref([
6     {
7       id: 1,
8       label: 'Level one 1',
9       children: [
10        {
11          id: 4,
12          label: 'Level two 1-1',
13          children: [
14            {
15              id: 9,
16              label: 'Level three 1-1-1'
17            },
18            {
19              id: 10,
20              label: 'Level three 1-1-2'
21            }

```

```
22     ]
23   }
24 ]
25 },
26 {
27   id: 2,
28   label: 'Level one 2',
29   children: [
30     {
31       id: 5,
32       label: 'Level two 2-1'
33     },
34     {
35       id: 6,
36       label: 'Level two 2-2'
37     }
38   ]
39 },
40 {
41   id: 3,
42   label: 'Level one 3',
43   children: [
44     {
45       id: 7,
46       label: 'Level two 3-1'
47     },
48     {
49       id: 8,
50       label: 'Level two 3-2'
51     }
52   ]
53 }
54 ])
55 </script>
56
57 <template>
58   <el-tree
59     style="max-width: 600px"
60     :data="data"
61     show-checkbox
62     node-key="id"
63     default-expand-all>
64     <!-- 作用域插槽自定义节点内容 -->
65     <!-- data: 当前行的节点对象数据 -->
66     <template #default="{ data }">
67       <span class="custom-tree-node">
68         <span>{{ data.label }}</span>
```

```

69     <span>
70       <a> Append </a>
71       <a style="margin-left: 8px"> Delete </a>
72     </span>
73   </span>
74 </template>
75 </el-tree>
76 </template>
77
78 <style scoped>
79   .custom-tree-node {
80     flex: 1;
81     display: flex;
82     align-items: center;
83     justify-content: space-between;
84     font-size: 14px;
85     padding-right: 8px;
86   }
87 </style>

```

5. Dialog弹框组件

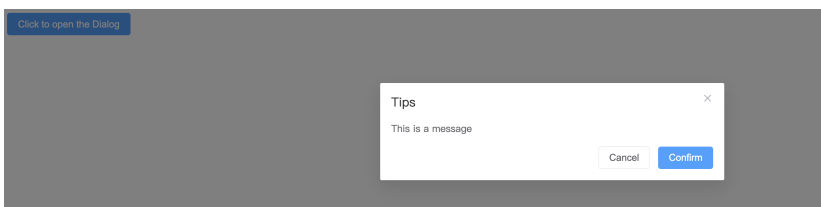
5.1 介绍

在保留当前页面状态的情况下, 告知用户并承载相关操作

5.2 基础用法

Dialog 弹出一个对话框, 适合需要定制性更大的场景。

需要设置 `v-model` 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。



5.3 示例代码

```

1 <script setup>
2   import { ref } from 'vue'
3
4   // 导入确认函数
5   import { ElMessageBox } from 'element-plus'

```

```

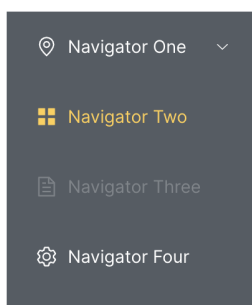
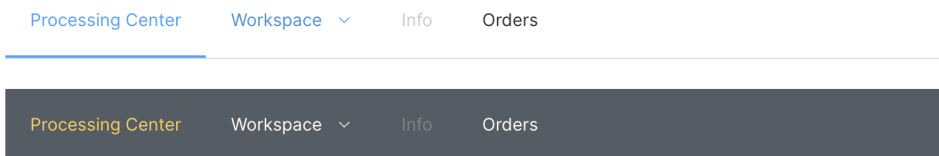
6
7 // 对话框是否可见
8 const visible = ref(false)
9
10 // 关闭前
11 const onBeforeClose = () => {
12   ElMessageBox.confirm('Are you sure to close this dialog?')
13     .then(() => {
14       done()
15     })
16     .catch(() => {
17       // catch error
18     })
19 }
20 </script>
21 <template>
22   <el-button
23     @click="visible = true"
24     type="primary">
25     Click to open the Dialog
26   </el-button>
27
28   <!--
29     v-model: 控制对话框显示隐藏
30     title: 标题
31     width: 宽度
32     before-close: 关闭前的回调函数
33   -->
34   <el-dialog
35     v-model="visible"
36     title="Tips"
37     width="500"
38     :before-close="onBeforeClose">
39     <!-- 默认插槽: 主体内容 -->
40     <span>This is a message</span>
41     <!-- 具名插槽: 底部内容 -->
42     <template #footer>
43       <div class="dialog-footer">
44         <el-button @click="visible = false">Cancel</el-button>
45         <el-button
46           type="primary"
47           @click="visible = false">
48           Confirm
49         </el-button>
50       </div>
51     </template>
52   </el-dialog>

```


6. Menu菜单组件

6.1 介绍

为网站提供导航功能的菜单。



6.2 基础用法

顶部栏菜单可以在各种场景中使用。

导航菜单默认为垂直模式，通过将 `mode` 属性设置为 `horizontal` 来使导航菜单变更为水平模式。另外，在菜单中通过 `sub-menu` 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。

6.3 示例代码

```
1 <script setup>
2   // 导入字体图标
3   import {
4     Document,
5     Menu as IconMenu,
6     Location,
7     Setting
8   } from '@element-plus/icons-vue'
9
10  // 打开(展开指定的子菜单)
11  const handleOpen = (key, keyPath) => {
12    console.log(key, keyPath)
13  }
14  // 关闭(关闭指定的子菜单)
```

```

15   const handleClose = (key, keyPath) => {
16     console.log(key, keyPath)
17   }
18 </script>
19 <template>
20   <!--
21     菜单外层盒子:
22     text-color: 文字颜色
23     active-text-color: 激活菜单项的文本颜色
24     background-color: 菜单的背景颜色
25     default-active: 页面加载时默认激活菜单的 index
26     class: 自定义类名
27   -->
28   <el-menu
29     active-text-color="#ffd04b"
30     background-color="#545c64"
31     class="el-menu-vertical-demo"
32     default-active="2"
33     text-color="#fff"
34     @open="handleOpen"
35     @close="handleClose">
36     <!-- 子菜单,用于生成二级菜单 -->
37     <el-sub-menu index="1">
38       <template #title>
39         <el-icon><location /></el-icon>
40         <span>Navigator One</span>
41       </template>
42       <!--
43         通过 el-menu-item-group 组件可以实现菜单进行分组,
44         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。
45       -->
46       <el-menu-item-group title="Group One">
47         <el-menu-item index="1-1">item one</el-menu-item>
48         <el-menu-item index="1-2">item two</el-menu-item>
49       </el-menu-item-group>
50       <el-menu-item-group title="Group Two">
51         <el-menu-item index="1-3">item three</el-menu-item>
52       </el-menu-item-group>
53       <el-sub-menu index="1-4">
54         <template #title>item four</template>
55         <el-menu-item index="1-4-1">item one</el-menu-item>
56       </el-sub-menu>
57     </el-sub-menu>
58     <el-menu-item index="2">
59       <el-icon><icon-menu /></el-icon>
60       <span>Navigator Two</span>
61     </el-menu-item>

```

```
62     <el-menu-item
63         index="3"
64         disabled>
65         <el-icon><document /></el-icon>
66         <span>Navigator Three</span>
67     </el-menu-item>
68     <el-menu-item index="4">
69         <el-icon><setting /></el-icon>
70         <span>Navigator Four</span>
71     </el-menu-item>
72 </el-menu>
73 </template>
```

比特就业课