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

```
<script setup></script>
<template></template>
<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

```
# npm
npm i element-plus

# yarn
yarn add element-plus

# pnpm
pnpm i element-plus
```

2. 完整引入

```
// 导入 ElementPlus 组件
import ElementPlus from 'element-plus'
// 导入组件样式
import 'element-plus/dist/index.css'

// 全局注册所有组件
app.use(ElementPlus)
```


3. 测试

以 Button 组件为例，在 App.vue 测试

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

三、学习ElementPlus常用组件

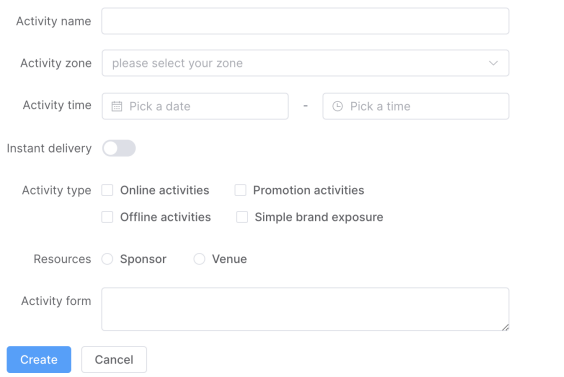
1. Form表单组件

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。



1.2 基本使用

```vue
<script setup>
  import { reactive } from 'vue'

  // 表单对象
  const form = reactive({
    name: '',
    region: '',
    date1: '',
    date2: '',
    delivery: false,
    type: [],
    resource: '',
    desc: ''
  })

  // 提交
  const onSubmit = () => {
    console.log('submit!')
  }
</script>
<template>
  <!--
    model: 表单数据对象
    label-width: 标签的宽度, 可使用 auto
    style: 行内样式
  -->
  <el-form
    :model="form"
```

```
    label-width="auto"
    style="max-width: 600px">
    <!--
      label: 标签文本
    -->
    <el-form-item label="Activity name">
      <!--
         v-model: 双向绑定, 收集或设置表单数据
      -->
      <el-input v-model="form.name" />
    </el-form-item>
    <el-form-item label="Activity zone">
      <!--
         下拉列表
         placeholder: 占位提示文本
      -->
      <el-select
        v-model="form.region"
        placeholder="please select your zone">
        <!--
           选项
           label: 展示的文本
           value: 同步到v-model的值

        -->
        <el-option
          label="Zone one"
          value="shanghai" />
        <el-option
          label="Zone two"
          value="beijing" />
      </el-select>
    </el-form-item>
    <el-form-item label="Activity time">
      <!--
        列
        span: 占据的宽度份数, 一行分为24份

      -->
      <el-col :span="11">
        <!-- 日期选择器 -->
        <el-date-picker
          v-model="form.date1"
          type="date"
          placeholder="Pick a date"
          style="width: 100%" />
      </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>
```

```
      </el-radio-group>
    </el-form-item>
    <el-form-item label="Activity form">
      <el-input
        v-model="form.desc"
        type="textarea" />
    </el-form-item>
    <el-form-item>
      <!-- 提交按钮 -->
      <el-button
        type="primary"
        @click="onSubmit"
        >Create</el-button
      >
      <!-- 取消按钮 -->
      <el-button>Cancel</el-button>
    </el-form-item>
  </el-form>
</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。

```
<script setup>
  import { reactive, ref } from 'vue'

  const formRef = ref(null)

  // 表单对象
  const form = reactive({
    name: '',
    region: '',
    date1: '',
    date2: '',
    delivery: false,
    type: [],
    resource: '',
    desc: ''
  })

  // 校验规则
  const rules = {
```

```
    name: [
      {
        required: true,
        message: 'Please input Activity name',
        trigger: 'blur'
      },
      { min: 3, max: 5, message: 'Length should be 3 to 5', trigger: 'blur' }
    ],
    region: [
      {
        required: true,
        message: 'Please select Activity zone',
        trigger: 'change'
      }
    ],
    count: [
      {
        required: true,
        message: 'Please select Activity count',
        trigger: 'change'
      }
    ],
    date1: [
      {
        type: 'date',
        required: true,
        message: 'Please pick a date',
        trigger: 'change'
      }
    ],
    date2: [
      {
        type: 'date',
        required: true,
        message: 'Please pick a time',
        trigger: 'change'
      }
    ],
    location: [
      {
        required: true,
        message: 'Please select a location',
        trigger: 'change'
      }
    ],
    type: [
      {
```

```
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"
```

```
    :rules="rules"
    ref="formRef"
    label-width="auto"
    status-icon>
    <!--
      prop: 校验的字段
    -->
    <el-form-item
      label="Activity name"
      prop="name">
      <el-input v-model="form.name" />
    </el-form-item>
    <el-form-item
      label="Activity zone"
      prop="region">
      <el-select
        v-model="form.region"
        placeholder="Activity zone">
        <el-option
          label="Zone one"
          value="shanghai" />
        <el-option
          label="Zone two"
          value="beijing" />
      </el-select>
    </el-form-item>
    <el-form-item
      label="Activity time"
      required>
      <el-col :span="11">
        <el-form-item prop="date1">
          <el-date-picker
            v-model="form.date1"
            type="date"
            aria-label="Pick a date"
            placeholder="Pick a date"
            style="width: 100%" />
        </el-form-item>
      </el-col>
      <el-col
        class="text-center"
        :span="2">
        <span class="text-gray-500">-</span>
      </el-col>
      <el-col :span="11">
        <el-form-item prop="date2">
          <el-time-picker
```

```
            v-model="form.date2"
            aria-label="Pick a time"
            placeholder="Pick a time"
            style="width: 100%" />
        </el-form-item>
      </el-col>
    </el-form-item>
    <el-form-item
      label="Instant delivery"
      prop="delivery">
      <el-switch v-model="form.delivery" />
    </el-form-item>
    <el-form-item
      label="Activity type"
      prop="type">
      <el-checkbox-group v-model="form.type">
        <el-checkbox
          value="Online activities"
          name="type">
          Online activities
        </el-checkbox>
        <el-checkbox
          value="Promotion activities"
          name="type">
          Promotion activities
        </el-checkbox>
        <el-checkbox
          value="Offline activities"
          name="type">
          Offline activities
        </el-checkbox>
        <el-checkbox
          value="Simple brand exposure"
          name="type">
          Simple brand exposure
        </el-checkbox>
      </el-checkbox-group>
    </el-form-item>
    <el-form-item
      label="Resources"
      prop="resource">
      <el-radio-group v-model="form.resource">
        <el-radio value="Sponsorship">Sponsorship</el-radio>
        <el-radio value="Venue">Venue</el-radio>
      </el-radio-group>
    </el-form-item>
    <el-form-item
```

```
      label="Activity form"
      prop="desc">
      <el-input
        v-model="form.desc"
        type="textarea" />
    </el-form-item>
    <el-form-item>
      <el-button
        type="primary"
        @click="submitForm">
        Create
      </el-button>
      <el-button @click="resetForm">Reset</el-button>
    </el-form-item>
  </el-form>
</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

```
<script setup>
  import { reactive, ref } from 'vue'

  import { ElMessage } from 'element-plus'

  // 登录表单
  const loginForm = reactive({
    username: '', // 用户名
    password: '' // 密码
```

```
})

const formRef = ref(null)

// 校验规则
const loginFormRules = {
  // 用户名
  username: [
    {
      required: true,
      message: '用户名不能为空',
      trigger: 'blur'
    },
    {
      min: 3,
      max: 6,
      message: '用户名长度在3-6之间',
      trigger: 'blur'
    }
  ],
  // 密码
  password: [
    {
      required: true,
      message: '密码不能为空',
      trigger: 'blur'
    },
    {
      min: 6,
      max: 15,
      message: '密码长度在6-15之间',
      trigger: 'blur'
    }
  ]
}

// 登录
const onLogin = () => {
  // 整体校验
  formRef.value.validate((valid) => {
    // 如果没通过校验，给出错题提示并提前结束代码的执行
    if (!valid) return ElMessage.error('用户名或密码不对')
    console.log('ok')
    // ...
  })
}
```

```
  // 取消
  const onCancel = () => {
    // 进行表单重置
    formRef.value.resetFields()
  }
</script>

<template>
  <!-- 登录外层盒子 -->
  <div class="login-box">
    <!-- 表单组件 -->
    <el-form
      ref="formRef"
      :model="loginForm"
      :rules="loginFormRules">
      <el-form-item prop="username">
        <!--
          prefix-icon: 在输入框的前面插入字体图标
        -->
        <el-input
          v-model="loginForm.username"
          placeholder="请填写用户名"
          prefix-icon="User" />
      </el-form-item>
      <el-form-item prop="password">
        <el-input
          v-model="loginForm.password"
          placeholder="请填写密码"
          prefix-icon="Lock"
          type="password"
          show-password />
      </el-form-item>
      <el-form-item>
        <el-button
          type="primary"
          @click="onLogin"
          >登 录</el-button
        >
        <el-button @click="onCancel">取 消</el-button>
      </el-form-item>
    </el-form>
  </div>
</template>

<style lang="scss" scoped>
  * {
    margin: 0;
```

```
  }
  body {
    background: #ddd;
  }

  .login-box {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    .el-form {
      width: 500px;
      padding: 40px 50px;
      background: #fff;
      .el-form-item {
        margin: 20px 0;
      }
      .el-input {
        height: 40px;
      }
    }
  }
</style>
```

2. Card卡片组件

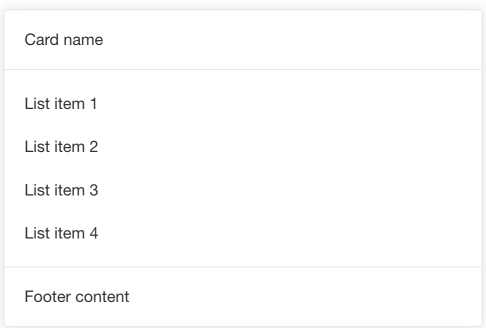
2.1 介绍

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。



2.3 示例代码

```vue
<template>
  <el-card style="max-width: 480px">
    <template #header>
      <div class="card-header">
        <span>Card name</span>
      </div>
    </template>
    <template #default>
      <p
        v-for="i in 4"
        :key="i"
        class="text item">
        {{ 'List item ' + i }}
      </p>
    </template>
    <template #footer>Footer content</template>
  </el-card>
</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

```vue
<script setup>
  const tableData = [
    {
      date: '2016-05-03',
      name: 'Tom',
```

```
      address: 'No. 189, Grove St, Los Angeles'
    },
    {
      date: '2016-05-02',
      name: 'Tom',
      address: 'No. 189, Grove St, Los Angeles'
    },
    {
      date: '2016-05-04',
      name: 'Tom',
      address: 'No. 189, Grove St, Los Angeles'
    },
    {
      date: '2016-05-01',
      name: 'Tom',
      address: 'No. 189, Grove St, Los Angeles'
    }
  ]
</script>

<template>
  <!--
    data: 数据源
    stripe: 带有斑马纹
    border: 带边框
  -->
  <el-table
    :data="tableData"
    stripe
    border
    style="width: 100%">
    <!--
       table-column: 列
          label: 列名
          prop: 渲染的字段名
          width: 列宽
    -->
    <el-table-column
      prop="date"
      label="Date"
      width="180" />
    <el-table-column
      prop="name"
      label="Name"
      width="180" />
    <el-table-column
      prop="address"
```

```
      label="Address" />
  </el-table>
</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

```
# npm
npm install @element-plus/icons-vue

# yarn
yarn add @element-plus/icons-vue

# pnpm
pnpm install @element-plus/icons-vue
```

3.4.2 导入并注册图标

```
// main.js
import { Delete, Edit } from '@element-plus/icons-vue'

app.component(Delete.name, Delete)
app.component(Edit.name, Edit)
```

3.4.3 插槽自定义列模版

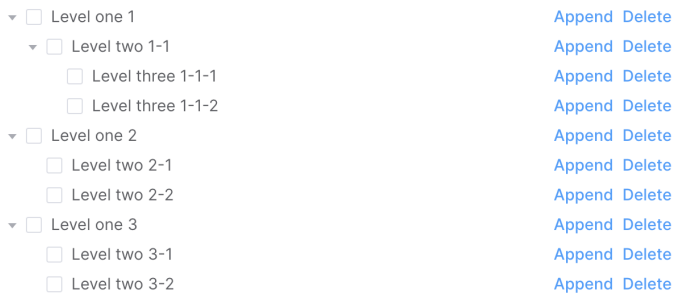
```
<script setup>
  const tableData = [
    {
      date: '2016-05-03',
      name: 'Tom',
      address: 'No. 189, Grove St, Los Angeles'
    },
    {
```

```
      date: '2016-05-02',
      name: 'Tom',
      address: 'No. 189, Grove St, Los Angeles'
    },
    {
      date: '2016-05-04',
      name: 'Tom',
      address: 'No. 189, Grove St, Los Angeles'
    },
    {
      date: '2016-05-01',
      name: 'Tom',
      address: 'No. 189, Grove St, Los Angeles'
    }
  ]
</script>

<template>
  <el-table
    stripe
    border
    :data="tableData"
    style="width: 60%">
    <el-table-column
      prop="date"
      label="Date"
      width="180" />
    <el-table-column
      label="Name"
      prop="name"
      width="180" />

    <el-table-column
      prop="address"
      label="Address" />
    <el-table-column label="操作">
      <template #default>
        <el-icon :size="18">
          <Delete />
        </el-icon>
        <el-icon :size="18">
          <Edit />
        </el-icon>
      </template>
    </el-table-column>
  </el-table>
</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。



4.2 基础用法及示例代码

```vue
<script setup>
  // 节点 点击时
  const handleNodeClick = (data) => {
    // data: 当前点击的节点
    console.log(data)
  }

  // 树形数据源
  const data = [
    {
      label: 'Level one 1',
      children: [
        {
          label: 'Level two 1-1',
          children: [
            {
              label: 'Level three 1-1-1'
            }
          ]
        }
      ]
    },
    {
      label: 'Level one 2',
      children: [
        {
          label: 'Level two 2-1',
          children: [
            {
```

```
              label: 'Level three 2-1-1'
            }
          ]
        },
        {
          label: 'Level two 2-2',
          children: [
            {
              label: 'Level three 2-2-1'
            }
          ]
        }
      ]
    },
    {
      label: 'Level one 3',
      children: [
        {
          label: 'Level two 3-1',
          children: [
            {
              label: 'Level three 3-1-1'
            }
          ]
        },
        {
          label: 'Level two 3-2',
          children: [
            {
              label: 'Level three 3-2-1'
            }
          ]
        }
      ]
    }
  ]

  // 默认渲染的属性名
  const defaultProps = {
    children: 'children',
    label: 'label'
  }
</script>

<template>
  <!--
    style: 样内样式
```

```
    data: 数据源
    props: 默认渲染的属性名  为 label 和 children
    show-checkbox: 节点是否可被选择
    accordion: 是否每次只打开一个同级树节点展开
    @node-click: 节点点击事件
  -->
  <el-tree
    style="max-width: 600px"
    :data="data"
    :props="defaultProps"
    show-checkbox
    accordion
    @node-click="handleNodeClick" />
</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。

```
<script setup>
  import { ref } from 'vue'

  // 树形数据
  const data = ref([
    {
      id: 1,
      label: 'Level one 1',
      children: [
        {
          id: 4,
          label: 'Level two 1-1',
          children: [
            {
              id: 9,
              label: 'Level three 1-1-1'
            },
            {
              id: 10,
              label: 'Level three 1-1-2'
            }
```

```
          ]
        }
      ]
    },
    {
      id: 2,
      label: 'Level one 2',
      children: [
        {
          id: 5,
          label: 'Level two 2-1'
        },
        {
          id: 6,
          label: 'Level two 2-2'
        }
      ]
    },
    {
      id: 3,
      label: 'Level one 3',
      children: [
        {
          id: 7,
          label: 'Level two 3-1'
        },
        {
          id: 8,
          label: 'Level two 3-2'
        }
      ]
    }
  ])
</script>

<template>
  <el-tree
    style="max-width: 600px"
    :data="data"
    show-checkbox
    node-key="id"
    default-expand-all>
    <!-- 作用域插槽自定义节点内容 -->
    <!-- data: 当前行的节点对象数据 -->
    <template #default="{ data }">
      <span class="custom-tree-node">
        <span>{{ data.label }}</span>
```

```
        <span>
          <a> Append </a>
          <a style="margin-left: 8px"> Delete </a>
        </span>
      </span>
    </template>
  </el-tree>
</template>

<style scoped>
  .custom-tree-node {
    flex: 1;
    display: flex;
    align-items: center;
    justify-content: space-between;
    font-size: 14px;
    padding-right: 8px;
  }
</style>
```

5. Dialog弹框组件

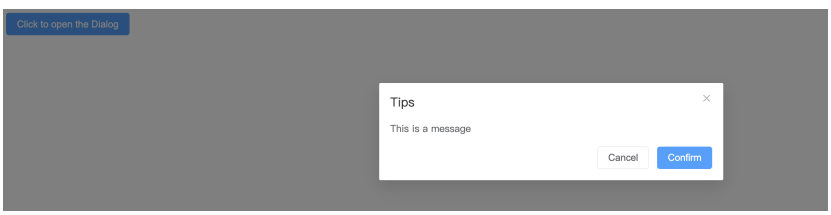
5.1 介绍

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

```
<script setup>
  import { ref } from 'vue'

  // 导入确认函数
  import { ElMessageBox } from 'element-plus'
```

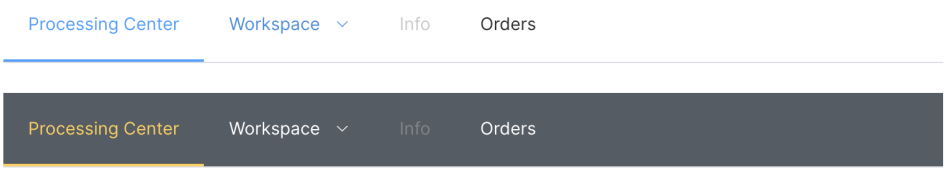
```
6
7   // 对话框是否可见
8   const visible = ref(false)
9
10  // 关闭前
11  const onBeforeClose = () => {
12    ElMessageBox.confirm('Are you sure to close this dialog?')
13      .then(() => {
14        done()
15      })
16      .catch(() => {
17        // catch error
18      })
19  }
20 </script>
21 <template>
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>
```

```
53 </template>
```

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>
```

```
    <el-menu-item
      index="3"
      disabled>
      <el-icon><document /></el-icon>
      <span>Navigator Three</span>
    </el-menu-item>
    <el-menu-item index="4">
      <el-icon><setting /></el-icon>
      <span>Navigator Four</span>
    </el-menu-item>
  </el-menu>
</template>
```