UI组件库 Ant Design

> 讲师：阿杜

一、课程概要

1. 学习路线

- 认识 Ant Design
- Ant Design 布局组件
- Ant Design 表单组件
- Ant Design 数据表格
- Ant Design 反馈组件

2. 课程目标

- 掌握Ant Design常用组件
- 结合Vue写出好看的页面

二、认识Ant Design

1. 什么是UI组件库

1.1 UI组件库介绍

- ui库就是常说的ui组件库，是在我们在Web端开发过程中用来搭建页面的组件集合，它可以作为单个组件独立存在，也可以通过多个组件组合而成的结构或模式来解决类似场景的设计问题。组件库是在约束条件下去构建解决方案的过程，所以组件的使用也需遵循一定的规范，按照一套标准化的体系复用于多个业务场景。
- 一个有效的组件库，可以帮助研发人员提高工作效率，提升专业度的同时让产品本身的体验更加一致、可学，品牌感更强，它所具备的基本特征一定是通用的、灵活的、复用的。

1.2 UI组件库特性

- 通用性：意味着足够基础和常见且不带业务属性，参与开发的每个人都应该知道这个组件的功能及目的，同时具备一定扩展性。
- 灵活性：要求组件的组合需灵活，可以在不同场景下通过互相组合来快速搭建Web界面，并根据不同页面结构的演变来适应新的业务需求。
- 复用性：指的是适用于多个业务或产品，在设计过程和研发过程中都可以高频复用。

优点知识

1.3 常用组件库

Element

- Element UI 它是由饿了么前端团队推出的基于 Vue 封装的 UI 组件库，提供PC 端组件，简化了常用组件的封装，降低开发难度。

- Vant 一款有赞出品轻量、可靠的的移动UI组件库，目前支持 Vue2、Vue3、React，微信和支付宝小程序，并由社区团队维护 React 版本和支付宝小程序版本。

- LayUI 经典开源模块化前端 UI 框架(官方文档镜像站)，JQuery 时代很好的 UI 库，是一套开源的 Web UI 解决方案,其内部采用的是自身经典的模块化规范,并遵循原生 HTML/CSS/JS 的开发方式,极易上手,拿来即用。

- Semi Design 出自字节跳动抖音前端与 UED 团队，是一款包含设计语言、React 组件、主题等开箱即用的中后台解决方案，可用于快速搭建美观的 React 应用。

- Vuetify 是建立在Vue.js之上的完备的界面框架。 该项目的目标是向开发人员提供他们积累的丰富且吸引用户体验所需的工具。 Vuetify从一开始就设计为易于学习的并且拥有来自 Material设计规范的数百个精心设计组件。每一个组件都经过精心设计，具有模块化、响应式和优秀的性能。

2.Ant Design介绍

- Ant Design 是一套企业级 UI 设计语言和 React 组件库，提供了一套非常完整的组件化设计规范与组件化编码规范，能大幅提高了部分产品的设计研发效率及质量。
- Ant Design Vue 是蚂蚁金服 Ant Design 官方唯一推荐的Vue版UI组件库，组件的风格与Ant Design保持同步，组件的html结构和css样式也保持一致。
- Ant Design特性
  - 企业级金融产品的交互语言和视觉体系。
  - 丰富实用的 React UI 组件。

    - 基于 React 的组件化开发模式。
    - 背靠 npm 生态圈。
    - 基于 webpack 的调试构建方案，支持 ES6。
- 组件库文档：https://antdv.com/components/overview-cn

3. 引入Ant Designa

3.1 版本选择

- Ant Design所有版本

  Ant Design Charts

  Ant Design Pro

  Ant Design Pro Components

  Ant Design of Angular (community)

  Ant Design of Vue (community)

- Ant Design of Vue 社区版

  3.2.14

  2.x (Not Recommended)

  1.x (For Vue 2)

- 我们使用：vue3 + ant-design-vue@3.2.11

3.2 引入方式

- 使用组件

```
npm i --save ant-design-vue
```

- 参考文档：https://antdv.com/docs/vue/getting-started-cn

4. 创建vue+antd项目

- 版本选择：vue3 + ant-design-vue@3.2.11
- 初始化Vue3项目

```
vue create antd-demo
#选择vue3项目。。。
```

- 安装ant-design-vue

```
cd antd-demo
npm install
npm install ant-design-vue
npm run serve
```

- 全局注册

```
// src/main.js
import { createApp } from 'vue';
import Antd from 'ant-design-vue';
import App from './App';
import 'ant-design-vue/dist/antd.css';

const app = createApp(App);

app.use(Antd).mount('#app');
```

- 注册路由
  - 安装vue-router

    ```
    npm install vue-router
    ```

  - 定义路由策略

    ```
    // src/router/index.js
    //导入router的路由模式
    import {createRouter, createWebHistory} from 'vue-router'

    //路由规则
    const routes = [
        {
            path: '/btn',  //url路径
            component: () => import('@/views/test/btn.vue'),  //视图组件
        }
    ]

    // createRouter 创建路由实例
    const router = createRouter({
        /**
         * hash模式: createWebHashHistory,
         * history模式: createWebHistory
         */
        history: createWebHistory(),
        routes
    })

    // 抛出路由实例, 在 main.js 中引用
    export default router
    ```

- 使用 <router-view/>，定义路由占位

```
// src/App.vue
<template>
  <!-- 路由占位符 -->
  <router-view></router-view>
</template>

<style>
#app {
  width: 100vw;
  height: 100vh;
}
</style>
```

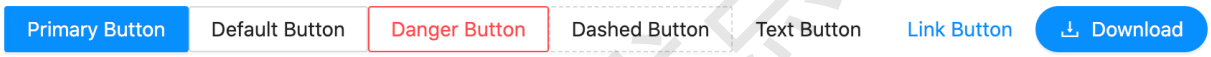
- 全局引入router和ant图标

```
// src/main.js
import { createApp } from 'vue';
import Antd from 'ant-design-vue';
import App from './App';
import 'ant-design-vue/dist/antd.css';
import * as Icons from '@ant-design/icons-vue'
import router from './router';

const app = createApp(App)
//全局引入图标组件，i是组件名，Icons[i]是具体组件
for (const i in Icons) {
    app.component(i, Icons[i])
}

app.use(Antd)
app.use(router)
app.mount('#app');
```

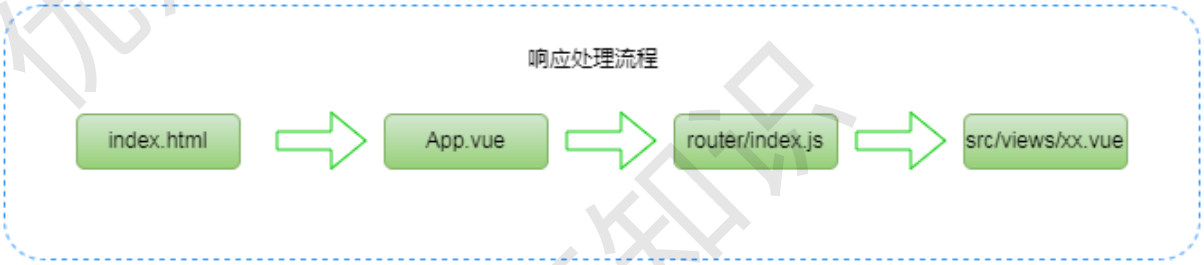
- 使用一个antd的按钮组件
  - 按钮页面

```
// src/views/btn.vue
<template>
    <div>
        <a-button type="primary">Primary Button</a-button>
        <a-button>Default Button</a-button>
        <a-button danger>Danger Button</a-button>
        <a-button type="dashed">Dashed Button</a-button>
        <a-button type="text">Text Button</a-button>
        <a-button type="link">Link Button</a-button>
    </div>
</template>
```

- 浏览器打开`localhost:8080/btn`



5. 开发与响应流程



三、Ant Design 布局组件

1. Card 卡片

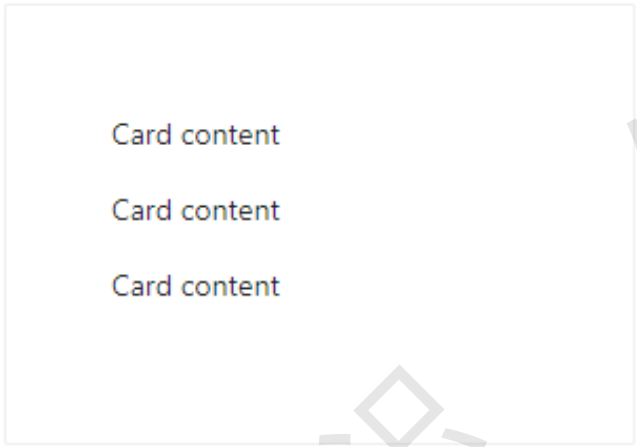
- 通用卡片容器，可承载文字、列表、图片、段落，常用于后台概览页面。
- 标签：`<a-card></a-card>`
- 常用属性：

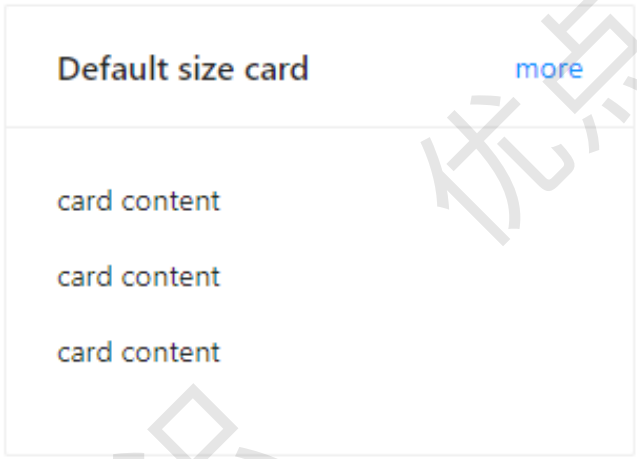
属性	说明	值
bodyStyle	内容区域自定义样式	css属性的对象，如"{padding: '10px'}"
bordered	是否有边框	true | false
extra	卡片右上角的操作区域	string | slot
size	卡片的尺寸	default | small
title	卡片标题	string | slot
bodyStyle	内容区域自定义样式	css属性的对象
headStyle	自定义标题区域样式	css属性的对象

- 示例代码：

```
<template>
    <div style="margin:20px">
        <!-- 没有标题的卡片 -->
        <a-card style="width: 300px" :bodyStyle="{padding: '50px'}">
            <p>Card content</p>
            <p>Card content</p>
            <p>Card content</p>
        </a-card>
        <br>
        <!-- 有标题的卡片 -->
        <a-card title="卡片标题" :bordered="false" style="width: 300px">
            <p>Card content</p>
            <p>Card content</p>
            <p>Card content</p>
        </a-card>
        <br>
        <!-- 带扩展项的卡片 -->
        <a-card title="Default size card" style="width: 300px">
            <template #extra><a href="#">more</a></template>
            <p>card content</p>
            <p>card content</p>
            <p>card content</p>
        </a-card>
        <br>
    </div>
</template>
```

- 运行结果





2. Affix 固钉

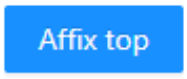
- 当内容区域比较长，需要滚动页面时，这部分内容对应的操作或者导航需要在滚动范围内始终展现。常用于侧边菜单和按钮组合。
- 常用属性

属性	说明	值
offsetBottom	距离窗口底部达到指定偏移量后触发	top
offsetTop	距离窗口顶部达到指定偏移量后触发	Bottom

- 示例代码

```
<template>
    <div style="margin:20px;height:1000px;">
        <a-affix :offset-top="top">
            <a-button type="primary" @click="top += 10">Affix top</a-button>
        </a-affix>
        <div style="height:500px;"/>
        <a-affix :offset-bottom="bottom">
            <a-button type="primary" @click="bottom += 10">Affix bottom</a-button>
        </a-affix>
    </div>
</template>
<script>
import { ref } from 'vue';
export default({
    setup() {
        const top = ref(10);
        const bottom = ref(10);
        return {
            top,
            bottom,
        };
    },
});
</script>
```

- 运行结果



3. Grid 栅格

- 布局的栅格化系统，我们是基于行（row）和列（col）来定义信息区块的外部框架，以保证页面的每个区域能够稳健地排布起来。下面简单介绍一下它的工作原理：
    - 通过 row 在水平方向建立一组 column （简写 col）
    - 你的内容应当放置于 col 内，并且，只有 col 可以作为 row 的直接元素
    - 栅格系统中的列是指 1 到 24 的值来表示其跨越的范围。例如，三个等宽的列可以使用 `` 来创建

- 如果一个 row 中的 col 总和超过 24，那么多余的 col 会作为一个整体另起一行排列

- 标签：
  - <a-row></a-row>
  - 常用属性：

属性	说明	值
align	flex 布局下的垂直对齐方式	top middle bottom
justify	flex 布局下的水平排列方式	start end center space-around space-between
gutter	格栅之间的间隔，支持水平和垂直的方式	number | array

  - <a-col></a-col>
  - 常用属性：

属性	说明	值
offset	栅格左侧的间隔格数，间隔内不可以有栅格	number
span	栅格占位格数，为 0 时相当于 display: none	number

- 示例代码

```
<template>
    <div class="gutter-example">
        <a-row :gutter="[10,10]" justify="center">
            <a-col class="gutter-row" :span="6">
                <div class="gutter-box">col-6</div>
            </a-col>
            <a-col class="gutter-row" :span="6">
                <div class="gutter-box">col-6</div>
            </a-col>
            <a-col class="gutter-row" :span="6">
                <div class="gutter-box">col-6</div>
            </a-col>
            <a-col class="gutter-row" :span="6">
                <div class="gutter-box">col-6</div>
            </a-col>
            <a-col class="gutter-row" :span="6">
                <div class="gutter-box">col-6</div>
            </a-col>
            <a-col class="gutter-row" :span="6">
                <div class="gutter-box">col-6</div>
            </a-col>
        </a-row>
    </div>
</template>

<style scoped>
```

```
.gutter-example :deep(.ant-row > div) {
    background: transparent;
    border: 0;
}
.gutter-box {
    background: #00a0e9;
    padding: 5px 0;
}
</style>
```

- 运行结果

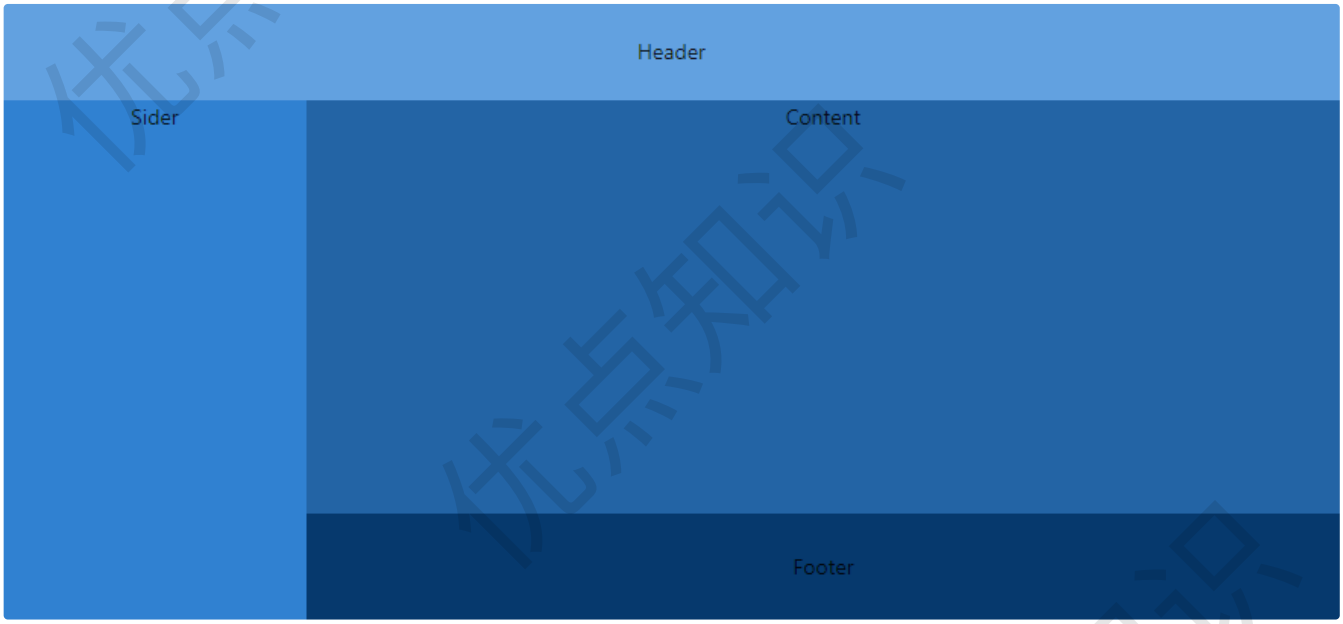
4. Layout 布局

- 用于页面整体布局，方便快速搭建页面的基本结构
- 组件概述：
  - `Layout`：布局容器，其下可嵌套 `Header` `Sider` `Content` `Footer` 或 `Layout` 本身，可以放在任何父容器中。
  - `Header`：顶部布局，自带默认样式，其下可嵌套任何元素，只能放在 `Layout` 中。
  - `Sider`：侧边栏，自带默认样式及基本功能，其下可嵌套任何元素，只能放在 `Layout` 中。
  - `Content`：内容部分，自带默认样式，其下可嵌套任何元素，只能放在 `Layout` 中。
  - `Footer`：底部布局，自带默认样式，其下可嵌套任何元素，只能放在 `Layout` 中。
- 常用标签：
  - `<a-layout></a-layout>`
  - `<a-layout-header></a-layout-header>`
  - `<a-layout-sider></a-layout-sider>`
  - `<a-layout-content></a-layout-content>`
  - `<a-layout-footer></a-layout-footer>`
- 示例代码：

```
<template>
    <div class="layout">
        <a-layout style="height:100vh;">
            <a-layout-header>Header</a-layout-header>
            <a-layout>
                <a-layout-sider>Sider</a-layout-sider>
                <a-layout>
                    <a-layout-content>Content</a-layout-content>
                    <a-layout-footer>Footer</a-layout-footer>
                </a-layout>
            </a-layout>
        </a-layout>
    </div>
</template>

<style scoped>
    .layout {
```

```
        text-align:center;
    }
    .ant-layout-header {
        background: rgb(98, 161, 224);
    }
    .ant-layout-sider {
        background: rgb(48, 129, 209);
    }
    .ant-layout-content {
        background: rgb(35, 100, 165);
    }
    .ant-layout-footer {
        background: rgb(6, 57, 109);
    }
</style>
```

- 运行结果



5. Menu 导航菜单

- 导航菜单是一个网站的灵魂，用户依赖导航在各个页面中进行跳转。一般分为顶部导航和侧边导航，顶部导航提供全局性的类目和功能，侧边导航提供多级结构来收纳和排列网站架构。
- 常用标签:
  - `<a-menu></a-menu>`
  - 常用属性:

属性	说明	值
mode	菜单类型，现在支持垂直、水平、和内嵌模式三种	vertical horizontal inline
openKeys(v-model)	当前展开的 SubMenu 菜单项 key 数组	string[]
selectedKeys(v-model)	当前选中的菜单项 key 数组	string[]
theme	主题颜色	lignt dark

- <a-menu-item></a-menu-item>
- <a-sub-menu></a-sub-menu>
- 常用属性:

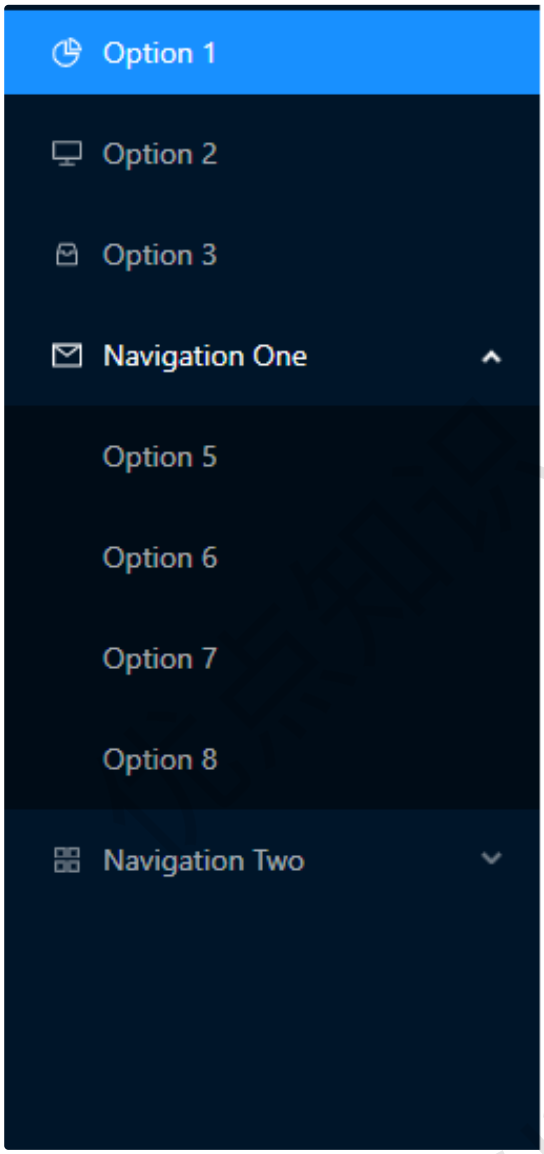
属性	说明	值
disabled	是否禁用	Boolean
icon	菜单图标	slot
key	item 的唯一标志	string
title	设置收缩时展示的悬浮标题	string | slot

- 示例代码

```
<template>
    <div style="width: 256px;height: 100vh;background: #001529;">
        <a-menu
        v-model:openKeys="openKeys"
        v-model:selectedKeys="selectedKeys"
        mode="inline"
        theme="dark">
            <a-menu-item key="1">
                <template #icon>
                    <PieChartOutlined />
                </template>
                <span>Option 1</span>
            </a-menu-item>
            <a-menu-item key="2">
                <template #icon>
                    <DesktopOutlined />
                </template>
                <span>Option 2</span>
            </a-menu-item>
            <a-menu-item key="3">
                <template #icon>
                    <InboxOutlined />
                </template>
                <span>Option 3</span>
            </a-menu-item>
```

```
            <a-sub-menu key="sub1">
                <template #icon>
                    <MailOutlined />
                </template>
                <template #title>Navigation One</template>
                <a-menu-item key="5">Option 5</a-menu-item>
                <a-menu-item key="6">Option 6</a-menu-item>
                <a-menu-item key="7">Option 7</a-menu-item>
                <a-menu-item key="8">Option 8</a-menu-item>
            </a-sub-menu>
            <a-sub-menu key="sub2">
                <template #icon>
                    <AppstoreOutlined />
                </template>
                <template #title>Navigation Two</template>
                <a-menu-item key="9">Option 9</a-menu-item>
                <a-menu-item key="10">Option 10</a-menu-item>
                <a-sub-menu key="sub3" title="Submenu">
                    <a-menu-item key="11">Option 11</a-menu-item>
                    <a-menu-item key="12">Option 12</a-menu-item>
                </a-sub-menu>
            </a-sub-menu>
        </a-menu>
    </div>
</template>

<script>
import { reactive, toRefs } from 'vue'
export default ({
    setup() {
        const state = reactive({
            selectedKeys: ['1'],
            openKeys: ['sub1']
        })
        return {
            ...toRefs(state)
        }
    },
})
</script>
```

- 运行结果



四、Ant Design 表单组件

1. Form 表单

- 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。
- 表单内常用组件：
  - `a-input`：输入框
  - `a-select`：下拉选择框
  - `a-checkbox-group`：多选框
  - `a-radio-group`：单选框
  - `a-switch`：开关
- 常用标签
  - `<a-form></a-form>`
  - 常用属性：

属性	说明	值
model	表单数据对象	object
labelCol	label 标签布局，同 <Col> 组件，设置 span offset 值， 如 {span: 3, offset: 12} 或 sm: {span: 3, offset: 12}	Object
rules	表单验证规则	object
layout	表单布局	horizontal | vertical | inline
labelAlign	label 标签的文本对齐方式	left | right

- <a-form-item></a-form-item>
- 常用属性：

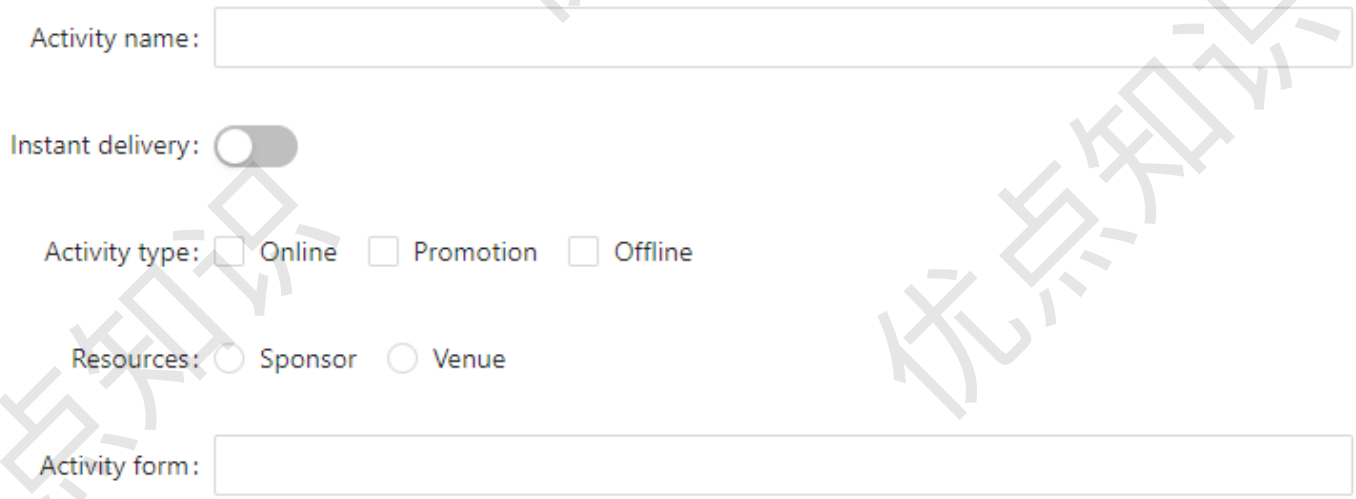
属性	说明	值
label	label 标签的文本	string | slot
name	表单域 model 字段，在使用 validate、resetFields 方法的情况下，该属性是必填的	String
rules	表单验证规则	object

- 示例代码：

```
<template>
    <div style="margin:20px;">
        <a-form :model="formState" :label-col="labelCol">
            <a-form-item label="Activity name">
                <a-input v-model:value="formState.name" />
            </a-form-item>
            <a-form-item label="Instant delivery">
                <a-switch v-model:checked="formState.delivery" />
            </a-form-item>
            <a-form-item label="Activity type">
                <a-checkbox-group v-model:value="formState.type">
                    <a-checkbox value="1" name="type">Online</a-checkbox>
                    <a-checkbox value="2" name="type">Promotion</a-checkbox>
                    <a-checkbox value="3" name="type">Offline</a-checkbox>
                </a-checkbox-group>
            </a-form-item>
            <a-form-item label="Resources">
                <a-radio-group v-model:value="formState.resource">
                    <a-radio value="1">Sponsor</a-radio>
                    <a-radio value="2">Venue</a-radio>
                </a-radio-group>
            </a-form-item>
            <a-form-item label="Activity form">
```

```
                <a-input v-model:value="formState.desc" type="textarea" />
            </a-form-item>
        </a-form>
    </div>
</template>

<script>
import { reactive } from 'vue';
export default ({
    setup() {
        const formState = reactive({
            name: '',
            delivery: false,
            type: [],
            resource: '',
            desc: '',
        });
        return {
            labelCol: {
                style: {
                    width: '150px',
                },
            },
            formState,
        };
    },
});
</script>
```

- 运行结果



优点知识

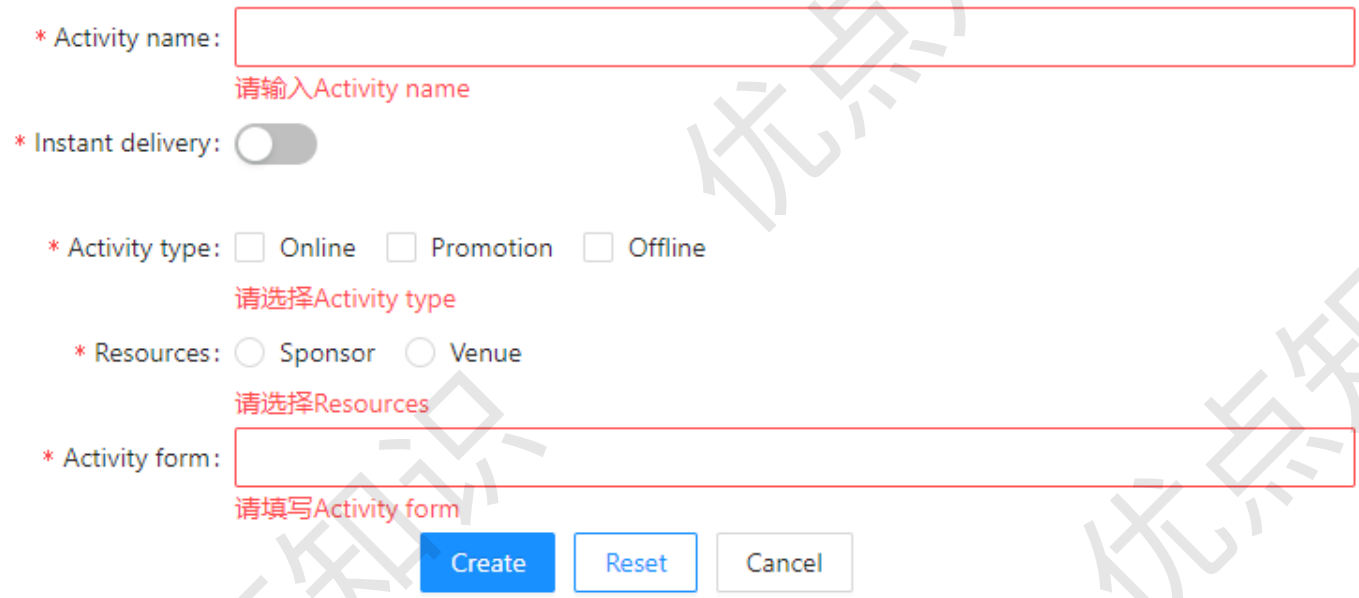
2．表单校验与重置

- 表单效验，用于检查表单输入是否合法。
- 实现方式：
  - `<a-form>` 添加 `ref`， `ref` 用来获取dom元素或者组件
  - `<a-form-item>` 添加 `rules` 和 `name` 属性
  - 编写验证和重置方法
- 示例代码：

```
<template>
    <div style="margin:20px">
        <a-form ref="formRef" :model="formState" :label-col="labelCol">
            <a-form-item
                label="Activity name"
                name="name"
                :rules="[{ required: true, message: '请输入Activity name' }]">
                <a-input v-model:value="formState.name" />
            </a-form-item>
            <a-form-item
                label="Instant delivery"
                name="delivery"
                :rules="[{ required: true, message: '请选择Instant delivery' }]">
                <a-switch v-model:checked="formState.delivery" />
            </a-form-item>
            <a-form-item
                label="Activity type"
                name="type"
                :rules="[{ required: true, message: '请选择Activity type' }]">
                <a-checkbox-group v-model:value="formState.type">
                    <a-checkbox value="1" name="type">Online</a-checkbox>
                    <a-checkbox value="2" name="type">Promotion</a-checkbox>
                    <a-checkbox value="3" name="type">Offline</a-checkbox>
                </a-checkbox-group>
            </a-form-item>
            <a-form-item
                label="Resources"
                name="resource"
                :rules="[{ required: true, message: '请选择Resources' }]">
                <a-radio-group v-model:value="formState.resource">
                    <a-radio value="1">Sponsor</a-radio>
                    <a-radio value="2">Venue</a-radio>
                </a-radio-group>
            </a-form-item>
            <a-form-item
                label="Activity form"
                name="form"
                :rules="[{ required: true, message: '请填写Activity form' }]">
                <a-input v-model:value="formState.desc" type="textarea" />
            </a-form-item>
```

```
            <a-form-item :wrapper-col="{ span: 14, offset: 8 }">
                <a-button type="primary" @click="formSubmit()">Create</a-button>
                <a-button style="margin-left: 10px" type="primary" ghost
@click="resetForm()">Reset</a-button>
                <a-button style="margin-left: 10px">Cancel</a-button>
            </a-form-item>
        </a-form>
    </div>
</template>

<script>
import { reactive, ref } from 'vue';
export default ({
    setup() {
        const formState = reactive({
            name: '',
            delivery: false,
            type: [],
            resource: '',
            desc: '',
        });
        //验证表单
        const formRef = ref()
        async function formSubmit() {
            try {
                await formRef.value.validateFields();
                //console.log('Success:', values);
            } catch (errorInfo) {
                console.log('Failed:', errorInfo);
            }
        }
        function resetForm() {
            formRef.value.resetFields();
        }
        return {
            labelCol: {
                style: {
                    width: '150px',
                },
            },
            formRef,
            formState,
            formSubmit,
            resetForm
        };
    },
});
</script>
```

- 运行结果



3. 自定义验证表单

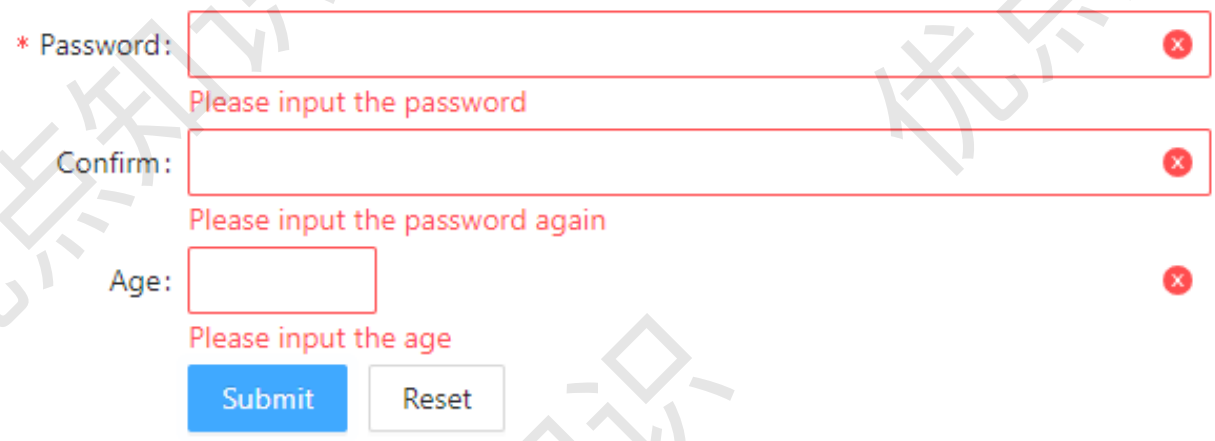
- 自定义验证规则，判断表单内容是否合法
- 实现方式：
  - `<a-form>` 添加 `ref`， `ref` 用来获取dom元素或者组件
  - `<a-form>` 添加 `rules` 属性
  - `<a-form-item>` 添加 `name` 属性
  - 编写验证方法，绑定到 `rules` 中
- 示例代码

```
<template>
    <div style="margin:20px">
        <a-form
            ref="formRef"
            name="custom-validation"
            :model="formState"
            :rules="rules"
            v-bind="layout">
            <a-form-item has-feedback label="Password" name="pass">
                <a-input v-model:value="formState.pass" type="password" />
            </a-form-item>
            <a-form-item has-feedback label="Confirm" name="checkPass">
                <a-input v-model:value="formState.checkPass" type="password" />
            </a-form-item>
            <a-form-item has-feedback label="Age" name="age">
                <a-input-number v-model:value="formState.age" />
            </a-form-item>
            <a-form-item :wrapper-col="{ span: 14, offset: 4 }">
                <a-button type="primary" html-type="submit">Submit</a-button>
                <a-button style="margin-left: 10px" @click="resetForm">Reset</a-
button>
            </a-form-item>
```

```
        </a-form>
    </div>
</template>

<script>
import { reactive, ref } from 'vue';
export default ({
    setup() {
        const formRef = ref();
        const formState = reactive({
            pass: '',
            checkPass: '',
            age: undefined,
        });
        let checkAge = async (_rule, value) => {
            console.log(_rule, value)
            if (!value) {
                return Promise.reject('Please input the age');
            }
            if (!Number.isInteger(value)) {
                return Promise.reject('Please input digits');
            } else {
                if (value < 18) {
                    return Promise.reject('Age must be greater than 18');
                } else {
                    return Promise.resolve();
                }
            }
        };
        let validatePass = async (_rule, value) => {
            if (value === '') {
                return Promise.reject('Please input the password');
            } else {
                if (formState.checkPass !== '') {
                    formRef.value.validateFields('checkPass');
                }
                return Promise.resolve();
            }
        };
        let validatePass2 = async (_rule, value) => {
            if (value === '') {
                return Promise.reject('Please input the password again');
            } else if (value !== formState.pass) {
                return Promise.reject("Two inputs don't match!");
            } else {
                return Promise.resolve();
            }
        };
        const rules = {
            pass: [{
```

```
                required: true,
                validator: validatePass,
                trigger: 'change',
            }],
            checkPass: [{
                validator: validatePass2,
                trigger: 'change',
            }],
            age: [{
                validator: checkAge,
                trigger: 'change',
            }],
            };
            const layout = {
            labelCol: {
                span: 4,
            },
            wrapperCol: {
                span: 14,
            },
        };
        const resetForm = () => {
            formRef.value.resetFields();
        };
        return {
            formState,
            formRef,
            rules,
            layout,
            resetForm,
        };
    },
});
</script>
```

- 运行结果



4. 上传图片

- 通过点击上传，常用于图片或文件的上传。
- 常用属性

属性	说明	值
accept	接受上传的文件类型	accept=".png, .jpg, .jpeg"
action	上传的地址	string | (file) ⇒ Promise
data	上传所需参数或返回上传参数的方法	object | (file) ⇒ object
fileList	已经上传的文件列表	object[]
listType	上传列表的内建样式，支持三种基本样式 text，picture 和 picture-card	string
maxCount	限制上传数量。当为 1 时，始终用最新上传的文件代替当前文件	number
customRequest	通过覆盖默认的上传行为，可以自定义自己的上传实现	Function(data)

- 常用事件

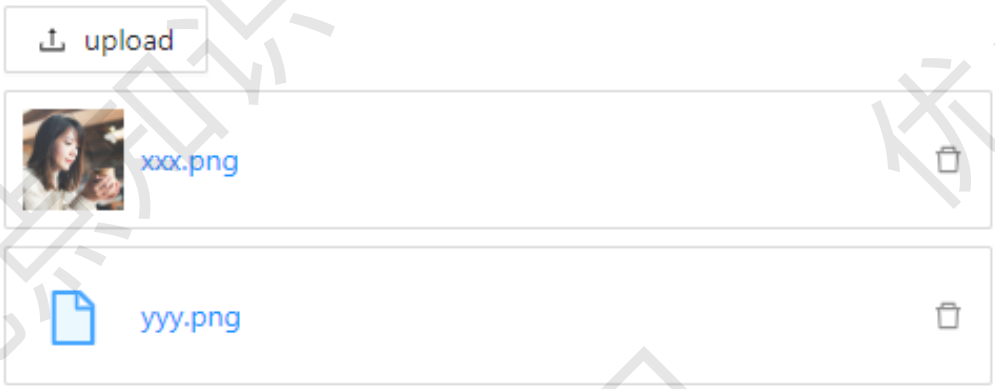
事件	回调参数
change	Function

- 示例代码

```
<template>
    <div style="margin:20px">
        <!-- action和customRequest用其一 -->
        <a-upload
            v-model:file-list="fileList"
            action="https://www.mocky.io/v2/5cc8019d300000980a055e76"
            list-type="picture"
            @change="onChange"
            :customRequest="customRequest">
            <a-button>
                <upload-outlined></upload-outlined>
                upload
            </a-button>
        </a-upload>
    </div>
</template>

<script>
import { ref } from 'vue';
export default ({
```

```
  setup() {
    const fileList = ref([{
        uid: '-1',
        name: 'xxx.png',
        status: 'done',
        url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png',
        thumbUrl:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png',
    }, {
        uid: '-2',
        name: 'yyy.png',
        status: 'done',
        url:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png',
        thumbUrl:
'https://zos.alipayobjects.com/rmsportal/jkjgkEfvpUPVyRjUImniVslZfWPnJuuZ.png11',
    }]);
    function onChange() {
        console.log(fileList)
    }
    //自定义上传
    function customRequest(data) {
        console.log(data)
    }
    return {
        fileList,
        onChange,
        customRequest
    };
  },
});
</script>
```

- 运行结果



五、Ant Design 数据表格

1. 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。
- 常用属性

属性	说明	值
bodyCell	个性化单元格	v-slot:bodyCell="{text, record, index, column}"
bordered	是否展示外边框和列边框	boolean
loading	页面是否加载中	boolean
columns	表格列的配置描述	array
dataSource	数据数组	object[]
pagination	分页器	object
size	表格大小	default | middle | small

- 常用事件

事件	说明	回调参数
change	分页、排序、筛选变化时触发	Function(pagination, filters, sorter, { currentDataSource })

- 示例代码

```
<template>
    <div style="margin:20px">
        <a-button type="primary">操作</a-button>
        <a-table
            :dataSource="dataSource"
            :columns="columns"
            :loading="false"
            size="small">
            <!-- column是定义的columns数据 -->
            <!-- record是定义的dataSource数据 -->
            <template #bodyCell="{ column, record }">
                <!-- 自定义每一列的数据 -->
                <template v-if="column.dataIndex === 'name'">
                    <span style="font-weight:bold;color:blue;">{{ record.name }}
</span>
                </template>
            </template>
            <!-- 扩展 -->
            <template #expandedRowRender="{ record }">
                <p style="margin: 0">
                    {{ record.address }}
```

```
                </p>
            </template>
        </a-table>
    </div>
</template>
<script>
import { ref } from 'vue'
export default {
    setup() {
        const dataSource =  ref([
            {
                key: '1',
                name: '胡彦斌',
                age: 32,
                address: '西湖区湖底公园1号',
            },
            {
                key: '2',
                name: '胡彦祖',
                age: 42,
                address: '西湖区湖底公园1号',
            },
        ])
        //用于定义表格的列信息
        const columns = ref([
            //title 列名字
            //dataIndex 列数据在数据项中对应的路径，支持通过数组查询嵌套路径
            //key Vue 需要的 key，如果已经设置了唯一的 dataIndex，可以忽略这个属性
            {
                title: '姓名',
                dataIndex: 'name',
                key: 'name',
            },
            {
                title: '年龄',
                dataIndex: 'age',
                key: 'age',
            },
            {
                title: '住址',
                dataIndex: 'address',
                key: 'address',
            },
        ])
        return {
            dataSource,
            columns,
        };
    },
}
```

```
</script>
```

- 运行结果

2. 多选和操作

- 用于表格的多选和批量操作。
- 实现方式：
  - 使用 `<a-table>` 的 `row-relection` 属性，指定 `selectedRowKeys` 和 `onChange`
  - `selectedRowKeys` 是个数组，决定当前选择了哪些行，默认的元素是 `dataSource` 中定义的 `key`
  - `rowKey` 可自定义 `selectedRowKeys` 中的元素
  - `onChange` 是个回调函数，用于表格选择框发生变化时执行的函数
- 示例代码

```
<template>
    <div style="margin:20px">
        <a-button type="primary">操作</a-button>
        <a-table
            :dataSource="dataSource"
            :columns="columns"
            :loading="false"
            size="small"
            rowKey="name"
            :rowSelection="{ selectedRowKeys: selectedRowKeys, onChange:
onSelectChange }">
            <!-- column是定义的columns数据 -->
            <!-- record是定义的dataSource数据 -->
            <template #bodyCell="{ column, record }">
                <!-- 自定义每一列的数据 -->
                <template v-if="column.dataIndex === 'name'">
                    <span style="font-weight:bold;color:blue;">{{ record.name }}
</span>
                </template>
            </template>
            <!-- 扩展 -->
            <template #expandedRowRender="{ record }">
                <p style="margin: 0">
                    {{ record.address }}
                </p>
            </template>
```

```
        </a-table>
    </div>
</template>
<script>
import { ref } from 'vue'
export default {
    setup() {
        const dataSource =  ref([
            {
                key: '1',
                name: '胡彦斌',
                age: 32,
                address: '西湖区湖底公园1号',
            },
            {
                key: '2',
                name: '胡彦祖',
                age: 42,
                address: '西湖区湖底公园1号',
            },
        ])
        //用于定义表格的列信息
        const columns = ref([
            //title 列名字
            //dataIndex 列数据在数据项中对应的路径，支持通过数组查询嵌套路径
            //key Vue 需要的 key，如果已经设置了唯一的 dataIndex，可以忽略这个属性
            {
                title: '姓名',
                dataIndex: 'name',
                key: 'name',
            },
            {
                title: '年龄',
                dataIndex: 'age',
                key: 'age',
            },
            {
                title: '住址',
                dataIndex: 'address',
                key: 'address',
            },
        ])
        const selectedRowKeys = ref([])
        const onSelectChange = key => {
            console.log('selectedRowKeys changed: ', key);
            selectedRowKeys.value = key;
        };
        return {
            dataSource,
            columns,
```

```
            selectedRowKeys,
            onSelectChange
        };
    },
}
</script>
```

- 运行结果

3. 排序

- 使用 `<a-table>` 组件自带的排序功能，进行简单的数据排序
- 排序原理

`compareFn(a, b)` return value	sort order
> 0	sort `a` after `b`
< 0	sort `a` before `b`
=== 0	keep original order of `a` and `b`

- 实现方式
  - 在表格列配置 `columns` 中添加 `sort` 对象，实现排序
    - `compare` 用于比较两者的大小
    - `multiple` 定义多个排序列的情况下的优先级，越大优先级越高

```
{
    title: 'Chinese Score',
    dataIndex: 'chinese',
    sorter: {
        compare: (a, b) => a.chinese - b.chinese,
        multiple: 3,
    },
}
```

- 示例代码

```
<template>
    <div style="margin:20px">
```

```
        <a-table :columns="columns" :data-source="data" />
    </div>
</template>
<script>
import { ref } from 'vue';
export default ({
    setup() {
        const columns = ref([{
            title: 'Name',
            dataIndex: 'name',
        }, {
            title: 'Chinese Score',
            dataIndex: 'chinese',
            sorter: {
                compare: (a, b) => a.chinese - b.chinese,
                multiple: 3,
            },
        }, {
            title: 'Math Score',
            dataIndex: 'math',
            sorter: {
                compare: (a, b) => a.math - b.math,
                multiple: 2,
            },
        }, {
            title: 'English Score',
            dataIndex: 'english',
            sorter: {
                compare: (a, b) => a.english - b.english,
                multiple: 1,
            },
        }])
        const data = ref([{
            key: '1',
            name: 'John Brown',
            chinese: 98,
            math: 60,
            english: 70,
        }, {
            key: '2',
            name: 'Jim Green',
            chinese: 98,
            math: 66,
            english: 89,
        }, {
            key: '3',
            name: 'Joe Black',
            chinese: 98,
            math: 90,
            english: 70,
```

```
        }, {
            key: '4',
            name: 'Jim Red',
            chinese: 88,
            math: 99,
            english: 89,
        }])
        return {
            data,
            columns,
        }
    },
});
</script>
```

- 运行结果

Name	Chinese Score	Math Score	English Score
Jim Red	88	99	89
John Brown	98	60	70
Jim Green	98	66	89
Joe Black	98	90	70

< 1 >

4. 分页

- 常用功能，用于表格数据的分页查询和展示。
- 实现方式
  - 在 `<a-table>` 中使用 `pagination` 分页器属性
  - `setup` 中声明分页器对象，该对象可控制分页器的元素展示，包括跳转、分页Size、条数等
  - 使用 `change` 事件监听表格变化，改变分页器属性的值
- 示例代码

```
<template>
    <div style="margin:20px">
        <a-table :columns="columns" :data-source="data" :pagination="pagination"/>
    </div>
</template>
<script>
import { ref,reactive } from 'vue';
export default ({
    setup() {
```

```
const columns = ref([{
    title: 'Name',
    dataIndex: 'name',
}, {
    title: 'Chinese Score',
    dataIndex: 'chinese',
    sorter: {
        compare: (a, b) => a.chinese - b.chinese,
        multiple: 3,
    },
}, {
    title: 'Math Score',
    dataIndex: 'math',
    sorter: {
        compare: (a, b) => a.math - b.math,
        multiple: 2,
    },
}, {
    title: 'English Score',
    dataIndex: 'english',
    sorter: {
        compare: (a, b) => a.english - b.english,
        multiple: 1,
    },
}])
const data = ref([{
    key: '1',
    name: 'John Brown',
    chinese: 98,
    math: 60,
    english: 70,
}, {
    key: '2',
    name: 'Jim Green',
    chinese: 98,
    math: 66,
    english: 89,
}, {
    key: '3',
    name: 'Joe Black',
    chinese: 98,
    math: 90,
    english: 70,
}, {
    key: '4',
    name: 'Jim Red',
    chinese: 88,
    math: 99,
    english: 89,
}])
```

优点知识

```
        const pagination = reactive({
            showSizeChanger: true,
            showQuickJumper: true,
            total: 0,
            currentPage: 1,
            pagesize: 10,
            pageSizeOptions: ["10", "20", "50", "100"],
            showTotal: total => `共 ${total} 条`
        })
        function handleTableChange(val) {
            pagination.currentPage = val.current
            pagination.pagesize = val.pageSize
        }
        return {
            data,
            columns,
            pagination,
            handleTableChange
        }
    },
});
</script>
```

- 运行结果

Name	Chinese Score	Math Score	English Score
John Brown	98	60	70
Jim Green	98	66	89
Joe Black	98	90	70
Jim Red	88	99	89

共 4 条 < 1 > 10 / page

六、Ant Design 反馈组件

1. Drawer 抽屉

- 抽屉从窗口边缘滑入，用户可在抽屉中完成相应操作后，回到原窗口，常用于数据展示、表单添加等。
- 常用属性

属性	说明	值
bodyStyle	可用于设置 Drawer 内容部分的样式	object
footerStyle	抽屉页脚部件的样式	object
placement	抽屉的方向	'top' | 'right' | 'bottom' | 'left'
title	抽屉的标题	string
v-model:visible	Drawer 是否可见	boolean
width	宽度	string | number

- 示例代码

```
<template>
    <div style="margin:20px">
        <a-button type="primary" @click="showDrawer">Open</a-button>
        <a-drawer
            v-model:visible="visible"
            class="custom-class"
            style="color: red"
            title="Basic Drawer"
            placement="right"
            :bodyStyle="{padding: 50 + 'px'}">
            <p>Some contents...</p>
            <p>Some contents...</p>
            <p>Some contents...</p>
        </a-drawer>
    </div>
</template>
<script>
import { ref } from 'vue';
export default ({
    setup() {
        const visible = ref(false);
        const showDrawer = () => {
            visible.value = true;
        };
        return {
            visible,
            showDrawer,
        };
    },
});
</script>
```

- 运行结果

2. Message 全局提示

- 全局展示操作反馈信息，常用语操作后的结果提示，可提供成功、警告和错误等反馈信息。
- Message类型
  - message.success(content, [duration], onClose)
  - message.error(content, [duration], onClose)
  - message.info(content, [duration], onClose)
  - message.warning(content, [duration], onClose)
  - message.warn(content, [duration], onClose) // alias of warning
  - message.loading(content, [duration], onClose)
- 常用属性

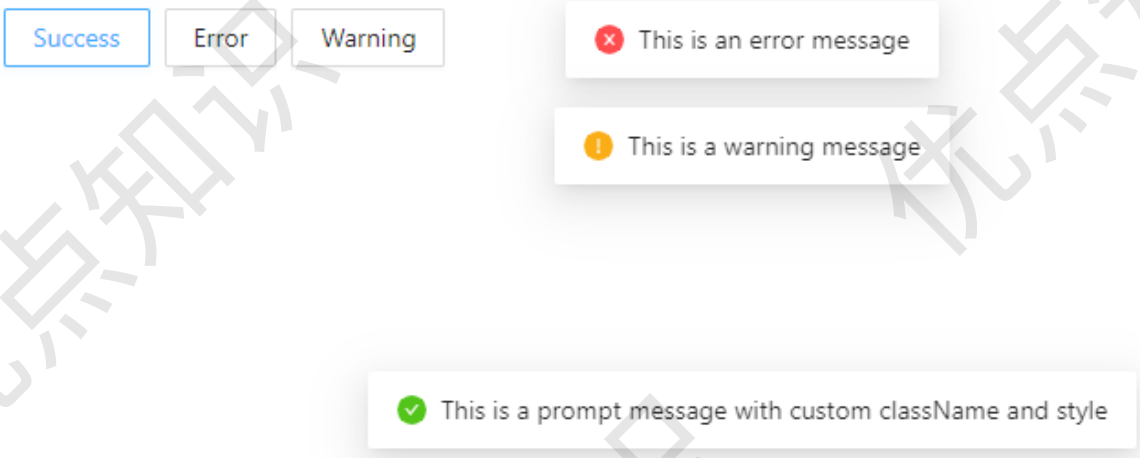
属性	说明	值
content	提示内容	string
class	绑定class	string
style	css属性	object

- 实例代码

```
<template>
    <div style="margin:20px">
        <a-space>
            <a-button @click="success">Success</a-button>
            <a-button @click="error">Error</a-button>
            <a-button @click="warning">Warning</a-button>
        </a-space>
    </div>
</template>
<script>
import { message } from 'ant-design-vue';
```

```
export default({
    setup() {
        const success = () => {
            message.success({
                content: () => 'This is a prompt message with custom className and
style',
                class: 'custom-class',
                style: {
                    marginTop: '20vh',
                },
            });
        };
        const error = () => {
            message.error('This is an error message');
        };
        const warning = () => {
            message.warning('This is a warning message');
        };
        return {
            success,
            error,
            warning,
        };
    },
});
</script>

<style scoped>
    .custom-class {
        color: red;
    }
</style>
```

- 运行结果



优点知识

3. Modal 对话框

- 需要用户处理事务，又不希望跳转页面以致打断工作流程时，可以使用 `Modal` 在当前页面正中打开一个浮层，承载相应的操作。
- 另外当需要一个简洁的确认框询问用户时，可以使用 `Modal.confirm()` 等语法糖方法。
- 常用属性

属性	说明	值
bodyStyle	Modal body 样式	object
cancelText	取消按钮文字	string
okText	确认按钮文字	string
v-model:visible	控制隐藏与展示	boolean
footer	底部内容，当不需要默认底部按钮时，可以设为 `:footer="null"`	string

- 常用事件

事件	说明	值
ok	点击确定回调	function(e)
cancel	点击遮罩层或右上角叉或取消按钮的回调	function(e)

- 示例代码

```
<template>
    <div style="margin:20px">
        <a-button type="primary" @click="showModal">Modal</a-button>
        <a-button @click="confirm">Confirm</a-button>
        <a-modal
            v-model:visible="visible"
            title="Modal"
            ok-text="确认"
            cancel-text="取消"
            @ok="hideModal"
            >
            <p>Bla bla ...</p>
            <p>Bla bla ...</p>
            <p>Bla bla ...</p>
        </a-modal>
    </div>
</template>
<script>
import { ExclamationCircleOutlined } from '@ant-design/icons-vue';
import { ref, createVNode } from 'vue';
import { Modal } from 'ant-design-vue';
export default({
```

```
    setup() {
        const visible = ref(false);
        const showModal = () => {
            visible.value = true;
        };
        const hideModal = () => {
            visible.value = false;
        };
        //createVNode用于创建虚拟节点，里面可以定义图标或者html
        const confirm = () => {
            Modal.confirm({
                title: 'Confirm',
                icon: createVNode(ExclamationCircleOutlined),
                content: 'Bla bla ...',
                okText: '确认',
                cancelText: '取消',
                onOk() {
                    console.log("ok")
                },
                onCancel() {
                    console.log("cancel")
                }
            });
        };
        return {
            visible,
            showModal,
            hideModal,
            confirm,
        };
    },
});
</script>
```

- 运行结果

优点知识

4. Popconfirm 气泡确认框

- 从绑定dom的周围跳出卡片，用于数据展示，不会遮挡原窗口。
- 常用属性

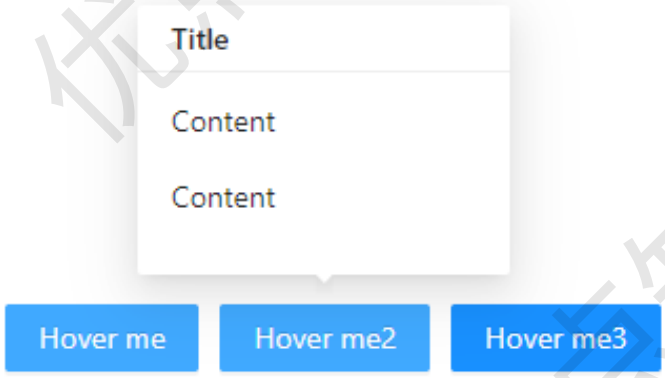
属性	说明	值
placement	卡片位置	top | topRight | topLeft| left |...
trigger	触发条件	hover | focus | click
title	卡片标题	string

- 示例代码

```
<template>
    <div style="margin:200px">
        <a-popover title="Title">
            <template #content>
                <p>Content</p>
                <p>Content</p>
            </template>
            <a-button type="primary">Hover me</a-button>
        </a-popover>
        <a-popover title="Title" placement="top">
            <template #content>
                <p>Content</p>
                <p>Content</p>
            </template>
            <a-button type="primary">Hover me2</a-button>
        </a-popover>
        <a-popover title="Title" trigger="click">
```

```
            <template #content>
                <p>Content</p>
                <p>Content</p>
            </template>
            <a-button type="primary">Hover me3</a-button>
        </a-popover>
    </div>
</template>

<style scoped>
    .ant-btn {
        margin-right: 10px;
    }
</style>
```

- 运行结果



七、整体布局实战

1. 初始化项目

- 创建项目
- 安装依赖
  - vue-router Vue路由
  - nprogress 进度条
  - ant-design-vue 组件库
- 配置路由与进度条，创建相关文件

```
// src/router/index.js

//导入router的路由模式
import {createRouter, createWebHistory} from 'vue-router'
//导入进度条组件
import NProgress from 'nprogress'
import 'nprogress/nprogress.css'
//导入整体布局Layout
import Layout from "@/layout/Layout"

//路由规则
const routes = [
    {
        path: '/',
        redirect: '/home' //重定向
    },
    {
        path: "/home",
        //引入布局组件
        component: Layout,
        children: [
            {
```

优点知识

```
            path: "/home",
            name: "概览",
            icon: "fund-outlined",
            meta: {title: "概览", requireAuth: true},
            component: () => import('@/views/home/Home.vue'),
        }
    ]
},
{
    path: "/cluster",
    name: "集群",
    component: Layout,
    icon: "cloud-server-outlined",
    children: [
        {
            path: "/cluster/node",
            name: "Node",
            meta: {title: "Node", requireAuth: true},
            component: () => import('@/views/cluster/Node.vue'),
        },
        {
            path: "/cluster/namespace",
            name: "Namespace",
            meta: {title: "Namespace", requireAuth: true},
            component: () => import('@/views/cluster/Namespace.vue'),
        },
        {
            path: "/cluster/pv",
            name: "PV",
            meta: {title: "PV", requireAuth: true},
            component: () => import('@/views/cluster/PV.vue'),
        }
    ]
},
{
    path: "/workload",
    name: "工作负载",
    component: Layout,
    icon: "block-outlined",
    children: [
        {
            path: "/workload/pod",
            name: "Pod",
            meta: {title: "Pod", requireAuth: true},
            component: () => import('@/views/workload/Pod.vue'),
        },
        {
            path: "/workload/deployment",
            name: "Deployment",
            meta: {title: "Deployment", requireAuth: true},
```

```
                component: () => import('@/views/workload/Deployment.vue'),
            },
            {
                path: "/workload/daemonset",
                name: "DaemonSet",
                meta: {title: "DaemonSet", requireAuth: true},
                component: () => import('@/views/workload/DaemonSet.vue'),
            },
            {
                path: "/workload/statefulset",
                name: "StatefulSet",
                meta: {title: "StatefulSet", requireAuth: true},
                component: () => import('@/views/workload/StatefulSet.vue'),
            },
        ]
    },
]

// createRouter 创建路由实例
const router = createRouter({
    /**
     * hash模式：createWebHashHistory,
     * history模式：createWebHistory
     */
    history: createWebHistory(),
    routes
})

//递增进度条，这将获取当前状态值并添加0.2直到状态为0.994
NProgress.inc(100)
//easing 动画字符串
//speed 动画速度
//showSpinner 进度环显示隐藏
NProgress.configure({ easing: 'ease', speed: 600, showSpinner: false })

//router.beforeEach () 一般用来做一些进入页面的限制。比如没有登录，就不能进入某些
//页面，只有登录了之后才有权限查看某些页面。。。说白了就是路由拦截。
//to 要去到某个页面的属性
//from 从哪个页面来的属性
//next 处理路由跳转及放行
router.beforeEach((to, from, next) => {
    // 启动进度条
    NProgress.start()

    // 设置头部
    if (to.meta.title) {
        document.title = to.meta.title
    } else {
        document.title = "kubeA"
    }
```

```
    //放行
    next()
})

router.afterEach(() => {
    // 关闭进度条
    NProgress.done()
})

// 抛出路由实例，在 main.js 中引用
export default router
```

- 配置main.js

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import Antd from 'ant-design-vue'
//暗黑风格主体
import 'ant-design-vue/dist/antd.dark.css'
import * as Icons from '@ant-design/icons-vue'


const app = createApp(App)
for (const i in Icons) {
    app.component(i, Icons[i])
  }
app.use(Antd)
app.use(router)
app.mount('#app')
```

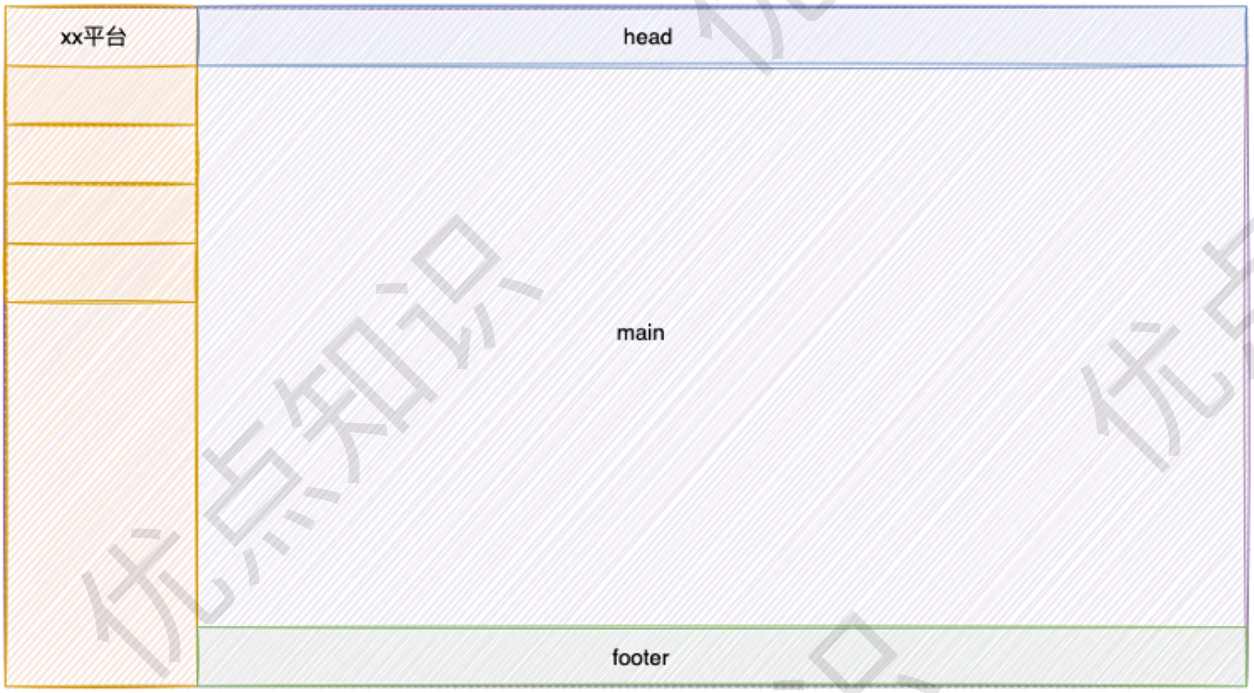
- 配置App.vue

```
<template>
    <router-view></router-view>
</template>

<style>
    html, body {
        width: 100vw;
        height: 100vh;
        min-width: 1425px;
    }
    #nprogress .bar {
        /*自定义进度条颜色*/
        background: #2186c0 !important;
    }
</style>
```

- 验证路由与进度条

http://localhost:8080/home

2. Layout 布局



- src/layout/Layout.vue

2.1 整体布局骨架

- 使用 <a-layout> 相关组件完成整体的布局框架
- 示例代码

```
<template>
    <a-layout>
```

```
        <a-affix>
            <!-- 布局头部 -->
            <a-layout-header>
            </a-layout-header>
        </a-affix>
        <a-layout style="height:calc(100vh - 68px);">
            <!-- 侧边栏，核心 -->
            <!-- collapsed处理展开和收缩  -->
            <a-layout-sider width="240" style="background: #fff" v-
model:collapsed="collapsed" collapsible>
            </a-layout-sider>
            <a-layout style="padding: 0 24px">
                <!-- main的部分 -->
                <a-layout-content
                :style="{
                    background: 'rgb(31, 30, 30)',
                    padding: '24px',
                    margin: 0,
                    minHeight: '280px',
                    overflowY: 'auto'}">
                    <!-- 路由占位符 -->
                    <router-view></router-view>
                </a-layout-content>
                <!-- footer部分 -->
                <a-layout-footer style="text-align: center">
                    ©2022 Created by adoo Devops
                </a-layout-footer>
            </a-layout>
        </a-layout>
    </a-layout>
</template>

<script>
import { ref } from 'vue'
export default({
    setup() {
        const collapsed = ref(false)

        return {
            collapsed
        }
    },
})
</script>
```

2.2 布局头部

- 头部内容：平台信息、集群选择栏、用户信息
- 相关知识点
  - <a-menu> 菜单栏
  - <a-dropdown> 下拉框
  - 灵活使用css进行头部的局部布局
- 示例代码

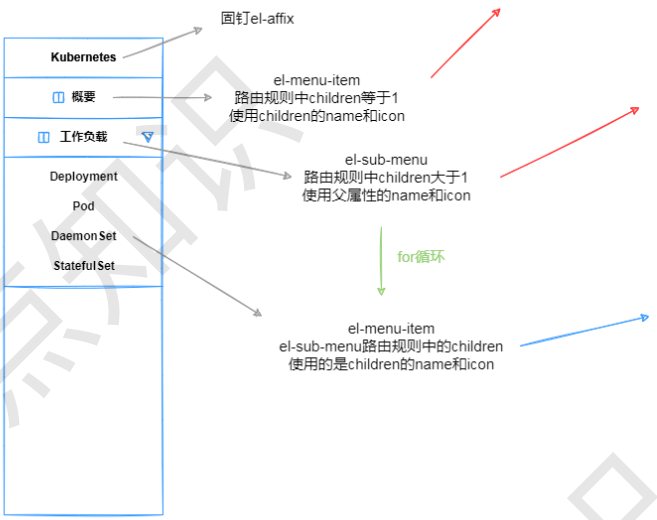
```
<template>
    <a-layout>
        <a-affix>
            <!-- 布局头部 -->
            <a-layout-header>
                <!-- 平台名 -->
                <div style="float:left;">
                    <img style="height:40px;margin-bottom:10px;" :src="kubeLogo" />
                    <span style="padding:0 50px 0 20px;font-size:25px;font-
weight:bold;color:#fff">KubeA</span>
                </div>
                <!-- 集群 -->
                <a-menu
                style="float:left;width:250px;"
                v-model:selectedKeys="selectedKeys1"
                theme="dark"
                mode="horizontal"
                :style="{ lineHeight: '64px' }">
                    <a-menu-item v-for="(item) in clusterList" :key="item">{{ item }}
</a-menu-item>
                </a-menu>
                <!-- 用户信息 -->
                <div style="float:right;">
                    <img style="height:40px;border-radius:50%;margin-right:10px;"
:src="avator"/>
                    <a-dropdown style="margin-top: 10px;">
                        <a class="ant-dropdown-link" @click.prevent>
                            Admin
                            <down-outlined />
                        </a>
                        <template #overlay>
                            <a-menu>
                                <a-menu-item>
                                <a @click="logout()">退出登录</a>
                                </a-menu-item>
                                <a-menu-item>
                                <a href="javascript:;">修改密码</a>
                                </a-menu-item>
                            </a-menu>
```

```
                    </template>
                </a-dropdown>
            </div>
        </a-layout-header>
    </a-affix>
    <a-layout style="height:calc(100vh - 68px);">
        <!-- 侧边栏，核心 -->
        <!-- collapsed处理展开和收缩  -->
        <a-layout-sider width="240" style="background: #fff" v-
model:collapsed="collapsed" collapsible>
        </a-layout-sider>
        <a-layout style="padding: 0 24px">
            <!-- main的部分 -->
            <a-layout-content
            :style="{
                background: 'rgb(31, 30, 30)',
                padding: '24px',
                margin: 0,
                minHeight: '280px',
                overflowY: 'auto'}">
                <!-- 路由占位符 -->
                <router-view></router-view>
            </a-layout-content>
            <!-- footer部分 -->
            <a-layout-footer style="text-align: center">
                ©2022 Created by adoo Devops
            </a-layout-footer>
        </a-layout>
    </a-layout>
</a-layout>
</template>

<script>
import { ref } from 'vue'
import kubeLogo from '@/assets/k8s-metrics.png';
import avator from '@/assets/avator.png';
export default({
    setup() {
        const collapsed = ref(false)
        //【头部】
        const selectedKeys1 = ref([])
        //集群列表
        const clusterList = ref([
            'TST-1',
            'TST-2'
        ])
        //退出登录
        const logout = () => {
            //移除用户名
            localStorage.removeItem('username');
```

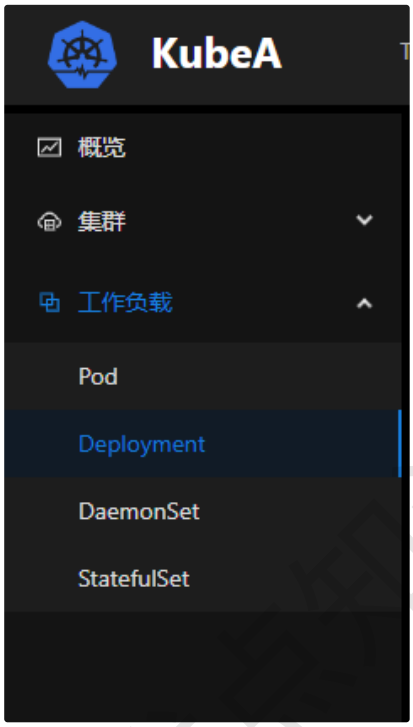
```
            //移除token
            localStorage.removeItem('token');
            //跳转至/login页面
            //router.push('/login');
        }
        return {
            collapsed,
            selectedKeys1,
            clusterList,
            kubeLogo,
            avator,
            logout,
        }
    },
})
</script>

<style>
    .ant-layout-header {
        padding: 0 30px !important;
    }
</style>
```

2.3 侧边栏*

```
{
    path: '/home',
    component: Layout,
    icon: 'odometer',
    children: [
        {
            path: '/home',
            name: '概要',
            icon: 'odometer'
            component: () => import('@/views/home/Home.vue')
        }
    ]
},
```




```
{
    path: '/workload',
    name: '工作负载',
    component: Layout,
    icon: 'menu',
    meta: {title: '工作负载', requireAuth: true},
    children: [
        {
            path: '/workload/deployment',
            name: 'Deployment',
            icon: 'el-icon-s-data',
            meta: {title: 'Deployment', requireAuth: true},
            component: () => import('@/views/deployment/Deployment.vue')
        },
        {
            path: '/workload/pod',
            name: 'Pod',
            icon: 'el-icon-document-add',
            meta: {title: 'Pod', requireAuth: true},
            component: () => import('@/views/pod/Pod.vue')
        }
    ]
},
```



- `<a-menu>` 结合 `<vue-router>` 实现根据路由数据动态展示的侧边栏
- 相关知识点
  - 通过使用 `<a-menu>` 的 `selectedKeys` 、`openKeys` 、`openChange` 等属性或事件自定义选中栏目效果
  - 通过使用 `vue-router` 获取路由信息，绑定菜单栏及控制跳转
  - 生命周期钩子 `onMounted` 的使用
- 示例代码

```
<template>
    <a-layout>
        <a-affix>
            <!-- 布局头部 -->
            <a-layout-header>
                <!-- 平台名 -->
                <div style="float:left;">
                    <img style="height:40px;margin-bottom:10px;" :src="kubeLogo" />
                    <span style="padding:0 50px 0 20px;font-size:25px;font-
weight:bold;color:#fff">KubeA</span>
                </div>
                <!-- 集群 -->
                <a-menu
                style="float:left;width:250px;"
                v-model:selectedKeys="selectedKeys1"
                theme="dark"
                mode="horizontal"
                :style="{ lineHeight: '64px' }">
                    <a-menu-item v-for="(item) in clusterList" :key="item">{{ item }}
</a-menu-item>
                </a-menu>
                <!-- 用户信息 -->
                <div style="float:right;">
```

```
                    <img style="height:40px;border-radius:50%;margin-right:10px;"
:src="avator"/>
                    <a-dropdown style="margin-top: 10px;">
                        <a class="ant-dropdown-link" @click.prevent>
                            Admin
                            <down-outlined />
                        </a>
                        <template #overlay>
                            <a-menu>
                                <a-menu-item>
                                <a @click="logout()">退出登录</a>
                                </a-menu-item>
                                <a-menu-item>
                                <a href="javascript:;">修改密码</a>
                                </a-menu-item>
                            </a-menu>
                        </template>
                    </a-dropdown>
                </div>
            </a-layout-header>
        </a-affix>
        <a-layout style="height:calc(100vh - 68px);">
            <!-- 侧边栏，核心 -->
            <!-- collapsed处理展开和收缩  -->
            <a-layout-sider width="240" style="background: #fff" v-
model:collapsed="collapsed" collapsible>
                <!-- selectedKeys表示点击选中的栏目,用于a-menu-item -->
                <!-- openKeys表示展开的栏目，用于a-sub-menu -->
                <!-- openChange事件监听 SubMenu 展开/关闭的回调 -->
                <a-menu
                :selectedKeys="selectedKeys2"
                :openKeys="openKeys"
                @openChange="onOpenChange"
                mode="inline"
                :style="{ height: '100%', borderRight: 0 }">
                    <!-- routers是router/index.js中的路由信息 -->
                    <template v-for="menu in routers" :key="menu">
                        <!-- 处理无子路由的情况，也就是单个栏目 -->
                        <!-- routeChange用于路由跳转 -->
                        <a-menu-item
                        v-if="menu.children && menu.children.length == 1"
                        :index="menu.children[0].path"
                        :key="menu.children[0].path"
                        @click="routeChange('item', menu.children[0].path)">
                            <component :is="menu.children[0].icon" />
                            <span>{{ menu.children[0].name }}</span>
                        </a-menu-item>
                        <!-- 处理有子路由的情况，也就是父栏目 -->
                        <a-sub-menu
                        v-else-if="menu.children && menu.children.length > 1"
```

```
                        :index="menu.path"
                        :key="menu.path">
                            <template #title>
                                <span>
                                    <component :is="menu.icon" />
                                    <span :class="[collapsed ? 'is-collapse' : '']">
{{ menu.name }}</span>
                                </span>
                            </template>
                            <!-- 子栏目（子路由）的处理 -->
                            <a-menu-item
                            v-for="child in menu.children"
                            :key="child.path"
                            :index="child.path"
                            @click="routeChange('sub', child.path)">
                                <span>{{ child.name }}</span>
                            </a-menu-item>
                        </a-sub-menu>
                    </template>
                </a-menu>
            </a-layout-sider>
            <a-layout style="padding: 0 24px">
                <!-- main的部分 -->
                <a-layout-content
                :style="{
                    background: 'rgb(31, 30, 30)',
                    padding: '24px',
                    margin: 0,
                    minHeight: '280px',
                    overflowY: 'auto'}">
                    <!-- 路由占位符 -->
                    <router-view></router-view>
                </a-layout-content>
                <!-- footer部分 -->
                <a-layout-footer style="text-align: center">
                    ©2022 Created by adoo Devops
                </a-layout-footer>
            </a-layout>
        </a-layout>
    </a-layout>
</template>

<script>
import { ref, onMounted } from 'vue'
import { useRouter } from 'vue-router'
import kubeLogo from '@/assets/k8s-metrics.png';
import avator from '@/assets/avator.png';
export default({
    setup() {
        const collapsed = ref(false)
```

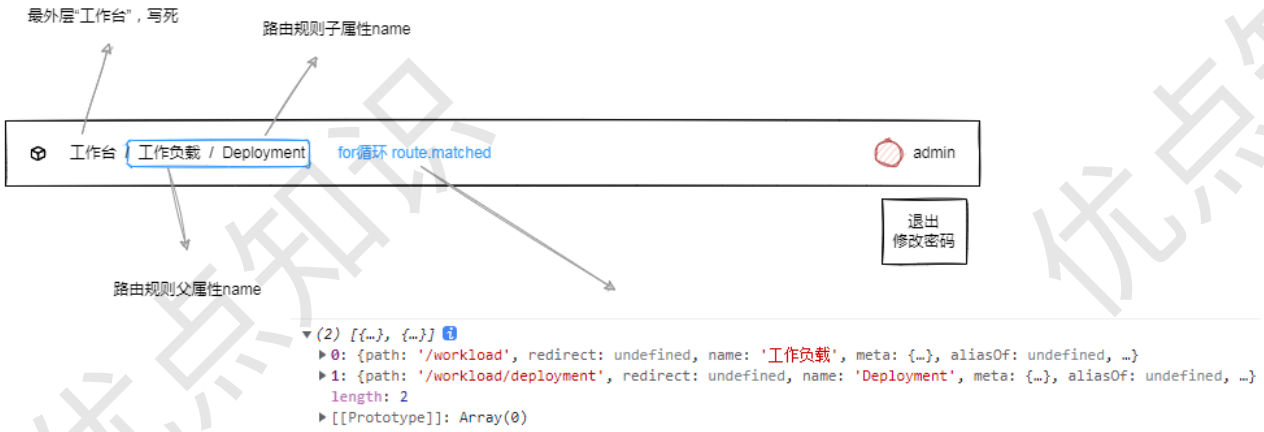
```
        //【头部】
        const selectedKeys1 = ref([])
        const clusterList = ref([
            'TST-1',
            'TST-2'
        ])
        //退出登录
        const logout = () => {
            //移除用户名
            localStorage.removeItem('username');
            //移除token
            localStorage.removeItem('token');
            //跳转至/login页面
            router.push('/login');
        }
        //【侧边栏】
        const routers = ref([])
        const selectedKeys2 = ref([])
        const openKeys = ref([])
        //使用useRouter方法获取路由配置以及当前路由信息
        const router = useRouter()
        //导航栏点击切换页面
        const routeChange =  (type, path) => {
            //判断点击的栏目是否为父栏目，如果不是，则关闭其他父栏目
            if (type !== 'sub') {
                openKeys.value = []
            }
            //表示选中当前path对应的栏目
            selectedKeys2.value = [path]
            if(router.currentRoute.value.path !== path){
                //获取路由对象并切换
                router.push(path)
            }
        }
        //根据url自动导航栏目高亮
        const getRouter = (val) => {
            selectedKeys2.value = [val[1].path]
            openKeys.value = [val[0].path]
        }
        //处理菜单栏展开与关闭
        const onOpenChange = (val) => {
            const latestOpenKey = val.find(key => openKeys.value.indexOf(key) ===
    -1);
            openKeys.value = latestOpenKey ? [latestOpenKey] : [];
        }
        //生命周期钩子
        onMounted(() => {
            routers.value = router.options.routes
            getRouter(router.currentRoute.value.matched)
        })
```

```
        return {
            collapsed,
            selectedKeys1,
            clusterList,
            kubeLogo,
            avator,
            routers,
            router,
            selectedKeys2,
            openKeys,
            routeChange,
            onOpenChange,
            logout,
        }
    },
})
</script>

<style>
    .ant-layout-header {
        padding: 0 30px !important;
    }
    .ant-layout-content::-webkit-scrollbar {
        width:6px;
    }
    .ant-layout-content::-webkit-scrollbar-track {
        background-color:rgb(164, 162, 162);
    }
    .ant-layout-content::-webkit-scrollbar-thumb {
        background-color:#666;
    }
    .ant-layout-footer {
        padding: 5px 50px !important;
        color: rgb(31, 30, 30);
    }
    .is-collapse {
        display: none;
    }
    .ant-layout-sider {
        background: #141414 !important;
        overflow-y: auto;
    }
    .ant-layout-sider::-webkit-scrollbar {
        display: none;
    }
    .ant-dropdown-content {
        margin-top: 30px;
    }
    .ant-menu-item {
        margin: 0 !important;
```

优点知识

```
    }
</style>
```

2.4 面包屑的使用



- 使用面包屑处理当前页面深度展示，增加用户体验
- 相关知识点
  - `<a-breadcrumb-item>` 面包屑
  - 通过使用 `vue-router` 获取路由matched信息，实现菜单的深度展示
- 示例代码

```
<!-- 面包屑 -->
<a-breadcrumb style="margin: 16px 0">
    <a-breadcrumb-item>工作台</a-breadcrumb-item>
    <!-- router.currentRoute.value.matched表示路由的match信息，能拿到父路由和子路由的信息 --
>
    <template v-for="(matched,index) in router.currentRoute.value.matched"
:key="index">
        <a-breadcrumb-item v-if="matched.name">
            {{ matched.name }}
        </a-breadcrumb-item>
    </template>
</a-breadcrumb>
```

3. 整体代码

```
<template>
    <a-layout>
        <a-affix>
            <!-- 布局头部 -->
            <a-layout-header>
                <!-- 平台名 -->
                <div style="float:left;">
                    <img style="height:40px;margin-bottom:10px;" :src="kubeLogo" />
                    <span style="padding:0 50px 0 20px;font-size:25px;font-
weight:bold;color:#fff">KubeA</span>
                </div>
                <!-- 集群 -->
                <a-menu
                style="float:left;width:250px;"
                v-model:selectedKeys="selectedKeys1"
                theme="dark"
                mode="horizontal"
                :style="{ lineHeight: '64px' }">
                    <a-menu-item v-for="(item) in clusterList" :key="item">{{ item }}
</a-menu-item>
                </a-menu>
                <!-- 用户信息 -->
                <div style="float:right;">
                    <img style="height:40px;border-radius:50%;margin-right:10px;"
:src="avator"/>
                    <a-dropdown style="margin-top: 10px;">
                        <a class="ant-dropdown-link" @click.prevent>
                            Admin
                            <down-outlined />
                        </a>
                        <template #overlay>
                            <a-menu>
                                <a-menu-item>
                                <a @click="logout()">退出登录</a>
                                </a-menu-item>
                                <a-menu-item>
                                <a href="javascript:;">修改密码</a>
                                </a-menu-item>
                            </a-menu>
                        </template>
                    </a-dropdown>
                </div>
            </a-layout-header>
        </a-affix>
        <a-layout style="height:calc(100vh - 68px);">
            <!-- 侧边栏，核心 -->
            <!-- collapsed处理展开和收缩 -->
```

```
            <a-layout-sider width="240" style="background: #fff" v-
model:collapsed="collapsed" collapsible>
                <!-- selectedKeys表示点击选中的栏目,用于a-menu-item -->
                <!-- openKeys表示展开的栏目, 用于a-sub-menu -->
                <!-- openChange事件监听 SubMenu 展开/关闭的回调 -->
                <a-menu
                :selectedKeys="selectedKeys2"
                :openKeys="openKeys"
                @openChange="onOpenChange"
                mode="inline"
                :style="{ height: '100%', borderRight: 0 }">
                    <!-- routers是router/index.js中的路由信息 -->
                    <template v-for="menu in routers" :key="menu">
                        <!-- 处理无子路由的情况, 也就是单个栏目 -->
                        <!-- routeChange用于路由跳转 -->
                        <a-menu-item
                        v-if="menu.children && menu.children.length == 1"
                        :index="menu.children[0].path"
                        :key="menu.children[0].path"
                        @click="routeChange('item', menu.children[0].path)">
                            <component :is="menu.children[0].icon" />
                            <span>{{ menu.children[0].name }}</span>
                        </a-menu-item>
                        <!-- 处理有子路由的情况, 也就是父栏目 -->
                        <a-sub-menu
                        v-else-if="menu.children && menu.children.length > 1"
                        :index="menu.path"
                        :key="menu.path">
                            <template #title>
                                <span>
                                    <component :is="menu.icon" />
                                    <span :class="[collapsed ? 'is-collapse' : '']">{{
menu.name }}</span>
                                </span>
                            </template>
                            <!-- 子栏目（子路由）的处理 -->
                            <a-menu-item
                            v-for="child in menu.children"
                            :key="child.path"
                            :index="child.path"
                            @click="routeChange('sub', child.path)">
                                <span>{{ child.name }}</span>
                            </a-menu-item>
                        </a-sub-menu>
                    </template>
                </a-menu>
            </a-layout-sider>
            <a-layout style="padding: 0 24px">
                <!-- 面包屑 -->
                <a-breadcrumb style="margin: 16px 0">
```

```
                    <a-breadcrumb-item>工作台</a-breadcrumb-item>
                    <!-- router.currentRoute.value.matched表示路由的match信息，能拿到父路由和
子路由的信息 -->
                    <template v-for="(matched,index) in
router.currentRoute.value.matched" :key="index">
                        <a-breadcrumb-item v-if="matched.name">
                            {{ matched.name }}
                        </a-breadcrumb-item>
                    </template>
                </a-breadcrumb>
                <!-- main的部分 -->
                <a-layout-content
                :style="{
                    background: 'rgb(31, 30, 30)',
                    padding: '24px',
                    margin: 0,
                    minHeight: '280px',
                    overflowY: 'auto'}">
                    <!-- 路由占位符 -->
                    <router-view></router-view>
                </a-layout-content>
                <!-- footer部分 -->
                <a-layout-footer style="text-align: center">
                    ©2022 Created by adoo Devops
                </a-layout-footer>
            </a-layout>
        </a-layout>
    </a-layout>
</template>
<script>
import { onMounted, ref } from 'vue';
import { useRouter } from 'vue-router'
import kubeLogo from '@/assets/k8s-metrics.png';
import avator from '@/assets/avator.png';
export default ({
    setup() {
        const routers = ref([])
        const openKeys = ref([])
        const collapsed = ref(false)
        const selectedKeys1 = ref([])
        const selectedKeys2 = ref([])
        //集群列表
        const clusterList = ref([
            'TST-1',
            'TST-2'
        ])
        //使用useRouter方法获取路由配置以及当前路由信息
        const router = useRouter()
        //【方法】
        //导航栏点击切换页面
```

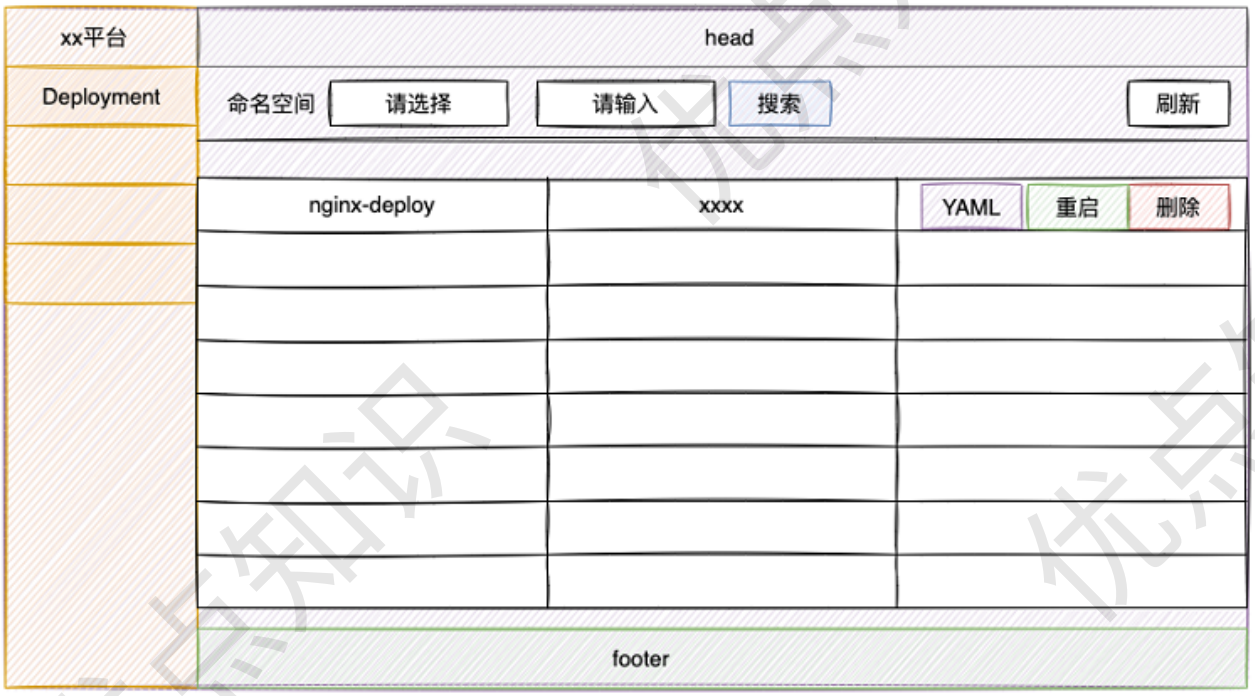
```
const routeChange =  (type, path) => {
    //判断点击的栏目是否为父栏目，如果不是，则关闭其他父栏目
    if (type !== 'sub') {
        openKeys.value = []
    }
    //表示选中当前path对应的栏目
    selectedKeys2.value = [path]
    if(router.currentRoute.value.path !== path){
        //获取路由对象并切换
        router.push(path)
    }
}
//根据url自动导航栏目高亮
const getRouter = (val) => {
    selectedKeys2.value = [val[1].path]
    openKeys.value = [val[0].path]
}
//处理菜单栏展开与关闭
const onOpenChange = (val) => {
    const latestOpenKey = val.find(key => openKeys.value.indexOf(key) === -1);
    openKeys.value = latestOpenKey ? [latestOpenKey] : [];
}
//退出登录
const logout = () => {
    //移除用户名
    localStorage.removeItem('username');
    //移除token
    localStorage.removeItem('token');
    //跳转至/login页面
    router.push('/login');
}
//生命周期钩子
onMounted(() => {
    routers.value = router.options.routes
    getRouter(router.currentRoute.value.matched)
})
//将变量和方法返回出去才能使用
return {
    kubeLogo,
    avator,
    collapsed,
    selectedKeys1,
    selectedKeys2,
    openKeys,
    router,
    routers,
    clusterList,
    routeChange,
    onOpenChange,
    logout,
```

```
        };
    },
});
</script>
<style>
    .ant-layout-header {
        padding: 0 30px !important;
    }
    .ant-layout-content::-webkit-scrollbar {
        width:6px;
    }
    .ant-layout-content::-webkit-scrollbar-track {
        background-color:rgb(164, 162, 162);
    }
    .ant-layout-content::-webkit-scrollbar-thumb {
        background-color:#666;
    }
    .ant-layout-footer {
        padding: 5px 50px !important;
        color: rgb(31, 30, 30);
    }
    .is-collapse {
        display: none;
    }
    .ant-layout-sider {
        background: #141414 !important;
        overflow-y: auto;
    }
    .ant-layout-sider::-webkit-scrollbar {
        display: none;
    }
    .ant-dropdown-content {
        margin-top: 30px;
    }
    .ant-menu-item {
        margin: 0 !important;
    }
</style>
```

八、课后作业

- 独立完成局部布局，包括选择框、搜索框、刷新按钮以及Table表格

优点知识


xx平台
head
Deployment
命名空间
请选择
请输入
搜索
刷新
nginx-deploy
xxxx
YAML
重启
删除
footer

优点知识