Pinia快速入门

什么是Pinia

Pinia 是 Vue 的专属的最新状态管理库 ， 是 Vuex 状态管理工具的替代品

1. 提供更加简单的API （去掉了 mutation ）
2. 提供符合组合式风格的API （和 Vue3 新语法统一）
3. 去掉了 modules 的概念，每一个 store 都是一个独立的模块
4. 搭配 TypeScript 一起使用提供可靠的类型推断

添加Pinia到Vue项目

1. 使用 create-vue 创建空的新项目

   npm init vue@latest

2. 按照官方文档安装 pinia 到项目

使用Pinia实现计数器案例

```js
import { defineStore } from 'pinia'
import { ref } from 'vue'
export const useCounterStore = defineStore('counter', () => {
  // 数据 (state)
  const count = ref(0)

  // 修改数据的方法  (action)
  const increment = () => {
    count.value++
  }

  // 以对象形式返回
  return { count, increment }
})
```

定义Store（state + action）

```vue
<script setup>
// 1. 导入 `useCounterStore` 方法
import { useCounterStore } from '@/stores/counter'
// 2. 执行方法得到counterStore对象
const counterStore = useCounterStore()
</script>

<template>
  <button @click="counterStore.increment">
    {{ counterStore.count }}
  </button>
</template>

```

组件使用Store

getters实现

Pinia中的 getters 直接使用 computed函数 进行模拟

```
// 数据 (state)
const count = ref(0)

// getter
const doubleCount = computed(() => count.value * 2)
```

action如何实现异步

action中实现异步和组件中定义数据和方法的风格完全一致

```
const API_URL = 'http://geek.itheima.net/v1_0/channels'
// 准备数据 (state)
const list = ref([])
// 异步action
const loadList = async () => {
  const res = await axios.get(API_URL)
  list.value = res.data.data.channels
}
```

storeToRefs

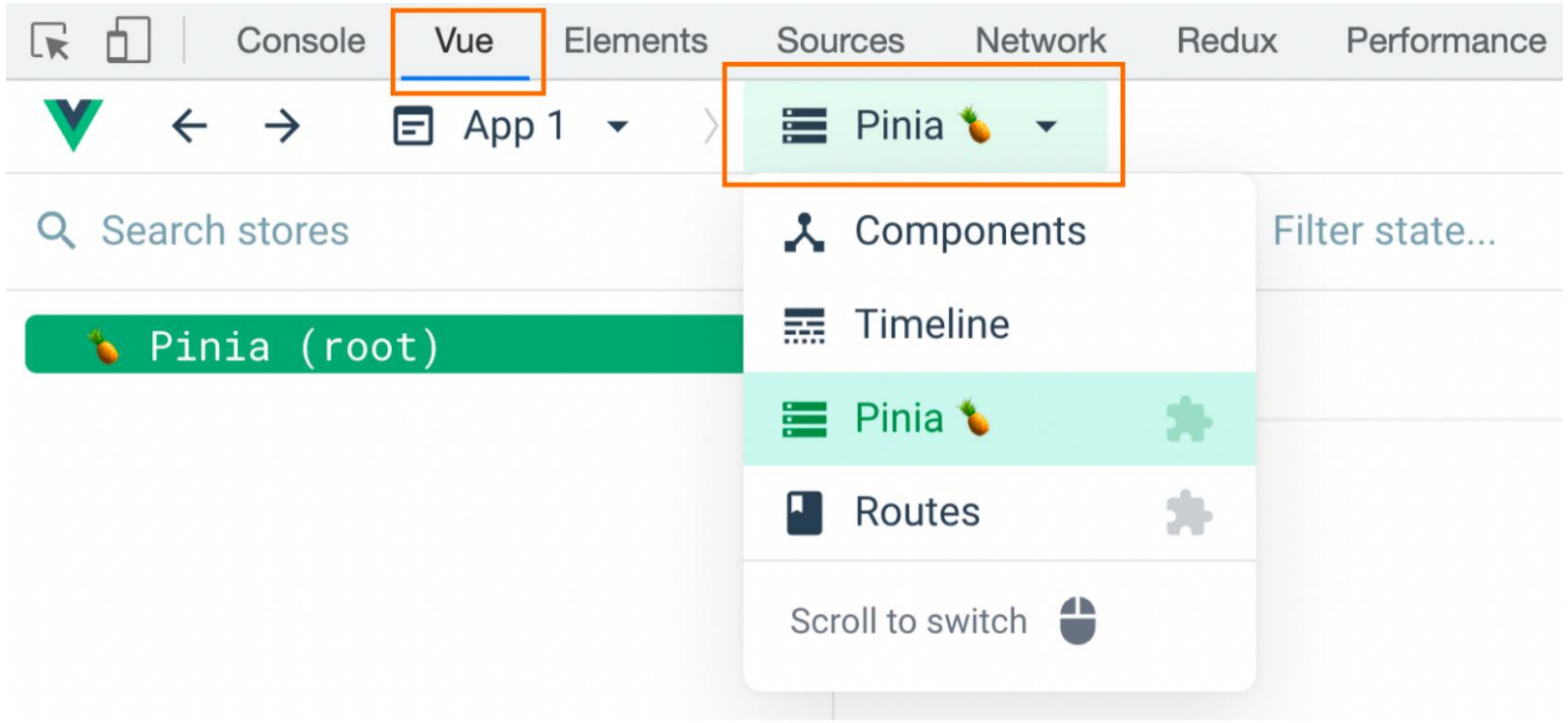
使用storeToRefs函数可以辅助保持数据（state + getter）的响应式解构

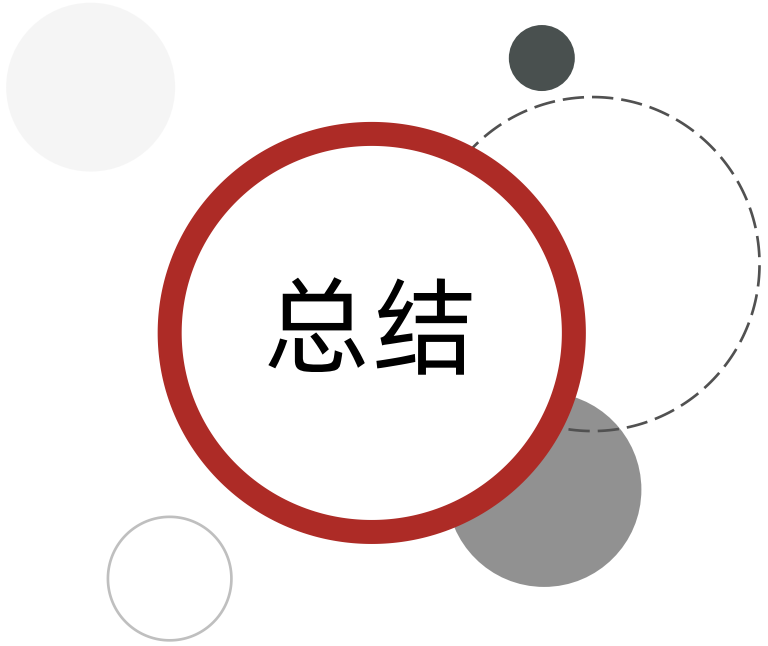
```
// 响应式丢失 视图不再更新
const { count, doubleCount } = counterStore
```

```
// 保持数据响应式
const { count, doubleCount } = storeToRefs(counterStore)
```

Pinia的调试

Vue官方的 dev-tools 调试工具 对 Pinia直接支持，可以直接进行调试





1. Pinia是用来做什么的？

   集中状态管理工具，新一代的vuex

2. Pinia中还需要mutation吗？

   不需要，action既支持同步也支持异步

3. Pinia如何实现getter？

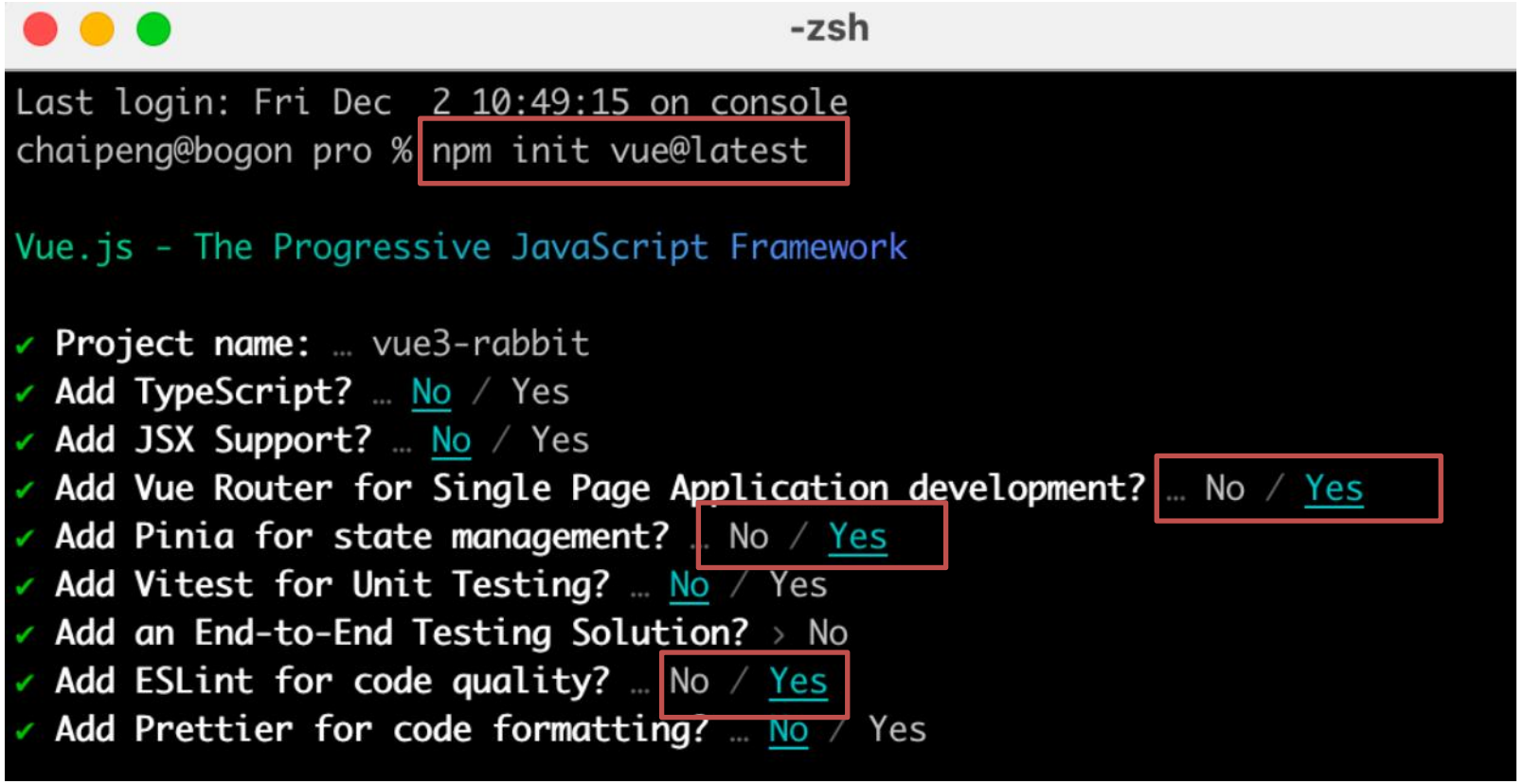
   computed计算属性函数

4. Pinia产生的Store如何解构赋值数据保持响应式？

   storeToRefs

项目起步–初始化项目并使用git管理

创建项目并精细化配置


-zsh
Last login: Fri Dec 2 10:49:15 on console
chaipeng@bogon pro % npm init vue@latest
Vue.js - The Progressive JavaScript Framework
✔ Project name: … vue3-rabbit
✔ Add TypeScript? … No / Yes
✔ Add JSX Support? … No / Yes
✔ Add Vue Router for Single Page Application development? … No / Yes
✔ Add Pinia for state management? … No / Yes
✔ Add Vitest for Unit Testing? … No / Yes
✔ Add an End-to-End Testing Solution? › No
✔ Add ESLint for code quality? … No / Yes
✔ Add Prettier for code formatting? … No / Yes

src目录调整


src
apis
assets
components
composables
directives
router
stores
styles
utils
views
App.vue
main.js
API接口文件夹
组合函数文件夹
全局指令文件夹
全局样式文件夹
工具函数文件夹

git 管理项目

基于 create-vue 创建出来的项目默认没有初始化git仓库，需要我们手动初始化

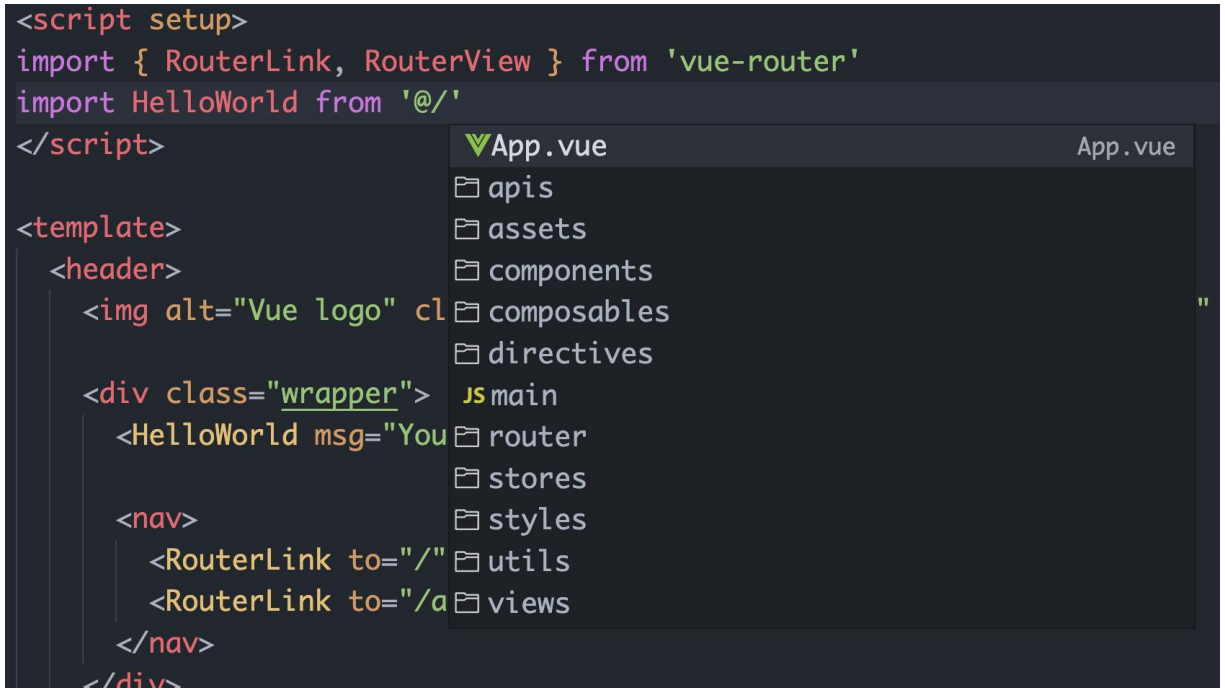
执行命令并完成首次提交

1. git init
2. git add .
3. git commit -m “init”

项目起步–配置别名路径联想提示

什么是别名路径联想提示

在编写代码的过程中，一旦 输入 @/ ，VSCode会立刻 联想出src下的所有子目录和文件, 统一文件路径访问不容易出错



如何进行配置

1. 在项目的根目录下新增 jsconfig.json 文件

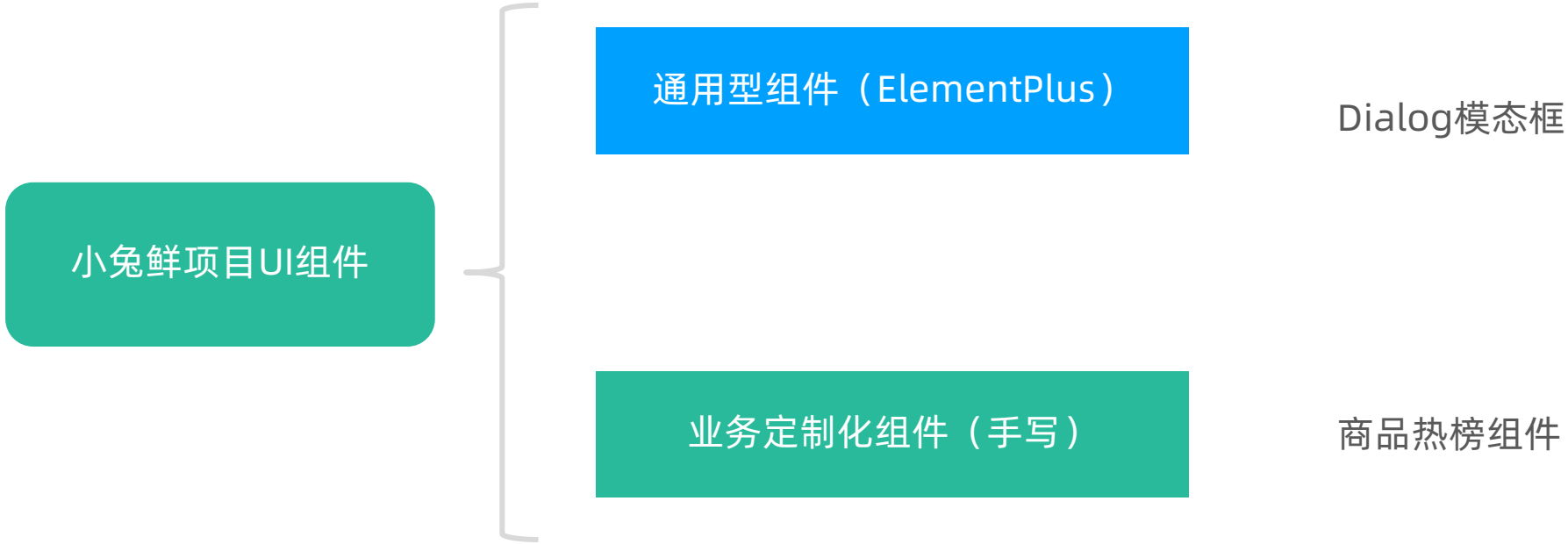
2. 添加json格式的配置项，如下：

```
{
  "compilerOptions": {
    "baseUrl": "./",
    "paths": {
      "@/*": [
        "src/*"
      ]
    }
  }
}
```

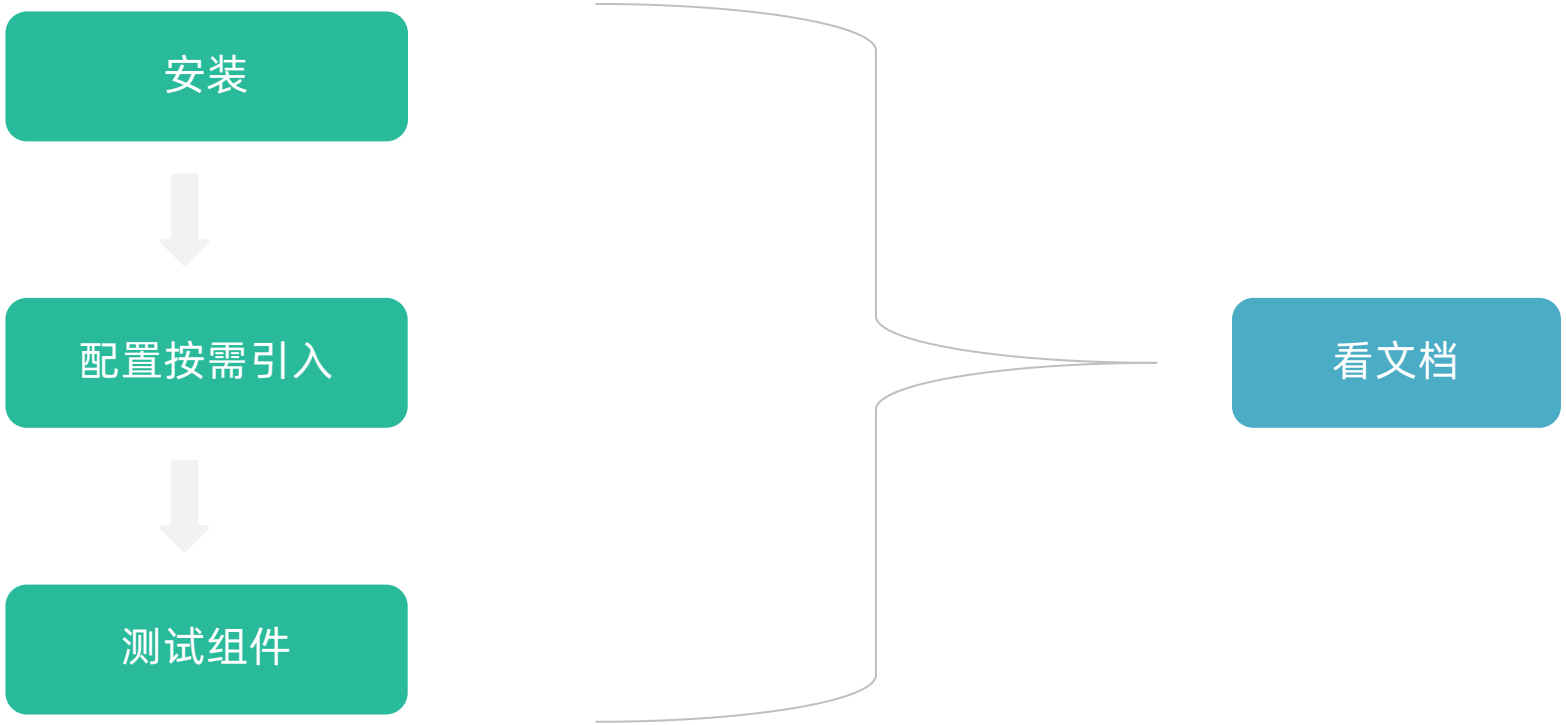
输入@ 自动联想src目录

项目起步–elementPlus引入

小兔鲜项目的组件分类



添加ElementPlus到项目（按需导入）



项目起步–elementPlus主题定制

为什么需要主题定制

小兔鲜主题色和elementPlus默认的主题色存在冲突，通过定制主题让elementPlus的主题色和小兔鲜项目保持一致

主色

小兔鲜儿主要品牌颜色为了表现新鲜、亲民，快捷的绿色

Brand Color
#27BA9B

辅助色

除了主色外的场景色，需要在不同的场景中使用（例如危险色表示危险的操作）。

辅助色1 #E26237
成功色 #1DC779
警示色 #FFB302
价格色 #CF4444

Color

Primary	Success	Warning	Danger	Info
#409EFF	#67C23A	#E6A23C	#F56C6C	#909399

Primary Text	Regular Text	Secondary Text	Placeholder
#303133	#606266	#909399	#C0C4CC

Border Base	Border Light	Border Lighter	Border Extralight
#DCDFE6	#E4E7ED	#EBEEF5	#F2F6FC

Background B	Background W	Background
#000000	#FFFFFF	Transparent

如何定制（scss变量替换方案）

安装sass

npm i sass -D

准备定制文件

styles/element/index.scss

对elementPlus样式进行覆盖

通知Element采用scss语言 -> 导入定制scss文件覆盖

如何验证主题替换成功

使用el-button按钮组件进行验证，type=”primary ”时显示主色，如果颜色变成了小兔鲜的主色，即为成功

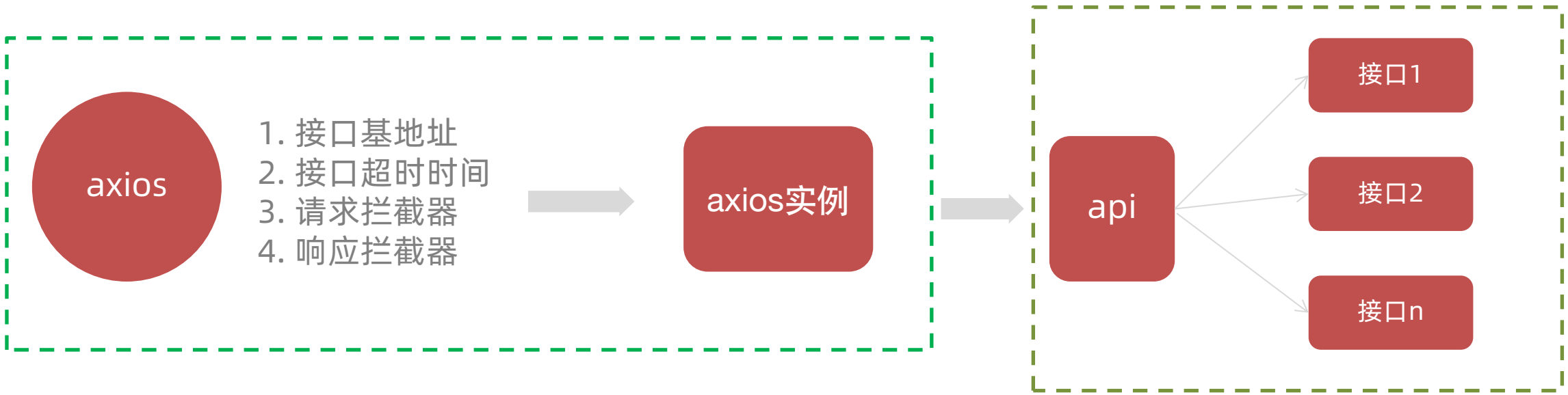
项目起步–axios基础配置

axios基础配置

1. 安装axios

npm i axios

2. 配置基础实例（统一接口配置）



如果项目里面不同的业务模块需要的接口基地址不同，

该如何来做？

axios.create() 方法可以执行多次，每次执行就会生成一个新的实例，比如：

```
1  const http1 = axios.create({ baseURL: 'url1' })
2  const http2 = axios.create({ baseURL: 'url2' })
```

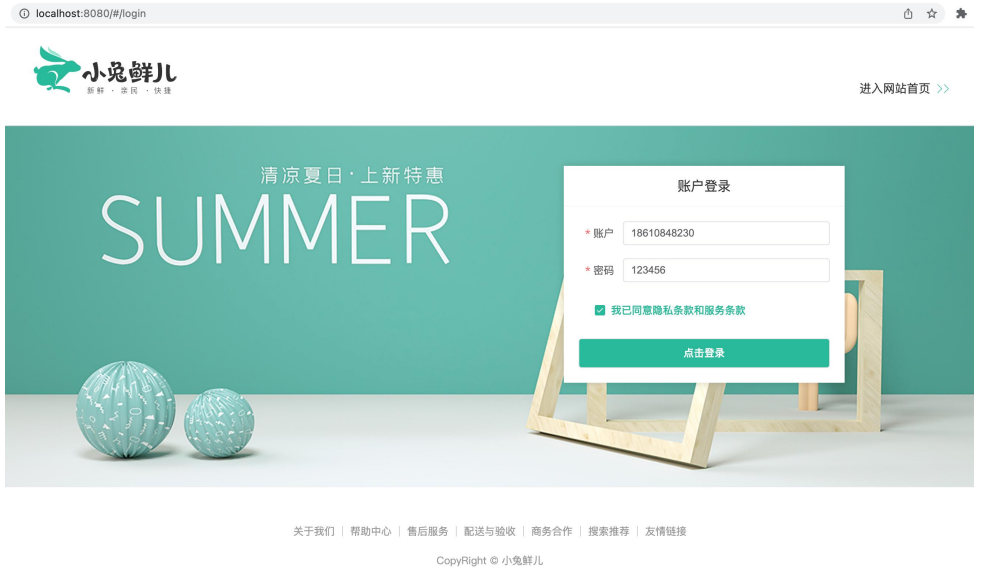
项目起步–项目路由设计

设计首页和登录页的路由（一级路由）

路由设计原则：找内容切换的区域，如果是页面整体切换，则为一级路由

路径 path：#/

路径 path: #/login

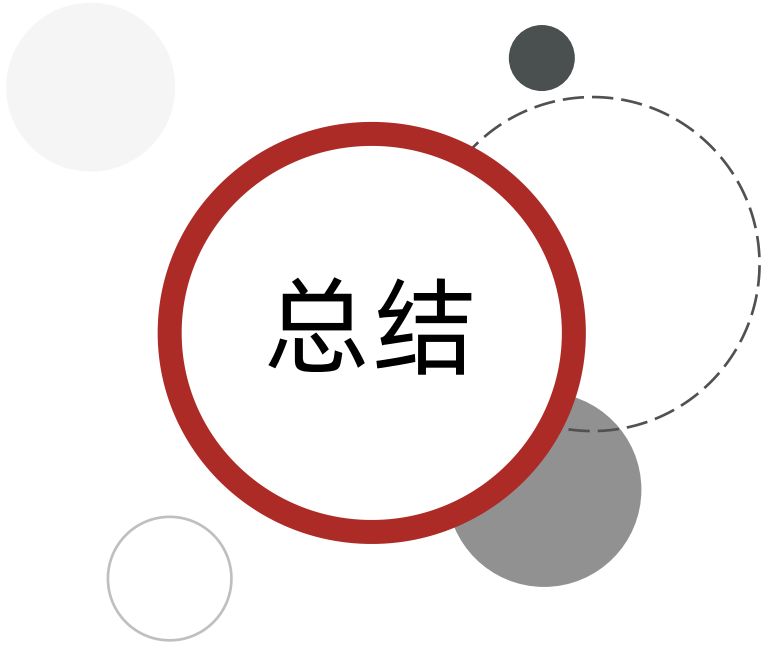


设计分类页和默认Home页路由（二级路由）

路由设计原则：找内容切换的区域，如果是在一级路由页的内部切换，则为二级路由

路径 path：#/

路径 path: #/category



1. 路由设计的依据是什么？

   内容切换的方式

2. 默认二级路由如何进行设置？

   path配置项置空

项目起步–静态资源初始化和 Error Lens 安装

图片资源和样式资源

资源说明

1. 实际工作中的图片资源通常由 UI设计师 提供，常见的图片格式有png,svg等都是由UI切图交给前端
2. 样式资源通常是指项目初始化的时候进行样式重置，常见的比如开源的 normalize.css 或者手写

资源操作

1. 图片资源 - 把 images 文件夹放到 assets 目录下
2. 样式资源 - 把 common.scss 文件放到 styles 目录下

error lens 安装

error lens是一个实时提供错误警告信息的VSCode插件，方便开发

项目起步–scss文件自动导入

为什么要自动导入

在项目里一些组件共享的色值会以scss变量的方式统一放到一个名为 var.scss 的文件中，正常组件中使用，需要先导入scss文件，再使用内部的变量，比较繁琐，自动导入可以免去手动导入的步骤，直接使用内部的变量

var.scss

src > styles > var.scss > ...

```scss
$xtxColor: #27ba9b;
$helpColor: #e26237;
$sucColor: #1dc779;
$warnColor: #ffb302;
$priceColor: #cf4444;
```


App.vue

src > App.vue > ...

```vue
<template>
  <div class="test">
    test
  </div>
</template>

<style lang="scss">
@import '@/styles/var.scss';

.test {
  color: $priceColor;
}
</style>
```

自动导入配置

1. 新增一个 var.scss 文件，存入色值变量
2. 通过 vite.config.js 配置自动导入文件

```
css: {
  preprocessorOptions: {
    scss: {
      // 自动导入scss文件
      additionalData: `
        @use "@/styles/element/index.scss" as *;
        @use "@/styles/var.scss" as *;
      `,
    }
  }
}
```

Layout–静态模版结构搭建

Layout模块静态模版搭建

Layout–字体图标引入

现在存在的小问题

价格亲民　　物流快捷　　品质新鲜

如何引入

阿里的字体图标库支持多种引入方式，小兔鲜项目里采用的是 font-class引用 的方式

```
<head>
  <meta charset="UTF-8" />
  <link rel="icon" href="/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="//at.alicdn.com/t/font_2143783_iq6z4ey5vu.css">
  <title>小兔鲜</title>
</head>
```

Layout–一级导航渲染

功能描述

使用后端接口渲染渲染一级路由导航

实现步骤

1. 根据接口文档封装接口函数
2. 发送请求获取数据列表
3. v-for渲染页面

Layout–吸顶导航交互实现

吸顶交互

要求：浏览器在上下滚动的过程中，如果距离顶部的滚动距离大于78px，吸顶导航显示，小于78px隐藏

准备吸顶导航组件 → 获取滚动距离 → 以滚动距离做判断条件控制组件盒子展示隐藏

```
import { useScroll } from '@vueuse/core'
const { y } = useScroll(window)
```

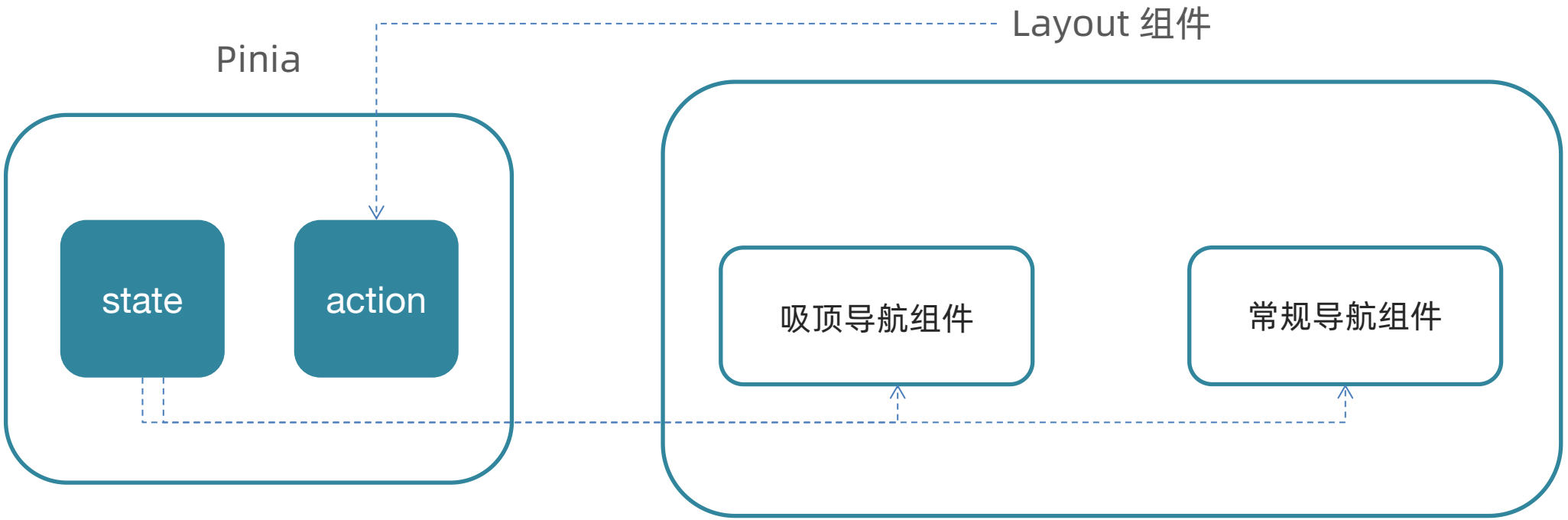
```
<div class="app-header-sticky" :class="{ show: y > 78 }">
    <!-- 省略 -->
 </div>
```

Layout–Pinia优化重复请求

为什么要优化

结论：俩个导航中的列表是完全一致的，但是要发送俩次网络请求，存在浪费。通过Pinia集中管理数据，再把数据给组件使用

如何优化



传智教育旗下高端IT教育品牌