

任务二 Vue-cli&ElementUI

1.Vue-cli

1.1 什么是Vue-cli

- Vue cli是基于Vue的应用开发提供的一个标准的脚手架工具.为应用搭建基础的框架结构,提供插件、开发服务、Preset、构建打包功能
- Vue cli 背后集成了现代化开发的诸多功能,通过简单的命令就可以完成 "零配置"的项目环境搭建

1.2 安装Vue-cli步骤

在安装vue-cli前, 要确认自己的电脑是否安装了nodejs和npm.

1.2.1 安装Node.js

安装了node.js才有使用npm, 才能安装vue-cli

1.2.1.1 什么是node.js

 本词条由“科普中国”科学百科词条编写与应用工作项目 审核。

Node.js 是一个基于 Chrome V8 引擎的 **JavaScript** 运行环境。Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型。 ^[1]

Node 是一个让 JavaScript 运行在**服务端**的开发平台, 它让 JavaScript 成为与**PHP**、**Python**、**Perl**、**Ruby** 等服务端语言平起平坐的**脚本语言**。 ^[2] 发布于2009年5月, 由Ryan Dahl开发, 实质是对Chrome V8引擎进行了封装。

Node对一些特殊用例进行优化, 提供替代的**API**, 使得V8在非浏览器环境下运行得更好。V8引擎执行**Javascript**的速度非常快, 性能非常好。Node是一个基于**Chrome JavaScript**运行时建立的平台, 用于方便地搭建响应速度快、易于扩展的网络应用。Node 使用**事件驱动**, 非阻塞**I/O** 模型而得以轻量 and 高效, 非常适合在分布式设备上运行数据密集型的实时应用。

为什么会有node.js?


传统意义上的 JavaScript 运行在浏览器上, Chrome 使用的 JavaScript 引擎是 V8, Node.js 是一个运行在服务端 的框架, 它的底层就使用了 V8 引擎, 这样就可以使用javascript去编写一些服务端的程序, 这样也就实现了用 javascript去开发 Apache + PHP 以及 Java Servlet所开发的服务端功能, 这样做的好处就是前端和后端都采用 javascript, 即开发一份js程序即可以运行在前端也可以运行的服务端, 这样比一个应用使用多种语言在开发效率上 要高, 不过node.js属于新兴产品, 一些公司也在尝试使用node.js完成一些业务领域, node.js基于V8引擎, 基于 事件驱动机制, 在特定领域性能出色, 比如用node.js实现消息推送、状态监控等的业务功能非常合适。

1.2.1.2 安装node.js

1) 下载对应你系统的Node.js版本:

<https://nodejs.org/en/download/>

我们统一安装: node-v12.18.1-x64.msi

 node-v12.18.1-x64.msi	2020/6/24 15:49	Windows Installer ...	19,292 KB
---	-----------------	-----------------------	-----------

2) 选安装目录进行安装, 我选择安装在了E盘: E:\Program Files\nodejs

3) 测试: 在命令提示符下输入命令

```
node -v //会显示当前node的版本
```

1.2.2 安装NPM

npm全称Node Package Manager，他是node包管理和分发的工具，使用NPM可以对应用的依赖进行管理，NPM 的功能和服务端项目构建工具maven的依赖管理功能差不多，我们通过npm 可以很方便地下载js库，打包js文件。

1.2.2.1 自动安装NPM

node.js已经集成了npm工具

在命令提示符输入 `npm -v` 可查看当前npm版本

```
npm -v
```

```
C:\Users\86187>npm -v  
6.14.5
```

1.2.2.2 查看包管理路径

包路径就是npm从远程下载的js包所存放的路径。使用 `npm config ls` 查询NPM管理包路径（NPM下载的依赖包所存放的路径）

```
npm config ls
```

我们发现NPM默认的管理包路径在：

```
C:\Users\86187\AppData\Roaming\npm
```

1.2.2.3 设置包管理路径

依赖包放在C盘不太合适,为了方便对依赖包管理,我们将管理包的路径设置在单独的地方：

1. 我们选择一个路径,专门存放这些依赖包.我选择创建一个目录: **H:\software\nodejs_package**
2. 在 **H:\software\nodejs_package** 下再创建 `npm_modules` 文件夹 和 `npm_cache` 文件夹:

▶ APTECH (H:) > software > nodejs_package

名称

📁 npm_cache

📁 npm_modules

3. 执行下边的命令,设置为自定义的包管理路径:

```
npm config set prefix "H:\software\nodejs_package\npm_modules"  
npm config set cache "H:\software\nodejs_package\npm_cache"
```

4. 此时再使用 `npm config ls` 查询NPM管理包路径发现路径已更改

```
C:\Users\86187>npm config ls
; cli configs
metrics-registry = "https://registry.npm.taobao.org/"
scope = ""
user-agent = "npm/6.14.5 node/v12.18.1 win32 x64"

; userconfig C:\Users\86187\.npmrc
cache = "H:\\software\\nodejs_package\\npm_cache"
prefix = "H:\\software\\nodejs_package\\npm_modules"
prefix = "E:\\package\\nodejs\\npm_modules"
registry = "https://registry.npm.taobao.org/"

; builtin config undefined

; node bin location = H:\software\nodejs\node.exe
; cwd = C:\Users\86187
; HOME = C:\Users\86187
; "npm config ls -l" to show all defaults.
```

CNPM淘宝镜像
提高下载速度

NPM的管理包路径,就是从npm
远程下载的JS包存放的路径

1.2.2.4 NPM环境变量配置

1) 查看npm的全局路径是什么

```
npm config get prefix
```

```
C:\Users\86187>npm config get prefix
H:\software\nodejs_package\npm_modules
```

2) 配置PATH环境变量

- 添加新的系统变量: **key=NODE_HOME** , **value= H:\software\nodejs_package**
- path中添加 `%NODE_HOME%\npm_modules`

%NODE_HOME%\npm_modules

编辑文本(T)...

确定

取消

1.2.2.5 安装cnpm

npm默认会去国外的镜像去下载js包, 在开发中通常我们使用国内镜像, 这里我们使用淘宝镜像

下边我们来安装cnpm：有时我们使用npm下载资源会很慢，所以我们可以安装一个cnpm(淘宝镜像)来加快下载速度。

1) 联网情况下, 输入命令, 进行全局安装淘宝镜像:

```
//安装
npm install -g cnpm --registry=https://registry.npm.taobao.org

//查看cnpm的版本
cnpm -v
```

1.2.3 安装vue-cli

目前主流版本是 2.x 和 3.x 版本,安装3.x 以上的版本是因为该版本既可以创建2.x项目与3.x 项目

注意: 以管理员身份打开命令行

1) 安装命令

```
npm install -g @vue/cli
```

2) 输入 vue命令

```
C:\Users\86187>vue
Usage: vue <command> [options]

Options:
  -V, --version                output the version number
  -h, --help                   output usage information
```

3) 输入 vue -V 查看版本

```
vue -V
```

```
C:\Users\86187>vue -V
@vue/cli 4.4.6
```

1.3 快速构建Vue项目

1.3.1 步骤说明


我们使用vue-cli 快速构建项目,步骤如下:

1. 桌面创建一个空的文件夹



vueTest

2. 以管理员身份运行cmd , 进入到vueTest文件夹

 管理员: 命令提示符

```
C:\Users\86187\Desktop\vueTest>
```

3. 执行下面的命令

1. 基于交互式命令方式, 创建项目
// 文件名 不支持驼峰 (含大写字母) 使用短横线方式
`vue create my-project`

```
C:\Users\86187\Desktop\vueTest>vue create my-project
```

4. 选择自定义安装, 点击回车

 npm

Vue CLI v4.4.6

? Please pick a preset:

default (babel, eslint)

> Manually select features

使用上下方向键进行选择,
选择第二项: 自定义安装

5. 在这列表中, 选择我们要安装的组件, 使用空格键选择, 选好后回车

? Check the features needed for your project:

```
(*) Babel
() TypeScript
() Progressive Web App (PWA) Support
(*) Router 选择安装路由
() Vuex
() CSS Pre-processors
> ( ) Linter / Formatter 取消掉语法检查
() Unit Testing
() E2E Testing
```

6. 按回车之后,提示选择什么模式的路由,我们输入 n (表示选择hash模式)

```
npm
Vue CLI v4.4.6
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Linter
? Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n) n
```

7. 选择项目配置文件单独存放

```
? Where do you prefer placing config for Babel, ESLint, etc.?
> In dedicated config files 选择将配置文件单独放置
In package.json
```

8. 是否保存模板,选择n 不创建

```
? Save this as a preset for future projects? (y/N) n
```

9. 安装完成,提示输入执行下面这两个命令

```
💎 Successfully created project my-project.
💎 Get started with the following commands:
```

```
$ cd my-project
$ npm run serve
```

安装成功,一次执行这两个命令

10. 首先进入项目目录

```
cd my-project
```

11. 启动项目

```
npm run serve
```

```
INFO Starting development server...
98% after emitting CopyPlugin

DONE Compiled successfully in 2708ms

App running at:
- Local: http://localhost:8080/
- Network: http://192.168.2.102:8080/
```

12. 访问项目: <http://localhost:8080/>



Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,
check out the [vue-cli documentation](#).

Installed CLI Plugins

[babel](#) [eslint](#)

Essential Links

[Core Docs](#) [Forum](#) [Community Chat](#) [Twitter](#) [News](#)

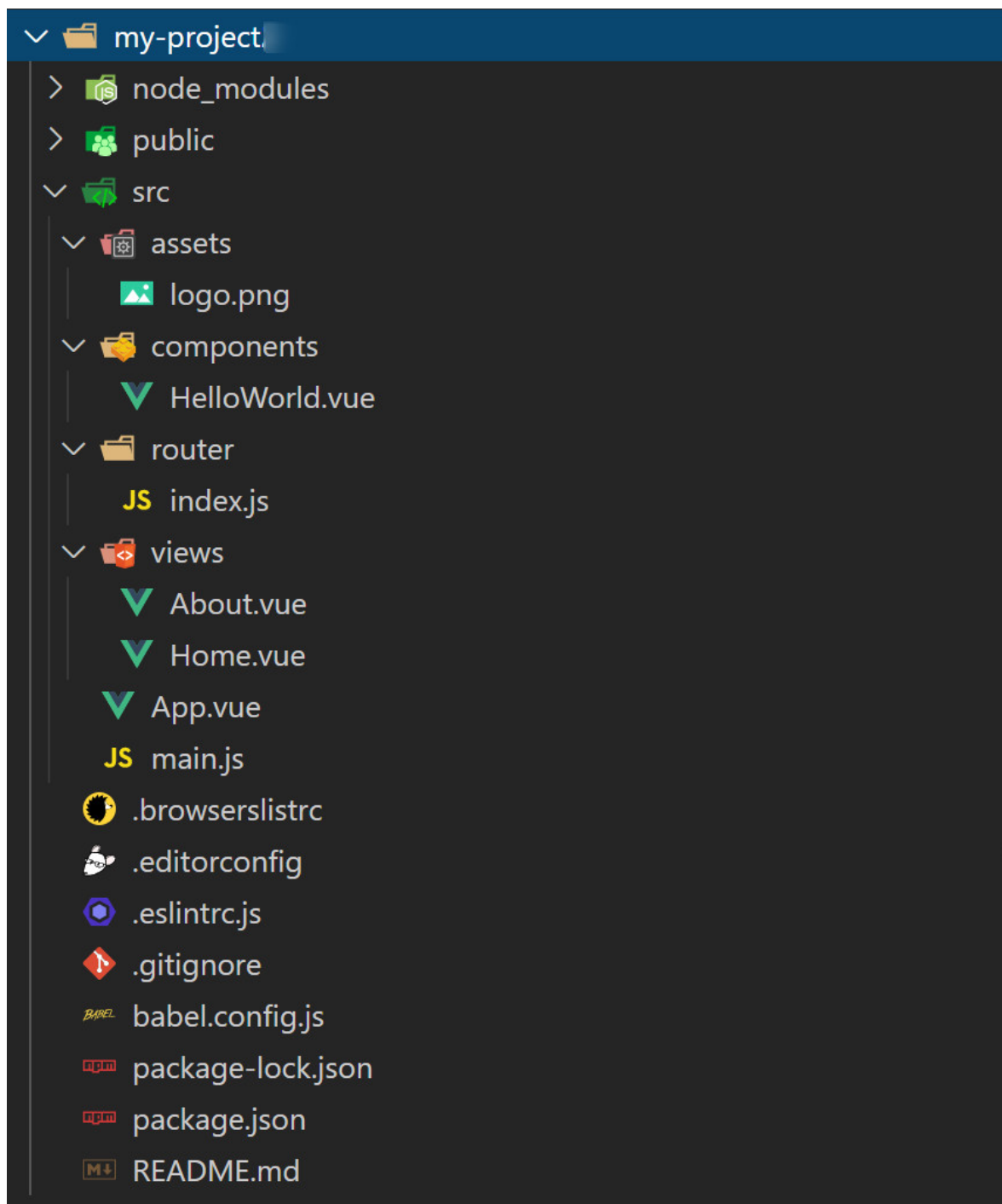
Ecosystem

[vue-router](#) [vuex](#) [vue-devtools](#) [vue-loader](#) [awesome-vue](#)

13. 停止项目 只要关闭命令行窗口就可以

1.3.2 导入Vue项目到VSCode

1. VSCode中右键选择打开文件夹
2. 选择桌面上的项目
3. 打开项目,可以看到如下项目结构



1.3.3 项目结构介绍

```
|--- my-project 项目名称
|--- node_modules 存放依赖包的目录
|--- public 静态资源管理目录
|--- src 组件源码目录(我们写的代码)
|   |--- assets 存放静态图片资源(CSS也可以放在这里)
|   |--- components 存放各种组件(一个页面可以看做一个组件)，各个组件联系在一起组成一个完整的项目
|   |--- router 存放了项目路由文件
|   |--- views 放置的为公共组件(主要还是各个主要页面)
|   |--- App.vue app.vue可以当做是网站首页，是一个vue项目的主组件，页面入口文件
|   |--- main.js 打包运行的入口文件，引入了vue模块和app.vue组件以及路由route
|--- babel.config.js babel配置文件，对源代码进行转码(把es6=>es5)
|--- package.json 项目及工具的依赖配置文件
|--- package-lock.json 依赖配置文件
|--- README.md 项目说明
```


1.3.4 Vue脚手架自定义配置

1.3.4.1 package.js 介绍

每个项目的根目录下面，一般都有一个 `package.json` 文件，定义了这个项目所需要的各种模块，以及项目的配置信息（比如名称、版本、许可证等元数据）。`npm install` 命令根据这个配置文件，自动下载所需的模块，也就是配置项目所需的运行和开发环境。

```
{
  //1.项目基本信息
  "name": "project3",
  "version": "0.1.0",
  "private": true,

  //2.指定运行脚本命令
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build"
  },

  //4.生产环境所依赖模块的版本
  "dependencies": {
    "core-js": "^3.6.5",
    "vue": "^2.6.11",
    "vue-router": "^3.2.0"
  },

  //5.本地环境开发所依赖的版本
  "devDependencies": {
    "@vue/cli-plugin-babel": "~4.4.0",
    "@vue/cli-plugin-router": "~4.4.0",
    "@vue/cli-service": "~4.4.0",
    "vue-template-compiler": "^2.6.11"
  }
}
```

1.3.4.2 通过package.json 配置项目

配置内容采用JSON格式,所有的内容都用双引号包裹

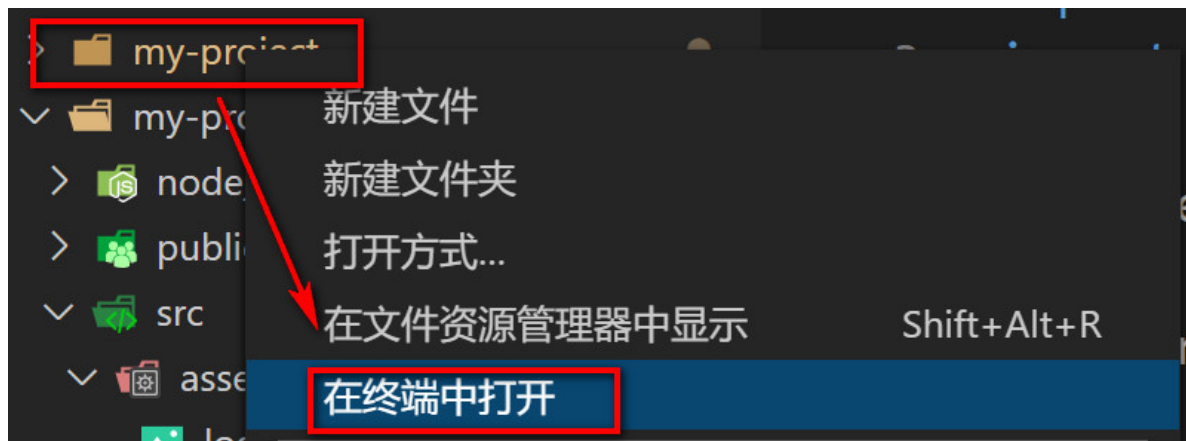
打开package.json,再最末端添加如下配置:

```
"vue":{
  "devServer":{
    "port":"8888",
    "open":true
  }
}
```

- 配置说明: 该配置设置打包时服务器相关的信息
 - port : 访问端口
 - open true: 打包完成自动打开浏览器

启动项目

1. 在VSCode中选择项目,右键在终端打开



2. 输入命令

```
npm run serve
```

3. 运行后发现端口号改为 8888,并且在打包完成后自动打开浏览器

localhost:8888/#/

注意: 不推荐这种方式,因为package.json 主要是用来管理包的配置信息.为了方便维护,我们将Vue脚手架相关的配置单独定义到 **vue.config.js** 配置文件中

1.3.4.3 单独的配置文件配置项目

1. 在项目的根目录创建文件 **vue.config.js**
2. 删除掉package中新添加的配置项.
3. 在vue.config.js 文件中进行相关配置

```
module.exports = {  
  devServer: {  
    open: true  
    port: 8889  
  }  
}
```

1.3.5 Vue 组件化开发

1.3.5.1 组件化开发

组件化是Vue的精髓,Vue项目就是由一个一个的组件构成的。我们主要的工作就是开发的组件。

1.3.5.2 组件介绍

1) 我们用 vue-cli 脚手架搭建的项目,里面有很多,如 index.vue 或者 App.vue 这一类的文件。

每一个*.vue 文件都是一个组件,是一个自定义的文件类型,比如 App.vue 就是整个项目的根组件。

2) 常见的组件:

- **页面级别的组件**
 - 页面级别的组件，通常是 `views` 目录下的 `.vue` 组件，是组成整个项目的各个主要页面
- **业务上可复用的基础组件**
 - 这一类组件通常是在业务中被各个页面复用的组件，这一类组件通常都写到 `components` 目录下，然后通过 `import` 在各个页面中使用

3) 组件的组成部分

- **template** : 组件的HTML部分
- **script**: 组件的JS脚本 (使用ES6语法编写)
- **style**: 组件的CSS样式

```
<!-- 1.template 代表html结构，template中的内容必须有且只有一个根元素
      编写页面静态部分 就是 view部分 -->
<template>
  <div>
    测试页面...
  </div>
</template>

<!-- 2.编写vue.js代码 -->
<script>
  //可以导入其组件
  // import Header from '../components/header.vue'

  //默认写法，输出该组件
  export default {
    name: "Home", // 组件名称，用于以后路由跳转
    data() { // 当前组件中需要使用的数据
      return {}
    },
    methods: {}
  }
</script>

<!-- 编写当前组件的样式代码 -->
<style scoped>
  /* 页面样式 加上scoped 表示样式就只在当前组件有效*/
</style>
```

1.4 项目运行流程

1.4.1 main.js

1. 项目运行 会加载入口文件 `main.js`

```
/*
```

html文件中,通过script src = 'xxx'标签引入js文件。

而vue中,通过 import 变量名 from 文件路径 的方式导入文件,不光可以导入js文件。

1. 变量名: 指的是为导入的文件起一个名称,不是指导入的文件的名称,相当于变量名。
2. 文件路径: 指的是文件的相对路径

```
*/
```

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'

//关闭启动提示
Vue.config.productionTip = false

//创建Vue实例
new Vue({
  router, //为整个项目添加路由
  render: h => h(App) //这是一个函数ES6语法,作用是生成模板: App = App.vue
}).$mount('#app') //挂载的是App.vue组件中的id为app的区域
```

1.4.2 App.vue

2. App.vue 是vue项目的主组件,是页面入口文件,所有页面都是在App.vue下进行切换的

App.vue 中的模板(HTML代码)

```
<template>
  <div id="app"> 挂载的是这个div

    <div id="nav">
      这里是两个路由导航链接
      1. to="/" 项目根路径 跳转的是首页
      <router-link to="/">Home</router-link> |
      2. to="/about" 点击About按钮,跳转到about组件
      <router-link to="/about">About</router-link>
    </div>

    router-view 的作用是 根据访问的路径,渲染路径匹配到的视图组件
    <router-view/>
  </div>
</template>
```

1.4.3 router 路由

3. 找到路由文件,来看一下具体的路由配置

```
// 引入所需文件
import Vue from 'vue' //vue库
import VueRouter from 'vue-router' //vue-router库
import Home from '../views/Home.vue' //首页

//使用路由功能
Vue.use(VueRouter)

//创建路由规则
const routes = [
  {
```

```

    path: '/', //路径
    name: 'Home', //名称
    component: Home //组件 Home.vue
  },
  {
    path: '/about',
    name: 'About',
    component: () => import(/* webpackChunkName: "about" */
'../views/About.vue')
  }
]

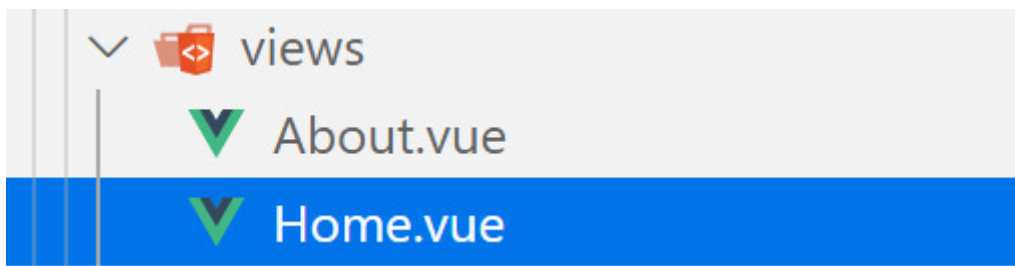
//创建路由管理器,管理routes
const router = new VueRouter({
  routes
})

//export 用来导出模块 router就代表了整个路由文件
export default router

```

1.4.4 Home.vue组件

1. 默认访问的是Home.vue 首页



视图部分

```

<template>
  <div class="home">
    首页的logo
    
    <HelloWorld msg="welcome to Your Vue.js App"/>
  </div>
</template>

```

JS部分

```

<script>

//导入了一个组件 HelloWorld.vue @符号表示 src这个目录
import HelloWorld from '@components/HelloWorld.vue'

export default {
  name: 'Home',
  components: {
    HelloWorld
  }
}
</script>

```

HelloWorld.vue 组件页面

```

<template>
  <div class="hello">
    <h1>{{ msg }}</h1>
    <p>
      For a guide and recipes on how to configure / customize this project,<br>
      check out the
      <a href="https://cli.vuejs.org" target="_blank" rel="noopener">vue-cli
documentation</a>.
    </p>
    <h3>Installed CLI Plugins</h3>
  </div>
</template>

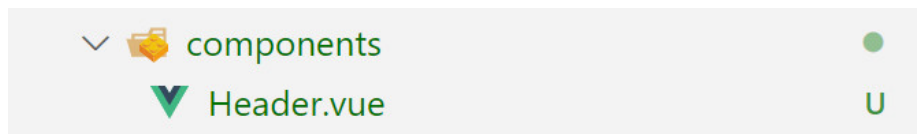
<script>
export default {
  name: 'HelloWorld',
  props: {
    msg: String
  }
}
</script>

```

1.5 组件的使用案例

1.5.1 创建Header.vue组件

1. 在components 目录下创建 Header.vue



2. 编写Header.vue

```

<template>
  <div class="header">{{msg}}</div>
</template>

<script>
//JS 部分
export default {
  name: "Header", //组件的名称
  data() { //data函数
    return {
      msg: "这是一个Header组件"
    }
  },
}
</script>

//scoped 表示当前的样式,只作用与当前组件中的 template 视图.
<style scoped>
.header {
  height: 100px;
  line-height: 100px;
  background-color: #eee;
}

```

```
text-align: center;
color: blue;
}
</style>
```

1.5.2 引入 Header组件

修改Home.vue

```
<template>
  <div class="home">
    
    <!-- <HelloWorld msg="Welcome to Your Vue.js App"/> -->
    <Header/>
  </div>
</template>

<script>
// @ is an alias to /src
//import HelloWorld from '@components/HelloWorld.vue'
import Header from '@components/Header.vue';

export default {
  name: 'Home',
  components: {
    //HelloWorld
    Header
  }
}
</script>
```

1.5.3 组件的传参

- props : 是组件中的属性, 表示组件可以接受参数,

```
<template>
  <div id="Header" class="header">{{msg}}</div>
</template>

<script>
export default {
  name: "Header", //组件名称,用于路由的跳转
  props: ['msg']
};
</script>
```

// scoped 表示当前style的样式只作用于当前组件的template代码中, 其他地方不会被影响

```
<style scoped>
.header {
  height: 100px;
  line-height: 100px;
  background-color: #eee;
}
```



```
text-align: center;
color: blue;
}
</style>
```

2.Element-UI

2.1 Element-UI介绍

element-ui 是饿了么前端出品的基于 Vue.js 的 后台组件库，方便程序员进行页面快速布局和构建

Element-UI官方网站点：

<https://element.eleme.cn/#/zh-CN>

2.2 Element-UI使用

2.2.1 命令行方式安装

1. 创建 一个新的项目
2. 当前项目下打开终端, 安装依赖包 ,执行下面的命令

```
npm i element-ui -S
```

问题 输出 终端 调试控制台

```
PS C:\Users\86187\Desktop\vueTest\my-project2> npm i element-ui -S
npm WARN deprecated core-js@2.6.11: core-js@<3 is no longer maintained
e your dependencies to the actual version of core-js@3.
```

3. 打开 main.js , 导入Element-UI 相关资源.

- main.js是工程的入口文件，在此文件中加载了很多第三方组件，如：Element-UI、Base64、VueRouter等。

```
//导入组件库
import ElementUI from 'element-ui'

//导入组件相关样式
import 'element-ui/lib/theme-chalk/index.css'

//配置Vue插件 将El安装到Vue上
Vue.use(ElementUI);
```

JS main.js X

my-project2 > src > JS main.js > ...

```
5 //手动配置ElementUI
6 import ElementUI from "element-ui";
7 import "element-ui/lib/theme-chalk/index.css";
8 Vue.use(ElementUI);
```

4. 复制Element 按钮样式 到app.vue文件的 template下

```
<template>
  <div id="app">
    <!-- 测试elementUI -->
    <el-row>
      <el-button>默认按钮</el-button>
      <el-button type="primary">主要按钮</el-button>
      <el-button type="success">成功按钮</el-button>
      <el-button type="info">信息按钮</el-button>
      <el-button type="warning">警告按钮</el-button>
      <el-button type="danger">危险按钮</el-button>
    </el-row>

    <div id="nav">
      <router-link to="/">Home</router-link>|
      <router-link to="/about">About</router-link>
    </div>
    <router-view />
  </div>
</template>
```

4. 启动项目 npm run serve, 查看页面



2.2.2 Vue-CLI工程改造

1. 删除components 目录下的 HelloWorld.vue组件
2. 删除App.vue中的部分内容,只保留如下部分

```
<template>
  <div id="app"></div>
</template>

<style>
</style>
```

3. 删除router文件下的路由文件 index.js部分内容,只保留如下部分

```
import Vue from 'vue'
import VueRouter from 'vue-router'

Vue.use(VueRouter)

const routes = [

]
```

```
const router = new VueRouter({
  routes
})

export default router
```

4. 删除views目录下的 About.vue 与 Home.vue

2.2.3 安装axios

1. npm安装：使用npm下载axios包

```
npm i axios
```

2. 在main.js文件中导入axios 相关资源

```
//引入axios
import axios from 'axios'

//Vue对象使用axios
Vue.prototype.axios = axios;
```

2.3 用户登录界面制作

2.3.1 Dialog对话框组件

Others

Dialog 对话框

Dialog 对话框

在保留当前页面状态的情况下，告知用户并承载相关操作。

我们可以用Dialog制作一个登陆弹窗,选择自定义内容

自定义内容

Dialog 组件的内容可以是任意的，甚至可以是表格或表单，下面是应用了 Element Table 和 Form 组件的两个样例。

打开嵌套表格的 Dialog 打开嵌套表单的 Dialog

```
<el-dialog title="收货地址" :visible.sync="dialogFormVisible">
  <el-form :model="form">
    <el-form-item label="活动名称" :label-width="formLabelWidth">
      <el-input v-model="form.name" autocomplete="off"></el-input>
    </el-form-item>
    <el-form-item label="活动区域" :label-width="formLabelWidth">
      <el-select v-model="form.region" placeholder="请选择活动区域">
        <el-option label="区域一" value="shanghai"></el-option>
```

```

        <el-option label="区域二" value="beijing"></el-option>
      </el-select>
    </el-form-item>
  </el-form>
  <div slot="footer" class="dialog-footer">
    <el-button @click="dialogFormVisible = false">取 消</el-button>
    <el-button type="primary" @click="dialogFormVisible = false">确 定</el-
button>
  </div>
</el-dialog>

```

2.3.2 创建login.vue 组件

1. 在components 下创建login.vue
2. 将Diglog组件的内容,拷贝到login.vue,进行修改:

```

<template>
  <el-dialog title="登录" :visible.sync="dialogFormVisible">
    <el-form>
      <el-form-item label="用户名称" :label-width="formLabelWidth">
        <el-input autocomplete="off"></el-input>
      </el-form-item>
      <el-form-item label="用户密码" :label-width="formLabelWidth">
        <el-input autocomplete="off"></el-input>
      </el-form-item>
    </el-form>

    <div slot="footer" class="dialog-footer">
      <el-button type="primary" @click="dialogFormVisible = false">登录</el-
button>
    </div>
  </el-dialog>
</template>

<script>
export default {
  data() {
    return {
      formLabelWidth: "120px", //宽度
      dialogFormVisible: true
    };
  }
};
</script>

<style scoped>
</style>

```

2.3.3 配置路由

```

import Vue from "vue";
import VueRouter from "vue-router";

import Login from "@/components/Login.vue"
Vue.use(VueRouter);

```

```
const routes = [
  //访问 /,也跳转到login
  {
    path: '/',
    redirect: 'login' //重定向到login
  },
  //登录
  {
    path: '/login',
    name: 'login',
    component: Login
  }
];

const router = new VueRouter({
  routes,
});

export default router;
```

2.3.4 修改App.vue

```
<template>
  <div id="app">
    <!-- router-view 的作用是根据访问的路径,渲染路径匹配到的视图组件 -->
    <router-view></router-view>
  </div>
</template>

<style>
</style>
```

2.3.5 编写登录功能

1. 去掉关闭按钮, 添加一个属性 `:show-close="false"`

```
<el-dialog title="登录" :show-close="false" :visible.sync="dialogFormVisible">
```

2. 修改登陆触发事件

```
<el-button type="primary" @click="login">登录</el-button>
```

3. 双向数据绑定

- data 中定义数据

```
data() {
  return {
    formLabelWidth: "120px", //宽度
    dialogFormVisible: true, //是否关闭对话框
    user: { username: "", password: "" }, //登录数据
  };
},
```

- 使用 v-model, 将视图与模型进行绑定

```

<el-form>
  <el-form-item label="用户名称" :label-width="formLabelWidth">
    <el-input v-model="user.username" autocomplete="off"></el-input>
  </el-form-item>
  <el-form-item label="用户密码" :label-width="formLabelWidth">
    <el-input v-model="user.password" autocomplete="off"></el-input>
  </el-form-item>
</el-form>

```

4. 编写login方法

```

methods: {
  login() {
    //定义常量保存 url
    const url = "http";

    //发送请求
    this.$axios
      .get(url, {
        //携带参数
        params: {
          username: this.user.username,
          password: this.user.password,
        },
      })
      .then((res) => {
        console.log();
        //成功就将对话框关闭
        this.dialogFormVisible = false;
      })
      .catch((error) => {
        //出现错误使用ElementUI提供的消息提示
        this.$message.error("对不起！登录错误！");
      });
  },
},

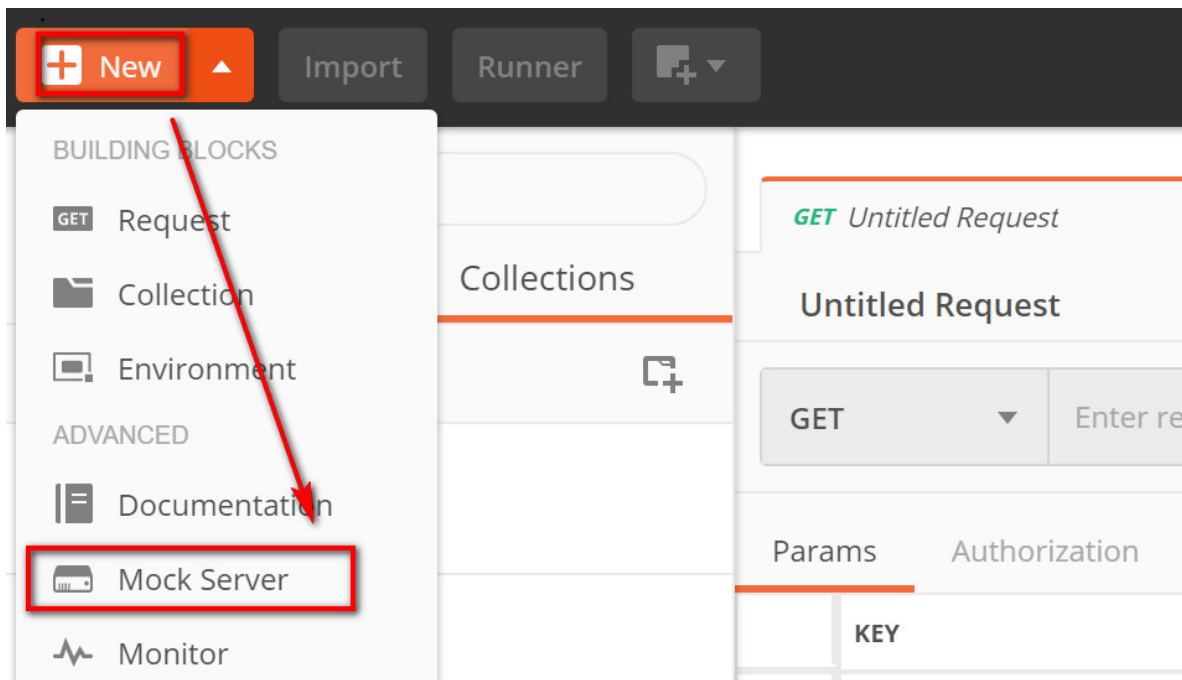
```

2.3.6 Postman搭建mock server

- Mock server就是模拟一个服务器，我们使用Mock server可以模拟后台接口,对请求进行响应.
- 在前后端分离的开发中 前端利用mockeserver模拟出对应接口，拿到返回数据来调试，无需等后端开发人员完成工作。

postman模拟出一个server 步骤:

1. 使用postman模拟出一个server

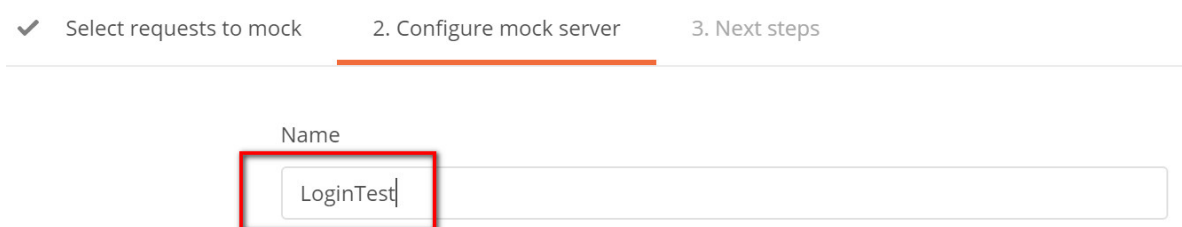


2. 打开如下窗体，创建一个伪服务

- 第一步



- 第二步



- 第三步

- 第四步 修改请求的URL

```
const url = "复制上面的地址/login";
```

2.3.7 登录成功后跳转

- 在js中设置跳转，常用的一种方法是 `this.$router.push`

```
methods: {  
  login() {  
    //定义常量保存 url  
    const url =
```



```

      "https://33284b33-e976-4124-a3a0-17044addc1e1.mock.pstmn.io/login";

      //发送请求
      this.axios
        .get(url, {
          //携带参数
          params: {
            username: this.user.username,
            password: this.user.password,
          },
        })
        .then((res) => {
          console.log(res.data);
          alert("登录成功!");
          //成功就将对话框关闭
          this.dialogFormVisible = false;

          //跳转页面,前端跳转页面必须使用路由,使用$router对象中的push方法
          this.$router.push('/index');
        })
        .catch((error) => {
          //出现错误使用ElementUI提供的消息提示
          this.$message.error("对不起! 登录错误!");
        });
    },
  },
},

```

2.4 首页布局页面制作

2.4.1 创建 index.vue

```

<template>
  <div>
    <el-button type="danger">布局页面</el-button>
  </div>
</template>

<script>
export default {
  }
</script>

<style scoped>

</style>

```

2.4.2 配置路由

router目录下的index.js 路由文件

```
//导入布局组件
import Index from "@/components/Index.vue"

//布局路由
{
  path: '/index',
  name: 'index',
  component: Index
}
];
```

2.4.3 布局容器

Container 布局容器,是用于布局的容器组件, 方便快速搭建页面的基本结构:

1. 在官方文档中找到布局的容器代码, 复制到 Index.vue

```
<el-container>
  <el-header>Header</el-header>
  <el-container>
    <el-aside width="200px">Aside</el-aside>
    <el-main>Main</el-main>
  </el-container>
</el-container>

<style scoped>
.el-container {
  height: 720px;
}
.el-header,
.el-footer {
  background-color: #b3c0d1;
  color: #333;
  text-align: center;
  line-height: 60px;
}

.el-aside {
  background-color: #d3dce6;
  color: #333;
  text-align: center;
  line-height: 200px;
}

.el-main {
  background-color: #e9eef3;
  color: #333;
  text-align: center;
  line-height: 30px;
}
</style>
```

2. 拷贝布局容器中的导航菜单代码, 进行修改,代码如下

```
<template>
  <div>
```

```

<el-container>
  <el-header>后台管理</el-header>
  <el-container>
    <!-- 侧边栏 -->
    <el-aside width="200px">
      <el-menu
        default-active="2"
        class="el-menu-vertical-demo"
        background-color="#d3dce6"
      >
        <el-submenu index="1">
          <template slot="title">
            <i class="el-icon-location"></i>
            <span>导航菜单</span>
          </template>
          <el-menu-item-group>
            <el-menu-item index="1-1"
              ><i class="el-icon-menu"></i>课程管理</el-menu-item
            >
          </el-menu-item-group>
        </el-submenu>
      </el-menu>
    </el-aside>

    <!-- 主要区域 -->
    <el-main>Main</el-main>
  </el-container>
</el-container>
</div>
</template>

<script>
export default {};
</script>

<style scoped>
.el-container {
  height: 725px;
}
.el-header,
.el-footer {
  background-color: #b3c0d1;
  color: #333;
  text-align: center;
  line-height: 60px;
}

.el-aside {
  background-color: #d3dce6;
  color: #333;
  text-align: center;
  line-height: 200px;
}

.el-main {
  background-color: #e9eef3;
  color: #333;
  text-align: center;

```

```
line-height: 160px;
}
</style>
```

2.5 课程列表组件制作

当我们点击导航菜单中的课程管理时,要显示课程信息



2.5.1 编写 Course.vue

```
<template>
  <el-button type="danger">课程信息</el-button>
</template>

<script>
export default {};
</script>

<style scoped></style>
```

2.5.2 配置路由

1.在index.js路由文件中, 为布局路由添加**children** 属性表示 子路由

```
//引入课程组件
import Course from "@/components/Course.vue"

//布局路由
{
  path: "/index",
  name: "index",
  component: Index,
  //添加子路由,使用 children属性 来表示子路由
  children:[
    //课程信息子路由
    {
      path:"/course",
      name:"course",
      component:Course
    }
  ]
},
```

2. 修改 Index.vue组件中的 导航菜单属性

router 表示是否使用 vue-router 的模式, 启用该模式会在激活导航时以 index 作为 path 进行路由跳转

el-menu中 添加一个 router属性

```
<el-menu default-active="2" class="el-menu-vertical-demo" background-color="#d3dce6" router >
```

3. 为index属性指定 路由

```
<el-menu-item-group>
  <!-- 修改 index的路由地址 -->
  <el-menu-item index="/course">
    <i class="el-icon-menu"></i>课程管理
  </el-menu-item>
</el-menu-item-group>
```

4. 设置路由的出口,将课程信息显示再 main

```
<!-- 主要区域 -->
<el-main>
  <router-view></router-view>
</el-main>
```

2.6 Table表格组件

我们通过table组件来实现一个课程页面展示的功能, 通过查看Element-UI库, 我们需要Table 表格. 进入Element-UI官方, 找到Table组件, 拷贝源代码到vue页面中, 如下

Data	Table 表格
Table 表格	用于展示多条结构类似的数据, 可对数据进行排序、筛选、对比或其他自定义操作。

2.6.1 添加表格组件

复制表格组件相关的代码到 Course.vue中

```
<template>
  <el-table :data="tableData" style="width: 100%">
    <el-table-column prop="date" label="日期" width="180"> </el-table-column>
    <el-table-column prop="name" label="姓名" width="180"> </el-table-column>
    <el-table-column prop="address" label="地址"> </el-table-column>
  </el-table>
</template>

<script>
export default {
  data() {
    return {
      tableData: [
        {
          date: "2016-05-02",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
```

```

    },
    {
      date: "2016-05-04",
      name: "王小虎",
      address: "上海市普陀区金沙江路 1517 弄"
    },
    {
      date: "2016-05-01",
      name: "王小虎",
      address: "上海市普陀区金沙江路 1519 弄"
    },
    {
      date: "2016-05-03",
      name: "王小虎",
      address: "上海市普陀区金沙江路 1516 弄"
    }
  ]
};
}
};
</script>

```

2.6.2 表格组件说明

我们查看一下,ElementUI的表格的代码,分析一下表格数据是如何显示的

```

//视图部分 进行页面展示
<template>
  //el-table组件 绑定了tableData数据
  <el-table :data="tableData" style="width: 100%">
    //el-table-column 表示表格的每列,prop属性与模型数据中的key对应 ,label 列名
    <el-table-column prop="date" label="日期" width="180"></el-table-column>
    <el-table-column prop="name" label="姓名" width="180"></el-table-column>
    <el-table-column prop="address" label="地址"></el-table-column>
  </el-table>
</template>

<script>
//export default 相当于提供一个接口给外界,让其他文件通过 import 来引入使用。
export default {
  //data() 函数
  data() {
    return {
      //数据部分
      tableData: [
        {
          date: "2016-05-02",
          name: "王小虎",
          address: "上海市普陀区金沙江路 1518 弄"
        }
      ]
    };
  }
};
</script>

```

2.7 课程内容展示

2.7.1 修改Course.vue

1.编写 template, 复制ElementUI的示例代码,进行改动

```
<template>
  <div class="table">
    <!-- ElementUI表格 -->
    <el-table
      style="width: 100%"
      border
      height="550"
      :data="courseList"
      v-loading="loading"
      element-loading-text="数据加载中..."
    >
      <el-table-column fixed="left" prop="id" label="ID"></el-table-column>
      <el-table-column prop="course_name" label="课程名称"></el-table-column>
      <el-table-column prop="price" label="价格"></el-table-column>
      <el-table-column prop="sort_num" label="排序"></el-table-column>
      <el-table-column prop="status" label="状态"></el-table-column>
    </el-table>
  </div>
</template>
```

3.编写VM部分代码

```
<script>
export default {
  name: "Course",
  title: "课程管理",
  //数据部分
  data() {
    return {
      //定义数据
      loading: false, //是否弹出加载提示
      courseList: [], //定义集合,保存从接口获取的参数
    };
  },

  //钩子函数,在DOM页面生成之前执行
  created() {
    //在页面生成之前, 调用loadCourse
    this.loadCourse();
  },

  //方法集合
  methods: {
    //方法1: 获取课程信息
    loadCourse() {
      //开启
      this.loading = true;

      //访问后台接口,获取数据并返回
      return this.$axios
        .get(
          "http://localhost:8080/lagou_edu_home/course?
          methodName=findCourseList"
        )
    }
  }
}
```



```

    )
    .then((res) => {
      console.log(res.data);
      //将获取到的数据赋值给 courseList
      this.courseList = res.data;
      this.loading = false;
    });
  },
},
};
</script>

```

2.7.2 跨域问题解决

2.7.2.1 出现跨域问题

当我们在前端项目中,向后端发送请求的获取课程数据的时候,出现了跨域问题:

- 已被CORS策略阻止: 请求的资源上没有'Access-Control-Allow-Origin'标头 (跨域请求失败)

Access to XMLHttpRequest at 'http://localhost:8080/lagou_edu_home/course?methodName=findCourseList' from origin 'http://localhost:8088' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

Access to XMLHttpRequest at 'http://localhost:8080/lagou_edu_home/course?methodName=findCourseList' from origin 'http://localhost:8088' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

2.7.2.2 什么是跨域

跨域是指通过JS在不同的域之间进行数据传输或通信, 比如用ajax向一个不同的域请求数据, 只要**协议、域名、端口**有任何一个不同, 都被当作是不同的域,浏览器就不允许跨域请求。

- 跨域的几种常见情

http://www.123.com/index.html 调用 http://www.123.com/server.PHP (非跨域)

http://www.123.com/index.html 调用 http://www.456.com/server.php (主域名不同:123/456, 跨域)

http://abc.123.com/index.html 调用 http://def.123.com/server.php (子域名不同:abc/def, 跨域)

http://www.123.com:8080/index.html 调用 http://www.123.com:8081/server.php (端口不同:8080/8081, 跨域)

http://www.123.com/index.html 调用 https://www.123.com/server.php (协议不同:http/https, 跨域)

2.7.2.3 解决跨域问题

跨域的允许主要由服务器端控制。服务器端通过在响应的 header 中设置 `Access-Control-Allow-Origin` 及相关一系列参数, 提供跨域访问的允许策略

- 设置响应头中的参数来允许跨域域请求:
 - Access-Control-Allow-Credentials
 - Access-Control-Allow-Origin 标识允许跨域的请求有哪些

1. 在POM文件中引入依赖

```
<!-- 解决跨域问题所需依赖 -->
<dependency>
  <groupId>com.thetransactioncompany</groupId>
  <artifactId>cors-filter</artifactId>
  <version>2.5</version>
</dependency>
```

2. 在web.xml中 配置跨域 filter

```
<!--配置跨域过滤器-->
<filter>
  <filter-name>corsFilter</filter-name>
  <filter-class>com.thetransactioncompany.cors.CORSFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>corsFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

2.7.2.4 再次查询

解决跨域问题之后,页面显示数据

2.8 条件查询

2.8.1 ElementUI输入框组件

- Input 输入框通过鼠标或键盘输入字符

Input 输入框

通过鼠标或键盘输入字符

Input 为受控组件，它总会显示 Vue 绑定值。

通常情况下，应当处理 `input` 事件，并更新组件的绑定值（或使用 `v-model`）。否则，输入框内显示的值将不会改变。

不支持 `v-model` 修饰符。

```
<el-input
  placeholder="请输入内容"
  v-model="input4">
  <i slot="prefix" class="el-input__icon el-icon-search"></i>
</el-input>
```

Course.vue 添加输入框

```

<template>
  <div>
    <!-- 条件查询 -->
    <el-input placeholder="请输入课程名称">
      <i slot="prefix" class="el-input__icon el-icon-search"></i>
    </el-input>

    <!-- 表单显示 ... -->
  </div>
</template>

```

2.8.2 Layout 布局

- 通过基础的 24 分栏，迅速简便地创建布局。
- 通过 row 和 col 组件，并通过 col 组件的 `span` 属性我们就可以自由地组合布局。

分栏间隔

分栏之间存在间隔。



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

```

<el-row :gutter="20">
  <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
  <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
  <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
  <el-col :span="6"><div class="grid-content bg-purple"></div></el-col>
</el-row>

```

2) 使用分隔栏,分隔查询条件

```

<!-- 条件查询 -->
<el-row :gutter="10">
  <el-col :span="5">
    <el-input clearable placeholder="课程名称">
      <i slot="prefix" class="el-input__icon el-icon-search"></i>
    </el-input>
  </el-col>
</el-row>

```

3) 添加一个按钮

```

<el-row :gutter="10">
  <el-col :span="5">
    <el-input clearable placeholder="课程名称">
      <i slot="prefix" class="el-input__icon el-icon-search"></i>
    </el-input>
  </el-col>
  <el-col :span="1">
    <el-button type="primary">查询</el-button>
  </el-col>
</el-row>

```

2.8.3 完成根据课程名查询

1. 双向数据绑定

- Model 模型

```
//数据部分
data() {
  //定义查询条件

  return {
    loading: false, //是否弹出加载提示
    courseList: [], //定义集合,保存从接口获取的参数
    filter: { course_name: "" } //查询条件
  };
},
```

- View 视图

```
<el-input v-model="filter.course_name" clearable placeholder="课程名称">
  <i slot="prefix" class="el-input__icon el-icon-search"></i>
</el-input>
```

2. 设置点击事件

```
<el-button type="primary" @click="search">查询</el-button>
```

3. methods中添加方法

```
search() {
  //开启加载提示
  this.loading = true;

  //发送请求
  return this.$axios
    .get("http://localhost:8080/lagou_edu_home/course", {
      //携带参数
      params: {
        methodName: "findByCourseNameAndStatus",
        course_name: this.filter.course_name
      }
    })
    .then(res => {
      console.log(res);
      this.courseList = res.data;

      //关闭加载
      this.loading = false;
    })
    .catch(error => {
      this.$message.error("获取数据失败!");
    });
}
```