

# 1 Ajax

## 1.1 Ajax介绍

### 1.1.1 Ajax概述

我们前端页面中的数据，如下图所示的表格中的学生信息，应该来自于后台，那么我们的后台和前端是互不影响的2个程序，那么我们前端应该如何从后台获取数据呢？因为是2个程序，所以必须涉及到2个程序的交互，所以这就需要用到我们接下来学习的Ajax技术。

编号	姓名	年龄	性别	成绩	等级
1	Tom	20	男	78	及格
2	Rose	18	女	86	优秀
3	Jerry	26	男	90	优秀
4	Tony	30	男	52	不及格

Ajax： 全称Asynchronous JavaScript And XML，异步的JavaScript和XML。其作用有如下2点：

- 与服务器进行数据交换：通过Ajax可以给服务器发送请求，并获取服务器响应的数据。
- 异步交互：可以在**不重新加载整个页面**的情况下，与服务器交换数据并**更新部分网页**的技术，如：搜索联想、用户名是否可用的校验等等。

### 1.1.2 Ajax作用

我们详细的解释一下Ajax技术的2个作用

- 与服务器进行数据交互

如下图所示前端资源被浏览器解析，但是前端页面上缺少数据，前端可以通过Ajax技术，向后台服务器发起请求，后台服务器接受到前端的请求，从数据库中获取前端需要的资源，然后响应给前端，前端在通过我们学习的vue技术，可以将数据展示到页面上，这样用户就能看到完整的页面了。此处可以对比JavaSE中的网络编程技术来理解。



- 异步交互：可以在**不重新加载整个页面**的情况下，与服务器交换数据并**更新部分网页**的技术。

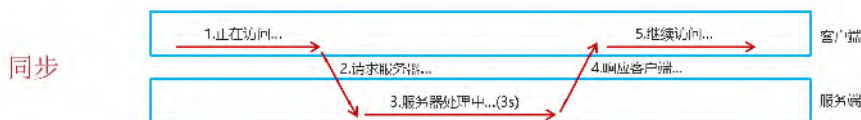
如下图所示，当我们再百度搜索java时，下面的联想数据是通过Ajax请求从后台服务器得到的，在整个过程中，我们的Ajax请求不会导致整个百度页面的重新加载，并且只针对搜索栏这局部模块的数据进行了数据的更新，不会对整个页面的其他地方进行数据的更新，这样就大大提升了页面的加载速度，用户体验高。



### 1.1.1.3 同步异步

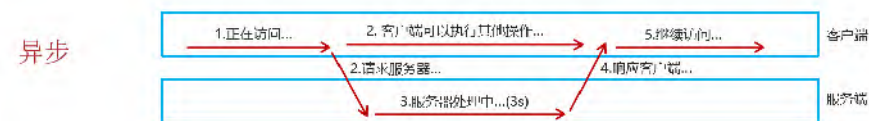
针对于上述Ajax的局部刷新功能是因为Ajax请求是异步的，与之对应的有同步请求。接下来我们介绍一下异步请求和同步请求的区别。

- 同步请求发送过程如下图所示：



浏览器页面在发送请求给服务器，在服务器处理请求的过程中，浏览器页面不能做其他的操作。只能等到服务器响应结束后才能，浏览器页面才能继续做其他的操作。

- 异步请求发送过程如下图所示：



浏览器页面发送请求给服务器，在服务器处理请求的过程中，浏览器页面还可以做其他的操作。

## 1.2 原生Ajax

对于Ajax技术有了充分的认知了，我们接下来通过代码来演示Ajax的效果。此处我们先采用原生的Ajax代码来演示。因为Ajax请求是基于客户端发送请求，服务器响应数据的技术。所以为了完成快速入门案例，我们需要提供服务器端和编写客户端。

- 服务器端

因为我们暂时还没学过服务器端的代码，所以此处已经直接提供好了服务器端的请求地址，我们前端直接通过Ajax请求访问该地址即可。**后台服务器地址：**<http://yapi.smart-xwork.cn/mock/169327/emp/list>

上述地址我们也可以直接通过浏览器来访问，访问结果如图所示：只截取部分数据



- 客户端

客户端的Ajax请求代码如下有如下4步，接下来我们跟着步骤一起操作一下。

第一步：首先我们再VS Code中创建AJAX的文件夹，并且创建名为01. Ajax-原生方式.html的文件，提供如下代码，主要是按钮绑定单击事件，我们希望点击按钮，来发送ajax请求

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
7  scale=1.0">
8      <title>原生Ajax</title>
9  </head>
10 <body>
11     <input type="button" value="获取数据" onclick="getData()">
12
13     <div id="div1"></div>
14
15 </body>
16 <script>
17     function getData() {
18
19     }
20 </script>
21 </html>
```

第二步：创建XMLHttpRequest对象，用于和服务器交换数据，也是原生Ajax请求的核心对象，提供了各种方法。代码如下：

```
1  //1. 创建XMLHttpRequest
2  var xmlHttpRequest = new XMLHttpRequest();
```

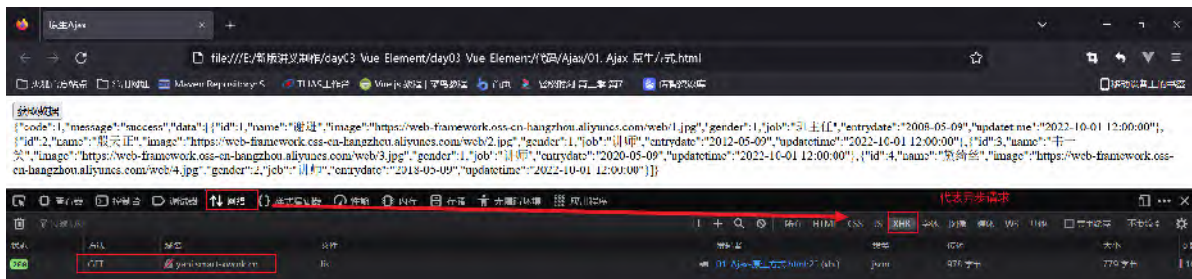
第三步：调用对象的`open()`方法设置请求的参数信息，例如请求地址，请求方式。然后调用`send()`方法向服务器发送请求，代码如下：

```
1 //2. 发送异步请求
2 xmlHttpRequest.open('GET','http://yapi.smart-
  xwork.cn/mock/169327/emp/list');
3 xmlHttpRequest.send();//发送请求
```

第四步：我们通过绑定事件的方式，来获取服务器响应的数据。

```
1 //3. 获取服务响应数据
2 xmlHttpRequest.onreadystatechange = function(){
3     //此处判断 4表示浏览器已经完全接受到Ajax请求得到的响应， 200表示这是一个正确的Http请求，没有错误
4     if(xmlHttpRequest.readyState == 4 && xmlHttpRequest.status == 200){
5         document.getElementById('div1').innerHTML =
          xmlHttpRequest.responseText;
6     }
7 }
```

最后我们通过浏览器打开页面，请求点击按钮，发送Ajax请求，最终显示结果如下图所示：



并且通过浏览器的f12抓包，我们点击网络中的XHR请求，发现可以抓到我们发送的Ajax请求。XHR代表的就是异步请求

## 1.3 Axios

上述原生的Ajax请求的代码编写起来还是比较繁琐的，所以接下来我们学习一门更加简单的发送Ajax请求的技术Axios。Axios是对原生的AJAX进行封装，简化书写。Axios官网是：

<https://www.axios-http.cn>

### 1.3.1 Axios的基本使用

Axios的使用比较简单，主要分为2步：

- 引入Axios文件

```
1 <script src="js/axios-0.18.0.js"></script>
```

- 使用Axios发送请求，并获取响应结果，官方提供的api很多，此处给出2种，如下

- 发送 get 请求

```
1 axios({
2   method:"get",
3   url:"http://localhost:8080/ajax-demo1/aJAXDemo1?
   username=zhangsan"
4 }).then(function (resp) {
5   alert(resp.data);
6 })
```

- 发送 post 请求

```
1 axios({
2   method:"post",
3   url:"http://localhost:8080/ajax-demo1/aJAXDemo1",
4   data:"username=zhangsan"
5 }).then(function (resp) {
6   alert(resp.data);
7 });
```

axios() 是用来发送异步请求的，小括号中使用 js的JSON对象传递请求相关的参数：

- method属性：用来设置请求方式的。取值为 get 或者 post。
- url属性：用来书写请求的资源路径。如果是 get 请求，需要将请求参数拼接到路径的后面，格式为： url?参数名=参数值&参数名2=参数值2。
- data属性：作为请求体被发送的数据。也就是说如果是 post 请求的话，数据需要作为 data 属性的值。

then() 需要传递一个匿名函数。我们将 then() 中传递的匿名函数称为 **回调函数**，意思是该匿名函数在发送请求时不会被调用，而是在成功响应后调用的函数。而该回调函数中的 resp 参数是对响应的数据进行封装的对象，通过 resp.data 可以获取到响应的数据。

### 1.3.2 Axios快速入门

- 后端实现

查询所有员工信息服务器地址: `http://yapi.smart-xwork.cn/mock/169327/emp/list`

根据员工id删除员工信息服务器地址: `http://yapi.smart-xwork.cn/mock/169327/emp/deleteById`

- 前端实现

首先在VS Code中创建js文件夹, 与html同级, 然后将资料/axios-0.18.0.js 文件拷贝到js目录下, 然后创建名为02. Ajax-Axios.html的文件, 工程结果如图所示:



然后在html中引入axios所依赖的js文件, 并且提供2个按钮, 绑定单击事件, 分别用于点击时发送ajax请求, 完整代码如下:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
7      <title>Ajax-Axios</title>
8      <script src="js/axios-0.18.0.js"></script>
9  </head>
10 <body>
11
12     <input type="button" value="获取数据GET" onclick="get()">
13
14     <input type="button" value="删除数据POST" onclick="post()">
15
16 </body>
17 <script>
18     function get() {
19         //通过axios发送异步请求-get
20     }
21
22     function post() {
23         //通过axios发送异步请求-post
24     }
```

```
25 </script>
26 </html>
```

然后分别使用Axios的方法，完整get请求和post请求的发送

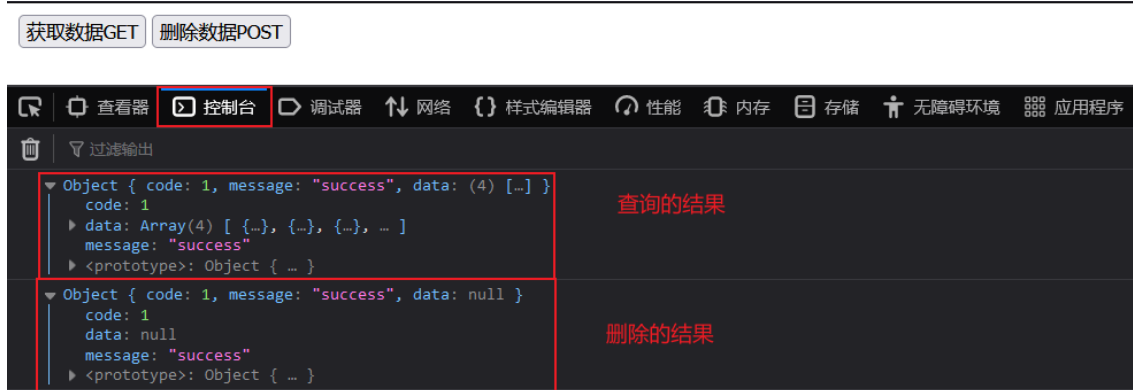
get请求代码如下：

```
1 //通过axios发送异步请求-get
2 axios({
3   method: "get",
4   url: "http://yapi.smart-xwork.cn/mock/169327/emp/list"
5 }).then(result => {
6   console.log(result.data);
7 })
```

post请求代码如下：

```
1 //通过axios发送异步请求-post
2 axios({
3   method: "post",
4   url: "http://yapi.smart-xwork.cn/mock/169327/emp/deleteById",
5   data: "id=1"
6 }).then(result => {
7   console.log(result.data);
8 })
```

浏览器打开，f12抓包，然后分别点击2个按钮，查看控制台效果如下：



完整代码如下：

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-
  scale=1.0">
```

```
7     <title>Ajax-Axios</title>
8     <script src="js/axios-0.18.0.js"></script>
9 </head>
10 <body>
11
12     <input type="button" value="获取数据GET" onclick="get()">
13
14     <input type="button" value="删除数据POST" onclick="post()">
15
16 </body>
17 <script>
18     function get() {
19         //通过axios发送异步请求-get
20         axios({
21             method: "get",
22             url: "http://yapi.smart-
xwork.cn/mock/169327/emp/list"
23         }).then(result => {
24             console.log(result.data);
25         })
26
27
28     }
29
30     function post() {
31         // 通过axios发送异步请求-post
32         axios({
33             method: "post",
34             url: "http://yapi.smart-
xwork.cn/mock/169327/emp/deleteById",
35             data: "id=1"
36         }).then(result => {
37             console.log(result.data);
38         })
39
40     }
41 </script>
42 </html>
```



### 1.3.3 请求方法的别名

Axios还针对不同的请求，提供了别名方式的api，具体如下：

方法	描述
<code>axios.get(url [, config])</code>	发送get请求
<code>axios.delete(url [, config])</code>	发送delete请求
<code>axios.post(url [, data[, config]])</code>	发送post请求
<code>axios.put(url [, data[, config]])</code>	发送put请求

我们目前只关注get和post请求，所以在上述的入门案例中，我们可以将get请求代码改写成如下：

```
1 axios.get("http://yapi.smart-xwork.cn/mock/169327/emp/list").then(result => {  
2     console.log(result.data);  
3 })
```

post请求改写成如下：

```
1 axios.post("http://yapi.smart-xwork.cn/mock/169327/emp/deleteById","id=1").then(result => {  
2     console.log(result.data);  
3 })
```

### 1.3.4 案例

- 需求：基于Vue及Axios完成数据的动态加载展示，如下图所示

编号	姓名	图像	性别	职位	入职日期	最后操作时间
1	谢逊		男	班主任	2008-05-09	2022-10-01 12:00:00
2	殷天正		男	讲师	2012-05-09	2022-10-01 12:00:00
3	韦一笑		男	讲师	2020-05-09	2022-10-01 12:00:00
4	黛绮丝		女	讲师	2018-05-09	2022-10-01 12:00:00

其中数据是来自于后台程序的，地址是：<http://yapi.smart-xwork.cn/mock/169327/emp/list>

- 分析：

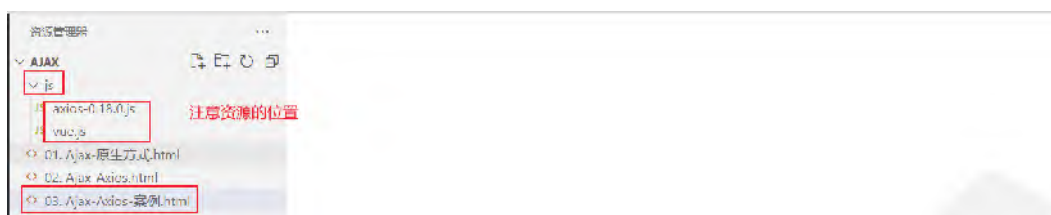
前端首先是一张表格，我们缺少数据，而提供数据的地址已经有了，所以意味这我们需要使用Ajax请求获取后台的数据。但是Ajax请求什么时候发送呢？页面的数据应该是页面加载完成，自动发送请求，展示数据，所以我们需要借助vue的mounted钩子函数。那么拿到数据了，我们该怎么将数据显示表格中呢？这里就得借助v-for指令来遍历数据，展示数据。

- 步骤：

1. 首先创建文件，提前准备基础代码，包括表格以及vue.js和axios.js文件的引入
2. 我们需要在vue的mounted钩子函数中发送ajax请求，获取数据
3. 拿到数据，数据需要绑定给vue的data属性
4. 在<tr>标签上通过v-for指令遍历数据，展示数据

- 代码实现：

1. 首先创建文件，提前准备基础代码，包括表格以及vue.js和axios.js文件的引入



提供初始代码如下：

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width,
7          initial-scale=1.0">
8      <title>Ajax-Axios-案例</title>
9      <script src="js/axios-0.18.0.js"></script>
10     <script src="js/vue.js"></script>
11 </head>
12 <body>
13     <div id="app">
14         <table border="1" cellspacing="0" width="60%">
15             <tr>
16                 <th>编号</th>
17                 <th>姓名</th>
18                 <th>图像</th>
19                 <th>性别</th>
20                 <th>职位</th>
21                 <th>入职日期</th>
22                 <th>最后操作时间</th>
23             </tr>
```

```

24         <tr align="center" >
25             <td>1</td>
26             <td>Tom</td>
27             <td>
28                 <img src="" width="70px" height="50px">
29             </td>
30             <td>
31                 <span>男</span>
32                 <!-- <span>女</span>-->
33             </td>
34             <td>班主任</td>
35             <td>2009-08-09</td>
36             <td>2009-08-09 12:00:00</td>
37         </tr>
38     </table>
39 </div>
40 </body>
41 <script>
42     new Vue({
43         el: "#app",
44         data: {
45
46         }
47     });
48 </script>
49 </html>

```

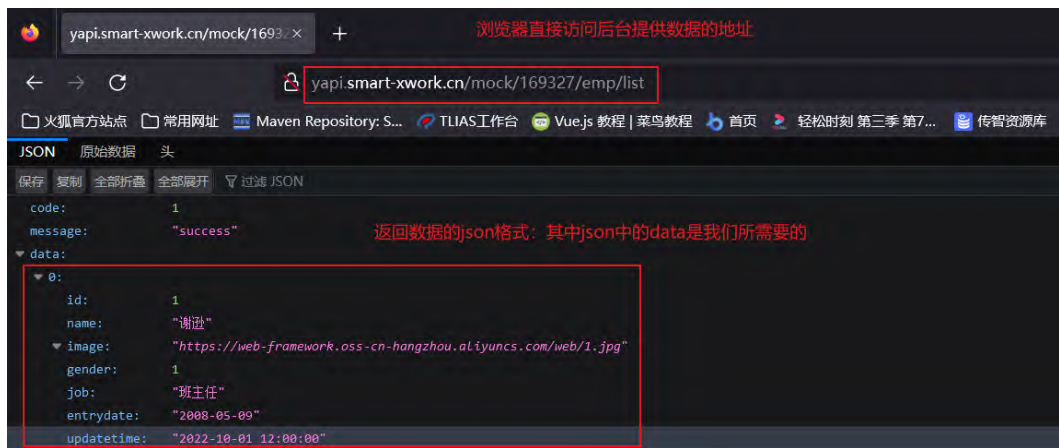
2. 在vue的mounted钩子函数，编写Ajax请求，请求数据，代码如下：

```

1  mounted () {
2      //发送异步请求,加载数据
3      axios.get("http://yapi.smart-
4      xwork.cn/mock/169327/emp/list").then(result => {
5
6      })

```

3. ajax请求的数据我们应该绑定给vue的data属性，之后才能进行数据绑定到视图；并且浏览器打开后台地址，数据返回格式如下图所示：



因为服务器响应的json中的data属性才是我们需要展示的信息，所以我们应该将员工列表信息赋值给vue的data属性，代码如下：

```
1 //发送异步请求,加载数据
2 axios.get("http://yapi.smart-
  xwork.cn/mock/169327/emp/list").then(result => {
3   this.emps = result.data.data;
4 })
```

其中，data中生命emps变量，代码如下：

```
1 data: {
2   emps: []
3 },
```

4. 在<tr>标签上通过v-for指令遍历数据，展示数据，其中需要注意的是图片的值，需要使用vue的属性绑定，男女的展示需要使用条件判断，其代码如下：

```
1 <tr align="center" v-for="(emp,index) in emps">
2   <td>{{index + 1}}</td>
3   <td>{{emp.name}}</td>
4   <td>
5     
6   </td>
7   <td>
8     <span v-if="emp.gender == 1">男</span>
9     <span v-if="emp.gender == 2">女</span>
10  </td>
11  <td>{{emp.job}}</td>
12  <td>{{emp.entrydate}}</td>
13  <td>{{emp.updatetime}}</td>
14 </tr>
```

完整代码如下：

```
1 <!DOCTYPE html>
```

```
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-
  scale=1.0">
7   <title>Ajax-Axios-案例</title>
8   <script src="js/axios-0.18.0.js"></script>
9   <script src="js/vue.js"></script>
10 </head>
11 <body>
12   <div id="app">
13     <table border="1" cellspacing="0" width="60%">
14       <tr>
15         <th>编号</th>
16         <th>姓名</th>
17         <th>图像</th>
18         <th>性别</th>
19         <th>职位</th>
20         <th>入职日期</th>
21         <th>最后操作时间</th>
22       </tr>
23
24       <tr align="center" v-for="(emp,index) in emps">
25         <td>{{index + 1}}</td>
26         <td>{{emp.name}}</td>
27         <td>
28           
29         </td>
30         <td>
31           <span v-if="emp.gender == 1">男</span>
32           <span v-if="emp.gender == 2">女</span>
33         </td>
34         <td>{{emp.job}}</td>
35         <td>{{emp.entrydate}}</td>
36         <td>{{emp.updatetime}}</td>
37       </tr>
38     </table>
39   </div>
40 </body>
41 <script>
42   new Vue({
43     el: "#app",
```

```

44     data: {
45         emps: []
46     },
47     mounted () {
48         //发送异步请求,加载数据
49         axios.get("http://yapi.smart-
xwork.cn/mock/169327/emp/list").then(result => {
50             console.log(result.data);
51             this.emps = result.data.data;
52         })
53     }
54 });
55 </script>
56 </html>

```

## 2 前后台分离开发

### 2.1 前后台分离开发介绍

在之前的课程中，我们介绍过，前端开发有2种方式：**前后台混合开发**和**前后台分离开发**。

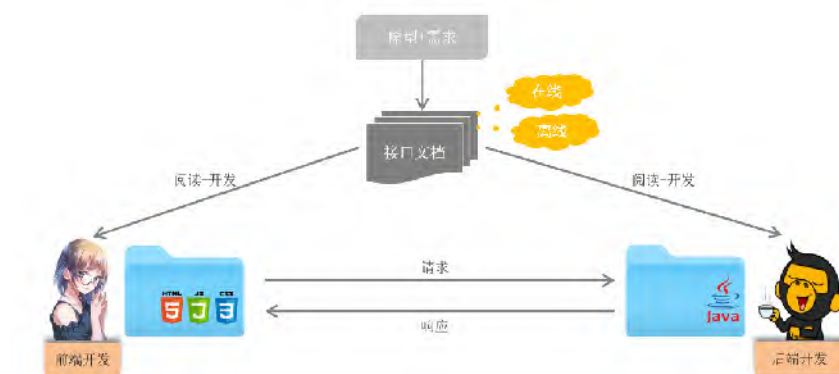
前后台混合开发，顾名思义就是前台后台代码混在一起开发，如下图所示：



这种开发模式有如下缺点：

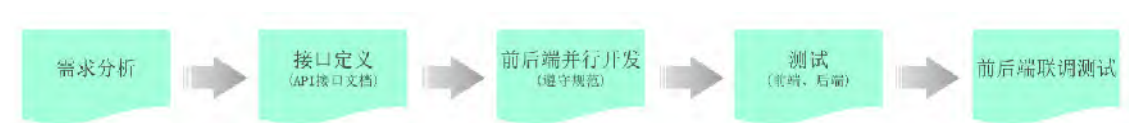
- 沟通成本高：后台人员发现前端有问题，需要找前端人员修改，前端修改成功，再交给后台人员使用
- 分工不明确：后台开发人员需要开发后台代码，也需要开发部分前端代码。很难培养专业人才
- 不便管理：所有的代码都在一个工程中
- 不便维护和扩展：前端代码更新，和后台无关，但是需要整个工程包括后台一起重新打包部署。

所以我们目前基本都是采用的前后台分离开发方式，如下图所示：



我们将原先的工程分为前端工程和后端工程这2个工程，然后前端工程交给专业的前端人员开发，后端工程交给专业的后端人员开发。前端页面需要数据，可以通过发送异步请求，从后台工程获取。但是，我们前后台是分开来开发的，那么前端人员怎么知道后台返回数据的格式呢？后端人员开发，怎么知道前端人员需要的数据格式呢？所以针对这个问题，我们前后台统一指定一套规范！我们前后台开发人员都需要遵循这套规范开发，这就是我们的**接口文档**。接口文档有离线版和在线版本，接口文档可以查询今天提供**资料/接口文档示例**里面的资料。那么接口文档的内容怎么来的呢？是我们后台开发者根据产品经理提供的产品原型和需求文档所撰写出来的，产品原型示例可以参考今天提供**资料/页面原型**里面的资料。

那么基于前后台分离开发的模式下，我们后台开发者开发一个功能的具体流程如何呢？如下图所示：



1. 需求分析：首先我们需要阅读需求文档，分析需求，理解需求。
2. 接口定义：查询接口文档中关于需求的接口的定义，包括地址，参数，响应数据类型等等
3. 前后台并行开发：各自按照接口文档进行开发，实现需求
4. 测试：前后台开发完了，各自按照接口文档进行测试
5. 前后段联调测试：前段工程请求后端工程，测试功能

## 2.2 YAPI

### 2.2.1 YAPI介绍

前后台分离开发中，我们前后台开发人员都需要遵循接口文档，所以接下来我们介绍一款撰写接口文档的平台。

YApi 是高效、易用、功能强大的 api 管理平台，旨在为开发、产品、测试人员提供更优雅的管理服务。

其官网地址：<http://yapi.smart-xwork.cn/>



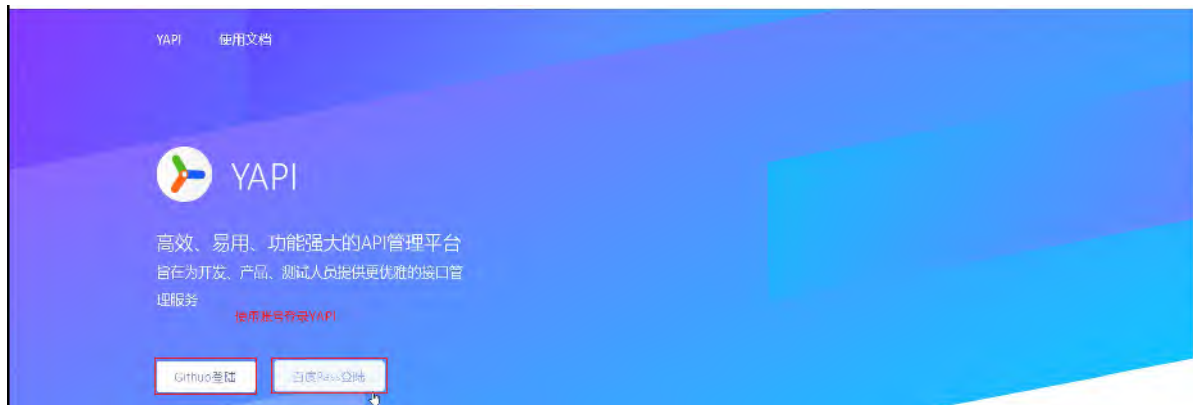
YApi主要提供了2个功能：

- API接口管理：根据需求撰写接口，包括接口的地址，参数，响应等信息。
- Mock服务：模拟真实接口，生成接口的模拟测试数据，用于前端的测试。

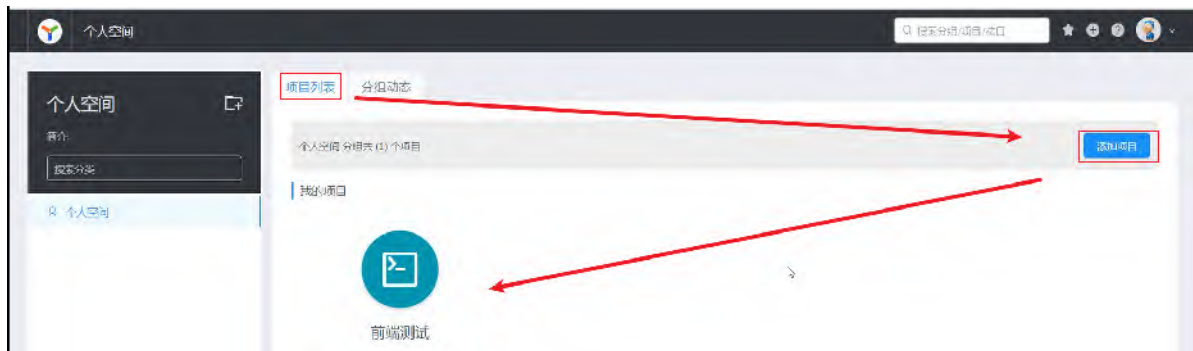
## 2.2.2 接口文档管理

接下来我们演示一下YApi是如何管理接口文档的。

首先我们登录YAPI的官网，然后使用github或者百度账号登录，没有的话去注册一个，如下图所示：



登录进去后，在个人空间中，选择项目列表->添加测试项目，效果如图所示：



然后点击创建的项目，进入到项目中，紧接着先添加接口的分类，如下图所示



然后我们选择当前创建的分类，创建接口信息，如下图所示：

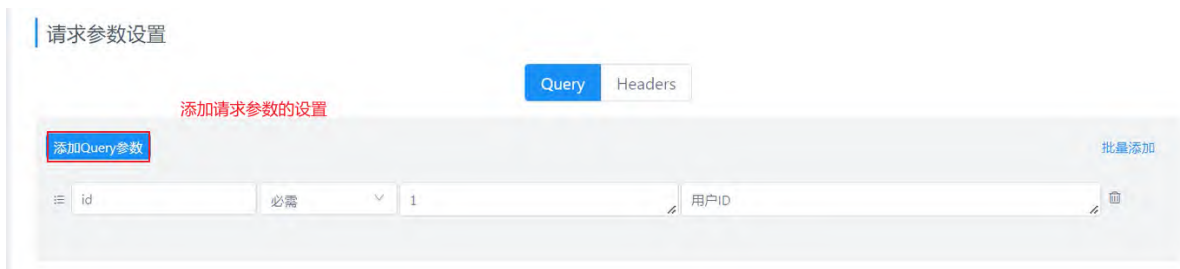




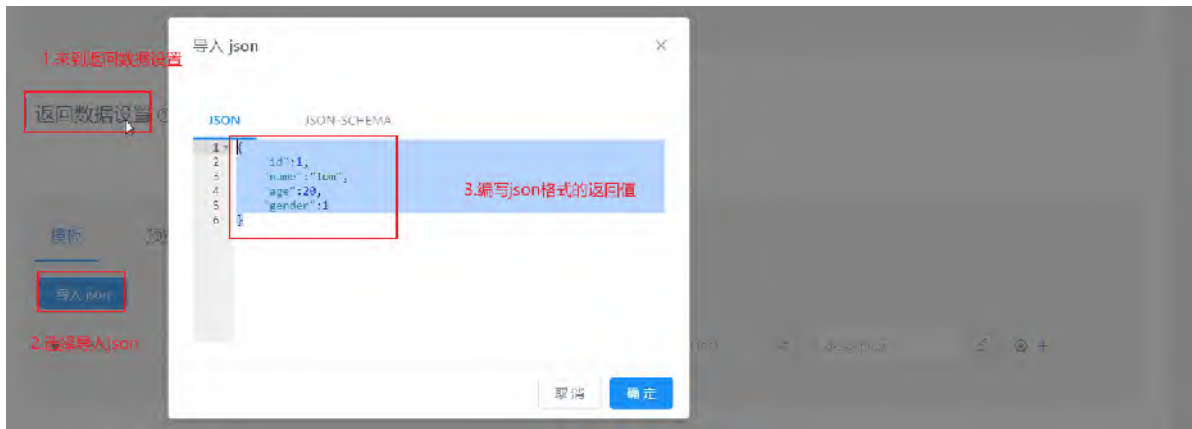
紧接着，我们来到接口的编辑界面，对接口做生层次的定制，例如：接口的参数，接口的返回值等等，效果图下图所示：



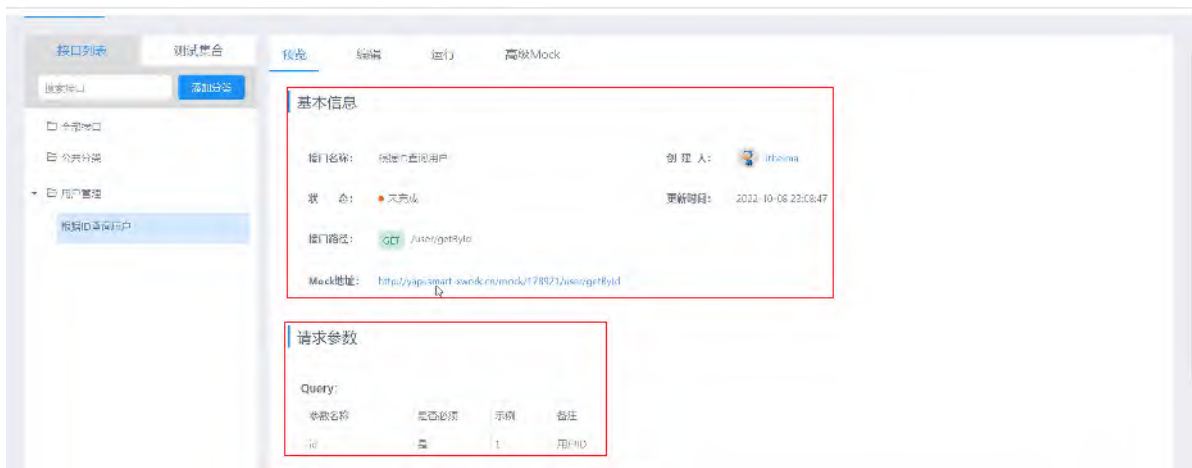
添加接口的请求参数，如下图所示：



添加接口的返回值，如下图所示：



然后保存上述设置，紧接着我们可以来到接口的预览界面，查询接口的信息，其效果如下图所示：篇幅有限，只截取部分



最后，我们还可以设置接口的mock信息，



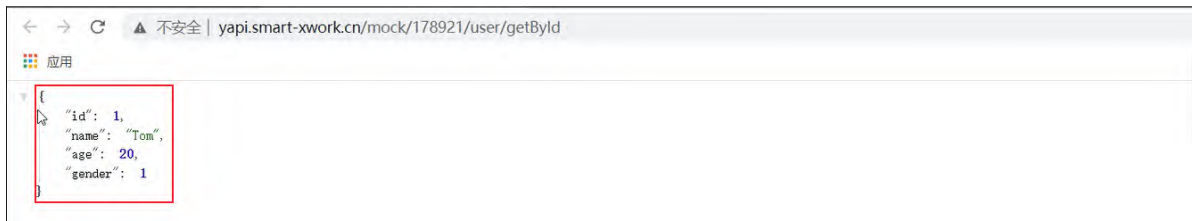
来到接口的Mock设置窗口，如下图所示：



紧接着我们来到接口的预览界面，直接点击Mock地址，如下图所示：



我们发现浏览器直接打开，并返回如下数据：



如上步骤就是YAPI接口平台中对于接口的配置步骤。

## 3 前端工程化

### 3.1 前端工程化介绍

我们目前的前端开发中，当我们需要使用一些资源时，例如：vue.js，和axios.js文件，都是直接再工程中导入的，如下图所示：



但是上述开发模式存在如下问题：

- 每次开发都是从零开始，比较麻烦
- 多个页面中的组件共用性不好
- js、图片等资源没有规范化的存储目录，没有统一的标准，不方便维护

所以现在企业开发中更加讲究前端工程化方式的开发，主要包括如下4个特点

- 模块化：将js和css等，做成一个个可复用模块
- 组件化：我们将UI组件，css样式，js行为封装成一个个的组件，便于管理
- 规范化：我们提供一套标准的规范的目录接口和编码规范，所有开发人员遵循这套规范
- 自动化：项目的构建，测试，部署全部都是自动完成

所以对于前端工程化，说白了，就是在企业级的前端项目开发中，把前端开发所需要的工具、技术、流程、经验进行规范化和标准化。从而提升开发效率，降低开发难度等等。接下来我们就需要学习vue的官方提供的脚手架帮我们完成前端的工程化。

## 3.2 前端工程化入门

### 3.2.1 环境准备

我们的前端工程化是通过vue官方提供的脚手架Vue-cli来完成的，用于快速的生成一个Vue的项目模板。Vue-cli主要提供了如下功能：

- 统一的目录结构
- 本地调试
- 热部署
- 单元测试
- 集成打包上线

我们需要运行Vue-cli，需要依赖NodeJS，NodeJS是前端工程化依赖的环境。所以我们需要先安装NodeJS，然后才能安装Vue-cli

- NodeJS安装和Vue-cli安装

详细安装步骤，请参考[资料/NodeJS安装文档/NodeJS安装文档.md](#)文件

名称	修改日期	类型	大小
assets	2022/11/22 15:35	文件夹	
NodeJS安装文档.md	2022/10/08 21:33	Markdown File	2 KB
node-v16.12.1-x64.msi	2022/10/08 21:30	Windows Install...	20,652 KB

### 3.2.2 Vue项目简介

环境准备好了，接下来我们需要通过Vue-cli创建一个vue项目，然后再学习一下vue项目的目录结构。Vue-cli提供了如下2种方式创建vue项目：

- 命令行：直接通过命令行方式创建vue项目

```
1 vue create vue-project01
```

- 图形化界面：通过命令先进入到图形化界面，然后再进行vue工程的创建

```
1 vue ui
```

图形化界面如下：



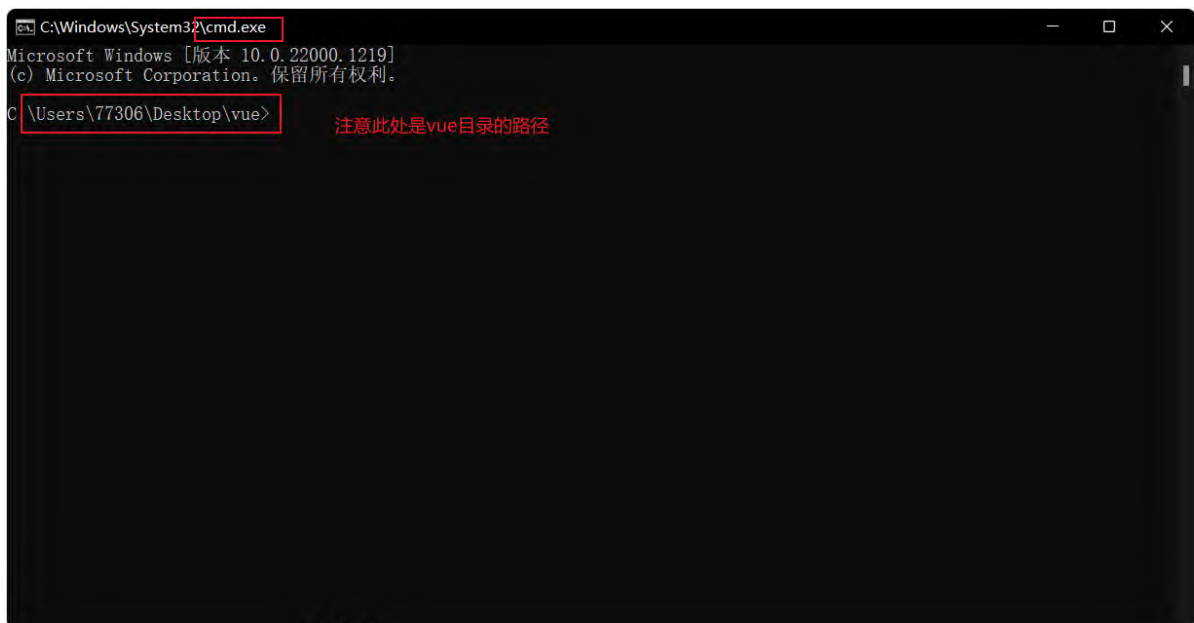
### 3.2.2.1 创建vue项目

此处我们通过第二种图形化界面方式给大家演示。

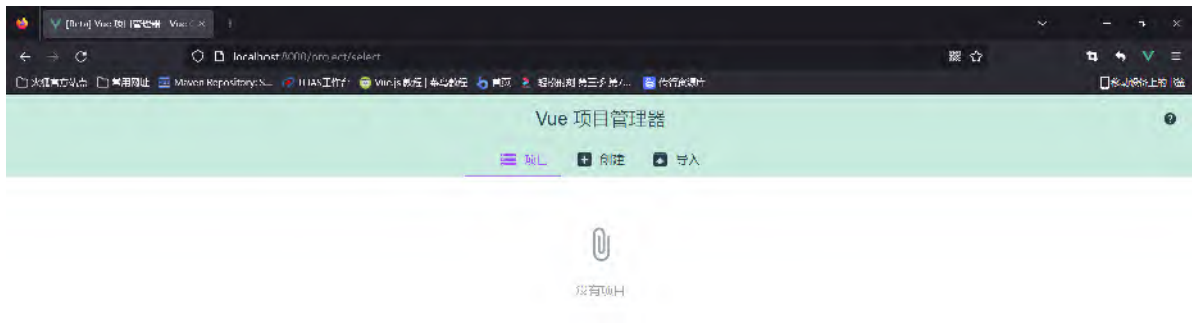
首先，我们再桌面创建vue文件夹，然后双击进入文件夹，来到地址目录，输入cmd，然后进入到vue文件夹的cmd窗口界面，如下图所示：



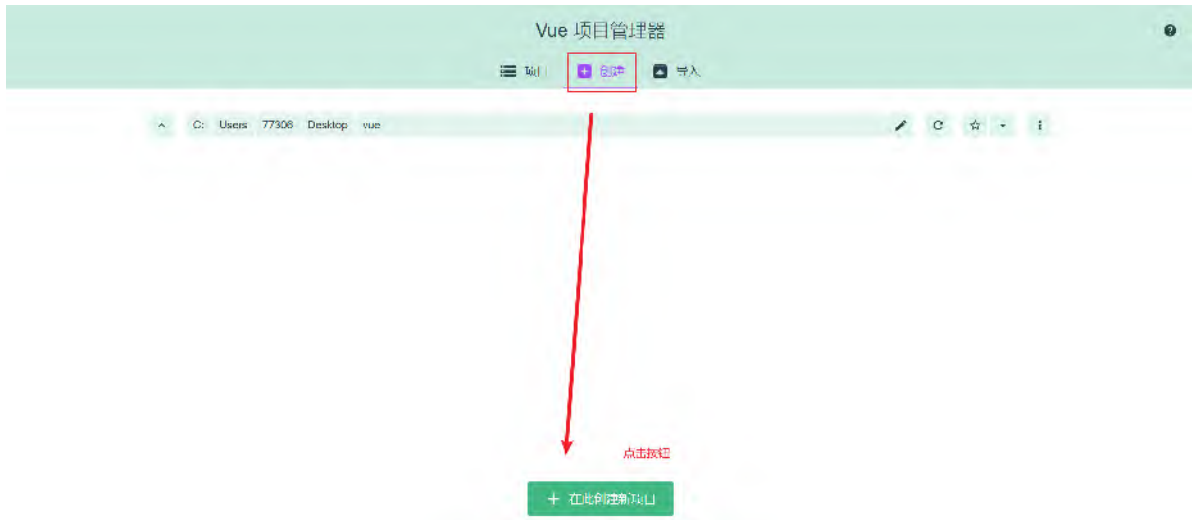
然后进入如下界面：



然后再当前目录下，直接输入命令 `vue ui` 进入到vue的图形化界面，如下图所示：



然后我们选择创建按钮，在vue文件夹下创建项目，如下图所示：



然后来到如下界面，进行vue项目的创建



然后预设模板选择手动，如下图所示：

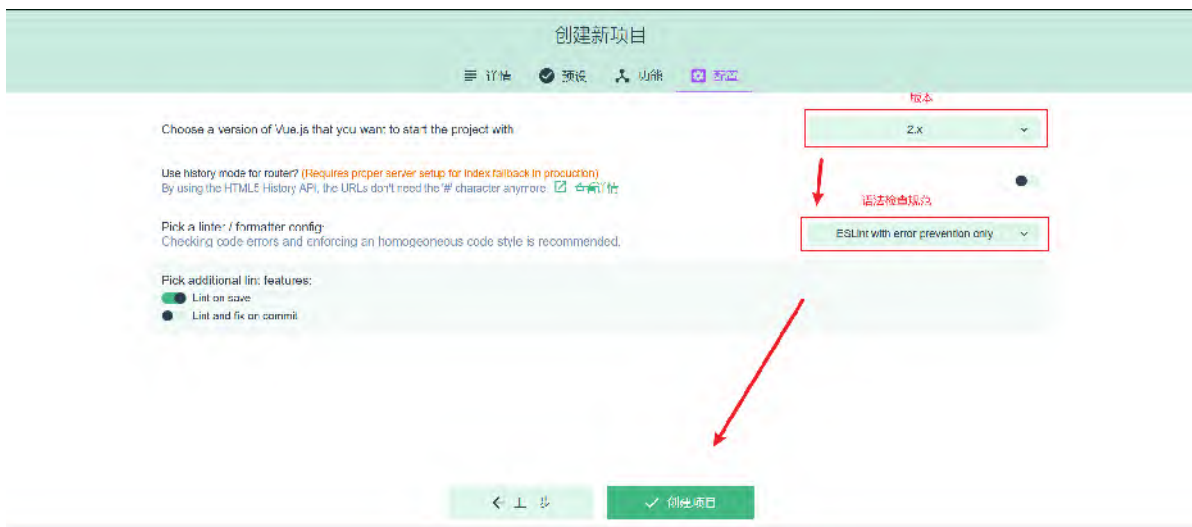




然后再功能页面开启路由功能，如下图所示：



然后再配置页面选择语言版本和语法检查规范，如下图所示：



然后创建项目，进入如下界面：



最后我们只需要等待片刻，即可进入到创建创建成功的界面，如下图所示：



到此，vue项目创建结束

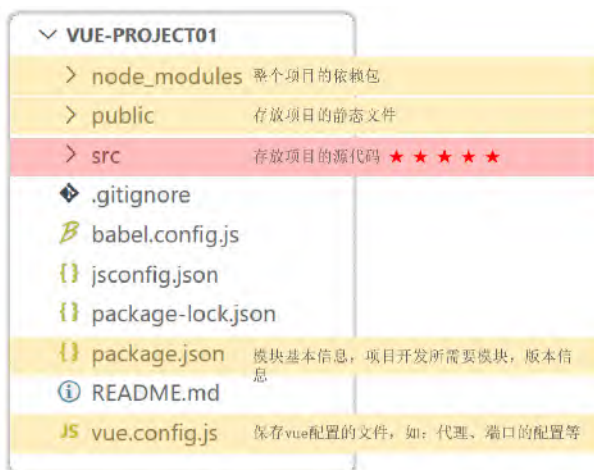
### 3.2.2.2 vue项目目录结构介绍

我们通过VS Code打开之前创建的vue文件夹，打开之后，呈现如下图所示页面：



vue项目的标准目录结构以及目录对应的解释如下图所示：



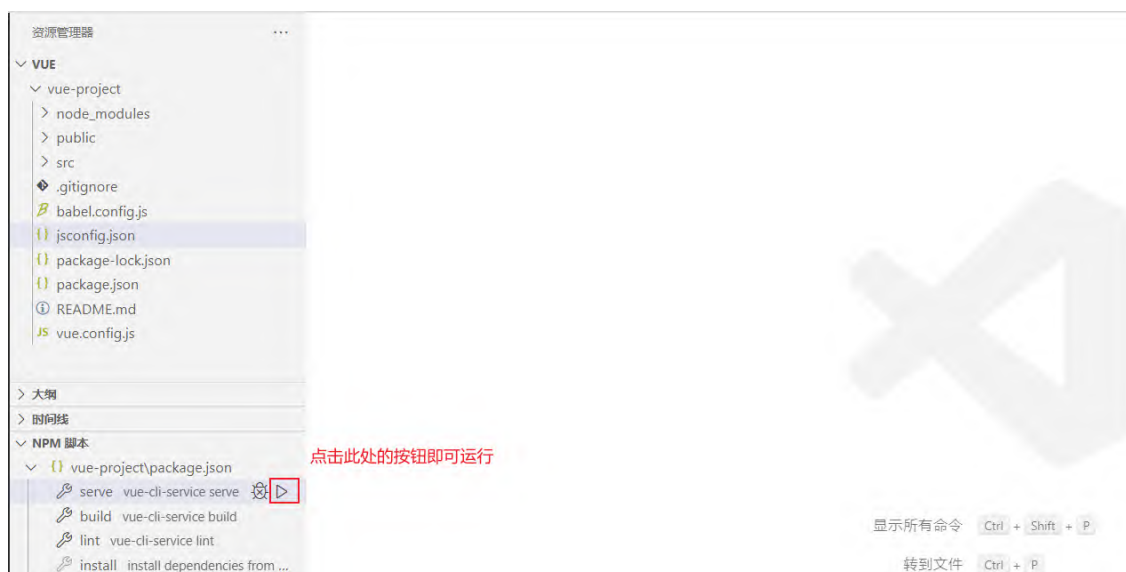


其中我们平时开发代码就是在src目录下

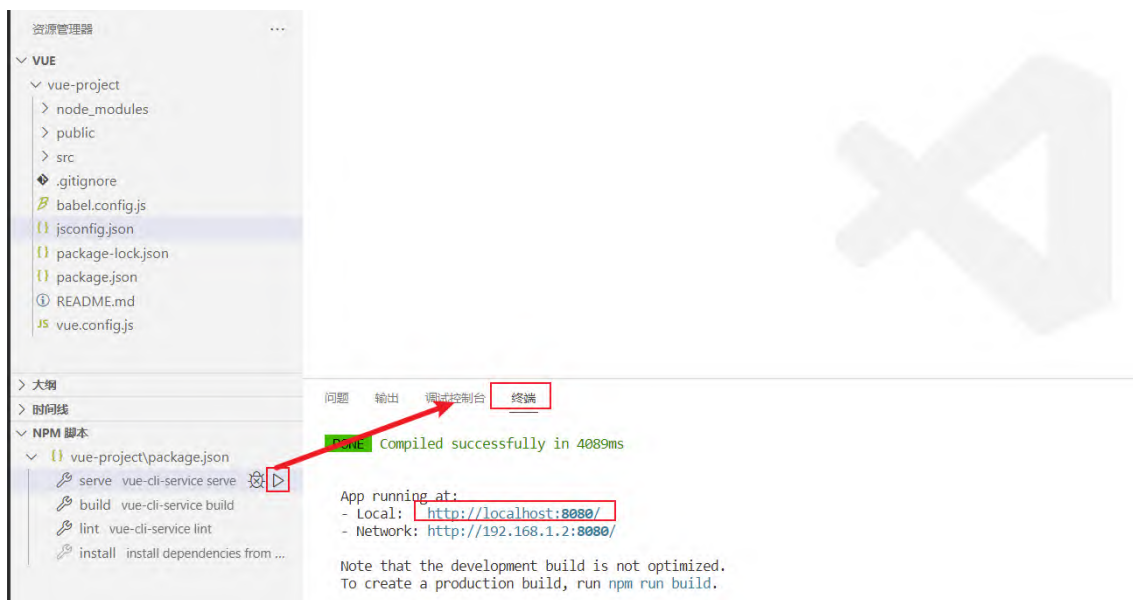
### 3.2.2.3 运行vue项目

那么vue项目开发好了, 我们应该怎么运行vue项目呢? 主要提供了2种方式

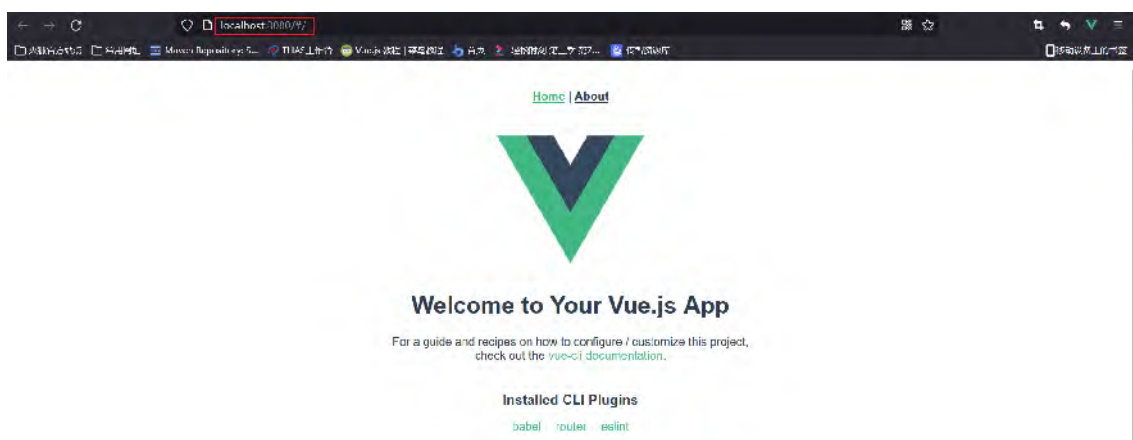
- 第一种方式: 通过VS Code提供的图形化界面, 如下图所示: (注意: NPM脚本窗口默认不显示, 可以参考本节的最后调试出来)



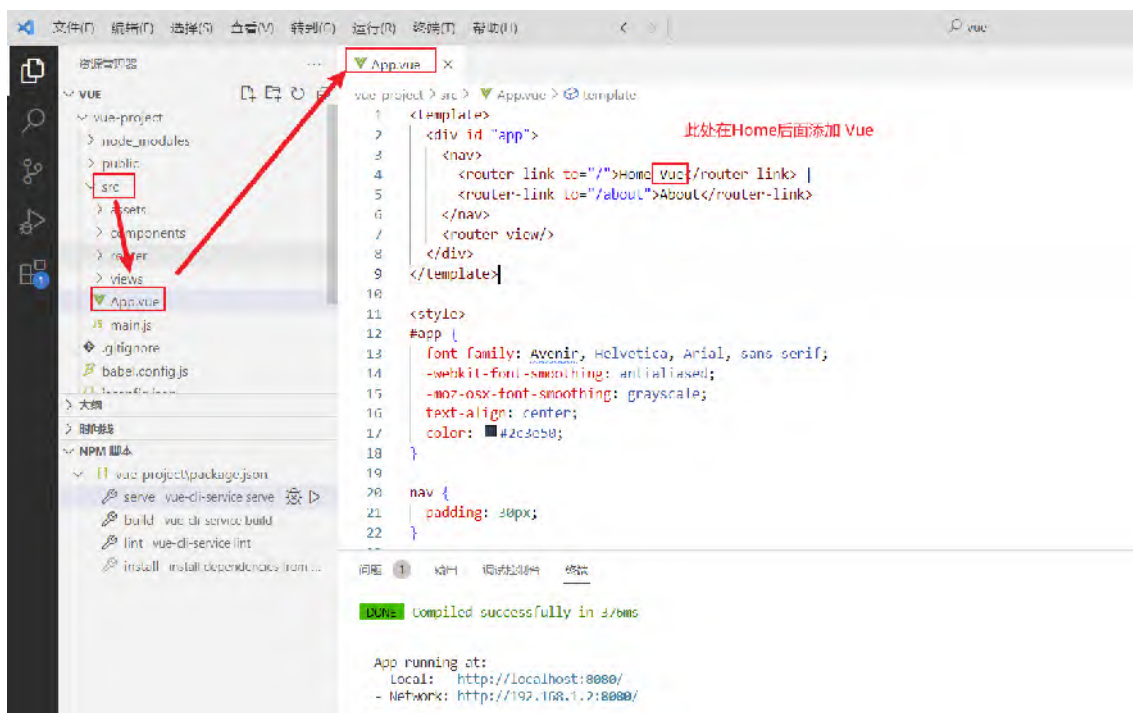
点击之后, 我们等待片刻, 即可运行, 在终端界面中, 我们发现项目是运行在本地服务的8080端口, 我们直接通过浏览器打开地址



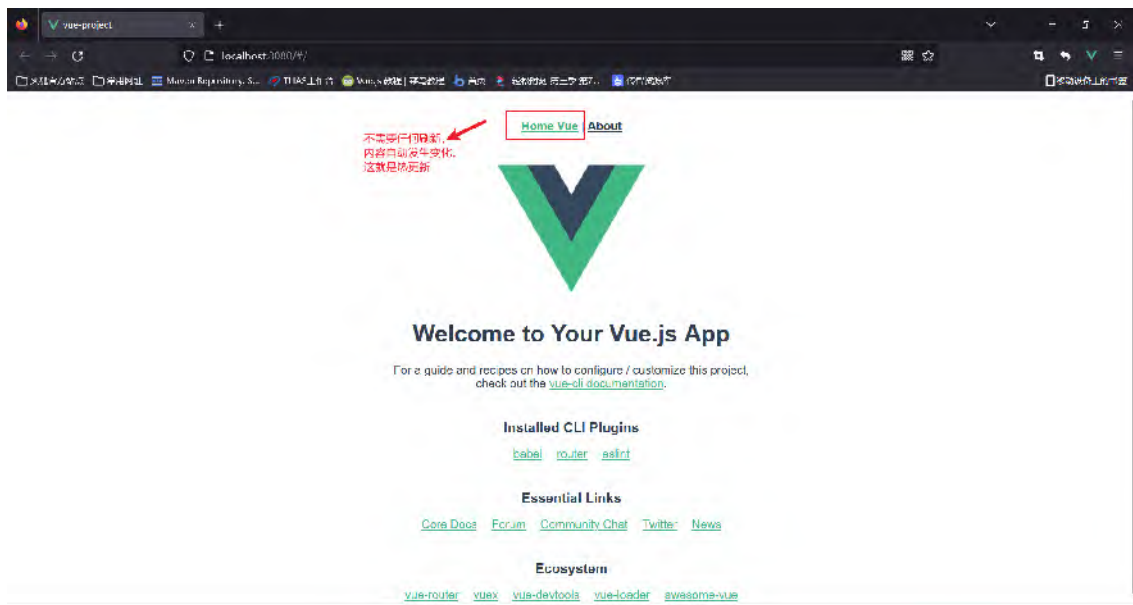
最终浏览器打开后，呈现如下界面，表示项目运行成功



其实此时访问的是 `src/App.vue` 这个根组件，此时我们打开这个组件，修改代码：添加内容Vue



只要我们保存更新的代码，我们直接打开浏览器，不需要做任何刷新，发现页面呈现内容发生了变化，如下图所示：

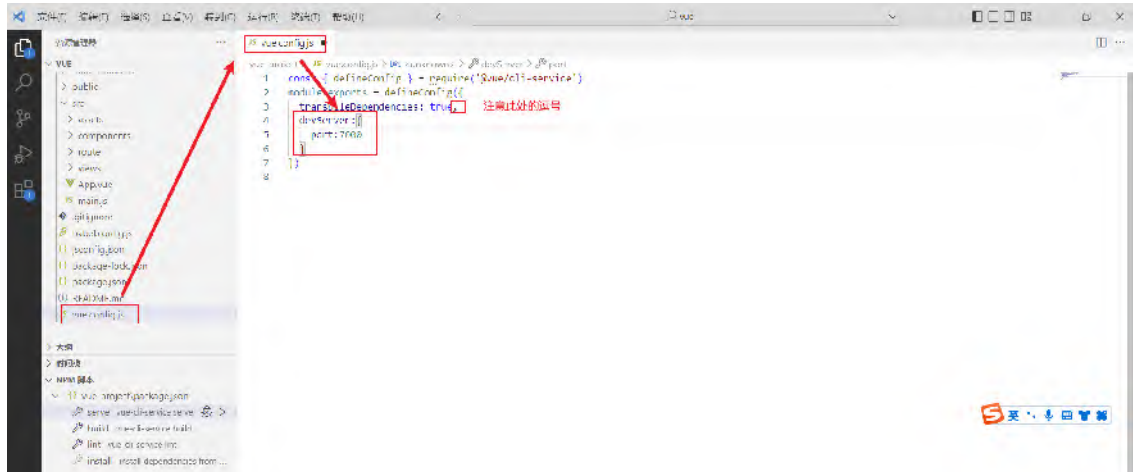


这就是我们vue项目的热更新功能

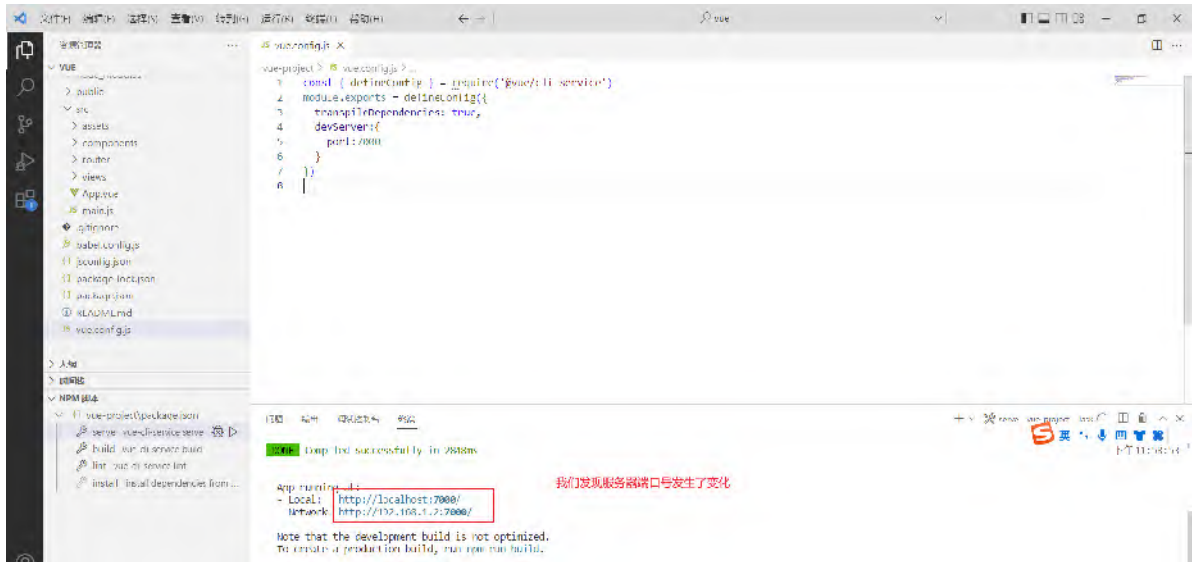
对于8080端口, 经常被占用, 所以我们可以去修改默认的8080端口。我们修改vue.config.js文件的内容, 添加如下代码:

```
1 devServer:{
2     port:7000
3 }
```

如下图所示, 然后我们关闭服务器, 并且重新启动,



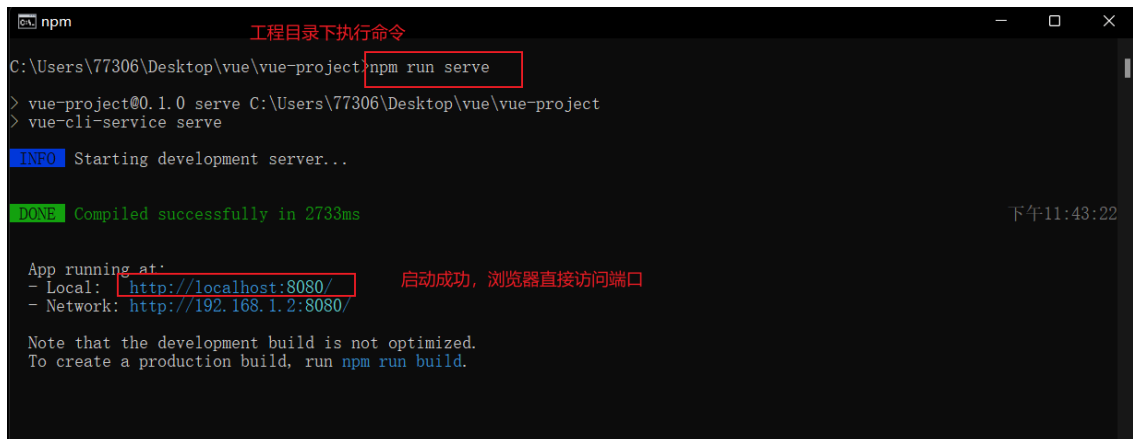
重新启动如下图所示:



端口更改成功，可以通过浏览器访问7000端口来访问我们之前的项目

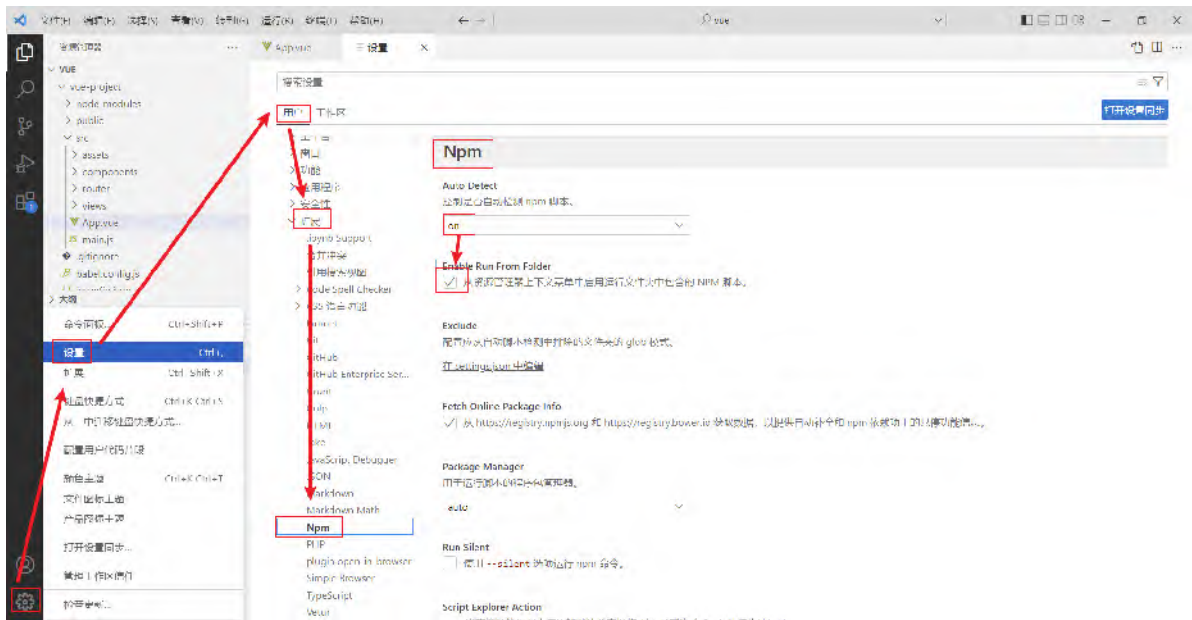
- 第二种方式：命令行方式

直接基于cmd命令窗口，在vue目录下，执行输入命令 `npm run serve` 即可，如下图所示：

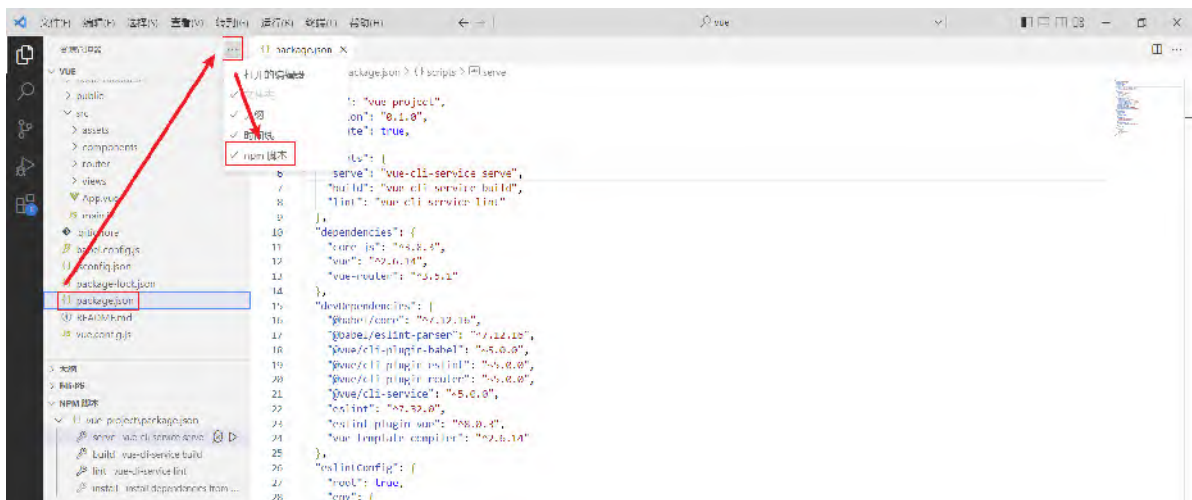


补充：NPM脚本窗口调试出来

第一步：通过设置/用户设置/扩展/MPM更改NPM默认配置，如下图所示



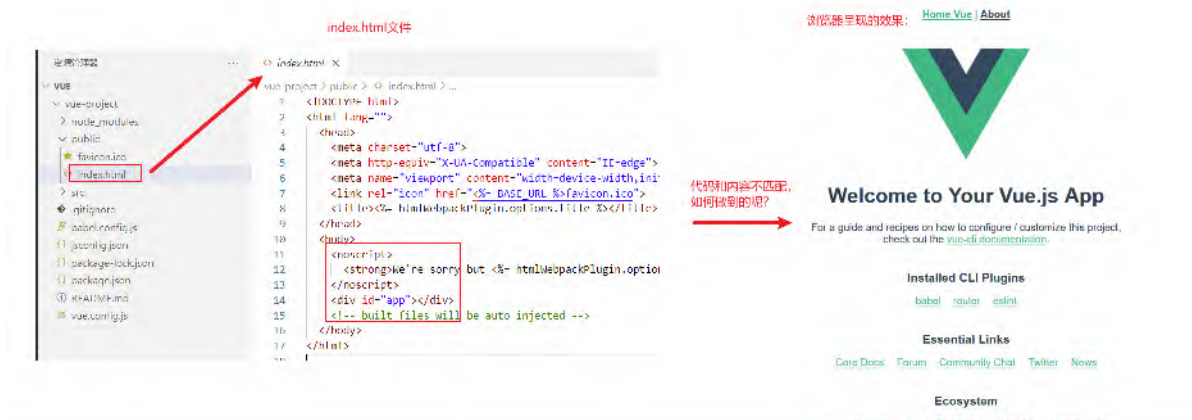
然后重启 VS Code，并且双击打开 `package.json` 文件，然后点击资源管理器处的三个小点，勾选 `npm` 脚本选项，如图所示



然后就能都显示 NPM 脚本小窗口了。

### 3.2.3 Vue项目开发流程

那么我们访问的首页是 `index.html`，但是我们找到 `public/index.html` 文件，打开之后发现，里面没有什么代码，但是能够呈现内容丰富的首页：如下图所示：





我们自习观察发现，index.html的代码很简洁，但是浏览器所呈现的index.html内容却很丰富，代码和内容不匹配，所以vue是如何做到的呢？接下来我们学习一下vue项目的开发流程。

对于vue项目，index.html文件默认是引入了入口函数main.js文件，我们找到src/main.js文件，其代码如下：

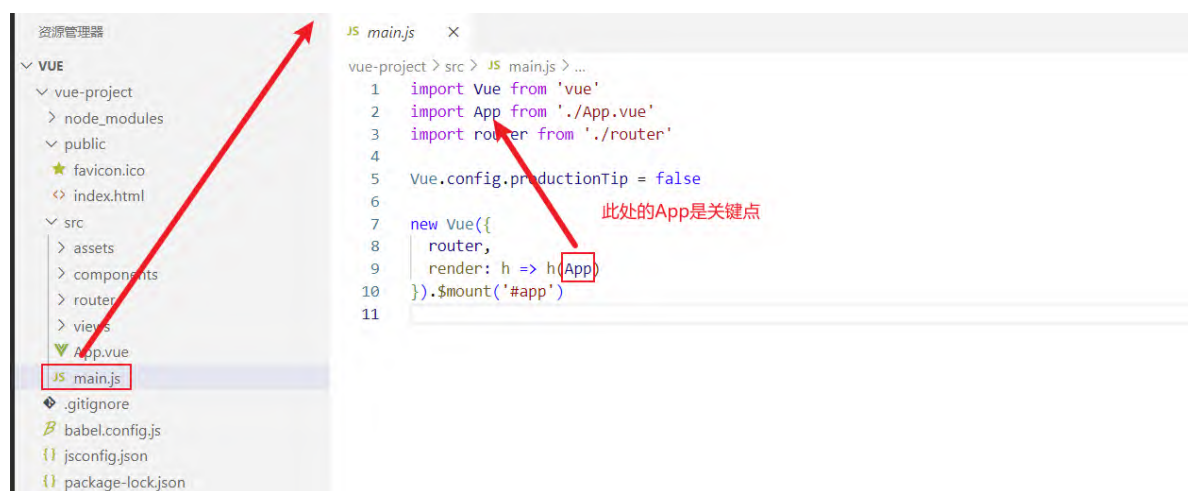
```
1  import Vue from 'vue'
2  import App from './App.vue'
3  import router from './router'
4
5  Vue.config.productionTip = false
6
7  new Vue({
8    router,
9    render: h => h(App)
10 }).$mount('#app')
```

上述代码中，包括如下几个关键点：

- import：导入指定文件，并且重新起名。例如上述代码 `import App from './App.vue'` 导入当前目录下得App.vue并且起名为App
- new Vue()：创建vue对象
- \$mount('#app')；将vue对象创建的dom对象挂在到id=app的这个标签区域中，作用和之前学习的vue对象的le属性一致。
- router： 路由，详细在后面的小节讲解
- render： 主要使用视图的渲染的。

来到public/index.html中，我们删除div的id=app属性，打开浏览器，发现之前访问的首页一片空白，如下图所示，这样就证明了，我们main.js中通过代码挂在到index.html的id=app的标签区域的。

此时我们知道了vue创建的dom对象挂在到id=app的标签区域，但是我们还是没有解决最开始的问题：首页内容如何呈现的？这就涉及到render中的App了，如下图所示：



那么这个App对象怎么回事呢，我们打开App.vue，注意的是.vue结尾的都是vue组件。而vue的组件文件包含3个部分：

- template：模板部分，主要是HTML代码，用来展示页面主体结构的
- script：js代码区域，主要是通过js代码来控制模板的数据来源和行为的
- style：css样式部分，主要通过css样式控制模板的页面效果得

如下图所示就是一个vue组件的小案例：



此时我们可以打开App.vue，观察App.vue的代码，其中可以发现，App.vue组件的template部分内容，和我们浏览器访问的首页内容是一致的，如下图所示：



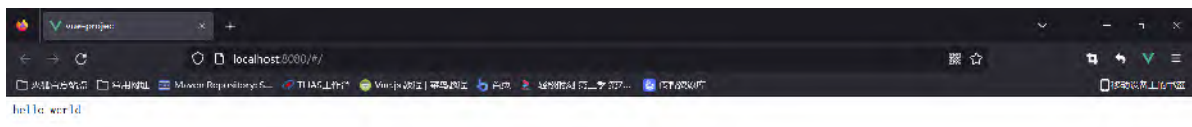
接下来我们可以简化模板部分内容，添加script部分的数据模型，删除css样式，完整代码如下：

```

1  <template>
2    <div id="app">
3      {{message}}
4    </div>
5  </template>
6
7  <script>
8    export default {
9      data() {
10        return {
11          "message": "hello world"
12        }
13      }
14    }
15  </script>
16  <style>
17
18  </style>

```

保存直接，回到浏览器，我们发现首页展示效果发生了变化，如下图所示：



## 4 Vue组件库Element

### 4.1 Element介绍



不知道同学们还否记得我们之前讲解的前端开发模式MVVM，我们之前学习的vue是侧重于VM开发的，主要用于数据绑定到视图的，那么接下来我们学习的ElementUI就是一款侧重于V开发的前端框架，主要用于开发美观的页面的。

Element：是饿了么公司前端开发团队提供的一套基于 Vue 的网站组件库，用于快速构建网页。

Element 提供了很多组件（组成网页的部件）供我们使用。例如 超链接、按钮、图片、表格等等。如下图所示就是我们开发的页面和ElementUI提供的效果对比：可以发现ElementUI提供的各式各样好看的按钮



ElementUI的学习方式和我们之前的学习方式不太一样，对于ElementUI，我们作为一个后台开发者，只需要学会如何从ElementUI的官网拷贝组件到我们自己的页面中，并且做一些修改即可。其官网地址：<https://element.eleme.cn/#/zh-CN>，我们主要学习的是ElementUI中提供的常用组件，至于其他组件同学们可以通过我们这几个组件的学习掌握到ElementUI的学习技巧，然后课后自行学习。

## 4.2 快速入门

首先我们要掌握ElementUI的快速入门，接下来同学们就一起跟着步骤来操作一下。

首先，我们先要安装ElementUI的组件库，打开VS Code，停止之前的项目，然后在命令行输入如下命令：

```
1 npm install element-ui@2.15.3
```

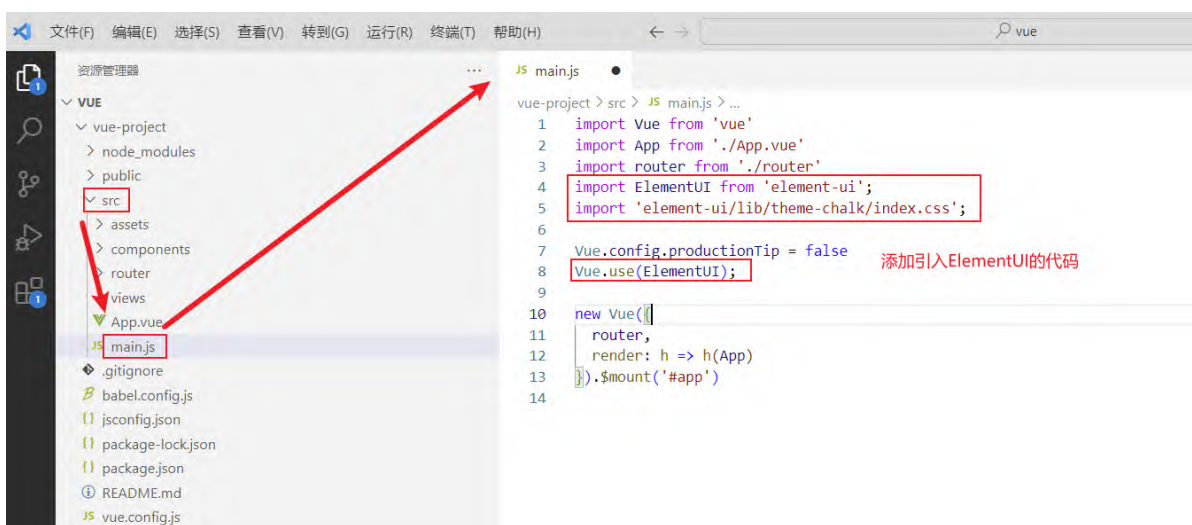
具体操作如下图所示：



然后我们需要在main.js这个入口js文件中引入ElementUI的组件库，其代码如下：

```
1 import ElementUI from 'element-ui';
2 import 'element-ui/lib/theme-chalk/index.css';
3
4 Vue.use(ElementUI);
```

具体操作如图所示：



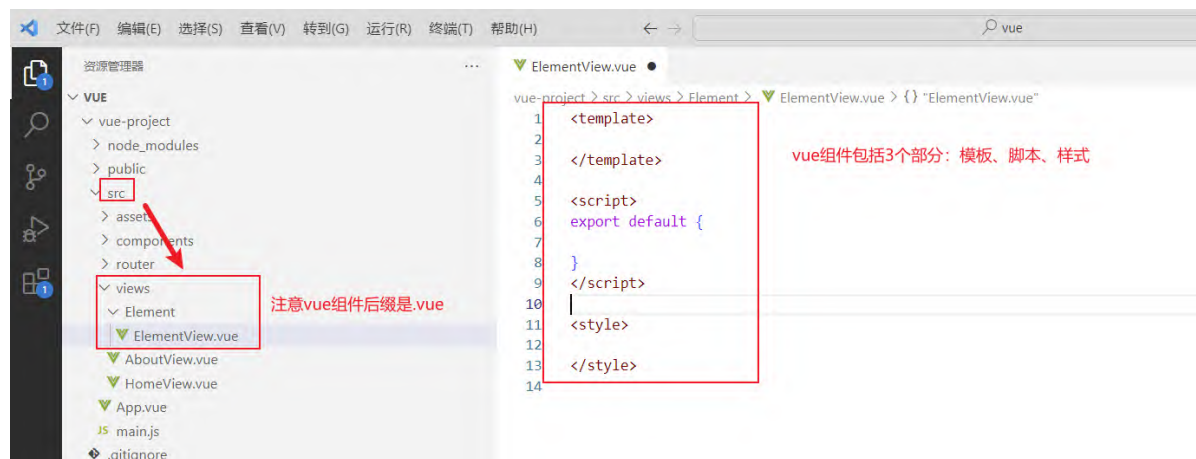
然后我们需要按照vue项目的开发规范，在src/views目录下创建一个vue组件文件，注意组件名称后缀是.vue，并且在组件文件中编写之前介绍过的基本组件语法，代码如下：

```

1  <template>
2
3  </template>
4
5  <script>
6  export default {
7
8  }
9  </script>
10
11 <style>
12
13 </style>

```

具体操作如图所示：



最后我们只需要去ElementUI的官网，找到组件库，然后找到按钮组件，抄写代码即可，具体操作如下图所示：



然后找到按钮的代码，如下图所示：

## Button 按钮

常用的操作按钮。

### 基础用法

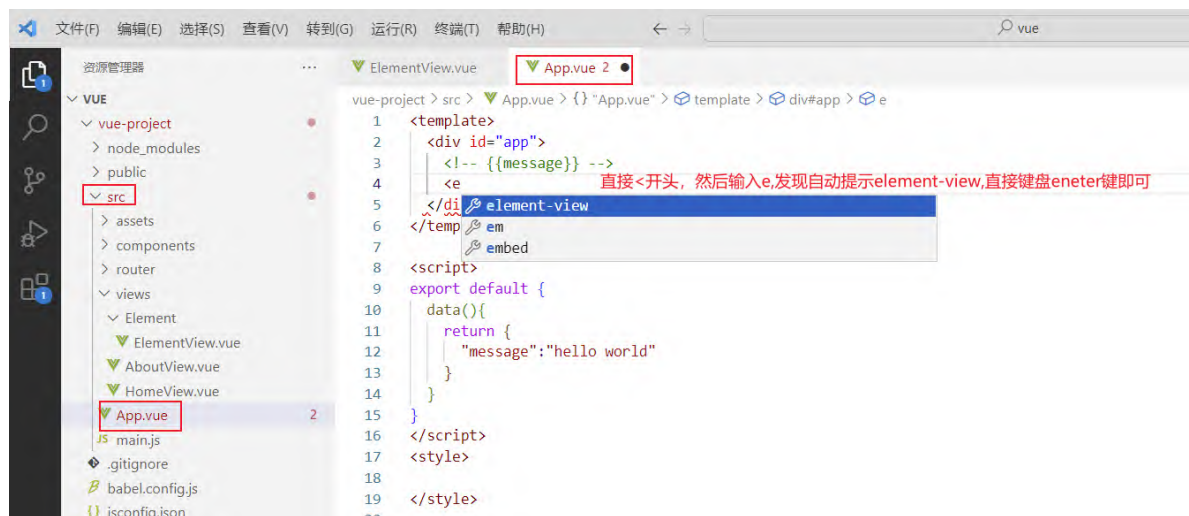
基础的按钮用法。



紧接着我们复制组件代码到我们的vue组件文件中，操作如下图所示：



最后，我们需要在默认访问的根组件src/App.vue中引入我们自定义的组件，具体操作步骤如下：



然后App.vue组件中的具体代码如下，代码是我们通过上述步骤引入element-view组件时自动生成的。

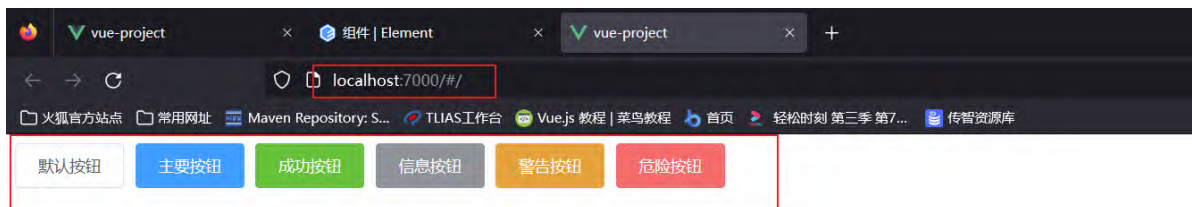
```
1 <template>
```

```

2     <div id="app">
3         <!-- {{message}} -->
4         <element-view></element-view>
5     </div>
6 </template>
7
8 <script>
9     import ElementView from './views/Element/ElementView.vue'
10    export default {
11        components: { ElementView },
12        data() {
13            return {
14                "message": "hello world"
15            }
16        }
17    }
18 </script>
19 <style>
20
21 </style>
22

```

然后运行我们的vue项目，浏览器直接访问之前的7000端口，展示效果如下图所示：



到此，我们ElementUI的入门程序编写成功

## 4.3 Element组件

接下来我们来学习一下ElementUI的常用组件，对于组件的学习比较简单，我们只需要参考官方提供的代码，然后复制粘贴即可。

### 4.3.1 Table表格



### 4.3.1.1 组件演示

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。

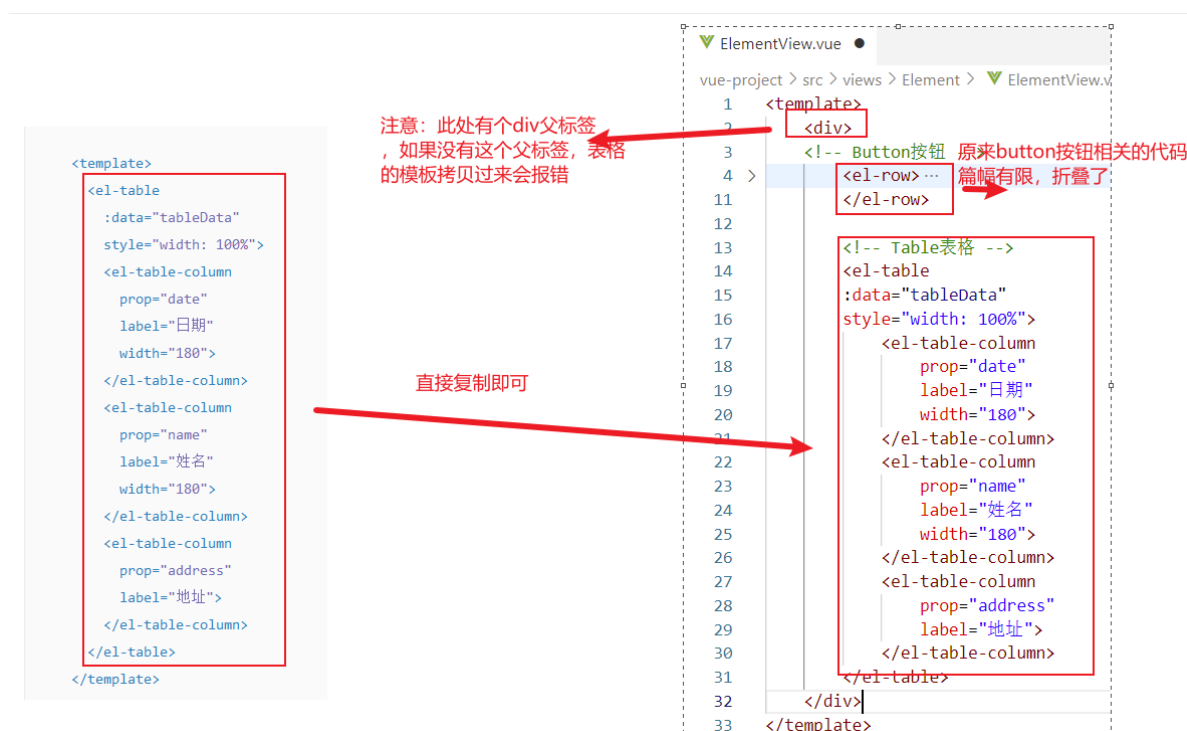
接下来我们通过代码来演示。

首先我们需要来到ElementUI的组件库中，找到表格组件，如下图所示：

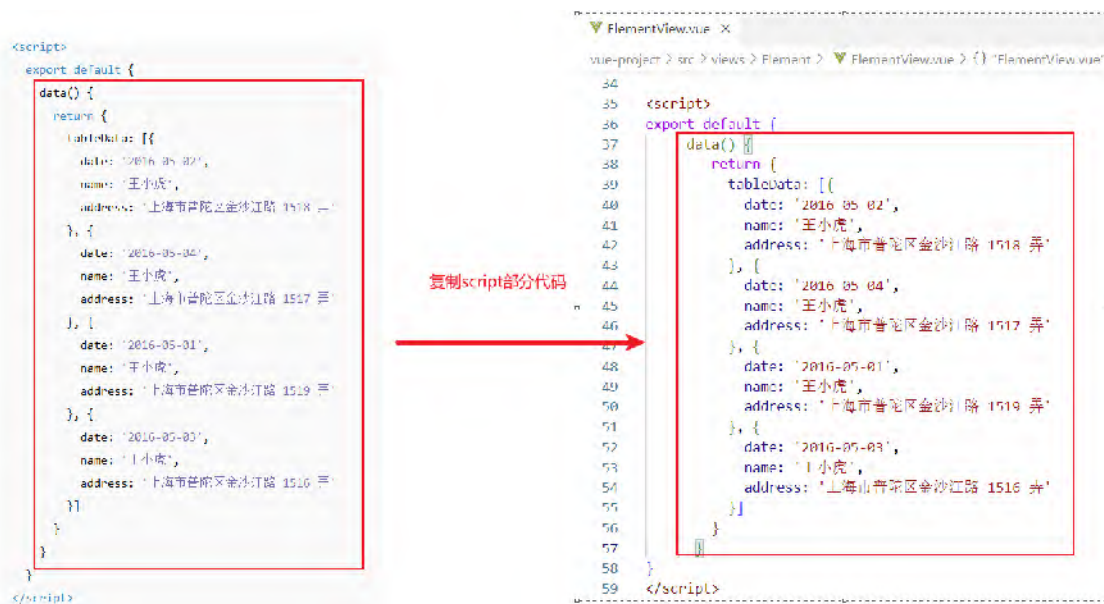


然后复制代码到我们之前的ElementVue.vue组件中，需要注意的是，我们组件包括了3个部分，如果官方有除了template部分之外的style和script都需要复制。具体操作如下图所示：

template模板部分：



script脚本部分



ElementView.vue组件文件整体代码如下：

```
1   <template>
2     <div>
3       <!-- Button按钮 -->
4       <el-row>
5         <el-button>默认按钮</el-button>
6         <el-button type="primary">主要按钮</el-button>
7         <el-button type="success">成功按钮</el-button>
8         <el-button type="info">信息按钮</el-button>
9         <el-button type="warning">警告按钮</el-button>
10        <el-button type="danger">危险按钮</el-button>
11      </el-row>
12
13      <!-- Table表格 -->
14      <el-table
15        :data="tableData"
16        style="width: 100%">
17        <el-table-column
18          prop="date"
19          label="日期"
20          width="180">
21        </el-table-column>
22        <el-table-column
23          prop="name"
24          label="姓名"
25          width="180">
26        </el-table-column>
27        <el-table-column
28          prop="address"
29          label="地址">
```

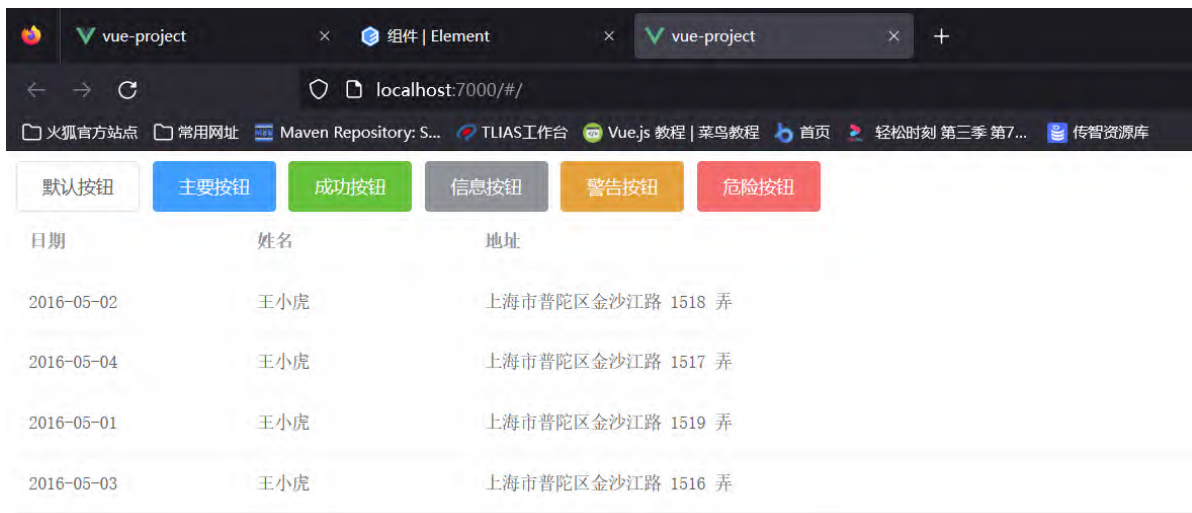
```

30         </el-table-column>
31     </el-table>
32 </div>
33 </template>
34
35 <script>
36 export default {
37     data() {
38         return {
39             tableData: [{
40                 date: '2016-05-02',
41                 name: '王小虎',
42                 address: '上海市普陀区金沙江路 1518 弄'
43             }, {
44                 date: '2016-05-04',
45                 name: '王小虎',
46                 address: '上海市普陀区金沙江路 1517 弄'
47             }, {
48                 date: '2016-05-01',
49                 name: '王小虎',
50                 address: '上海市普陀区金沙江路 1519 弄'
51             }, {
52                 date: '2016-05-03',
53                 name: '王小虎',
54                 address: '上海市普陀区金沙江路 1516 弄'
55             }]
56         }
57     }
58 }
59 </script>
60
61 <style>
62
63 </style>
64

```

此时回到浏览器，我们页面呈现如下效果：





#### 4.3.1.2 组件属性详解

那么我们的ElementUI是如何将数据模型绑定到视图的呢？主要通过如下几个属性：

- data：主要定义table组件的数据模型
- prop：定义列的数据应该绑定data中定义的具体的数据模型
- label：定义列的标题
- width：定义列的宽度

其具体示例含义如下图所示：



PS:Element组件的所有属性都可以在组件页面的最下方找到，如下图所示：



## 4.3.2 Pagination分页

### 4.3.2.1 组件演示

Pagination：分页组件，主要提供分页工具条相关功能。其展示效果图下图所示：



接下来我们通过代码来演示功能。

首先在官网找到分页组件，我们选择带背景色分页组件，如下图所示：



然后复制代码到我们的ElementView.vue组件文件的template中，拷贝如下代码：

```
1 <el-pagination
2   background
3   layout="prev, pager, next"
4   :total="1000">
5 </el-pagination>
```

浏览器打开呈现如下效果：



#### 4.3.2.2 组件属性详解

对于分页组件我们需要关注的是如下几个重要属性（可以通过查阅官网组件中最下面的组件属性详细说明得到）：

- background：添加北京颜色，也就是上图蓝色背景色效果。
- layout：分页工具条的布局，其具体值包含 sizes, prev, pager, next, jumper, ->, total, slot 这些值
- total：数据的总数量

然后根据官方分页组件提供的layout属性说明，如下图所示：

layout	组件布局，子组件名用逗号分隔	String	sizes, prev, pager, next, jumper, ->, total, slot	'prev, pager, next, jumper, ->, total'
--------	----------------	--------	---	--

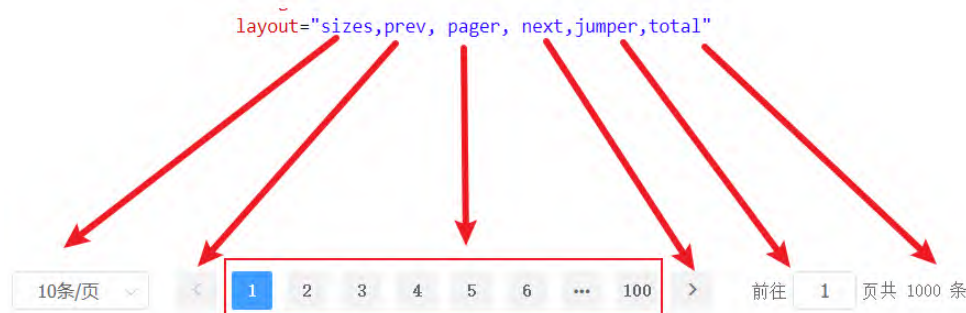
我们修改layout属性如下：

```
1 layout="sizes,prev, pager, next,jumper,total"
```

浏览器打开呈现如下效果：



发现在原来的功能上，添加了一些额外的功能，其具体对应关系如下图所示：



#### 4.3.2.3 组件事件详解

对于分页组件，除了上述几个属性，还有2个非常重要的事件我们需要去学习：

- size-change : pageSize 改变时会触发
- current-change : currentPage 改变时会触发

其官方详细解释含义如下图所示：

##### Events

事件名称	说明	回调参数
size-change	pageSize 改变时会触发	每页条数
current-change	currentPage 改变时会触发	当前页
prev-click	用户点击上一页按钮改变当前页后触发	当前页
next-click	用户点击下一页按钮改变当前页后触发	当前页

对于这2个事件的参考代码，我们同样可以通过官方提供的完整案例中找到，如下图所示：

Element

附加功能

根据场景需要，可以添加其他功能模块。

显示总数

共 1000 条 < 1 ... 3 4 5 6 7 ... 10 >

调整每页显示条数

100条/页 < 1 ... 3 4 5 6 7 ... 10 >

直接前往

< 1 ... 3 4 5 6 7 ... 10 > 前往 5 页

完整功能

共 400 条 100条/页 < 1 2 3 4 > 前往 4 页

此例是一个完整的用例，使用了 size-change 和 current-change 事件来处理页码大小和当前页会触发的事件。

page-sizes 接受一个整型数组，数组元素为展示的选择每次显示个数的选项，[100, 200, 300, 400] 表示四个选项，每页显示 100 个，200 个，300 个或者 400 个。

然后我们找到对应的代码，首先复制事件，复制代码如下：

```
1 @size-change="handleSizeChange"
2 @current-change="handleCurrentChange"
```

此时Panigation组件的template完整代码如下：

```
1 <!-- Pagination分页 -->
2 <el-pagination
3     @size-change="handleSizeChange"
4     @current-change="handleCurrentChange"
5     background
6     layout="sizes,prev, pager, next,jumper,total"
7     :total="1000">
8 </el-pagination>
```

紧接着需要复制事件需要的2个函数，需要注意methods属性和data同级，其代码如下：

```
1 methods: {
2     handleSizeChange(val) {
3         console.log(`每页 ${val} 条`);
4     },
5     handleCurrentChange(val) {
6         console.log(`当前页: ${val}`);
7     }
8 },
```

此时Panigation组件的script部分完整代码如下：

```
1 <script>
2 export default {
3     methods: {
4         handleSizeChange(val) {
5             console.log(`每页 ${val} 条`);
6         },
7         handleCurrentChange(val) {
8             console.log(`当前页: ${val}`);
9         }
10    },
11    data() {
12        return {
13            tableData: [{
14                date: '2016-05-02',
15                name: '王小虎',
16                address: '上海市普陀区金沙江路 1518 弄'
```

```

17         }, {
18             date: '2016-05-04',
19             name: '王小虎',
20             address: '上海市普陀区金沙江路 1517 弄'
21         }, {
22             date: '2016-05-01',
23             name: '王小虎',
24             address: '上海市普陀区金沙江路 1519 弄'
25         }, {
26             date: '2016-05-03',
27             name: '王小虎',
28             address: '上海市普陀区金沙江路 1516 弄'
29         }]
30     }
31 }
32 }
33 </script>

```

回到浏览器中，我们F12打开开发者控制台，然后切换当前页码和切换每页显示的数量，呈现如下效果：



### 4.3.3 Dialog对话框

#### 4.3.3.1 组件演示

Dialog：在保留当前页面状态的情况下，告知用户并承载相关操作。其企业开发应用场景示例如下图所示：





首先我们需要在ElementUI官方找到Dialog组件，如下图所示：



然后复制如下代码到我们的组件文件的template模块中

```

1    <br><br>
2    <!--Dialog 对话框 -->
3    <!-- Table -->
4    <el-button type="text" @click="dialogTableVisible = true">打开嵌套表格
      的 Dialog</el-button>
5
6    <el-dialog title="收货地址" :visible.sync="dialogTableVisible">
7      <el-table :data="gridData">
8        <el-table-column property="date" label="日期" width="150">
9          </el-table-column>
10       <el-table-column property="name" label="姓名" width="200">
11         </el-table-column>
12       <el-table-column property="address" label="地址"></el-table-
column>
13     </el-table>
14   </el-dialog>

```

并且复制数据模型script模块中：



```

1  gridData: [{
2      date: '2016-05-02',
3      name: '王小虎',
4      address: '上海市普陀区金沙江路 1518 弄'
5  }, {
6      date: '2016-05-04',
7      name: '王小虎',
8      address: '上海市普陀区金沙江路 1518 弄'
9  }, {
10     date: '2016-05-01',
11     name: '王小虎',
12     address: '上海市普陀区金沙江路 1518 弄'
13  }, {
14     date: '2016-05-03',
15     name: '王小虎',
16     address: '上海市普陀区金沙江路 1518 弄'
17  }],
18  dialogTableVisible: false,

```

其完整的script部分代码如下:

```

1  <script>
2  export default {
3      methods: {
4          handleSizeChange(val) {
5              console.log(`每页 ${val} 条`);
6          },
7          handleCurrentChange(val) {
8              console.log(`当前页: ${val}`);
9          }
10     },
11     data() {
12         return {
13             gridData: [{
14                 date: '2016-05-02',
15                 name: '王小虎',
16                 address: '上海市普陀区金沙江路 1518 弄'
17             }, {
18                 date: '2016-05-04',
19                 name: '王小虎',
20                 address: '上海市普陀区金沙江路 1518 弄'
21             }, {
22                 date: '2016-05-01',
23                 name: '王小虎',
24                 address: '上海市普陀区金沙江路 1518 弄'

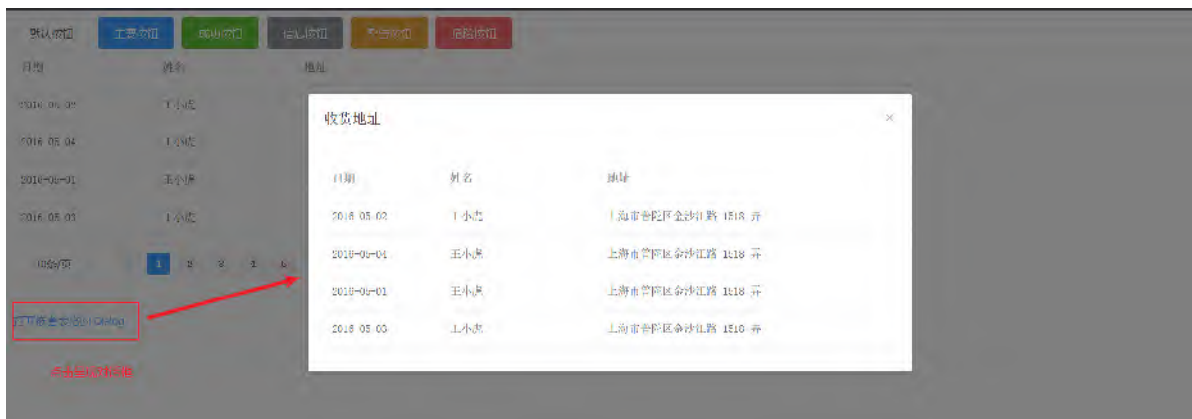
```

```

25     }, {
26         date: '2016-05-03',
27         name: '王小虎',
28         address: '上海市普陀区金沙江路 1518 弄'
29     }],
30     dialogTableVisible: false,
31     tableData: [{
32         date: '2016-05-02',
33         name: '王小虎',
34         address: '上海市普陀区金沙江路 1518 弄'
35     }, {
36         date: '2016-05-04',
37         name: '王小虎',
38         address: '上海市普陀区金沙江路 1517 弄'
39     }, {
40         date: '2016-05-01',
41         name: '王小虎',
42         address: '上海市普陀区金沙江路 1519 弄'
43     }, {
44         date: '2016-05-03',
45         name: '王小虎',
46         address: '上海市普陀区金沙江路 1516 弄'
47     }]
48     }
49     }
50 }
51 </script>

```

然后我们打开浏览器，点击按钮，呈现如下效果：

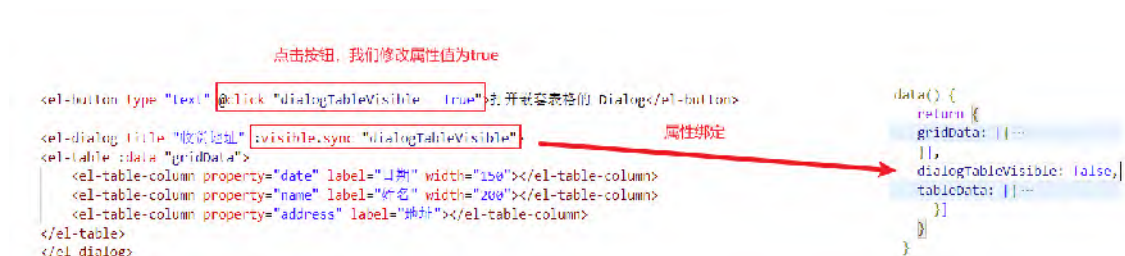


### 4.3.3.2 组件属性详解

那么ElementUI是如何做到对话框的显示与隐藏的呢？是通过如下的属性：

- `visible.sync` ：是否显示 Dialog

具体释意如下图所示：



`visible`属性绑定的`dialogTableVisible`属性一开始默认是`false`，所以对话框隐藏；然后我们点击按钮，触发事件，修改属性值为`true`，

然后对话框`visible`属性值为`true`，所以对话框呈现出来。

### 4.3.4 Form表单

#### 4.3.4.1 组件演示

Form 表单：由输入框、选择器、单选框、多选框等控件组成，用以收集、校验、提交数据。

表单在我们前端的开发中使用的还是比较多的，接下来我们学习这个组件，与之前的流程一样，我们首先需要在ElementUI的官方找到对应的组件示例：如下图所示：



我们的需求效果是：在对话框中呈现表单内容，类似如下图所示：



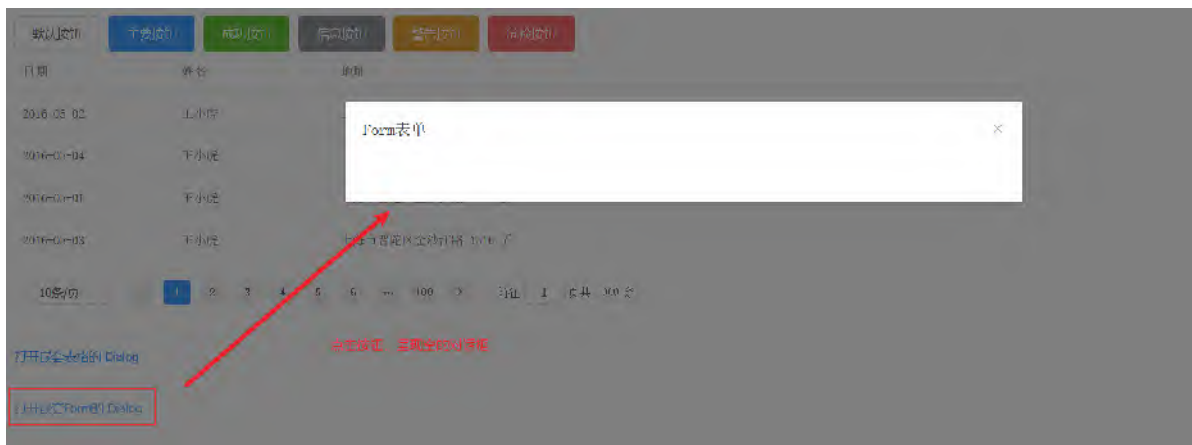
所以，首先我们要根据上一小结所学习的内容，制作一个新的对话框，其代码如下：

```
1 <br><br>
2 <!-- Dialog对话框-Form表单 -->
3 <el-button type="text" @click="dialogFormVisible = true">打开嵌套Form的
  Dialog</el-button>
4
5 <el-dialog title="Form表单" :visible.sync="dialogFormVisible">
6
7 </el-dialog>
```

还需要注意的是，针对这个新的对话框，我们需要在data中声明新的变量dialogFormVisible来控制对话框的隐藏与显示，代码如下：

```
1 dialogFormVisible: false,
```

打开浏览器，此时呈现如图所示的效果：



然后我们复制官网提供的template部分代码到我们的vue组件文件的Dialog组件中，但是，此处官方提供的表单项标签太多，所以我们只需要保留前面3个表单项组件，其他多余的删除，所以最终template部分代码如下：

```
1 <el-dialog title="Form表单" :visible.sync="dialogFormVisible">
2
```

```

3      <el-form ref="form" :model="form" label-width="80px">
4          <el-form-item label="活动名称">
5              <el-input v-model="form.name"></el-input>
6          </el-form-item>
7          <el-form-item label="活动区域">
8              <el-select v-model="form.region" placeholder="请
选择活动区域">
9                  <el-option label="区域一" value="shanghai"></el-
option>
10                 <el-option label="区域二" value="beijing"></el-
option>
11             </el-select>
12         </el-form-item>
13         <el-form-item label="活动时间">
14             <el-col :span="11">
15                 <el-date-picker type="date" placeholder="选择日
期" v-model="form.date1" style="width: 100%;"></el-date-picker>
16             </el-col>
17             <el-col class="line" :span="2">-</el-col>
18             <el-col :span="11">
19                 <el-time-picker placeholder="选择时间" v-
model="form.date2" style="width: 100%;"></el-time-picker>
20             </el-col>
21         </el-form-item>
22
23         <el-form-item>
24             <el-button type="primary" @click="onSubmit">立即
创建</el-button>
25             <el-button>取消</el-button>
26         </el-form-item>
27     </el-form>
28 </el-dialog>

```

观察上述代码，我们发现其中表单项标签使用了v-model双向绑定，所以我们需要在vue的数据模型中声明变量，同样可以从官方提供的代码中复制粘贴，但是我们需要去掉我们不需要的属性，通过观察上述代码，我们发现双向绑定的属性有4个，分别是

form.name, form.region, form.date1, form.date2, 所以最终数据模型如下：

```

<script>
  export default {
    data() {
      return {
        form: {
          name: '',
          region: '',
          date1: '',
          date2: '',
          delivery: false,
          type: [],
          resource: '',
          desc: ''
        }
      }
    },
    methods: {
      onSubmit() {
        console.log('submit!');
      }
    }
  }
}
</script>

```

数据模型，只保留前4个

```

1  form: {
2      name: '',
3      region: '',
4      date1: '',
5      date2: ''
6  },

```

同样，官方的代码中，在script部分中，还提供了onSubmit函数，表单的立即创建按钮绑定了此函数，我们可以输入表单的内容，而表单的内容是双向绑定到form对象的，所以我们修改官方的onSubmit函数如下即可，而且我们还需要关闭对话框，最终函数代码如下：

```

<el-form-item>
  <el-button type="primary" @click="onSubmit">立即创建</el-button>
  <el-button>取消</el-button>
</el-form-item>
</el-form>
<script>
  export default {
    data() {
      return {
        form: {
          name: '',
          region: '',
          date1: '',
          date2: '',
          delivery: false,
          type: [],
          resource: '',
          desc: ''
        }
      }
    },
    methods: {
      onSubmit() {
        console.log('submit!');
      }
    }
  }
}

```

事件绑定：我们需要修改函数，从而输出表单的内容

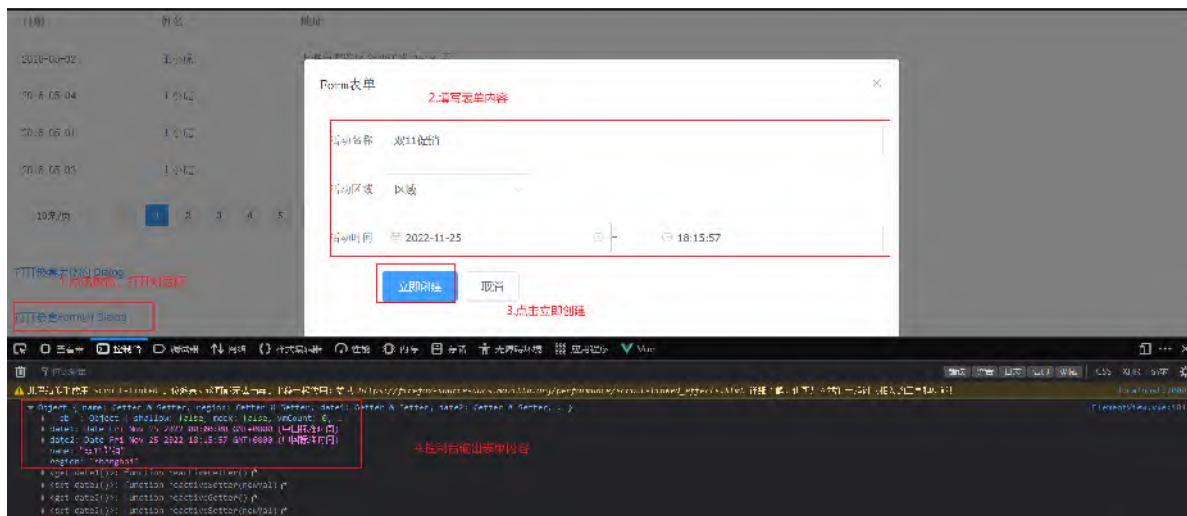


```

1  onSubmit() {
2      console.log(this.form); //输出表单内容到控制台
3      this.dialogFormVisible=false; //关闭表单案例的对话框
4  }

```

然后打开浏览器，我们打开对话框，并且输入表单内容，点击立即创建按钮，呈现如下效果；



最终vue组件完整代码如下，同学们可以针对form表单案例，参考该案例对应的template部分和script部分代码

```

1  <template>
2      <div>
3          <!-- Button按钮 -->
4          <el-row>
5              <el-button>默认按钮</el-button>
6              <el-button type="primary">主要按钮</el-button>
7              <el-button type="success">成功按钮</el-button>
8              <el-button type="info">信息按钮</el-button>
9              <el-button type="warning">警告按钮</el-button>
10             <el-button type="danger">危险按钮</el-button>
11         </el-row>
12
13         <!-- Table表格 -->
14         <el-table
15             :data="tableData"
16             style="width: 100%">
17             <el-table-column
18                 prop="date"
19                 label="日期"
20                 width="180">
21             </el-table-column>
22             <el-table-column

```

```
23         prop="name"
24         label="姓名"
25         width="180">
26     </el-table-column>
27     <el-table-column
28         prop="address"
29         label="地址">
30     </el-table-column>
31 </el-table>
32
33 <br>
34 <!-- Pagination分页 -->
35 <el-pagination
36     @size-change="handleSizeChange"
37     @current-change="handleCurrentChange"
38     background
39     layout="sizes,prev, pager, next,jumper,total"
40     :total="1000">
41 </el-pagination>
42
43 <br><br>
44 <!--Dialog 对话框 -->
45 <!-- Table -->
46     <el-button type="text" @click="dialogTableVisible = true">
打开嵌套表格的 Dialog</el-button>
47
48     <el-dialog title="收货地址"
:visible.sync="dialogTableVisible">
49         <el-table :data="gridData">
50             <el-table-column property="date" label="日期"
width="150"></el-table-column>
51             <el-table-column property="name" label="姓名"
width="200"></el-table-column>
52             <el-table-column property="address" label="地址"></el-
table-column>
53         </el-table>
54     </el-dialog>
55
56 <br><br>
57 <!-- Dialog对话框-Form表单 -->
58     <el-button type="text" @click="dialogFormVisible = true">打
开嵌套Form的 Dialog</el-button>
59
```

```
60         <el-dialog title="Form表单"
        :visible.sync="dialogFormVisible">
61
62         <el-form ref="form" :model="form" label-width="80px">
63             <el-form-item label="活动名称">
64                 <el-input v-model="form.name"></el-input>
65             </el-form-item>
66             <el-form-item label="活动区域">
67                 <el-select v-model="form.region"
placeholder="请选择活动区域">
68                     <el-option label="区域一" value="shanghai"></el-
option>
69                     <el-option label="区域二" value="beijing"></el-
option>
70                 </el-select>
71             </el-form-item>
72             <el-form-item label="活动时间">
73                 <el-col :span="11">
74                     <el-date-picker type="date" placeholder="选择日
期" v-model="form.date1" style="width: 100%;"></el-date-picker>
75                 </el-col>
76                 <el-col class="line" :span="2">-</el-col>
77                 <el-col :span="11">
78                     <el-time-picker placeholder="选择时间" v-
model="form.date2" style="width: 100%;"></el-time-picker>
79                 </el-col>
80             </el-form-item>
81
82             <el-form-item>
83                 <el-button type="primary" @click="onSubmit">立即
创建</el-button>
84                 <el-button>取消</el-button>
85             </el-form-item>
86         </el-form>
87     </el-dialog>
88 </div>
89 </template>
90
91 <script>
92 export default {
93     methods: {
94         handleSizeChange(val) {
95             console.log(`每页 ${val} 条`);
96         },
```

```
97     handleCurrentChange(val) {
98         console.log(`当前页: ${val}`);
99     },
100     //表单案例的提交事件
101     onSubmit() {
102         console.log(this.form); //输出表单内容到控制台
103         this.dialogFormVisible=false; //关闭表案例的对话框
104     }
105 },
106 data() {
107     return {
108         //表单案例的数据双向绑定
109         form: {
110             name: '',
111             region: '',
112             date1: '',
113             date2:''
114         },
115         gridData: [{
116             date: '2016-05-02',
117             name: '王小虎',
118             address: '上海市普陀区金沙江路 1518 弄'
119         }, {
120             date: '2016-05-04',
121             name: '王小虎',
122             address: '上海市普陀区金沙江路 1518 弄'
123         }, {
124             date: '2016-05-01',
125             name: '王小虎',
126             address: '上海市普陀区金沙江路 1518 弄'
127         }, {
128             date: '2016-05-03',
129             name: '王小虎',
130             address: '上海市普陀区金沙江路 1518 弄'
131         }
132     ],
133     dialogTableVisible: false,
134     dialogFormVisible: false, //控制form表单案例的对话框
135     tableData: [{
136         date: '2016-05-02',
137         name: '王小虎',
138         address: '上海市普陀区金沙江路 1518 弄'
139     }, {
140         date: '2016-05-04',
141         name: '王小虎',
```

```

141         address: '上海市普陀区金沙江路 1517 弄'
142     }, {
143         date: '2016-05-01',
144         name: '王小虎',
145         address: '上海市普陀区金沙江路 1519 弄'
146     }, {
147         date: '2016-05-03',
148         name: '王小虎',
149         address: '上海市普陀区金沙江路 1516 弄'
150     }]
151     }
152 }
153 }
154 </script>
155
156 <style>
157
158 </style>
159

```

## 4.4 案例

### 4.4.1 案例需求

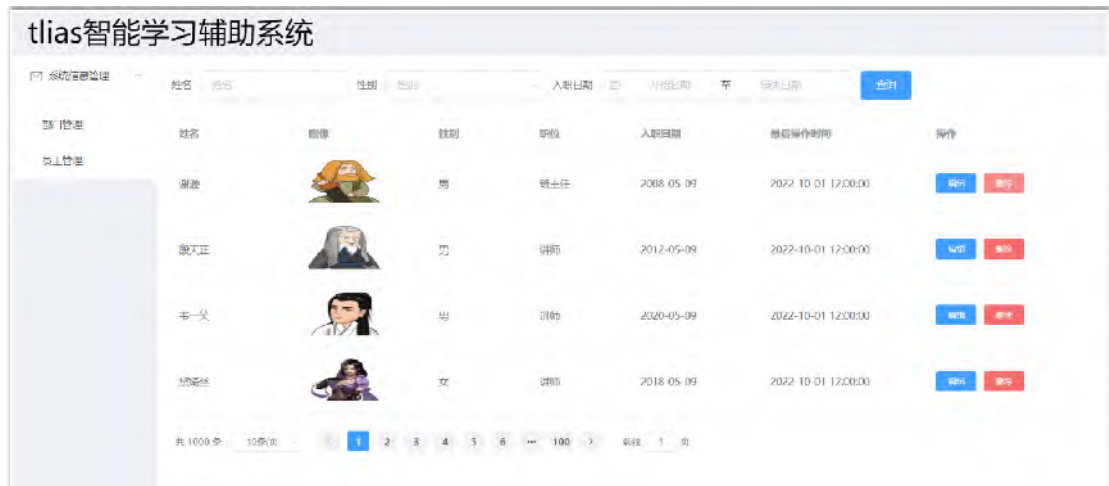
参考 [资料/页面原型/tlias智能学习辅助系统/首页.html](#) 文件，浏览器打开，点击页面中的左侧栏的员工管理，如下所示：

The screenshot shows the 'tlias 智能学习辅助系统' interface. The sidebar on the left contains several menu items, with '员工管理' (Employee Management) highlighted by a red box. The main content area displays a table of employee information. The table has columns for '姓名' (Name), '性别' (Gender), '职位' (Position), '入职时间' (Entry Time), and '最后操作时间' (Last Operation Time). The table lists several employees, including '王小虎' and '王小虎'.

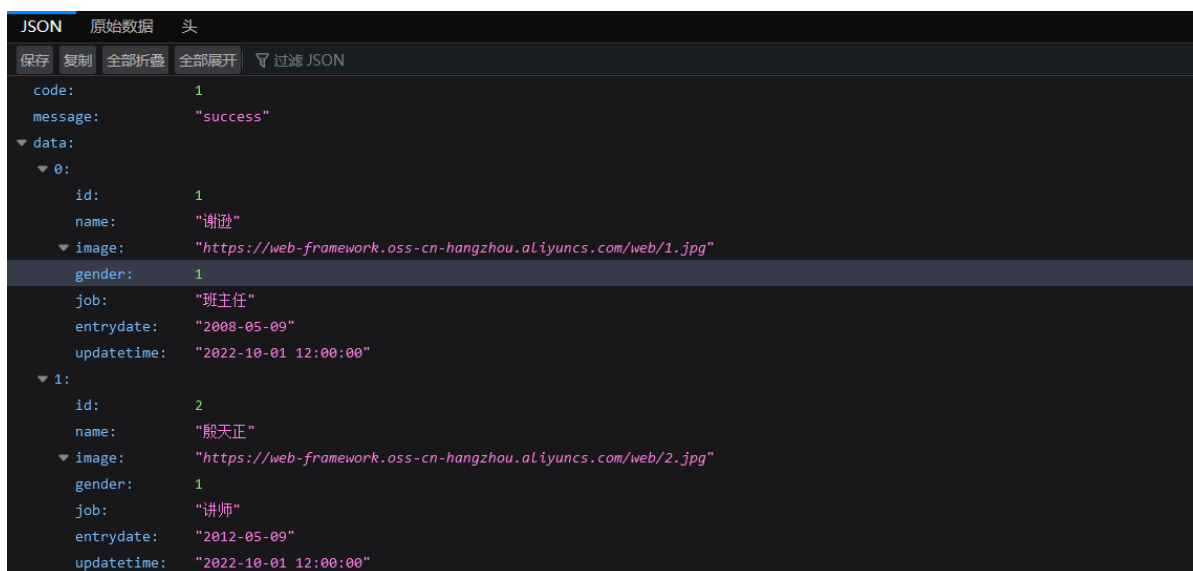
姓名	性别	职位	入职时间	最后操作时间	操作
王小虎	男	讲师	2016-07-22	2022-07-21 15:00:00	编辑 删除
王小虎	男	讲师	2016-07-22	2022-07-21 15:00:00	编辑 删除
王小虎	男	讲师	2016-07-22	2022-07-21 15:00:00	编辑 删除
王小虎	男	讲师	2016-07-22	2022-07-21 15:00:00	编辑 删除
王小虎	男	讲师	2016-07-22	2022-07-21 15:00:00	编辑 删除
王小虎	男	讲师	2016-07-22	2022-07-21 15:00:00	编辑 删除
王小虎	男	讲师	2016-07-22	2022-07-21 15:00:00	编辑 删除
王小虎	男	讲师	2016-07-22	2022-07-21 15:00:00	编辑 删除
王小虎	男	讲师	2016-07-22	2022-07-21 15:00:00	编辑 删除
王小虎	男	讲师	2016-07-22	2022-07-21 15:00:00	编辑 删除

需求说明：

1. 制作类似格式的页面  
即上面是标题，左侧栏是导航，右侧是数据展示区域
2. 右侧需要展示搜索表单
3. 右侧表格数据是动态展示的，数据来自于后台
4. 实际示例效果如下图所示：



数据Mock地址：<http://yapi.smart-xwork.cn/mock/169327/emp/list>，浏览器打开，数据格式如下图所示：



通过观察数据，我们发现返回的json数据的data属性中，才是返回的人员列表信息



## 4.4.2 案例分析

整个案例相对来说功能比较复杂，需求较多，所以我们需要先整体，后局部细节。整个页面我们可以分为3个部分，如下图所示：



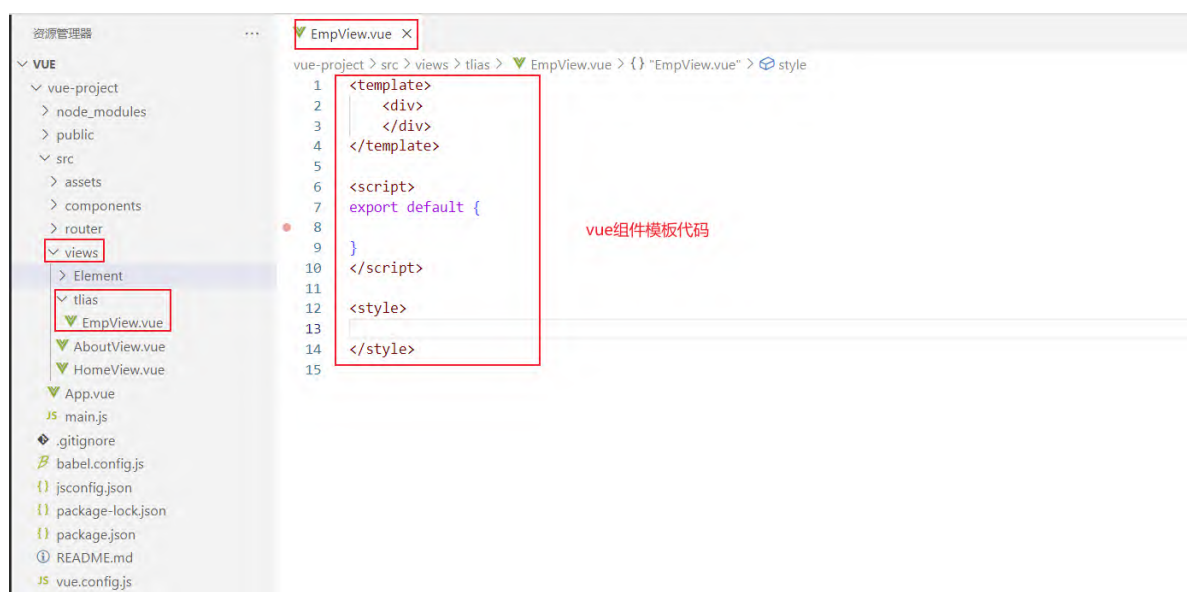
一旦这样拆分，那么我们的思路就清晰了，主要步骤如下：

1. 创建页面，完成页面的整体布局规划
2. 然后分别针对3个部分进行各自组件的具体实现
3. 针对于右侧核心内容展示区域，需要使用异步加载数据，以表格渲染数据

## 4.4.3 代码实现

### 4.4.3.1 环境搭建

首先我们来到VS Code中，在views目录下创建 `tlia/EmpView.vue` 这个vue组件，并且编写组件的基本模板代码，其效果如下图所示：其中模板代码在之前的案例中已经提供，此处不再赘述



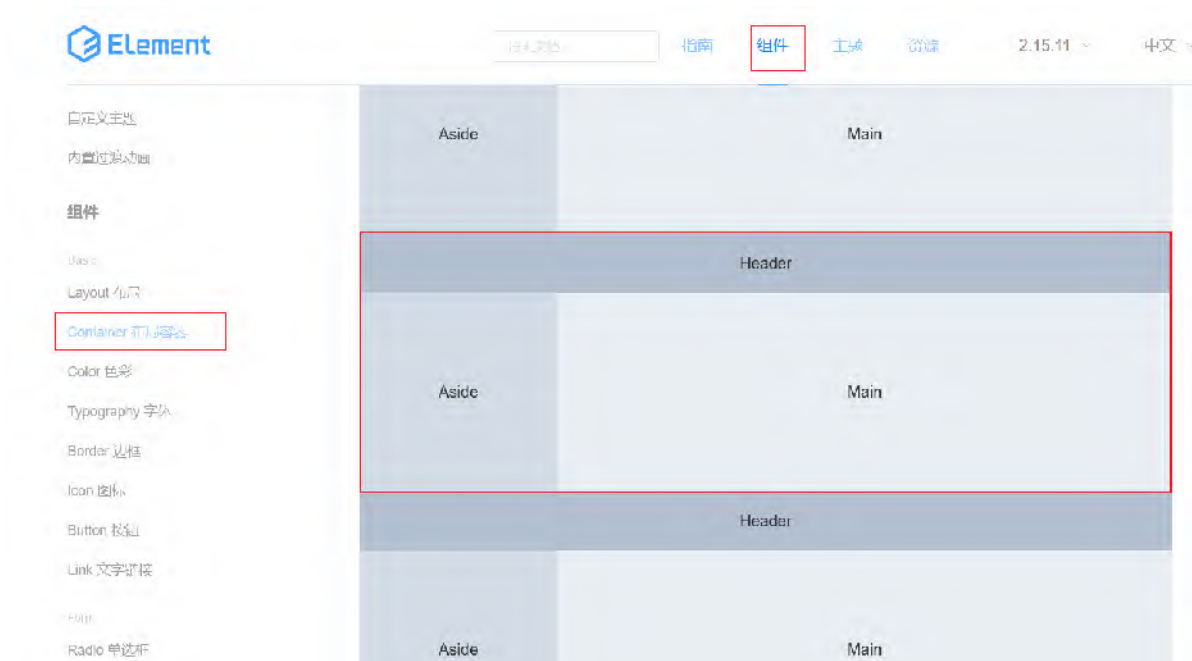
并且需要注意的是，我们默认访问的是 `App.vue` 这个组件，而我们 `App.vue` 这个组件之前是引入了 `element-view` 这个组件，此时我们需要修改成引入 `emp-view` 这个组件，并且注释掉之前的 `element-view` 这个组件，此时 `App.vue` 整体代码如下：

```
1   <template>
2     <div id="app">
3       <!-- {{message}} -->
4       <!-- <element-view></element-view> -->
5       <emp-view></emp-view>
6     </div>
7   </template>
8
9   <script>
10    import EmpView  './views/tlias/EmpView.vue'
11    // import ElementView  './views/Element/ElementView.vue'
12    export default {
13      components: {EmpView },
14      data() {
15        return {
16          "message": "hello world"
17        }
18      }
19    }
20  </script>
21  <style>
22
23  </style>
24
```

打开浏览器，我们发现之前的element案例内容没了，从而呈现的是一片空白，那么接下来我们就可以继续开发了。

#### 4.4.3.2 整体布局

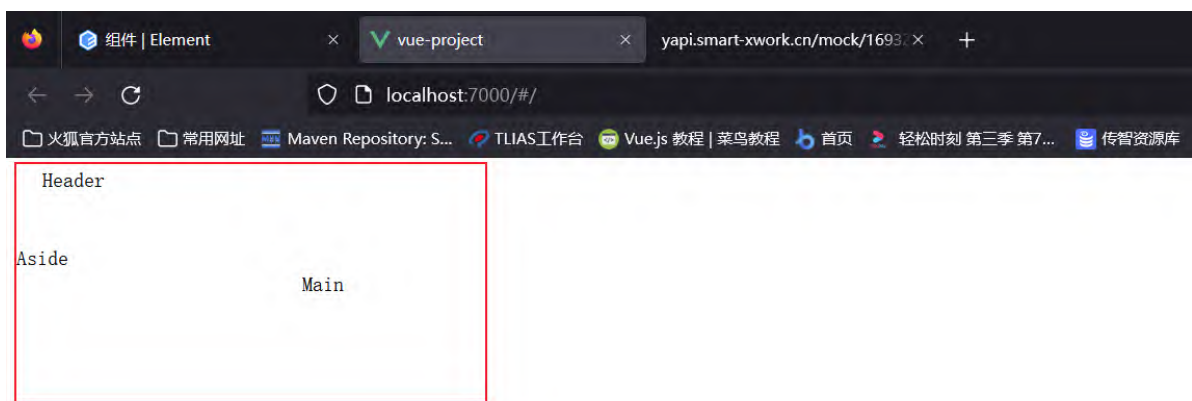
此处肯定不需要我们自己去布局的，我们直接来到ElementUI的官网，找到布局组件，如下图所示：



从官网提供的示例，我们发现由现成的满足我们需求的布局，所以我们只需要做一位代码搬运工即可。拷贝官方提供的如下代码直接粘贴到我们EmpView.vue组件的template模块中即可：

```
1 <el-container>
2   <el-header>Header</el-header>
3   <el-container>
4     <el-aside width="200px">Aside</el-aside>
5     <el-main>Main</el-main>
6   </el-container>
7 </el-container>
```

打开浏览器，此时呈现如下效果：



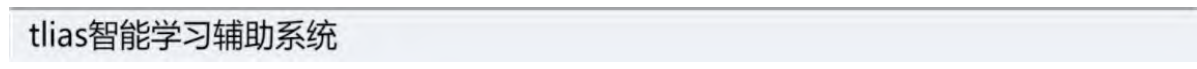
因为我们没有拷贝官方提供的css样式，所以和官方案例的效果不太一样，但是我们需要的布局格式已经有，具体内容我们有自己的安排。首先我们需要调整整体布局的高度，所以我们需要在<el-container>上添加一些样式，代码如下：

```
1 <!-- 设置最外层容器高度为700px,在加上一个很细的边框 -->
2 <el-container style="height: 700px; border: 1px solid #eee">
```

到此我们布局功能就完成了

### 4.4.3.3 顶部标题

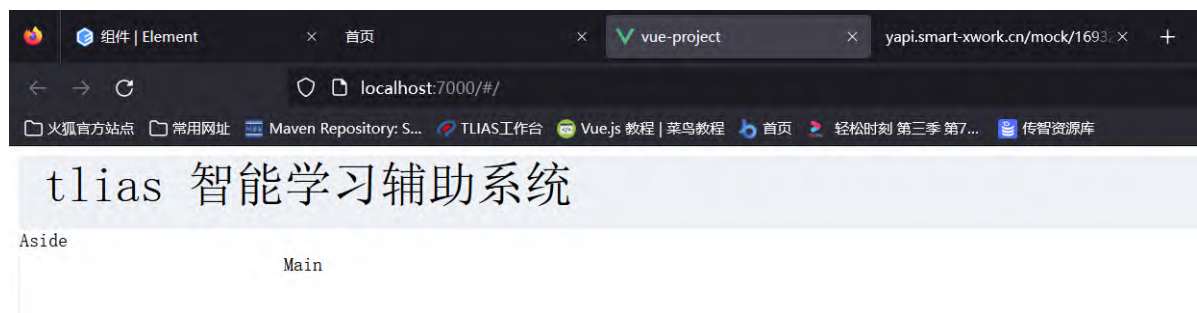
对于顶部，我们需要实现的效果如下图所示：



所以我们需要修改顶部的文本内容，并且提供背景色的css样式，具体代码如下：

```
1 <el-header style="font-size:40px;background-color: rgb(238, 241, 246)">tliaS 智能学习辅助系统</el-header>
```

此时浏览器打开，呈现效果如下图所示：



至此，我们的顶部标题就搞定了

此时整体代码如下：

```
1 <template>
2   <div>
3     <!-- 设置最外层容器高度为700px,在加上一个很细的边框 -->
4     <el-container style="height: 700px; border: 1px solid #eee">
5       <el-header style="font-size:40px;background-color:
6         rgb(238, 241, 246)">tliaS 智能学习辅助系统</el-header>
7       <el-container>
8         <el-aside width="200px">Aside</el-aside>
9         <el-main>Main</el-main>
10      </el-container>
11    </div>
12  </template>
13
14  <script>
15    export default {
16
17    }
18  </script>
19
20  <style>
21
22  </style>
```

#### 4.4.3.4 左侧导航栏

接下来我们来实现左侧导航栏，那么还是在上述布局组件中提供的案例，找到左侧栏的案例，如下图所示：



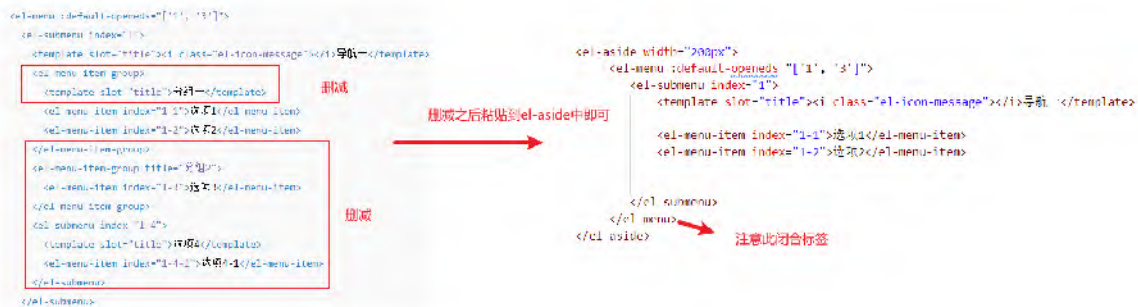
所以我们依然只需要搬运代码，然后做简单修改即可。官方提供的导航太多，我们不需要，所以我们需要做删减，在我们的左侧导航栏中粘贴如下代码即可：

```

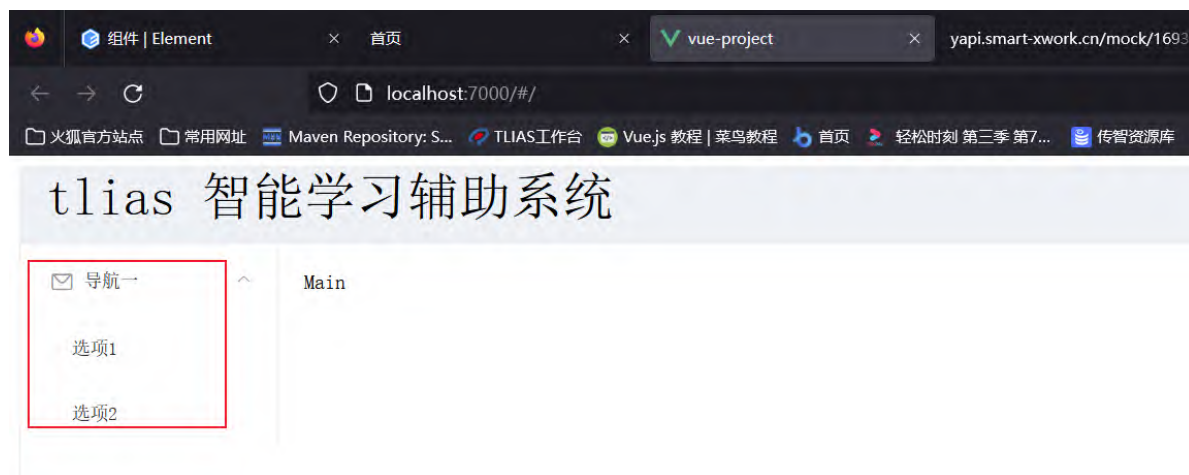
1   <el-menu :default-openeds=["1", '3']>
2     <el-submenu index="1">
3       <template slot="title"><i class="el-icon-message"></i>导航一
4       </template>
5       <el-menu-item index="1-1">选项1</el-menu-item>
6       <el-menu-item index="1-2">选项2</el-menu-item>
7
8
9     </el-submenu>
10  </el-menu>

```

删减前后对比图：



然后我们打开浏览器，展示如下内容：



最后我们只需要替换文字内容即可。

此时整体代码如下：

```
1 <template>
2   <div>
3     <!-- 设置最外层容器高度为700px,在加上一个很细的边框 -->
4     <el-container style="height: 700px; border: 1px solid #eee">
5       <el-header style="font-size:40px;background-color:
6         rgb(238, 241, 246)">tlias 智能学习辅助系统</el-header>
7       <el-container>
8         <el-aside width="200px">
9           <el-menu :default-openeds="['1', '3']">
10            <el-submenu index="1">
11              <template slot="title"><i class="el-
12                icon-message"></i>系统信息管理</template>
13              <el-menu-item index="1-1">部门管理</el-
14                menu-item>
15              <el-menu-item index="1-2">员工管理</el-
16                menu-item>
17            </el-submenu>
18          </el-menu>
```



```

18         </el-aside>
19         <el-main>
20
21         </el-main>
22     </el-container>
23 </el-container>
24 </div>
25 </template>
26
27 <script>
28 export default {
29
30 }
31 </script>
32
33 <style>
34
35 </style>
36

```

#### 4.4.3.5 右侧核心内容

##### 4.4.3.5.1 表格编写

右侧显示的是表单和表格，首先我们先来完成表格的制作，我们同样在官方直接找表格组件，也可以直接通过我们上述容器组件中提供的案例中找到表格相关的案例，如下图所示：



然后找到表格的代码，复制到我们布局容器的主题区域，template模块代码如下：

```

1   <el-table :data="tableData">
2       <el-table-column prop="date" label="日期" width="140">
3       </el-table-column>
4       <el-table-column prop="name" label="姓名" width="120">
5       </el-table-column>
6       <el-table-column prop="address" label="地址">
7       </el-table-column>
8   </el-table>

```

表格是有数据模型的绑定的，所以我们需要继续拷贝数据模型，代码如下：

```

1   data() {
2       return {
3           tableData: [
4               {
5                   date: '2016-05-02',
6                   name: '王小虎',
7                   address: '上海市普陀区金沙江路 1518 弄'
8               }
9           ]
10      }

```

浏览器打开，呈现如下效果：



但是这样的表格和数据并不是我们所需要的，所以，接下来我们需要修改表格，添加列，并且修改列名。代码如下：

```

1 <el-table-column prop="name" label="姓名" width="180"></el-table-
  column>
2 <el-table-column prop="image" label="图像" width="180"></el-table-
  column>
3 <el-table-column prop="gender" label="性别" width="140"></el-table-
  column>
4 <el-table-column prop="job" label="职位" width="140"></el-table-
  column>
5 <el-table-column prop="entrydate" label="入职日期" width="180"></el-
  table-column>
6 <el-table-column prop="updatetime" label="最后操作时间" width="230">
  </el-table-column>
7 <el-table-column label="操作" >
8   <el-button type="primary" size="mini">编辑</el-button>
9   <el-button type="danger" size="mini">删除</el-button>
10 </el-table-column>

```

需要注意的是，我们列名的prop属性值得内容并不是乱写的，因为我们将需要绑定后台的数据的，所以如下图所示：



并且此时我们data中之前的数据模型就不可用了，所以需要清空数据，设置为空数组，代码 如下：

```

1 data() {
2   return {
3     tableData: [
4
5     ]
6   }
7 }

```

此时打开浏览器，呈现如下效果：



此时整体页面代码如下：

```

1  <template>
2    <div>
3      <!-- 设置最外层容器高度为700px,在加上一个很细的边框 -->
4      <el-container style="height: 700px; border: 1px solid #eee">
5        <el-header style="font-size:40px;background-color:
6          rgb(238, 241, 246)">tliaas 智能学习辅助系统</el-header>
7        <el-container>
8          <el-aside width="200px">
9            <el-menu :default-openeds="['1', '3']">
10              <el-submenu index="1">
11                <template slot="title"><i class="el-
12                  icon-message"></i>系统信息管理</template>
13                <el-menu-item index="1-1">部门管理</el-
14                  menu-item>
15                <el-menu-item index="1-2">员工管理</el-
16                  menu-item>
17              </el-submenu>
18            </el-menu>
19          </el-aside>
20          <el-main>
21            <el-table :data="tableData">
22              <el-table-column prop="name" label="姓
23                名" width="180"></el-table-column>
24              <el-table-column prop="image" label="图
25                像" width="180"></el-table-column>
26              <el-table-column prop="gender" label="性
27                别" width="140"></el-table-column>
28              <el-table-column prop="job" label="职
29                位" width="140"></el-table-column>
30              <el-table-column prop="entrydate" label="入职
31                日期" width="180"></el-table-column>

```

```

26         <el-table-column prop="updatetime" label="最
    后操作时间" width="230"></el-table-column>
27         <el-table-column label="操作" >
28             <el-button type="primary" size="mini">编
    辑</el-button>
29             <el-button type="danger" size="mini">删除
    </el-button>
30         </el-table-column>
31     </el-table>
32
33     </el-main>
34 </el-container>
35 </el-container>
36 </div>
37 </template>
38
39 <script>
40 export default {
41     data() {
42         return {
43             tableData: [
44
45             ]
46         }
47     }
48 }
49 </script>
50
51 <style>
52
53 </style>
54

```

#### 4.4.3.5.2 表单编写

在表格的上方，还需要如下图所示的表单：

The image shows a horizontal search form. It consists of four input fields: '姓名' (Name), '性别' (Gender), '入职日期' (Start Date), and '结束日期' (End Date). The '姓名' and '性别' fields have placeholder text. The '入职日期' and '结束日期' fields have date pickers. To the right of these fields is a blue button labeled '查询' (Search).

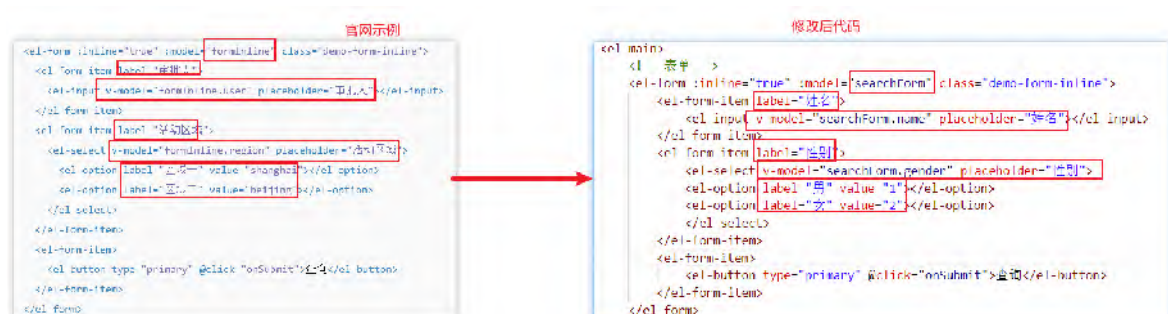
所以接下来我们需要去ElementUI官网，在表单组件中找到与之类似的示例，加以修改从而打成我们想要的效果，官方示例如下图所示：



所以我们直接拷贝代码主体区域的table组件的上方即可，并且我们需要修改数据绑定的变量名，最终代码如下：

```
1      <!-- 表单 -->
2      <el-form :inline="true" :model="searchForm" class="demo-form-inline">
3          <el-form-item label="姓名">
4              <el-input v-model="searchForm.name" placeholder="姓名"></el-input>
5          </el-form-item>
6          <el-form-item label="性别">
7              <el-select v-model="searchForm.gender" placeholder="性别">
8                  <el-option label="男" value="1"></el-option>
9                  <el-option label="女" value="2"></el-option>
10             </el-select>
11         </el-form-item>
12         <el-form-item>
13             <el-button type="primary" @click="onSubmit">查询</el-button>
14         </el-form-item>
15     </el-form>
```

代码修改前后对比图：



既然我们表单使用v-model进行数据的双向绑定了，所以我们紧接着需要在data中定义searchForm的数据模型，代码如下：

```
1    data() {  
2        return {  
3            tableData: [  
4  
5            ],  
6            searchForm: {  
7                name: '',  
8                gender: ''  
9            }  
10        }  
11    }
```

而且，表单的提交按钮，绑定了onSubmit函数，所以我们还需要在methods中定义onSubmit函数，代码如下：

注意的是methods属性需要和data属性同级

```
1    methods: {  
2        onSubmit: function () {  
3            console.log(this.searchForm);  
4        }  
5    }
```

浏览器打开如图所示：



可以发现我们还缺少一个时间，所以可以从elementUI官网找到日期组件，如下图所示：



参考官方代码，然后在我们之前的表单中添加一个日期表单，具体代码如下：



```

1   </el-form-item>
2     <el-form-item label="入职日期">
3       <el-date-picker
4         v-model="searchForm.entrydate"
5         type="daterange"
6         range-separator="至"
7         start-placeholder="开始日期"
8         end-placeholder="结束日期">
9     </el-date-picker>
10  </el-form-item>

```

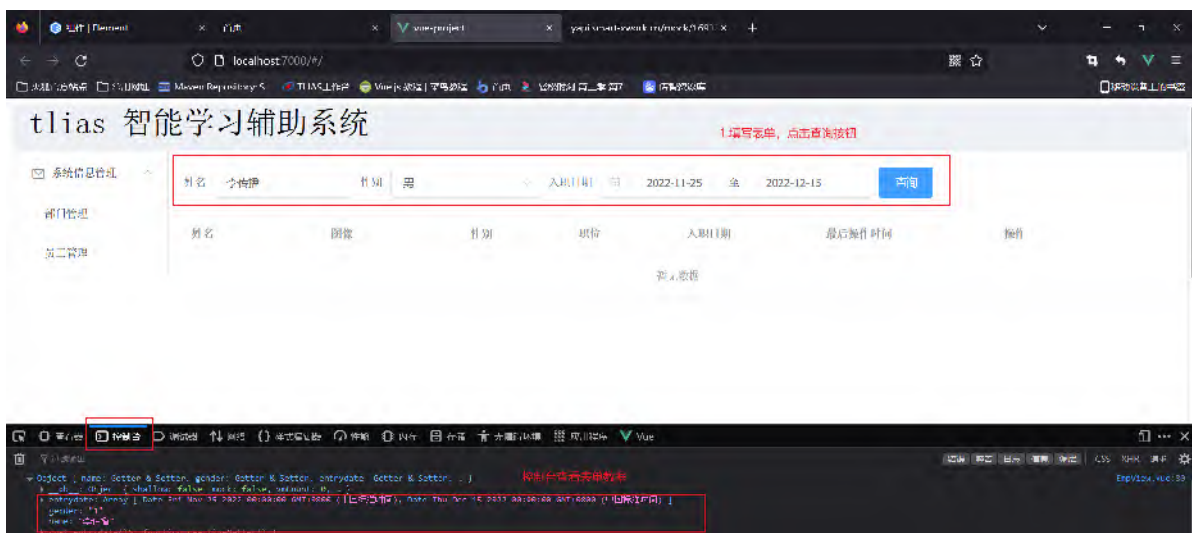
我们添加了双向绑定，所以我们需要在data的searchForm中定义出来，需要注意的是这个日期包含2个值，所以我们定义为数组，代码如下：

```

1   searchForm: {
2     name: '',
3     gender: '',
4     entrydate: []
5   }

```

此时我们打开浏览器，填写表单，并且点击查询按钮，查看浏览器控制台，可以看到表单的内容，效果如下图所示：



此时完整代码如下所示：

```

1   <template>
2     <div>
3       <!-- 设置最外层容器高度为700px,在加上一个很细的边框 -->
4       <el-container style="height: 700px; border: 1px solid #eee">
5         <el-header style="font-size:40px;background-color:
6           rgb(238, 241, 246)">tlia 智能学习辅助系统</el-header>

```

```
6         <el-container>
7             <el-aside width="200px">
8                 <el-menu :default-openeds="['1', '3']">
9                     <el-submenu index="1">
10                         <template slot="title"><i class="el-
11 icon-message"></i>系统信息管理</template>
12
13                         <el-menu-item index="1-1">部门管理</el-
14 menu-item>
15
16                         <el-menu-item index="1-2">员工管理</el-
17 menu-item>
18
19                     </el-submenu>
20                 </el-menu>
21             </el-aside>
22             <el-main>
23                 <!-- 表单 -->
24                 <el-form :inline="true" :model="searchForm"
25 class="demo-form-inline">
26                     <el-form-item label="姓名">
27                         <el-input v-model="searchForm.name"
28 placeholder="姓名"></el-input>
29                     </el-form-item>
30                     <el-form-item label="性别">
31                         <el-select v-model="searchForm.gender"
32 placeholder="性别">
33                             <el-option label="男" value="1"></el-
34 option>
35                             <el-option label="女" value="2"></el-
36 option>
37                         </el-select>
38                     </el-form-item>
39                     <el-form-item label="入职日期">
40                         <el-date-picker
41                             v-model="searchForm.entrydate"
42                             type="daterange"
43                             range-separator="至"
44                             start-placeholder="开始日期"
45                             end-placeholder="结束日期">
46                         </el-date-picker>
47                     </el-form-item>
48                 </el-form>
49             </el-main>
50         </el-container>
```

```
41         <el-button type="primary"
@click="onSubmit">查询</el-button>
42     </el-form-item>
43 </el-form>
44 <!-- 表格 -->
45     <el-table :data="tableData">
46         <el-table-column prop="name"      label="姓
名" width="180"></el-table-column>
47         <el-table-column prop="image"      label="图
像" width="180"></el-table-column>
48         <el-table-column prop="gender"      label="性
别" width="140"></el-table-column>
49         <el-table-column prop="job"      label="职
位" width="140"></el-table-column>
50         <el-table-column prop="entrydate" label="入职
日期" width="180"></el-table-column>
51         <el-table-column prop="updatetime" label="最
后操作时间" width="230"></el-table-column>
52         <el-table-column label="操作" >
53             <el-button type="primary" size="mini">编
辑</el-button>
54             <el-button type="danger" size="mini">删除
</el-button>
55         </el-table-column>
56     </el-table>
57
58 </el-main>
59 </el-container>
60 </el-container>
61 </div>
62 </template>
63
64 <script>
65 export default {
66     data() {
67         return {
68             tableData: [
69
70             ],
71             searchForm: {
72                 name: '',
73                 gender: '',
74                 entrydate: []
75             }
```

```

76     }
77   },
78   methods:{
79     onSubmit:function(){
80       console.log(this.searchForm);
81     }
82   }
83 }
84 </script>
85
86 <style>
87
88 </style>
89

```

#### 4.4.3.5.3 分页工具栏

分页条我们之前做过，所以我们直接找到之前的案例，复制即可，代码如下：

其中template模块代码如下：

```

1   <!-- Pagination分页 -->
2   <el-pagination
3     @size-change="handleSizeChange"
4     @current-change="handleCurrentChange"
5     background
6     layout="sizes,prev, pager, next,jumper,total"
7     :total="1000">
8   </el-pagination>

```

同时methods中需要声明2个函数，代码如下：

```

1   handleSizeChange(val) {
2     console.log(`每页 ${val} 条`);
3   },
4   handleCurrentChange(val) {
5     console.log(`当前页： ${val}`);
6   }

```

此时打开浏览器，效果如下图所示：



此时整体代码如下：

```

1  <template>
2    <div>
3      <!-- 设置最外层容器高度为700px,在加上一个很细的边框 -->
4      <el-container style="height: 700px; border: 1px solid
5        #eee">
6        <el-header style="font-size:40px;background-color:
7          rgb(238, 241, 246)">tliaas 智能学习辅助系统</el-header>
8        <el-container>
9          <el-aside width="200px">
10             <el-menu :default-openeds="['1', '3']">
11               <el-submenu index="1">
12                 <template slot="title"><i class="el-
13                   icon-message"></i>系统信息管理</template>
14
15                 <el-menu-item index="1-1">部门管理</el-
16                   menu-item>
17
18                 <el-menu-item index="1-2">员工管理</el-
19                   menu-item>
20
21               </el-submenu>
22             </el-menu>
23           </el-aside>
24           <el-main>
25             <!-- 表单 -->
26             <el-form :inline="true" :model="searchForm"
27               class="demo-form-inline">
28               <el-form-item label="姓名">
29                 <el-input v-model="searchForm.name"
30                   placeholder="姓名"></el-input>
31               </el-form-item>
32               <el-form-item label="性别">
33                 <el-select v-model="searchForm.gender"
34                   placeholder="性别">

```

```

27         <el-option label="男" value="1"></el-
option>
28         <el-option label="女" value="2"></el-
option>
29     </el-select>
30 </el-form-item>
31 <el-form-item label="入职日期">
32     <el-date-picker
33         v-model="searchForm.entrydate"
34         type="daterange"
35         range-separator="至"
36         start-placeholder="开始日期"
37         end-placeholder="结束日期">
38     </el-date-picker>
39 </el-form-item>
40 <el-form-item>
41     <el-button type="primary"
@click="onSubmit">查询</el-button>
42 </el-form-item>
43 </el-form>
44 <!-- 表格 -->
45 <el-table :data="tableData">
46     <el-table-column prop="name" label="姓
名" width="180"></el-table-column>
47     <el-table-column prop="image" label="图
像" width="180"></el-table-column>
48     <el-table-column prop="gender" label="性
别" width="140"></el-table-column>
49     <el-table-column prop="job" label="职
位" width="140"></el-table-column>
50     <el-table-column prop="entrydate" label="入
职日期" width="180"></el-table-column>
51     <el-table-column prop="updatetime"
label="最后操作时间" width="230"></el-table-column>
52     <el-table-column label="操作" >
53         <el-button type="primary" size="mini">
编辑</el-button>
54         <el-button type="danger" size="mini">删
除</el-button>
55     </el-table-column>
56 </el-table>
57
58 <!-- Pagination分页 -->
59 <el-pagination

```

```
60         @size-change="handleSizeChange"
61         @current-change="handleCurrentChange"
62         background
63         layout="sizes,prev, pager,
next,jumper,total"
64         :total="1000">
65     </el-pagination>
66 </el-main>
67 </el-container>
68 </el-container>
69 </div>
70 </template>
71
72 <script>
73 export default {
74   data() {
75     return {
76       tableData: [
77
78     ],
79       searchForm:{
80         name:'',
81         gender:'',
82         entrydate:[]
83       }
84     }
85   },
86   methods:{
87     onSubmit:function(){
88       console.log(this.searchForm);
89     },
90     handleSizeChange(val) {
91       console.log(`每页 ${val} 条`);
92     },
93     handleCurrentChange(val) {
94       console.log(`当前页: ${val}`);
95     }
96   }
97 }
98 </script>
99
100 <style>
101
102 </style>
```



#### 4.4.3.6 异步数据加载

##### 4.4.3.6.1 异步加载数据

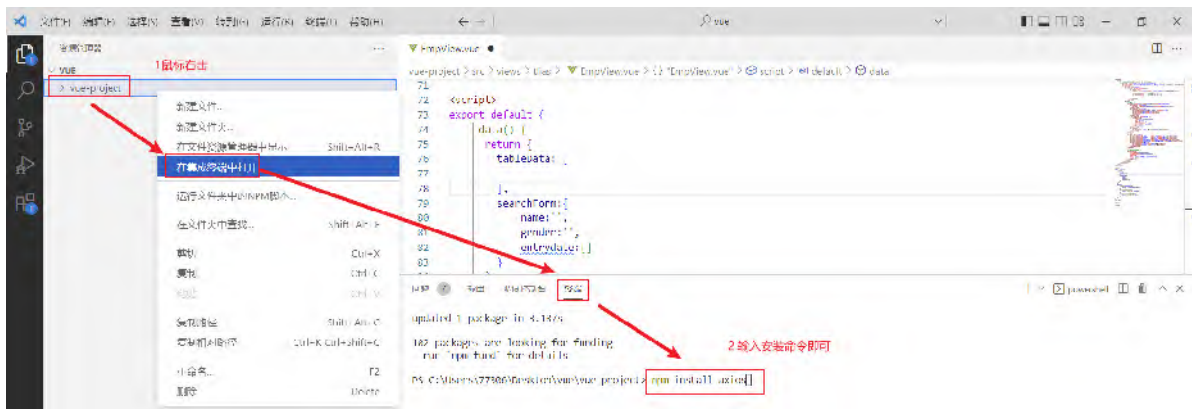
对于案例，我们只差最后的数据了，而数据的mock地址已经提供：<http://yapi.smart-xwork.cn/mock/169327/emp/list>

我们最后要做的就是异步加载数据，所以我们需要使用axios发送ajax请求。

在vue项目中，对于axios的使用，分为如下2步：

1. 安装axios: `npm install axios`
2. 需要使用axios时，导入axios: `import axios 'axios'`

接下来我们先来到项目的执行终端，然后输入命令，安装axios，具体操作如下图所示：



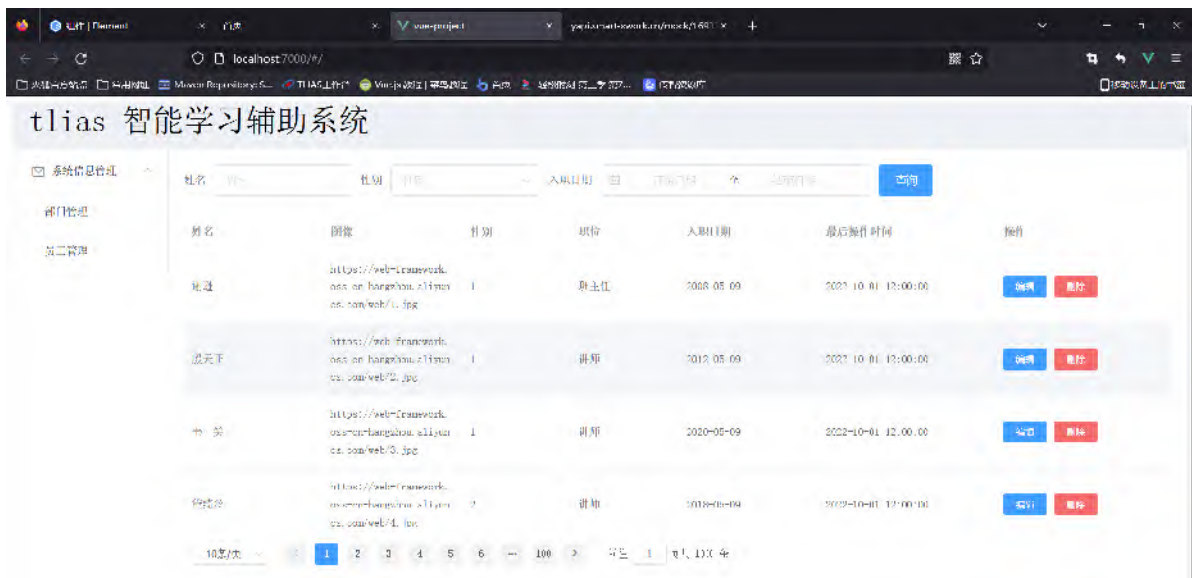
然后重启项目，来到我们的EmpView.vue组件页面，通过import命令导入axios，代码如下：

```
1 import axios 'axios';
```

那么我们什么时候发送axios请求呢？页面加载完成，自动加载，所以可以使用之前的mounted钩子函数，并且我们需要将得到的员工数据要展示到表格，所以数据需要赋值给数据模型tableData，所以我们编写如下代码：

```
1 mounted() {
2     axios.get("http://yapi.smart-xwork.cn/mock/169327/emp/list")
3     .then(resp=>{
4         this.tableData=resp.data.data; //响应数据赋值给数据模型
5     });
6 }
```

此时浏览器打开，呈现如下效果：



但是很明显，性别和图片的内容显示不正确，所以我们需要修复。

#### 4.4.3.6.2 性别内容展示修复

首先我们来到ElementUI提供的表格组件，找到如下示例：



我们仔细对比效果和功能实现代码，发现其中涉及2个非常重要的点：

- `<template>` ： 用于自定义列的内容
  - `slot-scope`： 通过属性的`row`获取当前行的数据

所以接下来，我们可以通过上述的标签自定义列的内容即可，修改性别列的内容代码如下：

```
1 <el-table-column prop="gender" label="性别" width="140">
2   <template slot-scope="scope">
3     {{scope.row.gender=="男"?"男":"女"}}
4   </template>
5 </el-table-column>
```

此时打开浏览器，效果如下图所示：性别一列的值修复成功

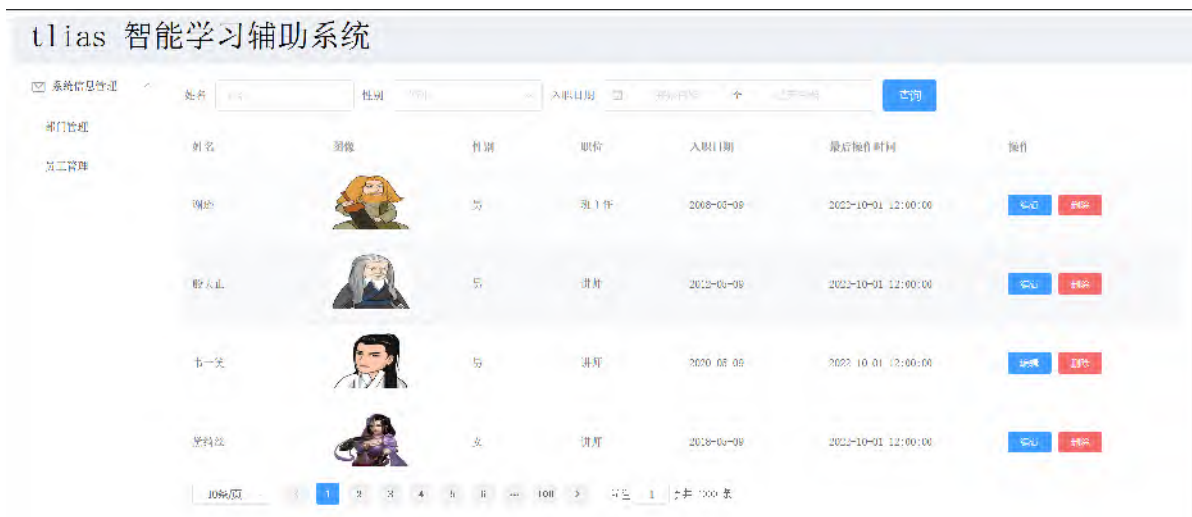


#### 4.4.3.6.3 图片内容展示修复

图片内容的修复和上述一致，需要借助<template>标签自定义列的内容，需要需要展示图片，直接借助<img>标签即可，并且需要设置图片的宽度和高度，所以直接修改图片列的代码如下：

```
1 <el-table-column prop="image" label="图像" width="180">
2   <template slot-scope="scope">
3     
4   </template>
5 </el-table-column>
```

此时回到浏览器，效果如下图所示：图片展示修复成功



此时整个案例完整，其完整代码如下：

```
1 <template>
2   <div>
3     <!-- 设置最外层容器高度为700px,在加上一个很细的边框 -->
4     <el-container style="height: 700px; border: 1px solid #eee">
```

```
5         <el-header style="font-size:40px;background-color:
rgb(238, 241, 246)">智能学习辅助系统</el-header>
6         <el-container>
7             <el-aside width="230px" style="border: 1px solid
#eee">
8                 <el-menu :default-openeds="['1', '3']">
9                     <el-submenu index="1">
10                         <template slot="title"><i class="el-
icon-message"></i>系统信息管理</template>
11
12                         <el-menu-item index="1-1">部门管理</el-
menu-item>
13
14                         <el-menu-item index="1-2">员工管理</el-
menu-item>
15
16                     </el-submenu>
17                 </el-menu>
18             </el-aside>
19             <el-main>
20                 <!-- 表单 -->
21                 <el-form :inline="true" :model="searchForm"
class="demo-form-inline">
22                     <el-form-item label="姓名">
23                         <el-input v-model="searchForm.name"
placeholder="姓名"></el-input>
24                     </el-form-item>
25                     <el-form-item label="性别">
26                         <el-select v-model="searchForm.gender"
placeholder="性别">
27                             <el-option label="男" value="1"></el-
option>
28                             <el-option label="女" value="2"></el-
option>
29                         </el-select>
30                     </el-form-item>
31                     <el-form-item label="入职日期">
32                         <el-date-picker
33                             v-model="searchForm.entrydate"
34                             type="daterange"
35                             range-separator="至"
36                             start-placeholder="开始日期"
37                             end-placeholder="结束日期">
38                         </el-date-picker>
```

```

39         </el-form-item>
40         <el-form-item>
41             <el-button type="primary"
@click="onSubmit">查询</el-button>
42         </el-form-item>
43     </el-form>
44     <!-- 表格 -->
45     <el-table :data="tableData">
46         <el-table-column prop="name"      label="姓
名" width="180"></el-table-column>
47         <el-table-column prop="image"      label="图
像" width="180">
48             <template slot-scope="scope">
49                 
50             </template>
51         </el-table-column>
52         <el-table-column prop="gender"      label="性
别" width="140">
53             <template slot-scope="scope">
54                 {{scope.row.gender==1?"男":"女"}}
55             </template>
56         </el-table-column>
57         <el-table-column prop="job"          label="职
位" width="140"></el-table-column>
58         <el-table-column prop="entrydate" label="入
职日期" width="180"></el-table-column>
59         <el-table-column prop="updatetime"
label="最后操作时间" width="230"></el-table-column>
60         <el-table-column label="操作" >
61             <el-button type="primary" size="mini">
编辑</el-button>
62             <el-button type="danger" size="mini">删
除</el-button>
63         </el-table-column>
64     </el-table>
65
66     <!-- Pagination分页 -->
67     <el-pagination
68         @size-change="handleSizeChange"
69         @current-change="handleCurrentChange"
70         background
71         layout="sizes,prev, pager,
next,jumper,total"

```

```
72         :total="1000">
73         </el-pagination>
74     </el-main>
75 </el-container>
76 </el-container>
77 </div>
78 </template>
79
80 <script>
81 import axios  'axios'
82 export default {
83     data() {
84         return {
85             tableData: [
86
87             ],
88             searchForm:{
89                 name:'',
90                 gender:'',
91                 entrydate:[]
92             }
93         }
94     },
95     methods:{
96         onSubmit:function(){
97             console.log(this.searchForm);
98         },
99         handleSizeChange(val) {
100             console.log(`每页 ${val} 条`);
101         },
102         handleCurrentChange(val) {
103             console.log(`当前页: ${val}`);
104         }
105     },
106     mounted(){
107         axios.get("http://yapi.smart-
108 xwork.cn/mock/169327/emp/list")
109         .then(resp=>{
110             this.tableData=resp.data.data;
111         });
112     }
113 </script>
114
```

```
115 <style>
116
117 </style>
118
```

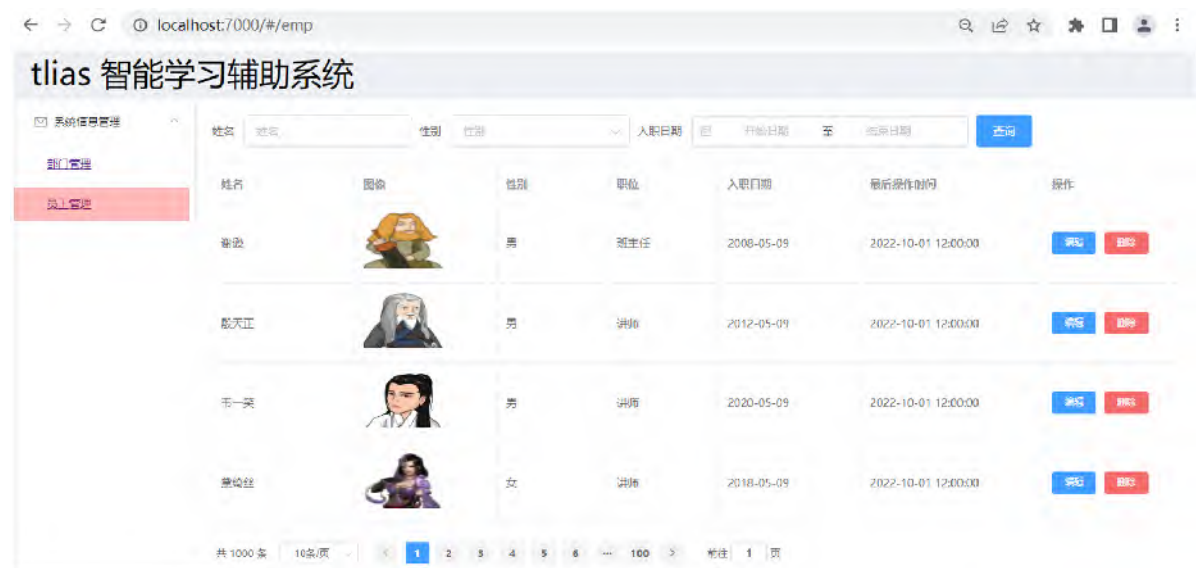
## 5 Vue路由

### 5.1 路由介绍

将代码/vue-project(路由)/vue-project/src/views/tlias/DeptView.vue拷贝到我们当前EmpView.vue同级，其结构如下：



此时我们希望基于4.4案例中的功能，实现点击侧边栏的部门管理，显示部门管理的信息，点击员工管理，显示员工管理的信息，效果如下图所示：

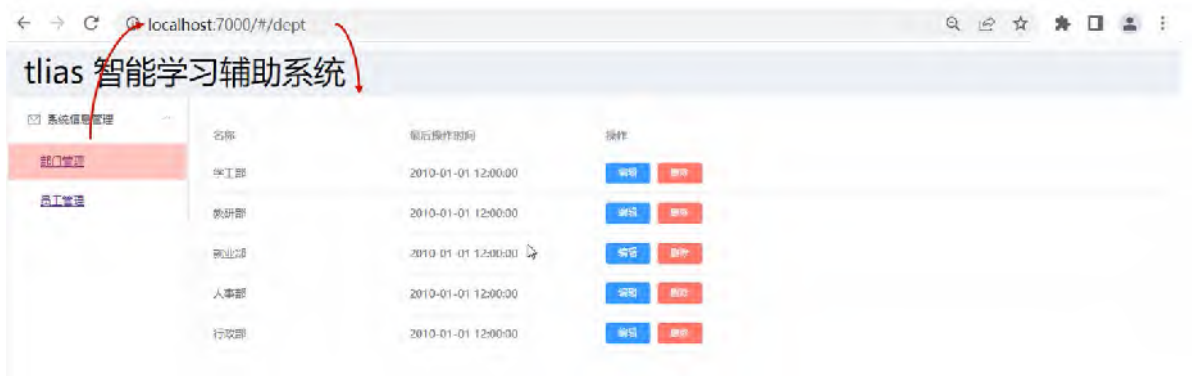






这就需要借助我们的vue的路由功能了。

前端路由：URL中的hash（#号之后的内容）与组件之间的对应关系，如下图所示：

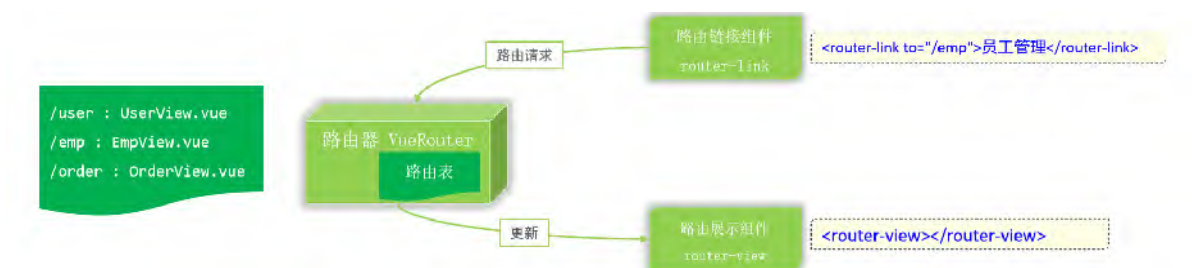


当我们点击左侧导航栏时，浏览器的地址栏会发生变化，路由自动更新显示与url所对应的vue组件。

而我们vue官方提供了路由插件Vue Router,其主要组成如下：

- VueRouter：路由器类，根据路由请求在路由视图中动态渲染选中的组件
- <router-link>：请求链接组件，浏览器会解析成<a>
- <router-view>：动态视图组件，用来渲染展示与路由路径对应的组件

其工作原理如下图所示：



首先VueRouter根据我们配置的url的hash片段和路由的组件关系去维护一张路由表；

然后我们页面提供一个<router-link>组件,用户点击,发出路由请求;

接着我们的VueRouter根据路由请求,在路由表中找到对应的vue组件;

最后VueRouter会切换<router-view>中的组件,从而进行视图的更新

## 5.2 路由入门

接下来我们来演示vue的路由功能。

首先我们需要先安装vue-router插件,可以通过如下命令

```
1 npm install vue-router@3.5.1
```

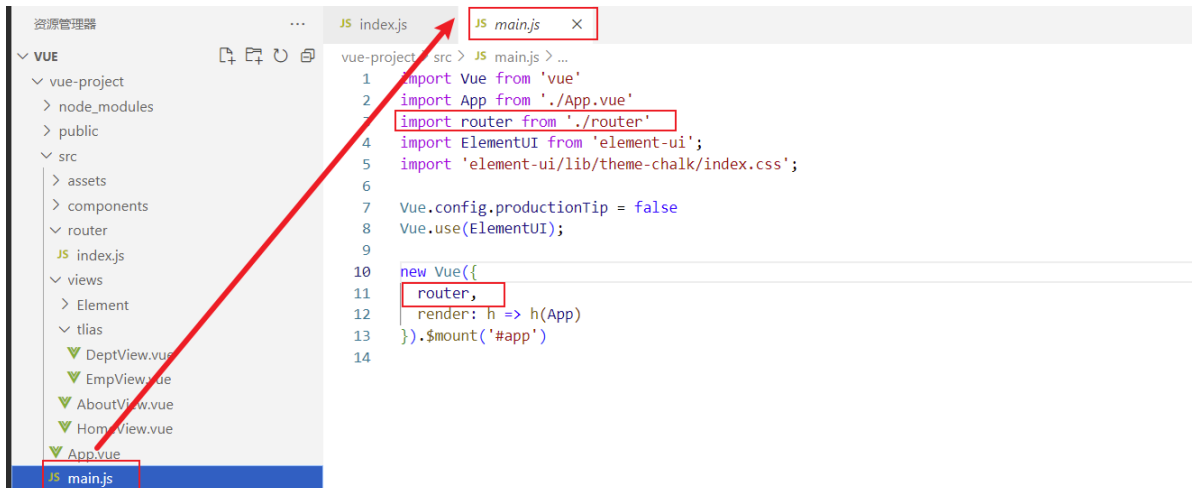
**但是我们不需要安装,因为当初我们再创建项目时,已经勾选了路由功能,已经安装好了。**

然后我们需要在src/router/index.js文件中定义路由表,根据其提供的模板代码进行修改,最终代码如下:

```
1  import Vue  'vue'
2  import VueRouter  'vue-router'
3
4  Vue.use(VueRouter)
5
6  const routes = [
7    {
8      path: '/emp',    //地址hash
9      name: 'emp',
10     component: () => import('../views/tlias/EmpView.vue')    //对应的
                        vue组件
11   },
12   {
13     path: '/dept',
14     name: 'dept',
15     component: () => import('../views/tlias/DeptView.vue')
16   }
17 ]
18
19 const router = new VueRouter({
20   routes
21 })
22
23 export default router
24
```

注意需要去掉没有引用的import模块。

在main.js中，我们已经引入了router功能，如下图所示：



路由基本信息配置好了，路由表已经被加载，此时我们还缺少2个东西，就是<router-link>和<router-view>，所以我们需要修改2个页面（EmpView.vue和DeptView.vue）我们左侧栏的2个按钮为router-link，其代码如下：

```
1 <el-menu-item index="1-1">
2   <router-link to="/dept">部门管理</router-link>
3 </el-menu-item>
4 <el-menu-item index="1-2">
5   <router-link to="/emp">员工管理</router-link>
6 </el-menu-item>
```

然后我们还需要在内容展示区域即App.vue中定义route-view，作为组件的切换，其App.vue的完整代码如下：

```
1 <template>
2   <div id="app">
3     <!-- {{message}} -->
4     <!-- <element-view></element-view> -->
5     <!-- <emp-view></emp-view> -->
6     <router-view></router-view>
7   </div>
8 </template>
9
10 <script>
11 // import EmpView './views/tlias/EmpView.vue'
12 // import ElementView './views/Element/ElementView.vue'
13 export default {
14   components: { },
15   data() {
16     return {
17       "message": "hello world"
```

```

18     }
19   }
20 }
21 </script>
22 <style>
23
24 </style>
25

```

但是我们浏览器打开地址：<http://localhost:7000/>，发现一片空白，因为我们默认的路由路径是/，但是路由配置中没有对应的关系，

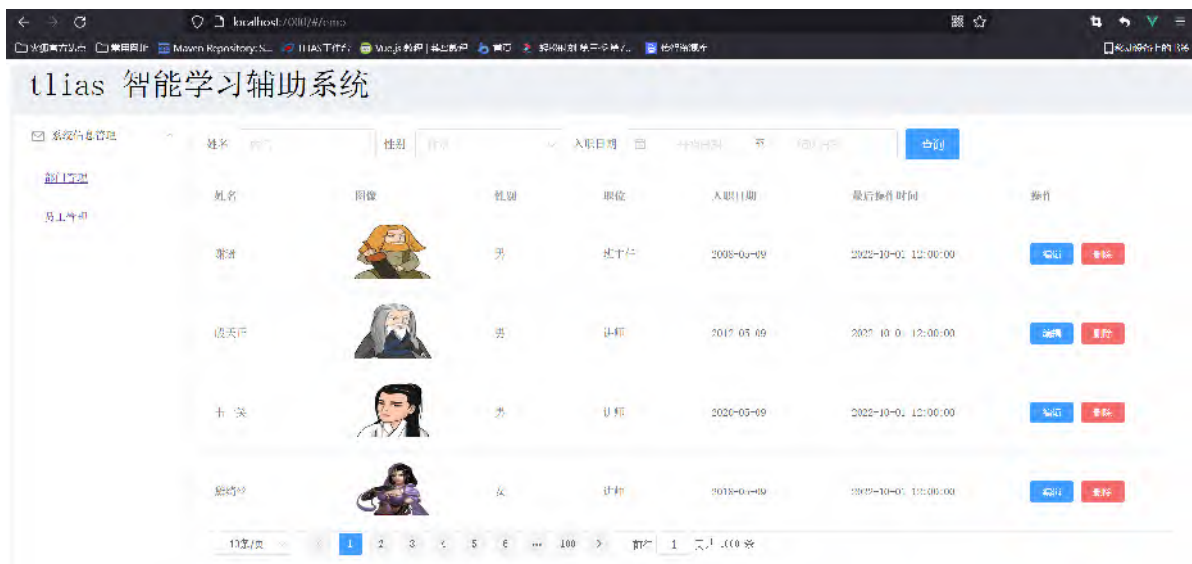
所以我们需要在路由配置中/对应的路由组件，代码如下：

```

1  const routes = [
2    {
3      path: '/emp',
4      name: 'emp',
5      component: () => import('../views/tlias/EmpView.vue')
6    },
7    {
8      path: '/dept',
9      name: 'dept',
10     component: () => import('../views/tlias/DeptView.vue')
11   },
12   {
13     path: '/',
14     redirect: '/emp' //表示重定向到/emp即可
15   },
16 ]

```

此时我们打开浏览器，访问<http://localhost:7000> 发现直接访问的是emp的页面，并且能够进行切换了，其具体如下图所示：



到此我们的路由实现成功。

## 6 打包部署

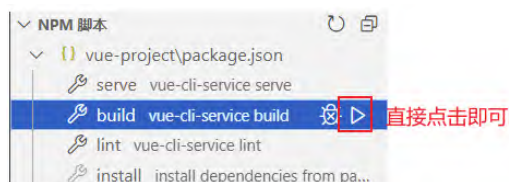
我们的前端工程开发好了，但是我们需要发布，那么如何发布呢？主要分为2步：

1. 前端工程打包
2. 通过nginx服务器发布前端工程

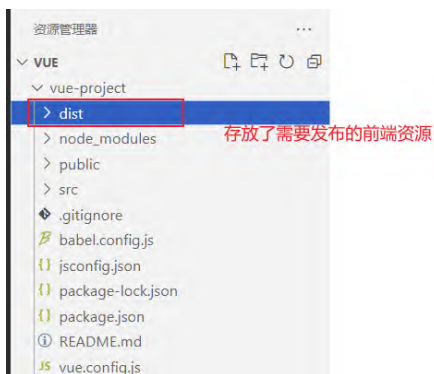
### 6.1 前端工程打包

接下来我们先来对前端工程进行打包

我们直接通过VS Code的NPM脚本中提供的build按钮来完整，如下图所示，直接点击即可：



然后会在工程目录下生成一个dist目录，用于存放需要发布的前端资源，如下图所示：

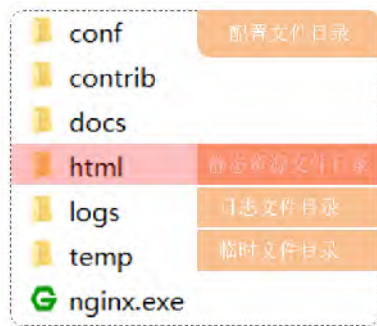


### 6.2 部署前端工程

#### 6.2.1 nginx介绍

nginx：Nginx是一款轻量级的Web服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器。其特点是占有内存少，并发能力强，在各大型互联网公司都有非常广泛的使用。

nginx在windows中的安装是比较方便的，直接解压即可。所以我们直接将资料中的nginx-1.22.0.zip压缩文件拷贝到**无中文的目录下**，直接解压即可，如下图所示就是nginx的解压目录以及目录结构说明：



很明显，我们如果要发布，直接将资源放入到html目录中。

## 6.2.2 部署

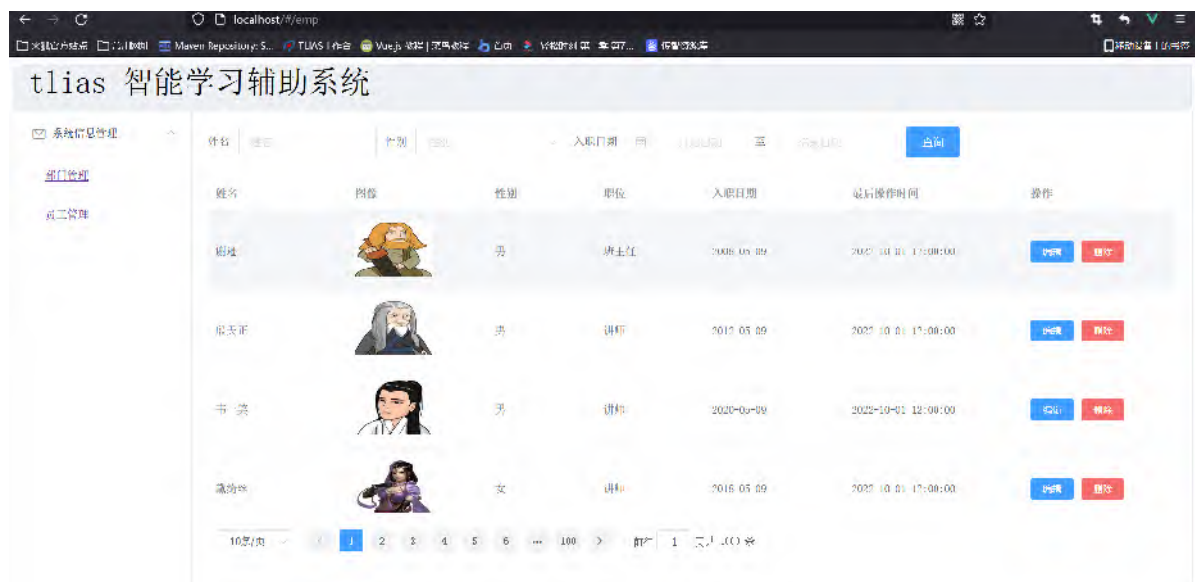
将我们之前打包的前端工程dist目录下得内容拷贝到nginx的html目录下，如下图所示：



然后通过双击nginx下得nginx.exe文件来启动nginx，如下图所示：



nginx服务器的端口号是80，所以启动成功之后，我们浏览器直接访问<http://localhost:80> 即可，其中80端口可以省略，其浏览器展示效果如图所示：



到此，我们的前端工程发布成功。

PS：如果80端口被占用，我们需要通过`conf/nginx.conf`配置文件来修改端口号。如下图所示：

```
35 server {
36     listen      80;
37     server_name localhost;
38
39     #charset koi8-r;
40
41     #access_log logs/host.access.log main;
42
43     location / {
44         root    html;
45         index  index.html index.htm;
46     }
47
48     #error_page 404          /404.html;
49 }
```

此处可以更换端口号