**SpringBoot+vue前后端分离项目实战-前端部分**

**第01讲 课程介绍与项目演示**

**1.1、技术要点**

**1.1.1前端知识：**

vue、element-ui、css3、html5

**1.1.2 后端知识:**

Spring Boot2.x 、MyBatis Plus

**1.1.3 数据库**

mysql 5.5

**1.2、学习收货**

1.2.1、掌握Vue、Element开发后台页面的能力，从而深入理解Vue在后台管理系统中的开发流程；

1.2.2、掌握运用SpringBoot开发后台接口的能力

1.2.3、最终学会用Vue Element Spring Boot 从零开始搭建小型前后端分离项目的能力，从而更深入的理解系统中整个数据的流向，从哪里来，到哪里去；

**3、项目演示：**

****

****

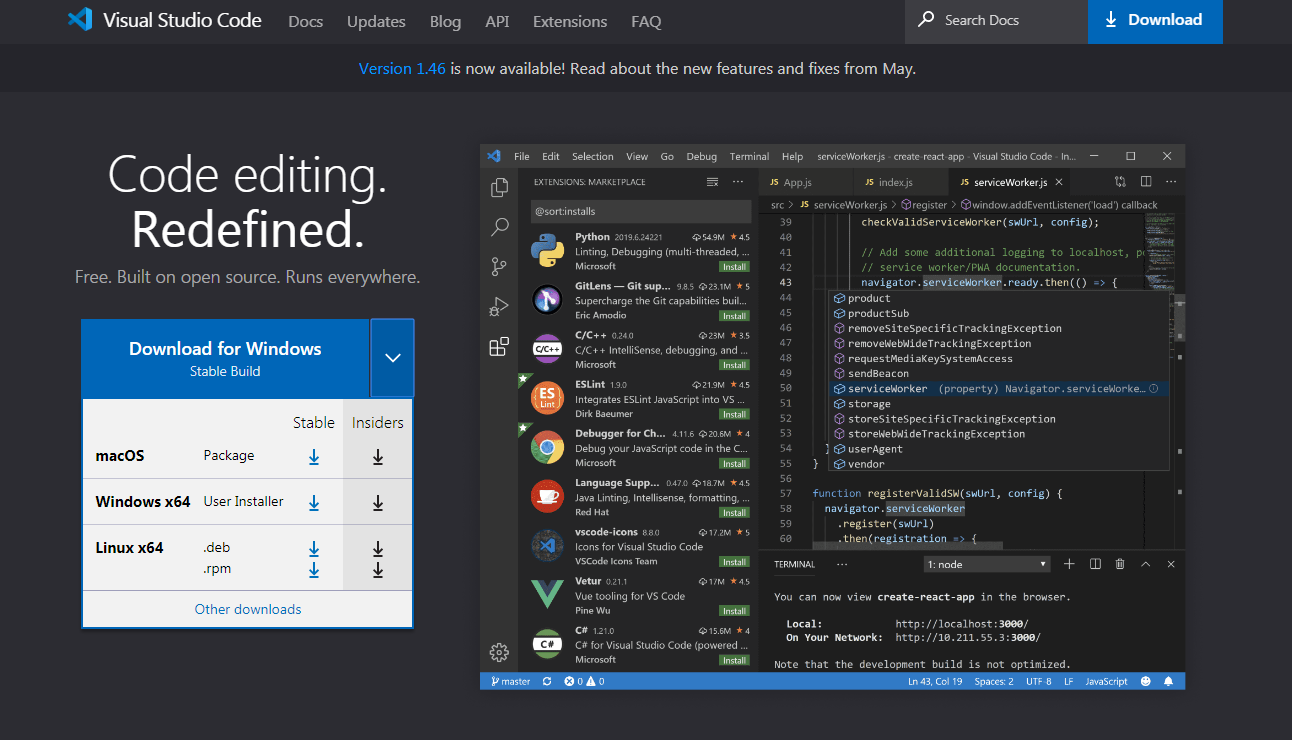
**第02讲 项目工具安装及环境搭建**

**2.1开发工具:Visual studio code**

在visual studio code 官网下载地址

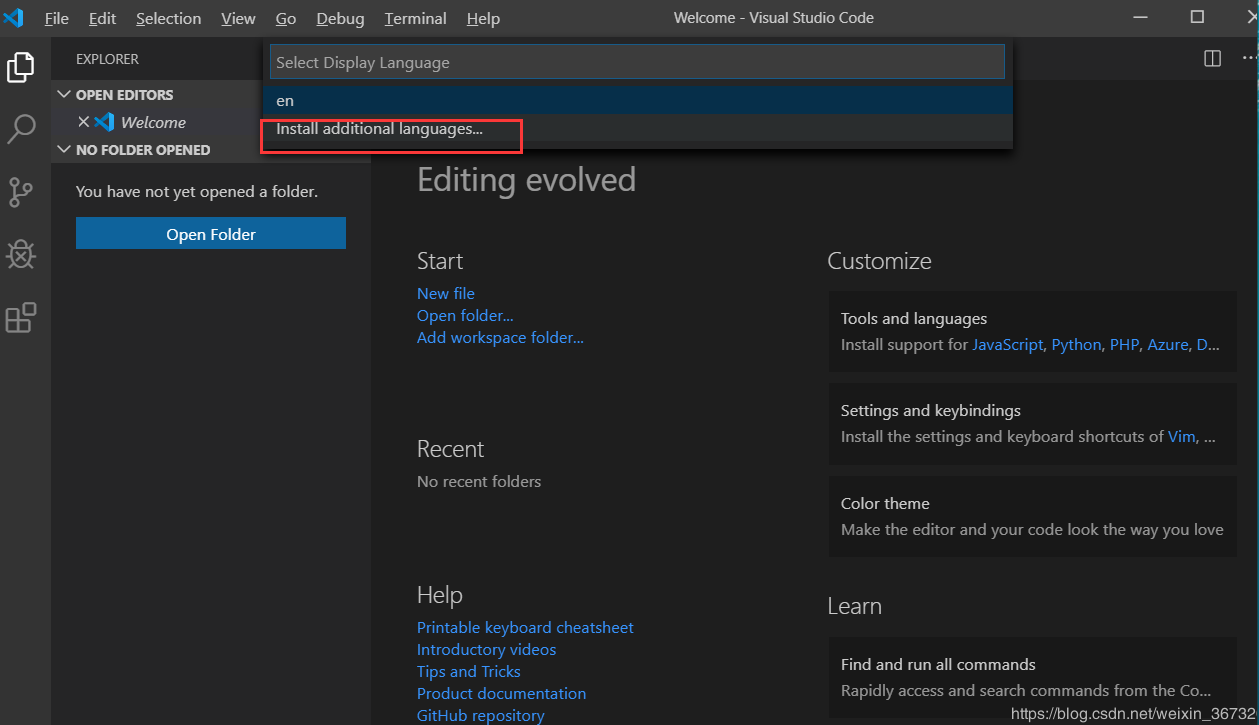
<https://code.visualstudio.com/>

打开官网，如下所示，根据自己电脑的系统，下载对应的vscode版本

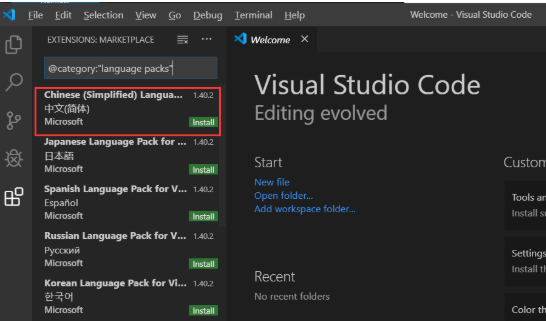


下载完成后，点击安装，一直 next，到最后安装完成即可。

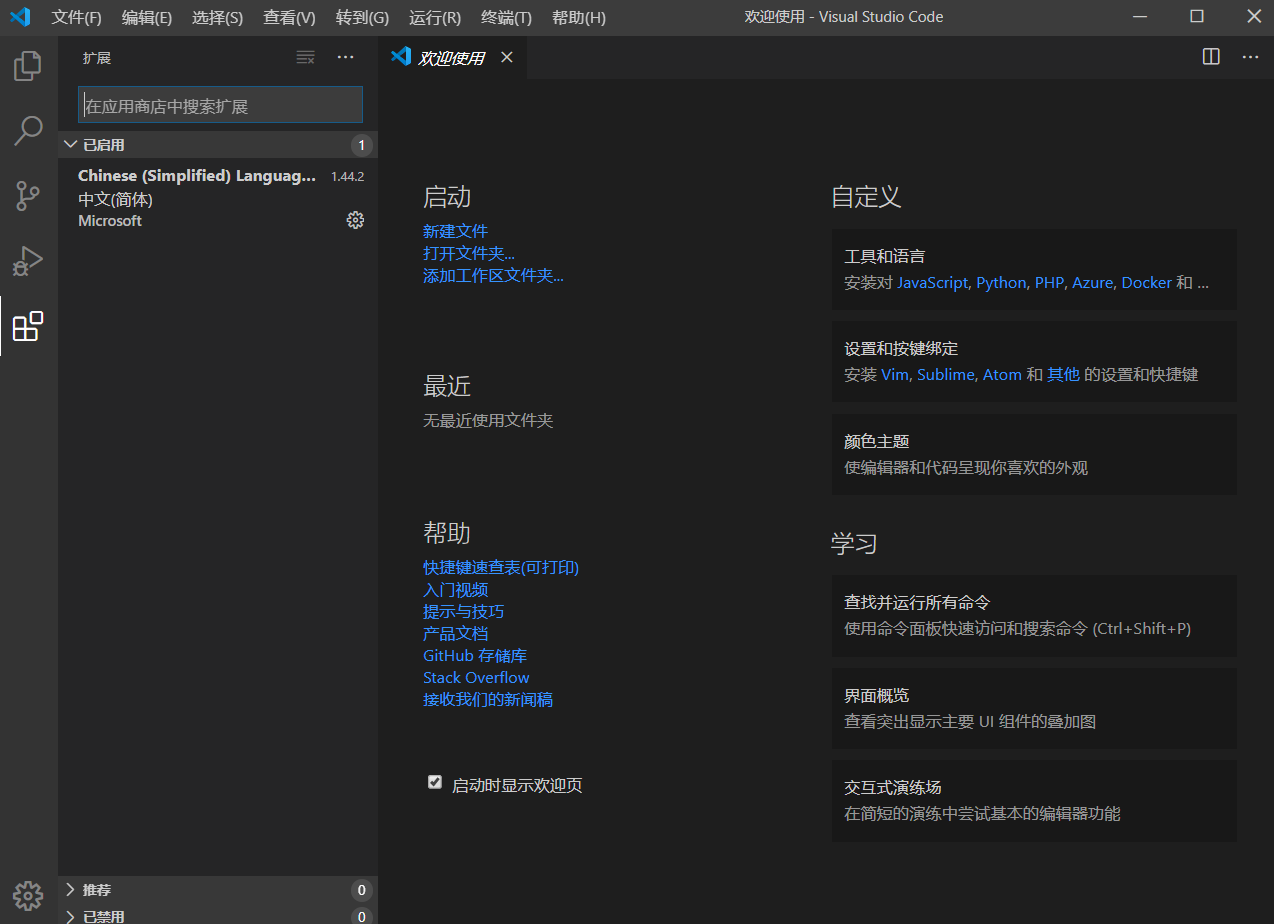
VS code安装汉化插件，**打开vscode ,按"f1",选择"Configure Display Language"，选择第2项“install additional languages...”。**



**点击中文（简体） 右边的“install”**



**安装完成后，关闭软件再重新打开，就已经变成中文了。**

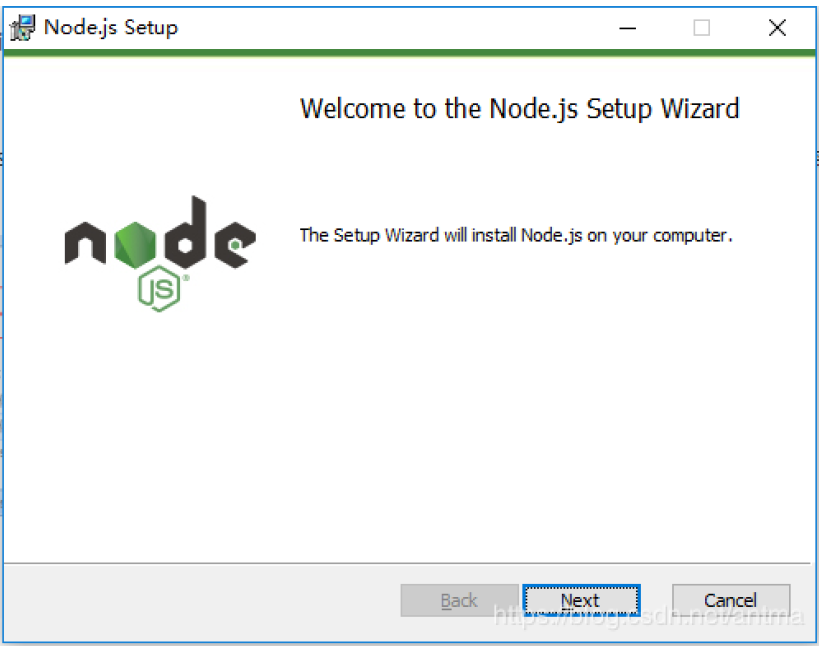
**2.2、依赖环境**

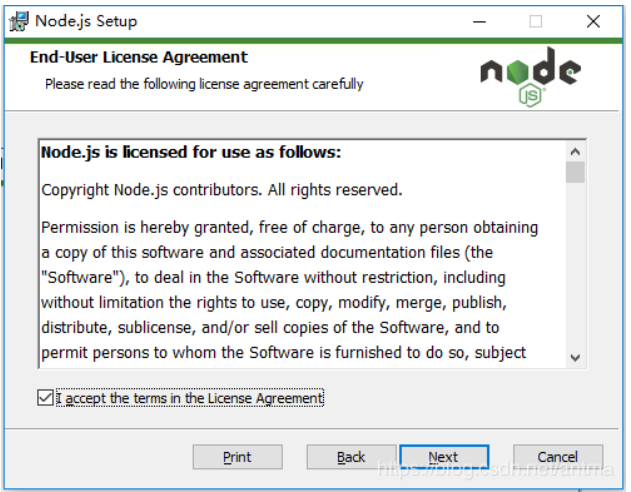
**2.2.1**、Node.js安装

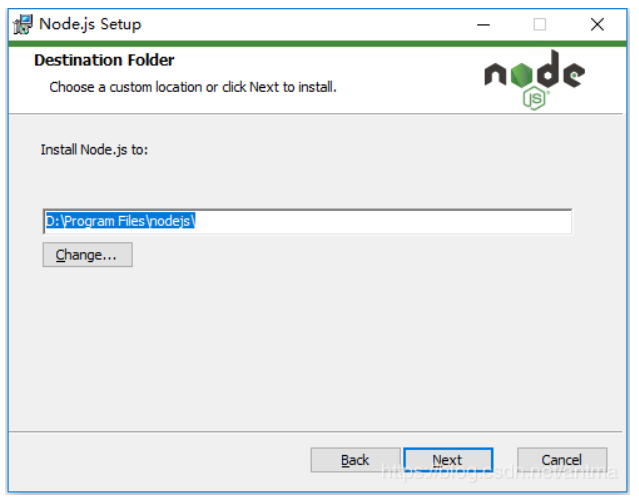
中文官网下载地址 <https://nodejs.org/zh-cn/download/>

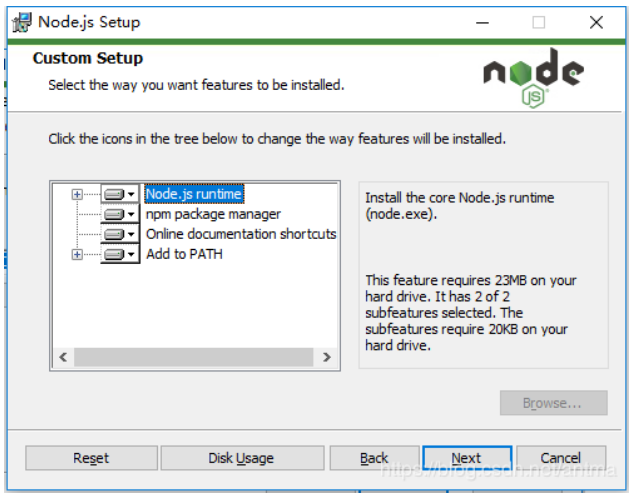
到官网下载对应系统下的node.js安装，可以下载最新版本

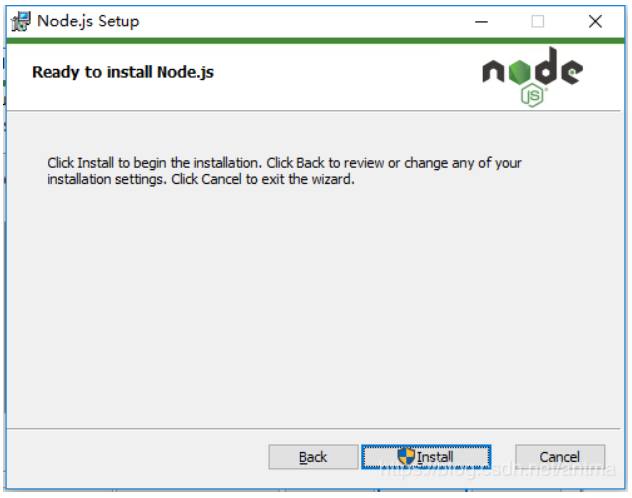
下载后一直next，直到安装完成。

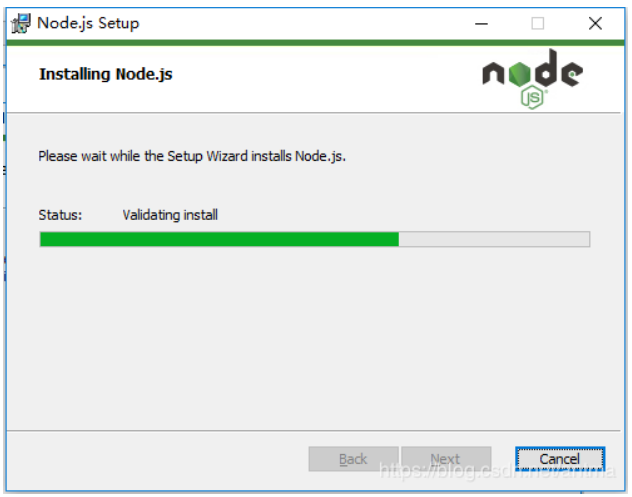


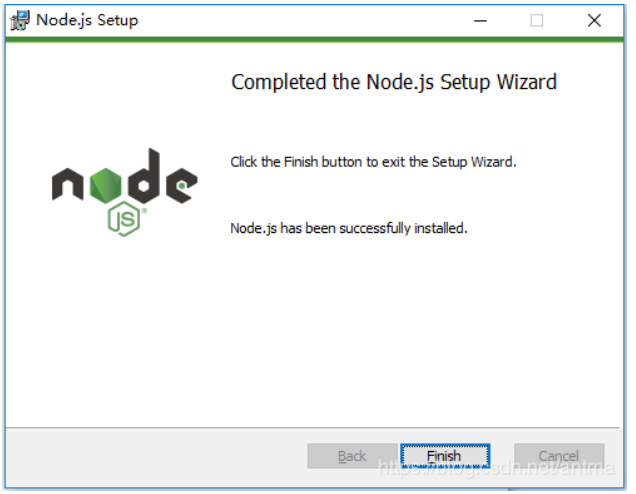
选择node.js的安装目录

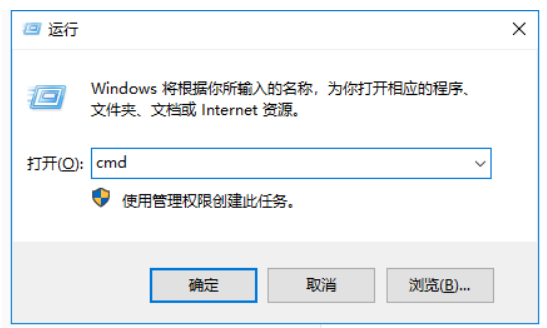


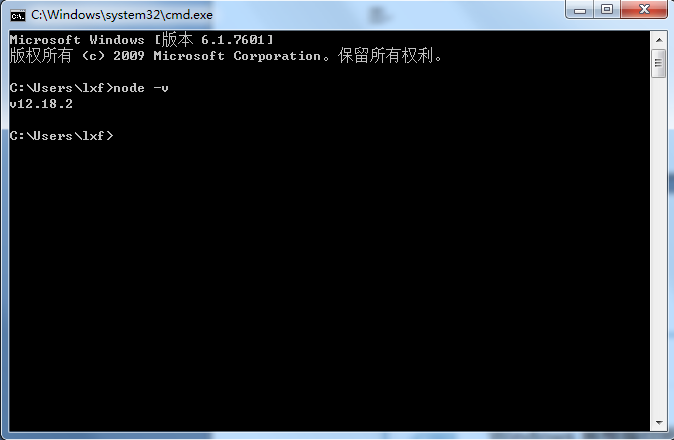






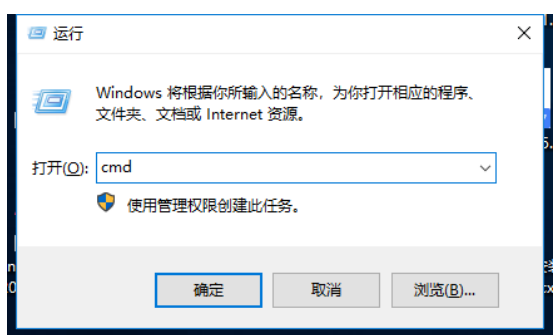
node-v查看安装版本号，win10系统 win+R组合键快速打开运行窗口，输入cmd，如下图

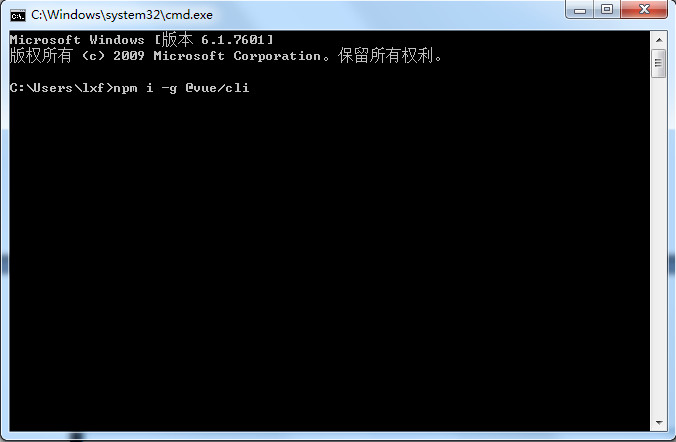




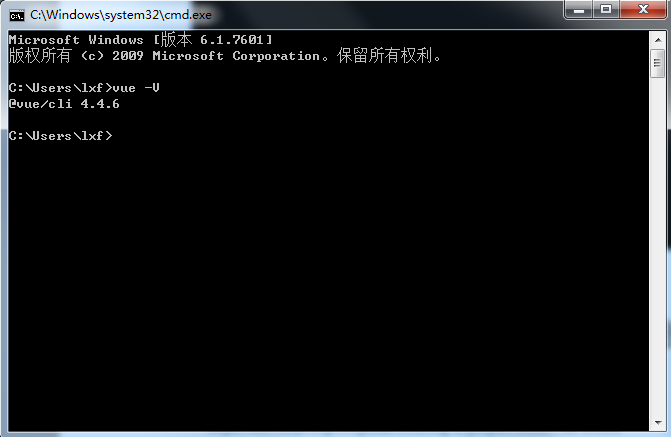
**2.2.2 Vue CLI 脚手架**

安装命令npm i -g @vue/cli，win10左下角右键，运行，输入cmd，进入命令行窗口输入：npm i -g @vue/cli





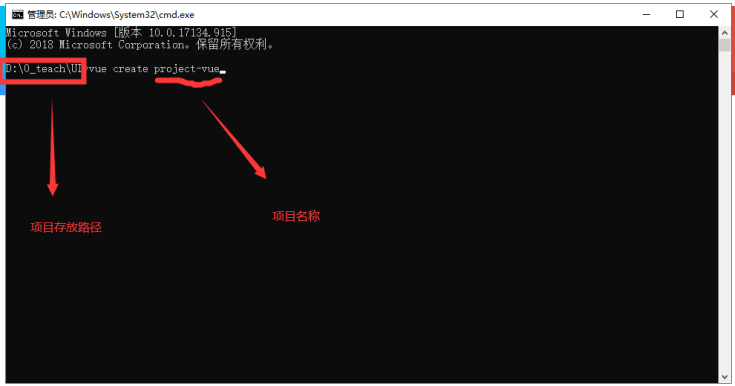
安装中会有提示你选择taobao镜像，可以选择taobao镜像安装，可以加快安装的速度，安装成功后，输入:vue -V 命令可以查看版本。

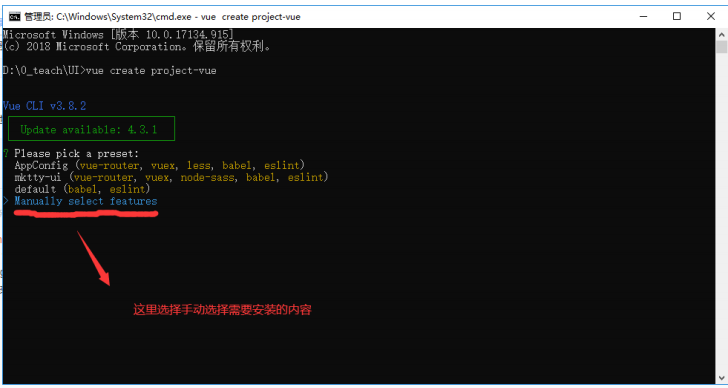


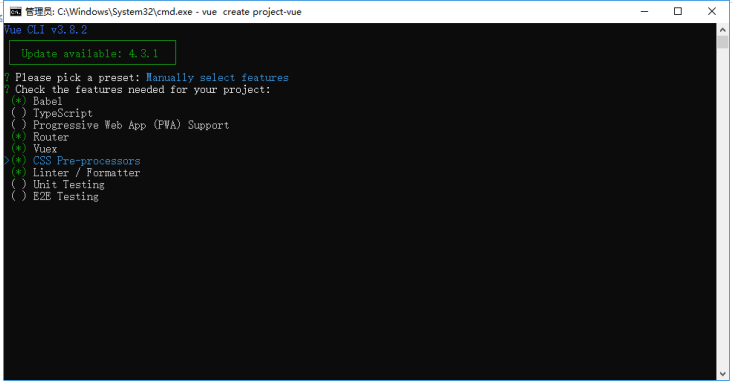
**第03讲 前端Vue项目创建**

vuecli官网：<https://cli.vuejs.org/zh/guide/creating-a-project.html#vue-create>,

在自己电脑本地磁盘新建一个文件夹，用于保存项目 ，文件夹命名最好使用英文名称，如下，D:\0\_teach\UI。cd进入文件夹，vue create 项目名称，输入 vue create project-vue回车，进入到项目保存的目录，输入vue create project-vue回车，

1、回车进入到如下界面，键盘上、下键选择需要的创建方式，我们这里选择 manually select features，手动选择配置的方式，然后回车。

2、回车进入到如下界面，键盘上、下键移动选择需要的项，按 空格 键可以选中。我们选择如下带\* 的项目，然后回车。



**选项说明**

Babel： 将 ES6 编 译 成 ES5

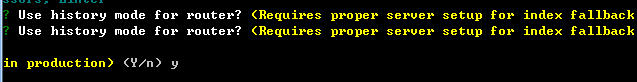
TypeScript：使 用 TypeScript

Router和Vuex`：路由和状态管理

Linter/ Formatter：代码检查工具

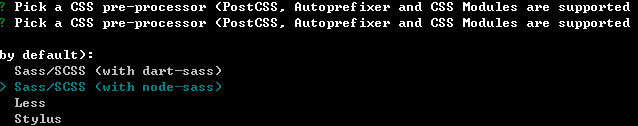
CSS Pre-processors：css预编

**3、**

****

路由使用历史模式? 这种模式充分利用 history.pushState API 来完成 URL 跳转而无须重新加载页面

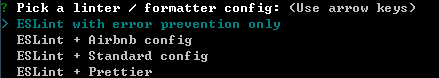
**4、**

****

使用什么css预编译器？

**选 择 Sass/SCSS (with node-sass)**

**5、**

****

**选择ESLint with error prevention only**

eslint w...: 只进行报错提醒；

eslint + A...: 不严谨模式；

eslint + S...`:正常模式；

eslint + P...`:严格模式；

**6、**

****

**Pick additionall intfeatures：代码检查方式, 选择Lint on save保存时检查**

**7、**

****

选择配置信息存放位置：单独存放或者并入package.json ，

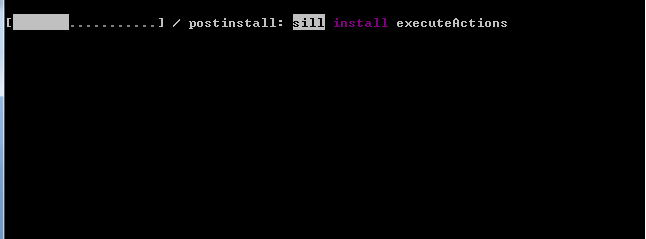
**选择In dedicated configfiles**

**8、**

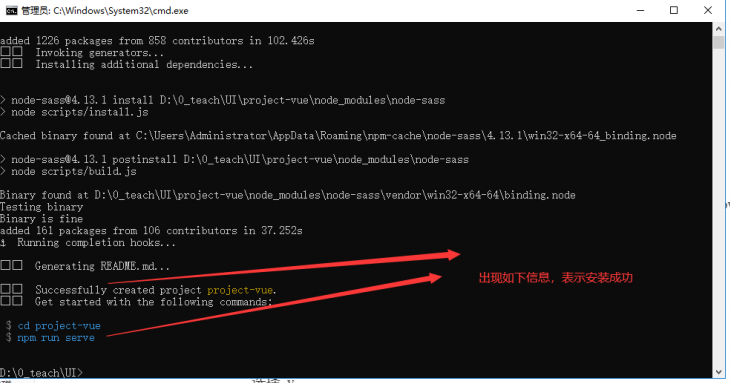
****

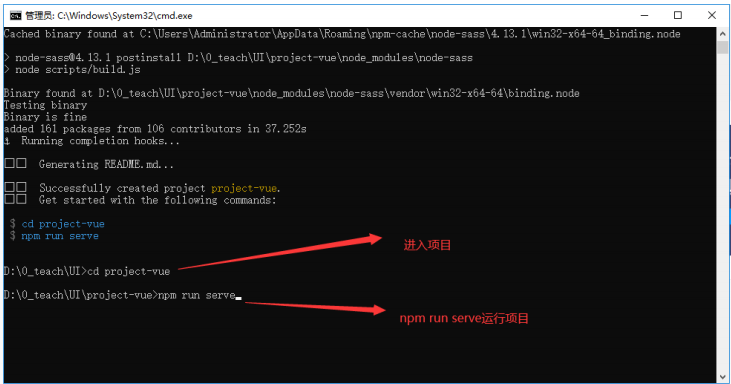
是否保存当前预设，下次构建无需再次配置

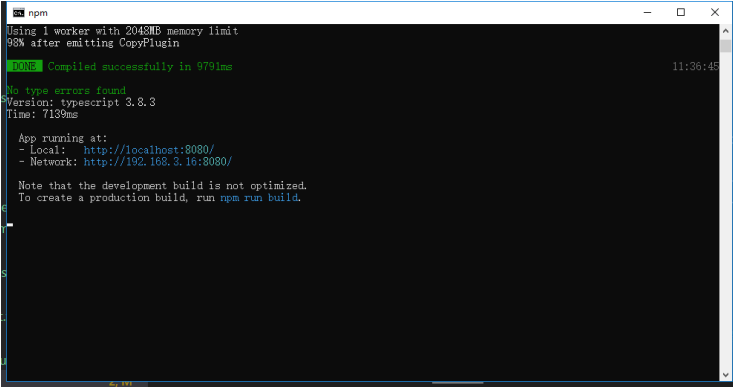
选择 n**，**回车，等待下载依赖

****

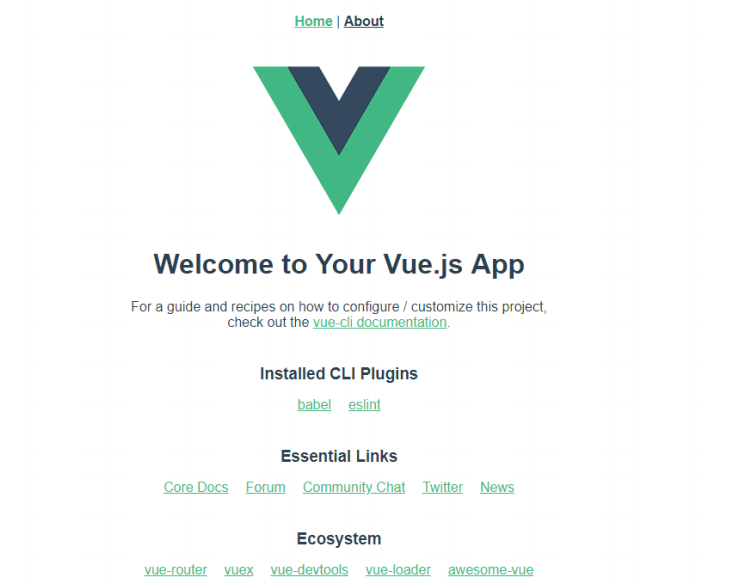
成功后出现如界面：

cd project-vue 进入到项目 ，npmrunserve运行项目





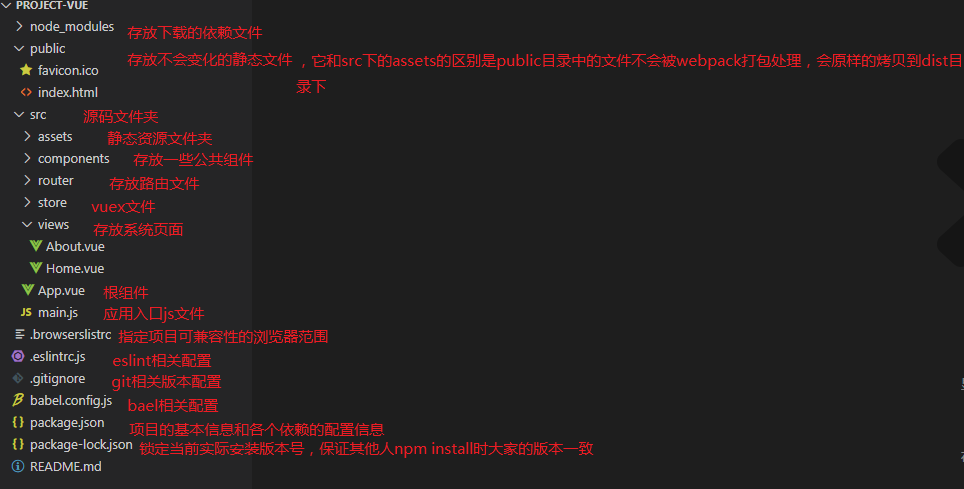
浏览器 http://localhost:8080访问项目，看到如下页面表示项目创建成功



**第04讲 项目目录介绍与Element依赖引入**

**1、Vue项目目录介绍**

一个vue-cli的项目结构如下，其中src文件夹是需要掌握的，至于其他相关文件，了解一下即可。

****

**node\_modules——[依赖包]**

node\_modules里面是项目依赖包，其中包括很多基础依赖，自己也可以根据需要安装其他依赖。安装方法为打开cmd，进入项目目录，输入npm install [依赖包名称],回车。

在两种情况下我们会自己去安装依赖：

（1）项目运行缺少该依赖包：例如项目加载外部css会用到的css-loader，路由跳转vue-loader等（安装方法示例：npm install css-loader）

（2）安装插件：如vux（基于WEUI的移动端组件库），vue-swiper（轮播插件

注：有时会安装指定依赖版本，需在依赖包名称后加上版本号信息，如安装11.1.4版本的vue-loader，输入npm install vue-loader@11.1.4

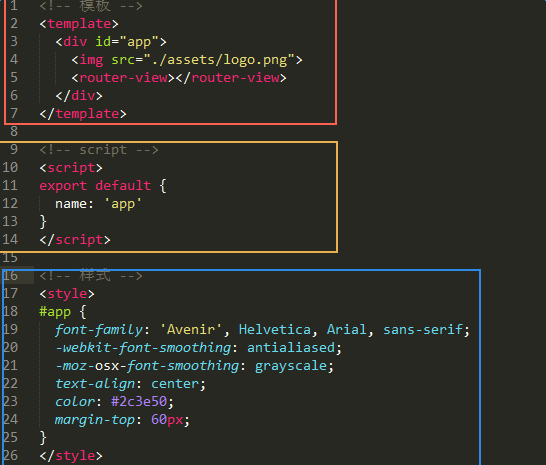
**src——[项目核心文件]**

index.html——[主页]  
index.html如其他html一样，但一般只定义一个空的根节点，在main.js里面定义的实例将挂载在根节点下，内容都通过vue组件来填充



App.vue——[根组件]

一个vue页面通常由三部分组成:模板(template)、js(script)、样式(style)



【template】

其中模板只能包含一个父节点，也就是说顶层的div只能有一个（例如上图，父节点为#app的div，其没有兄弟节点）

是子路由视图，后面的路由页面都显示在此处

打一个比喻吧,类似于一个插槽，跳转某个路由时，该路由下的页面就插在这个插槽中渲染显示

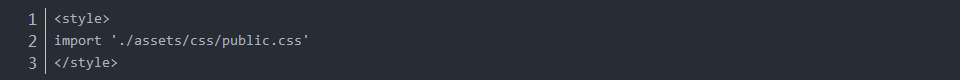
【script】

vue通常用es6来写，用export default导出，其下面可以包含数据data，生命周期(mounted等)，方法(methods)等，具体语法请看vue.js文档，在后面我也会通过例子来说明。

【style】

样式通过style标签

如要引入外部css文件，首先需给项目安装css-loader依赖包，打开cmd，进入项目目录，输入npm install css-loader,回车。安装完成后，就可以在style标签下import所需的css文件，例如：

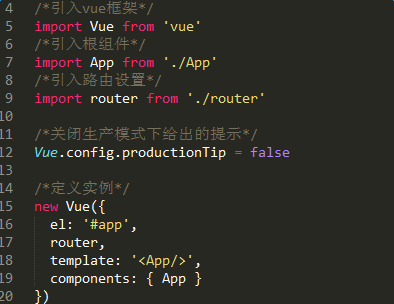


这样，我们就可以把style下的样式封装起来，写到css文件夹，再引入到页面使用，整个vue页面也看上去更简洁。

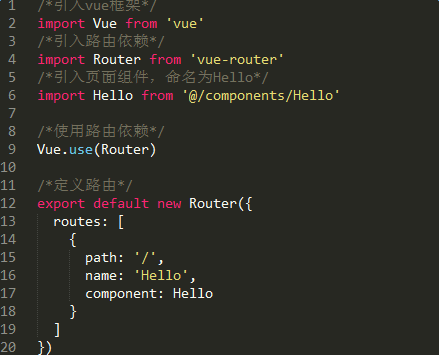
main.js——[入口文件]  
main.js主要是引入vue框架，根组件及路由设置，并且定义vue实例，下图中的

components:{App}就是引入的根组件App.vue

后期还可以引入插件，当然首先得安装插件。



router——[路由配置]  
router文件夹下，有一个index.js，即为路由配置文件



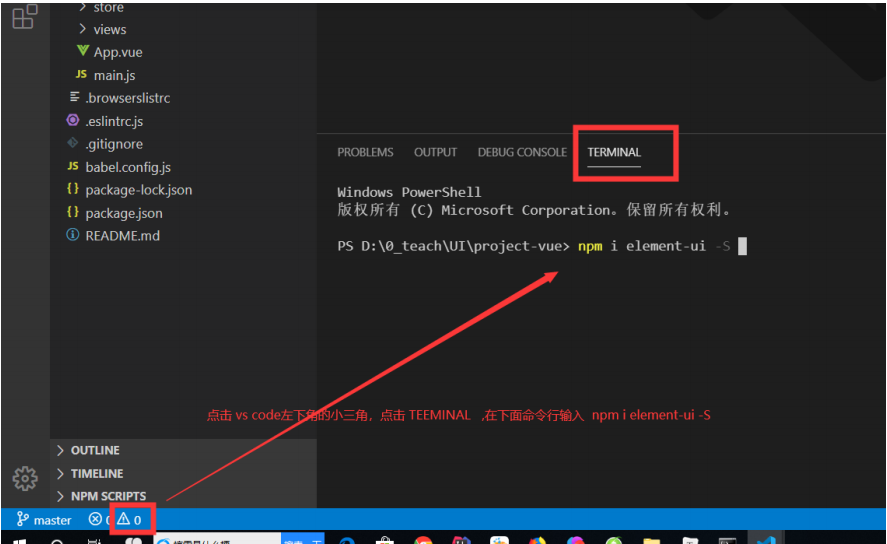
这里定义了路径为’/'的路由，该路由对应的页面是Hello组件，所以当我们在浏览器url访问http://localhost:8080/#/时就渲染的Hello组件

类似的，我们可以设置多个路由，‘/index’,’/list’之类的，当然首先得引入该组件，再为该组件设置路由。

**2**、**element安装**

elementUI官网[https://element.eleme.cn/#/zh-CN/component/installation](https://element.eleme.cn/%23/zh-CN/component/installation)

安装element npm i element-ui -S

element引入 [https://element.eleme.cn/#/zh-CN/component/quickstart](https://element.eleme.cn/%23/zh-CN/component/quickstart)

**在项目中main.js中引入**

import ElementUI from 'element-ui'; //引入element组件库

import 'element-ui/lib/theme-chalk/index.css'; //引入element样式文件

Vue.use(ElementUI); //使用element

**测试element是否引入成功：**

在About.vue中添加如下代码：

<template>

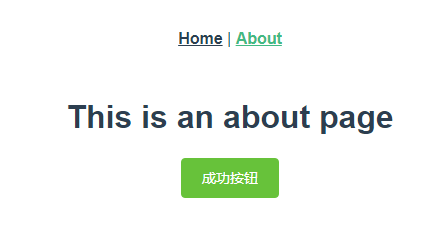
<div class="about">

<el-button type="success">成功按钮</el-button>

</div>

</template>

启动项目如果出现如下效果，则说明element引入成功。



**3、cs code插件安装**

Vetur 这个插件是 vscode 能优雅写 Vue 的核心，代码高亮，语法检查等

Vue VSCode Snippets 代码补全

Element UI VSCode Snippets 、vscode-element-helper、vue-element-snippet、vue-element-sinppets、element UI Snippets等 element的代码提示

**4、解决css属性在html标签里面不提示问题**

设置 ->搜索prevent->把Snippets Prevent QuickSuggestions勾掉即可

如果是vue页面，需要把vscode右下角的页面模式选择为 html即可

**第05讲 Css3弹性盒子基础讲解**

**1、盒子模型特点：**

1)div默认是从上到下排列的

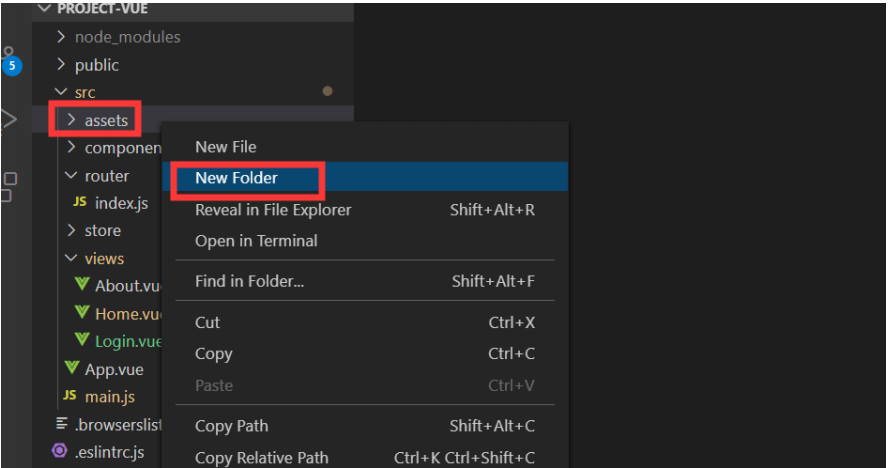
2)当把一个div变成一个盒子模型的时候，div子元素将横向排列

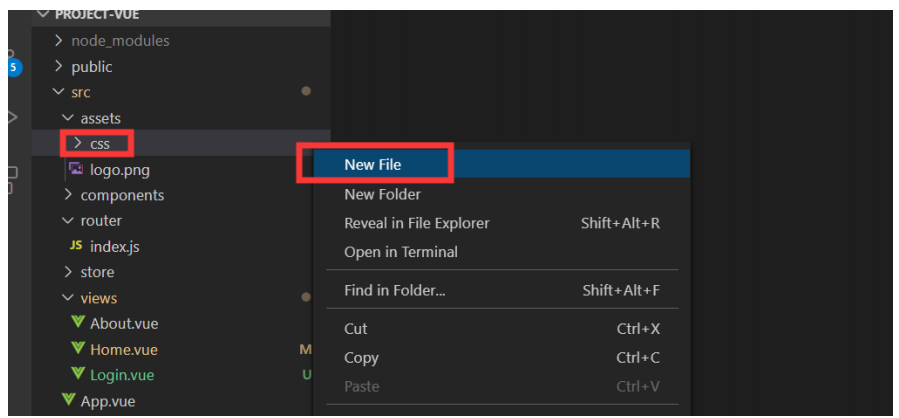
3)盒子模型默认存在两个轴，主轴(x轴，水平方向)和交叉轴(y轴，垂直方向)

<https://www.runoob.com/cssref/css3-pr-align-content.html>

**2、在assets目录下新建css目录**

1)、找到项目assets目录，右键->NewFolder->输入css,按回车

****2）新建flex.css，找到上面新建的css目录，右键 -> New File 录入flex.css，按回车。

****

**3）录入如下CSS**

/\*!css公共样式 \*/

.ub{

    display: -webkit-box;

    display: -webkit-flex;

    display: -ms-flexbox;

    display: flex

}

.row-left {

    -webkit-box-orient: horizontal;

    -webkit-box-direction: normal;

    -webkit-flex-direction: row;

    -ms-flex-direction: row;

    flex-direction: row

}

.row-right {

    -webkit-box-orient: horizontal;

    -webkit-box-direction: reverse;

    -webkit-flex-direction: row-reverse;

    -ms-flex-direction: row-reverse;

    flex-direction: row-reverse;

    -webkit-box-pack: end

}

.column-top {

    -webkit-box-orient: vertical;

    -webkit-box-direction: normal;

    -webkit-flex-direction: column;

    -ms-flex-direction: column;

    flex-direction: column

}

.column-bottom {

    -webkit-box-orient: vertical;

    -webkit-box-direction: reverse;

    -webkit-flex-direction: column-reverse;

    -ms-flex-direction: column-reverse;

    flex-direction: column-reverse;

    -webkit-box-pack: end

}

.main-left {

    -webkit-box-pack: start;

    -webkit-justify-content: flex-start;

    -ms-flex-pack: start;

    justify-content: flex-start

}

.main-right {

    -webkit-box-pack: end;

    -webkit-justify-content: flex-end;

    -ms-flex-pack: end;

    justify-content: flex-end

}

.main-justify {

    -webkit-box-pack: justify;

    -webkit-justify-content: space-between;

    -ms-flex-pack: justify;

    justify-content: space-between

}

.main-center {

    -webkit-box-pack: center;

    -webkit-justify-content: center;

    -ms-flex-pack: center;

    justify-content: center

}

.cross-top {

    -webkit-box-align: start;

    -webkit-align-items: flex-start;

    -ms-flex-align: start;

    align-items: flex-start

}

.cross-bottom {

    -webkit-box-align: end;

    -webkit-align-items: flex-end;

    -ms-flex-align: end;

    align-items: flex-end

}

.cross-center {

    -webkit-box-align: center;

    -webkit-align-items: center;

    -ms-flex-align: center;

    align-items: center

}

.ub-f0 {

    -webkit-box-flex: 0;

    -webkit-flex-grow: 0;

    -ms-flex-positive: 0;

    flex-grow: 0;

    -webkit-flex-shrink: 0;

    -ms-flex-negative: 0;

    flex-shrink: 0

}

.ub-f1 {

    -webkit-box-flex: 1;

    -webkit-flex-grow: 1;

    -ms-flex-positive: 1;

    flex-grow: 1;

    -webkit-flex-shrink: 1;

    -ms-flex-negative: 1;

    flex-shrink: 1

}

.ub-f2 {

    -webkit-box-flex: 2;

    -webkit-flex-grow: 2;

    -ms-flex-positive: 2;

    flex-grow: 2;

    -webkit-flex-shrink: 2;

    -ms-flex-negative: 2;

    flex-shrink: 2

}

.ub-f3 {

    -webkit-box-flex: 3;

    -webkit-flex-grow: 3;

    -ms-flex-positive: 3;

    flex-grow: 3;

    -webkit-flex-shrink: 3;

    -ms-flex-negative: 3;

    flex-shrink: 3

}

.ub-f4 {

    -webkit-box-flex: 4;

    -webkit-flex-grow: 4;

    -ms-flex-positive: 4;

    flex-grow: 4;

    -webkit-flex-shrink: 4;

    -ms-flex-negative: 4;

    flex-shrink: 4

}

.ub-f5 {

    -webkit-box-flex: 5;

    -webkit-flex-grow: 5;

    -ms-flex-positive: 5;

    flex-grow: 5;

    -webkit-flex-shrink: 5;

    -ms-flex-negative: 5;

    flex-shrink: 5

}

.ub-f6 {

    -webkit-box-flex: 6;

    -webkit-flex-grow: 6;

    -ms-flex-positive: 6;

    flex-grow: 6;

    -webkit-flex-shrink: 6;

    -ms-flex-negative: 6;

    flex-shrink: 6

}

.ub-f7 {

    -webkit-box-flex: 7;

    -webkit-flex-grow: 7;

    -ms-flex-positive: 7;

    flex-grow: 7;

    -webkit-flex-shrink: 7;

    -ms-flex-negative: 7;

    flex-shrink: 7

}

.ub-f8 {

    -webkit-box-flex: 8;

    -webkit-flex-grow: 8;

    -ms-flex-positive: 8;

    flex-grow: 8;

    -webkit-flex-shrink: 8;

    -ms-flex-negative: 8;

    flex-shrink: 8

}

.ub-f9 {

    -webkit-box-flex: 9;

    -webkit-flex-grow: 9;

    -ms-flex-positive: 9;

    flex-grow: 9;

    -webkit-flex-shrink: 9;

    -ms-flex-negative: 9;

    flex-shrink: 9

}

.ub-f10 {

    -webkit-box-flex: 10;

    -webkit-flex-grow: 10;

    -ms-flex-positive: 10;

    flex-grow: 10;

    -webkit-flex-shrink: 10;

    -ms-flex-negative: 10;

    flex-shrink: 10

}

4）在项目main.js中引入flex.css

//引入弹性盒子布局样式

import './assets/css/flex.css'

**3、弹性盒子模型常用属性讲解**

<https://www.cnblogs.com/qcloud1001/p/9848619.html>

**a、flex-direction决定主轴的方向**

row（默认值）：主轴为水平方向，起点在左端。

row-reverse：主轴为水平方向，起点在右端。

column：主轴为垂直方向，起点在上沿。

column-reverse：主轴为垂直方向，起点在下沿。

**b、justify-content主轴上的对齐方式**

flex-start（默认值）：左对齐

flex-end：右对齐

center： 居中

space-between：两端对齐，项目之间的间隔都相等。

space-around：每个项目两侧的间隔相等。所以，项目之间的间隔比项目与边框的间隔大一倍。

**c、align-items属性定义在交叉轴上的对齐方式**

flex-start：交叉轴的起点对齐。

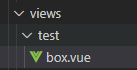
flex-end：交叉轴的终点对齐。

center：交叉轴的中点对齐。

**d、flex-grow属性定义div所占的份数**

**4、示例代码**

在view目录下新建一个test目录，在test目录下新建box.vue

****

在router目录下的index.js中配置路由：

 const routes = [

  {

    path: '/',

    name: 'Home',

    component: Home

  },

  {

    path: '/about',

    name: 'About',

    component: () => import(/\* webpackChunkName: "about" \*/ '../views/About.vue')

  },

  {

    path: '/box',

    name: 'Box',

    component: () => import( '../views/test/box.vue')

  }

]

<template>

  <div>

    <h3>div默认是垂直方向排列的</h3>

    <div style="background-color: #d0d0d0;height: 250px;">

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型默认是水平方向排列</h3>

    <div style="background-color: #d0d0d0;height: 250px;display: flex;">

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型默认是水平方向排列flex-direction: row</h3>

    <div style="background-color: #d0d0d0;height: 250px;display: flex; flex-direction: row;">

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型从右到左排列flex-direction: row-reverse</h3>

    <div

      style="background-color: #d0d0d0;height: 250px;display: flex; flex-direction: row-reverse;"

    >

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型默认是垂直方向排列</h3>

    <div style="background-color: #d0d0d0;height: 250px;display: flex; flex-direction: column;">

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型默认是垂直方向排列反转</h3>

    <div

      style="background-color: #d0d0d0;height: 250px;display: flex; flex-direction: column-reverse;"

    >

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型右对齐方式</h3>

    <div style="background-color: #d0d0d0;height: 250px;display: flex; justify-content: flex-end;">

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型中间对齐方式</h3>

    <div style="background-color: #d0d0d0;height: 250px;display: flex; justify-content: center;">

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型左对齐方式</h3>

    <div

      style="background-color: #d0d0d0;height: 250px;display: flex; justify-content: flex-start;"

    >

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型两边对齐方式</h3>

    <div

      style="background-color: #d0d0d0;height: 250px;display: flex; justify-content: space-between;"

    >

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型两边around对齐方式</h3>

    <div

      style="background-color: #d0d0d0;height: 250px;display: flex; justify-content: space-around;"

    >

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型交叉轴起点对齐方式</h3>

    <div style="background-color: #d0d0d0;height: 250px;display: flex; align-items: flex-start;">

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型交叉轴居中对齐方式</h3>

    <div style="background-color: #d0d0d0;height: 250px;display: flex; align-items: center;">

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型交叉轴尾部对齐方式</h3>

    <div style="background-color: #d0d0d0;height: 250px;display: flex; align-items:flex-end;">

      <div style="background-color: #D9006C;height: 50px;width: 50px;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px;">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px;">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px;">4</div>

    </div>

    <h3>盒子模型平分</h3>

    <div style="background-color: #d0d0d0;height: 250px;display: flex;">

      <div style="background-color: #D9006C;height: 50px;width: 50px; flex-grow: 1;">1</div>

      <div style="background-color: #FF8F59;height: 50px;width: 50px; flex-grow: 1">2</div>

      <div style="background-color: rgb(82, 76, 76);height: 50px;width: 50px; flex-grow: 1">3</div>

      <div style="background-color: #AD5A5A;height: 50px;width: 50px; flex-grow: 1">4</div>

    </div>

  </div>

</template>

<script>

export default {};

</script>

<style lang="scss" scoped>

</style>

**第06讲 登录页面布局**

1、设置public下面index.html高度100% ,内边距、外边距为 0

<style>

  html,body{

    height: 100%;

    margin: 0;

    padding: 0;

  }

</style>

2、设置home.vue、app.vue高度为 100%：

home.vue

<style lang="scss" scope>

.home{

  height: 100%;

}

</style>

app.vue

<style lang="scss" scope>

#app {

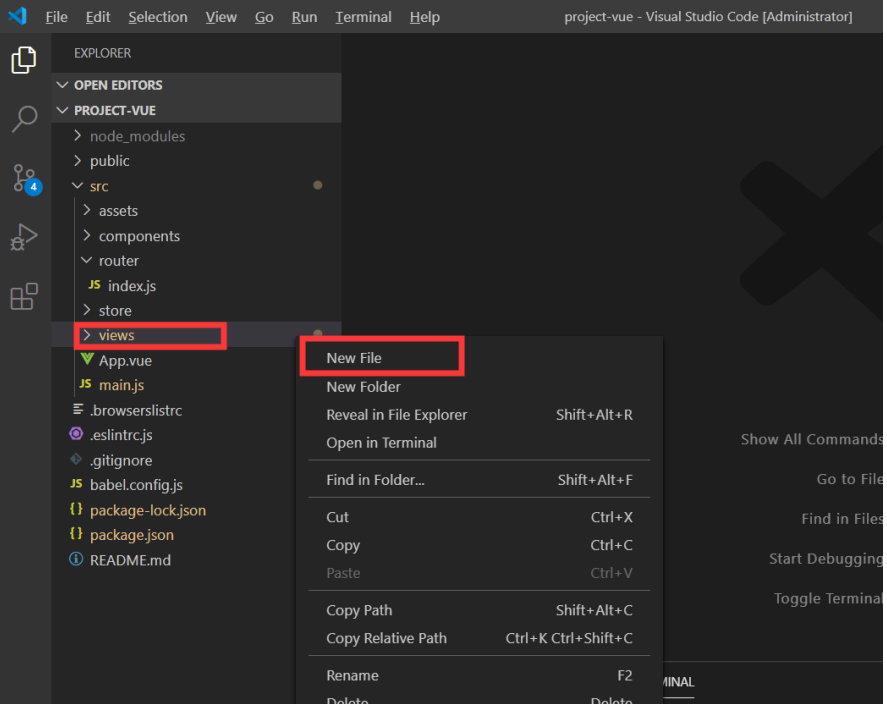
 height: 100%;

}

</style>

3、在views目录下新建Loign.vue页面

找到项目目录views，右键->NewFile->输入 Login.vue回车

****

4、实现代码

1）、用到组件 el-form 、el-form-item、el-input、el-row

2）、页面100%高度

3）、表单

宽高：350 px 300px

表单阴影 box-shadow: 0 0 25px #cac6c6;

内边距 20px 35px

圆角：10px

el-form-item：左边外边距 0

按钮宽度：100%

5、最终代码

<template>

    <div  class="login-container ub main-center cross-center">

       <el-form

         @submit.native.prevent="submit"

         size="medium"

         :rules="rules"

         ref="loginForm"

         :model="loginForm"

         label-width="80px"

         class="login-form"

       >

<el-form-item label>

<div class="login-title  ub main-center cross-center">

  系统登录

</div>

</el-form-item>

<el-form-item label>

  <el-input type="text" v-model="loginForm.username" placeholder="请输入用户名"></el-input>

</el-form-item>

<el-form-item label>

<el-input type="password" v-model="loginForm.password" placeholder="请输入密码"></el-input>

</el-form-item>

<el-form-item label>

  <el-row :gutter="10">

    <el-col :span="16">

     <el-input v-model="loginForm.code" placeholder="请输入验证码"></el-input>

    </el-col>

    <el-col :span="8">

      <el-input placeholder="点击刷新" readonly ></el-input>

    </el-col>

  </el-row>

</el-form-item>

<el-form-item label>

<el-row :gutter="20">

  <el-col :span="12">

  <el-button class="my-button" type="primary">登录</el-button>

  </el-col>

  <el-col :span="12">

   <el-button class="my-button" type="primary">重置</el-button>

  </el-col>

</el-row>

</el-form-item>

       </el-form>

    </div>

  </template>

   <script>

   export default {

     data() {

       return {

         loginForm: {

           username:"",

           password:""

         },

         rules: {},

       }

     },

     methods: {

       submit() {

         this.refs.loginForm.validate(valid => {

           if (valid) {

             console.log(valid)

           }

         })

       },

     },

   }

   </script>

<style lang="scss" scoped>

    .login-container{

        height: 100%;

    }

    .login-title{

      font-size: 24px;

      font-weight: 600;

    }

    .login-form{

      height: 300px;

      width: 350px;

      border-radius: 10px;

      box-shadow: 0 0 25px #cac6c6;

      padding: 20px 35px;

    }

    .login-container /deep/ .el-form-item\_\_content {

  margin-left: 0 !important;

}

.my-button{

  width: 100%;

}

</style>

6、最终效果图



总结：

size:用于控制该表单内组件的尺寸大小，有三个值medium/samll/mini

rules:表单验证规则

ref:类似于平时的div给定的id

model：绑定数据

el-row：相当于css3的弹性盒子模型布局，只是一个行分为24等份

**第07讲 登录表单验证**

1、表单非空验证

在不输入用户号码、密码、验证码的情况下，不能提交表单。

2、验证规则

prop、ref、model、rules这几个属性一定要添加，否则校验不生效。

ref表单被引用时的名称，标识 this.$refs.loginForm.validate()与这个对应

model表单数据对象，和data中的loginForm对应

rules表单校验规则，和data中保持一致

prop：表单域model字段，要和data中保持一致，在使用validate、resetFields方法的情况下，该属性是必填的

3、最终代码

<template>

    <div  class="login-container ub main-center cross-center">

       <el-form

         size="medium"

         :rules="rules"

         ref="loginForm"

         :model="loginForm"

         class="login-form"

       >

<el-form-item label>

<div class="login-title  ub main-center cross-center">

  系统登录

</div>

</el-form-item>

<el-form-item prop="username" label>

  <el-input type="text" v-model="loginForm.username" placeholder="请输入用户名"></el-input>

</el-form-item>

<el-form-item prop="password" label>

<el-input type="password" v-model="loginForm.password" placeholder="请输入密码"></el-input>

</el-form-item>

<el-form-item prop="code" label>

  <el-row :gutter="10">

    <el-col :span="16">

     <el-input v-model="loginForm.code" placeholder="请输入验证码"></el-input>

    </el-col>

    <el-col :span="8">

      <el-input placeholder="点击刷新" readonly ></el-input>

    </el-col>

  </el-row>

</el-form-item>

<el-form-item label>

<el-row :gutter="20">

  <el-col :span="12">

  <el-button class="my-button" @click="submit" type="primary">登录</el-button>

  </el-col>

  <el-col :span="12">

   <el-button class="my-button" @click="restSubmit" type="primary">重置</el-button>

  </el-col>

</el-row>

</el-form-item>

       </el-form>

    </div>

  </template>

   <script>

   export default {

     data() {

       return {

         loginForm: {

           username:"",

           password:"",

           code:""

         },

         rules: {

         username:[

             {

              required:true,

              trigger:"change",

              message:"请输入用户名"

             }

         ],

         password:[

           {

             required:true,

              trigger:"change",

              message:"请输入密码"

           }

         ],

         code:[

           {

             required:true,

              trigger:"change",

              message:"请输入验证码"

           }

         ]

         },

       }

     },

     methods: {

//响应登录事件

       submit() {

         this.$refs.loginForm.validate(valid => {

           if (valid) {

             console.log(valid)

           }

         })

       },

     },

//响应重置事件

    restSubmit(){

       this.$refs.loginForm.resetFields();

    }

   }

   </script>

<style lang="scss" scoped>

    .login-container{

        height: 100%;

    }

    .login-title{

      font-size: 24px;

      font-weight: 600;

    }

    .login-form{

      height: 300px;

      width: 350px;

      border-radius: 10px;

      box-shadow: 0 0 25px #cac6c6;

      padding: 20px 35px;

    }

    .login-container /deep/ .el-form-item\_\_content {

  margin-left: 0 !important;

}

.my-button{

  width: 100%;

}

</style>

4、最终效果



**第08讲 主界面布局**

1、采用上、下布局，下分为左右两部分，布局查看官方文档

[https://element.eleme.io/#/zh-CN/component/installation](https://element.eleme.io/%23/zh-CN/component/installation)

2、总体布局

修改home.vue文件，建立如下布局

2.1总体布局

样式需要设置高度为100%，才能占满屏幕高度

代码：

<template>

  <el-container class="home">

  <el-header style="background-color: darkkhaki;">Header</el-header>

  <el-container>

    <el-aside width="200px" style="background-color: brown;">Aside</el-aside>

    <el-container>

      <el-main style="background-color:bisque">Main</el-main>

      <el-footer style="background-color: darkcyan;">Footer</el-footer>

    </el-container>

  </el-container>

</el-container>

</template>

<script>

export default {

  name: 'Home',

  components: {

  }

}

</script>

<style lang="scss" scope>

.home{

  height: 100%;

}

</style>

2.2头部布局



头部代码：

<!--    头部 -->

    <el-header class="header ub main-justify cross-center">

       <div class="header-left-text">电子商务平台后管理系统</div>

       <div class="header-right ub main-center cross-center">

         <div>

          <el-dropdown placement="bottom-start">

            <img class="user-img" src="../assets/images/avatar.jpg" alt="用户头像"/>

            <el-dropdown-menu slot="dropdown">

              <el-dropdown-item >个人中心</el-dropdown-item>

              <el-dropdown-item >退出</el-dropdown-item>

            </el-dropdown-menu>

          </el-dropdown>

         </div>

       <div class="user-role">

         <div class="header-wellcome">欢迎你，管理员</div>

         <div class="header-time">2020.7.10 12:10:20 星期五</div>

       </div>

      </div>

    </el-header>

头部样式：

<style lang="scss" scope>

  .header{

    background-color: #167bd8;

    color: #fff;

    padding: 0 20px;

  }

  .header-left-text{

    font-size: 22px;

font-weight: bolder;

  }

  .user-img{

    height: 45px;

    width: 45px;

    border-radius: 50%;

    cursor: pointer;

  }

  .header-wellcome{

   font-size: 15px;

   font-weight: bolder;

  }

  .header-time{

    margin-top: 2px;

    font-size: 14px;

  }

  .user-role{

    margin-left: 10px;

  }

  .header-right{

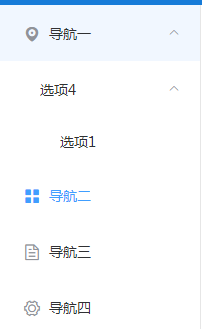
    margin-right: 30px;

  }

</style>

**第09讲 左侧菜单实现**

**1、左侧菜单栏布局**

****

2、**用到的组件 el-menu组件**

主要属性：

default-active 当前激活菜单的 index

unique-opened 是否只保持一个子菜单的展开

router 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

collapse 是否水平折叠收起菜单（仅在 mode 为 vertical 时可用）

示例代码：

<el-aside width="auto">

        <el-menu default-active="1-4-1" class="menu-bar" @open="handleOpen" @close="handleClose" :collapse="isCollapse">

          <el-submenu index="1">

            <template slot="title">

              <i class="el-icon-location"></i>

              <span slot="title">导航一</span>

            </template>

            <el-menu-item-group>

              <span slot="title">分组一</span>

              <el-menu-item index="1-1">选项1</el-menu-item>

              <el-menu-item index="1-2">选项2</el-menu-item>

            </el-menu-item-group>

            <el-menu-item-group title="分组2">

              <el-menu-item index="1-3">选项3</el-menu-item>

            </el-menu-item-group>

            <el-submenu index="1-4">

              <span slot="title">选项4</span>

              <el-menu-item index="1-4-1">选项1</el-menu-item>

            </el-submenu>

          </el-submenu>

          <el-menu-item index="2">

            <i class="el-icon-menu"></i>

            <span slot="title">导航二</span>

          </el-menu-item>

          <el-menu-item index="3" disabled>

            <i class="el-icon-document"></i>

            <span slot="title">导航三</span>

          </el-menu-item>

          <el-menu-item index="4">

            <i class="el-icon-setting"></i>

            <span slot="title">导航四</span>

          </el-menu-item>

        </el-menu>

      </el-aside>

样式

 .el-container /deep/ .el-menu{

   border-right: none !important;

 }

  .el-aside{

    border-right:1px solid #e6e6e6;

  }

.menu-bar:not(.el-menu--collapse){

  width: 200px;

  min-height: 400px;

}

**3、菜单组件重构**

3.1在components目录下新一MenuBar.vue组件，将Home.vue中的el-menu拷贝到MenuBar.vue中。在MenuBar.vue的 export default 中定义组件名MenuBar。

代码如下：

<template>

   <el-menu default-active="1-4-1" class="menu-bar" @open="handleOpen" @close="handleClose" :collapse="isCollapse">

          <el-submenu index="1">

            <template slot="title">

              <i class="el-icon-location"></i>

              <span slot="title">导航一</span>

            </template>

            <el-submenu index="1-4">

              <span slot="title">选项4</span>

              <el-menu-item index="1-4-1">选项1</el-menu-item>

            </el-submenu>

          </el-submenu>

          <el-menu-item index="2">

            <i class="el-icon-menu"></i>

            <span slot="title">导航二</span>

          </el-menu-item>

          <el-menu-item index="3">

            <i class="el-icon-document"></i>

            <span slot="title">导航三</span>

          </el-menu-item>

          <el-menu-item index="4">

            <i class="el-icon-setting"></i>

            <span slot="title">导航四</span>

          </el-menu-item>

        </el-menu>

</template>

<script>

    export default {

        name:'MenuBar'

    }

</script>

<style lang="scss" scoped>

</style>

在Home.vue中引入MenuBar组件

<script>

import MenuBar from '../components/MenuBar.vue'

components: {

    MenuBar

  }

</script>

在菜单的位置引入MenuBar

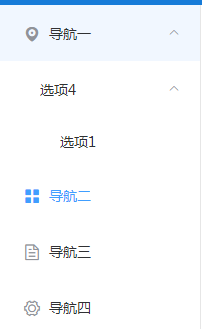
 <!-- 菜单栏 -->

      <el-aside width="auto">

     <menu-bar></menu-bar>

      </el-aside>

效果：



3.2新建MeuItem组件，对菜单进一步的封装

在components目录下新MenuItem.vue，代码如下：

<template>

  <div>

    <el-submenu index="1">

      <template slot="title">

        <i class="el-icon-location"></i>

        <span slot="title">导航一</span>

      </template>

      <el-submenu index="1-4">

        <span slot="title">选项4</span>

        <el-menu-item index="1-4-1">选项1</el-menu-item>

      </el-submenu>

    </el-submenu>

    <el-menu-item index="2">

      <i class="el-icon-menu"></i>

      <span slot="title">导航二</span>

    </el-menu-item>

    <el-menu-item index="3">

      <i class="el-icon-document"></i>

      <span slot="title">导航三</span>

    </el-menu-item>

    <el-menu-item index="4">

      <i class="el-icon-setting"></i>

      <span slot="title">导航四</span>

    </el-menu-item>

  </div>

</template>

<script>

export default {

   name: 'MenuItem'

};

</script>

<style lang="scss" scoped>

</style>

在MenuBar.vue中引入MenuItem

<template>

   <el-menu default-active="1-4-1" class="menu-bar" @open="handleOpen" @close="handleClose" :collapse="isCollapse">

         <menu-item ></menu-item>

        </el-menu>

</template>

<script>

import MenuItem from './MenuItem'

    export default {

         name:'MenuBar',

        components:{

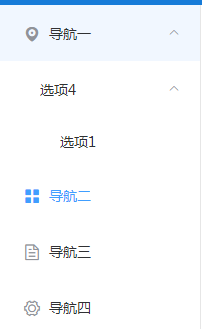
            MenuItem

        }

    }

</script>

效果：



3.3动态生成菜单

菜单的菜单项动态的生成，在MenuBar.vue中添加data(){}

代码如下：

export default {

  name: "MenuBar",

  components: {

    MenuItem

  },

  data() {

    return {

      menuList: [

        {

          children: [

            {

              children: [],

              code: "sys:dept",

              createTime: 1586703509000,

              icon: "el-icon-copy-document",

              id: 33,

              isHome: 0,

              label: "机构管理",

              name: "departmentList",

              orderNum: 2,

              parentId: 17,

              path: "/departmentList",

              remark: "机构管理",

              type: "1",

              updateTime: 1586337139000,

              url: "/system/Department/DepartmentList"

            },

            {

              children: [],

              code: "sys:user",

              createTime: 1691464271000,

              icon: "el-icon-s-custom",

              id: 18,

              isHome: 0,

              label: "用户管理",

              name: "userList",

              orderNum: 3,

              parentId: 17,

              path: "/userList",

              type: "1",

              updateTime: 1691565988000,

              url: "/system/User/UserList"

            },

            {

              children: [],

              code: "sys:role",

              createTime: 1691464271000,

              icon: "el-icon-rank",

              id: 23,

              isHome: 0,

              label: "角色管理",

              name: "roleList",

              orderNum: 4,

              parentId: 17,

              path: "/roleList",

              type: "1",

              updateTime: 1691565988000,

              url: "/system/Role/RoleList"

            },

            {

              children: [],

              code: "sys:menu",

              createTime: 1691464271000,

              icon: "el-icon-menu",

              id: 28,

              isHome: 0,

              label: "权限管理",

              name: "menuList",

              orderNum: 5,

              parentId: 17,

              path: "/menuList",

              type: "1",

              updateTime: 1691565988000,

              url: "/system/Menu/MenuList"

            }

          ],

          code: "sys:manage",

          createTime: 1691464271000,

          icon: "el-icon-document",

          id: 17,

          isHome: 0,

          label: "系统管理",

          orderNum: 1,

          parentId: 0,

          path: "/system",

          type: "0",

          updateTime: 1691565988000

        },

        {

          children: [

            {

              children: [],

              code: "sys:goodsCategory",

              createTime: 1586703272000,

              icon: "el-icon-s-data",

              id: 36,

              isHome: 0,

              label: "分类管理",

              name: "goodCategory",

              orderNum: 1,

              parentId: 34,

              path: "/goodCategory",

              type: "1",

              updateTime: 1586683590000,

              url: "/goods/goodsCategory/goodsCategoryList"

            },

            {

              children: [],

              code: "sys:goodsBrand",

              createTime: 1586683924000,

              icon: "el-icon-tickets",

              id: 37,

              isHome: 0,

              label: "品牌管理",

              name: "goodsBrand",

              orderNum: 2,

              parentId: 34,

              path: "/goodsBrand",

              type: "1",

              updateTime: 1586683924000,

              url: "/goods/goodsBrand/goodsBrandList"

            }

          ],

          code: "sys:goods",

          createTime: 1586702987000,

          icon: "el-icon-picture",

          id: 34,

          isHome: 0,

          label: "商品管理",

          name: "",

          orderNum: 2,

          parentId: 0,

          path: "/goods",

          type: "0",

          updateTime: 1586683323000

        },

        {

          children: [

            {

              children: [],

              code: "sys:systemCode",

              createTime: 1587012282000,

              icon: "el-icon-files",

              id: 43,

              isHome: 0,

              label: "代码生成",

              name: "systemCode",

              orderNum: 0,

              parentId: 42,

              path: "/systemCode",

              type: "1",

              updateTime: 1586684646000,

              url: "/system/config/code"

            },

            {

              children: [],

              code: "sys:document",

              createTime: 1586748705000,

              icon: "el-icon-s-operation",

              id: 77,

              isHome: 0,

              label: "接口文档",

              name: "document",

              orderNum: 0,

              parentId: 42,

              path: "/document",

              type: "1",

              updateTime: 1586748705000,

              url: "/system/config/systemDocument"

            }

          ],

          code: "sys:systenConfig",

          createTime: 1586703003000,

          icon: "el-icon-receiving",

          id: 42,

          isHome: 0,

          label: "系统工具",

          name: "",

          orderNum: 3,

          parentId: 0,

          path: "/systenConfig",

          type: "0",

          updateTime: 1586684441000

        }

      ]

    };

  }

};

在MenuBar.vue中向子组件传递数据

   <menu-item :menuList='menuList'></menu-item>

在MenuItem.vue接收父组件传递过来的数据

export default {

   name: 'MenuItem',

   props:['menuList'],

};

修改MenuItem中的菜单

<template>

  <div>

  <template v-for="menu in menuList">

   <el-submenu  v-if='menu.children.length>0' :index="menu.path" :key='menu.path'>

      <template slot="title">

        <i :class="menu.icon"></i>

        <span style="font-size:15px;font-weight:600"

slot="title">{{menu.label}}</span>

      </template>

     <!--循环调用子菜单-->

    <menu-item :menuList='menu.children'></menu-item>

    </el-submenu>

    <el-menu-item v-else :index="menu.path" :key='menu.path'>

      <i :class="menu.icon"></i>

      <span style="font-size:15px;font-weight:600"

slot="title">{{menu.label}}</span>

    </el-menu-item>

  </template>

  </div>

</template>

<script>

import MenuItem from './MenuItem.vue'

export default {

   name: 'MenuItem',

   props:['menuList'],

   components:{

       MenuItem

   }

};

</script>

此外需要在MenuBar.vue中设置，unique-opened="true",是否只保持一个子菜单的展开

  <el-menu

    default-active="1-4-1"

    class="menu-bar"

    @open="handleOpen"

    @close="handleClose"

    :collapse="isCollapse"

    :unique-opened="true" >

最终效果如下：



**第10 讲 tabs选项卡实现与菜单联动**

1、tabs选项卡组件：

1） tabs 组件

2）常用属性：

value ：选中选项卡的name

type ：选项卡风格 可选择 card / border-card

closable ： 选项卡是否可可以关闭

2、实现效果：

点击左侧菜单，右边内容展示区要显示对应的tabs菜单选项卡，tabs选项卡可以关闭

3、实现原理：

点击左侧菜单，把当前点击的菜单对象，添加到tabs选项卡里面；

关闭tabs选项卡：如果为首页桌面不能关闭

4、tabs组件实现

在 components下新建tabs.vue组件，在官网上拷贝自定义增加标签页触发器的代码，

代码如下：

<template>

  <el-tabs v-model="editableTabsValue" type="card" closable @tab-remove="removeTab">

    <el-tab-pane

      v-for="item in editableTabs"

      :key="item.name"

      :label="item.title"

      :name="item.name"

    >{{item.content}}</el-tab-pane>

  </el-tabs>

</template>

<script>

export default {

  data() {

    return {

      editableTabsValue: "2",

      editableTabs: [

        {

          title: "Tab 1",

          name: "1",

          content: "Tab 1 content"

        },

        {

          title: "Tab 2",

          name: "2",

          content: "Tab 2 content"

        }

      ],

      tabIndex: 2

    };

  },

  methods:{

       removeTab(targetName) {

        let tabs = this.editableTabs;

        let activeName = this.editableTabsValue;

        if (activeName === targetName) {

          tabs.forEach((tab, index) => {

            if (tab.name === targetName) {

              let nextTab = tabs[index + 1] || tabs[index - 1];

              if (nextTab) {

                activeName = nextTab.name;

              }

            }

          });

        }

        this.editableTabsValue = activeName;

        this.editableTabs = tabs.filter(tab => tab.name !== targetName);

  }

}

};

</script>

<style lang="scss" scoped>

</style>

在Home.vue中引入tabs组件

<script>

import MenuBar from '../components/MenuBar.vue'

import tabs from '../components/tabs.vue'

  export default {

    name: 'Home',

    components: {

      MenuBar,

      tabs

    }

  }

</script>

   <!-- 右边显示区 -->

        <el-main style="padding: 0px;">

          <tabs></tabs>

        </el-main>

运行效果：



5、实现菜单与tabs选项卡联动(难点)

5.1将tabs选项卡数据存放在store里面，在store目录下新建一个MenuStore.js，如下所示

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex)

export default {

  state: {

  },

  mutations: {

  },

  actions: {

  }

}

在store下的index.js中引入

import Vue from 'vue'

import Vuex from 'vuex'

import MenuStore from '../store/MenuStore'

Vue.use(Vuex)

export default new Vuex.Store({

  modules: {

    MenuStore

  }

})

修改MenuStore.js中的代码

state相当于一个内存，存储数据，但不能直接操作state里面的数据，需要mutations来操作state里面存储的数据。将tabs.vue中的return中的数据屏蔽掉，将此数据交给MenuStore状态来管理。

修改tabs.vue

<script>

import { mapState } from "vuex";

export default {

  computed: {

    ...mapState({

      //tabs选项卡数据

      editableTabs: state => state.MenuStore.tabs,

      //当前激活的选项卡

      editableTabsValue: state => state.MenuStore.editableTabsValue

    })

  },

  data() {

    return {

      // editableTabsValue: "2",

      // editableTabs: [

      //   {

      //     title: "Tab 1",

      //     name: "1",

      //     content: "Tab 1 content"

      //   },

      //   {

      //     title: "Tab 2",

      //     name: "2",

      //     content: "Tab 2 content"

      //   }

      // ],

      // tabIndex: 2

    };

  },

  methods: {

    removeTab(targetName) {

      let tabs = this.editableTabs;

      let activeName = this.editableTabsValue;

      if (activeName === targetName) {

        tabs.forEach((tab, index) => {

          if (tab.name === targetName) {

            let nextTab = tabs[index + 1] || tabs[index - 1];

            if (nextTab) {

              activeName = nextTab.name;

            }

          }

        });

      }

      this.editableTabsValue = activeName;

      this.editableTabs = tabs.filter(tab => tab.name !== targetName);

    }

  }

};

</script>

修改MenuStore.js

export default {

  state: {

    //当前激活的选项卡

    editableTabsValue:'desktop',

    //tabs选项卡数据

    tabs:[

      {

        title:'首页',

        name:'desktop'

      }

    ]

  },

  mutations: {

  },

  actions: {

  }

}

运行效果



5.2实现菜单与选项卡联动功能

1、在MenuItem.vue中给el-menu-item加点击事件

<el-menu-item  @click="selectMenu(menu)"  v-else :index="menu.path" :key='menu.path'>

2、修改MenuStore.js在 mutations中添加如下代码：

mutations: {

     //菜单点击时候调用

     selectMenu(state,val){

      console.log(val);

      //1.把当前点击的菜单对象，加到tabs里面

      let res = state.tabs.findIndex(item => item.name === val.name);

      //2.判断tabs是否存在，不存在才加入到tabs

      if(res === -1){

          let obj = {};

          obj.title = val.label;

          obj.name = val.name;

          state.tabs.push(obj);

      }

      //3、设置当前选中对象

      state.editableTabsValue = val.name;

     }

  },

3、在MenuItem.vue中添加 methods

methods:{

      selectMenu(item){

        //设置选项卡

        this.$store.commit('selectMenu',item);

      }

   }

运行效果：



关闭选项卡的功能失效，修改tabs.vue中computed的代码

//import { mapState } from "vuex";

export default {

  computed: {

    // ...mapState({

    //   //tabs选项卡数据

    //   editableTabs: state => state.MenuStore.tabs,

    //   //当前激活的选项卡

    //   editableTabsValue: state => state.MenuStore.editableTabsValue

    // })

    //当前激活的选项卡

    editableTabsValue: {

      get() {

        return this.$store.state.MenuStore.editableTabsValue;

      },

      set(val) {

        this.$store.state.MenuStore.editableTabsValue = val;

      }

    },

    //tabs选项卡数据

    editableTabs: {

      get() {

        return this.$store.state.MenuStore.tabs;

      },

      set(val) {

        this.$store.state.MenuStore.tabs = val;

      }

    }

  },



实现首页不能关的功能，修改tabs.vue中的removeTab

  removeTab(targetName) {

      //首页不能关闭

       if (targetName === "desktop") {

        return;

      }

修改MenuBar.vue，删除

@open="handleOpen"

@close="handleClose"

在data中

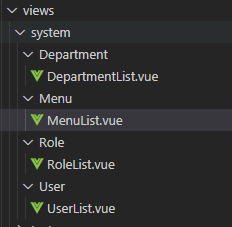
data() {

    return {

      isCollapse:false,｝｝

6、解决刷新浏览器，vuex里的 tabs不存在的问题

1)、在views下新建system目录，在此目录下新建几个页面如下：



页面中的代码：

<template>

    <div>

部门管理

    </div>

</template>

<script>

    export default {

    }

</script>

<style lang="scss" scoped>

</style>

修改路由router下的，index.js文件，在home路由添加children

{

    path: '/',

    name: 'Home',

    component: Home,

    children:[

      {

        path:'/deparmentList',

        name: 'departmentList',

        component: () => import( '@/views/system/Department/DepartmentList.vue')

      },

      {

        path:'/userList',

        name: 'userList',

        component: () => import( '@/views/system/User/UserList.vue')

      },

      {

        path:'/menuList',

        name: 'menuList',

        component: () => import( '@/views/system/Menu/MenuList.vue')

      },

      {

        path:'/roleList',

        name: 'roleList',

        component: () => import( '@/views/system/Role/RoleList.vue')

      },

    ]

  },

并在home.vue中添加路由标签

   <!-- 右边显示区 -->

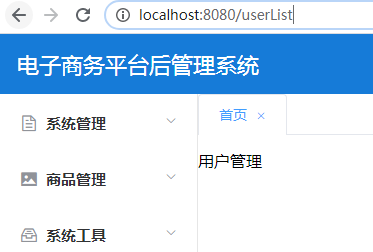
        <el-main style="padding: 0px;">

         <tabs></tabs>

         <router-view></router-view>

        </el-main>

运行效果：



2)、解决刷新浏览器，vuex里的tabs不存在的问题

修改MenuStore.js，在MenuStore.js中 selectMenu()中添加如下代码：

 //设置当前打开的选项卡

      sessionStorage.setItem('tabsList',JSON.stringify(state.tabs));

在MenuStore.js的mutations中添加两个方法，用于获取当前tabsList的数据和当前需要激活的tabs选项卡。

//设置当前选中选项卡

     getTabs(state){

      let tabs = sessionStorage.getItem('tabList');

      if(tabs){

          state.tabs =  JSON.parse(tabs);

      }

  },

  //设置当前激活的选项卡

  setActiveTabs(state,val){

      state.editableTabsValue = val;

  }

修改后的mutations最终代码为：

 mutations: {

    //菜单点击时候调用

    selectMenu(state, val) {

      console.log(val);

      //1.把当前点击的菜单对象，加到tabs里面

      let res = state.tabs.findIndex(item => item.name === val.name);

      //2.判断tabs是否存在，不存在才加入到tabs

      if (res === -1) {

        let obj = {};

        obj.title = val.label;

        obj.name = val.name;

        state.tabs.push(obj);

      }

      //3、设置当前选中对象

      state.editableTabsValue = val.name;

      //设置当前打开的选项卡

      sessionStorage.setItem('tabsList',JSON.stringify(state.tabs));

    },

    //设置当前选中选项卡

     getTabs(state){

      let tabs = sessionStorage.getItem('tabsList');

      if(tabs){

        let currentTabsList =JSON.parse(tabs);

        state.tabs = currentTabsList;

      }

  },

  //设置当前激活的选项卡

  setActiveTabs(state,val){

      state.editableTabsValue = val;

  }

  },

在main.js中添加路由拦截，在new Vue({ router, store, render: h => h(App)}).$mount('#app')方法之前加入如下代码：

router.beforeEach((to,from,next) => {

  //to 即将进入的路由

  //from即将离开的路由

  //设置tabs数据

  store.commit('getTabs');

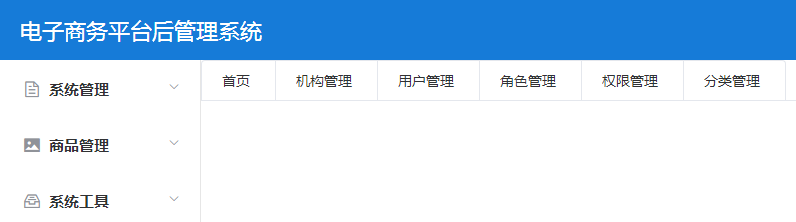
  //设置激活选项卡

  store.commit('setActiveTabs',to.name);

 next(); //继续往下执行

  })

效果



7、解决联动问题

修改MenuBar.vue，实现点击菜单项联动选项卡

  <el-menu

    :default-active="$route.path"

    class="menu-bar"

    :collapse="isCollapse"

  >

内容的联动实现，修改MenuItem.vue，在methods下添加如下代码： this.$router.push({name:item.name});

methods:{

      selectMenu(item){

        //设置选项卡

        this.$store.commit('selectMenu',item);

        //设置路由

        this.$router.push({name:item.name});

      }

   }



8、tabs点击事件

修改tabs.vue, 在tabs组件上添加点击事件.

 <el-tabs :value="editableTabsValue" type="card" closable @tab-click="tabClick" @tab-remove="removeTab">

在methods中添加方法tabClick()，代码如下：

 //点击选项卡

    tabClick(tab){

      let obj={};

      if(tab.name==="desktop"){

        obj.label="首页";

      }else{

        obj.label=tab.label;

      }

      obj.name=tab.name;

      this.$store.commit("selectMenu",obj);

      this.$router.push({name:tab.name});

    },

针对首页无效的情况，在Views下新建Desktop.vue，

<template>

    <div>

首页

    </div>

</template>

<script>

    export default {

    }

</script>

<style lang="scss" scoped>

</style>

在路由home的children:下添加如下代码：

 {

        path:' /desktop',

        name: 'desktop',

        component: () => import( '@/views/Desktop.vue')

      },

效果：



9、解决几个小bug

1）修改关闭选项卡的小bug,tab的bug：在打开多个标签的情况下，如果关闭了部分标签，在下一次刷新时这些标签又会回来,修改tabs.vue，在removeTab方法的末尾添加如下代码：

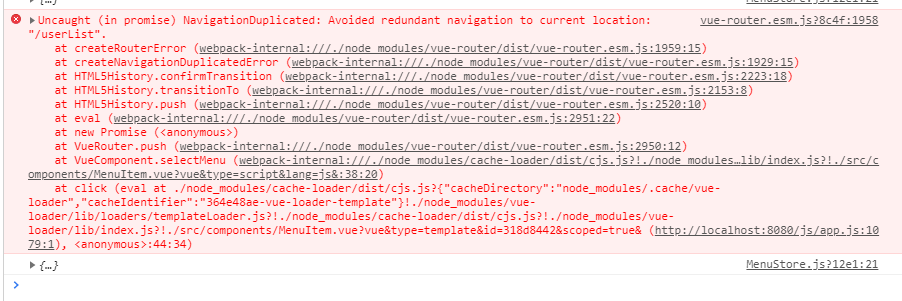
 //清除sessionStorate

     sessionStorage.clear();

2）解决项目中报错 Error: Avoided redundant navigation to current location: “/xxx”. 的解决方案

**问题产生的操作：点击左侧同一菜单两次，路由切换两次**

**原因：在路由跳转的时候同一个路由多次添加是不被允许的**



// 解决ElementUI导航栏中的vue-router在3.0版本以上重复点菜单报错问题

const originalPush = VueRouter.prototype.push

VueRouter.prototype.push = function push(location) {

  return originalPush.call(this, location).catch(err => err)

}

**第11讲 动态路由的生成**

1、使用动态路由的原因

由于系统左侧菜单是根据不同用户，拥有不同的菜单，所以左侧菜单和路由是动态的生成的。

2、实现原理

利用 vue-router 的 addRoutes方法可以动态添加路由 ，router.addRoutes(routes: Array)，动态添加路由，参数必是一个符合 routes选项要求的数组。

3、实现思路

1）用户登录时，会根据用户自己拥有的权限，返回菜单数据和路由数据

2）登录成功，返回菜单和路由数据，前端保存到sessionStorage中；在Login.vue页面登录提交事件中添加如下代码，mentList表示后台返回成功的左侧菜单数据，routerList表示后台返回的路由数据。

 //后台登录成功返回的菜单数据

          let menuList = [

            {

              children: [

                {

                  children: [],

                  code: "sys:dept",

                  createTime: 1586703509000,

                  icon: "el-icon-copy-document",

                  id: 33,

                  isHome: 0,

                  label: "机构管理",

                  name: "departmentList",

                  orderNum: 2,

                  parentId: 17,

                  path: "/departmentList",

                  remark: "机构管理",

                  type: "1",

                  updateTime: 1586337139000,

                  url: "/system/Department/DepartmentList"

                },

                {

                  children: [],

                  code: "sys:user",

                  createTime: 1691464271000,

                  icon: "el-icon-s-custom",

                  id: 18,

                  isHome: 0,

                  label: "用户管理",

                  name: "userList",

                  orderNum: 3,

                  parentId: 17,

                  path: "/userList",

                  type: "1",

                  updateTime: 1691565988000,

                  url: "/system/User/UserList"

                },

                {

                  children: [],

                  code: "sys:role",

                  createTime: 1691464271000,

                  icon: "el-icon-rank",

                  id: 23,

                  isHome: 0,

                  label: "角色管理",

                  name: "roleList",

                  orderNum: 4,

                  parentId: 17,

                  path: "/roleList",

                  type: "1",

                  updateTime: 1691565988000,

                  url: "/system/Role/RoleList"

                },

                {

                  children: [],

                  code: "sys:menu",

                  createTime: 1691464271000,

                  icon: "el-icon-menu",

                  id: 28,

                  isHome: 0,

                  label: "权限管理",

                  name: "menuList",

                  orderNum: 5,

                  parentId: 17,

                  path: "/menuList",

                  type: "1",

                  updateTime: 1691565988000,

                  url: "/system/Menu/MenuList"

                }

              ],

              code: "sys:manage",

              createTime: 1691464271000,

              icon: "el-icon-document",

              id: 17,

              isHome: 0,

              label: "系统管理",

              orderNum: 1,

              parentId: 0,

              path: "/system",

              type: "0",

              updateTime: 1691565988000

            },

            {

              children: [

                {

                  children: [],

                  code: "sys:goodsCategory",

                  createTime: 1586703272000,

                  icon: "el-icon-s-data",

                  id: 36,

                  isHome: 0,

                  label: "分类管理",

                  name: "goodCategory",

                  orderNum: 1,

                  parentId: 34,

                  path: "/goodCategory",

                  type: "1",

                  updateTime: 1586683590000,

                  url: "/goods/goodsCategory/goodsCategoryList"

                },

                {

                  children: [],

                  code: "sys:goodsBrand",

                  createTime: 1586683924000,

                  icon: "el-icon-tickets",

                  id: 37,

                  isHome: 0,

                  label: "品牌管理",

                  name: "goodsBrand",

                  orderNum: 2,

                  parentId: 34,

                  path: "/goodsBrand",

                  type: "1",

                  updateTime: 1586683924000,

                  url: "/goods/goodsBrand/goodsBrandList"

                }

              ],

              code: "sys:goods",

              createTime: 1586702987000,

              icon: "el-icon-picture",

              id: 34,

              isHome: 0,

              label: "商品管理",

              name: "",

              orderNum: 2,

              parentId: 0,

              path: "/goods",

              type: "0",

              updateTime: 1586683323000

            },

            {

              children: [

                {

                  children: [],

                  code: "sys:systemCode",

                  createTime: 1587012282000,

                  icon: "el-icon-files",

                  id: 43,

                  isHome: 0,

                  label: "代码生成",

                  name: "systemCode",

                  orderNum: 0,

                  parentId: 42,

                  path: "/systemCode",

                  type: "1",

                  updateTime: 1586684646000,

                  url: "/system/config/code"

                },

                {

                  children: [],

                  code: "sys:document",

                  createTime: 1586748705000,

                  icon: "el-icon-s-operation",

                  id: 77,

                  isHome: 0,

                  label: "接口文档",

                  name: "document",

                  orderNum: 0,

                  parentId: 42,

                  path: "/document",

                  type: "1",

                  updateTime: 1586748705000,

                  url: "/system/config/systemDocument"

                }

              ],

              code: "sys:systenConfig",

              createTime: 1586703003000,

              icon: "el-icon-receiving",

              id: 42,

              isHome: 0,

              label: "系统工具",

              name: "",

              orderNum: 3,

              parentId: 0,

              path: "/systenConfig",

              type: "0",

              updateTime: 1586684441000

            }

          ];

          //路由数据

          let routerList = [

            {

              children: [],

              code: "sys:systemCode",

              createTime: 1587012282000,

              icon: "el-icon-files",

              id: 43,

              isHome: 0,

              label: "代码生成",

              name: "systemCode",

              orderNum: 0,

              parentId: 42,

              path: "/systemCode",

              type: "1",

              updateTime: 1586684646000,

              url: "/system/config/code"

            },

            {

              children: [],

              code: "sys:document",

              createTime: 1586748705000,

              icon: "el-icon-s-operation",

              id: 77,

              isHome: 0,

              label: "接口文档",

              name: "document",

              orderNum: 0,

              parentId: 42,

              path: "/document",

              type: "1",

              updateTime: 1586748705000,

              url: "/system/config/systemDocument"

            },

            {

              children: [],

              code: "sys:goodsCategory",

              createTime: 1586703272000,

              icon: "el-icon-s-data",

              id: 36,

              isHome: 0,

              label: "分类管理",

              name: "goodCategory",

              orderNum: 1,

              parentId: 34,

              path: "/goodCategory",

              type: "1",

              updateTime: 1586683590000,

              url: "/goods/goodsCategory/goodsCategoryList"

            },

            {

              children: [],

              code: "sys:goodsBrand",

              createTime: 1586683924000,

              icon: "el-icon-tickets",

              id: 37,

              isHome: 0,

              label: "品牌管理",

              name: "goodsBrand",

              orderNum: 2,

              parentId: 34,

              path: "/goodsBrand",

              type: "1",

              updateTime: 1586683924000,

              url: "/goods/goodsBrand/goodsBrandList"

            },

            {

              children: [],

              code: "sys:dept",

              createTime: 1586703509000,

              icon: "el-icon-copy-document",

              id: 33,

              isHome: 0,

              label: "机构管理",

              name: "departmentList",

              orderNum: 2,

              parentId: 17,

              path: "/departmentList",

              remark: "机构管理",

              type: "1",

              updateTime: 1586337139000,

              url: "/system/Department/DepartmentList"

            },

            {

              children: [],

              code: "sys:user",

              createTime: 1691464271000,

              icon: "el-icon-s-custom",

              id: 18,

              isHome: 0,

              label: "用户管理",

              name: "userList",

              orderNum: 3,

              parentId: 17,

              path: "/userList",

              type: "1",

              updateTime: 1691565988000,

              url: "/system/User/UserList"

            },

            {

              children: [],

              code: "sys:role",

              createTime: 1691464271000,

              icon: "el-icon-rank",

              id: 23,

              isHome: 0,

              label: "角色管理",

              name: "roleList",

              orderNum: 4,

              parentId: 17,

              path: "/roleList",

              type: "1",

              updateTime: 1691565988000,

              url: "/system/Role/RoleList"

            },

            {

              children: [],

              code: "sys:menu",

              createTime: 1691464271000,

              icon: "el-icon-menu",

              id: 28,

              isHome: 0,

              label: "权限管理",

              name: "menuList",

              orderNum: 5,

              parentId: 17,

              path: "/menuList",

              type: "1",

              updateTime: 1691565988000,

              url: "/system/Menu/MenuList"

            }

          ];

          //保存菜单数据

          sessionStorage.setItem("menuList",JSON.stringify(menuList));

          //保存路由数据

          sessionStorage.setItem("routerList",JSON.stringify(routerList));

          //动态生成路由

          this.$store.commit("getMenuList",this.$router);

          //跳转到home页面

          this.$router.push("home");

3）调用store中动态的生成菜单和路由，在MenuStore.js的state中加入

//左侧菜单数据menu\_data:[]，mutations中添加如下方法，代码如下：

//获取菜单数据和生成路由

    getMenuList(state, router) {

      //1、取出菜单数据

      let menuList = sessionStorage.getItem('menuList');

      //2、设置菜单数据

      if(menuList){

        state.menu\_data = JSON.parse(menuList);

    }

      //3、取出路由数据

      let oldrouterList = sessionStorage.getItem('routerList');

      let routerList = [];

      if(oldrouterList){

          routerList = JSON.parse(oldrouterList);

      }

      //4、动态生成路由

      //4.1获取原来的路由

      let oldRouter = router.options.routes;

      //遍历后台返回的路由数据，动态生成路由

      //component: () => import(/\* webpackChunkName: "about" \*/ '../views/Login.vue')

      routerList.forEach(item =>{

          item.component = () => import(`@/views${item.url}.vue`);

          oldRouter[1].children.push(item);

      })

      //5、添加到现有路由里面

      router.addRoutes(oldRouter);

    }

4）router下面的index.js中的routes只留如下代码(这个顺序很重要)：

 const routes = [

    {

    {

      path:'/login',

      name:'Login',

      component: () => import( '../views/Login.vue')

    },

  {

    path: '/',

    name: 'Home',

    component: Home,

    children:[

      {

        path:'/desktop',

        name: 'desktop',

        component: () => import( '@/views/Desktop.vue')

      },

]

5）在MenuBar.vue组件中注释menuList的代码，在computed中获取store中的menu\_data数据代码如下：

import {mapState} from 'vuex';

 computed:{

   ...mapState({

    menuList: state => state.MenuStore.menu\_data

   })

   },

6）防止刷新后store中的menu\_data不存在，那么需要在main.js做如下判断，如果store中的menu\_data不存在，要重新加载一个sessionStorage中的数据,在main.js中做如下更改:

//添加路由拦截

router.beforeEach((to, from, next) => {

  //to即将进入的路由

  //from即将离开的路由

  store.commit('getTabs');

  store.commit('setActiveTabs', to.name);

  //如果store中的菜单数据menu\_data被刷新了，那么从新加载

  if (store.state.MenuStore.menu\_data.length == 0) {

    store.commit('getMenuList', router); next({ path: to.path })

  } else {

    //next断续往下执行

    next();

  }

})

上述代码会产生内存溢出，修改成如下代码：

//添加路由拦截

router.beforeEach((to, from, next) => {

  //to即将进入的路由

  //from即将离开的路由

  store.commit('getTabs');

  store.commit('setActiveTabs', to.name);

  //如果store中的菜单数据menu\_data被刷新了，那么从新加载

   //从sessionStorage中获得数据

  let menuData=sessionStorage.getItem('menuList');

  //如果访问路为登录

   if(to.path==='/login'){

     //如果有数据

     if(menuData){

       //跳到首页

     next({path:'/home'})

     }else{

       next();

     }

   }else{

     //如果没有数据，路由不等于login

     if(!menuData && to.name!=='login'){

       next({path:'/login'});

     }else{

      if (store.state.MenuStore.menu\_data.length == 0) {

        store.commit('getMenuList', router);

         next({ path: to.path })

      } else {

        //next断续往下执行

        next();

      }

     }

   }

})

修改登录的路由为如下代码：

{

    path: '/login',

    name: 'Login',

    component: () => import('../views/Login.vue')

  },

**第13 讲 实现左侧菜单的展开和收缩**

**1、实现效果**

点击收缩图标，左侧菜单收缩

**2、实现原理**

利用el-menu中的属性collapse可以设置菜单展开和收缩

**3、收缩图标制作**

展开时图标 el-icon-s-fold

收缩时图标el-icon-s-unfold

**3.1在home页面el-main中tabs前面添加收缩图标，代码如下：**

  <i class="el-icon-s-fold arrow-icon"></i>

设置样式如下：

.arrow-icon{

  float: left;

  background: #eaedf1;

  border: 1px solid transparent;

  font-size: 25px;

  height: 39px;

  line-height: 39px!important;

  width: 40px;

  text-align: center;

}

**3.2为收缩图标添加点击事件**

 <i class="el-icon-s-fold arrow-icon" @click="arrowBtn"></i>

在methods中添加点击事件响应方法

 methods:{

    //收缩箭头点击事件

    arrowBtn(){

     console.log("点击了我");

    }

  }

由于tabbar把按钮隐藏了，因此需要添加以下样式

//解扭图票不能点击的问题

 .el-container /deep/.el-tabs\_\_header {

  position: static !important;

}

**3.3设置菜单收缩属性 isCollapse**

1）在MenuStore.js的state中添加 isCollapse:false

2）在MenuStore.js的mutations中添加如下代码

//设置图标收缩属性

        setOpenOrClose(state) {

            state.isCollapse = !state.isCollapse;

        }

**3）**在MenuBar.vue中通过计算属性获取 isCollapse，把原来的注释，代码如下

 computed: {

     ...mapState({

       //获取菜单数据

       menuList:state=>state.MenuStore.menu\_data,

       //获取vuex中的isCollapse收缩属性

       isCollapse: state => state.MenuStore.isCollapse

     })

  },

**4）在Home.vue页面收缩图标点击事件中调用MenuStore.js 的setOpenOrClose方法**

  methods: {

    //收缩箭头点击事件

    arrowBtn() {

      console.log("点击了我");

      this.$store.commit('setOpenOrClose');

    }

  }

**5）让点击图标自动切换方向**

在home.vue中获取MenuStore.js中的isCollapse

import {mapState} from 'vuex'

  computed: {

    ...mapState({

      isCollapse: (state) => state.MenuStore.isCollapse,

    }),

  },

动态设置图标样式<i class="arrow-icon" :class="[isCollapse ? 'el-icon-s-unfold' : ' el-icon-s-fold']" @click="arrowBtn"></i>

**3.4解决图标收缩时，字不能影藏的问题**

**1）**安装 npm install --save vue-fragment

2）引入fragement

**在** main.js中加入如下代码

//引入fragment插件

import Fragment from'vue-fragment'

//使用插件

Vue.use(Fragment.Plugin)

3）修改MenuItem.vue，用fragment包裹，代码如下

<template>

<fragment>

    <template v-for="menu in menuList">

    <el-submenu  v-if='menu.children.length>0' :index="menu.path" :key='menu.path'>

      <template slot="title">

        <i :class="menu.icon"></i>

        <span style="font-size:15px;font-weight:600" slot="title">{{menu.label}}</span>

      </template>

      <!-- 循环调用子菜单-->

      <menu-item :menuList='menu.children'></menu-item>

    </el-submenu>

    <el-menu-item @click="selectMenu(menu)" v-else :index="menu.path" :key='menu.path'>

      <i :class="menu.icon"></i>

      <span style="font-size:15px;font-weight:600" slot="title">{{menu.label}}</span>

    </el-menu-item>

    </template>

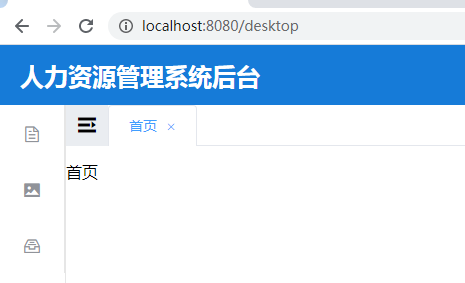
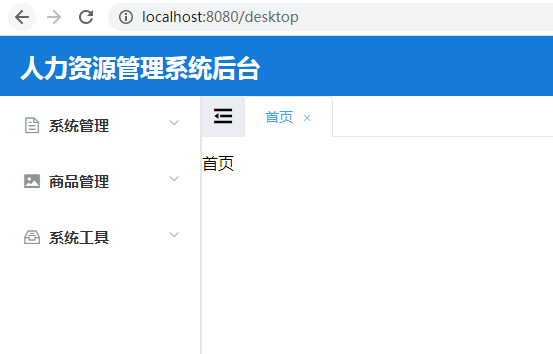
</fragment>

</template>

将el-aside的宽度设置为auto

 <el-aside width="auto">

效果：

****

**第14讲 首页与头部时间动态显示**

1、首页制作

首页进行图表显示，使用v-charts。

在使用 echarts 生成图表时，经常需要做繁琐的数据类型转化、修改复杂的配置项，v-charts 的出现正是为了解决这个痛点。基于 Vue2.0 和 echarts 封装的 v-charts 图表组件，只需要统一提供一种对前后端都友好的数据格式设置简单的配置项，便可轻松生成常见的图表。

官网：<https://v-charts.js.org/>

1）[npm安装](https://v-charts.js.org/#/?id=npm%e5%ae%89%e8%a3%85)

npm i v-charts echarts -S

2）在main.js中引入

import VCharts from 'v-charts'

Vue.use(VCharts)

3)基本属性



4)简单例子

<template>

<ve-line :data="chartData"></ve-line>

</template>

<script>

export default {

data: function () {

return {

chartData: {

columns: ['日期', '访问用户', '下单用户', '下单率'],

rows: [

{ '日期': '1/1', '访问用户': 1393, '下单用户': 1093, '下单率': 0.32 },

{ '日期': '1/2', '访问用户': 3530, '下单用户': 3230, '下单率': 0.26 },

{ '日期': '1/3', '访问用户': 2923, '下单用户': 2623, '下单率': 0.76 },

{ '日期': '1/4', '访问用户': 1723, '下单用户': 1423, '下单率': 0.49 },

{ '日期': '1/5', '访问用户': 3792, '下单用户': 3492, '下单率': 0.323 },

{ '日期': '1/6', '访问用户': 4593, '下单用户': 4293, '下单率': 0.78 }

]

}

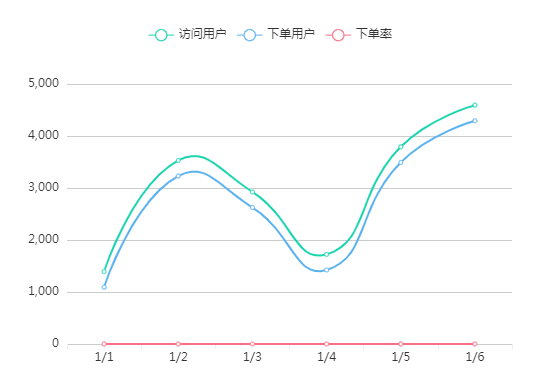
}

}

}

</script>

效果图：



5）首页代码

<template>

  <div style="margin:20px 20px;">

    <el-row :gutter="20" type="flex" class="row-bg" justify="center">

      <!-- 柱状图-->

      <el-col :span="12">

          <div >

           <div class="graphTitile ub  main-center"> 柱状图</div>

          <ve-histogram :data="chartData1"  class="graphTop"></ve-histogram>

           </div>

      </el-col>

      <!-- 折线图-->

      <el-col :span="12">

        <div>

          <div class="graphTitile ub  main-center"> 折线图</div>

          <ve-line :data="chartData2"  class="graphTop"></ve-line>

         </div>

      </el-col>

    </el-row>

    <el-row :gutter="20" type="flex" class="row-bg" justify="center" style="margin-top:20px;">

     <!--饼形图-->

      <el-col :span="12">

       <div>

        <div class="graphTitile ub main-center"> 饼形图</div>

        <ve-pie :data="chartData3" class="graphTop"></ve-pie>

        </div>

      </el-col>

      <!-- 条形图-->

      <el-col :span="12">

        <div>

          <div class="graphTitile ub main-center"> 条形图</div>

       <ve-bar :data="chartData4"  class="graphTop"></ve-bar>

          </div>

      </el-col>

    </el-row>

  </div>

</template>

<script>

export default {

  data () {

      return {

        chartData1: {

          columns: ['日期', '访问用户', '下单用户', '下单率'],

          rows: [

            { '日期': '1/1', '访问用户': 1393, '下单用户': 1093, '下单率': 0.32 },

            { '日期': '1/2', '访问用户': 3530, '下单用户': 3230, '下单率': 0.26 },

            { '日期': '1/3', '访问用户': 2923, '下单用户': 2623, '下单率': 0.76 },

            { '日期': '1/4', '访问用户': 1723, '下单用户': 1423, '下单率': 0.49 },

            { '日期': '1/5', '访问用户': 3792, '下单用户': 3492, '下单率': 0.323 },

            { '日期': '1/6', '访问用户': 4593, '下单用户': 4293, '下单率': 0.78 }

          ]

        },

        chartData2: {

          columns: ['日期', '访问用户', '下单用户', '下单率'],

          rows: [

            { '日期': '1/1', '访问用户': 1393, '下单用户': 1093, '下单率': 0.32 },

            { '日期': '1/2', '访问用户': 3530, '下单用户': 3230, '下单率': 0.26 },

            { '日期': '1/3', '访问用户': 2923, '下单用户': 2623, '下单率': 0.76 },

            { '日期': '1/4', '访问用户': 1723, '下单用户': 1423, '下单率': 0.49 },

            { '日期': '1/5', '访问用户': 3792, '下单用户': 3492, '下单率': 0.323 },

            { '日期': '1/6', '访问用户': 4593, '下单用户': 4293, '下单率': 0.78 }

          ]

        },

         chartData3: {

          columns: ['日期', '访问用户'],

          rows: [

            { '日期': '1/1', '访问用户': 1393 },

            { '日期': '1/2', '访问用户': 3530 },

            { '日期': '1/3', '访问用户': 2923 },

            { '日期': '1/4', '访问用户': 1723 },

            { '日期': '1/5', '访问用户': 3792 },

            { '日期': '1/6', '访问用户': 4593 }

          ]

        },

        chartData4: {

          columns: ['日期', '访问用户', '下单用户', '下单率'],

          rows: [

            { '日期': '1/1', '访问用户': 1393, '下单用户': 1093, '下单率': 0.32 },

            { '日期': '1/2', '访问用户': 3530, '下单用户': 3230, '下单率': 0.26 },

            { '日期': '1/3', '访问用户': 2923, '下单用户': 2623, '下单率': 0.76 },

            { '日期': '1/4', '访问用户': 1723, '下单用户': 1423, '下单率': 0.49 },

            { '日期': '1/5', '访问用户': 3792, '下单用户': 3492, '下单率': 0.323 },

            { '日期': '1/6', '访问用户': 4593, '下单用户': 4293, '下单率': 0.78 }

          ]

        }

      }

    }

};

</script>

<style lang="scss" scoped>

  .graphTitile{

    font-size: 20px;

    font-weight: bolder;

  }

  .graphTop{

    padding-top: 20px;

  }

</style>

效果：



2、头部实现时间动态显示

1）修改Home.vue中的代码

 <div class="user-role">

          <div class="header-wellcome">欢迎您,管理员</div>

          <div class="header-time">2020.7.13 15:55 星期一</div>

        </div>

改为<**div class="header-time"**>{{**homeDate**}}</**div**>

2) 添加Vue的生命周期钩子函数

**var** $vueIndex;（定义在export default 外面，全局变量）

mounted() {  
$vueIndex = **this**;  
**this**.showTime();  
*setInterval*(**function**() {  
$vueIndex.**showTime**();  
 }, 1000);  
}

3)在data()中添加变量

 data() {

    return{

    //显示头部时间

    homeDate: ""

    }

  }

4)在method中编写 showtime（）函数

   //显示时间

showTime() {  
var week = new *Array*(  
"星期日",  
"星期一",  
"星期二",  
"星期三",  
"星期四",  
"星期五",  
"星期六"  
);  
var date = new *Date*();  
var year = date.getFullYear();  
var month = date.getMonth() + 1;  
var day = date.getDate();  
var hour = date.getHours();  
var minutes = date.getMinutes();  
var second = date.getSeconds();  
this. homeDate =year +"."+(month <10 ? "0"+ month : month) +"."+day +" "+" "+ (hour < 10 ? "0" + hour : hour) +":"+(minutes < 10 ? "0" + minutes : minutes) +":"+ (second <10 ? "0"+ second : second) +""+ (week[date.getDay()] || "");  
}

效果：



**第15讲 部门管理列表设计**

1、部门管理

1.1部门搜索表单制作

列表搜索框、搜索按钮、新增按钮实现，使用组件 el-form、el-input、el-row，代码实现如下

<template>

<el-main>

<el-form :model="deptForm" size="mini" label-width="80px">

<el-row>

  <el-col :span="5">

    <el-form-item label="名称">

    <el-input v-model=" deptForm.deptName" placeholder="请输入部门名称"></el-input>

    </el-form-item>

  </el-col>

  <el-button class="searchBtn" type="primary" size="mini" icon="el-icon-search">查询</el-button>

  <el-button class="newBtn" type="primary" size="mini" icon=" el-icon-plus">新增</el-button>

</el-row>

</el-form>

</el-main>

</template>

<script>

    export default {

        data(){

            return{

           deptForm:{

               deptName:""

           }

            }

        }

    }

</script>

<style lang="scss" scoped>

.searchBtn{

    margin-left: 30px;

}

.el-main{

    padding-top: 5px !important;

}

</style>

效果：



**1.2部门列表制作**

使用组件table组件分页组件, 使用固定表头的table组件，只要在el-table元素中定义了height属性，即可实现固定表头的表格，而不需要额外的代码，代码如下：

 <!--表格

    data:数据绑定

    height：只要在el-table元素中定义了height属性，即可实现固定表头的表格，而不需要额外的代码。

    border：表格边框

    prop：字段属性需要跟表格的数据对应

    -->

    <el-table :data="tableData" :height="tableHeight" border style="width: 100%">

      <el-table-column prop="deptNo" label="部门编号"></el-table-column>

      <el-table-column prop="deptName" label="部门名称"></el-table-column>

      <el-table-column prop="enabled" label="是否可用"></el-table-column>

    </el-table>

<script>

export default {

  data() {

    return {

      //搜索框数据绑定

      userForm: {

        deptName: ""

      },

      //表格高度 window.innerHeight窗口文档显示高度

      tableHeight: window.innerHeight,

      //表格数据绑定

       tableData: [

        {

          deptNo: "2016-05-03",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-02",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-04",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-01",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-08",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-06",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-07",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        }

      ]

    };

  },

  //该钩子函数执行时所有的DOM挂载和渲染都已完成，此时在该钩子函数中进行任何DOM操作都不 会有问题

// 在数据变化后要执行的某个操作，而这个操作需要使用随数据改变而改变的DOM结构的时候，

// 这个操作都应该放进Vue.nextTick()的回调函数中

  mounted() {

    this.$nextTick(() => {

      this.tableHeight = window.innerHeight - 210; //后面的50：根据需求空出的高度，自行调整

    });

  }

};

</script>

主要属性：

:data 绑定表格数据

size: 表格尺寸 可选 medium / small / mini stripe 是否为斑马线

height 表格高度

设置表格显示高度

 //该钩子函数执行时所有的DOM挂载和渲染都已完成，此时在该钩子函数中进行任何DOM操作都不 会有问题

// 在数据变化后要执行的某个操作，而这个操作需要使用随数据改变而改变的DOM结构的时候，

// 这个操作都应该放进Vue.nextTick()的回调函数中

  mounted() {

    this.$nextTick(() => {

      this.tableHeight = window.innerHeight - 500; //后面的50：根据需求空出的高度，自行调整

    });

  }

**1.3表格分页组Pagination**

<!-- 分页组件-->

    <el-pagination

      @size-change="handleSizeChange"

      @current-change="handleCurrentChange"

      :current-page.sync="currentPage1"

      :page-size="100"

      layout="total, prev, pager, next"

      :total="1000">

    </el-pagination

methods:{

       handleSizeChange(val) {

        console.log(`每页 ${val} 条`);

      },

      handleCurrentChange(val) {

        console.log(`当前页: ${val}`);

      }

  },

组件属性

size-change: 当page-sizes 改变时触发事件current-change：当页数发生变化时触发事件， current-page：当前是第几页

page-size：页容量，也就是每页多少条数据total:总共有多少条数据,后台返回数据数值注意：

1、静态数据的时候，table的data要从新计算，表格数据才会改变

2、:current-page.sync要加 sync才会自动改变

currentPage的数据,上一页和下一页时，表格数据才会改变

3、表中的prop要跟返回的数据字段对应才能显示

表格的最终代码：

<template>

  <el-main>

    <!-- 搜索框-->

    <el-form :model="userForm" size="mini" label-width="80px">

      <el-row>

        <el-col :span="5">

          <el-form-item label="名称">

            <el-input v-model="userForm.deptName" placeholder="请输入部门名称"></el-input>

          </el-form-item>

        </el-col>

        <el-button class="searchBtn" type="primary" size="mini" icon="el-icon-search">查询</el-button>

        <el-button class="newBtn" type="primary" size="mini" icon="el-icon-search">新增</el-button>

      </el-row>

    </el-form>

    <!--表格

    data:数据绑定

    height：只要在el-table元素中定义了height属性，即可实现固定表头的表格，而不需要额外的代码。

    border：表格边框

    prop：字段属性需要跟表格的数据对应

    -->

   <el-table :data="tableData" :height="tableHeight" border style="width: 100%">

      <el-table-column prop="deptNo" label="部门编号"></el-table-column>

      <el-table-column prop="deptName" label="部门名称"></el-table-column>

      <el-table-column prop="enabled" label="是否可用"></el-table-column>

    </el-table>

    <!-- 分页组件-->

    <el-pagination

      @size-change="handleSizeChange"

      @current-change="handleCurrentChange"

      :current-page.sync="currentPage1"

      :page-size="10"

      layout="total, prev, pager, next"

      :total="this.tableData.length">

    </el-pagination>

  </el-main>

</template>

<script>

export default {

  data() {

    return {

        //当前页

        currentPage:1,

        pageSize:10,

      //搜索框数据绑定

      userForm: {

        userName: ""

      },

      //表格高度 window.innerHeight窗口文档显示高度

      tableHeight: window.innerHeight,

      //表格数据绑定

      tableData: [

        {

          deptNo: "2016-05-03",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-02",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-04",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-01",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-08",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-06",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        },

        {

          deptNo: "2016-05-07",

          deptName: "王小虎",

          enabled: "上海市普陀区金沙江路 1518 弄"

        }

      ]

    };

  },methods:{

       handleSizeChange(val) {

        console.log(`每页 ${val} 条`);

      },

      handleCurrentChange(val) {

        console.log(`当前页: ${val}`);

      }

  },

  //该钩子函数执行时所有的DOM挂载和渲染都已完成，此时在该钩子函数中进行任何DOM操作都不 会有问题

// 在数据变化后要执行的某个操作，而这个操作需要使用随数据改变而改变的DOM结构的时候，

// 这个操作都应该放进Vue.nextTick()的回调函数中

  mounted() {

    this.$nextTick(() => {

      this.tableHeight = window.innerHeight - 600; //后面的50：根据需求空出的高度，自行调整

    });

  }

};

</script>

<style lang="scss" scoped>

.searchBtn {

  margin-left: 30px;

}

.el-main {

  padding-top: 5px !important;

}

</style>

**第16 讲 部门表格操作按钮与新增弹框**

**1.1表格编辑、删除按钮**

只要在表中添加一列即可，slot-scope="scope"：作用域插槽，返回当前行数据

 <el-table-column label="操作" width="160" align="center">

<template slot-scope="scope">

<el-button @click.native.prevent="editRow(scope.$index,tableData)" type="primary" size="mini">编辑

</el-button>

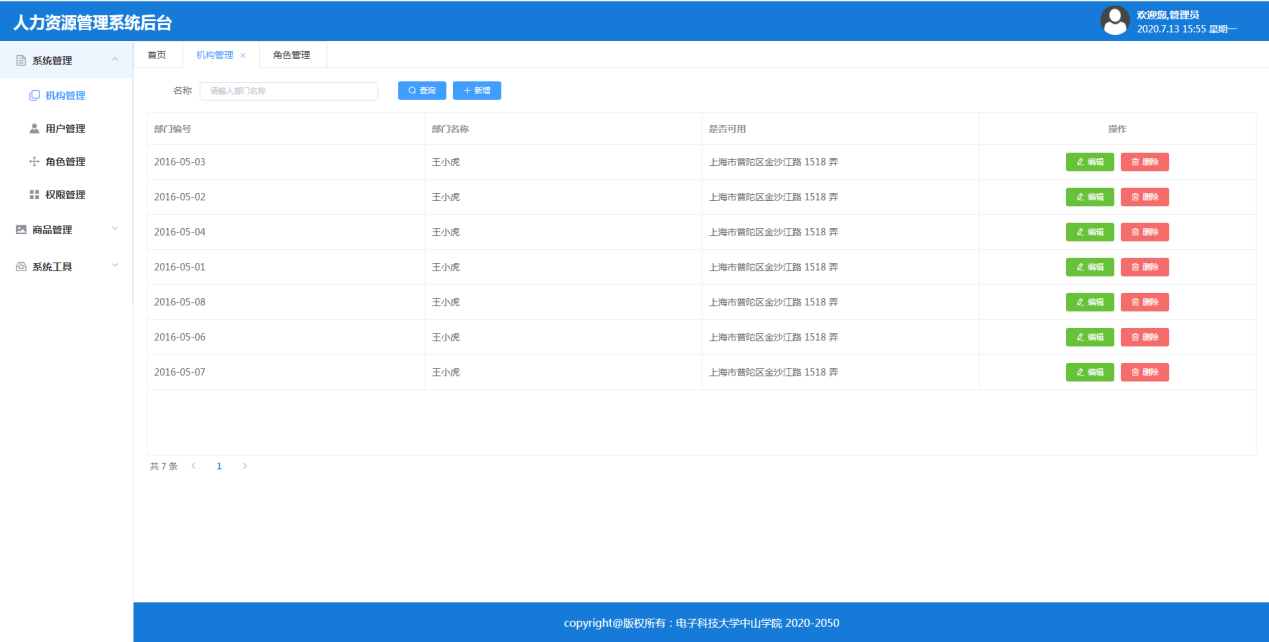
<el-button @click.native.prevent="deleteRow(scope.$index,tableData)" type="danger" size="mini">删除

</el-button>

</template>

</el-table-column>

**效果：**

****

**1.2新增弹出框**

实现原理：控制：visible.sync为true和 false来控制弹框的显示和隐藏

:visible.sync为true时显示

:visible.sync为false时隐藏

1、表单代码：

 <!-- 新增对话框-->

    <el-dialog title="添加部门" :visible.sync="visible" width="30%" center>

      <!-- 对话框内容-->

    <el-form :rules="rules" ref="addDept" :model="addDeptForm" label-width="100px">

        <el-form-item prop="deptNo" label="部门编号">

          <el-input v-model="addDeptForm.deptNo" placeholder="部门编号"></el-input>

        </el-form-item>

        <el-form-item prop="deptName" label="部门名称">

         <el-input v-model="addDeptForm.deptName" placeholder="部门名称"></el-input>

        </el-form-item>

        <el-form-item label="是否可用">

          <el-switch

            v-model="addDeptForm.enabled"

            active-color="#13ce66"

            inactive-color="#ff4949"

            active-value="1"

            inactive-value="0"

          ></el-switch>

        </el-form-item>

      </el-form>

      <!-- 对话框底部-->

      <span slot="footer" class="dialog-footer">

        <el-button @click="visible = false">取 消</el-button>

        <el-button type="primary" @click="confirmBtn">确 定</el-button>

      </span>

    </el-dialog>

2、控制表单弹出

data()方法中添加如下变量：

 //控制对话框的显示和隐藏，true为显示，false为隐藏

        visible:false,

 //绑定对话框数据

      addDeptForm: {

        deptNo: "",

        deptName: "",

        enabled: ""

      },

在methods中添加如下方法：

 //弹出对话框

    addDeptShow() {

      //调用对话框表单清空方法

      this. resetForm("addDept");

      this.visible = true;

    },

     //编辑

    editRow(row) {

      console.log(row);

    },

    //删除

    deleteRow(row) {

      console.log(row);

    },

      //清空表单数据

      resetForm(formName){

        if(this.$refs[formName]){

         this.$refs[formName].resetFields();

        }

  }

 //对话框确定

    confirmBtn() {

          \_this.visible = false;

    },

效果



3、表单验证

增加验证规则

//数据验证

      rules: {

        deptNo: [

          {

            required: true,

            trigger: "change",

            message: "请输入编号"

          }

        ],

        deptName: [

          {

            required: true,

            trigger: "change",

            message: "请输入部门名"

          }

        ]

      },

添加验证方法：

//对话框确定

    confirmBtn() {

      let \_this = this;

      \_this.$refs.addDept.validate(valid => {

        if (valid) {

          \_this.visible = false;

        }

      });

    },

效果：

****