

第2章 品牌管理前端与图片上传

学习目标

- 掌握ElementUI常用组件的使用
- 掌握ES6常用语法的使用
- 完成品牌管理前端代码
- 掌握图片上传代码的编写
- 掌握阿里云OSS的使用

1. ElementUI介绍

1.1 什么是ElementUI

Element，“饿了么”出品的一套为开发者、设计师和产品经理准备的基于 Vue 2.0 的桌面端组件库。

1.2 ElementUI 常用组件

详见elementUI官网 <http://element-cn.eleme.io/2.0/#/zh-CN>

2. ES6

2.1 什么是ES6

编程语言JavaScript是ECMAScript的实现和扩展。ECMAScript是由ECMA（一个类似W3C的标准组织）参与进行标准化的语法规则。ECMAScript定义了：

[语言语法](#) – 语法解析规则、关键字、语句、声明、运算符等。

[类型](#) – 布尔型、数字、字符串、对象等。

[原型和继承](#)

内建对象和函数的[标准库](#) – [JSON](#)、[Math](#)、[数组方法](#)、[对象自省方法](#)等。

ECMAScript标准不定义HTML或CSS的相关功能，也不定义类似DOM（文档对象模型）的[Web API](#)，这些都在独立的标准中进行定义。ECMAScript涵盖了各种环境中JS的使用场景，无论是浏览器环境还是类似[node.js](#)的非浏览器环境。

ECMAScript标准的历史版本分别是1、2、3、5。

那么为什么没有第4版？其实，在过去确实曾计划发布提出巨量新特性的第4版，但最终却因想法太过激进而惨遭废除（这一版标准中曾经有一个极其复杂的支持泛型和类型推断的内建静态类型系统）。

ES4饱受争议，当标准委员会最终停止开发ES4时，其成员同意发布一个相对谦和的ES5版本，随后继续制定一些更具实质性的新特性。这一明确的协商协议最终命名为“Harmony”，因此，ES5规范中包含这样两句话

ECMAScript是一门充满活力的语言，并在不断进化中。

未来版本的规范中将持续进行重要的技术改进

2009年发布的改进版本ES5，引入了[Object.create\(\)](#)、[Object.defineProperty\(\)](#)、[getters](#)和[setters](#)、[严格模式](#)以及[JSON](#)对象。

ECMAScript 6.0（以下简称ES6）是JavaScript语言的下一代标准，2015年6月正式发布。它的目标，是使得JavaScript语言可以用来编写复杂的大型应用程序，成为企业级开发语言。

2.2 常用语法新特性

ES6提供了很多语法新特性，下面我们讲解的是我们《青橙》项目中使用到的语法新特性。

2.2.1 变量声明let

我们都是知道在ES6以前，var关键字声明变量。无论声明在何处，都会被视为声明在函数的最顶部(不在函数内即在全局作用域的最顶部)。这就是函数变量提升例如

```
function aa() {  
  if(true) {  
    var test = 'hello man'  
  }  
  alert(test)  
}
```

以上的代码实际上是:

```
function aa() {  
  var test;  
  if(true) {  
    test = 'hello man'  
  }  
  alert(test)  
}
```

接下来ES6主角登场:

我们通常用let和const来声明，let表示变量、const表示常量。let和const都是块级作用域。怎么理解这个块级作用域？在一个函数内部，在一个代码块内部。看以下代码

```
function aa() {  
  if(bool) {  
    let test = 'hello man'  
  } else {  
    //test 在此处访问不到  
    console.log(test)  
  }  
}
```

2.2.2 常量声明

const 用于声明常量，看以下代码

```
const name = 'lux'  
name = 'joe' //再次赋值此时会报错
```

2.2.3 模板字符串

es6模板字符简直是开发者的福音啊，解决了ES5在字符串功能上的痛点。

第一个用途，基本的字符串格式化。将表达式嵌入字符串中进行拼接。用`${}`来界定。

```
//es5
var name = 'lux'
console.log('hello' + name)
//es6
const name = 'lux'
console.log(`hello ${name}`) //hello lux
```

第二个用途，在ES5时我们通过反斜杠`()`来做多行字符串或者字符串一行行拼接。ES6反引号```直接搞定。

```
// es5
var msg = "Hi \
man!"
// es6
const template = `

<span>hello world</span>
</div>`


```

2.2.4 箭头函数

ES6很有意思的一部分就是函数的快捷写法。也就是箭头函数。

箭头函数最直观的三个特点。

1不需要`function`关键字来创建函数

2省略`return`关键字

3继承当前上下文的 `this` 关键字

看下面代码（ES6）

```
(response,message) => {
  .....
}
```

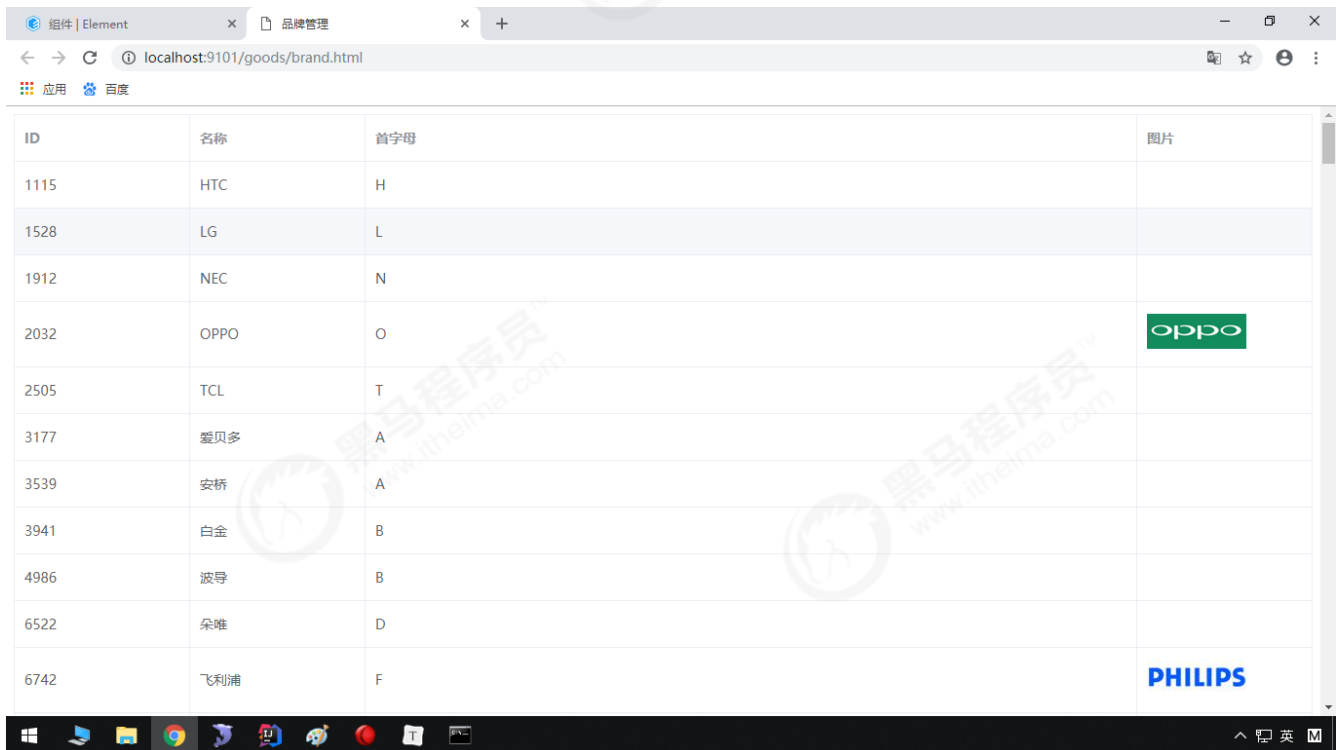
相当于ES5代码



```
function(response,message){  
    .....  
}
```

3. 青橙管理后台-品牌管理前端

3.1 列表展示

需求：如下图所示，使用表格控件显示全部的品牌数据，不用分页与条件查询。



| ID | 名称 | 首字母 | 图片 |
|------|------|-----|---|
| 1115 | HTC | H | |
| 1528 | LG | L | |
| 1912 | NEC | N | |
| 2032 | OPPO | O |  |
| 2505 | TCL | T | |
| 3177 | 爱贝多 | A | |
| 3539 | 安桥 | A | |
| 3941 | 白金 | B | |
| 4986 | 波导 | B | |
| 6522 | 朵唯 | D | |
| 6742 | 飞利浦 | F |  |

代码实现：

(1) 拷贝资源：将资料\前端相关\js 和css文件夹 拷贝到qingcheng_web_manager工程的webapp下，在qingcheng_web_manager工程的webapp下创建goods文件夹, 在goods文件夹下创建brand.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>品牌管理</title>
  <link rel="stylesheet" href="../css/elementui.css">
</head>
<body>
<script src="../js/vue.js"></script>
<script src="../js/elementui.js"></script>
<script src="../js/axios.js"></script>
</body>
</html>
```

(2) 编写页面js:

```
<script>
  new Vue({
    el: '#app',
    data() {
      return {
        tableData: []
      }
    },
    created() {
      axios.get('/brand/findAll.do').then(response=> {
        this.tableData = response.data;
      });
    }
  })
</script>
```

(3) 使用表格控件展现数据

```

<el-table :data="tableData" border style="width: 100%">
  <el-table-column
    prop="id"
    label="ID"
    width="180">
  </el-table-column>
  <el-table-column
    prop="name"
    label="名称"
    width="180">
  </el-table-column>
  <el-table-column
    prop="letter"
    label="首字母">
  </el-table-column>
  <el-table-column
    label="图片"
    width="180">
    <template slot-scope="scope">
      
    </template>
  </el-table-column>
</el-table>

```

3.2 分页列表展示

(1) 页面的表格下方添加分页组件

```

<el-pagination
  @size-change="fetchData"
  @current-change="fetchData"
  :current-page.sync="currentPage"
  :page-sizes="[10, 20, 30, 40]"
  :page-size="size"
  layout="total, sizes, prev, pager, next, jumper"
  :total="total">
</el-pagination>

```

size-change 页大小更改时触发的事件

current-change 当前页更改时触发的事件

current-page 当前页变量

page-sizes 每页记录数选项列表

page-size 当前选择的每页记录数

layout布局列表 total 总记录数 sizes 每页记录数选择框 prev 上一页 next下一页

pager 页码 jumper 为页码跳转

total 为总记录数

(2) 修改js脚本

```
new Vue({
  el: '#app',
  data(){
    return {
      tableData: [],
      currentPage: 1,
      total: 10,
      size: 10
    }
  },
  created(){
    this.fetchData();
  },
  methods:{
    fetchData (){
      axios.get(
        `/brand/findPage.do?page=${this.currentPage}&size=${this.size}`).then(
        response=> {
          this.tableData = response.data.rows;
          this.total = response.data.total;
        });
    }
  }
})
```

3.3 条件查询

实现思路：表格上方添加表单，表单内的文本框等控件绑定查询对象，点击查询按钮触发方法，将查询对象传递给后端进行查询。

代码实现：

(1) 在表格上方添加表单

```
<!-- 查询表单 -->
<el-form :inline="true" >
  <el-form-item label="品牌名称">
    <el-input v-model="searchMap.name" placeholder="品牌名称">
  </el-input>
  </el-form-item>
  <el-form-item label="品牌的首字母">
    <el-input v-model="searchMap.letter" placeholder="品牌的首字母">
  </el-input>
  </el-form-item>
  <el-button type="primary" @click="fetchData()">查询</el-button>
</el-form>
```

(2) 修改 js 代码，data 新增 searchMap 属性

```
searchMap: {}
```

更改 axios 调用的代码，改为 post 提交，传递 searchMap 对象

```
axios.post(`/brand/findPage.do?
page=${this.currentPage}&size=${this.size}`, this.searchMap)
```

3.4 新增品牌

需求：页面上添加“新增”按钮，点击新增按钮弹出窗口，窗口中包含表单和保存、关闭按钮。填写数据后点击保存按钮关闭窗口，刷新列表数据。

(1) 窗口的弹出与关闭：data 添加属性，用于控制窗口显示

```
formVisible: false
```

添加弹出窗口

```

<!--弹出窗口-->
<el-dialog title="编辑" :visible.sync="formVisible" >
  <el-button @click="formVisible = false" >关闭</el-button>
</el-dialog>

```

添加按钮，控制窗口的弹出

(2) data添加属性pojo 用于保存实体

```
pojo: {}
```

在窗口添加表单

```

<el-form label-width="80px">
  <el-form-item label="品牌名称">
    <el-input placeholder="品牌名称" v-model="pojo.name">
  </el-input>
  </el-form-item>
  <el-form-item label="品牌首字母">
    <el-input placeholder="品牌首字母" v-model="pojo.letter">
  </el-input>
  </el-form-item>
  <el-form-item label="品牌图片">
    <el-input placeholder="品牌图片" v-model="pojo.image">
  </el-input>
  </el-form-item>
  <el-form-item label="排序">
    <el-input placeholder="排序" v-model="pojo.seq"></el-
input>
  </el-form-item>
  <el-form-item>
    <el-button @click="save()">保存</el-button>
    <el-button @click="formVisible = false">关闭</el-button>
  </el-form-item>
</el-form>

```

(3) 添加save方法

```

save (){
    axios.post('/brand/add.do',this.pojo).then(response => {
        this.fetchData ();//刷新列表
        this.formVisible = false ;//关闭窗口
    });
}

```

3.5 修改品牌

需求：在表格中添加操作列，操作列中有修改按钮，点击修改按钮弹出窗口加载数据。用户修改数据后点击保存，保存后关闭窗口刷新列表。

(1) 在表格中新增模板列

```

<el-table-column
    label="操作" >
    <template slot-scope="scope">
        <el-button @click="edit(scope.row.id)" type="text" size="small">
修改</el-button>
    </template>
</el-table-column>

```

(2) 增加edit方法，用于加载数据

```

edit (id){
    this.formVisible = true // 打开窗口
    // 调用查询
    axios.get(`/brand/findById.do?id=${id}`).then(response => {
        this.pojo = response.data;
    })
}

```

(3) 修改save方法

```

save (){

axios.post(`/brand/${this.pojo.id==null?'add':'update'}.do`,this.pojo).then(response => {
    this.fetchData (); //刷新列表
    this.formVisible = false ;//关闭窗口
});
}

```

3.6 删除品牌

需求: 表格操作列增加“删除”按钮，点击删除，弹出提示，确定后删除数据刷新列表。

(1) 操作列中新增“删除”按钮

```

<el-button @click="dele(scope.row.id)" type="text" size="small">删除</el-button>

```

(2) 新增dele方法

```

dele (id){
    this.$confirm('确定要删除此记录吗?', '提示', {
        confirmButtonText: '确定',
        cancelButtonText: '取消',
        type: 'warning'
    }).then( () => {
        axios.get(`/brand/delete.do?id=${id}`).then(response => {
            if(response.data.code==0){
                this.fetchData (); //刷新列表
            }else{
                this.$alert(response.data.message)
            }
        })
    })
}

```

4. 图片上传

4.1 图片上传组件

(1) 在弹出窗口中放置图片上传组件

```
<el-upload
  class="avatar-uploader"
  action="/upload/native.do"
  :show-file-list="false"
  :on-success="handleAvatarSuccess"
  :before-upload="beforeAvatarUpload">
  
  <i v-else class="el-icon-plus avatar-uploader-icon"></i>
</el-upload>
```

(2) 修改脚本，data 增加属性

```
imageUrl: ''
```

增加方法

```
handleAvatarSuccess(res, file) {
  this.imageUrl = file.response;
},
beforeAvatarUpload(file) {
  const isJPG = file.type === 'image/jpeg';
  const isLt2M = file.size / 1024 / 1024 < 2;
  if (!isJPG) {
    this.$message.error('上传头像图片只能是 JPG 格式!');
  }
  if (!isLt2M) {
    this.$message.error('上传头像图片大小不能超过 2MB!');
  }
  return isJPG && isLt2M;
}
```

(3) 修改edit方法和save方法

```
//save方法第一句添加
this.pojo.image= this.imageUrl;
```

//edit方法回调时添加

this.imageUrl=this.pojo.image //显示图片

4.2 SpringMVC接收图片上传

SpringMVC 中，文件的上传，是通过 MultipartResolver 实现的。所以，如果要实现文件的上传，只要在 spring-mvc.xml 中注册相应的 MultipartResolver 即可。

MultipartResolver 的实现类有两个：

1. CommonsMultipartResolver
2. StandardServletMultipartResolver

两个的区别：

1. 第一个需要使用 Apache 的 commons-fileupload 等 jar 包支持，但它能在比较旧的 servlet 版本中使用。
2. 第二个不需要第三方 jar 包支持，它使用 servlet 内置的上传功能，但是只能在 Servlet 3 以上的版本使用。

(1) qingcheng_common_web增加配置

```
<!-- 多部分文件上传 -->
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver"
">
    <property name="maxUploadSize" value="104857600" />
    <property name="defaultEncoding" value="UTF-8"></property>
</bean>
```

maxUploadSize: 设置允许上传的最大文件大小，以字节为单位计算。当设为-1时表示无限制，默认是-1。

defaultEncoding: 表示用来解析request请求的默认编码格式，当没有指定的时候根据Servlet规范会使用默认值ISO-8859-1。当request自己指明了它的编码格式的时候就会忽略这里指定的defaultEncoding。

(2) qingcheng_web_manager工程新建controller

```

@RestController
@RequestMapping("/upload")
public class UploadController {

    @Autowired
    private HttpServletRequest request;

    @PostMapping("/native")
    public String nativeUpload(@RequestParam("file") MultipartFile file)
    {
        String
path=request.getSession().getServletContext().getRealPath("img");
        String filePath = path + "/" + file.getOriginalFilename();
        File desFile = new File(filePath);
        if(!desFile.getParentFile().exists()){
            desFile.mkdirs();
        }
        try {
            file.transferTo(desFile);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return "http://localhost:9101/img/"+file.getOriginalFilename();
    }
}

```

MultipartFile是spring类型，代表HTML中form data方式上传的文件，包含二进制数据+文件名称

5. 云存储解决方案-阿里云OSS

5.1 什么是阿里云OSS

阿里云对象存储服务（Object Storage Service，简称OSS）为您提供基于网络的数据存取服务。使用OSS，您可以通过网络随时存储和调用包括文本、图片、音频和视频等在内的各种非结构化数据文件。

阿里云OSS将数据文件以对象（object）的形式上传到存储空间（bucket）中。

您可以进行以下操作：

- 创建一个或者多个存储空间，向每个存储空间中添加一个或多个文件。
- 通过获取已上传文件的地址进行文件的分享和下载。
- 通过修改存储空间或文件的属性或元信息来设置相应的访问权限。
- 在阿里云管理控制台执行基本和高级OSS任务。
- 使用阿里云开发工具包或直接在应用程序中进行RESTful API调用执行基本和高级OSS任务

5.2 OSS开通

(1) 打开<https://www.aliyun.com/>，申请阿里云账号并完成实名认证。

(2) 充值

(3) 开通OSS: 登录阿里云官网。将鼠标移至产品找到并单击对象存储OSS打开OSS产品详情页面。在OSS产品详情页中的单击立即开通。开通服务后，在OSS产品详情页面单击管理控制台直接进入OSS管理控制台界面。您也可以单击位于官网首页右上方菜单栏的控制台，进入阿里云管理控制台首页，然后单击左侧的对象存储OSS菜单进入OSS管理控制台界面。



(4) 创建存储空间

新建Bucket，命名为qingcheng，读写权限为公共读

新建 Bucket

命名：

Bucket 命名规范：

- 只能包含小写字母，数字和短横线
- 必须以小写字母和数字开头和结尾
- 长度限制在 3-63 之间

所属地域：

华北 2

相同地域内的产品内网可以互通；订购后不支持更换地域，请谨慎选择

EndPoint：

oss-cn-beijing.aliyuncs.com

存储类型：

标准存储

- 标准存储类型：高可靠、高可用、高性能，数据会经常被访问到
- 低频访问类型：数据长期存储、较少访问，存储单价低于标准类型
- 归档存储类型：数据长期存储、基本不访问，存储单价低于低频访问型

读写权限：

私有

- 私有：对 Object 的所有访问操作需要进行身份验证
- 公共读：对 Object 写操作需要进行身份验证；可以对 Object 进行匿名读
- 公共读写：所有人都可以对 Object 进行读写操作

5.3 OSS快速入门

(1) 创建测试工程，引入依赖

```
<dependency>
  <groupId>com.aliyun.oss</groupId>
  <artifactId>aliyun-sdk-oss</artifactId>
  <version>2.8.2</version>
</dependency>
```

(2) 新建类和main方法

```
String endpoint = "http://oss-cn-beijing.aliyuncs.com";
// 云账号AccessKey有所有API访问权限，建议遵循阿里云安全最佳实践，创建
// 并使用RAM子账号进行API访问或日常运维，请登录
String accessKeyId = "LTAIdmst7udCr3vB";
String accessKeySecret = "嘻嘻不告诉你: ) ";
String bucketName = "qing-cheng";
// 创建OSSClient实例
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);
// 上传文件流
InputStream inputStream = null;
try {
    inputStream = new FileInputStream("d:/test.html");
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
ossClient.putObject(bucketName, "test.html", inputStream);
// 关闭client
ossClient.shutdown();
```

5.4 青橙管理后台-OSS图片上传

(1) qingcheng_common_web添加依赖

```
<dependency>
    <groupId>com.aliyun.oss</groupId>
    <artifactId>aliyun-sdk-oss</artifactId>
    <version>2.8.2</version>
</dependency>
```

(2) qingcheng_common_web添加配置

```

<!--阿里云OSS-->
<bean id="ossClient" class="com.aliyun.oss.OSSClient">
    <constructor-arg index="0" value="oss-cn-beijing.aliyuncs.com">
</constructor-arg>
    <constructor-arg index="1" value="LTAI4mbt7uyCr3tB">
</constructor-arg>
    <constructor-arg index="2"
value="Q8Yje9xS4z4Z2BTUhuPaT1bxz8qiDK"></constructor-arg>
</bean>

```

(3) UploadController新增方法

```

@Autowired
private OSSClient ossClient;

@PostMapping("/oss")
public String ossUpload(@RequestParam("file") MultipartFile
file,String folder){
    String bucketName="qingchengdianshang";
    String fileName= folder+ "/" + UUID.randomUUID()+
file.getOriginalFilename();
    try {

ossClient.putObject(bucketName,fileName,file.getInputStream());
        } catch (IOException e) {
            e.printStackTrace();
        }
        return "https://"+bucketName +".oss-cn-
beijing.aliyuncs.com/"+fileName;
    }
}

```

(4) 修改brand.html页面的上传组件

```

<el-upload
.....
action="/upload/oss.do?folder=brand"
.....
>
</el-upload>

```

