[EUIADMIN]

[开发参考手册]

2020-10-12

作者：RADISH

捐赠我们

我只是一个爱好者，希望能真正为广大开发者起到一定的助力作用；为了开源项目能够走得更远希望能捐赠我们，一分两分也是爱。

前言

EuiAdmin 是什么

EuiAdmin 是基于 vue+element-ui 和其他 JavaScript 库共同构建的一个专门为后端管理的快速开发框架，通过前期的基础优化和 JavaScript 库的安装你可以快速完成开发。

EuiAdmin 存在的意义

EuiAdmin 是为了能够更快的架构一个网站后台管理模板，让开发人员将更多精力放到功能实现上减少对框架外观的关注，大大减少个人网站后台管理的开发时间。通过 EuiAdmin 提供的基础美化框架，你可以通过 EuiAdmin 的颜色和宽度、高度自定义功能实现符合自己风格的后台模板。

然而更重要的是 EuiAdmin 是一个开源的后台开发框架，这意味着你可以以近乎 0 成本使用一个基本配置好的后台框架，减少开发人员在 JavaScript 安装浪费时间。

EuiAdmin 可以为您做什么

截止目前 EuiAdmin 已经安装支持 tinymce、axios、vue-cookies 和 js-xlsx 等 JavaScript 库通过简单的调用你可以快速完成开发。

1、 EuiAdmin 支持 tinymce 富文本编辑器，你如果使用

EuiAdmin 那么你可以直接获得一个富文本编辑器。

2、 集成 axios 并支持跨域配置和 baseUrl 的配置，你可以快速调用。

3、 通过 vue-cookies 的加持，你要快速完成 cookie 的设置、获取和删除。

4、 Js-xlsx 实现 json 数据的导入和导出。

同时通过多种多样的表格和应用页面支持，你可以直接拿起构建逻辑即可完成开发。

目前存在的问题

在运行和打包时偶触发 src/assets/china.js 或 src/assets/langs/zh_CN.js 找不到发送错误的问题；解决方法关闭或终止当前代码，重新执行代码即可（如失败可多次重试）。

怎样运行 EuiAdmin

构建运行环境

请下载安装 node.js，然后打开命令提示符输入‘npm install -g @vue/cli’；按下回车键等待加载完毕，此时运行环境便安装完成。

获取 EuiAdmin 并运行

获取：请前往 http://euiadmin.com 获取源码，然后解压；在解压文件根目录按 shift 不放，然后按下鼠标右键选择“在此执行命令

提示符”，输入“npm run serve”（运行），“npm run build”（打包）便可完成运行;（如错误请重试）。

集成 JavaScript 库情况

截止 EuiAdmin0.5.1 版本，当前集成 element-ui、tinymce、js-xlsx、vue-aplayer、vue-cookies、Echarts、axios 共计 8 个 JavaScript 库。

在 EuiAdmin 中怎样调用 vue-cookies

相关值说明:

key: cookie 关键 key 值。

value: key 值绑定的变量（值）。

Vue-cookies 怎样设置和获取 cookie

简单设置和获取

设置:

```
this.$cookies.set(key,value)
```

获取:

```
this.$cookies.get(key)
```

删除已经设置的 cookie

```
this.$cookies.remove(key)
```

判断 cookie 是否存在

this.$cookies.isKey(key)//返回 false 或者 true

设置 cookie 设置过期时间:

时间表达形式

内容	表达式
年	Y
月	m
日	d
时	h
分	min
秒	s

例子:

this.$cookies.set(key,value)//不输入时间默认为一天。

this.$cookies.set(key,value,"60s")//60 秒后过期。

this.$cookies.set(key,value,"20min")//20 分钟后过期。

this.$cookies.set(key,value,"20h")//20 小时后过期。

this.$cookies.set(key,value,"5d")//5 天后过期。

this.$cookies.set(key,value,"2m")//2 月后过期。

this.$cookies.set(key,value,"2y")//2 年后过期。

注意时间可以支持时间表达式，也可以指定过期时间。

设置 cookie 永不过期:

this.$cookies.set(key,value,Infinity)//永不过期。

this.$cookies.set(key,value,-1)//永不过期。

cookie 储存 json 数据:

```
var user = { id:1, name:'Journal',session:'122' };
this.$cookies.set('user',user);//储存 json 数据
```

获取 cookie:

this.$cookies.get(key)//获取该 key 的值。

this.$cookies.keys()//获取所有设置的 cookie; 返回值是一个数组。

this.$cookies.get('user').name//获取 json 数组中 name 值

删除 cookie:

this.$cookies.remove(key)//删除指定 key 的 cookie

this.$cookies.keys().forEach(cookie=>this.$cookies.remove(cookie))//删除所有 cookie。

参考链接: https://www.npmjs.com/package/vue-cookies

在 EuiAdmin 中怎样调用 echarts

参考代码:

```
<template>
  <div>
    <div slot="header" class="clearfix">
      <span>基础折线图</span>
    </div>
    <div id="charts" style="width: 100%; height: 300px"></div>
```

```
<!-- 高度和宽度为必须，不能为空 -->
  </div>
</template>
<script>
import "echarts/theme/macarons.js";
export default {
  data() {
    return {};
  },
  methods: {
    draw_charts() {
      let myChart = this.$echarts.init(
        document.getElementById("charts"),
        "macarons"
      );
    //    id 与 html 中 id 名称一致，macarons 是主体名称
      myChart.setOption({
        //    这里开始是 charts 官方例子中的 option 内容，可以去官方复制
        xAxis: {
          type: "category",
          data: ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"],
        },
        yAxis: {
          type: "value",
        },
        series: [
          {
            data: [820, 932, 901, 934, 1290, 1330, 1320],
            type: "line",
          },
        ],
        // 复制内容结束
      });
    },
    },
  },
  mounted() {
    this.draw_charts();//加载前必须先执行函数
  },
};
</script>
```

注意：

1、 id 与 getElementById 必须一致。

2、 在页面执行前先执行 Echarts 绘画函数。

在 EuiAdmin 中怎样调用 axios

在官方例子中使用 axios 进行开头，在 EuiAdmin 中使用 this.$axios 替换官方例子中的 axios 即可，其他部分可以直接使用官方例子。

例如，发送一个 post 请求:

```
this.$axios.post('/user', {
    firstName: 'Fred',
    lastName: 'Flintstone'
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

通过上述代码便可简单实现 axios 发送一个 post 请求。

关于使用 axios 你可能要了解 qs

什么是 qs:

qs 是 npm 仓库所管理的包，主要作用是将 axios 需要传递的参数格式化成为 url 形式，让后台服务端（java php）等能够正常接收传递的参数。

有时候我们直接使用 axios 的 data 传递未经 qs 格式化的数据往往会存在后台无法正常接收前端传递的值。

qs 常用的两个函数:

qs.stringify():将对象数列和数组格式化成为 url 格式。

qs.parse():将 url 格式，格式化为对象或数组形式。

在 EuiAdmin 中怎样调用 qs。

两个例子:

this.$qs.stringify():将对象数列和数组格式化成为 url 格式。

this.$qs.parse():将 url 格式，格式化为对象或数组形式。

在 EuiAdmin 中想要调用 qs 库，只需要将 qs 替换为 this.$qs 即可完成调用。

EuiAdmin 怎样注册全局公共组件

什么是全局组件:

常常我们使用一般单一的组件的时候，我们只需要在需要使用的 vue 文件中进行 import 引用，然后使用 components 进行组件注册然后进行使用即可。

而全局组件则是在 main.js 组件中进行注册，然后在需要的地方直接使用即可，无需再该页面进行引用和注册，减少开发代码量，保障项目的简洁程度。

示例（以 EuiAdmin 为例）：

新建一个 js 文件位于“src/plugins/Component.js”。

文件内容代码：

```
export default {
 AsideSpace:resolve=>{require(['@/components/system/AsideSpace'],resolve)},//这是需要引入的组件
 HeadSpace:resolve=>{require(['@/components/system/HeadSpace'],resolve)},//这是需要引入的组件
 EditorSpace:resolve=>{require(['@/components/module/EditorSpace'],resolve)},//这是需要引入的组件
}
```

然后在 main.js 文件中写入如下内容：

```
import components from './plugins/Component'//加载公共组件
Object.keys(components).forEach((key) => {
  var name = key.replace(/(\w)/, (v) => v.toUpperCase())
  Vue.component('v'+name, components[key])
})
```

怎样引用公共组件：

例如要引用 AsideSpace 公共组件模块，只需要在需要在要引用的地方写入“<vAsideSpace/>”即可完成完成公共组件。

什么是组件传参

组件传参是指在组件之间不依托 router 等其他 JavaScript 库实现组件间的传值，让 vue 开发更加简单；传参形式一般有四种：

父组件向子组件传递参数。

子组件向父组件传递参数。

兄弟共同注册于父组件，兄弟组件之间传递参数。

兄弟组件不共同注册于父组件，兄弟组件间传参。

例如：

父组件向子组件传递参数：

父组件：

```
<template>
  <child name="radish"/>
</template>

<script>
  import child from "./child-component"
  export default {
    name: "parent-component",
    components:{
      child
    }
  }
</script>
```

子组件接收：

```
<template>
  <div>
    {{name}}
  </div>
</template>
<script>
  export default {
    name: "child-component",
    props: [“name”]
    }
  }
</script>
```

子组件向父组件传递参数:

传参子组件（需要通过$emit 实现传参）：

```
<template>
  <div>
 <button @click="up">向父组件传值</button>
  </div>

</template>
```

```
<script>
  export default {
    methods: {
      up() {
        this.$emit("upload",'radish')
      }
    }
  }
</script>
```

父组件:

```
<template>
  <div>
    <child name="小明" @ upload ="getValueFromChild"/>
  </div>
</template>

<script>
  // 引入子组件
  import child from "./child-component"

  export default {
    name: "parent-component",
    components: {
      child
    },
    methods: {
      getValueFromChild(value) {
        console.log(value)
      }
    }
  }
</script>
```

兄弟共同注册于父组件，兄弟组件之间传递参数。

实现原理，将子组件传递到父组件，赋值给指定的值，然后将该值传递给相应的子组件即可完成传参，一般不常用。

兄弟组件不共同注册于父组件，兄弟组件间传参。

以 EuiAdmin 为例:

先新建一个新的 js 文件（名称随便）EuiAdmin 为 Pass.js 位于“src/plugins/Pass.js”。

需要传递的子组件:

```
<template>
  <div>
<I @click="change_aside_state" class=" el-icon-s-fold"/>
</div>
</template>
<script>
import Pass from "@/plugins/Pass.js";
export default {
methods: {
      change_aside_state() {
         this.aside_state = !this.aside_state;
         Pass.$emit("aside_state", this.aside_state);
      },
      }
}
```

需要接收传递值的子组件:

```
<script>
import Pass from "@/plugins/Pass.js";
export default {
created() {
      Pass.$on("aside_state", (val) => {
         console.log(val)
      });
  },
}
}
```

以上便是四种传递形式。

EuiAdmin 实现顶栏按钮实现左侧菜单栏的开关状态控制。

顶部组件位于: ‘src/components/system/ HeadSpace.vue。

左侧组件位于: ‘src/components/system/ AsideSpace.vue’。

实现原理:

点击顶部图标触发传递参数函数，通过 pass 实现数据的传递，在接收组件的生命周期函数中接收通过 pass 函数传递的参数，通过参数实现左侧状态切换控制。

EuiAdmin 怎样实现框架颜色控制

EuiAdmin 的框架颜色控制是通过 cookies 的设置和取出实现颜色的储存和取出，从而实现整体框架颜色的控制。

EuiAdmin 中怎样调用富文本编辑器

EuiAdmin 的富文本编辑器是通过集成 tinymce 富文本编辑器实现的，为了能够好的效果，未使用 cdn 的形式进行引用，但使用了懒加载组件的形式给予加载，所有在网站首次使用富文本编辑器时会出加载时间过长的情况，需要耐心等待。

EuiAdmin 的富文本编辑器可以支持表格、文字颜色、格式调整、图片自动插入上传等复杂功能，EuiAdmin 富文本编辑器源码位于”src/components/module/ EditorSpace.vue 文件中，如需更改和查看可以打开该源码查看和修改，注释已然全面。

EuiAdmin 怎样使用富文本编辑器; EuiAdmin 富文本编辑器已经实现全局注册，在需要使用的地方直接 “<vEditorSpace v-model="editor" />” 便可完成使用，model 返回的内容是 html 格式的内容。

EuiAdmin 动画实现和运用

EuiAdmin 的动画是基于 Animate.css 实现的可以完美支持 Animate 的 96 种动画。通过 euiadmin 的官方支持，你可以快速实现对动画重复的次数、动画的速度和延迟时间重要参数进行设置。你可以通过前往 Animate.css（https://animate.style/#attention_seekers）复制 class 名称点击 euiadmin 的添加，然后点击新增的动画名；然后复制动画 class 名括号中的内容，放到需要该动画的 div、p 等元素中即可实现动画效果。

EuiAdmin 怎样集成 js-xlsx

前言:

在以前需要将数据导出为 excel 文件时，往往需要后端提供支持，后端导出后以文件流的形式进行下载。但也带了一些问题，如:

1、 代码量复杂，一般前端难以完成。

2、 实现数据导出往往会大量消耗服务器的性能，对于网站而言是非常不利的。

随着前端的不断发展，现在只需要将数据传到前端，让前端实现数据导出为 excel。

JavaScript 组件库：js-xlsx。

安装：

将命令提示符切换到项目根目录执行安装代码（npm install js-xlsx --save），待进度条完成后便可实现安装。

Js-xlsx 导出 json 数据到表格

导出原理：

将 json 数据的 key 替换为需要定义的名称，然后将 json 数据进行传递即可完成输出。源码（位于:'src/components/module/excel/OutEXcel.vue'）:

```
<template>
<div>
<el-button
slot="reference"
type="danger"
size="small"
icon="el-icon-finished"
:disabled="selection_button_state"
```

```
@click="dialogVisible = true"
>{{ selection_button_title }}</el-button
>
<el-dialog title="文件名和 sheet 不能为空" :visible.sync="dialogVisible">
<div>
<el-form ref="form" :model="form" label-width="100px">
<el-form-item label="导出文件名">
<el-input v-model="form.file_name"></el-input>
</el-form-item>
<el-form-item label="sheet 名称">
<el-input v-model="form.sheet_name"></el-input>
</el-form-item>
</el-form>
</div>
<span slot="footer" class="dialog-footer">
<el-button size="small" type="warning" @click="dialogVisible = false"
>取 消</el-button
>
<el-button
size="small"
type="danger"
:disabled="form.sheet_name == ''"
@click="outExcel()"
>确认导出</el-button
>
</span>
</el-dialog>
</div>
</template>
<script>
import XLSX from "xlsx";
export default {
props: ["selection_button_state", "selection_button_title", "selection_data"],
data() {
return {
dialogVisible: false,
form: {
file_name: "",
sheet_name: "",
},
};
},
methods: {
outExcel() {
```

```
let tableData = [
[
"昵称",
"用户名",
"用户邮箱",
"用户电话",
"性别",
"用户年龄",
"用户类别",
],
]; //  自定义表格表头
this.selection_data.forEach((item) => {
let rowData = [];
rowData = [
item.user_pet_name,
item.user_name,
item.user_email,
item.user_phone,
item.user_sex,
item.user_age,
item.user_label,
]; //每行对应的数据
tableData.push(rowData);
});
let ws = XLSX.utils.aoa_to_sheet(tableData);
let wb = XLSX.utils.book_new();
XLSX.utils.book_append_sheet(wb, ws, this.form.sheet_name); //  工作簿名称
XLSX.writeFile(wb, this.form.file_name + ".xlsx"); //  保存的文件名
this.$message.success("已导出选中数据成功，请您查看下载的 excel");
},
},
};
</script>
```

Js-xlsx 提取 excel 文件数据为 json 数据

原理:

要将 excel 文件的数据取出，在 js-xlsx 库的支持下需要三步便可完成数据的导入和数据 key 值的替换。

1. 文件导入，将需要提取的 excel 文件导入到前端。

2. Excel 文件读取，通过 js-xlsx 库将数据读取出来。

3. 通过 key 字典转换，将 excel 中每列的标题替换为数据对应的 key 方便后端将数据存入数据库。

示例代码:

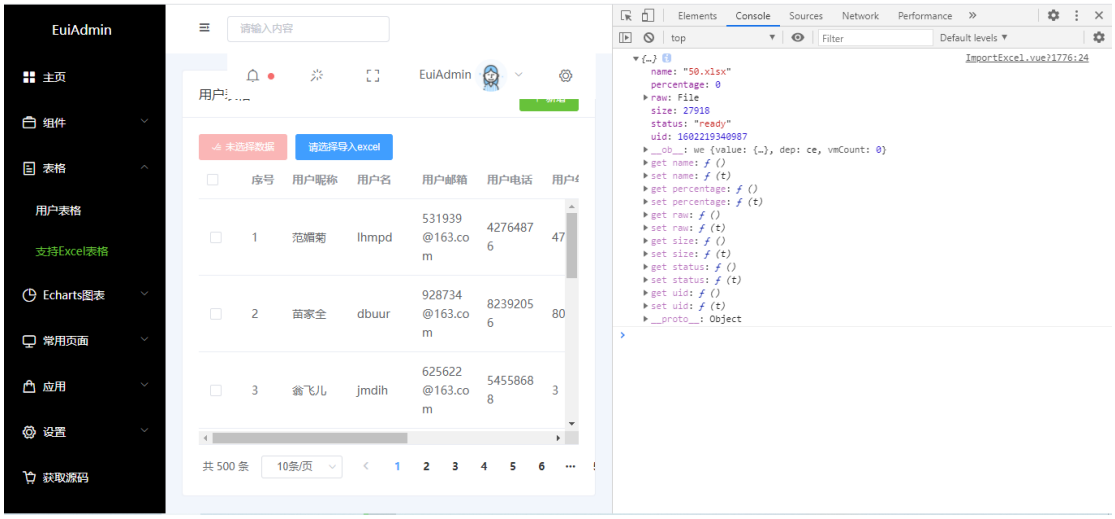
```
<template>
<el-upload
class="upload-demo"
action
:auto-upload="false"
:show-file-list="false"
:on-change="choose_file"
>
<el-button size="small" type="primary">请选择导入 excel</el-button>
</el-upload>
</template>
<script>
import XLSX from "xlsx";
export default {
data() {
return {
file: "",
excel_import_data:''
};
},
methods: {
choose_file(file) {
this.file = file.raw;//这是 element 的导入数据选择，必须要添加.raw 才能获取，其他表单不需要
// console.log(this.file);//上传文件信息
this.importExcel(this.file)
},
importExcel(file) {
var excelData = [];
//声明一个文件读取器
const fileReader = new FileReader();
//文件读取成功时触发事件
fileReader.onload = (ev) => {
try {
```

```
//读取的文件;

const data = ev.target.result;
//以二进制流方式读取得到整份 excel 表格
const workbook = XLSX.read(data, { type: "binary" });
//  循环遍历 excel 的 sheet 对象
Object.keys(workbook.Sheets).forEach((sheet) => {
console.info(workbook.Sheets[sheet]["!ref"]);
excelData.push(
//将 excel  转换成 json 对象放入数组中
...XLSX.utils.sheet_to_json(workbook.Sheets[sheet])
);
});
//  自定义事件，比如校验 excel 数据。转换数据格式…
// console.log(excelData)//未转换 key 值的数据
this.changeKey(excelData)//调用转换 key 值
} catch (e) {
this.$message.danger('文件类型不正确');
}
};
//读取文件
fileReader.readAsBinaryString(file);
},
changeKey(excelData){
var newKeyMap={//key 转换字典，左侧为旧的 key 值。右侧为新 key 值
'昵称':'user_pet_name',
'用户名':'user_name',
'用户邮箱':'user_email',
'用户电话':'user_phone',
'性别':'user_sex',
'用户年龄':'user_age',
'用户类别':'user_label',
}
for (let i = 0; i < excelData.length; i++) {//进行字典替换
var obj = excelData[i];
for (let key in obj) {
var newKey=newKeyMap[key]
if(newKey){
obj[newKey]=obj[key]
delete obj[key]
}
}

}
```

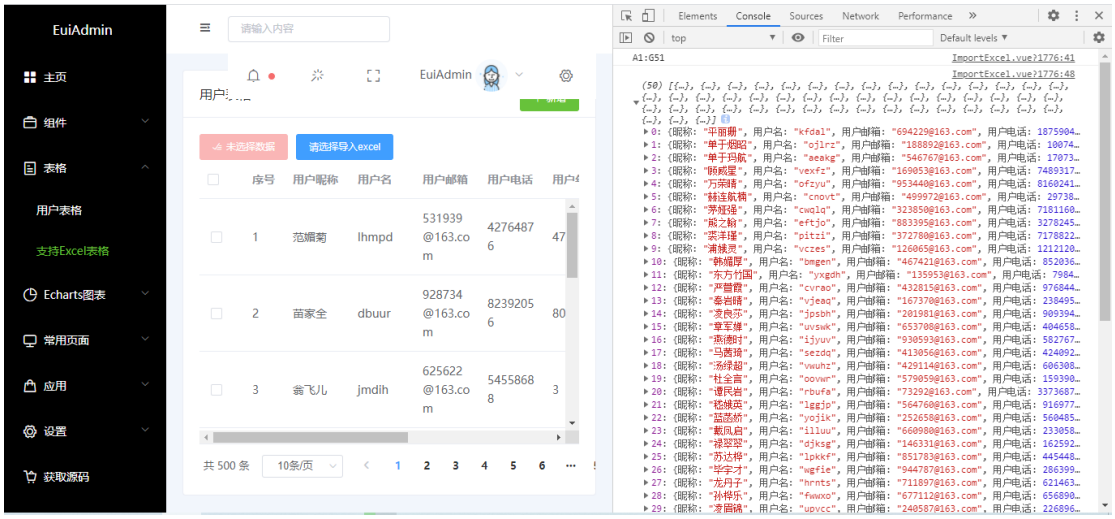
```
this.excel_import_data=excelData//转换成功后的数据
// console.log(this.excel_import_data)
this.$emit('excel_data',this.excel_import_data)//将数据传到父组件
this.$message.success("导入成功，已为您显示到数据表格，您也可以查看 console");
}
},
};
</script>
```

示例图片：

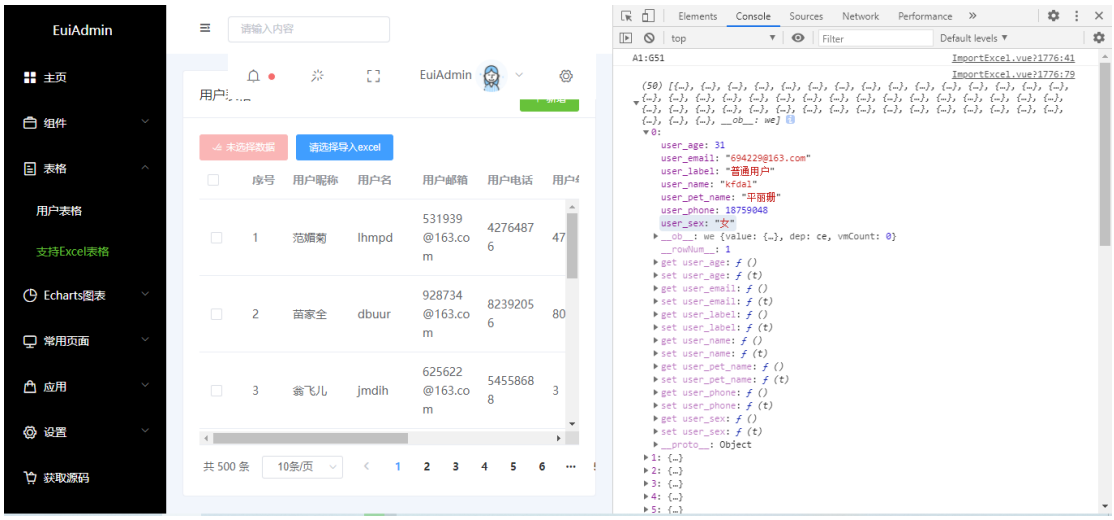
选择 excel 文件：



读取 50excel 数据：



替换 key 值后：



注意事项:

1、注意 key 字典数据的替换，左侧为 excel 文件中的标题，右边为要替换的内容；如果要存储到数据库那么要和数据库的 key 相对应。

结语:

你可以通过访问 euiadmin.com 获取源码进行参考，源码位于文件 src/module/excel/ImportExcel.vue 文件中；你可以通过源码更好的了解。

关于 EuiAdmin 官方音乐播放器

自编原因:

EuiAdmin 之前的音乐播放器是基于 vue-aplayer 实现的音乐播放功能，也基本实现了列表、播放暂停控制等功能，但经使用发现此 JavaScript 库在自定义上不够深，无法快速完成自定义。

在音乐播放时无法将歌词进行全面展示，仅支持小歌词模式，交互能

力有限；所以我决定自己写一个音乐播放器。

实现原理:

通过使用 html 标签<audio/>实现音乐播放功能，然后通过<audio/>的 api 实现数据的读取和响应控制。

<audio/>相关事件和 Api

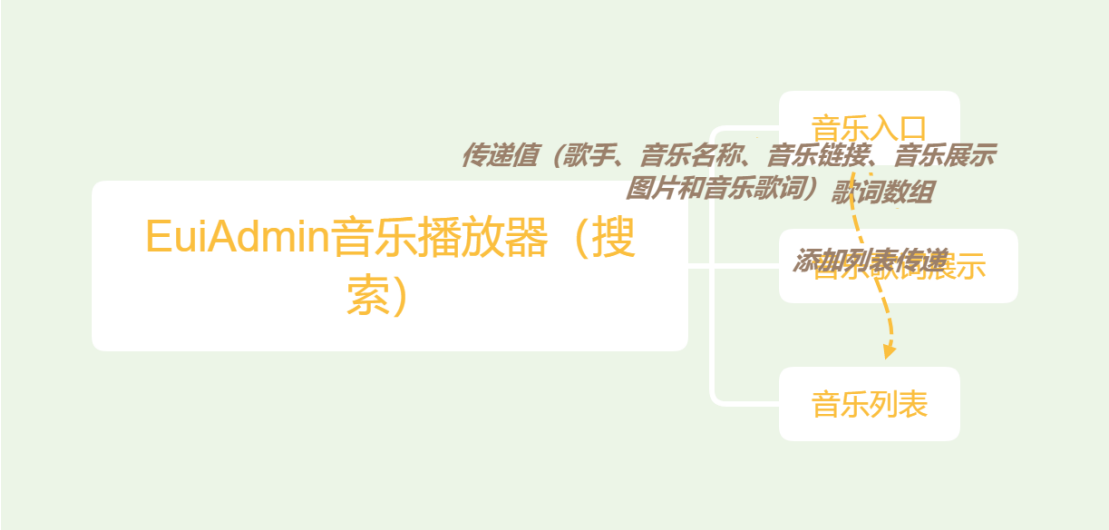
只读属性	描述
duration	获取媒体文件的播放时长，以 s 为单位，如果无法获取则为 NaN，当触发 canplay 事件后就可以获取当前总长度
startTime	返回起始播放时间，一般是 0.0,除非是缓冲过的媒体文件，并一部分内容已经不再缓冲区(此属性好像已经不可用)
paused	判断是否已经暂停，返回 true/false
ended	判断是否已经播放完毕，返回 true/false
error	在发生了错误后，返回错误代码
currentSrc	以字符串的形式发挥正在播放或已经加载的文件，对应浏览器在 source 元素中选择的文件
buffered	获取当前缓冲区大小，返回 TimeRanges 对象，点击更多参考
可控制属性	
src	指定音频的文件位置
autoplay	是否自动播放
preload	是否预加载
loop	是否循环播放
controls	显示或隐藏用户控制界面
autobuffer	媒体文件播放前是否进行缓冲加载，如果设置了 autoplay，则忽略此特性(此属性好像已经不可用)
muted	设置是否静音
volume	在 0.0 到 1.0 间的音量值，或查询当前音量值
currentTime	以 s 为单位返回从开始播放到目前所花的时间，也可设置 currentTime 的值来跳转到特定位置
方法	
load()	加载音频、视频软件
paly()	播放
pause()	暂停
canPlayType(obj)	测试饭后指定指定的 Mime 类型的文件

事件	
loadstart	客户端开始请求数据
progress	正在播放的时候不停触发，如果暂停不会触发，触发的时间间隔比较大
play	play()和 autopaly 播放时，类似事件 onplaying
pause	pause()方法触发时
ended	当结束播放时
timeupdate	当前播放时间发生改变的时候，播放中常用的时间处理，如果暂停不会触发，触发的时间间隔比较小
canplaythrough	歌曲已经载入完成
canplay	缓冲至可播放状态，类似事件 onloadedmetadata
onloadedmetadata	当元数据（比如分辨率和时长）被加载时运行的脚本
更多属性	
audioTracks	返回表示可用音频滚到的 AudioTrackList 对象。
controller	返回表示音频大年媒体控制器的 MediaController 对象。
crossOrigin	设置或返回音频的 CORS 设置
defaultMuted	设置或返回音频默认是否静音
defaultPlaybackRate	设置或返回音频的默认播放速度
mediaGroup	设置或返回音频叔叔的美肌组合的名称
networkState	返回音频的当前网络状态
playbackRate	设置或返回音频的播放速度。
seekable	返回标识音频可寻址不烦的 TimeRanges 对象
seeking	返回用户当前收正在音频中进行查找。
textTracks	返回标识文本滚到的 TextTrackList 对象
更多方法	
load()	重新加载音频元素
getStartDate()	返回新的 Date 对象，表示当前时间线偏移量。
fastSeek()	在音频播放器中指定播放时间。
addTextTrack()	想音频添加新的文本轨道。

EuiAdmin 音乐播放器实现原理:

通过将音乐播放器做成一个组件，在搜索界面加载音乐组件，通过将搜索结果与 EiAdmin 的 api 获取音乐链接，然后将数据通过组件传参的形式将(歌手、音乐名称、音乐链接、音乐展示图片和音乐歌词“数组”)传递给音乐组件; 然后将值赋给<audio/>的响应 api 然后实现音乐播放等控制。

流程图:



实现数据的控制和传递，要了解音乐播放器你的了解组件传值。

搜索功能实现:

```
<template>
  <div>
    <el-card style="margin-bottom: 120px">
      <div slot="header" class="clearfix">
        <span>EuiAdmin 音乐</span>
        <el-button
          size="small"
          type="danger"
          @click="search_music()"
          icon="el-icon-search"
          style="float: right"
          >搜索</el-button
        >
        <el-input
          v-model="search_value"
          placeholder="请输入音乐相关信息"
          @change="search_music()"
          size="small"
          style="float: right; width: 200px"
        ></el-input>
      </div>
      <div style="min-height: 20vh">
        <el-table :data="music_data.abslist" style="width: 100%">
```

```
  <el-table-column type="index" width="50" prop="date" label="序号">
  </el-table-column>
  <el-table-column label="歌名" width="500">
    <template slot-scope="scope">
      <el-avatar shape="square" :src="scope.row.hts_MVPIC"></el-avatar>
      <span v-html="scope.row.SONGNAME"></span>
      <div style="float: right">
        <i
          class="el-icon-video-play"
          @click="play_music(scope.$index, scope.row)"
          style="font-size: 150%; color: #409eff; cursor: pointer"
        />
        <i
          class="el-icon-plus"
          @click="add_music_list(scope.$index, scope.row)"
          style="
            font-size: 150%;
            color: #f56c6c;
            margin-left: 10px;
            cursor: pointer;
          "
        />
      </div>
    </template>
  </el-table-column>
  <el-table-column label="歌手">
    <template slot-scope="scope">
      <span v-html="scope.row.ARTIST"></span>-
      <span v-html="scope.row.AARTIST"></span>
    </template>
  </el-table-column>
  <el-table-column label="专辑">
    <template slot-scope="scope">
      <span v-html="scope.row.ALBUM"></span>
    </template>
  </el-table-column>
  <el-table-column
    prop="PLAYCNT"
    label="播放次数"
    width="100"
  ></el-table-column>
</el-table>
<el-pagination
  @size-change="page_size_change"
```

```
          @current-change="current_change"
          :current-page="page_data.current_page"
          :page-sizes="[10, 20, 50, 100, 200, 500]"
          :page-size="page_data.page_size"
          layout="total, sizes, prev, pager, next, jumper"
          :total="page_data.page_total"
          style="margin-top: 10px"
        ></el-pagination>
      </div>
      <MusicSpace :music="music_play_data" auto_play="1" :list="music_list"  style="margin-top:10px"/>
    </el-card>

  </div>
</template>
<script>
export default {
  components: {
    MusicSpace: (resolve) => {
      require(["@/components/apply/mymusic/MusicSpace"], resolve);
    },
  },
  data() {
    return {
      search_value: "",
      music_list: [],
      music_play_data: {
        music_name: "",
        music_artist: "",
        music_url: "",
        music_pic: undefined,
        music_lrc: [],
      },
      page_data: {
        page_size: 10,
        current_page: 1,
        page_total: 0,
      },
      music_data: [],
    };
  },
  methods: {
    play_music(index) {
      this.$axios({
```

```
      method: "post",
      url: "/music/play",
      data: this.$qs.stringify({
        music_id: this.music_data.abslist[index].MUSICRID,
      }),
    }).then((response) => {
      this.music_play_data = {
        music_name: this.music_data.abslist[index].SONGNAME,
        music_artist: this.music_data.abslist[index].ARTIST,
        music_url: response.data.music_url,
        music_pic: this.music_data.abslist[index].hts_MVPIC,
        music_lrc: response.data.lrc_data,
      };
    });
  },
  add_music_list(index) {
    this.$axios({
      method: "post",
      url: "/music/play",
      data: this.$qs.stringify({
        music_id: this.music_data.abslist[index].MUSICRID,
      }),
    }).then((response) => {
      this.music_list = {
        music_name: this.music_data.abslist[index].SONGNAME,
        music_artist: this.music_data.abslist[index].ARTIST,
        music_url: response.data.music_url,
        music_pic: this.music_data.abslist[index].hts_MVPIC,
        music_lrc: response.data.lrc_data,
      };
    });
  },
  page_size_change(page_size) {
    this.page_data.page_size = page_size;
    this.$message.success("每页显示" + page_size + "条数据");
    this.search_music();
  },
  current_change(click_page) {
    this.page_data.current_page = click_page;
    this.$message.success("正在展示第" + click_page + "页数据");
    this.search_music();
  },
  search_music() {
    this.$axios({
```

```
        method: "post",
        url: "/music/search",
        data: this.$qs.stringify({
          search_value: this.search_value,
          page_size: this.page_data.page_size,
          current_page: this.page_data.current_page,
        }),
      }).then((response) => {
        var json = eval("(" + response.data + ")");
        this.music_data = json;
        this.page_data.page_total = parseInt(json.TOTAL);
      });
    },
  },
};
</script>
```

音乐播放页:

音乐播放和暂停

```
play(state) {//音乐播放传递 state（true/false）控制播放和暂停
      if (
        this.format_date(this.play_time_data.music_duration_time) == "0:00:00"
      ) {
        this.$message.warning("未选择音乐");
        return false;
      }
      if (state) {
        this.play_state = state;
        this.$refs.audio.play();
        this.play_class =
          "animate__animated animate__flipOutY animate__slow animate__infinite
infinite";
      } else {
        this.play_state = state;
        this.$refs.audio.pause();
        this.play_class = "";
      }
    },
```

获取音乐时长

```
musci_time_length() {
      this.play_time_data.music_duration_time = this.$refs.audio.duration;
```

```
    },
```

获取音乐已播放时长

```
music_played_time(res) {
      this.play_time_data.music_current_time = res.target.currentTime;
}
```

音乐滑动控制

```
change_music_play_time(time) {
      this.play_time = time;
      this.$refs.audio.currentTime =
        (this.play_time / 100) * this.play_time_data.music_duration_time;
    },
```

修改播放速度

```
change_rate() {
      this.play_rate += 0.5;
      if (this.play_rate > 2) {
        this.play_rate = 0.5;
      }
      this.$refs.audio.playbackRate = this.play_rate;
    },
```

音乐播放结束触发

```
music_play_end() {
      this.play(false);
      this.$message.warning("播放结束");
      this.play_end_state = this.music;
    },
```

时间格式化

```
format_date(second) {
      var secondType = typeof second;
      if (secondType === "number" || secondType === "string") {
        second = parseInt(second);
        var hours = Math.floor(second / 3600);
        second = second - hours * 3600;
        var mimute = Math.floor(second / 60);
        second = second - mimute * 60;
        return (
          hours +
          ":" +
          ("0" + mimute).slice(-2) +
          ":" +
```

```
          ("0" + second).slice(-2)
        );
      } else {
        return "0:00:00";
      }
    },
```

歌词组件

```
<template>
  <div>
    <p id="lrc_one" @click="dialog_state = true">{{ text }}</p>
    <el-dialog :visible.sync="dialog_state">
      <template slot="title">
        <span id="music_lrc_title">{{ music_play_title }}</span>
      </template>
      <div style="min-height: 30vh">
        <p v-for="(up, i) in lrc.slice(index - 5, index)" :key="i">
          {{ up.lineLyric }}
        </p>
        <p id="lrc_list">{{ text }}</p>
        <p v-for="(lr, i) in lrc.slice(index + 1, index + 10)" :key="i">
          {{ lr.lineLyric }}
        </p>
      </div>
    </el-dialog>
  </div>
</template>
<script>
export default {
  props: ["lrc", "played_time"],
  data() {
    return {
      music_play_title: "EuiAdmin自写音乐服务",
      dialog_state: false,
      index: 0,
      lrc_time: 0.0,
      text: "EuiAdmin暂无歌词",
    };
  },
  methods: {
    get_lrc(time) {
      if (time == 0) {
        this.index = 0;
```

```
        this.text = this.lrc[this.index].lineLyric;
        this.lrc_time = this.lrc[this.index].time;
      } else if (time > this.lrc_time && time > this.lrc[this.index + 1].time)
{
        this.index += 1;
        this.text = this.lrc[this.index].lineLyric;
        this.lrc_time = this.lrc[this.index].time;
      }
      // console.log(this.text)
    },
  },
  watch: {
    played_time(val) {
      let played_lrc_time = Number(val.toFixed(2));
      this.get_lrc(played_lrc_time);
    },
  },
};
</script>
<style scoped>
#lrc_one {
  color: #f56c6c;
  cursor: pointer;
}
#lrc_list {
  color: #f56c6c;
  font-size: 120%;
}
#music_lrc_title {
  float: left;
  color: #409eff;
}
</style>
```

音乐列表组件实现:

添加歌曲到列表:

点击列表播放实现:

点击删除:

播放完毕自动下一曲:

目前仍存在的问题:

1、拖动音乐时歌词会出现“追赶”现象。

2、上下一曲未实现。

附件:

Animate 的 class 名称附表

class 前缀	动画名称	class 名称
animate__	bounce	animate__bounce
animate__	flash	animate__flash
animate__	pulse	animate__pulse
animate__	rubberBand	animate__rubberBand
animate__	shakeX	animate__shakeX
animate__	shakeY	animate__shakeY
animate__	headShake	animate__headShake
animate__	swing	animate__swing
animate__	tada	animate__tada
animate__	wobble	animate__wobble
animate__	jello	animate__jello
animate__	heartBeat	animate__heartBeat
animate__	backInDown	animate__backInDown
animate__	backInLeft	animate__backInLeft
animate__	backInRight	animate__backInRight
animate__	backInUp	animate__backInUp
animate__	backOutDown	animate__backOutDown
animate__	backOutLeft	animate__backOutLeft
animate__	backOutRight	animate__backOutRight
animate__	backOutUp	animate__backOutUp
animate__	bounceIn	animate__bounceIn
animate__	bounceInDown	animate__bounceInDown
animate__	bounceInLeft	animate__bounceInLeft
animate__	bounceInRight	animate__bounceInRight
animate__	bounceInUp	animate__bounceInUp
animate__	bounceOut	animate__bounceOut
animate__	bounceOutDown	animate__bounceOutDown
animate__	bounceOutLeft	animate__bounceOutLeft
animate__	bounceOutRight	animate__bounceOutRight

animate__	bounceOutUp	animate__bounceOutUp
animate__	fadeIn	animate__fadeIn
animate__	fadeInDown	animate__fadeInDown
animate__	fadeInDownBig	animate__fadeInDownBig
animate__	fadeInLeft	animate__fadeInLeft
animate__	fadeInLeftBig	animate__fadeInLeftBig
animate__	fadeInRight	animate__fadeInRight
animate__	fadeInRightBig	animate__fadeInRightBig
animate__	fadeInUp	animate__fadeInUp
animate__	fadeInUpBig	animate__fadeInUpBig
animate__	fadeInTopLeft	animate__fadeInTopLeft
animate__	fadeInTopRight	animate__fadeInTopRight
animate__	fadeInBottomLeft	animate__fadeInBottomLeft
animate__	fadeInBottomRight	animate__fadeInBottomRight
animate__	fadeOut	animate__fadeOut
animate__	fadeOutDown	animate__fadeOutDown
animate__	fadeOutDownBig	animate__fadeOutDownBig
animate__	fadeOutLeft	animate__fadeOutLeft
animate__	fadeOutLeftBig	animate__fadeOutLeftBig
animate__	fadeOutRight	animate__fadeOutRight
animate__	fadeOutRightBig	animate__fadeOutRightBig
animate__	fadeOutUp	animate__fadeOutUp
animate__	fadeOutUpBig	animate__fadeOutUpBig
animate__	fadeOutTopLeft	animate__fadeOutTopLeft
animate__	fadeOutTopRight	animate__fadeOutTopRight
animate__	fadeOutBottomRight	animate__fadeOutBottomRight
animate__	fadeOutBottomLeft	animate__fadeOutBottomLeft
animate__	flip	animate__flip
animate__	flipInX	animate__flipInX
animate__	flipInY	animate__flipInY
animate__	flipOutX	animate__flipOutX
animate__	flipOutY	animate__flipOutY
animate__	lightSpeedInRight	animate__lightSpeedInRight
animate__	lightSpeedInLeft	animate__lightSpeedInLeft
animate__	lightSpeedOutRight	animate__lightSpeedOutRight
animate__	lightSpeedOutLeft	animate__lightSpeedOutLeft
animate__	rotateIn	animate__rotateIn
animate__	rotateInDownLeft	animate__rotateInDownLeft
animate__	rotateInDownRight	animate__rotateInDownRight
animate__	rotateInUpLeft	animate__rotateInUpLeft
animate__	rotateInUpRight	animate__rotateInUpRight
animate__	rotateOut	animate__rotateOut
animate__	rotateOutDownLeft	animate__rotateOutDownLeft

animate__	rotateOutDownRight	animate__rotateOutDownRight
animate__	rotateOutUpLeft	animate__rotateOutUpLeft
animate__	rotateOutUpRight	animate__rotateOutUpRight
animate__	hinge	animate__hinge
animate__	jackInTheBox	animate__jackInTheBox
animate__	rollIn	animate__rollIn
animate__	rollOut	animate__rollOut
animate__	zoomIn	animate__zoomIn
animate__	zoomInDown	animate__zoomInDown
animate__	zoomInLeft	animate__zoomInLeft
animate__	zoomInRight	animate__zoomInRight
animate__	zoomInUp	animate__zoomInUp
animate__	zoomOut	animate__zoomOut
animate__	zoomOutDown	animate__zoomOutDown
animate__	zoomOutLeft	animate__zoomOutLeft
animate__	zoomOutRight	animate__zoomOutRight
animate__	zoomOutUp	animate__zoomOutUp
animate__	slideInDown	animate__slideInDown
animate__	slideInLeft	animate__slideInLeft
animate__	slideInRight	animate__slideInRight
animate__	slideInUp	animate__slideInUp
animate__	slideOutDown	animate__slideOutDown
animate__	slideOutLeft	animate__slideOutLeft
animate__	slideOutRight	animate__slideOutRight
animate__	slideOutUp	animate__slideOutUp

注：

1、请在 euiadmin 动画组件中进行测试，动画名称请复制表格中的 class 名称。