学校端后台管理系统

当前版本：v1.0.0

安装依赖

```
// install dependencies
npm install
```

运行

开发环境

```
npm run dev
```

生产环境打包(Build)

```
npm run build
```

文件结构

```
.
├── build  项目构建配置
└── src
    ├── images  图片文件
    ├── libs  工具方法
        ├── htmlToPdf.js html文件转PDF的第三方库
        ├── util.js 公用工具方法
    ├── router  路由配置
    ├── store  状态管理
    ├── styles  样式文件
    ├── components 共用组件
    └── views 文件页面
```

环境地址

测试环境： https://patriarch-tm.tanmasports.com/organ 正式环境： https://patriarch.tanmasports.com/organ
答题系统测试环境--正式环境去掉-test： https://html-test.tanmasports.com/page/

swagger文档地址

http://patriarch-tm.tanmasports.com:8184/swagger-ui.html

页面结构

```
<template>
    // html内容
</template>
<script>
    data () {
        // 页面数据
    },
    components: {
        // 引入组件
    },
    computed () {
        // 计算属性
    },
    watch: {
        // 数据监听
    },
    mounted () {
        // 生命周期
    },
    methods: {
        // 页面方法
    }

</script>
// 每个页面需要加 scoped 参数，避免全局污染！！！
<style scoped lang="less">
</style>
```

命名规范

变量

命名方法: 小驼峰式命名法 命名规范：前缀为形容词 （函数前缀为动词, 以此来区分函数和变量）

```
let maxCount = 10;
let tableTitle = '啦啦啦';
```

常量

命名方法：名词全部大写 命名规范：使用大写字母和下划线来组合命名，下划线用来分割单词。

```
const MAX_COUNT = 10;
const URL = '//www.huifenqi.com';
```

函数 & 方法

命名方法： 小驼峰式命名法 命名规范： 前缀应该为动词 命名建议：常用动词约定

动词	含义
can	判断是否可执行某个动作
has	判断是否含义某个值
is	判断是否为某个值
get	获取某个值
set	设置某个值
load	加载某些数据

页面参数

定义`propsData`来统一获取页面参数

例如：

```
data () {
    sessionData: {
        userId: '' // 定义好该页面要接收的参数，默认为空字符串
    }
},
mounted () { // mounted声明周期
    this.sessionData = {
        // 参数名： 获取参数
        userId: sessionStorage.getItem('userId') || '' // 注意：必须使用 || 进行判断，避免未取到值时赋值为null
    }
}
```

全局API管理

baseUrl

定义系统所有接口URL，按功能模块进行整理，统一暴露出去

```
// baseUrl.js
const roleManage = { // 人员管理模块
    // 常量名： 接口地址
    STUDENT_INFO_LIST: '/v1/studentinfo/getStudentInfoList'
}
// 更多模块...
module.exports = {
```

```
  roleManage, // 把定义好的模块暴露出去
}
```

baseApi

定义系统所有接口

```
// baseApi.js
import http from 'axios' // 引入axios
import {roleManage} from './baseUrl' // 按需引入对应的url模块

const api = {
  studentInfoList: (params) => http.post(roleManage['STUDENT_INFO_LIST'], params)
// 定义请求方法
}
export default api
```

使用

在需要调用的页面引入全局API模块

```
import api from '@/api/baseApi
```

使用 ES6 `async/await` 异步解决方案调用api

```
methods: {
    // 格式: async 方法名() {}
    async getStudentInfoList(schoolId) {
      let data = {
        // 请求参数
      }
      const result = await api.studentInfoList(data)
      if (result.data.code === 10000) {
        // 请求成功处理程序...
      } else {
        // 请求失败处理程序
      }
    },
}
```

使用`then()`方法调用

```
methods: {
    getStudentInfoList(schoolId) {
        let data = {
            // 请求参数
        }
        api.studentInfoList(data).then(res => {
            if (result.data.code === 10000) {
                // 请求成功处理程序...
            } else {
                // 请求失败处理程序
            }
        })
    },
}
```