**前端调用接口规范v1.1(待更新)**

前端调用接口规范内容包含列表查询、新增提交、删除、重置数据，var let const使用场景。备注：

**一、前端接口使用**

## 1、列表查询

*// 初始化数据[列表查询]*

initData(s) {

GetselectByEntity(this.formList).then(response => {

console.log(response)

if (response.code === 0) {

this.tableData = response.data

this.total1 = response.total

} else {

this.$message({

message: response.message,

type: ‘warning’

})

}

}).catch()

},

## 2、新增提交代码

*// 某新增提交*

SubmitAdd(val, e) {

switch (e) {

case 1:

var json = {

'orgNo': this.templateSelection.orgNo, *// 机构*

}

//判断是否是定点机构

if (this.getOrgbaseinfoFrom.objType === '1') {

json.hospDeptCode = this.templateSelection2.hospDeptCode

}

sceneDeviceOrgCAdd(json).then(response => {

console.log(response)

if (response.code === 0) {

this.$notify({

title: '成功,

type: 'success',

message: response.message

})

this.initData(1)

}

}).catch()

break

}

},

*// 提交*

SubmitAdd() {

//传值命名

const json = { 'focusTypeId': this.formInEdit.focusTypeId }

focustypemanageUpdate(json).then(response => {

console.log(response)

if (response.code === 0) {

//请求成功提示

this.$notify({

title: '成功,

type: 'success',

message: response.message

})

//关闭弹出层

this.dialogVisible = false

//成功后执行一次列表接口

this.initData(1)

} else {

//请求错误提示

this.$message({

message: response.message,

type: ‘warning’

})

}

}).catch()

},

## 3、删除

*// 删除*

deleteData(e) {

this.$confirm('此操作将永久删除该数据 是否继续?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}).then(() => {

deleteInfo({ id: e.sceneDeviceId }).then(response => {

console.log(response)

if (response.code === 0) {

this.$notify({

title: '成功,

type: 'success',

message: response.message

})

this.initData()

*// this.sceneDeviceId = response.sceneDeviceId*

}

})

}).catch(() => {

this.$message({

type: 'info',

message: '已取消删除'

})

})

},

## 4、重置信息

1、关闭弹出层

*// 关闭弹出层*

handleClose() {

this.dialogVisible = false

Object.assign(this.$data, this.$options.data())

this.initData()

},

2、重置菜单

图形用户界面, 文本

描述已自动生成



文本

描述已自动生成

*// 重置菜单*

resetForm(formName) {

if (this.$refs[formName] !== undefined) {

this.$refs[formName].resetFields()

}

},

二、命名规范

## 1、代码命名使用场景

首先必须使用**驼峰命名法，**

**大驼峰： SumName ------，如click 事件名称**

**小驼峰：sumName ------，如data参数初始化 命名**

## 2、变量命名规则

统一使用：

Const 变量名

const json = { }

focustypemanageUpdate(json).then(response => {

console.log(response)

if (response.code === 0) {

this.$notify({

title: '成功,

type: 'success',

message: response.message

})

this.dialogVisible = false

this.initData(1)

}

})

尽量少用var值，因为会引起一些错误，如果需要写可替换性循环值 可使用var如：

for (var i = 0; i < 10; i++) {

console.log(i)

}

备注：let是不可更改变量，使用场景较少。

## 3、新建业务模块命名规则

使用大驼峰命名并以英文方式展示。如下图。

图形用户界面, 文本

中度可信度描述已自动生成

## 子模块中API文件命名应与当前模块名称一致，如下图：

