前言

该文档主要是在开发项目中整理出来的一些比较常用的一些知识点。

开发环境: Node.js

网站：https://nodejs.org/en/

一、VUE基础

1.1 Vue 环境搭建

1.1.1 Node安装

通过官网下载 Node 安装文件，安装，然后可以通过命令`node -v`查看node版本，`npm -v`查看npm版本

```
C:\Users\Administrator\Desktop>node -v
v12.13.0
C:\Users\Administrator\Desktop>npm -v
6.12.0
```

npm常用命令：

- npm install moduleNames：安装Node模块
- npm view moduleNames：查看node模块的package.json文件夹
- npm list：查看当前目录下已安装的node包
- npm help：查看帮助命令
- npm view moudleName dependencies：查看包的依赖关系
- npm view moduleName repository.url：查看包的源文件地址
- npm view moduleName engines：查看包所依赖的Node的版本
- npm help folders：查看npm使用的所有文件夹
- npm rebuild moduleName：用于更改包内容后进行重建
- npm outdated：检查包是否已经过时，此命令会列出所有已经过时的包，可以及时进行包的更新
- npm update moduleName：更新node模块
- npm uninstall moudleName：卸载node模块

1.1.2 淘宝镜像

通过 使用淘宝镜像的命令安装淘宝镜像，安装完毕后，可以通过命令`cnpm-v`查看版本是否安装成功。

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
C:\Users\Administrator\Desktop>cnpm -v
cnpm@6.1.0
```

1.1.3 脚手架安装

通过 npm 命令安装 vue-cli 脚手架

```
npm install vue-cli -g //2.0
npm install @vue/cli -g //3.0

C:\Users\Administrator\Desktop>vue -V
2.9.6
```

1.2 Vue项目创建(vue-cli2.9.6)

1.2.1 创建项目

通过命令`vue init webpack [项目名称]`命令启动创建项目向导，按照向导提示操作完成项目创建

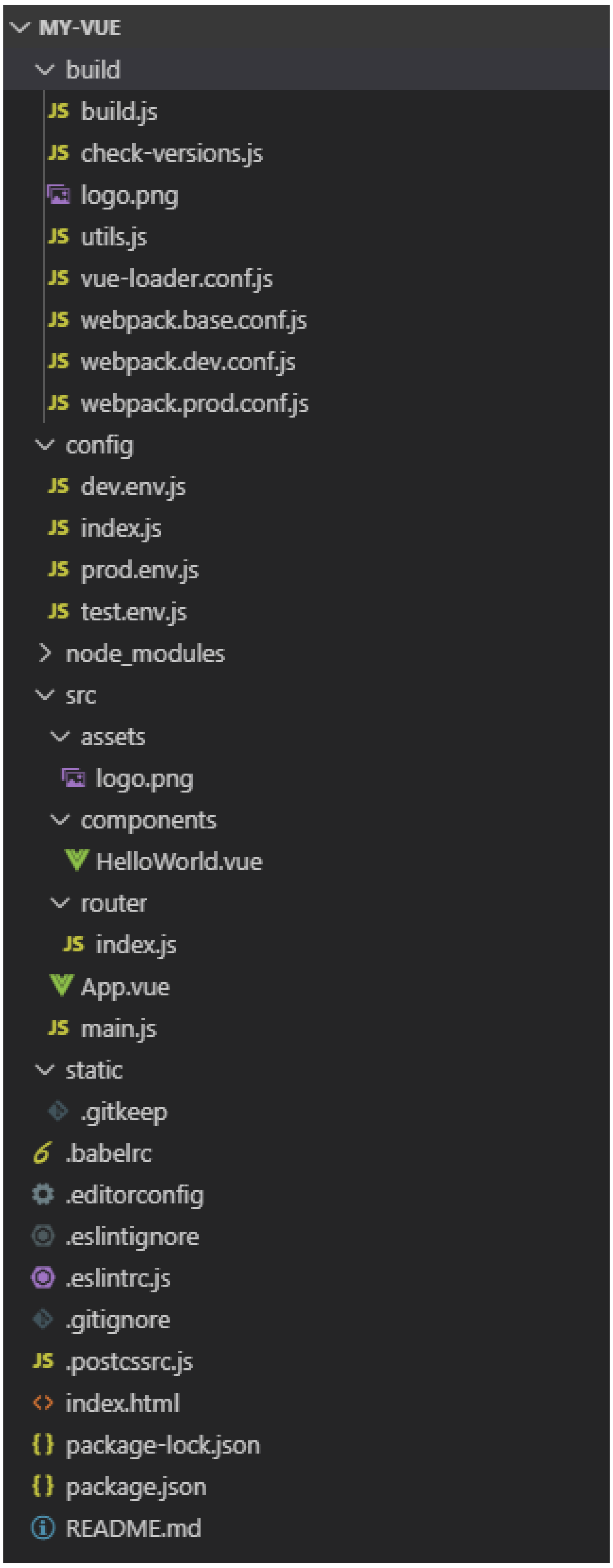
```
C:\Users\Administrator\Desktop>vue init webpack my-vue  //创建项目名称为my-vue
? Project name text //项目名称
? Project description wanghui //项目描述
? Author wh //项目作者
> Runtime + Compiler: recommended for most users  //vue构建方式选择 默认直接回车
  Runtime-only: about 6KB lighter min+gzip, but templates (or any Vue-specific H

TML) are ONLY allowed in .vue files - render functions are required elsewhere
? Vue build (Use arrow keys)
? Vue build standalone
? Install vue-router? (Y/n) y //是否安装路由，根据项目情况而定一般都需要安装 输入Y回车
? Use ESLint to lint your code? (Y/n)y //是否安装代码检查【新手N，实际项目Y】
> Standard (https://github.com/standard/standard)  //选择ESLint代码检查规范 默认直接
回车
  Airbnb (https://github.com/airbnb/javascript)
  none (configure it yourself)
? Set up unit tests (Y/n) n //单元测试，推荐N
? Setup e2e tests with Nightwatch? (Y/n) //e2e测试，推荐N
? Should we run `npm install` for you after the project has been created? (recom
mended) (Use arrow keys) //选择安装方式 选择NPM安装
> Yes, use NPM
  Yes, use Yarn
  No, I will handle that myself

mended) npm
   vue-cli · Generated "my-vue".
# Installing project dependencies ...
# ========================
//安装中.....................................
# Project initialization finished!
# ========================

To get started:
  cd my-vue
  npm run dev
```

1.2.2 运行项目

在项目更目录，打开命令窗口，使用命令`npm run dev`或者 npm start `启动项目`，启动后通过浏览器打开运行地址，就可以在浏览器显示查看了。

```
C:\Users\Administrator\Desktop\my-vue>npm start
I  Your application is running here: http://localhost:8080
```

1.2.3 项目目录



```
MY-VUE
  build
    build.js
    check-versions.js
    logo.png
    utils.js
    vue-loader.conf.js
    webpack.base.conf.js
    webpack.dev.conf.js
    webpack.prod.conf.js
  config
    dev.env.js
    index.js
    prod.env.js
    test.env.js
  node_modules
  src
    assets
      logo.png
    components
      HelloWorld.vue
    router
      index.js
    App.vue
    main.js
  static
    .gitkeep
  .babelrc
  .editorconfig
  .eslintignore
  .eslintrc.js
  .gitignore
  .postcssrc.js
  index.html
  package-lock.json
  package.json
  README.md
```

1.3 Vue项目创建(vue-cli4.0.5)

1.3.1 创建项目

```
C:\Users\Administrator\Desktop>vue -V
```

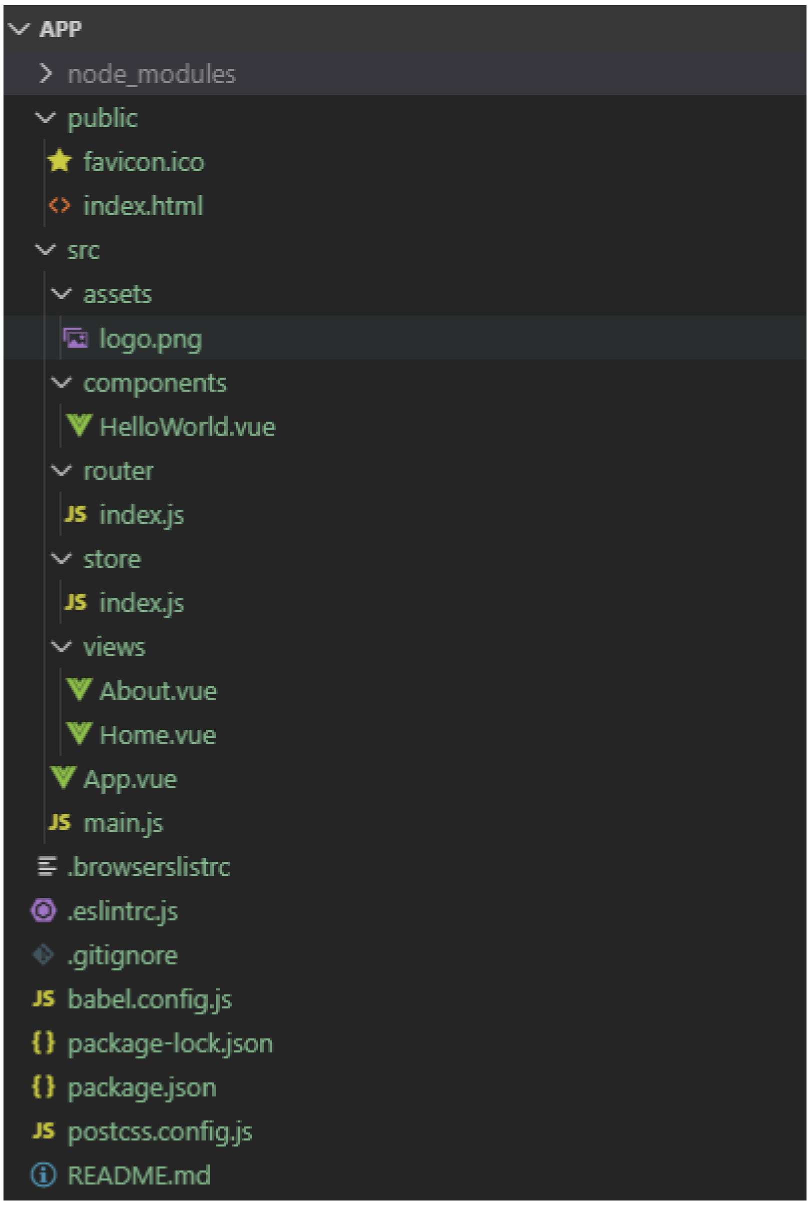
```
@vue/cli 4.0.5

C:\Users\Administrator\Desktop>vue create vueapp
?  Your connection to the default npm registry seems to be slow. //是否需要使用淘宝镜像
   Use https://registry.npm.taobao.org for faster installation? (Y/n)n
? Please pick a preset: (Use arrow keys) //按键盘上下键选择默认（default）还是手动（Manually），如果选择default，一路回车执行下去就行了（注：现在vue-cli3.0默认使用yarn下载），这里我选择手动
> default (babel, eslint)
  Manually select features
Vue CLI v4.0.5
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to invert selection) //通过方向下键和空格键选中需要安装的插件，选择完毕直接回车。
>(*) Babel
 ( ) TypeScript
 ( ) Progressive Web App (PWA) Support
 (*) Router
 (*) Vuex
 (*) CSS Pre-processors
 (*) Linter / Formatter
 ( ) Unit Testing
 ( ) E2E Testing
 Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n) //模块安装询问，应为安装了Router选择是 否使用路由 history router，其实直白来说就是是否路径带 # 号，建议选择 N，否则服务器还要进行配置
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): (Use arrow keys) //选择预处理CSS样式语言css 的预处理器我选择的是Sass/SCSS(with dart-sass) 。node-sass是自动编译实时的，dart-sass需要保存后才会生效sass 官方目前主力推 dart-sass 最新的特性都会在这个上面先实现
> Sass/SCSS (with dart-sass)
  Sass/SCSS (with node-sass)
  Less
  Stylus
? Pick a linter / formatter config: (Use arrow keys)  //选择 ESLint 代码校验规则，提供一个插件化的javascript代码检测工具，ESLint + Prettier 使用较多
> ESLint with error prevention only
  ESLint + Airbnb config
  ESLint + Standard config
  ESLint + Prettier
?Pick additional lint features:(Press <space> to select, <a> to toggle all, <i> to invert selection) //然后选择什么时候进行代码校验，Lint on save 保存就检查，Lint and fix on commit   fix 或者 commit 的时候检查，建议第一个回车
 >(*) Lint on save
  ( ) Lint and fix on commit
? Where do you prefer placing config for Babel, PostCSS, ESLint, etc.? (Use arrow keys)
> In dedicated config files //下面就是如何存放配置了，In dedicated config files 存放到独立文件中，In package.json 存放到 package.json 中本着项目结构简单的想法，我选择了第二个
  In package.json
? Save this as a preset for future projects? (y/N) n // 是否保存配置文件，
```

1.3.2 运行项目

在项目更目录，打开命令窗口，使用命令npm run serve 启动项目`，启动后通过浏览器打开运行地址，就可以在浏览器显示查看了。

```
C:\Users\Administrator\Desktop\vueapp>npm run serve
  App running at:
  - Local:   http://localhost:8080/
  - Network: http://192.168.3.101:8080/
  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

1.3.3 项目目录



```
APP
  node_modules
  public
    favicon.ico
    index.html
  src
    assets
      logo.png
    components
      HelloWorld.vue
    router
      index.js
    store
      index.js
    views
      About.vue
      Home.vue
    App.vue
    main.js
  .browserslistrc
  .eslintrc.js
  .gitignore
  babel.config.js
  package-lock.json
  package.json
  postcss.config.js
  README.md
```

1.4 Vue常用指令

1.4.1 v-model

在表单控件或者组件上创建双向绑定，随表单控件类型不同而不同

例：

```
<template>
    <div class="Text">
        <input type="text" v-model="name">
    </div>
</template>
//数据
data() {
    return {
        name: "大仙儿"
    }
}
```

1.4.2 v-for

基于源数据多次渲染元素或模板块,可以循环数组`Array`、字符串`String` 、对象`Object` 、数字`number`

1.4.2.1 循环一般数组

例：

```
<template>
    <div class="Text">
        <ul class="woman">
            <li v-for="(item,index) in woman" :key="index"></li>
        </ul>
    </div>
</template>
//数据
data() {
    return {
        woman: ["秋儿", "杨幂", "冰冰", "秀琴", "春花"]
    }
}
```

渲染结果：

```
<ul class="woman">
    <li>秋儿</li>
    <li>杨幂</li>
    <li>冰冰</li>
    <li>秀琴</li>
    <li>春花</li>
</ul>
```

14.2.2 循环对象数组

例：

```
<template>
    <div class="Text">
        <table class="person">
            <tr>
```

```
                <td>姓名</td>
                <td>性别</td>
                <td>年龄</td>
            </tr>
            <tr v-for="(item,index) in person" :key="index">
                <td>{{item.name}}</td>
                <td>{{item.sex}}</td>
                <td>{{item.age}}</td>
            </tr>
        </table>
    </div>
</template>
//数据
data() {
    return {
        person: [
            { name: "杨幂", sex: "女", age: 30 },
            { name: "邓超", sex: "男", age: 26 },
            { name: "周冬雨", sex: "女", age: 29 },
            { name: "王俊凯", sex: "男", age: 22 },
            { name: "刘亦菲", sex: "女", age: 45 },
        ]
    }
}
```

渲染结果：

```
<table class="person">
    <tr>
        <td>姓名</td>
        <td>性别</td>
        <td>年龄</td>
    </tr>
    <tr>
        <td>杨幂</td>
        <td>女</td>
        <td>30</td>
    </tr>
    <tr>
        <td>邓超</td>
        <td>男</td>
        <td>26</td>
    </tr>
    <tr>
        <td>周冬雨</td>
        <td>女</td>
        <td>29</td>
    </tr>
    <tr>
        <td>王俊凯</td>
        <td>男</td>
        <td>22</td>
    </tr>
    <tr>
        <td>刘亦菲</td>
        <td>女</td>
        <td>45</td>
    </tr>
</table>
```

1.4.2.3　循环对象

例：

```
<template>
    <div class="Text">
        <ul class="star">
            <li v-for="(item,key) in star" :key="key">{{key}}:{{item}}</li>
        </ul>
    </div>
</template>
//数据
data() {
    return {
        star: {
            name:"杨紫",
            age:27,
            height:"167cm",
            blood:"O",
            address:"成都市"
        }
    }
}
```

渲染结果：

```
<ul class="star">
    <li>name:杨紫</li>
    <li>age:27</li>
    <li>height:167cm</li>
    <li>blood:O</li>
    <li>address:成都市</li>
</ul>
```

1.4.2.4　循环字符串

例：

```
<template>
    <div class="Text">
        <ul class="strName">
            <li v-for="(item,index) in strName" :key="index">{{item}}</li>
        </ul>
    </div>
</template>
//数据
data() {
    return {
        strName:"我是前端工程师"
    }
}
```

渲染结果：

```
<ul class="star">
    <li>我</li>
    <li>是</li>
    <li>前</li>
    <li>端</li>
    <li>工</li>
    <li>程</li>
    <li>师</li>
</ul>
```

1.4.3 v-show

  显示内容，根据表达式之真假值，切换元素的 `display` CSS 属性

例：

```
<template>
    <div class="Text">
        <div class="v-show">
            <p v-show="showTrue">我现在的状态是：  true</p>
            <p v-show="showFalse">我现在的状态是：  false</p>
        </div>
    </div>
</template>
//数据
data() {
    return {
        showTrue:true,
        showFalse:false,
    }
}
```

渲染结果：

```

<div class="v-show">
    <p>我现在的状态是：  true</p>
    <p style="display: none;">我现在的状态是：  false</p>
</div>
```

1.4.4 if条件表达式

隐藏内容，根据表达式之真假值,来创建元素

1.4.4.1　v-if

例：

```
<template>
    <div class="Text">
        <div class="v-if">
            <p v-if ="ifTrue">我现在的状态是：  true</p>
            <p v-if ="ifFalse">我现在的状态是：  false</p>
        </div>
    </div>
</template>
//数据
data() {
    return {
        ifTrue:true,
        ifFalse:false,
    }
}
```

渲染结果：

```
<div class="v-if">
    <p>我现在的状态是：  true</p>
    <!---->
</div>
```

1.4.4.2 v-else

例：

```
<template>
    <div class="Text">
        <div class="v-else">
            <p v-if="count>=60">我的成绩为：及格</p>
            <p v-else>我的成绩为：不及格</p>
        </div>
    </div>
</template>
//数据
data() {
    return {
        count: 70,
    }
}
```

渲染结果：

```
<div class="v-else">
    <p>我的成绩为：及格</p>
</div>
```

1.4.4.3 v-else-if

例：

```
<template>
    <div class="Text">
        <div class="v-else">
            <p v-if="count>=85">我的成绩为：优</p>
            <p v-else-if="count>=75">我的成绩为：良</p>
            <p v-else-if="count>=60">我的成绩为：中</p>
            <p v-else>=75">我的成绩为：差</p>
        </div>
    </div>
</template>
//数据
data() {
    return {
        count: 70,
    }
}
```

渲染结果：

```
<div class="v-else">
    <p>我的成绩为：中</p>
</div>
```

1.4.5 v-bind

动态绑定作用： 及时对页面的数据进行更改

class 三种绑定方法

1、对象型 '{red:isred}'

2、三元型 'isred?"red":"blue"'

3、数组型 '[{red:"isred"},{blue:"isblue"}]'

例：

```
<!-- 绑定一个属性 -->
<img v-bind:src="imageSrc">

<!-- 动态特性名 (2.6.0+) -->
<button v-bind:[key]="value"></button>

<!-- 缩写 -->
<img :src="imageSrc">

<!-- 动态特性名缩写 (2.6.0+) -->
<button :[key]="value"></button>

<!-- 内联字符串拼接 -->
<img :src="'/path/to/images/' + fileName">

<!-- class 绑定 -->
<div :class="{ red: isRed }"></div>
<div :class="[classA, classB]"></div>
<div :class="[classA, { classB: isB, classC: isC }]">

<!-- style 绑定 -->
<div :style="{ fontSize: size + 'px' }"></div>
<div :style="[styleObjectA, styleObjectB]"></div>

<!-- 绑定一个有属性的对象 -->
<div v-bind="{ id: someProp, 'other-attr': otherProp }"></div>
```

```
<!-- 通过 prop 修饰符绑定 DOM 属性 -->
<div v-bind:text-content.prop="text"></div>

<!-- prop 绑定。“prop”必须在 my-component 中声明。-->
<my-component :prop="someThing"></my-component>

<!-- 通过 $props 将父组件的 props 一起传给子组件 -->
<child-component v-bind="$props"></child-component>
```

1.4.6 v-on:click

给标签绑定函数，可以缩写为@，例如绑定一个点击函数 函数必须写在methods里面

例：

```
<template>
    <div class="Text">
        <div class="v-else-if">
            <button @click="changeCount">点击事件</button>
            <p>当前分数：{{count}}</p>
        </div>
    </div>
</template>
//数据
data() {
    return {
        count: 70,
    }
},
methods:{
    //改变分数
    changeCount() {
       this.count = 90;
    }
}
```

渲染结果：

```
<div class="click">
    <button>点击事件</button>
    <p>当前分数：90</p> <!--点击按钮后将70修改为90-->
</div>
```

1.4.7 v-text

v-text解析文本和{{}} 效果一样

例：

```
<template>
    <div class="Text">
        <div class="text">
            <p class="no">{{name}}</p>
            <p class="no-v-text" v-text="name"></p>
        </div>
    </div>
</template>
 //数据
data() {
    return {
         name:"大仙儿"
    }
},
```

渲染结果：

```
<div class="text">
    <p class="no">大仙儿</p>
    <p class="no-v-text">大仙儿</p>
</div>
```

1.4.8 v-html

解析html标签

例：

```
<template>
    <div class="Text">
        <div class="v-html" v-html="htmlContent"></div>
    </div>
</template>
//数据
data() {
    return {
        htmlContent:`<h2>我是标题2</h2><div class="html">你的酒馆对我打了样</div>`
    }
}
```

渲染结果：

```
<div class="v-html">
    <h2>我是标题2</h2>
    <div class="html">你的酒馆对我打了样</div>
</div>
```

1.5 Vue 路由表格式

```
/* jshint esversion: 6 */
import Vue from "vue";
import Router from "vue-router";
import Login from "@/view/Login"; // 登陆界面
import Home from "@/view/Home/Home"; // 首页
import Server from "@/view/Server/Server"; // 终端管理页面
import Client from "@/view/Client/Client"; // 客户端管理页面
import ClientUser from "@/view/Client/SubPage/ClientUser"; // 客户端管理页面
import ClientAccess from "@/view/Client/SubPage/ClientAccess"; // 客户端管理页面
import System from "@/view/System/System"; // 系统管理页面
import SystemSet from "@/view/System/SubPage/SystemSet"; // 系统设置
import SystemGroup from "@/view/System/SubPage/SystemGroup"; // 分组管理
import SystemUser from "@/view/System/SubPage/SystemUser"; // 分组管理
Vue.use(Router);
export default new Router({
mode: "hash",
routes: [
    { path: "/", name: "Login", component: Login },
    { path: "/Home", name: "Home", component: Home },
    { path: "/Server", name: "Server", component: Server },
    { path: "/Client", name: "Client", component: Client, redirect:
"/Client/ClientUser",
      children: [
         { path: "/Client/ClientUser", name: "ClientUser", component: ClientUser
},
         { path: "/Client/ClientAccess", name: "ClientAccess", component:
ClientAccess },
      ]
    },
    { path: "/System", name: "System", redirect: "/System/SystemSet", component:
System,       children: [
        { path: "/System/SystemSet", name: "SystemSet", component: SystemSet },
        { path: "/System/SystemGroup", name: "SystemGroup", component:
SystemGroup },
        { path: "/System/SystemUser", name: "SystemUser", component: SystemUser
},
      ]
    }
  ]
});
```

二 VUE常用框架

2.1 Element UI

Element，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

官网：Element

2.1.1 安装

npm 安装

```
npm install element-ui -Save
```

CDN安装

```
<!-- 引入样式 -->
<link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-
chalk/index.css">
<!-- 引入组件库 -->
<script src="https://unpkg.com/element-ui/lib/index.js"></script>
```

2.1.2 配置`main.js`

全局引用

```
import ElementUI from 'element-ui';//element-ui框架
import 'element-ui/lib/theme-chalk/index.css';   //引入css样式文件
Vue.use(ElementUI);//注册组件
```

按需引用(使用组件极少使用)

https://element.eleme.cn/#/zh-CN/component/quickstart

2.1.3 使用

例：

```
<template>
    <div class="Text">
        <div class="element-button">
           <el-button type="primary">主要按钮</el-button> <!--Element 按钮组件-->
        </div>
    </div>
</template>
```

渲染结果：

```
<div class="element-button">
    <button type="button" class="el-button el-button--primary"><!----><!---->
        <span>主要按钮</span>
    </button>
</div>
```

2.1.4 国际化语言

```
// element 语言
import enLocale from "element-ui/lib/locale/lang/en";  //引入语言
import zhLocale from "element-ui/lib/locale/lang/zh-CN";
import locale from "element-ui/lib/locale";

if (localStorage.getItem("lang") === "zh-Cn") {
    locale.use(zhLocale);
} else {
    locale.use(enLocale);
}
locale.use(enLocale); //切换语言的时候改变
```

2.2 Echarts图表

官网：Echarts

实例网站：gallery.echartsjs

2.2.1 安装

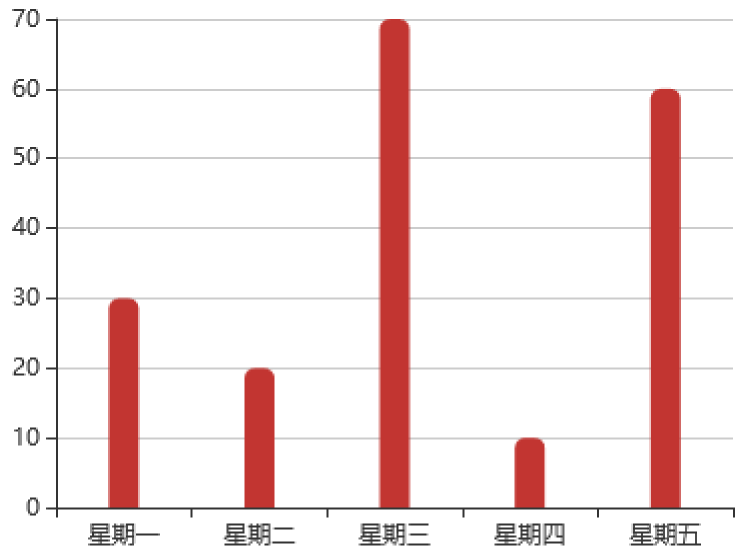
```
npm install echarts --save
```

2.2.2 配置 `main.js`

```
import echarts from "echarts"; // 图表框架
Vue.prototype.$echarts = echarts;
```

2.2.3 实例条形柱状图

```
<template>
  <div class="echLine"></div>
</template>
<script>
export default {
    name: "EchLine",
    data() {
    return {
        lineData: {
            title: ["星期一", "星期二", "星期三", "星期四", "星期五"],
            data: [
                { name: "星期一", value: 30 },
                { name: "星期二", value: 20 },
                { name: "星期三", value: 70 },
                { name: "星期四", value: 10 },
                { name: "星期五", value: 60 }
            ]
        }
    };
    },
    mounted() {
        this.drawLine();
    },
    methods: {
        drawLine() {
            // 初始化echarts实例
            let myChart =
            this.$echarts.init(document.getElementsByClassName("echLine")[0]);
            myChart.setOption({
            tooltip: {
                trigger: "axis"
            },
            grid: {
                top: "10%",
                left: "5%",
                right: "5%",
                bottom: "2%",
                containLabel: true
            },
            xAxis: {
                type: "category",
                data: this.lineData.title
            },
            yAxis: {
                type: "value"
            },
            series: [
                {
                name: "风险数量",
                type: "bar",
                data: this.lineData.data,
                barWidth: 15,
                // 设置柱状图样式
                itemStyle: {
                    normal: {
                        barBorderRadius: [5, 5, 0, 0] // 圆角
                    }
                }
                }
            ]
            });
            window.addEventListener("resize", () => {
                myChart.resize();
            });
            myChart.on('click', (param)=> {
                console.log(param,"点击事件1")
            })
        }
    }
};
</script>
<style>
.echLine{width:400px;height: 300px;}
</style>
```



三 VUE 常用组件

3.1 MD5加密

3.1.1 安装

```
npm install --save js-md5
```

3.1.2 配置`main.js`

```
import md5 from 'js-md5';  //md5加密
Vue.prototype.$md5 = md5;  //全局使用
```

3.1.3 案列

例：

```
<template>
  <div class="Text">
      <div class="md5">
          <p>未加密：{{md5}}</p>
          <p>MD5加密后：{{md5Format}}</p>
      </div>
  </div>
</template>
<script>
export default {
    data() {
        return {
            md5:"520520",
        };
    },
    computed:{
        md5Format(){
            return this.$md5(this.md5);
        }
    }
};
</script>
```

渲染结果：

```
<div class="md5">
    <p>未加密：520520</p>
    <p>MD5加密后：1104959d53dc3b60f2d40cd4a47d79e7</p>
</div>
```

3.2 html打印`vue-print-nb`

打印指定html部分内容，打印效果包含域名、页码、和时间

3.2.1 安装

```
npm install vue-print-nb --save
```

3.2.2 配置`main.js`

```
import Print from 'vue-print-nb';   //js打印
Vue.use(Print);  //注册
```

3.2.3 使用打印功能

```
<template>
  <div class="Text">
      <div id="printContent">我就是打印的内容</div>
      <el-button type="primary"  v-print="'#printContent'">打印</el-button>
  </div>
</template>
```

3.3 Axios 请求

3.3.1 安装

```
npm install --save axios //API请求方式
```

3.3.2 封装 `request.js`

该文件主要是axios拦截器统一处理 目录：src/service/request.js

以下文件为常用内容，不要剔除相应代码即可

```
/* jshint esversion: 6 */
import axios from "axios";
import store from "@/store/store.js"; //引入Vuex状态管理
import crypto from "crypto"; //前后端数据加密插件/
import md5 from "js-md5"; //MD5加密插件
import Vue from "vue";

const EncryptionOnOff = true;  // 数据加密开关
// 创建axios实例
const service = axios.create({
    baseURL: "http://www.network.com",
    timeout: 15000, // 请求超时时间,
});
Vue.prototype.$Service = service; //绑定VUE，可以全局查看AXIOS内容
// 添加请求拦截器
service.interceptors.request.use(
    function(config) {
        let systemTime = Math.round(new Date().getTime() / 1000).toString();
        let Siv = sessionStorage.getItem("iv");  // 登陆返回
        let Skey = sessionStorage.getItem("key");  // 登陆返回
        // 添加Token;
        config.headers.Authorization = "Bearer " +
sessionStorage.getItem("Token");
        // 统一参数
        if (config.data) {
            config.data.time_tamp = systemTime;
        } else {
            config.params.time_tamp = systemTime;
        }
        // 验证码、登陆、获取用户信息不需要数据加密需排除
        if (!config.dataJiami && EncryptionOnOff) {
            if (config.data) {
                let dataObj = Object.assign({}, config.data);
                config.data = Encryption(dataObj, Skey, Siv);
            }
            if (config.params) {
                let dataObj = Object.assign({}, config.params);
                config.params = Encryption(dataObj, Skey, Siv);
            }
        }
        // 签名处理
        if (config.data) {
            let sc_sign = objKeySort(objToJson(config.data)); // 将对象键名按小写从
a-z排序
            config.data._sign = signatureFormart(sc_sign); // 签名验证处理数据
        }
        if (config.params) {
            let sc_sign = objKeySort(objToJson(config.params)); // 将对象键名按小写
从a-z排序
            config.params._sign = signatureFormart(sc_sign); // 签名验证处理数据
        }
        console.log(config, "配置");

        return config;
    },
    function(error) {
        // 对请求错误做些什么
        return Promise.reject(error);
    }
);

// 添加响应拦截器
service.interceptors.response.use(
    function(response) {
        console.log(response, "响应");
        let Siv = sessionStorage.getItem("iv");  // 登陆返回
        let Skey = sessionStorage.getItem("key");  // 登陆返回
        // Token刷新
        if (Reflect.has(response.headers, "authorization") ||
Reflect.has(response.headers, "Authorization")) {
            try {
                sessionStorage.setItem("Token", response.headers.authorization);
            } catch (err) {
                sessionStorage.setItem("Token", response.headers.Authorization);
            }
        }
        // 数据解密
        // 验证码、登陆、获取用户信息不需要数据解密需排除
        if (!response.config.dataJiami && EncryptionOnOff) {
            let dataSource = response.data.data.body;
            let decipher = crypto
                .createDecipheriv("aes-256-cbc", Skey, Siv)
                .setAutoPadding(true);
            let decode = decipher.update(dataSource, "base64", "utf8");
            decode += decipher.final("utf8");
            decode = decodeURIComponent(decode);
            response.data.data.body = JSON.parse(decode);
        }
        return Promise.resolve(response);
    },
    function(error) {
        // 请求超时处理
        let originalRequest = error.config;
        if (error.code === "ECONNABORTED" && error.message.indexOf("timeout")
!== -1 && !originalRequest._retry) {
            store.commit("ON_TIMEOUT", 1);
        }
         const httpError = {
             hasError: true,
            status: error.response.status,
             statusText: error.response.statusText
         };
         store.commit("ON_HTTP_ERROR", httpError);
        return Promise.reject(error);
    }
);

/**
 * 数据加密方法
 * @name Encryption
 * @param dataObj 数据 | Skey  后台返回| Siv 后台返回
 */
function Encryption(dataObj, Skey, Siv) {
    let dataResults = {};
    for (let key in dataObj) {
        if (dataObj[key] !== "") {
            let str = encodeURIComponent(dataObj[key].toString()); // 请先编码
            let cipher = crypto.createCipheriv("aes-256-cbc", Skey, Siv);
            let crypt = cipher.update(str, "utf8", "base64");
            crypt += cipher.final("base64");
            dataResults[key] = crypt;
        }
    }
    return dataResults;
}
/**
 * 数组对象转换JSON格式
 * @name objToJson
 * @param  obj 数据Array
 */
function objToJson(obj) {
    for (let key in obj) {
        if (typeof obj[key] === "object") {
            obj[key] = JSON.stringify(obj[key]);
        }
    }
    return obj;
}
/**
 * 数字签名排序A-Z
 * @name objKeySort
 * @param  obj 数据Object
 */
function objKeySort(obj) {
    let newkey = Object.keys(obj).sort();
    let newObj = {}; // 创建一个新的对象，用于存放排好序的键值对
    for (let i = 0; i < newkey.length; i++) {
    // 遍历newkey数组
        newObj[newkey[i].toLowerCase()] = encodeURIComponent(obj[newkey[i]]); //
向新创建的对象中按照排好的顺序依次增加键值对
    // console.log(newkey[i].toLowerCase(),"编
码",newObj[newkey[i].toLowerCase()])
    }
    return newObj; // 返回排好序的新对象
}
/**
 * 数字签名数据处理格式
 * @name objKeySort
 * @param  obj 数据Object
 */
function signatureFormart(obj) {
    if (Reflect.has(obj, "_sign")) {
        delete obj._sign;
    }
    let newObj = obj;
    newObj = JSON.stringify(newObj); // 对象转换成字符串
    newObj = newObj.replace(/,/g, "&"); // ， 替换|
    newObj = newObj.replace(/"/g, ""); // "替换
    newObj = newObj.replace(/:/g, "="); // :替换=
    newObj = newObj.replace(/}/g, "&ZHENGDI"); // ZHENGDI
    newObj = newObj.replace(/{/g, "");
    console.log(newObj, "签名");
    return md5(newObj); // 对签名字段MD5加密

}
export default service;
```

3.3.3 API接口管理 `api.js`

统一管理API接口，接口名称，接口版本，请求方式

```
/* jshint esversion: 6 */
import request from "@/service/request"; //引入封装Axios文件
const apiVer = "/v1"; //接口版本控制
let dataJiami = true; //数据加密开关 一般登录验证码、登录、获取用户信息接口不需要加密
/**
 * 系统登陆模块接口
 * **/
export function loginPost(data) {
   return request({ url: apiVer + "/login", method: "post", data: data,
dataJiami });
}// 系统登陆，自定义属性dataJiami,版本不能大于0.18.0，否则不生效
export function verificationGet(data) {
   return request({ url: apiVer + "/yzm", method: "get", params: data, dataJiami
});
}// 登陆验证码
export function userInfoGet(data) {
    return request({ url: apiVer + "/user", method: "get", params: data,
dataJiami });
}// 获取用户信息

// 分组管理
```

```
/** **************** 分组管理*****************/
export function groupListGet(data) {
    return request({ url: apiVer + "/user_group", method: "get", params: data
});
}// 查询用户分组
export function groupAddPost(data) {
    return request({ url: apiVer + "/user_group", method: "post", data: data });
}// 添加用户分组
export function groupDelDelete(data) {
    return request({ url: apiVer + "/user_group", method: "delete", params: data
});
}// 删除用户分组
export function groupChangePut(data) {
    return request({ url: apiVer + "/user_group", method: "put", params: data
});
}// 修改用户分组
```

3.3.4 页面调用接口

```
data() {
    retrun {
        oQueryForms: {
            size: 10,
            form:1,
        }
    }
}
mounted() {
    this.apiGroupListGet();  // 分组列表
},
methods: {
    // 获取分组列表
    apiGroupListGet() {
        groupListGet(this.oQueryForms).then(response => {
            this.aBackData = response.data.data.body;
            this.iPageTotal = response.data.data.count;
        });
    },
}
```

3.4 粒子特效`vue-particles`

3.4.1 安装

```
npm install vue-particles --save-dev
```

3.4.2 配置main.js

```
mport VueParticles from 'vue-particles' //引入

Vue.use(VueParticles) //注册
```

3.4.2 页面使用

```
<vue-particles
class="parBg" color="#00D2FF" linesColor="#113C6D" shapeType="circle"
hoverMode="grab" clickMode="push":particleOpacity="0.7"
:particlesNumber="80":particleSize="6" :linesWidth="0"
:lineLinked="false":lineOpacity="0.4" :linesDistance="150"
:moveSpeed="2":hoverEffect="true" :clickEffect="true">
</vue-particles>
```

3.4.2 参数说明

参数	类型	默认值	说明
color	String	'#dedede'	粒子颜色
particleOpacity	Number	0.7	粒子透明度
particlesNumber	Number	80	粒子数量
shapeType	String	circle	可用的粒子外观类型有："circle","edge","triangle", "polygon","star"
particleSize	Number	80	单个粒子大小
linesColor	String	#dedede	线条颜色
linesWidth	Number	1	线条宽度
lineLinked	Boolean	true	连接线是否可用
lineOpacity	Number	0.4	线条透明度
linesDistance	Number	150	线条距离
moveSpeed	Number	3	粒子运动速度
hoverEffect	Boolean	true	是否有hover特效
hoverMode	String	“true”	可用的hover模式有: "grab", "repulse", "bubble"
clickEffect	布尔类型	true	是否有click特效
clickMode	String	true	可用的click模式有: "push", "remove", "repulse", "bubble"

四 常用数据处理

4.1 数据类型转换

常用数据类型有：字符串String、数字Number 、数组 Array、对象Object、布尔型Boolean

4.1.1 字符串(String)转换

4.1.1.1 字符串转换数字

parseInt() //转换为整数

parseFloat() //转换为浮点数

```
console.log(parseInt("中国人")) //NaN
console.log(parseFloat("中国人")) //NaN
console.log(parseInt("abc")) //NaN
console.log(parseFloat("abc")) //NaN
console.log(parseInt("abc123")) //NaN
console.log(parseFloat("abc123")) //NaN
console.log(parseInt("80.520abc")) //80
console.log(parseFloat("80.520abc")) //80.520
console.log(parseInt("520abc")) //520
console.log(parseFloat("520abc")) //520
```

4.1.1.2 字符串转换数组

```
let strOne = "我是中国人"
console.log([...strOne]); //[ "我", "是", "中", "国", "人" ]
let strTwo = "aa,bb,cc,dd";
console.log(strTwo.split(","));//[ "aa", "bb", "cc", "dd" ]
```

4.1.1.3 JSON字符串转换对象或数组

```
let strArr = '[{"name":"周杰伦","age":50},{"name":"苏丹","age":20}]';
console.log(JSON.parse(strArr));
/*[
    { name: "周杰伦", age: 50 },
    { name: "苏丹", age: 20 }
]*/
let strObj = '{"name":"杨紫","age":33}'
console.log(JSON.parse(strObj)); //
/*{
    age: 33
    name: "杨紫"
}*/
let strArrOne = "[123,213,2,12,23]";
console.log(JSON.parse(strArrOne)) //[123,213,2,12,23]
```

4.1.1.4 字符串转换布尔型

```
console.log(Boolean("")); //false
console.log(Boolean("我有值")); //true
```

4.1.2 对象Object转换

4.1.2.1 对象转换字符串

```
let objOne = {
    name: "杨紫",
    age: 33
}
console.log(JSON.stringify(objOne)); //{"name":"杨紫","age":33}
let objArr = [
    { name: "周杰伦" ,age: 50},
    { name: "苏丹" ,age: 20},
]
console.log(JSON.stringify(objArr)) //[{"name":"周杰伦","age":50},{"name":"苏丹","age":20}]
```

4.1.3 数组Array转换

4.1.3.1 数组转换字符串

```
let arrOne = ["A","BB","CC"];
console.log(arrOne.toString()); //A,BB,CC
console.log(arrOne.join("")); //ABBCC
console.log(arrOne.join("-")); //A-BB-CC
console.log(arrOne.toLocaleString()); //A,BB,CC
console.log(JSON.stringify(arrOne)) //["A","BB","CC"]
```

4.1.4 数字Number转换

4.1.4.1 数字转换字符串

```
let num1 = 1000;
console.log(num1.toString()); //1000
console.log(num1+""); //1000
```

4.1.4.2 数字转换布尔型

```
console.log(Boolean(1)); //true
console.log(Boolean(0)); //false
console.log(Boolean(-1)); //true
```

4.1.4.3 数字转换数组

```
let num1 = 200000;
console.log(Array.of(num1)) //[200000]
```

4.1.5 布尔型Boolean转换

4.1.5.1 布尔型转换字符串

```
let bTrue = true;
let bFalse = false
console.log(bTrue.toString()) //true
console.log(bFalse.toString()) //false
console.log(bTrue.toLocaleString()) //true
console.log(bFalse.toLocaleString()) //false
```

4.1.5.2　布尔型转换数字

```
let bTrue = true;
let bFalse = false
console.log(Number(true)); //1
console.log(Number(false)); //0
```

4.2 数据排序

4.2.1 数组排序

4.2.1.1　普通数组

```
let arrDataA = [10, 80, 70, 20, 30, 60, 40, 50, 100, 90];
let arrDataB = [10, 80, 70, 20, 30, 60, 40, 50, 100, 90];
let arrDataC = ["A","D","B","E","C","G","F"];
let arrDataD = ["A","D","B","E","C","G","F"];
let arrDataE = ["age","name","address","height","width"];
let arrDataF = ["age","name","address","height","width"];
let newArrA = arrDataA.sort(); //升序
let newArrB = arrDataB.sort(function(a, b){return b - a}); //降序
let newArrC = arrDataC.sort();
let newArrD = arrDataD.sort(function(a, b){return b - a});
let newArrE = arrDataE.sort();
let newArrF = arrDataF.sort(function(a, b){return b - a});
console.log("升序：",newArrA);//[ 10, 100, 20, 30, 40, 50, 60, 70, 80, 90 ]
console.log("降序：",newArrB);//[ 100, 90, 80, 70, 60, 50, 40, 30, 20, 10 ]
console.log("升序：",newArrC);//[ "A", "B", "C", "D", "E", "F", "G" ]
console.log("降序：",newArrD);//[ "F", "G", "C", "E", "B", "D", "A" ]
console.log("升序：",newArrE);//[ "address", "age", "height", "name", "width" ]
console.log("降序：",newArrF);//[ "width", "height", "address", "name", "age" ]
```

4.2.1.2　数组对象

制定字段排序

例：按照age排序

```
    let objDataA = [
        { name: "秋儿", age: 30, sex: "女", address: "成都" },
        { name: "潘儿", age: 25, sex: "男", address: "北京" },
        { name: "吉儿", age: 40, sex: "男", address: "青岛" },
        { name: "佳儿", age: 15, sex: "女", address: "天津" },
        { name: "芳儿", age: 60, sex: "男", address: "湖南" },
    ]
    let objDataB = [
        { name: "秋儿", age: 30, sex: "女", address: "成都" },
        { name: "潘儿", age: 25, sex: "男", address: "北京" },
        { name: "吉儿", age: 40, sex: "男", address: "青岛" },
        { name: "佳儿", age: 15, sex: "女", address: "天津" },
        { name: "芳儿", age: 60, sex: "男", address: "湖南" },
    ]
    let newObjA =objDataA.sort(function(a, b){
        return b.age - a.age
    });//降序
    let newObjB =objDataB.sort(function(a, b){
        return a.age - b.age
    });//升序
    console.log(newObjA);
    //降序排序结果
    [
        { name: "芳儿", age: 60, sex: "男", address: "湖南" },
        { name: "吉儿", age: 40, sex: "男", address: "青岛" },
        { name: "秋儿", age: 30, sex: "女", address: "成都" },
        { name: "潘儿", age: 25, sex: "男", address: "北京" },
        { name: "佳儿", age: 15, sex: "女", address: "天津" },
    ]
    console.log(newObjB);
    //升序排序结果
    [
        { name: "佳儿", age: 15, sex: "女", address: "天津" },
        { name: "潘儿", age: 25, sex: "男", address: "北京" },
        { name: "秋儿", age: 30, sex: "女", address: "成都" },
        { name: "吉儿", age: 40, sex: "男", address: "青岛" },
        { name: "芳儿", age: 60, sex: "男", address: "湖南" },
    ]
```

4.2.2 对象排序

4.2.2.1 按对象属性值排序

```
let objData = {
    "张三": 40,
    "秋子": 20,
    "马苏": 23,
    "洋子": 29,
    "仓子": 28,
}
let newKeyList = Object.keys(objData).sort(function(a,b){
    return objData[b]-objData[a]
})  //排序健名
let newObj = {};
newKeyList.forEach(item=>{
    newObj[item] = objData[item];
});
console.log(newObj);
//{ "张三": 40, "洋子": 29, "仓子": 28, "马苏": 23, "秋子": 20 }
```

4.2.2.2 按对象键值排序

```
let objData = {
    name: "秋儿",
    sex: "男",
    address: "成都",
    nation: "汉族",
    nationality: "中国",
    birthday: "1992-02-09"
}
let newKeyList = Object.keys(objData).sort()   //升序排序健名
let newKeyList = Object.keys(objData).sort(function(a,b){return
b.localeCompare(a)})   //降序排序健名
let newObj = {};
newKeyList.forEach(item=>{
    newObj[item] = objData[item];
});
console.log(newObj);
//{ address: "成都", birthday: "1992-02-09", name: "秋儿", nation: "汉族",
nationality: "中国", sex: "男" }
```

4.3 API返回数据处理

4.3.1 返回数据格式化

```
let apiBackData = [
    { name: "芳儿", age: 60, sex: 0, address: "湖南" },
    { name: "吉儿", age: 40, sex: 1, address: "青岛" },
    { name: "秋儿", age: 30, sex: 2, address: "成都" },
    { name: "潘儿", age: 25, sex: 1, address: "北京" },
    { name: "佳儿", age: 15, sex: 0, address: "天津" },
]
let sexType = {
    0: "女",
    1: "男",
    2: "中性",
}
let newData = apiBackData.map(item => {
    return {
        ...item,
        sexFormat:sexType[item.sex],
    }
})
console.log(newData);
//处理结果
[
    { name: "芳儿", age: 60, sex: 0, sexFormat:"女", address: "湖南" },
    { name: "吉儿", age: 40, sex: 1, sexFormat:"男", address: "青岛" },
    { name: "秋儿", age: 30, sex: 2, sexFormat:"中性", address: "成都" },
    { name: "潘儿", age: 25, sex: 1, sexFormat:"男", address: "北京" },
    { name: "佳儿", age: 15, sex: 0, sexFormat:"女", address: "天津" },
]
```

4.3.2 取指定字段数据

```
let apiBackData = [
    { name: "芳儿", age: 60, sex: 0, address: "湖南" },
    { name: "吉儿", age: 40, sex: 1, address: "青岛" },
    { name: "秋儿", age: 30, sex: 2, address: "成都" },
    { name: "潘儿", age: 25, sex: 1, address: "北京" },
    { name: "佳儿", age: 15, sex: 0, address: "天津" },
]

let newData = apiBackData.map(item => {
    return {
        name: item.name,
        age: item.age
    }
})
console.log(newData);
//处理结果
[
    { name: "芳儿", age: 60 },
    { name: "吉儿", age: 40 },
    { name: "秋儿", age: 30 },
    { name: "潘儿", age: 25 },
    { name: "佳儿", age: 15 },
]
```

4.3.3 求数组之和

```
/**
 * 函数功能  求数组之和
 * @name ArrSum
 * @param data Array
 * @return int 和
 */
let arrData = [10, 20, 30 , 40, 50, 60, 70];
console.log(arrSum(ArrData));//280
function ArrSum(data){
    let sum=0;
    arrData.forEach(item => {
        sum+=item;
    })
    return sum;
}
```

4.3.4 UTF8和Base64互转

```
let objMan = {
    name: "邓紫棋",
    age: 20,
    sex: "女",
    address: "成都"
}
/**
 * 函数功能  数据转换Base64
 * @name ToBase64
 * @param data all
 * @return base64字符串
**/
function Utf8ToBase64(data) {
    return window.btoa(unescape(encodeURIComponent(JSON.stringify(data))));
}
let baseString = Utf8ToBase64(objMan)
```

```
console.log(baseString)
//eyJuYW1lIjoi6YKT57Sr5qOLIiwiYWdlIjoyMCwic2V4Ijoi5aWzIiwiYWRkcmVzcyI6IuaIkOmDvS
J9
/**
     * 函数功能  数据转换Base64
     * @name ToBase64
     * @param baseStr base64的字符串
     * @return 转换前数据
     */
function Base64ToUtf8( baseStr ) {
    return JSON.parse(decodeURIComponent(escape(window.atob( baseStr ))));
}
console.log(Base64ToUtf8(baseString));
/*
 {
    name: "邓紫棋",
    age: 20,
    sex: "女",
    address: "成都"
}
*/
```

4.3.5 时间戳格式化

```
/**
 * 函数功能
 * @name dateFormat
 * @param timestamp int 时间戳
 * @return 返回年月日时分秒和上午下午对象
 */

function dateFormat (timestamp="") {
    let date="";
    if(timestamp === ""){
        date = new Date();
    } else {
        timestamp=parseInt(timestamp);
        //时间戳为10位需*1000，时间戳为13位的话不需乘1000
        if (timestamp.toString().length == 10){
            date = new Date(timestamp * 1000);
        } else if(timestamp.toString().length == 13) {
            date = new Date (timestamp);
        } else{
            return "传入参数不合法,传入参数为10或13位时间戳,为空返回当前系统时间"
        }
    }
    let dataTime = {
        dateY: date.getFullYear(), //年
        dateM: date.getMonth() + 1 < 10 ? '0' + (date.getMonth() + 1) :
date.getMonth() + 1, //月
        dateD: date.getDate() < 10 ? "0" + date.getDate() : date.getDate(), //日
        timeH: date.getHours() < 10 ? "0" + date.getHours() : date.getHours(),
//时
        timeM: date.getMinutes() < 10 ? "0" + date.getMinutes() :
date.getMinutes(), //分
        timeS: date.getSeconds() < 10 ? "0" + date.getSeconds() :
date.getSeconds(), //秒
```

```
        timeMs: date.getMilliseconds() < 10 ? "0" + date.getMilliseconds() :
date.getMilliseconds(), //毫秒
        timeAp: date.getHours() >= 12 ? "下午" : "上午", //中文上下午
        timeEap: date.getHours() >= 12 ? "PM" : "AM" //英文上下午
    };
    return dataTime;
  }
console.log(dateFormat(1573630199));
/*
{
    dateD: 13
    dateM: 11
    dateY: 2019
    timeAp: "下午"
    timeEap: "PM"
    timeH: 15
    timeM: 29
    timeMs: "00"
    timeS: 59
}*/
```

4.4 vue路由表后台数据处理

4.4.1 菜单数组对象转换Vue对象

```
/**
 * 函数功能
 * @name filterAsyncRouter
 * @param array数组用户权限路由表
 * @return 转换vue对象
 * @ 使用地方：后台获取路由，
 */
let routerData= [
    { path: "/Home",name: "Home",permis: {add: true,del: true,change:
true,query: true},component: "Home/Home"},
    { path: "/Warning",name: "Warning",permis: {add: true,del: true,change:
true,query: true},component: "Warning/Warning",},
    { path: "/Terminal",name: "Terminal",permis: {add: true,del: true,change:
true,query: true},component: "Terminal/Terminal"},
    { path: "/Analyze",name: "Analyze",permis: {add: true,del: true,change:
true,query: true}, component: "Analyze/Analyze"},
    {
        path: "/Rule",name: "Rule",permis: {add: true,del: true,change:
true,query: true},component: "Rule/Rule",redirect: "/Rule/RuleManage",
        "children": [
            { path: "/Rule/RuleManage",name: "RuleManage",permis: {add:
true,del: true,change: true,query: true},component:
"Rule/RuleManage/RuleManage"},
            { path: "/Rule/RuleReal",name: "RuleReal",permis: {add: true,del:
true,change: true,query: true},component: "Rule/RuleReal/RuleReal"},
            { path: "/Rule/RuleTask",name: "RuleTask",permis: {add: true,del:
true,change: true,query: true},component: "Rule/RuleTask/RuleTask"}
        ]
    },
    {
        path: "/System",name: "System",permis: {add: true,del: true,change:
true,query: true},component: "System/System",redirect: "/System/SystemSet",
        "children": [
            { path: "/System/SystemSet",name: "SystemSet",permis: {add:
true,del: true,change: true,query: true},component:
"System/SystemSet/SystemSet"},
            { path: "/System/SystemUser",name: "SystemUser",permis: {add:
true,del: true,change: true,query: true},component:
"System/SystemUser/SystemUser"},
            { path: "/System/SystemGroup",name: "SystemGroup",permis: {add:
true,del: true,change: true,query: true},component:
"System/SystemUser/SystemGroup"},
            { path: "/System/SystemLog",name: "SystemLog",permis: {add:
true,del: true,change: true,query: true},component:
"System/SystemLog/SystemLog"},
            { path: "/System/SystemAbout",name: "SystemAbout",permis: {add:
true,del: true,change: true,query: true},component:
"System/SystemAbout/SystemAbout"}
        ]
    }
];
let routers = filterAsyncRouter(routerData);
this.$router.addRoutes(getRoutes); // 动态添加路由VUE
console.log(routers);
/* [
      {"path": "/Home","name": "Home","permis": {"add": true,"del":
true,"change": true,"query": true},"component": "@/view/Home/Home.vue"},
      {"path": "/Warning","name": "Warning","permis": {"add": true,"del":
true,"change": true,"query": true},"component": "@/view/Warning/Warning.vue"},
      {"path": "/Terminal","name": "Terminal","permis": {"add": true,"del":
true,"change": true,"query": true},"component": "@/view/Terminal/Terminal.vue"},
      {"path": "/Analyze","name": "Analyze","permis": {"add": true,"del":
true,"change": true,"query": true},"component": "@/view/Analyze/Analyze.vue"},
      {
        "path": "/Rule","name": "Rule","permis": {"add": true,"del":
true,"change": true,"query": true},"component":
"@/view/Rule/Rule.vue","redirect": "/Rule/RuleManage",
        "children": [
          {"path": "/Rule/RuleManage","name": "RuleManage","permis": {"add":
true,"del": true,"change": true,"query": true},"component":
"@/view/Rule/RuleManage/RuleManage.vue"},
          {"path": "/Rule/RuleReal","name": "RuleReal","permis": {"add":
true,"del": true,"change": true,"query": true},"component":
"@/view/Rule/RuleReal/RuleReal.vue"},
          {"path": "/Rule/RuleTask","name": "RuleTask","permis": {"add":
true,"del": true,"change": true,"query": true},"component":
"@/view/Rule/RuleTask/RuleTask.vue"}
        ]
      },
      {
        "path": "/System","name": "System","permis": {"add": true,"del":
true,"change": true,"query": true},"component":
"@/view/System/System.vue","redirect": "/System/SystemSet",
        "children": [
          {"path": "/System/SystemSet","name": "SystemSet","permis": {"add":
true,"del": true,"change": true,"query": true},"component":
"@/view/System/SystemSet/SystemSet.vue"},
          {"path": "/System/SystemUser","name": "SystemUser","permis": {"add": true,"del": true,"change": true,"query": true},"component": "@/view/System/SystemUser/SystemUser.vue"},
          {"path": "/System/SystemGroup","name": "SystemGroup","permis": {"add": true,"del": true,"change": true,"query": true},"component": "@/view/System/SystemUser/SystemGroup.vue"},
          {"path": "/System/SystemLog","name": "SystemLog","permis": {"add": true,"del": true,"change": true,"query": true},"component": "@/view/System/SystemLog/SystemLog.vue"},
          {"path": "/System/SystemAbout","name": "SystemAbout","permis": {"add": true,"del": true,"change": true,"query": true},"component": "@/view/System/SystemAbout/SystemAbout.vue"}
        ]
      }
    ]
    */

function filterAsyncRouter(asyncRouterMap) { // 遍历后台传来的路由字符串，转换为组件对象
    const accessedRouters = asyncRouterMap.filter(route => {
        if (route.component) {
            // route.component=
            //require("@/view/" + route.component+ ".vue").default; //vue真实使用
            route.component= "@/view/" + route.component+ ".vue"; //演示
        }
        if (route.children && route.children.length) {
            route.children = filterAsyncRouter(route.children);
        }
        return true;
    });
    return accessedRouters;
}
```

4.4.2 获取指定菜单指定属性值

```
    let routerData= [
        { path: "/Home",name: "Home",permis: {add: true,del: true,change: true,query: false},component: "Home/Home"},
        { path: "/Warning",name: "Warning",permis: {add: true,del: true,change: true,query: true},component: "Warning/Warning",},
        { path: "/Terminal",name: "Terminal",permis: {add: true,del: true,change: true,query: true},component: "Terminal/Terminal"},
        { path: "/Analyze",name: "Analyze",permis: {add: true,del: true,change: true,query: true}, component: "Analyze/Analyze"},
        {
             path: "/Rule",name: "Rule",permis: {add: true,del: true,change: true,query: true},component: "Rule/Rule",redirect: "/Rule/RuleManage",
            "children": [
                { path: "/Rule/RuleManage",name: "RuleManage",permis: {add: true,del: true,change: true,query: true},component: "Rule/RuleManage/RuleManage"},
                { path: "/Rule/RuleReal",name: "RuleReal",permis: {add: true,del: true,change: true,query: true},component: "Rule/RuleReal/RuleReal"},
                { path: "/Rule/RuleTask",name: "RuleTask",permis: {add: true,del: true,change: true,query: true},component: "Rule/RuleTask/RuleTask"}
            ]
        },
        {
            path: "/System",name: "System",permis: {add: true,del: true,change:
true,query: true},component: "System/System",redirect: "/System/SystemSet",
            "children": [
                { path: "/System/SystemSet",name: "SystemSet",permis: {add:
true,del: true,change: true,query: true},component:
"System/SystemSet/SystemSet"},
                { path: "/System/SystemUser",name: "SystemUser",permis: {add:
true,del: true,change: true,query: true},component:
"System/SystemUser/SystemUser"},
                { path: "/System/SystemGroup",name: "SystemGroup",permis: {add:
true,del: true,change: true,query: true},component:
"System/SystemUser/SystemGroup"},
                { path: "/System/SystemLog",name: "SystemLog",permis: {add:
true,del: true,change: true,query: true},component:
"System/SystemLog/SystemLog"},
                { path: "/System/SystemAbout",name: "SystemAbout",permis: {add:
false,del: false,change: false,query: false},component:
"System/SystemAbout/SystemAbout"}
            ]
        }
    ];

 /**
 * 函数功能  根据不同页面返回相应页面权限
 * @name getPermiss
 * @param data Array 菜单数据
 * @param ifKey String 判断键名
 * @param backKey String 返回数据的键名
 * @param keyValue String 判断键值
 * @return 返回指定菜单的指定属性值
 */
function getPermiss(arrData,ifKey,backKey,keyValue = "Home") {
    console.log(backKey);
    let result = null;
    for (let index = 0; index < arrData.length; index++) {
        const element = arrData[index];
        if (element[ifKey] === keyValue) {
            result = element[backKey];
        }
        if (Reflect.has(element, "children") && result == null) {
            result = getPermiss(element.children, ifKey, backKey, keyValue);
        }
    }
    return result;
}

let Home=getPermiss(routerData,"name","permis","Home")
let SystemAbout=getPermiss(routerData,"name","permis","SystemAbout")
console.log(Home); // { add: true, del: true, change: true, query: false }
console.log(SystemAbout); //{ add: false, del: false, change: false, query:
false }
```