Vue项目规范

Vue 项目规范

> :snake: PascalCase 即大写字母开头的驼峰命名法

一、组件规范

1.1 组件名应该是多个单词

组件应该是由多个单词组成(大于等于2)，且命名规范为 PascalCase 格式。这样是为了避免与现在以及未来的HTML元素冲突。

推荐 :white_check_mark:

```
export default {
  name : 'TodoItem'
}
```

不推荐 :negative_squared_cross_mark:

```
export default {
  name : 'Todo'
}
export default {
  name : 'todo'
}
```

1.2 组件文件名

组件文件名应该是 kebab-case 格式

推荐 :white_check_mark:

```
components/
|- my-component.vue
```

不推荐 :negative_squared_cross_mark:

```
components/
|- myComponent.vue
|- MyComponent.vue
```

1.3 基础组件命名

基础组件文件名为 base 开头，使用完整单词而不是缩写

```
components/
|- base-button.vue
```

反例

```
components/
|- MyButton.vue
|- VueTable.vue
```

1.4 子组件命名

子组件若与父组件紧密耦合，则必须以父组件作为前缀命名

推荐 :white_check_mark:

```
components/
|- todo-list.vue
|- todo-list-item.vue
|- todo-list-item-button.vue
```

不推荐 :negative_squared_cross_mark:

```
components/
|- TodoList.vue
|- TodoItem.vue
|- TodoButton.vue
```

1.5 模板标签书写

在模板标签中使用组件，应该使用 PascalCase 格式，并且使用自闭合组件

推荐 :white_check_mark:

```
<MyComponent />
<Row><table :colum="data" /></Row>
```

不i推荐 :negative_squared_cross_mark:

```
<my-component />
<row><table :column="data"/></row>
```

1.6 组件data必须是函数

因为函数可以保证每次返回的都是一个新的对象，否则，将所有组件共用一个对象

1.7 Props 定义必须详细

- 必须使用 camelCase 命名
- 必须指定类型

- 必须加上注释，表明其含义
- 必须加上 required 或者 default，或者二选一
- 如果有业务需要，必须加上 validator 验证

```
export default {
  props : {
  status : {
    type : String,
   required : true,
   validator(value){
    return [
     'succ',
     'info',
     'error'
    ].indexOf(value) > -1
   }
  },
  userLevel : {
    type : String,
   required : true
  }
 }
}
```

1.8 为样式设置作用域

`<style>` 标签添加 `scoped` 属性

1.9. 特性元素较多，必须换行

推荐 :white_check_mark:

```
<data-time
 v-model="start_time"
 :width="180"
 @change="oncheck"
 @focus="onclaer"
 :options="start"
 @blur="onBlur"
></data-time>
```

不推荐 :negative_squared_cross_mark:

```
<data-time v-model="start_time" :width='180' @change="oncheck" @focus="onclaer" :options="start" @blur="onBlur"></data-time>
```

1.10. 模板中使用简单的表达式

组件模板应该只包含简单的表达式，负责的表达式应该重构为计算属性或方法。复杂表达式会让你的模板变得不那么声明式。 我们应该尽量描述应该出现的是什么， 而非如何计算那个值。 而且计算属性和方法使得代码可以重用。

推荐 :white_check_mark:

```
<div v-if="status == 3 || status == 4 || !hasPlayBack">直播结束</div>

<script>
export default {
   computed : {
   isLiveEnd() {
      return this.status == 3 || this.status == 4 || !this.hasPlayBack
   }
  }
}
</script>
```

不推荐 :white_check_mark:

```
<div v-if="status == 3 || status == 4 || !hasPlayBack">直播结束</div>
```

1.11. 指令使用缩写形式

推荐 :white_check_mark:

```
<input
  @input="onInput"
  @focus="onFocus"
>
```

不推荐 :negative_squared_cross_mark:

```
<input
  @input="onInput"
  @focus="onFocus"
>
```

1.12. 标签顺序保持一致

单文件组件总是让标签顺序保持为

```
<template></template>
<script></script>
<style></style>
```

二、Vue-Router 规范

2.1 页面跳转使用路由传参

不要保存在 `store` 中，而是直接用 `query` 带过去，因为保存在 `store` 中，会因为刷新页面导致数据丢失

2.2 使用路由懒加载

```
{
  path: '/uploadAttachment',
  name: 'uploadAttachment',
  meta: {
    title: '上传附件'
  },
  component: () => import('@/view/components/uploadAttachment/index.vue')
}
```

2.3 router命名规范

- path、childrenPoints 命名使用 kebeb-case 命名规范(与vue的目录结构保持一致)
- name 命名使用 PascalCase 命名，但是要与component组件名保持一致(因为要保持keep-alive特性，所以必须一致)
- path 命名使用 kebeb-case 命名，而且必须是 `/` 开头，另外子路由也需要使用 `/` 完整路径(这样查找会比较容易)

```
const routes = [
  {
      path : '/login',
    name : 'File',
    component : Man,
    meta : {...},
    children : [
      {
        path : '/file/file-list',
        name : 'FileList',
        component : ()=> import('@/views/file/file-list.vue')
      }
    ]
  }
]
```

三、项目目录规范

3.1 前后端命名统一

vue 项目中所有命名一定要与后端命名统一。

例如：后端 `privilege`，前端无论是 route 、store 、api 等都必须使用 `privilege`

3.2 使用 vue-cli 或者 vite

脚手架

3.3 目录说明

目录名按照下面命名，目录下的文件，没有特别原因，统一用 kebab-case 命名

```
src           源码目录
|-- api     所有api接口
|-- assets 静态资源，images、icons、styles等
|-- components 公共组件
|-- config 配置信息
|-- constants 常量信息，项目所有Enum，全局变量等
|-- directives 自定义指令
|-- filters 过滤器，全局工具
|-- datas 模拟数据，临时存放
|-- lib 外部引用的插件存放以及修改文件
|-- mock 模拟接口，临时存放
|-- plugins 插件，全局使用
|-- router 路由，统一管理
|-- store vuex，统一管理
|-- themes 自定义样式主题
|-- views 视图目录
|   |--role role 模块名
|   |-- |-- role-list.vue role 列表页面
```

api 目录

- 文件、变量名要与后端保持一致
- 此目录对应后端API接口，一个Controller对应一个js文件。若项目比较大，则按照业务划分子目录，并且与后端保持一致
- api 的方法命名尽量与后端 api url 保持语义的高度一致性
- 对于 api 的每个方法要添加注释，注释要与后端 swagger 文档保持一致

```
// 后端 EmployeeController.java
/**
 * /employee/add
 * /employee/delete/{id}
 * /employee/update
*/
function addEmplyoee(data) {
  return request({
        url : '/employee/add',
    data,
    method : 'post'
  })
}
...
```

assets 目录

静态资源目录，里面存放 images、styles、icons等资源，也是用 kebab-case 命名

```
|assets
|-- icons
|-- images
|     |-- background-color.png
|     |-- upload-header.png
|-- styles
```

components 目录

组件存放目录，目录命名为 kebab-case ， 组件命名规则也是 kebab-case

```
|components
|-- error-log
|    |-- index.vue
|    |-- index.scss
```

constants 目录

此目录存放项目的所有常量，如果常量在vue中使用，请使用 vue-enum 插件

```
|constants
|-- index.js
|-- role.js
|-- employee.js
```

router 与 store

这两个目录一定要按照业务拆分，不能放到一个js文件中

- router 尽量按照 views 中的结构保持一致
- store 按照业务进行拆分不同的js

views目录

命名要与后端、router、api保持一致

```
|views
| |-- role 组件模块名
| |    |-- role-list.vue role列表页面
| |    |-- role-add.vue role新建页面
| |    |-- role-update.vue role更新页面
| |    |-- index.less role模块样式
| |    |-- components role 模块 通用模块组件文件夹
| |    |    |-- role-header.vue role头部组件
```

3.4 其他说明

注释

整理必须加注释的地方

- 公共组件使用说明
- api 目录的接口 js 文件必须添加注释
- store 中 state，mutation，action 等必须添加注释
- vue 文件中的 methods，每个 method 必须添加注释
- vue 文件中的 data，非常见单词需要添加注释

手动操作DOM

因为已经使用Vue数据驱动框架，尽量通过数据驱动的形式更新DOM，不到万不得已不要手动操作DOM，包括

- 增删改DOM元素

- 更改样式
- 添加事件

删除无效代码

例如`console.log`等废弃或调试的代码