VUE 代码风格指南

目录结构

```
my-project-name
|- assets
|- components
|- pages
|- utils
|- app.vue
|- index.js
|- index.html
```

目录结构

```
my-project-name
|- assets
|- components
|- pages
|- utils
|- app.vue
|- index.js
|- index.html
```

目录命名/文件命名

推荐所有目录命名采用 kebab-case（短横线连接式）

推荐所有文件命名采用 camelCase（小驼峰式命名法 —— 首字母小写）

推荐所有vue组件命名采用 PascalCase（大驼峰式命名法 —— 首字母大写）

它看上去就像是这样

```
my-project-name
|- assets
|- - background-container.png
|- components
|- - header
|- - - Index.vue
|- - footer
|- - - Index.vue
|- - list
|- - - - LstHelp.vue
|- - - - Index.vue
|- pages
|- - Home.vue
|- - Details.vue
|- utils
```

它看上去就像是这样

```
|- - background-container.png
|- components
|- - header
|- - - Index.vue
|- - footer
|- - - Index.vue
|- - list
|- - - - LstHelp.vue
|- - - - Index.vue
|- pages
|- - Home.vue
|- - Details.vue
|- utils
|- - ua.js
|   index.js
```

VUE 单文件组件引用

```
<template>
  // 组件内没有内容
  <empty-component />

  // 组件标签之间有内容
  <container-component>
      This is content.
  </container-component>
</template>

<script>
import EmptyComponent from '../path';
import ContainerComponent from '../../path';

export default {
```

VUE 单文件组件引用

```
<template>
  // 组件内没有内容
  <empty-component />

  // 组件标签之间有内容
  <container-component>
      This is content.
  </container-component>
</template>

<script>
import EmptyComponent from '../path';
import ContainerComponent from '../../path';

export default {
```

VUE 单文件组件引用

```
  </container-component>
</template>

<script>
import EmptyComponent from '../path';
import ContainerComponent from '../../path';

export default {
  name: 'MyComponent',
  components: {
    EmptyComponent,
    ContainerComponent
  }
}
```

VUE 单文件属性声明顺序

```
<script>
export default {
  name: 'MyComponent',
  components: { },
  mixins: [],
  props: { },
  data() { },
  provide() { },
  inject: [],
  watch: { },
  computed: { },
  activated(): { },
  deactivated(): { },
  beforeCreate() { },
  created() { }
```

自定义事件

@myEvent 会被自动转化成 @myevent

事件处理函数使用handle+动词

```
this.$emit('on-change');
```

```
<my-component @on-change="handleChange" />
```

变量

```
1 // bad
2 var curTitle = 'banxian';
3
4 // good
5 let curTitle = 'banxian';
```

常量

```
1 // bad
2 const maxCount = 10;
3
4 // good
5 const MAX_COUNT = 10;
```

引号混用

html模板中使用""

js中使用''

```
// bad
let name = "banxian";
let profession = 'front-end';

// good
let name = 'banxian';
let profession = 'front-end';
```

```
// bad
<my-component data-attributes='Single quotes' />

// good
<my-component data-attributes="Double quotes" />
```

组件DATA声明为函数

```
// bad
export default {
  data: {
    // ...
  }
}

// good
export default {
  data() {
    return {
      // ...
    }
  }
}
```

组件PROPS声明详细

```
// bad
export default {
  props: ['name', 'age']
}

// good
export default {
  props: {
    name: {
      type: String,
      default: '半仙君_'
    },
    age: {
      type: Number,
      default: 18
```

组件COMPUTED一定要返回值

```
// bad
export default {
  computed: {
    fullName() {
      if (!this.firstName) return this.lastName;
      if (this.firstName && this.lastName) return this.firs
    }
  }
}

// good
export default {
  computed: {
    fullName() {
      if (!this.firstName) return this.lastName;
```

V-FOR要设置KEY

并且最好有id作为key，而不是idx

```
// bad
<ul>
  <li/
    v-for="(todo, idx) in todos"
    :key="idx"
  >
    {{ todo.text }}
  </li>
</ul>

// good
<ul>
  <li/
    v-for="(todo, idx) in todos"
    :key="todo.id"
```

V-IF和V-FOR互斥

最好在另一个包裹元素上使用v-if

或者对v-for的列表进行过滤

```
// bad
<ul>
  <li/
    v-for="(todo, idx) in todos"
    v-if="showCurTodo"
    :key="todo.id"
  >
    {{ todo.text }}
  </li>
</ul>

// good
<ul>
  <li/
    v-for="(todo, idx) in todos"
```

多个属性时多行来写

```
// bad
<li class="item" v-for="(todo, idx) in todos" @click="handl
  {{ todo.text }}
</li>

// good
<li
  class="item"
  @click="handleClick"
  v-for="(todo, idx) in todos"
  :key="todo.id"
>
  {{ todo.text }}
</li>
```

模板中应该只包含简单的表达式

```
// bad
{{
  fullName.split(' ').map((word) => {
    return word[0].toUpperCase() + word.slice(1)
  }).join(' ')
}}

// good
{{ normalizedFullName }}

computed: {
  normalizedFullName: function () {
    return this.fullName.split(' ').map(function (word) {
      return word[0].toUpperCase() + word.slice(1)
    }).join(' ')
```

组件名为多个单词

这能有效预防与html标准标签冲突

```
// bad
Vue.component('Todo', {
  // ...
})

// good
Vue.component('TodoList', {
  // ...
})
```

组件样式声明尽可能设置作用域

由于组件化开发很容易样式命名冲突

```
// bad
button {
  // ...
}

// good
.todo-list-btn {
  // ...
}
.todo-list-btn .list-item {
  // ...
}
```

单文件的顶级元素顺序

```
<!-- ComponentA.vue -->
<template>...</template>
<script>/* ... */</script>
<style>/* ... */</style>

<!-- ComponentB.vue -->
<template>...</template>
<script>/* ... */</script>
<style>/* ... */</style>
```

感谢大家 晚安