**内容回顾:动画-自定义class类名-混合-自定义指令**

**今天内容:vue插件开发-vue-touch-vue-router**

**一，Vue插件**

<https://cn.vuejs.org/v2/guide/plugins.html>

**1、概述**

简单来说，插件就是指对Vue的功能的增强或补充。

比如说，让你在每个单页面的组件里，都可以调用某个方法，或者共享使用某个变量，或者在某个方法之前执行一段代码等.

**2、使用方法**

总体流程应该是：

【声明插件】——【写插件】——【注册插件】——【使用插件】

写插件和声明插件是同步的，然后注册到Vue对象中（不用担心重复注册），最后在写Vue组件的时候使用写的插件

**声明插件**

先写一个js文件，这个js文件就是插件文件，里面的基本内容如下：

/\*  说明：

 \*  插件文件：service.js

 \* \*/

export **default** {

  install: **function** (Vue, options) {

    // 添加的内容写在这个函数里面（有4种方法实现，见下文）

  }

};

其中install的第一个参数Vue表示的是Vue的实例，第二个参数表示的是一些设置选项。

Vue实例好理解，就是Vue对象。

而options设置选项就是指，在调用这个插件时，可以传一个对象。

例如这个对象有一个属性float，然后在写插件的一个方法/变量时，我需要输出一个数字，然后写一个if判断语句，

假如options.float为true时，输出浮点数；

假如为false或undefined（即没传参）时，输出为整数。

具体怎么添加，之后再说。

**注册插件**

如果使用过Vue-router，就很好理解，通过import引入后，然后通过 Vue.use(插件名) 注册插件；

例如，我们通常在main.js里引入各种东西，并且组件的根实例也在这里

//main.js

import Vue from 'vue'

import App from './App.vue'

//关键是这两行

import service from './service.js' //1.引入一个js文件，保持到一个模块（service，就是一个变量）

Vue.use(service) //2.使用插件

**new** Vue({

  el: '#app',

  render: (h) => h(App)

})

如代码中注释所说，关键是通过import导入service文件，然后在创建根组件之前，让Vue对象通过use方法来注册插件service。

通过这样简单的两步，就可以使用插件了。

**写插件、使用插件**

按照官方文档，写插件有4种方法，先给出官方的代码：

//以下内容都是添加到上面install的函数里面的

// 1. 添加全局方法或属性

Vue.myGlobalMethod = **function** () {

  // 逻辑...

}

// 2. 添加全局资源：指令，过渡器，data,....

Vue.directive('my-directive', {

  bind (el, binding, vnode, oldVnode) {

    // 逻辑...

  }

  ...

})

// 3. 注入组件

Vue.mixin({

  created: **function** () {

    // 逻辑...

  }

  ...

})

// 4. 添加实例方法

Vue.prototype.$myMethod = **function** (options) {

  // 逻辑...

}

先给出最常用的：【4. **添加实例方法**】的写法和使用方法

**3.1【添加Vue实例的方法或属性】**

1、核心思想：

通过prototype来添加方法和属性。这个插件的方法，属性只能在Vue实例中使用（通过this调用）。

2、写：

//让输出的数字翻倍，如果不是数字或者不能隐式转换为数字，则输出null

Vue.prototype.doubleNumber = **function** (val) {

  **if** (**typeof** val === 'number') {

    **return** val \* 2;

  } **else** **if** (!isNaN(Number(val))) {

    **return** Number(val) \* 2;

  } **else** {

    **return** **null**

  }

}

3、用：

假设有这样一个组件：

<template>

  <div>

    {{num}}

    <button @click="double">点击后让左边的数字翻倍</button>

  </div>

</template>

<script>

  export **default**{

    data(){

      **return** {

        num: 1

      }

    },

    methods: {

      double: **function** () {

        //这里的this.doubleNumber()方法就是上面写的组件里的方法

        **this**.num = **this**.doubleNumber(**this**.num);

      }

    }

  }

</script>

我们便可以通过点击button按钮，让num的值，在每次点击都翻倍了。

4、假如添加的是属性：

例如：

Vue.prototype.number = 1;

会发生什么事情呢？

1、不管是【按值传递类型】还是【按引用传递类型】，该变量都不会被不同组件所共享，更准确的说，假如有A、B两个组件。A组件里的number数值改变，B组件里的number数值是不会跟着改变的。因此不要想着引用这样一个变量，然后修改了A中的值，B里也自动跟着改变了；

2、当组件里没有该属性时，调用时，显示的是通过插件获取的值；

当组件里有该属性时，调用时，显示的是组件里该属性的值；

由此而推，函数也是这样的，组件里的同名函数总是会覆盖插件提供的函数。

也就是说，当插件提供一个属性时，组件里没这个属性，就用插件的属性；组件有，就用组件自己的。

**3.2【添加全局方法或属性】**

1、核心思想：

就是给Vue对象添加一个属性。不用实例化，直接用Vue.方法名/Vue.属性 调用 。

初次接触很容易和上面3.1弄混，实际上，3.1是给组件里使用的，而3.2是给Vue对象使用的。

例如，假如添加一个方法test()，那么：

通过3.1添加，是在组件里，通过this.test()来调用

通过3.2添加，是在外面，通过Vue实例，如Vue.test()来调用

2、写：

//放在哪里参考上面

Vue.test = **function** () {

  alert("123")

}

3、用：

//注意先导入Vue对象才能使用

Vue.test()

使用时会执行对应的方法，比如这里就是alert弹窗

**3.3【注入组件】**

**3.4【添加全局资源】**

<http://www.jb51.net/article/108960.htm>

**二，vue-touch：移动端触摸事件插件，基于hammer.js。**

**使用：注意，vuejs2.x 以上版本，需要hammer.js 2.0以上版本.**

**A：安装**

**vue-touch**

**git方式下载：**

[**https://github.com/vuejs/vue-touch**](https://github.com/vuejs/vue-touch)

**git init**

**git clone**

**hammer.js**

[**https://github.com/hammerjs/hammer.js/tree/v2.0.8**](https://github.com/hammerjs/hammer.js/tree/v2.0.8)

js:

**var VueTouch = require('vue-touch') //或者直接引入
Vue.use(VueTouch)**

**B：使用**

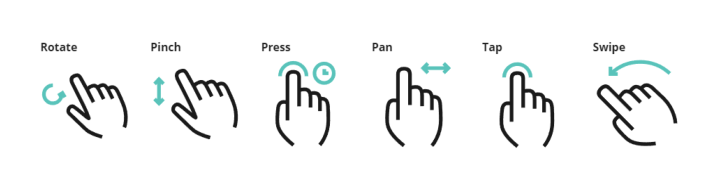
**<!-- Renders a div element by default -->
<!-- Renders a div element by default -->
<v-touch v-on:swipeleft="onSwipeLeft">Swipe me!</v-touch>
<!-- Render as other elements with the 'tag' prop -->
<v-touch tag="a" v-on:tap="onTap">Tap me!</v-touch>**

**C：配置**

**// change the threshold for all swipe recognizers
VueTouch.config.swipe = {
threshold: 200
}**

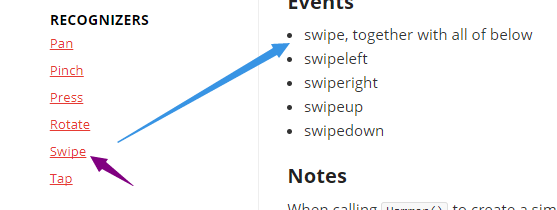
**<a
v-touch:pan="onPan"
v-touch-options:pan="{ direction: 'horizontal', threshold: 100 }">
</a>**

**D:事件类型（大分类）：**



具体参考hammer.js 事件类型：<http://hammerjs.github.io/recognizer-swipe/>

这里是 swipe 具体事件类型：



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**实训项目：1.vue插件开发：2种常用方式-vue-touch触摸事件**

**2.Q3 手机app**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**下一篇: vuejs-路由-单文件组件**

**笔记在线地址：http://note.youdao.com/noteshare?id=fb8d89f63ef434550e432bff0da31d93**