## 安装

详情参照官网：<https://cn.vuejs.org/v2/guide/installation.html>

在项目中安装 $npm install vue ，安装完毕后在项目node\_modules中

引入：

1.可以在html文件中直接用script标签引入

2.也可以利用 import Vue from "vue/dist/vue.js";再通过webpack打包，详见项目实战部分。

## 二、介绍

Vue属于一个渐进式框架，你可以根据项目的实际情况使用vue其中的一部分功能。也可以和其他框架混合使用。

## 基础

### vue实例

每个 Vue.js 应用都是通过构造函数 Vue 创建一个 **Vue 的根实例** 启动的：

var vm = new Vue({

// 选项

})

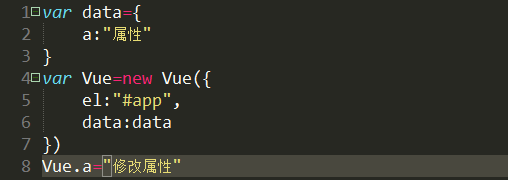
可以扩展vue组件

var MyComponent = Vue.extend({

// 扩展选项

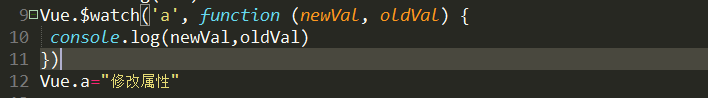
})

属性与方法



可以修改通过构造函数返回的对象来修改data的值

除了 data 属性， Vue 实例暴露了一些有用的实例属性与方法。这些属性与方法都有前缀 $，以便与代理的 data 属性区分。例如：



var vm = new Vue({

data: {

a: 1

},

created: function () {

// `this` 指向 vm 实例

console.log('a is: ' + this.a)

}

})

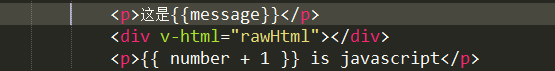
// -> "a is: 1"

此外还有一些其他的钩子函数，在生命周期不同阶段调用，如mounted，updated,destroyed 等，其中this指向这个实例化的函数。

### 模板语法

### 插值：

采用{{message}}的形式进行数据绑定，可以使纯文本也可以是js表达式,也可以是html



### 指令：

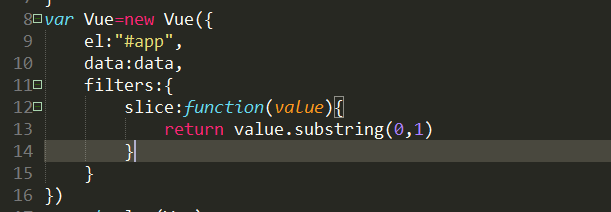
参数<a v-bind:href="url">参数</a>

修饰符：修饰符（Modifiers）是以半角句号 . 指明的特殊后缀，用于指出一个指令应该以特殊方式绑定。例如，.prevent 修饰符告诉 v-on 指令对于触发的事件调用 event.preventDefault()：

|  |
| --- |
| <form v-on:submit.prevent="onSubmit"></form> |

### 过滤器：

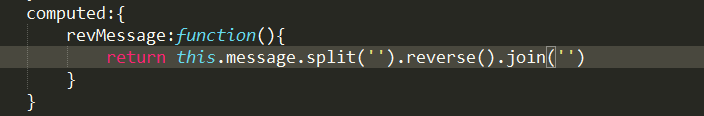
<p>自定义过滤器：{{ message | slice }}</p>



### 计算属性

主要使用computed

<p>这是计算属性的结果：<i>{{revMessage}}</i></p>



### class与style绑定

class对象语法：

<p v-bind:class="{active:isActive}">class绑定切换</p>

class数组语法：



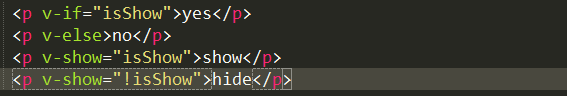
Style对象语法：

<div v-bind:style="styleObject"></div>

Style数组语法：

<p v-bind:style="[baseStyles, overridingStyles]">style数组语法</p>

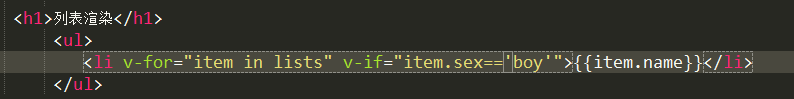
### 条件渲染

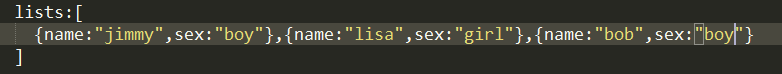


V-if是渲染与否该节点，v-show是显示隐藏与否，注意区分它们之间的区别，一般来说， v-if 有更高的切换开销，而 v-show 有更高的初始渲染开销。因此，如果需要非常频繁地切换，则使用 v-show 较好；如果在运行时条件不太可能改变，则使用 v-if 较好。

### 列表渲染

V-for with v-if





整数迭代

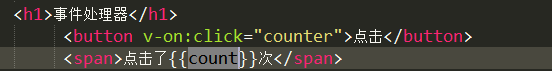
<div>

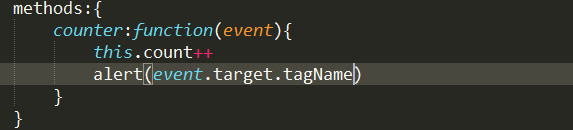
<span v-for="n in 10">{{ n }}</span>

</div>

### 事件处理器

可以用 v-on 指令监听 DOM 事件来触发一些 JavaScript 代码。





事件修饰符

<!-- 阻止单击事件冒泡 -->

<a v-on:click.stop="doThis"></a>

<!-- 提交事件不再重载页面 -->

<form v-on:submit.prevent="onSubmit"></form>

<!-- 修饰符可以串联 -->

<a v-on:click.stop.prevent="doThat"></a>

<!-- 只有修饰符 -->

<form v-on:submit.prevent></form>

<!-- 添加事件侦听器时使用事件捕获模式 -->

<div v-on:click.capture="doThis">...</div>

<!-- 只当事件在该元素本身（而不是子元素）触发时触发回调 -->

<div v-on:click.self="doThat">...</div>

按键修饰符

在监听键盘事件时，我们经常需要监测常见的键值。 Vue 允许为 v-on 在监听键盘事件时添加按键修饰符：

|  |
| --- |
| <!-- 只有在 keyCode 是 13 时调用 vm.submit() -->  <input v-on:keyup.13="submit"> |

记住所有的 keyCode 比较困难，所以 Vue 为最常用的按键提供了别名：

|  |
| --- |
| <!-- 同上 -->  <input v-on:keyup.enter="submit">  <!-- 缩写语法 -->  <input @keyup.enter="submit"> |

全部的按键别名：

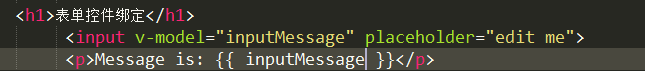
* .enter
* .tab
* .delete (捕获 “删除” 和 “退格” 键)
* .esc
* .space
* .up
* .down
* .left
* .right

可以通过全局 config.keyCodes 对象[自定义按键修饰符别名](https://cn.vuejs.org/v2/api/" \l "keyCodes)：

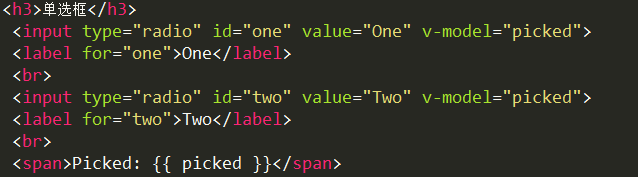
|  |
| --- |
| // 可以使用 v-on:keyup.f1  Vue.config.keyCodes.f1 = 112 |

### 表单控件绑定

Input框：

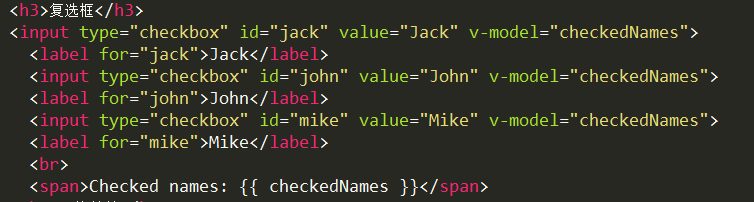


Radio:



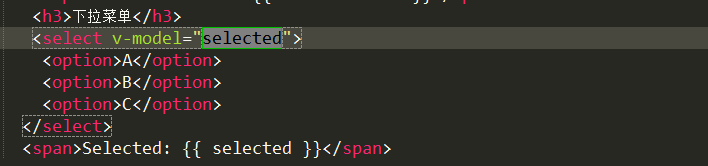


checkBox:





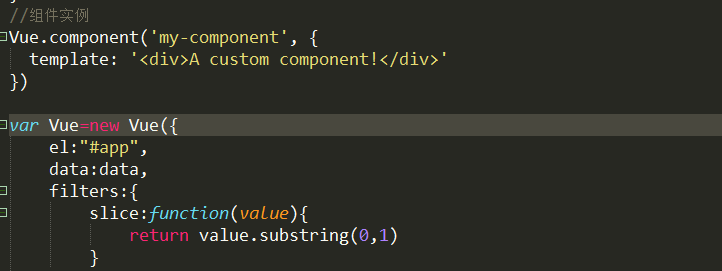
Select:





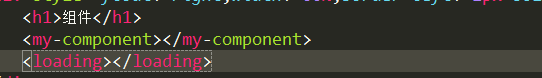
### 组件

1. 全局注册要先于vue构造函数实例化

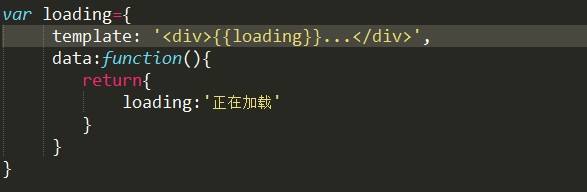


1. 局部注册，放在components参数中

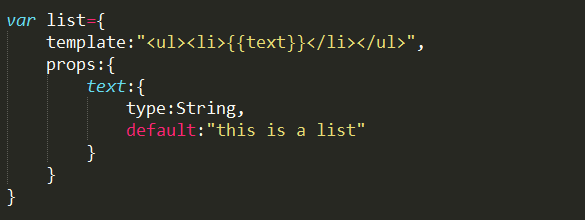


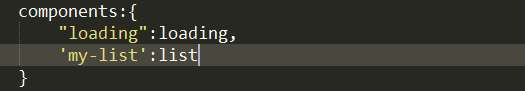


1. data必须是函数



1. prop参数传递

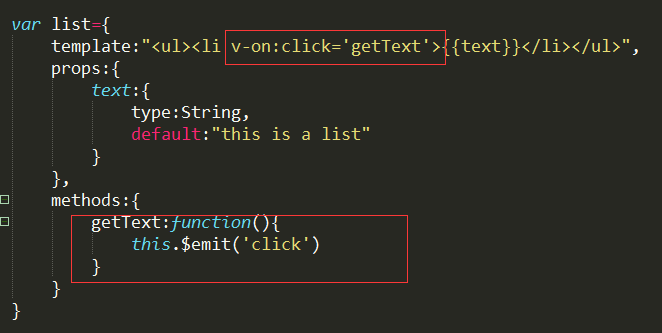






参数传递搭配v-for 使用

1. 自定义事件

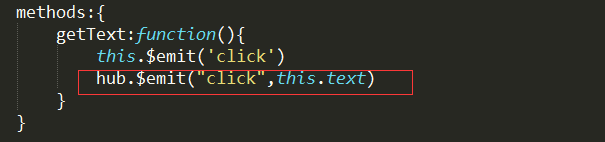




1. 非父子组件通信

*var* hub=new Vue()，hub作为两个组件之间的桥梁，

A组件中：



B组件中：

