# vue基础

Vue.js（读音 /vjuː/, 类似于 view） 是一套构建用户界面的渐进式框架。

Vue 只关注视图层， 采用自底向上增量开发的设计。

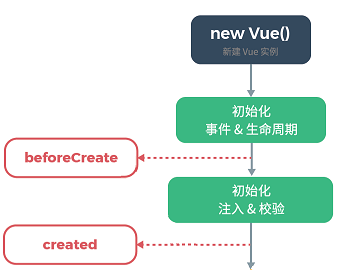
Vue 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

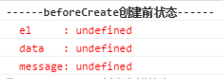
## 生命周期详解：

Vue实例有一个完整的生命周期，也就是说从开始创建、初始化数据、编译模板、挂在DOM、渲染-更新-渲染、卸载等一系列过程，而 Vue 实例的生命周期钩子函数就是在某个阶段给你一个做某些处理的机会。

创建vue实例，即在代码中对应：

var vm = new Vue({ … })

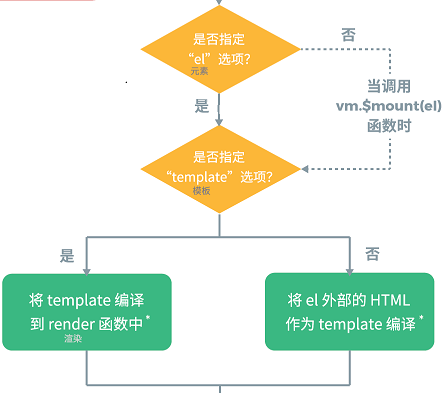






**beforeCreate：**当该钩子函数可以执行时，实例、事件已经经过初始化事件和生命周期。此时data没有挂载到vm对象上，所以无法访问data和真实的挂载元素el;

**created:** 当该钩子函数可以使用时，数据挂载data和绑定事件已经完成。此时可以使用、更改data中的数据。在该地方更改不会触发updated钩子函数，所以这里是渲染前倒数第二次更改数据的地方，一般在这做初始数据获取。



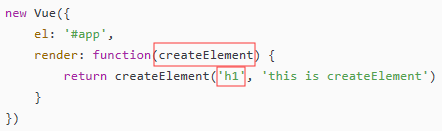
render函数选项 > template选项 > (template不存在时)outer HTML.

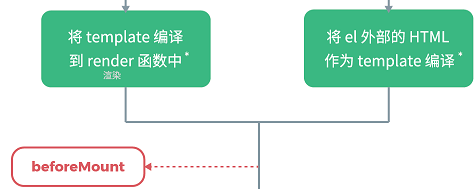
（1）如果vue实例对象中有template参数选项，则将其作为模板template编译成虚拟dom放入到render函数中准备渲染。  
（2）如果没有template选项，则将挂载元素el的outerHTML作为模板template编译。

判断是否有el元素，没有判断是否是通过vm.$mount(el)手动挂载，如果没有挂载，声明周期变会停止。

当注释掉：el: ‘App’时会报错：









render (h) {

return h('div', {}, this.text)

}

render函数里面的传参h就是Vue里面的createElement方法，return返回一个createElement方法，其中要传3个参数，第一个参数就是创建的div标签；第二个参数传了一个对象，对象里面可以是我们组件上面的props，或者是事件之类的东西；第三个参数就是div标签里面的内容，这里我们指向了data里面的text。

**beforeMount:** 在该钩子函数能使用之前，虚拟dom已经创建完成，即将进行渲染，此时可以更改数据不引发其他钩子函数(updated)；此时已经可以获取到$el成员，并可以进行替换。

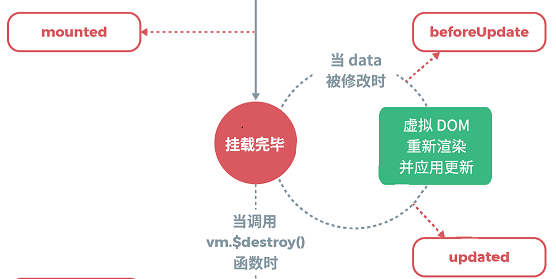
注意的是，此时的$el已经不是undefined了，但h1中还是用{{message}}进行占位。





**mounted:** 该钩子函数能使用的时候，已经将虚拟dom渲染成了真实dom并挂载到页面；此时获取到的message，已经是真实的内容.

此时修改数据会触发update函数







vm.message = '触发组件更新'

当vue发现data中的数据发生了改变，会**触发对应组件的重新渲染。**

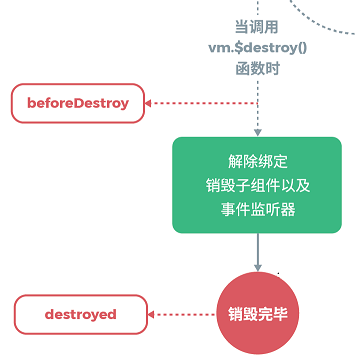
 当更新完成后，执行updated，数据已经更改完

成，dom也重新render完成，可以操作更新后

的虚拟dom。在beforeUpdate和updated都不能

更改数据，否则会进入死循环。

vue的虚拟dom机制会重新构建虚拟dom与上一次的虚拟dom树利用**diff算法**进行对比之后重新渲染



在调用$destroy()函数后，在销毁之前可以执行beforeDestroy钩子函数，此时实例可以正常使用。一般用来做善后工作：

清除计时器，清除非指令绑定的事件。

组件的数据绑定、监听...去掉后只剩下dom空壳，这个时候，执行 **destroyed**钩子函数。调用后，Vue 实例指示的所有东西都会解除绑定，所有的事件监听器会被移除，所有的子实例也会被销毁

## MVVM(Model-View-ViewModel)

作用：简单来说不用再让我们修改内容时还要操作dom了



view(html,css,模板文件) 和 viewModel(js,runtime,compiler)之间进行数据的双向绑定。

viewModel 和 Model(java…mysql…)之间通过ajax请求，和json返回。

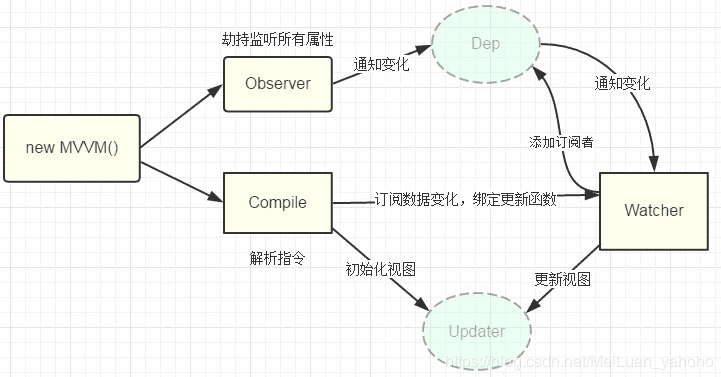
ViewModel所封装出来的数据模型包含视图的状态和行为两部分，

Model的数据模型只包含状态，这样的封装使得ViewModel可以完整地去描述View层。

MVVM最标志的**特性是数据绑定**，MVVM的**核心理念**是通过数据绑定来实现View的分离，完全解耦View。

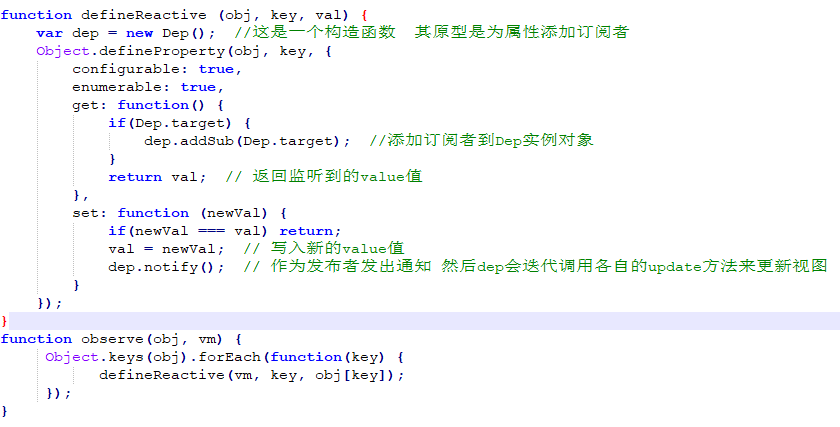
响应式双向数据绑定的实现原理：

vue.js采用的是**数据劫持结合发布-订阅者模式**的方式，通过**Object.defineProperty()**来劫持各个属性的setter,getter，在数据变动时发布消息给订阅者，触发相应的监听回调。



Vue内部通过Object.defineProperty方法属性拦截的方式，把data对象里每个数据的读写转化成getter/setter，当数据变化时通知视图更新

1. 需要实现数据监听器Observer监听数据对象的所有属性变化，如果有变化就会拿到最新值并通知订阅者
2. 实现一个指令解析器Compile，对每个元素节点的指令进行扫描和解析，根据指令模板替换数据，以及绑定相应的更新函数
3. 实现一个Watcher，作为连接Observer和Compile的桥梁，能够订阅并收到每个属性变动的通知，执行指令绑定的相应回调函数，从而更新视图
4. mvvm入口函数，整合以上三者



其中Object.defineProperty()方法接收三个参数：属性所在对象，属性的名字，描述符对象。

那么实现数据双向绑定的核心就是利用**为每一个属性都创建了订阅者的实例对象**, 以便观察；getter函数里面返回一个value值,在setter函数中写入修改后的值并调用update方法更新视图的数据值,

# 模板语法

## 插值

### 插入文本

例：<div id="app"> <p>{{ message }}</p> </div>

new Vue{ (el: "#app",

data: {

message: "hello, world!"}

)}

### 插入html v-html

例：<div id="app"> <div v-html="message"></div> </div>

message: '<h1>菜鸟教程</h1>'

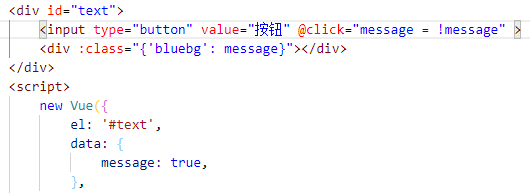
html块都可以，h、p、ul、table…

### 、属性值的插入 v-bind

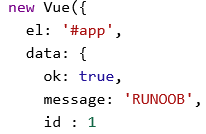
例：<div v-bind:class="{'class1': use}">

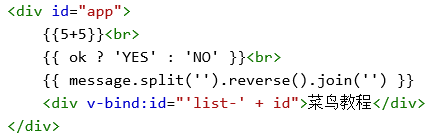
use: false

当use为false时，表示不用该className



1. **表达式**





{{ }}中可以加入运算符和方法

## 指令

是带有v-的特殊属性，比如之前的v-bind v-html

例：<p v-if=”text”>hello world</p> 若给text: false，则该段不插入

### 参数

参数在指令后以**冒号**指明。

例1：v-bind指令被用来响应的更新HTML属性

<div id="app"> <pre><a **v-bind***:href*="url">菜鸟教程</a></pre> </div>

这里的href是参数，bind将该属性和url表达式绑定

例2：v-on用于监听DOM事件

<a v-on:click="doSomething">

### 修饰符

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

此处的.prevent则是修饰符，该意思是在v-on监听到按钮触发的指令时，.prevent告诉v-on指令调用的是default.preventDefault()

## 用户输入 v-model

input中v-model指令可以实现双向数据绑定

例：<div id="app"><p>{{ message }}</p>

<input v-model="message"></div>

v-model绑定p中的数据，当input框中输入字符，p中也会出现相同的字符

v-model 指令用来在 **input、select、textarea、checkbox、radio** 等表单控件元素上创建双向数据绑定，根据表单上的值，自动更新绑定的元素的值。

## 过滤器 管道符 |

例1{{ message | capitalize }}

例2：<div v-bind:id="rawId | formatId"></div>

在第一个的基础上执行第二条方法，可以有多个 |

## 缩写

v-bind: href=”url1”的缩写 :href=”url1”

v-on:click=”myFunction” 缩写 @click=”myFunction”

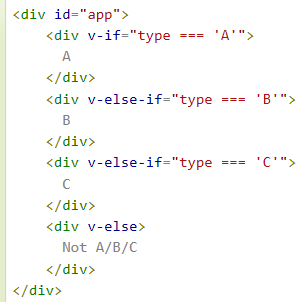
# 条件、循环语句

## 条件语句 v-if v-else v-else-if v-show

通过判定seen和ok为true或false,决定该模块是删除还是插入，这里的删除是指从dom树中删除。

可以在元素和template中使用v-if语句





data:{

type: ‘C’

}

v-show: <h1 v-show="ok">Hello!</h1>

注：v-if和v-show的区别：切换v-if的时候vue.js会有一个局部编译/卸载的过程，因为v-if中的模块也可能会有数据绑定和子组件；v-if 是真实的条件渲染，因为它会确保条件块在切换当中合适地销毁与重建条件块内的事件监听器和子组件。所以v-if每次卸载都会直接从dom树中删除。

v-show的元素在编译后被保存，在改变true或false的时候知识简单的css切换。

综上所述：v-if 有更高的切换消耗而 v-show 有更高的初始渲染消耗。因此，如果需要频繁切换 v-show 较好，如果在运行时条件不大可能改变 v-if 较好。

当 <template> 元素被包含在已被 vue 实例化之后的组件中时浏览器会正常渲染 可以用 v-if="false" 隐藏浏览器不会解析，不能使用 v-show。

## 循环语句 v-for

1. <li v-for="site in sites"> {{ site.name }} </li>

data: { sites: [ { name: 'Runoob' }, { name: 'Google' }, { name: 'Taobao' } ] }

1. 模板中同上
2. 迭代：

<li v-for="value in object">{{ value }}</li> 按顺序输出object中的value

<li v-for="(key,value) in object">{{ key }}:{{ value }}</li> 按顺序输出key: value

<li v-for="(value, key, value) in object">{{ value }}.{{ key }}:{{ value }}</li>

data:{ object{ name: “kangkang”, age: 10, tel: 12345678 }}

遍历整数<li v-for="a in 10">{{ a }}</li> 居然是1-10

遍历数组<li v-for="a in [1, 3, 4, 5]">{{ a }}</li>

# 计算属性 computed methods

<div id="app"> {{ message.split('').reverse().join('') }} </div>



两种输出结果相同；computed 是基于它的**依赖缓存**，**只有相关的数据(上面的message内容)发生改变时才会重新取值**。而使用 methods ，在重新渲染(每刷新)的时候，函数总会重新调用执行。所以computed性能会更好，但对于一些进入页面时的实时信息获取(进入页面的时间)，必须用methods

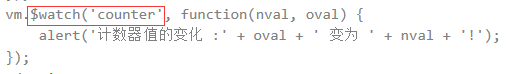
computed用途：在页面中使用大量或是复杂的表达式去**处理数据**，对页面的维护会有很大的影响。这个时候就需要用到computed 计算属性来处理复杂的逻辑运算。

总结：computed特性1：是计算值；2：应用于简化tempalte里面{{}}计算和处理props或$emit的传值；3：具有**缓存性**，页面重新渲染值不变化,计算属性会立即返回之前的计算结果，而不必再次执行函数

# 监听属性 watch



绑定按钮和p的内容,每按一次，counter加一，并更新到p中



watch监听指定的属性的变化，监听props，$emit或本组件的值执行异步操作

配合v-model实时监听input的内容进行交互

监听路由'$router'(to, from){}

1. 数组的监听：只能获取数组然后将里面的值进行遍历对比



也可以直接监听，加参数immediate:true

1. 对象和对象属性的监听，可以直接监听





1. 详解：

popInfo:{

sTime:'',

location:'',

}, 需要监听的父组件的值

# 样式绑定 class style bind

## class属性绑定

<div v-bind:class="{ active: isActive }"></div>

类名是active；当data中isActive是false时，该类的块消失

当有多个类时：

<div v-bind:class="{ active: isActive, active1: isActive1 }"></div>

可以把上诉情况的多个类写在一个对象里面：

<div v-bind:class="classObject"></div>

data：{

classObject:{

active: isActive,

active1: isActive1 }

}

数组语法:

<div v-bind:class="[activeClass, errorClass]"></div>

## 内联样式 style

<div v-bind:style="{ color: activeColor, fontSize: fontSize + 'px' }">菜鸟教程</div>

以上可简写：

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

data: {

styleObject: {

color: 'green',

fontSize: '30px'

}

}

注：双大括号写法不能在 HTML 属性中使用，应使用 v-bind 指令



通过按钮控制字体大小 (用于购物车中用该方法记录选择个数，计算总金额)

上面的方法：简写的不能动态调节

<div v-bind:style="{color: 'red', fontSize: fontSize + 'px'}">可以动态调节</div>

<div v-bind:style="styleObject"> 不可以动态调节</div>

# 事件处理器 v-on

简单的：



若要调用js的方法，需要创建一个方法：

<button v-on:click="greet">点击我</button>





事件修饰符：



按键修饰符：keyup

<input v-on:keyup.enter="submit"> 当按下enter的时候调用submit函数

按键别名：

.enter .tab .delete (捕获 "删除" 和 "退格" 键) .esc .space

.up .down .left .right .ctrl .alt .shift .meta

# 表单

## 输入框 v-model input textarea



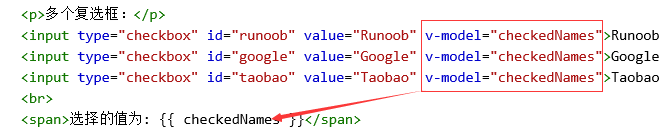
通过v-model进行数据的双向绑定

## 复选框

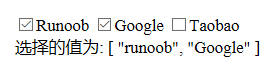
单个复选框：选中为true，取消为false



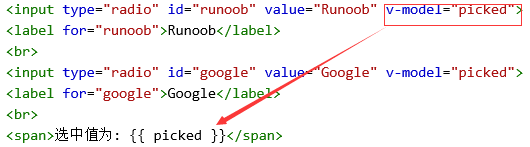
多个复选框：checkedNames初始化为**数组**，勾选的会以value的值出现在数组内



data: { checked : false, **checkedNames: []** }



## 单选框



data中设置picked初始值，选中哪个，picked的值就是哪个按钮的value值

## select

方法同上

**修饰符：**

.lazy 用于输入框中，输入时不会同步到另一个绑定的内容，回车后同步

<input v-model.lazy="message" placeholder="编辑我……">

<p>消息是: {{ message }}</p>

.number 一般默认输出的是字符串类型，加上后会变成Number

.trim 自动过滤用户输入的收尾空格

# 组件

组件可以扩展 HTML 元素，封装可重用的代码。

语法：Vue.component(tagName, options)

调用：<tagName></tagName>

## 全局组件

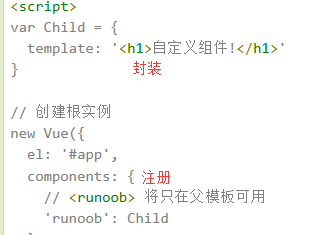
所有实例都能用全局组件

注册：Vue.component('runoob', { template: '<h1>自定义组件!</h1>' })

创建跟实例：new Vue({ el: '#app' })

调用：<runoob></runoob>

## 局部组件

只能在id=app中使用

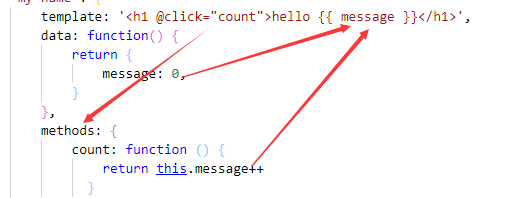
data中设置的值无法传入到组件中

给局部组件中传值：



必须要在局部组件中的data中设置，且只能设置函数，在返回值中返回插入信息

添加methods属性



注：vue组件是一个可扩展的vue实例。如上图一个局部组件里同样有vue实例提供的属性

## prop

prop 是**父组件用来传递数据**的一个自定义属性。

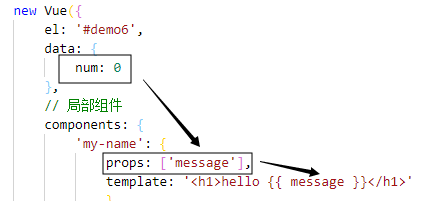
父组件的数据需要通过 props 把数据传给子组件，子组件需要显式地用 props 选项声明 "prop"：





类似于用 v-bind 绑定 HTML 特性到一个表达式，也可以用 v-bind 动态绑定 props 的值到父组件的数据中。每当父组件的数据变化时，该变化也会传导给子组件：





两种方法都可以传值，以上为父组件传值给子组件；即跟实例中定义num，局部组件用v-bind绑定属性，局部组件中需要用props进行声明，声明之后即可传值。

## $emit

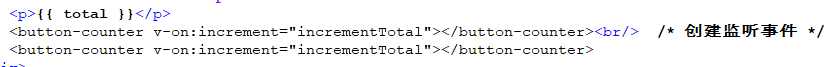
子组件传值给父组件：

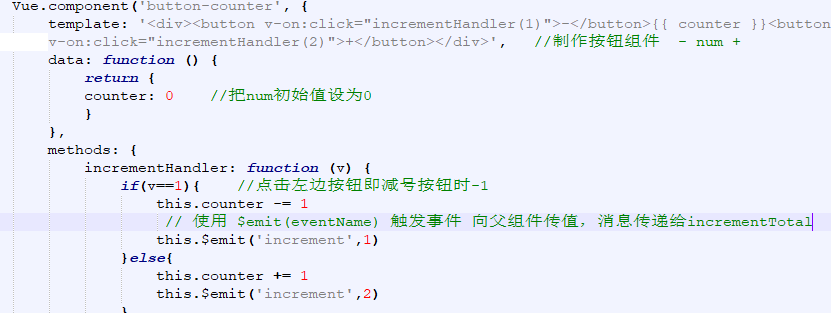
我们可以使用 v-on 绑定自定义事件, 每个 Vue 实例都实现了事件接口(Events interface)，即：

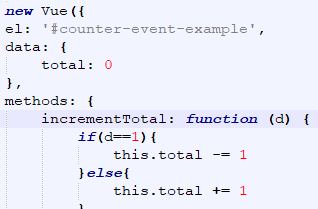
使用 $on(eventName) 监听事件

使用 $emit(eventName) 触发事件

另外，父组件可以在使用子组件的地方直接用 v-on 来监听子组件触发的事件。







注1：template中如果元素多变复杂后，需要加上div属性，且可读性很差，需要进行格式化

方法1. /

**方法2**. ` … `

**方法3**. template: ‘#demo’, 并在body体中写

<template id=”demo”>

<div>

…

</div>

</template>

方法4. template: ‘#demo’, 把块元素放在<script type=”text/x-template”></script>中

**方法5**. vue 把元素块单独放在vue文件中

注2：可以对props传入的值进行验证，props: [‘message’]写法改为：

props: { ‘message’: {

type: string} } 若传入的不是string则会报错

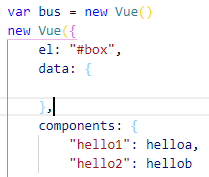
## 非父子组件之间的传值：

方法1：emit on 空实例(中央事件总线)和自定义事件；mounted生命周期中监听

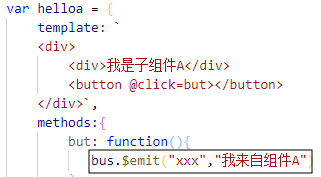
方发2：vuex状态管理：state mutations commit (复杂的时候考虑)

方法1：

首先定义一个空组件实例： var bus = new Vue()，创建一个跟实例和两个子组件。



然后通过点击事件触发$emit ，用空实例做中转站，所以他的名字xxx无所谓，后面跟上传输的内容

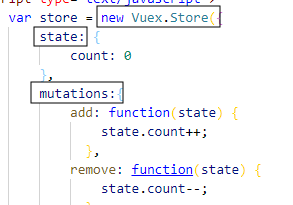


bus.$on接收数据，2中的bind(this)解决this的指向问题，否则指向的是组件A



再次注意的地方，组件名要规范；子组件中的data是以函数的方式返回。

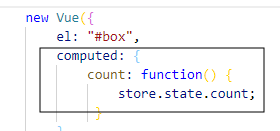
方法2：



第一步：定义一个vuex实例，

第二步state状态，里面是共享数据

第三步：mutation中写入相关方法



添加计算属性，获取count值，渲染到box中



在点击事件中，通过提交commit的方式，触发方法

注意的地方：

1. 正确的嵌套方式与is属性
2. 单向数据流: 子组件数据不影响父组件
3. 数据的作用域
4. 原生事件.native
5. $refs

## 组件slot(插槽分发)

如果想在组件之中插入一些东西，需要借助v-slot

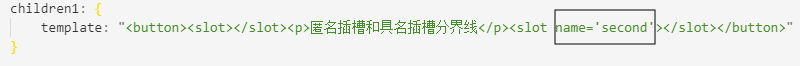
1. 单个slot：也称默认、匿名slot

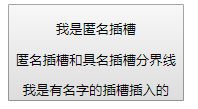


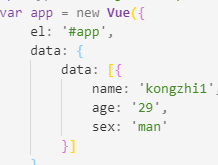
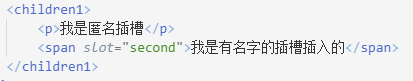


<p>中的文字将会插入到<slot>的位置，但由于是默认的，如果在template中同时出现多次，则<p>也会重复对应的次数。

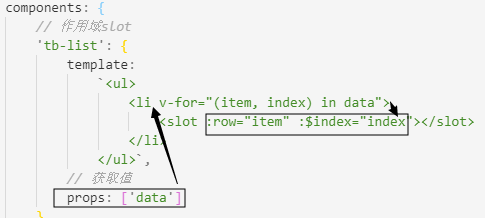
1. 具名slot,加上了name属性，就可以在一个template多次使用v-slot了







1. 作用域slot





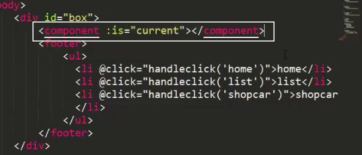
## 动态组件 <component>

<component> 元素，动态地绑定到它的 is 特性，通过is选择对应的组件名，让多个组件可以使用同一个挂载点，并动态切换。

下例这是定义三个局部组件，给<li>一个点击事件，返回对应的组件名，然后通过methods方法修改current为对应的组件名，这样就可以实现动态切换。

因为属性栏不能用{{}}，只能用v-bind()进行属性样式绑定。

但是这样动态切换是将切换之后的组件在dom树中删除，所以无法保存数据，需要<keep-alive></keep-alive>把<component> 嵌套起来，保留之前的渲染，这样在动态切回来的时候数据也会保存。



## 过渡效果

### 单元素/组件过渡

1. **css过渡**



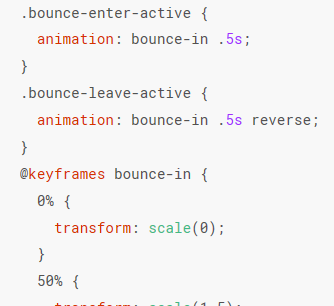
注：name的fade名class名相同；v-if和v-show都可以

1. 自动嗅探目标元素是否应用了 CSS 过渡或动画，如果是，在恰当的时机添加/删除 CSS 类名。
2. 如果过渡组件提供了 [**JavaScript 钩子函数**](https://cn.vuejs.org/v2/guide/transitions.html#JavaScript-钩子)，这些钩子函数将在恰当的时机被调用。
3. 如果没有找到 JavaScript 钩子并且也没有检测到 CSS 过渡/动画，DOM 操作 (插入/删除) 在下一帧中立即执行。(注意：此指浏览器逐帧动画机制，和 Vue 的 nextTick 概念不同)

过渡的类名见vue官网

1. **css动画**

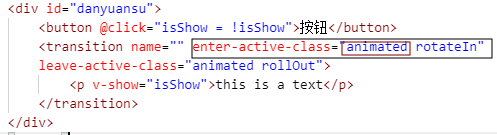
CSS 动画用法同 CSS 过渡，都是用<transition>，区别是在动画中 v-enter 类名在节点插入 DOM 后不会立即删除，而是在 animationend 事件触发时删除。



1. **结合css3 animate.css动画库**

引入下面的animate.css库，调用里面的动画即可

<link href="https://cdn.jsdelivr.net/npm/animate.css@3.5.1" rel="stylesheet" type="text/css">



class中必须有animated, 后面接动画效果，github上可查。

1. 过渡持续时间

<transition :duration="{ enter: 500, leave: 800 }">...</transition>

制定移入移出的持续时间

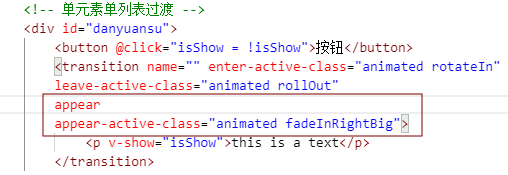
1. js钩子

略

### 初始渲染过渡

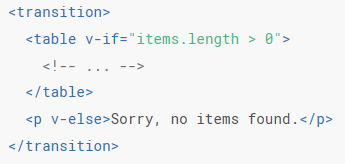
可以通过 appear 特性设置节点在初始渲染的过渡。

添加apper属性，加入appear-active-class设置页面刷新初始动画，同单元素过渡，可以自定义和加入css库，以下为加入css库

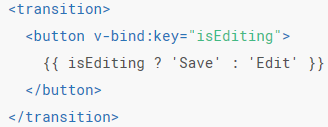


### 多个元素过渡(设置key)

多组件过渡，可以用v-if和v-else，最常用的多标签过渡是一个列表和描述这个列表为空消息的元素：



若在v-if v-else中出现相同的标签名时需要加入key属性区分；还可以给key加上v-bind进行属性修改





**mode属性**：in-out out-in

### 多个组件过渡

多组件过渡使用动态组件，外面包裹<transition>,其他同单元素过渡。



### 列表过渡(设置key)



注：<li>中的key必须唯一，且不能用index，因为删除的时候是最后一个索引值减少，所以删除的动画会一直加给最后一个列表。

设置成data，若内容重复也会有相同的问题

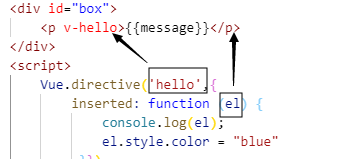
### 可复用过渡

封装到组件中

## 自定义指令

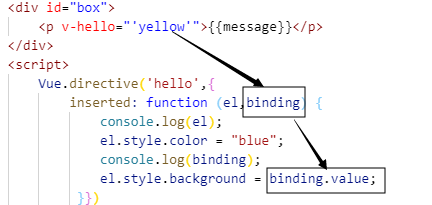
对**纯DOM元素**进行底层操作

1. 自定义指令介绍：directives



el接收到的是p标签及其内容

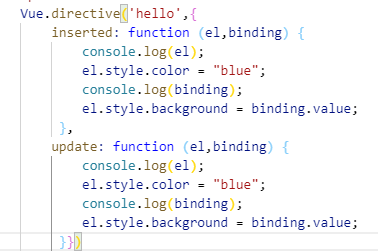
1. 钩子函数
   1. 参数el, binding, vnode(vnode.content)
   2. bind,inserted,update



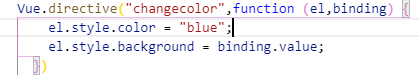


value: 指令的绑定值

简写：



可简写为：



相对于bind和inserted只调用一次，update可以随时更新。

指令传入多个值：

<div v-demo="{ color: 'white', text: 'hello!' }"></div> 传入对象

<div v-demo="obj"></div>

data:{

obj: { … }

}

## 组件之间通信

1：父组件向子组件传值：使用prop向子组件传值；

2：子组件实时监听父组件传来的值的变化：使用watch去监听父组件传来的值；

3：父组件可以通过this.$refs.name.去访问子组件的值或方法；

4：子组件可以通过this.$parent.去访问父组件的值或方法；

父组件监控子组件数据

尽量通过子组件显式的通过this.$dispatch向父级派发事件，保持父子间数据通讯的简洁干净

子组件监控父组件数据

使用prop只能是在子组件加载的时候接收值，一次性；如果需要实时更新的话需要使用watch

# 路由

Vue.js 路由允许我们通过不同的 URL 访问不同的内容。

通过 Vue.js 可以实现多视图的单页Web应用（single page web application，SPA）。

路由是根据不同的url地址展示不同的内容或页面。

前端路由：把不同路由对应的内容或页面交给前端来做；

之前是通过从服务端根据url的不同返回不同的页面实现。

可以理解为，之前是前端发送请求路由给服务端，服务端返回页面或内容；现在是前端路由调用前端存储的页面，不走后端；实现前后端分离

前端路由使用情况：单页面应用，**大部分页面结构不变，只改变部分内容**

优点：用户体验好，不用发送后端请求，可以快速展示

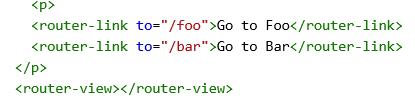
缺点：不利于SEO，看似10张页面实则只有1张，其他是模拟出来的

使用浏览器的前进、后退时会重新发请求，不能合理利用缓存

单页面无法记住滚动条位置

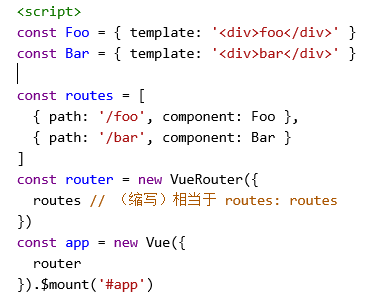
## 基本样式

Vue.js + vue-router 可以很简单的实现单页应用



**<router-link>** 是一个组件，可看成一个a标签，该组件用于设置一个导航链接，切换不同 HTML 内容。**to** 属性为目标地址，即要显示的内容。

**<router-view>**为路由出口，路由匹配到的组件将渲染在该组件这里



创建和挂载跟实例，整个应用都可以拥有路由功能

创建router实例，里面传入routes配置

定义路由，每条路由映射一个组件；可通过Vue-extend()创建组件构造器；以下是一个组件配置对象

定义路由组件，可以从其他文件import…from…过来

注：点击过的路由class ="router-link-exact-active router-link-active"

## <route-link>相关属性

this.$route.push({path:’’})等同于<route-link>；用于js代码中

可用于在一个页面在点击操作后，回到主页面

route.push()封装的是history.pushState()

1. to: 表示目标路由的链接，等同于<a>中的href

当路由的链接被点击时，内部会立刻把to的值传到router.push(); 所以可以传入一个字符串(链接)，也可以传入一个对象(v-bind绑定后)；如记得加’ ’

<router-link :to="'home'">Home</router-link>

<router-link :to="{ path: 'home' }">Home</router-link>



1. replace 封装的是history.replaceState()

如果设置了replace属性，点击时会调用**route-replace()**而不是route-push(); 这样**不会留下历史记录**：

<router-link :to="{ path: '/abc'}" replace></router-link>

1. append

设置后会在当前路径前添加基路径

<router-link :to="{ path: 'relative/path'}" append></router-link>

有了append,relative路径会自动添加

1. tag

添加后可以渲染成某种标签，若想让路由是li的样式

<router-link to="/foo" tag="li">foo</router-link>

会变成：<li>foo</li>

1. **active-class 像给<a>添加visited伪类**

链接被点击后激活css的类样式：active-class = "\_active"

<router-link v-bind:to = "{ path: '/route1'}" **active-class = "\_active"**>Router Link 1</router-link>

可以css中定义 ： .\_active{ color: “red”}

1. **exact-active-class 5)的强化**

默认路由是模糊匹配，当匹配的是/route1/a时，/route1的路由的类也会被激活。改成exact-active-class后会精确匹配链接，就不会发生上述情况。

<router-link v-bind:to = "{ path: '/route1'}" **active-class = "\_active"**>Router Link 1</router-link>

<router-link v-bind:to = "{ path: '/route1/a'}" >Router Link 2</router-link>

1. event

该声明可以触发导航的事件。可以是一个字符串和包含字符串的数组。

<router-link v-bind:to = "{ path: '/route1'}" event = "mouseover">Router Link 1</router-link>

## 动态路由



商城中不同的物品点击查看详情的页面需要同一个组件来渲染

或者不同的用户用不同的ID查询一个内容，需要同一个组件渲染

此时就要用到动态路由：

const User = { template: “<div>User: {{ $route.params.id }}</div>”}

const router = new VueRouter{

routers : [{

path: ‘/user/:id’, component: User }] }

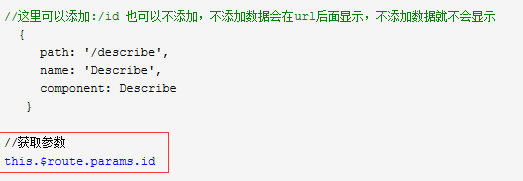
此时输入用户id就可以获取组件内容。

上图第二类需要分两断填写

const User = { template: “<div>User: {{ $route.params.name }} {{ $route.params.id }}</div>”}

动态路由传参：

1. this.$router.push({path:`/user/${this.id}`})



1. this.$router.push({ name: 'Describe', params: { id: this.id } })

路由配置同上

1. this.$router.push({ path: '/describe', query: { id: id } })

this.$route.query.id

注意：在子组件中 获取参数的时候是$route.params 而不是$router

上面的方法可以直接写进<router-link>标签中

<router-link :to="'/index/'+id">

路由配置还是为：

响应路由参数的变化：

在使用路由参数时。例两个路由a,跳转到b, 他们的内容都是由一个组件渲染的；这时原来的组件实例会被复用；这意味着组件的生命周期钩子不会再被调用。

（钩子函数可以理解为回调函数）

未完

## 嵌套路由 children



在定义路由的里面加入children: [ {xxx}, [yyy] ]

可以定义多个子路由，子路由不能/开头，这让会变成绝对地址



1. **编程式路由**

this.$route.push({“name”})

this.$route.push({path:’name’})

this.$route.push({path:’name?a=123’}) 或者this.$route.push({path:’name’, query{a:123}})

this.route.go(1)



# vuex

专为 Vue.js 应用程序开发的**状态管理模式,** 采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化

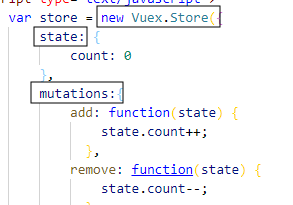
* state，驱动应用的数据源； 指向view
* view，以声明方式将 state 映射到视图； 指向actions
* actions，响应在 view 上的用户输入导致的状态变化 指向state

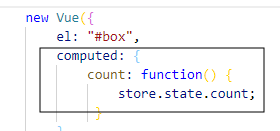
提取共享数据，单独管理适用于中大型项目

在vue组件开发中，经常会需要将当前的组件的数据传递给其他的组件，即非父子组件相互传值的时候。如果采用$emit $on，组件多了就太绕了，数据也没办法管理，所以采用Vuex。

new Vuex.store({}) 含义是创建一个Vuex实列。store是vuex的一个核心方法，字面含义为 '仓库'的意思。实列化完成后，需要注入到vue实列中(需要在入口js文件中导入并在vue实例中添加：store, )，它有五个核心的选项，state、mutations、getters、actions和modules。

非父子组件：





添加计算属性，获取count值，渲染到box中



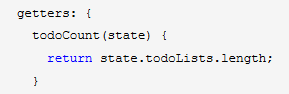
在点击事件中，通过提交commit的方式，触发方法

## 核心选项state、mutations、getters、actions和modules

**state**是**存储**状态 定义应用状态全局的数据结构；

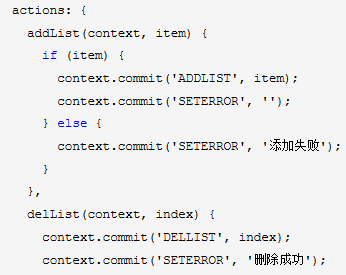
**mutations**是提交状态**修改**，也就是set、get中的set，这是vuex中唯一修改state的方式，但是不支持异步操作。 每个mutation都有一个字符串的事件类型(type)和一个回调函数(handler) 第一个参数默认是state，外部调用方式为：store.commit('SET\_AGE', 30).

**getters**是从state中派生出状态的。也就是set、get中的get，它有两个可选的参数，state和getters， 分别可以**获取**state中的变量和其他的getters。外部调用的方式：store.getters.todoCount()



**actions**和上面的mutations类似，但是actions支持**异步**操作的，外部调用方式为：store.dispatch('nameAction')

常见的使用是：从服务器端获取数据，在数据获取完成后会调用 store.commit()来更改Store中的状态。 Action函数接收一个与store实列具有相同方法和属性的context对象，因此我们可以使用 context.commit 提交一个 mutation，或者通过 context.state 和 context.getters来获取state和getters



modules的作用是将数据模块化。

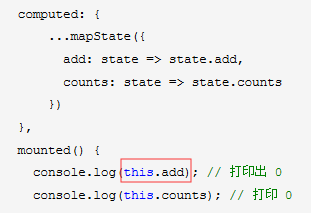
关键属性: 添加namespaced将actions,mutations,getters限定在模块命名空间中

## 辅助函数mapState mapMutations…

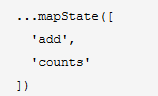
1. mapState

当我们的组件需要获取多个状态的时候，将这些状态都声明为计算属性会有些重复和冗余，为了解决这个问题，我们可以使用mapState的辅助函数来帮助我们生成计算属性。

使用方式1：通过该方法可以把多个要获取的状态放在一起；通过this取值



使用方式2：数组形式



两种方法都可以将this.add映射到this.$store.state.add

1. mapAction 思想同上，对应的核心选项actions



1. mapMutations 对应核心选项mutations, 获取里面的方法

方式一： ...mapMutations(['ADD'])

方式二： ...mapMutations({ 'ADD': 'ADD' })

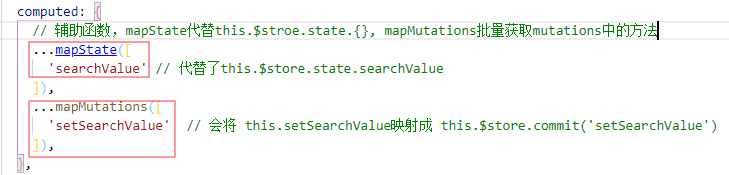
使用：



如果不适用mapMutations，则需要this.$store.commit('ADD', 1)

在拉勾网项目中的使用：



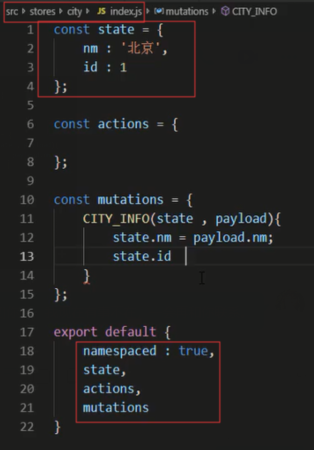




this.setSearchValue(this.inputValue)

## vuex的数据模块化





类似于路由模块化，在stores路径下创建一个index.js的主文件。

该例为添加关于city的模块。

通过import导入，然后添加到modules将数据模块化；

在该路径下创建/city/index.js

填写相关数据。注意的是需要添加namespaced: true；将state、mutations等限定到一个命名空间。

# 可复用性&组合

## 混入

## 自定义指令

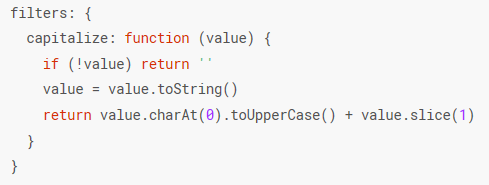
## 插件

## 过滤器

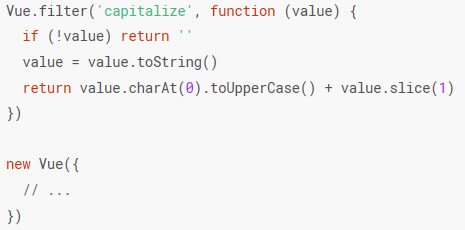
处理一些常见的文本格式化

使用：{{ message | formatMsg }} :id=”rawId | formatId”

定义：本地定义



全局定义：



# vue全家桶之Vue-Resource

安装：1.通过<script>方式应用 2.npm install vue-resource –save(推荐)

7个api:

get(url, [option])

。。。

使用axios好了

# webpack

作用：分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（Sass，TypeScript等），并将其转换和打包为合适的格式供浏览器使用。在3.0出现后，Webpack还肩负起了优化项目的责任。

1. 打包：可以把多个Javascript文件打包成一个文件，减少服务器压力和下载带宽。
2. 转换：把拓展语言(目前浏览器不支持的语法等)转换成为普通的JavaScript，让浏览器顺利运行。
3. 优化：前端变的越来越复杂后，性能也会遇到问题，而WebPack也开始肩负起了优化和提升性能的责任。

webpack是一个打包工具，可以帮你把你的项目(**这里的项目其实就是指通过模块化开发的项目)**的复杂依赖关系的静态文件打包为简洁版的浏览器可识别的静态资源；可以将一些浏览器尚不支持的新特性进行转换，**提高兼容性**

开发环境配置代理：

/config/index.js 的module.exports 部分的proxyTable

# vue中的ref(refs)的使用

1、ref 加在**普通的元素**上，用this.ref.name 获取到的是dom元素，修改其样式

2、ref 加在**子组件**上，用this.ref.name 获取到的是**组件实例，可以使用组件的所有方法(可以进行父传子)**。

<child ref=”message” />

this.$refs.message.getMessage('我是子组件一！')  
 子组件中：

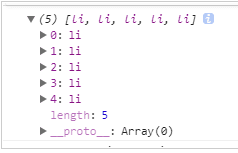
getMessage(msg){this.message = msg}

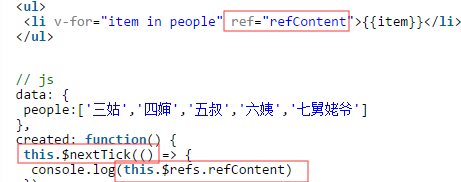
3、如何利用 **v-for 和 ref** 获取一组数组或者dom 节点

说明：ref的注册时间是在dom渲染之后，所以需要用在mounted或者this.$nextTick(() => {})中；一般this.$nextTick()用在create生命周期中对dom进行一个异步操作，比较此时dom还没创建出来。

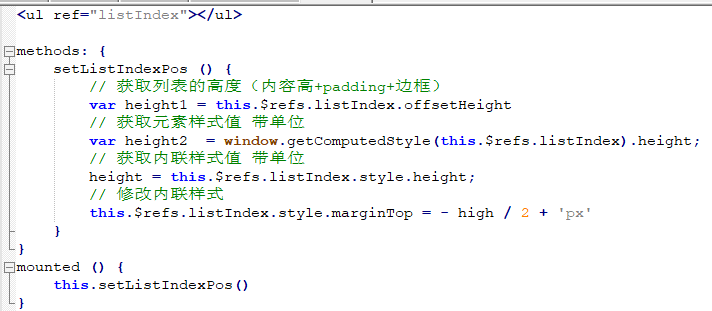
如果ref所在的是循环出来的，那么将返回一个数组，使用时将其遍历即可

例：





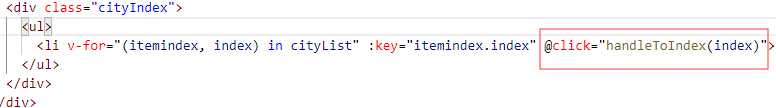
例：对一个列表进行dom(改变css样式)操作



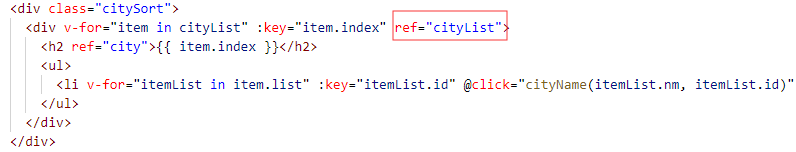
例：配合scrollTop和offsetTop实现，城市列表点击索引后调转到对应的城市集



1. 给每个索引添加点击事件，传入li的索引值

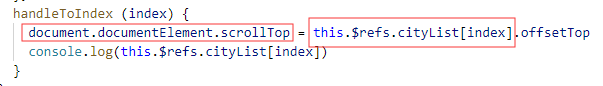


1. 在城市集的循环中添加ref属性，cityList将返回的是例: 首字母为C城市列表集





1. 点击索引进行A-Z的跳转；因为传入的索引的index和for循环中cityList的索引值相对应，可以直接使用不用循环获取值来对比。



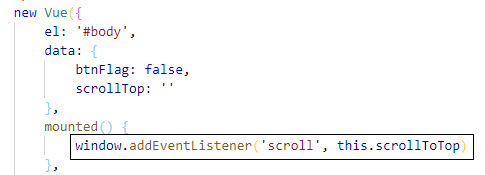
offsetTop获取元素到上级层顶部的高度；

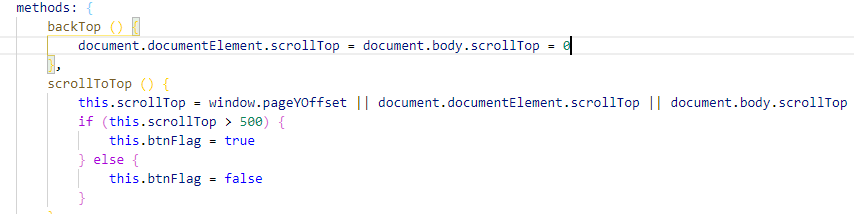
scrollTop将该高度作为顶部

注：页面指定了DTD，即指定了DOCTYPE时，使用document.documentElement

页面没有DTD，即没指定DOCTYPE时，使用document.body

回到顶部：





mounted中设置事件监听器，监听滚动，触发scrolltoTop()方法

该方法获取当前页面作为顶部位置的高度，添加判断在超过多少显示“回到页面按钮”；

添加点击事件，将作为顶部位置的高度重置为0(可以加上延时往上翻的效果)

**核心：document.documentElement.scrollTop(获取滚动条位置)**

**[skroʊl]**

页可见区域宽： document.documentElement.clientWidth;  
网页可见区域高： document.body.clientHeight;

window.innerHeight;  
网页可见区域宽： document.documentElement.offsetWidth   (包括边线的宽);  
网页可见区域高： document.documentElement.offsetHeight (包括边线的宽);  
网页正文全文宽： document.documentElement.scrollWidth;  
网页正文全文高： document.documentElement.scrollHeight;  
**网页被卷去的高： document.documentElement.scrollTop;**  
网页被卷去的左： document.documentElement.scrollLeft;  
网页正文部分上： window.screenTop;  
网页正文部分左： window.screenLeft;  
屏幕分辨率的高： window.screen.height;  
屏幕分辨率的宽： window.screen.width;

屏幕可用工作区高度： window.screen.availHeight;

# 插件汇总

## UI组件

1. [**element**](https://github.com/ElemeFE/element)**- 饿了么出品的Vue2的web UI工具套件**
2. [Vux](https://github.com/airyland/vux)- 基于Vue和WeUI的组件库
3. [**mint-ui- Vue 2的移动UI元素**](file:///D:\WorkSpace\笔记\插件汇总\UI-mint-ui.htm)

**http://mint-ui.github.io/#!/zh-cn**

1. [iview](https://github.com/iview/iview)- 基于 Vuejs 的开源 UI 组件库
2. [Keen-UI](https://github.com/JosephusPaye/Keen-UI)- 轻量级的基本UI组件合集
3. [vue-material](https://github.com/marcosmoura/vue-material)- 通过Vue Material和Vue 2建立精美的app应用
4. [muse-ui](https://github.com/museui/muse-ui)- 三端样式一致的响应式 UI 库
5. [vuetify](https://github.com/vuetifyjs/vuetify)- 为移动而生的Vue JS 2组件框架
6. [vonic](https://github.com/wangdahoo/vonic)- 快速构建移动端单页应用
7. [eme](https://github.com/egoist/eme)- 优雅的Markdown编辑器
8. [vue-multiselect](https://github.com/monterail/vue-multiselect)- Vue.js选择框解决方案
9. [vue-table](https://github.com/ratiw/vue-table)- 简化数据表格
10. [VueCircleMenu](https://github.com/OYsun/VueCircleMenu)- 漂亮的vue圆环菜单
11. [vue-chat](https://github.com/Coffcer/vue-chat)- vuejs和vuex及webpack的聊天示例
12. [radon-ui](https://github.com/luojilab/radon-ui)- 快速开发产品的Vue组件库
13. [vue-waterfall](https://github.com/MopTym/vue-waterfall)- Vue.js的瀑布布局组件
14. [vue-carbon](https://github.com/myronliu347/vue-carbon)- 基于 vue 开发MD风格的移动端
15. [vue-beauty](https://github.com/FE-Driver/vue-beauty)- 由vue和ant design创建的优美UI组件
16. [vue-blu](https://github.com/chenz24/vue-blu)- 帮助你轻松创建web应用
17. [**vueAdmin**](https://github.com/taylorchen709/vueAdmin)**- 基于vuejs2和element的简单的管理员模板**
18. [vue-syntax-highlight](https://github.com/vuejs/vue-syntax-highlight)- Sublime Text语法高亮
19. [vue-infinite-scroll](https://github.com/ElemeFE/vue-infinite-scroll)- VueJS的无限滚动指令
20. [Vue.Draggable](https://github.com/David-Desmaisons/Vue.Draggable)- 实现拖放和视图模型数组同步
21. [vue-awesome-swiper](https://github.com/surmon-china/vue-awesome-swiper)- vue.js触摸滑动组件
22. [vue-calendar](https://github.com/jinzhe/vue-calendar)- 日期选择插件
23. [bootstrap-vue](https://github.com/pi0/bootstrap-vue)- 应用于Vuejs2的Twitter的Bootstrap 4组件
24. [vue-swipe](https://github.com/ElemeFE/vue-swipe)- VueJS触摸滑块
25. [vue-amap](https://github.com/ElemeFE/vue-amap)- 基于Vue 2和高德地图的地图组件
26. [vue-chartjs](https://github.com/apertureless/vue-chartjs)- vue中的Chartjs的封装
27. [vue-datepicker](https://github.com/hilongjw/vue-datepicker)- 日历和日期选择组件
28. [markcook](https://github.com/jrainlau/markcook)- 好看的markdown编辑器
29. [vue-google-maps](https://github.com/GuillaumeLeclerc/vue-google-maps)- 带有双向数据绑定Google地图组件
30. [vue-progressbar](https://github.com/hilongjw/vue-progressbar)- vue轻量级进度条
31. [vue-picture-input](https://github.com/alessiomaffeis/vue-picture-input)- 移动友好的图片文件输入组件
32. [vue-infinite-loading](https://github.com/PeachScript/vue-infinite-loading)- VueJS的无限滚动插件
33. [vue-upload-component](https://github.com/lian-yue/vue-upload-component)- Vuejs文件上传组件
34. [vue-datetime-picker](https://github.com/Haixing-Hu/vue-datetime-picker)- 日期时间选择控件
35. [vue-scroller](https://github.com/wangdahoo/vue-scroller)- Vonic UI的功能性组件
36. [vue2-calendar](https://github.com/icai/vue2-calendar)- 支持lunar和日期事件的日期选择器
37. [vue-video-player](https://github.com/surmon-china/vue-video-player)- VueJS视频及直播播放器
38. [vue-fullcalendar](https://github.com/Wanderxx/vue-fullcalendar)- 基于vue.js的全日历组件
39. [rubik](https://github.com/ccforward/rubik)- 基于Vuejs2的开源 UI 组件库
40. [VueStar](https://github.com/OYsun/VueStar)- 带星星动画的vue点赞按钮
41. [vue-mugen-scroll](https://github.com/egoist/vue-mugen-scroll)- 无限滚动组件
42. [mint-loadmore](https://github.com/mint-ui/mint-loadmore)- VueJS的双向下拉刷新组件
43. [vue-tables-2](https://github.com/matfish2/vue-tables-2)- 显示数据的bootstrap样式网格
44. [vue-virtual-scroller](https://github.com/Akryum/vue-virtual-scroller)- 带任意数目数据的顺畅的滚动
45. [DataVisualization](https://github.com/SimonZhangITer/DataVisualization)- 数据可视化
46. [vue-quill-editor](https://github.com/surmon-china/vue-quill-editor)- 基于Quill适用于Vue2的富文本编辑器
47. [Vueditor](https://github.com/hifarer/Vueditor)- 所见即所得的编辑器
48. [vue-html5-editor](https://github.com/PeakTai/vue-html5-editor)- html5所见即所得编辑器
49. [vue-msgbox](https://github.com/ElemeFE/vue-msgbox)- vuejs的消息框
50. [vue-slider](https://github.com/warpcgd/vue-slider)- vue 滑动组件
51. [vue-core-image-upload](https://github.com/Vanthink-UED/vue-core-image-upload)- 轻量级的vue上传插件
52. [vue-slide](https://github.com/hilongjw/vue-slide)- vue轻量级滑动组件
53. [**vue-lazyload-img**](https://github.com/JALBAA/vue-lazyload-img)**- 移动优化的vue图片懒加载插件**
54. [vue-drag-and-drop-list](https://github.com/Alex-fun/vue-drag-and-drop-list)- 创建排序列表的Vue指令
55. [vue-progressive-image](https://github.com/MatteoGabriele/vue-progressive-image)- Vue的渐进图像加载插件
56. [vuwe](https://github.com/vuwe/vuwe)- 基于微信WeUI所开发的专用于Vue2的组件库
57. [vue-dropzone](https://github.com/rowanwins/vue-dropzone)- 用于文件上传的Vue组件
58. [vue-charts](https://github.com/hchstera/vue-charts)- 轻松渲染一个图表
59. [vue-swiper](https://github.com/weilao/vue-swiper)- 易于使用的滑块组件
60. [vue-images](https://github.com/littlewin-wang/vue-images)- 显示一组图片的lightbox组件
61. [vue-carousel-3d](https://github.com/Wlada/vue-carousel-3d)- VueJS的3D轮播组件
62. [vue-region-picker](https://github.com/QingWei-Li/vue-region-picker)- 选择中国的省份市和地区
63. [vue-typer](https://github.com/cngu/vue-typer)- 模拟用户输入选择和删除文本的Vue组件
64. [vue-impression](https://github.com/NewDadaFE/vue-impression)- 移动Vuejs2 UI元素
65. [vue-datatable](https://github.com/galenyuan/vue-datatable)- 使用Vuejs创建的DataTableView
66. [vue-instant](https://github.com/santiblanko/vue-instant)- 轻松创建自动提示的自定义搜索控件
67. [vue-dragging](https://github.com/hilongjw/vue-dragging)- 使元素可以拖拽
68. [vue-slider-component](https://github.com/NightCatSama/vue-slider-component)- 在vue1和vue2中使用滑块
69. [vue2-loading-bar](https://github.com/BosNaufal/vue2-loading-bar)- 最简单的仿Youtube加载条视图
70. [vue-datepicker](https://github.com/weifeiyue/vue-datepicker)- 漂亮的Vue日期选择器组件
71. [vue-video](https://github.com/hilongjw/vue-video)- Vue.js的HTML5视频播放器
72. [vue-toast-mobile](https://github.com/ElemeFE/vue-toast-mobile)- VueJS的toast插件
73. [vue-image-crop-upload](https://github.com/dai-siki/vue-image-crop-upload)- vue图片剪裁上传组件
74. [vue-tooltip](https://github.com/Akryum/vue-tooltip)- 带绑定信息提示的提示工具
75. [vue-highcharts](https://github.com/weizhenye/vue-highcharts)- HighCharts组件
76. [vue-touch-ripple](https://github.com/surmon-china/vue-touch-ripple)- vuejs的触摸ripple组件
77. [coffeebreak](https://github.com/Kocisov/coffeebreak)- 实时编辑CSS组件工具
78. [vue-datasource](https://github.com/coderdiaz/vue-datasource)- 创建VueJS动态表格
79. [vue2-timepicker](https://github.com/phoenixwong/vue2-timepicker)- 下拉时间选择器
80. [vue-date-picker](https://github.com/Bubblings/vue-date-picker)- VueJS日期选择器组件
81. [vue-scrollbar](https://github.com/BosNaufal/vue-scrollbar)- 最简单的滚动区域组件
82. [vue-quill](https://github.com/CroudSupport/vue-quill)- vue组件构建quill编辑器
83. [vue-google-signin-button](https://github.com/phanan/vue-google-signin-button)- 导入谷歌登录按钮
84. [vue-svgicon](https://github.com/MMF-FE/vue-svgicon)- 创建svg图标组件的工具
85. [vue-float-label](https://github.com/bkzl/vue-float-label)- VueJS浮动标签模式
86. [vue-baidu-map](https://github.com/Dafrok/vue-baidu-map)- 基于 Vue 2的百度地图组件库
87. [vue-social-sharing](https://github.com/nicolasbeauvais/vue-social-sharing)- 社交分享组件
88. [vue2-editor](https://github.com/davidroyer/vue2-editor)- HTML编辑器
89. [vue-tagsinput](https://github.com/Ginhing/vue-tagsinput)- 基于VueJS的标签组件
90. [vue-easy-slider](https://github.com/shhdgit/vue-easy-slider)- Vue 2.x的滑块组件
91. [datepicker](https://github.com/vue-bulma/datepicker)- 基于flatpickr的时间选择组件
92. [vue-chart](https://github.com/miaolz123/vue-chart)- 强大的高速的vue图表解析
93. [vue-music-master](https://github.com/yunyi1895/vue-music-master)- vue手机端网页音乐播放器
94. [handsontable](https://github.com/vue-bulma/handsontable)- 网页表格组件
95. [vue-simplemde](https://github.com/F-loat/vue-simplemde)- VueJS的Markdown编辑器组件
96. [vue-popup-mixin](https://github.com/myronliu347/vue-popup-mixin)- 用于管理弹出框的遮盖层
97. [cubeex](https://github.com/fangyongbao/cubeex)- 包含一套完整的移动UI
98. [vue-fullcalendar](https://github.com/CroudSupport/vue-fullcalendar)- vue FullCalendar封装
99. [vue-material-design](https://github.com/loujiayu/vue-material-design)- Vue MD风格组件
100. [vue-morris](https://github.com/bbonnin/vue-morris)- Vuejs组件封装Morrisjs库
101. [we-vue](https://github.com/tianyong90/we-vue)- Vue2及weui1开发的组件
102. [vue-image-clip](https://github.com/legeneek/vue-image-clip)- 基于vue的图像剪辑组件
103. [vue-bootstrap-table](https://github.com/jbaysolutions/vue-bootstrap-table)- 可排序可检索的表格
104. [vue-radial-progress](https://github.com/wyzant-dev/vue-radial-progress)- Vue.js放射性进度条组件
105. [vue-slick](https://github.com/staskjs/vue-slick)- 实现流畅轮播框的vue组件
106. [vue-pull-to-refresh](https://github.com/bajian/vue-pull-to-refresh)- Vue2的上拉下拉
107. [vue-form-2](https://github.com/matfish2/vue-form-2)- 全面的HTML表单管理的解决方案
108. [vue-side-nav](https://github.com/vue-comps/vue-side-nav)- 响应式的侧边导航
109. [mint-indicator](https://github.com/mint-ui/mint-indicator)- VueJS移动加载指示器插件
110. [chartjs](https://github.com/vue-bulma/chartjs)- Vue Bulma的chartjs组件
111. [vue-scroll](https://github.com/suguangwen/vue-scroll)- vue滚动
112. [vue-ripple](https://github.com/BosNaufal/vue-ripple)- 制作谷歌MD风格涟漪效果的Vue组件
113. [vue-touch-keyboard](https://github.com/icebob/vue-touch-keyboard)- VueJS虚拟键盘组件
114. [vue-chartkick](https://github.com/ankane/vue-chartkick)- VueJS一行代码实现优美图表
115. [vue-ztree](https://github.com/lisiyizu/vue-ztree)- 用 vue 写的树层级组件
116. [vue-m-carousel](https://github.com/shiye515/vue-m-carousel)- vue 移动端轮播组件
117. [vue-datepicker-simple](https://github.com/dai-siki/vue-datepicker-simple)- 基于vue的日期选择器
118. [vue-tabs](https://github.com/alexqdjay/vue-tabs)- 多tab页轻型框架
119. [vue-verify-pop](https://github.com/aweiu/vue-verify-pop)- 带气泡提示的vue校验插件
120. [vue-parallax](https://github.com/vue-comps/vue-parallax)- 整洁的视觉效果
121. [vue-img-loader](https://github.com/JackGit/vue-img-loader)- 图片加载UI组件
122. [vue-typewriter](https://github.com/eduardostuart/vue-typewriter)- vue组件类型
123. [vue-smoothscroll](https://github.com/Teddy-Zhu/vue-smoothscroll)- smoothscroll的VueJS版本
124. [vue-city](https://github.com/xinxingyu/vue-city)- 城市选择器
125. [vue-tree](https://github.com/weibangtuo/vue-tree)- vue树视图组件
126. [vue-ios-alertview](https://github.com/Treri/vue-ios-alertview)- iOS7+ 风格的alertview服务
127. [dd-vue-component](https://github.com/ibufu/dd-vue-component)- 订单来了的公共组件库
128. [paco-ui-vue](https://github.com/yeseason/paco-ui-vue)- PACOUI的vue组件
129. [vue-cmap](https://github.com/doodlewind/vue-cmap)- Vue China map可视化组件
130. [vue-button](https://github.com/steven5538/vue-button)- Vue按钮组件

## 开发框架

1. [vue.js](https://github.com/vuejs/vue)- 流行的轻量高效的前端组件化方案
2. [vue-admin](https://github.com/fundon/vue-admin)- Vue管理面板框架
3. [electron-vue](https://github.com/SimulatedGREG/electron-vue)- Electron及VueJS快速启动样板
4. [vue-2.0-boilerplate](https://github.com/petervmeijgaard/vue-2.0-boilerplate)- Vue2单页应用样板​
5. [vue-spa-template](https://github.com/hanan198501/vue-spa-template)- 前后端分离后的单页应用开发
6. [Framework7-Vue](https://github.com/nolimits4web/Framework7-Vue)- VueJS与Framework7结合
7. [vue-bulma](https://github.com/wangxg2016/vue-bulma)- 轻量级高性能MVVM Admin UI框架
8. [vue-webgulp](https://github.com/rodzzlessa24/vue-webgulp)- 仿VueJS Vue loader示例
9. [vue-element-starter](https://github.com/Metnew/vue-element-starter)- vue启动页

## 实用库

1. [**vuex**](https://github.com/vuejs/vuex)**- 专为 Vue.js 应用程序开发的状态管理模式**
2. [vuelidate](https://github.com/monterail/vuelidate)- 简单轻量级的基于模块的Vue.js验证
3. [qingcheng](https://github.com/zerqu/qingcheng)- qingcheng主题
4. [vue-desktop](https://github.com/ElemeFE/vue-desktop)- 创建管理面板网站的UI库
5. [vue-meta](https://github.com/declandewet/vue-meta)- 管理app的meta信息
6. [**vue-axios**](https://github.com/imcvampire/vue-axios)**- 将axios整合到VueJS的封装**
7. [vue-svg-icon](https://github.com/cenkai88/vue-svg-icon)- vue2的可变彩色svg图标方案
8. [avoriaz](https://github.com/eddyerburgh/avoriaz)- VueJS测试实用工具库
9. [vue-framework7](https://github.com/lmk123/vue-framework7)- 结合VueJS使用的Framework7组件
10. [vue-bootstrap-modal](https://github.com/Coffcer/vue-bootstrap-modal)- vue的Bootstrap样式组件
11. [vuep](https://github.com/QingWei-Li/vuep)- 用实时编辑和预览来渲染Vue组件
12. [vue-online](https://github.com/Sopamo/vue-online)- reactive的在线和离线组件
13. [**vue-lazy-render**](https://github.com/yeyuqiudeng/vue-lazy-render)**- 用于Vue组件的延迟渲染**
14. [vue-password-strength-meter](https://github.com/apertureless/vue-password-strength-meter)- 交互式密码强度计
15. [element-admin](https://github.com/lynzz/element-admin)- 支持 vuecli 的 Element UI 的后台模板
16. [vue-electron](https://github.com/SimulatedGREG/vue-electron)- 将选择的API封装到Vue对象中的插件
17. [cleave](https://github.com/vue-bulma/cleave)- 基于cleave.js的Cleave组件
18. [vue-events](https://github.com/cklmercer/vue-events)- 简化事件的VueJS插件
19. [vue-shortkey](https://github.com/iFgR/vue-shortkey)- 应用于Vue.js的Vue-ShortKey 插件
20. [vue-cordova](https://github.com/kartsims/vue-cordova)- Cordova的VueJS插件
21. [vue-router-transition](https://github.com/weinot/vue-router-transition)- 页面过渡插件
22. [vue-gesture](https://github.com/mlyknown/vue-gesture)- VueJS的手势事件插件
23. [http-vue-loader](https://github.com/FranckFreiburger/http-vue-loader)- 从html及js环境加载vue文件
24. [vue-qart](https://github.com/superman66/vue-qart)- 用于qartjs的Vue2指令
25. [vuemit](https://github.com/gocanto/vuemit)- 处理VueJS事件
26. [vue-websocket](https://github.com/icebob/vue-websocket)- VueJS的Websocket插件
27. [vue-local-storage](https://github.com/pinguinjkeke/vue-local-storage)- 具有类型支持的Vuejs本地储存插件
28. [lazy-vue](https://github.com/gocanto/lazy-vue)- 懒加载图片
29. [vue-bus](https://github.com/yangmingshan/vue-bus)- VueJS的事件总线
30. [vue-reactive-storage](https://github.com/ropbla9/vue-reactive-storage)- vue插件的Reactive层
31. [vue-notifications](https://github.com/se-panfilov/vue-notifications)- 非阻塞通知库
32. [vue-lazy-component](https://github.com/Coffcer/vue-lazy-component)- 懒加载组件或者元素的Vue指令
33. [v-media-query](https://github.com/AStaroverov/v-media-query)- vue中添加用于配合媒体查询的方法
34. [vue-observe-visibility](https://github.com/Akryum/vue-observe-visibility)- 当元素在页面上可见或隐藏时检测
35. [vue-ts-loader](https://github.com/HerringtonDarkholme/vue-ts-loader)- 在Vue装载机检查脚本
36. [vue-pagination-2](https://github.com/matfish2/vue-pagination-2)- 简单通用的分页组件
37. [vuex-i18n](https://github.com/dkfbasel/vuex-i18n)- 定位插件
38. [Vue.resize](https://github.com/David-Desmaisons/Vue.resize)- 检测HTML调整大小事件的vue指令
39. [vuex-shared-mutations](https://github.com/xanf/vuex-shared-mutations)- 分享某种Vuex mutations
40. [vue-file-base64](https://github.com/BosNaufal/vue-file-base64)- 将文件转换为Base64的vue组件
41. [modal](https://github.com/vue-bulma/modal)- Vue Bulma的modal组件
42. [Famous-Vue](https://github.com/irwansyahwii/Famous-Vue)- Famous库的vue组件
43. [leo-vue-validator](https://github.com/LeoHuiyi/leo-vue-validator)- 异步的表单验证组件
44. [Vue-Easy-Validator](https://github.com/MetinSeylan/Vue-Easy-Validator)- 简单的表单验证
45. [vue-truncate-filter](https://github.com/imcvampire/vue-truncate-filter)- 截断字符串的VueJS过滤器
46. [vue-zoombox](https://github.com/vue-comps/vue-zoombox)- 一个高级zoombox
47. [vue-input-autosize](https://github.com/syropian/vue-input-autosize)- 基于内容自动调整文本输入的大小
48. [**vue-lazyloadImg**](https://github.com/yodfz/vue-lazyloadImg)**- 图片懒加载插件**

## 服务端

1. [nuxt.js](https://github.com/nuxt/nuxt.js)- 用于服务器渲染Vue app的最小化框架
2. [express-vue](https://github.com/danmademe/express-vue)- 简单的使用服务器端渲染vue.js
3. [vue-ssr](https://github.com/ccforward/vue-ssr)- 非常简单的VueJS服务器端渲染模板
4. [vue-ssr](https://github.com/hilongjw/vue-ssr)- 结合Express使用Vue2服务端渲染
5. [vue-easy-renderer](https://github.com/leaves4j/vue-easy-renderer)- Nodejs服务端渲染

## 辅助工具

1. [DejaVue](https://github.com/MiCottOn/DejaVue)- Vuejs可视化及压力测试
2. [vue-play](https://github.com/vue-play/vue-play)- 展示Vue组件的最小化框架
3. [vscode-VueHelper](https://github.com/OYsun/vscode-VueHelper)- 目前vscode最好的vue代码提示插件
4. [vue-generate-component](https://github.com/NetanelBasal/vue-generate-component)- 轻松生成Vue js组件的CLI工具
5. [vue-multipage-cli](https://github.com/xwpongithub/vue-multipage-cli)- 简单的多页CLI
6. [VuejsStarterKit](https://github.com/MetinSeylan/VuejsStarterKit)- vuejs starter套件

## 应用实例

1. [koel](https://github.com/phanan/koel)- 基于网络的个人音频流媒体服务
2. [pagekit](https://github.com/pagekit/pagekit)- 轻量级的CMS建站系统
3. [vuedo](https://github.com/Vuedo/vuedo)- 博客平台
4. [jackblog-vue](https://github.com/jackhutu/jackblog-vue)- 个人博客系统
5. [vue-cnode](https://github.com/lzxb/vue-cnode)- 重写vue版cnode社区
6. [CMS-of-Blog](https://github.com/ycwalker/CMS-of-Blog)- 博客内容管理器
7. [rss-reader](https://github.com/mrgodhani/rss-reader)- 简单的rss阅读器
8. [vue-ghpages-blog](https://github.com/viko16/vue-ghpages-blog)- 依赖GitHub Pages无需本地生成的静态博客
9. [swoole-vue-webim](https://github.com/wh469012917/swoole-vue-webim)- Web版的聊天应用
10. [vue-dashing-js](https://github.com/thelinuxlich/vue-dashing-js)- nuvo-dashing-js的fork
11. [fewords](https://github.com/sapjax/fewords)- 功能极其简单的笔记本
12. [vue-blog](https://github.com/surmon-china/vue-blog)- 使用Vue2.0 和Vuex的vue-blog

## Demo示例

1. [Vue-cnodejs](https://github.com/shinygang/Vue-cnodejs)- 基于vue重写Cnodejs.org的webapp
2. [NeteaseCloudWebApp](https://github.com/javaSwing/NeteaseCloudWebApp)- 高仿网易云音乐的webapp
3. [vue-zhihu-daily](https://github.com/hilongjw/vue-zhihu-daily)- 知乎日报 with Vuejs
4. [vue-wechat](https://github.com/useryangtao/vue-wechat)- vue.js开发微信app界面
5. [vue2-demo](https://github.com/lzxb/vue2-demo)- 从零构建vue2 + vue-router + vuex 开发环境
6. [eleme](https://github.com/liangxiaojuan/eleme)- 高仿饿了么app商家详情
7. [vue-demo](https://github.com/kenberkeley/vue-demo)- vue简易留言板
8. [maizuo](https://github.com/zhengguorong/maizuo)- vue/vuex/redux仿卖座网
9. [spa-starter-kit](https://github.com/codecasts/spa-starter-kit)- 单页应用启动套件
10. [vue-music](https://github.com/Sioxas/vue-music)- Vue 音乐搜索播放
11. [douban](https://github.com/jiakeqi/douban)- 模仿豆瓣前端
12. [vue-Meizi](https://github.com/liangxiaojuan/vue-Meizi)- vue最新实战项目
13. [zhihudaily-vue](https://github.com/yatessss/zhihudaily-vue)- 知乎日报web版
14. [vue-demo-kugou](https://github.com/lavyun/vue-demo-kugou)- vuejs仿写酷狗音乐webapp
15. [VueDemo\_Sell\_Eleme](https://github.com/SimonZhangITer/VueDemo_Sell_Eleme)- Vue2高仿饿了么外卖平台
16. [vue2.0-taopiaopiao](https://github.com/canfoo/vue2.0-taopiaopiao)- vue2.0与express构建淘票票页面
17. [vue-leancloud-blog](https://github.com/jiangjiu/vue-leancloud-blog)- 一个前后端完全分离的单页应用
18. [node-vue-server-webpack](https://github.com/yjj5855/node-vue-server-webpack)- Node.js+Vue.js+webpack快速开发框架
19. [mi-by-vue](https://github.com/wendaosanshou/mi-by-vue)- VueJS仿小米官网
20. [vue-fis3](https://github.com/okoala/vue-fis3)- 流行开源工具集成demo
21. [vue2.x-douban](https://github.com/superman66/vue2.x-douban)- Vue2实现简易豆瓣电影webApp
22. [vue-demo-maizuo](https://github.com/ChuckCZC/vue-demo-maizuo)- 使用Vue2全家桶仿制卖座电影
23. [vue-zhihudaily](https://github.com/iHaPBoy/vue-zhihudaily)- 知乎日报 Web 版本
24. [vue-adminLte-vue-router](https://github.com/liujians/vue-adminLte-vue-router)- vue和adminLte整合应用
25. [vue-axios-github](https://github.com/superman66/vue-axios-github)- 登录拦截登出功能
26. [Zhihu-Daily-Vue.js](https://github.com/pomelo-chuan/Zhihu-Daily-Vue.js)- Vuejs单页网页应用
27. [hello-vue-django](https://github.com/rokups/hello-vue-django)- 使用带有Django的vuejs的样板项目
28. [vue-cnode](https://github.com/wszgxa/vue-cnode)- vue单页应用demo
29. [x-blog](https://github.com/CommanderXL/x-blog)- 开源的个人blog项目
30. [vue-express-mongodb](https://github.com/xrr2016/vue-express-mongodb)- 简单的前后端分离案例
31. [websocket\_chat](https://github.com/secreter/websocket_chat)- 基于vue和websocket的多人在线聊天室
32. [photoShare](https://github.com/beidan/photoShare)- 基于图片分享的社交平台
33. [vue-zhihudaily-2.0](https://github.com/cs1707/vue-zhihudaily-2.0)- 使用Vue2.0+vue-router+vuex创建的zhihudaily
34. [notepad](https://github.com/lin-xin/notepad)- 本地存储的记事本
35. [vueBlog](https://github.com/elva2596/vueBlog)- 前后端分离博客
36. [vue-ruby-china](https://github.com/hql123/vue-ruby-china)- VueJS框架搭建的rubychina平台
37. [Zhihu\_Daily](https://github.com/littlewin-wang/Zhihu_Daily)- 基于Vue和Nodejs的Web单页应用
38. [vue-koa-demo](https://github.com/Molunerfinn/vue-koa-demo)- 使用Vue2和Koa1的全栈demo
39. [vue2.x-Cnode](https://github.com/vincentSea/vue2.x-Cnode)- 基于vue全家桶的Cnode社区
40. [life-app-vue](https://github.com/shaqihe/life-app-vue)- 使用vue2完成多功能集合到小webapp
41. [github-explorer](https://github.com/SidKwok/github-explorer)- 寻找最有趣的GitHub库
42. [vue-trip](https://github.com/wenye123/vue-trip)- vue2做的出行webapp
43. [vue-ssr-boilerplate](https://github.com/albertchan/vue-ssr-boilerplate)- 精简版的ofvue-hackernews-2
44. [vue-bushishiren](https://github.com/nswbmw/vue-bushishiren)- 不是诗人应用
45. [houtai](https://github.com/peng1992/houtai)- 基于vue和Element的后台管理系统
46. [ios7-vue](https://github.com/QRL909109/ios7-vue)- 使用vue2.0 vue-router vuex模拟ios7
47. [Framework7-VueJS](https://github.com/tyllo/Framework7-VueJS)- 使用移动框架的示例
48. [cnode-vue](https://github.com/BubblyPoker/cnode-vue)- 基于vue和vue-router构建的cnodejs web网站SPA
49. [vue-cli-multipage-bootstrap](https://github.com/zhoou/vue-cli-multipage-bootstrap)- 将vue官方在线示例整合到组件中
50. [vue-cnode](https://github.com/jiananle/vue-cnode)- 用 Vue 做的 CNode 官网
51. [HyaReader](https://github.com/GitaiQAQ/HyaReader)- 移动友好的阅读器
52. [zhihu-daily](https://github.com/xrr2016/zhihu-daily)- 轻松查看知乎日报内容
53. [seeMusic](https://github.com/Alex-xd/seeMusic)- 跨平台云音乐播放器
54. [vue-cnode](https://github.com/Damonlw/vue-cnode)- 使用cNode社区提供的接口
55. [zhihu-daily-vue](https://github.com/moonou/zhihu-daily-vue)- 知乎日报
56. [sls-vuex2-demo](https://github.com/sailengsi/sls-vuex2-demo)- vuex2商城购物车demo
57. [vue-dropload](https://github.com/ITCNZ/vue-dropload)- 用以测试下拉加载与简单路由
58. [vue-cnode-mobile](https://github.com/soulcm/vue-cnode-mobile)- 搭建cnode社区
59. [Vuejs-SalePlatform](https://github.com/fishenal/Vuejs-SalePlatform)- vuejs搭建的售卖平台demo
60. [v-notes](https://github.com/Halfeld/v-notes)- 简单美观的记事本
61. [vue-starter](https://github.com/BosNaufal/vue-starter)- VueJs项目的简单启动页
62. [vue-memo](https://github.com/youknowznm/vue-memo)- 用 vue写的记事本应用

# 组件化和模块化

**对比：**



说明：

**组件**：最初的目的是**代码重用**，功能相对单一或者独立。在整个系统的代码层次上位于最底层，被其他代码所依赖，所以说组件化是**纵向分层**。

**模块**：最初的目的是将**同一类型的代码**整合在一起，所以模块的功能相对复杂，但都同属于一个业务。不同模块之间也会存在依赖关系，但大部分都是业务性的互相跳转，从地位上来说它们都是平级的。

模块化开发的作用也解决了**命名冲突**(变量函数名相同)；**文件依赖**(引入外来文件的顺序和数量)

总结：整个业务由多个模块构成，依赖小，大部分只存在跳转依赖；每个模块的功能也会很复杂；其中每个模块都含有相同的功能块，此时需要通过组件进行代码的重用；两者是相辅相成的。

例：组件化就像一个库，比如一个有样式的404页面或alert弹窗，很多模块都会用到，所以不封装成组件调用，就会产生重复代码，而且组件功能单一，没什么依赖，所以是高重用、低依赖。

而模块并不存在重用，每个模块都是该业务的不同功能；比如一个线上教学业务；通过功能可以分为：直播教学模块、社区问答模块、登录注册模块、信息查询模块等；这些模块基本上只存在于跳转上的依赖。

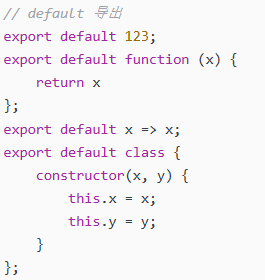
模块化进程：

1. script标签：一个文件看做一个模块；这样导致1.调用后都是全局作用域，有可能出现命名冲突；2.只能按照标签顺序进行加载；3.文件之间的依赖关系靠开发者主观解决。
2. ES6模块规范：

详情见web笔记ES6的export、import









vue中引入iconfont矢量图标库。

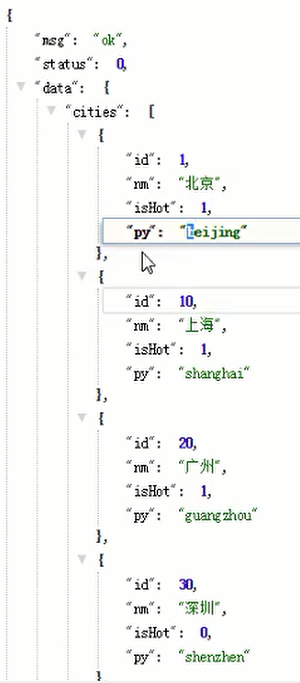
# 获取后端api数据

获取api内容是用json文件写的；可以用postman查看

例：获取后端提供关于城市选择的api里的数据：<http://39.97.33.178>/api/cityList

通过ajax获取该数据；会影响跨域问题，可以通过配置反向代理解决：

1. 创建vue.config.js文件(该方法没起作用)



1. 用了webpack的配置方法：config中的index.js

中的proxyTable中

1. 配置反向代理文件

export default {

devServer: {

proxy: {

‘api’: {

target: ‘http://39.97.33.178’,

changeOrigin: true

}

}

}

}

1. 重启脚手架
2. 下载axios

注：Axios是一个基于Promise（ES6中用

于处理异步的）的HTTP库，用于浏览器和

node.js中。

1. 引入(axios相当于是一个类)

在入口main.js文件中进行全局引用；



import axios from ‘axios’;

Vue.prototype.axios = axios;

这样就可以通过this引用

1. 使用

在需要请求数据的组件中调用

name: ‘City’,

mounted () {

this.axios.get(‘/api/cityList’).then( (data) => {

console.log(data);

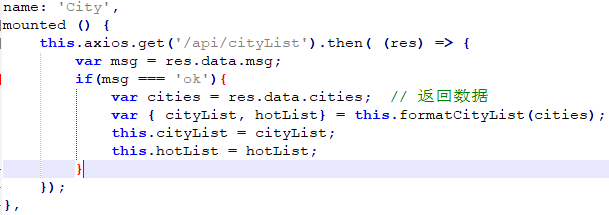
}

)

}

1. 渲染数据到页面

**数据处理：**

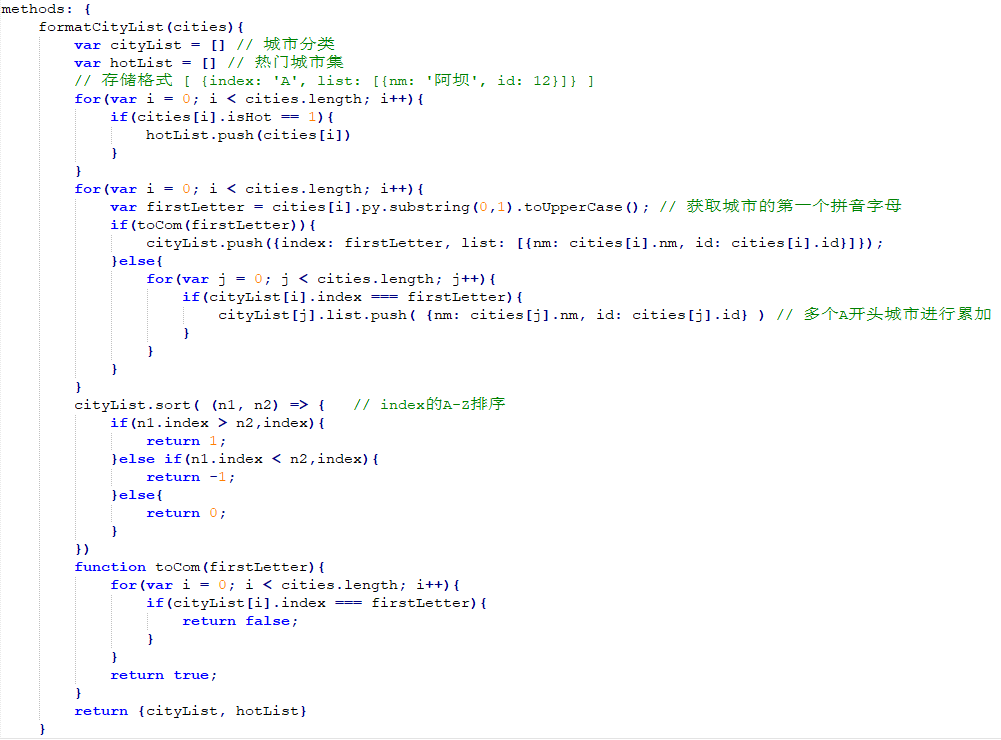


注：有提出vue中data中的数据无法通过this引用；需要在箭头函数上声明var that = this，使用that; 但是箭头函数和函数作用域不同，在对象中是不需要处理的。

k

勘误：res.data.data.cities

z

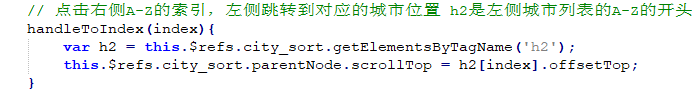


此时已经将掉接口获取到的城市信息进行整理排序存储在两个列表中

**数据展示：**

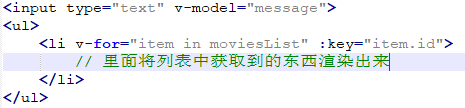
data(){ return{ cityList: [], hotList: []} }

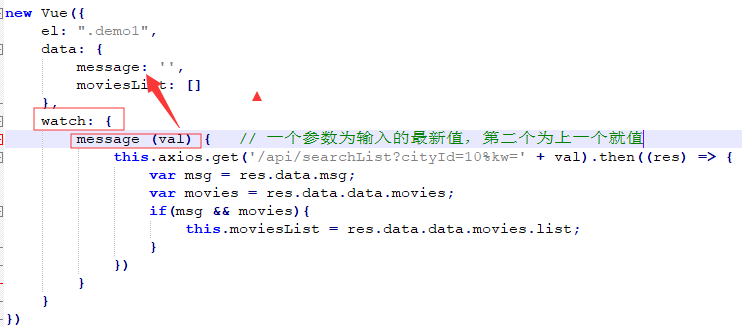
然后在template中通过li中v-for进行对数组元素进行遍历取值以及展示



# 搜索模块的创建

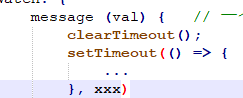
1. input中通过v-model= "message"获取到数据
2. 通过watch监听message





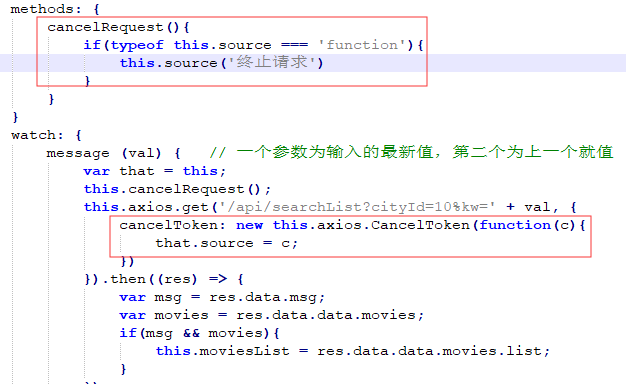
该地方有个防抖的问题，对于快速的多次请求，只执行最后一次请求，之前的请求清空。

方法1：将message中的请求内容外加上延时



这样，在延时时间内被再次触发就会被清理掉上次延时，并重新进行延时处理。

方法2：使用axios自带的方法



# 拉勾网项目源码学习

## watch监听$route的方式（主页baseIndex）

$route 作为vue实例的一个**响应式属性**，和在data中写的属性本质上是一样的，都可以通过this的方式拿到。

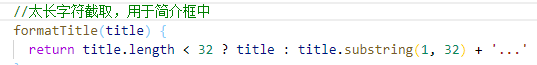
watch中监听的对象默认回调函数中的参数值就是newVal,oldVal。作为 $route 属性来说当然也就是 to 和 from 的概念了。



## v-for和v-if最好不要同时写在一个标签内

## 介绍框中，文字太长需要用省略号处理

（home分页使用searchBar组件）



用CSS也可以实现：

## 移动端轮播图插件swiper

（使用率极高）pc也可以使用 （home分页的banner组件）

npm install vue-awesome-swiper –save

使用到全局：

import Vue from 'vue'

import VueAwesomeSwiper from 'vue-awesome-swiper'

// require styles

import 'swiper/dist/css/swiper.css'

Vue.use(VueAwesomeSwiper, /\* { default global options } \*/)

使用到组件：

import 'swiper/dist/css/swiper.css'

import { swiper, swiperSlide } from 'vue-awesome-swiper'

export default {

components: {

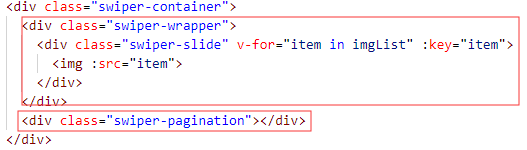
swiper,

swiperSlide

}

}

例：



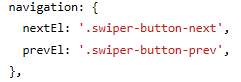
注：类名就这样写



按钮功能定义：（不定义样式就放在swiper-conainer里面）



js中加上：



其他功能可以看swiper官网api

## created()与activated() & deactivated()

created()：在创建vue对象时，当html渲染之前就触发；但是注意，全局vue.js不强制刷新或者重启时只创建一次，也就是说，created()只会触发一次。

**activated()**：在vue对象存活的情况下，进入当前存在activated()函数的页面时，**页面打开时触发**；可用于初始化页面数据等。会在**keep-alive组件激活时调用**；该钩子在**服务器端渲染期间不被调用**

**注意**：activated,deactivated这两个生命周期函数一定是要在使用了**keep-alive组件**后才会有的，否则则不存在

deactivated；keep-alive组件停用时调用。该钩子在服务端渲染期间不被调用。

该项目中的应用：

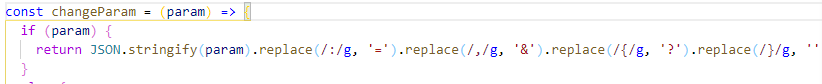


该项目中在home分页面使用了<keep-alive>对<router-view>的包裹，不然，无法生效。



## 封装axios







## mint ui

1. npm install mint-ui –S
2. 引入到入口js文件中：

// 引入全部组件

import Vue from 'vue';

import Mint from 'mint-ui';

Vue.use(Mint);

1. 使用（该项目中使用）：

import Mint, {DatetimePicker, MessageBox, Switch, Toast} from 'mint-ui';

Vue.component(DatetimePicker.name, DatetimePicker);

Vue.component(Switch.name, Switch);

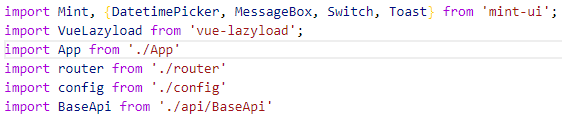
Vue.use(Mint);

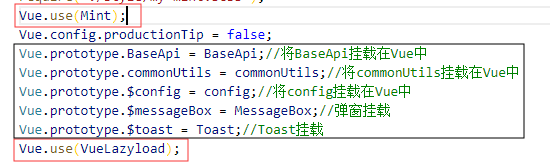


## vuex的使用

见[辅助函数mapState](#_辅助函数mapState)

## axios关于原型链和Vue.use()挂载方式





用Vue.use全局注册的插件，必须含有install方法，vue.use()通过bind()置换install()的this为vue实例环境.

Vue.use 会自动阻止多次注册相同插件，届时只会注册一次该插件   
Vue 是可访问的全局变量时会自动调用 Vue.use()

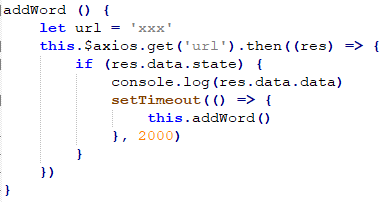
假如有些插件不是专门为VUE服务；里面没有install方法，那么就通过添加到vue原型链上的方式使用。该方式定义后可以通过this.axios调用属性和方法

使用轮询递归的方法每隔几秒请求一次接口刷新数据

window.setInterval(() => {

setTimeout(fun, 0)

}, 30000)



解释：setInterval不会清除定时器队列，每重复执行1次都会导致定时器叠加，最终卡死你的网页。  
但是setTimeout是自带清除定时器的