# Vue

## 前端介绍

HTML+CSS+JS 传统视图层三要素。

其中CSS有很多缺陷，语法不够强大导致模块开发会有很多重复选择器，没有变量函数等复用机制。为了解决这个问题，签到开发人员使用一种CSS预处理器工具。

CSS预处理器定义了一种新的语言，基本思想是一种专门的编程语言，为CSS增加了一种编程的特性。简单来说就是一种生产CSS样式的编程语言！

常见CSS预处理器：SASS,LESS,

Vue:一款用于构建用户界面的JavaScript框架，它基于前端三要素构建，并提供一套声明式，组件化的编程模型。

网络通信axios：前端通信框架，因为Vue的边界很明确，就是为了处理DOM，所以不具备通信能力，此时就需要额外使用一个通信框架与服务器交互，当然也可以直接使用ajax

UI框架：

* Ant-Design
* ElementUI
* BootStrap
* Ama

Vue的引入：

<script src="https://cdn.jsdelivr.net/npm/vue@2.5.16/dist/vue.js">

</script>

## Vue的作用

MVVM模式的实现者

1. Model：模型层，这里表示JS对象
2. View：视图层，这里表示DOM
3. ViewModel：连接视图层和模型的中间件，Vue.js就是该层的实现者

在MVVM模式中，不允许数据和视图直接通信，只能通过ViewModel进行通信，而ViewModel定义了一个观察则会Observer

1. ViewModel能够观察数据的变化，并对视图中对应的内容进行修改
2. 监听视图的变化，通知数据发生改变

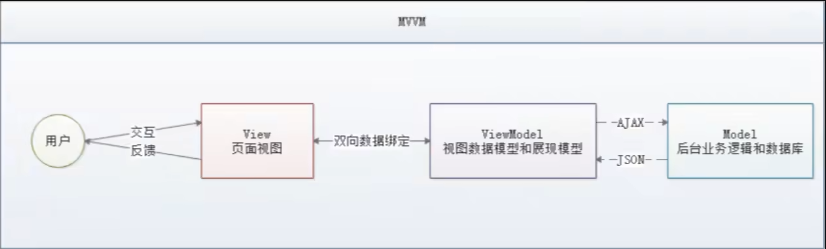
总结：Vue.js是MVVM的实现者，核心就是DOM监听与数据绑定！

示例：需要先引入Vue.js,和引入jquery一样！

示例中div容器中的数据绑定了JS，且由于ViewModel的监听功能，当数据发生改变，view层也会改变数据！



原理图：



MVVM思想：模式的核心与MVC一致，都是向分离视图和模型！

好出：低耦合，视图独立于model变化和修改，一个viewmodel可以绑定到不同的view上，当view和model的改变互不影响

## 虚拟DOM

虚拟DOM就是一个普通的JS对象，只不过该对象描述了真实DOM结构，所以称之为虚拟DOM。

## Vue生命周期

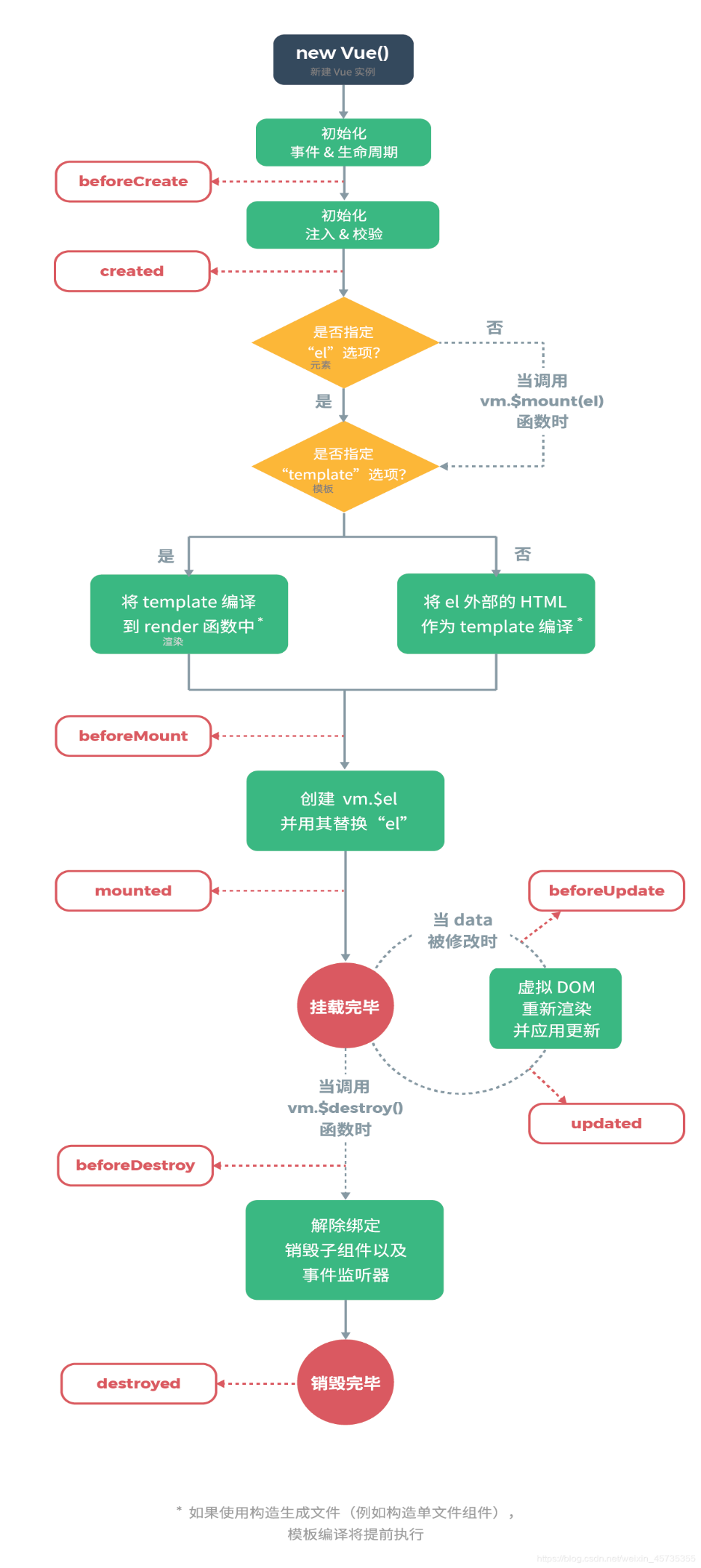
Vue的声明周期即vue从创建到销毁的过程，就是生命周期。也就是从开始创建，初始化数据，编译模板，挂载DOM-》渲染，更新-》渲染，卸载等一系列过程。

声命周期的作用：vue生命周期中有多个事件钩子，让我们在控制整个vue实例过程中更容易形成好的逻辑！

声明周期总共分为8个阶段：创建前后，载入前后，更新前后，销毁前后

钩子的使用方法：

* beforecreate:在此阶段加loading事件，加载实例时触发
* created:初始化完成时的事件写在这里
* mouneted:挂载元素，获取DOM节点
* updated:如果对数据统一处理，在这里写上对应函数
* beforeDestory：可以左一个确认停止事件的确认框
* nextTick:更新数据后立刻操作Dom



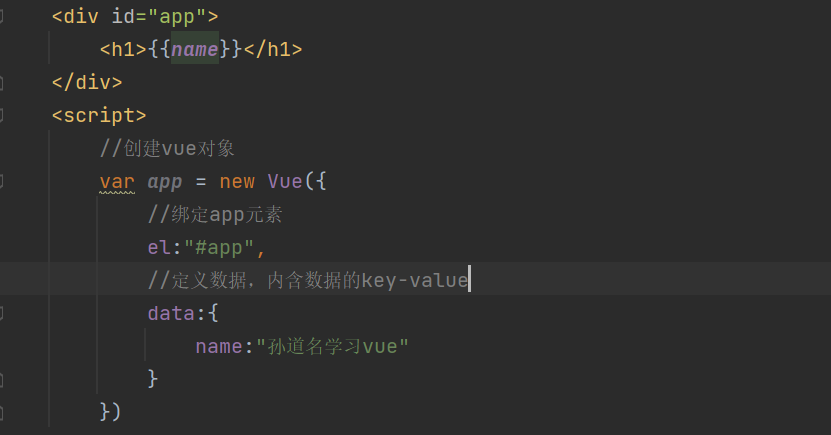
### 数据显示

#### 插值表达式

插值表达式即用户把vue定义的数据显示在页面上

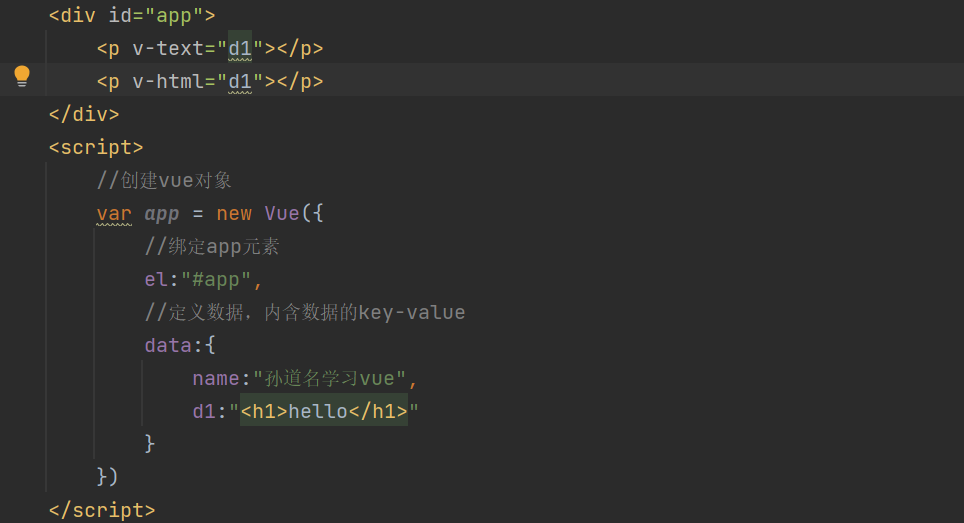
语法：{{变量名/对象.属性名}}

插值表达式的闪烁问题：当网速过慢，网页可能会显示出该表达式的原始模样，然后数据加载后覆盖掉原始样貌，从而产生数据闪烁问题



#### v-text和v-html

v-text和v-html专门用于显示数据，前者把数据作为纯文本处理，后者可以接卸数据内部的标签



#### v-bind

作用类似于插值表达式，只不过v-bind主要用于动态设置标签的属性值

语法：

<标签名 v-bind:标签属性名=“vue实例中的数据值”/>

简化

<标签名 :标签属性名=“vue实例中的数据值”/>

<input :value="message">

#### 双向数据绑定

Vue.js是一个MVVM框架，前面的示例都是数据改变，视图改变。双向数据绑定即视图发生变化，数据也会改变。

值得注意的是：我们所说的数据双向绑定是针对UI控件来说的，非UI控件不会涉及到数据的双向绑定。

我们使用v-model指令在表单input,textarea,select元素上创建双向数据绑定，它会将用户输入数据和表单数据双向绑定。

v-model会忽略所有表单元素的初始值，而将vue实例数据作为数据来源。

示例：



单选框绑定数据：



下拉框绑定数据：注意如果v-model表达式的初始值未匹配任何选项，select元素会被渲染成未选中状态，在IOS中，这会使用户无法选择第一个选项，所以这里添加了一个提示选项。



### 事件绑定

事件绑定标准语法：

v-on:事件=“vue表达式/函数名”

@事件=“vue表达式/函数名”

绑定的方法写在vue的methods属性中

<button @click="f1">点击弹窗</button>

<button v-on:click="f1">点击弹窗</button>

事件绑定的参数，函数如果包含参数，则再引用函数时写入参数

#### 事件修饰符

概述：对事发发生的范围进行修饰

语法：@事件.事件修饰符=“vue表达式/函数名”

事件修饰符：

1. .stop:阻止事件冒泡，也就是当前元素发生事件，当前元素的父元素不发生事件
2. .prevent:阻止默认事件发生
3. .capture:使用事件捕获模式，主动获取子元素发生事件，把获取到的事件当自己的事件执行
4. .self:只有元素自身触发事件才执行（冒泡或捕获的都不执行）
5. .once:只执行一次

#### 键盘事件

@keydown

@keyup

### 循环遍历

v-for

遍历数组语法：index是数组的索引，item的数组内的元素

* v-for=“item in items”
* v-for=“（item,index）in items”

<ul>  
 <li v-for="(p,index) in people">  
 {{p}}--{{index}}  
 </li>  
</ul>

遍历对象：value是对象的值，key是键，index是索引

* v-for=“value in object”
* v-for= ”(value,key) in object”
* v-for= “(value,key,index) in object”

### 逻辑判断

v-if 和 v-show

概述：v-if和v-show根据条件的结果决定是否显示内容

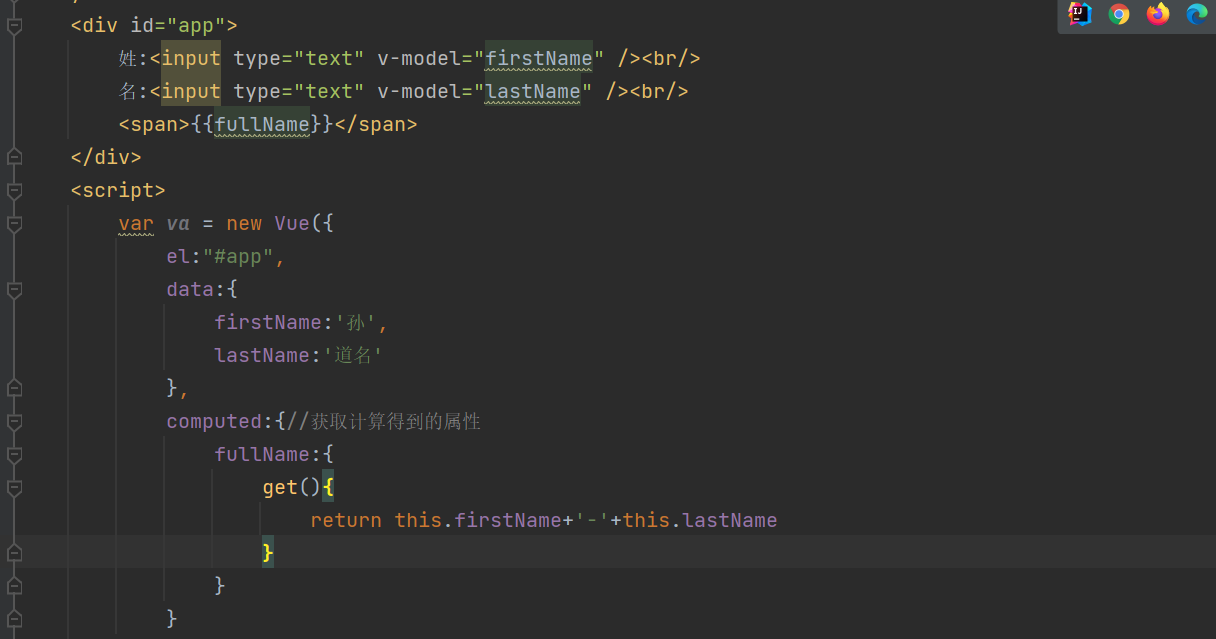
1. v-if:条件不满足时，元素不会存在
2. v-show:条件不满足时，元素不会显示，但是仍然存在

//show是data中的一个布尔值

<button @click="show = !show">点我改变show</button>  
<h1 v-if="show">hello v-if</h1>  
<h1 v-show="show">hello v-show</h1>

### 计算属性

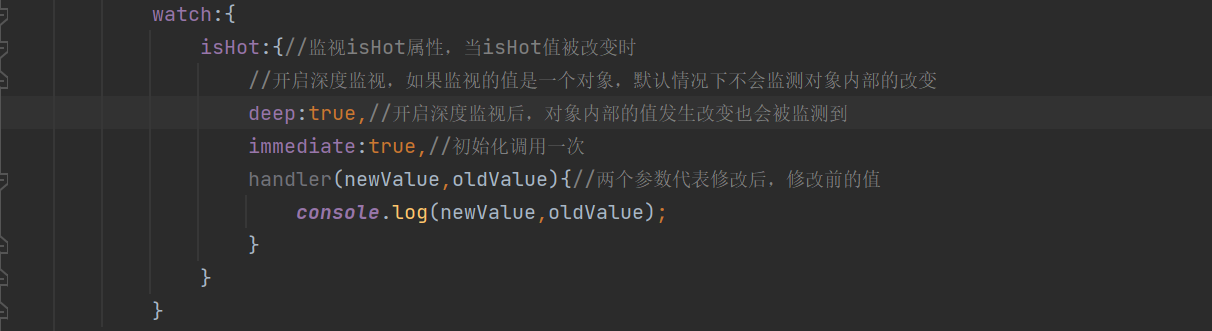
简单来说，就是一个能够将计算结果缓存起来的属性，将行为转化成静态属性。



methods的值通过方法调用，computed的值通过属性调用，且存在于内存中。

fullName的get方法初次读取会执行一次，当依赖的数据发生改变时会被再次调用！

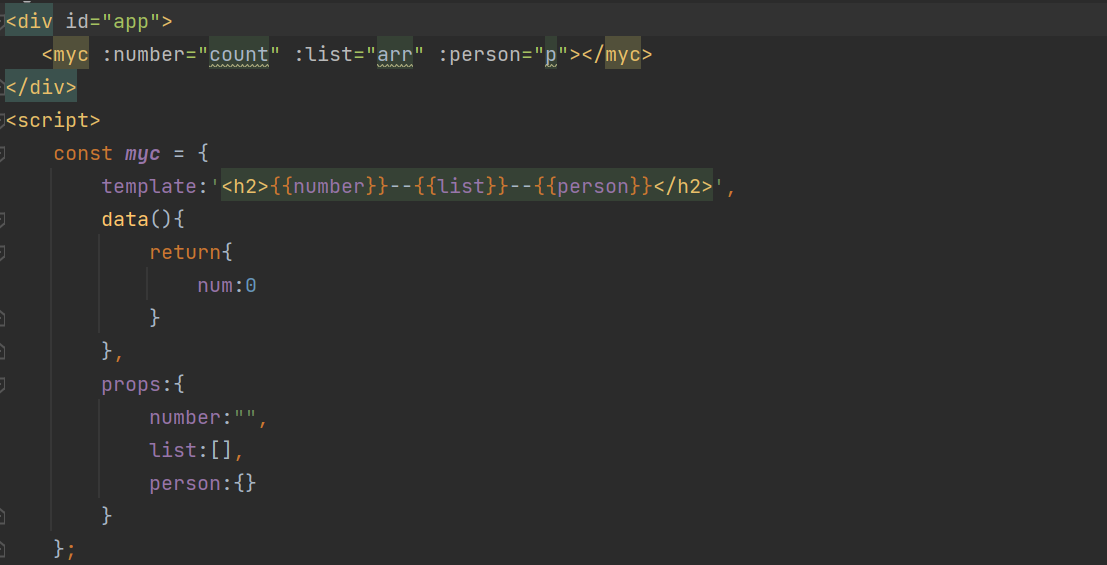
### 监视属性



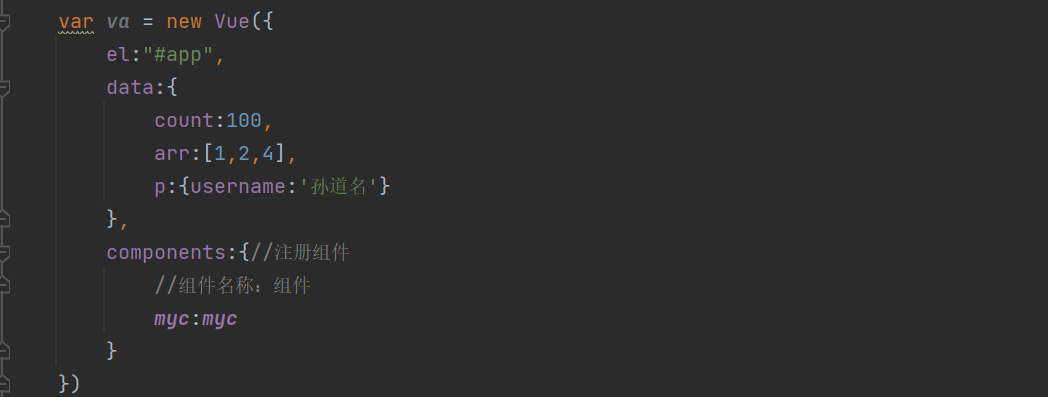
### Vue组件

自定义组件，并且在vue实例中将组件注册！（components属性）

定义组件的模板，定义三个属性number,list,person



vue中注册组件，并绑定数据



如果组件名称中包含大写字母，例如：myList:xxx,使用组件时写成<my-List>

--

组件通信问题：

子组件无法直接使用父组件中的数据，如果需要使用，必须由父组件把数据传递给子组件才可以。

本质：让子组件中的属性与父组件中的属性进行关联绑定，然后子组件使用该属性。

### axios异步请求

axios是一个基于promise的HTTP库，主要发送异步请求以获取数据，使用axios前需要引入axios

<script type="text/javascript" src="https://cdn.bootcss.com/axios/0.19.0-beta.1/axios.js"></script>

常见方法：

* axios(config)
* axios.get(url,[config])
* axios.post(url,[data])

请求config常用参数：

* url:请求地址
* method:请求方式
* params:{参数名：参数值………}
* data:{参数名：参数值}
* responseType:json

响应数据常用参数：

* data:{} //响应的数据
* status:200 //响应的状态
* statusText:OK //状态描述
* headers:{}, //响应头
* config:{} //其他配置信息

示例：和JS中的AJAX类似，发送异步请求确定请求地址，请求参数，以及回调函数（即请求成功和请求失败后的行为）



### Vue路由

vue的一个插件库，专门用来实现SPA应用

SPA应用：

* 单页Web应用
* 整个应用只有一个完整的页面
* 点击导航链接不会刷新页面，只会做页面的局部更新
* 数据需要异步请求获取

路由：一个路由就是一组映射关系key:value,key为路径，value可能是函数或者组件

后端路由：key为控制器地址，value就是函数，用于处理客户端请求

前端路由：value是一个组件，用于展示页面内容，当浏览器路径改变，对应的组件就会显示。

### 插槽

示例：

<div id="app">  
 <todo>  
 <todo-title slot="todo-title" v-bind:title="title">

</todo-title>  
 <todo-item slot="todo-item" v-for="i in todoItems"

v-bind:item="i"></todo-item>  
 </todo>  
 </div>  
<script>  
 Vue.component("todo",{  
 template:  
 '<div>'+  
 '<slot name="todo-title"></slot>'+  
 '<ul>' +  
 '<slot name="todo-item"></slot>'+  
 ' </ul>'+  
 '</div>'  
 })  
 Vue.component("todo-title",{  
 props:['title'],  
 template: '<div>{{title}}</div>'  
 })  
 Vue.component("todo-item",{  
 props:['item'],  
 template: '<li>{{item}}</li>'  
 })  
 var ***vm*** = new Vue({  
 el:"#app",  
 data:{  
 title:"sss",  
 message:"hello",  
 todoItems:["java","c","python"]  
 }  
 });