Vue 技术栈(全家桶)

尚硅谷前端研究院

# 第 1 章：Vue 核心

## Vue 简介

### 官网

* + - 1. 英文官网: <https://vuejs.org/>
      2. 中文官网: <https://cn.vuejs.org/>

### 介绍与描述

* + - 1. 动态构建用户界面的**渐进式** JavaScript 框架
      2. 作者: 尤雨溪

### Vue 的特点

* + - 1. 遵循 **MVVM** 模式
      2. 编码简洁, 体积小, 运行效率高, 适合移动/PC 端开发
      3. 它本身只关注UI, 也可以引入其它第三方库开发项目

### 与其它 JS 框架的关联

* + - 1. 借鉴Angular 的**模板**和**数据绑定**技术
      2. 借鉴React 的**组件化**和**虚拟DOM** 技术

### Vue 周边库

* + - 1. vue-cli: vue 脚手架
      2. vue-resource
      3. axios
      4. vue-router: 路由
      5. vuex: 状态管理
      6. element-ui: 基于vue 的UI 组件库(PC 端)

……

## 初识 Vue



* 1. **模板语法**

### 效果



* + 1. **模板的理解**

html 中包含了一些JS 语法代码，语法分为两种，分别为：

* + - 1. 插值语法（双大括号表达式）
      2. 指令（以v-开头）

### 插值语法

* + - 1. 功能: 用于解析标签体内容
      2. 语法: {{xxx}} ，xxxx 会作为js 表达式解析

### 指令语法

* + - 1. 功能: 解析标签属性、解析标签体内容、绑定事件
      2. 举例：v-bind:href = 'xxxx' ，xxxx 会作为js 表达式被解析
      3. 说明：Vue 中有有很多的指令，此处只是用v-bind 举个例子

## 数据绑定

### 效果



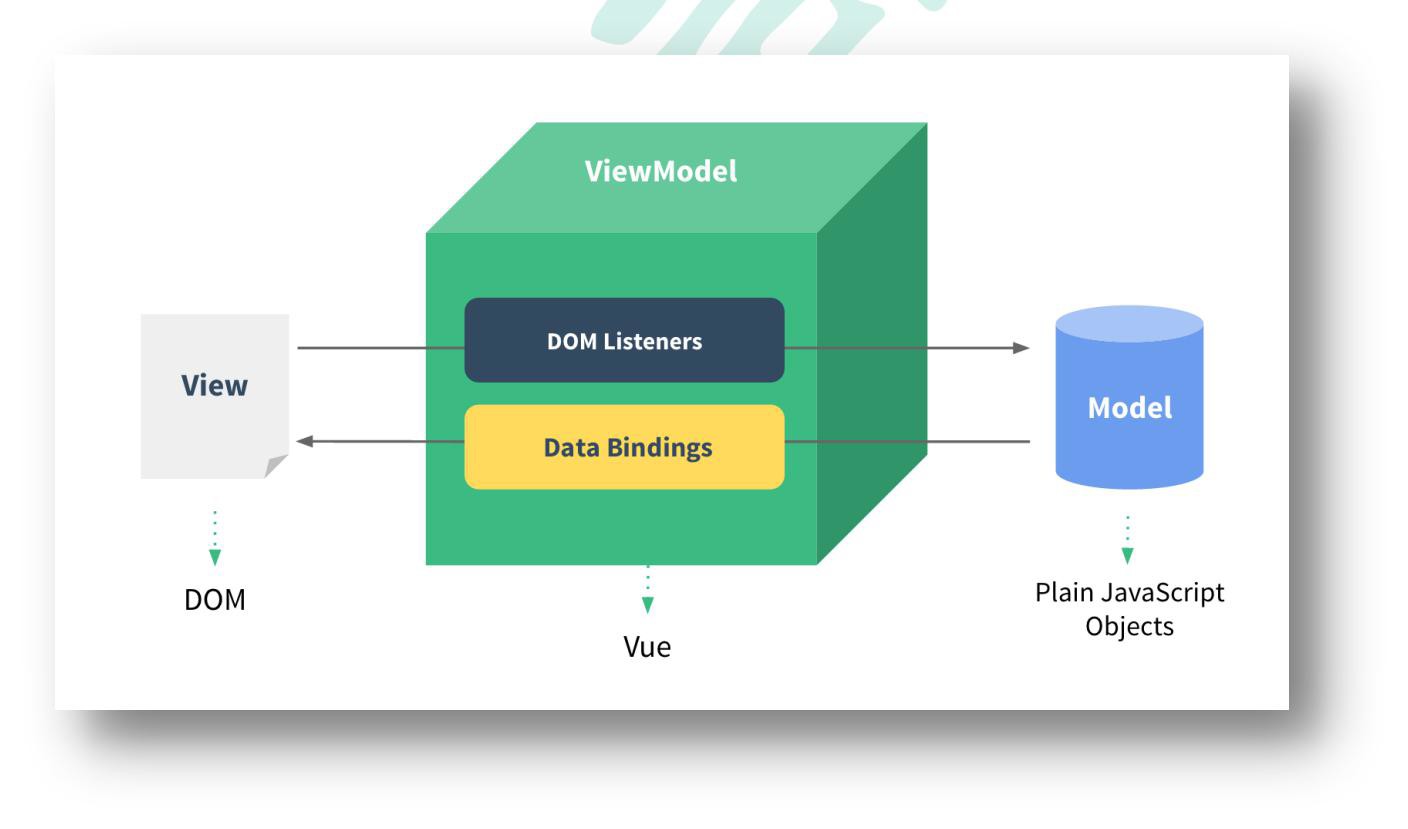
* + 1. **单向数据绑定**
       1. 语法：v-bind:href ="xxx" 或简写为 :href
       2. 特点：数据只能从data 流向页面

### 双向数据绑定

* + - 1. 语法：v-mode:value="xxx" 或简写为 v-model="xxx"
      2. 特点：数据不仅能从data 流向页面，还能从页面流向 data

## MVVM 模型

1. M：模型(Model) ：对应data 中的数据
2. V：视图(View) ：模板
3. VM：视图模型(ViewModel) ： Vue 实例对象



## 事件处理

### 效果



* + 1. **绑定监听**
       1. v-on:xxx="fun"
       2. @xxx="fun"
       3. @xxx="fun(参数)"
       4. 默认事件形参: event
       5. 隐含属性对象: $event

### 事件修饰符

* + - 1. .prevent : 阻止事件的默认行为 event.preventDefault()
      2. .stop : 停止事件冒泡 event.stopPropagation()

### 按键修饰符

* + - 1. keycode : 操作的是某个keycode 值的键
      2. .keyName : 操作的某个按键名的键(少部分)

## 计算属性与监视

### 效果



* + 1. **计算属性-computed**
       1. 要显示的数据不存在，要通过计算得来。
       2. 在computed 对象中定义计算属性。
       3. 在页面中使用{{方法名}}来显示计算的结果。

### 监视属性-watch



* + - 1. 通过通过vm 对象的$watch()或watch 配置来监视指定的属性
      2. 当属性变化时, 回调函数自动调用, 在函数内部进行计算

## class 与 style 绑定

### 理解

* + - 1. 在应用界面中, 某个(些)元素的样式是变化的
      2. class/style 绑定就是专门用来实现动态样式效果的技术

### class 绑定

* + - 1. :class='xxx'
      2. 表达式是字符串: 'classA'
      3. 表达式是对象: {classA:isA, classB: isB}
      4. 表达式是数组: ['classA', 'classB']

### style 绑定

* + - 1. :style="{ color: activeColor, fontSize: fontSize + 'px' }"
      2. 其中activeColor/fontSize 是data 属性

## 条件渲染

### 条件渲染指令

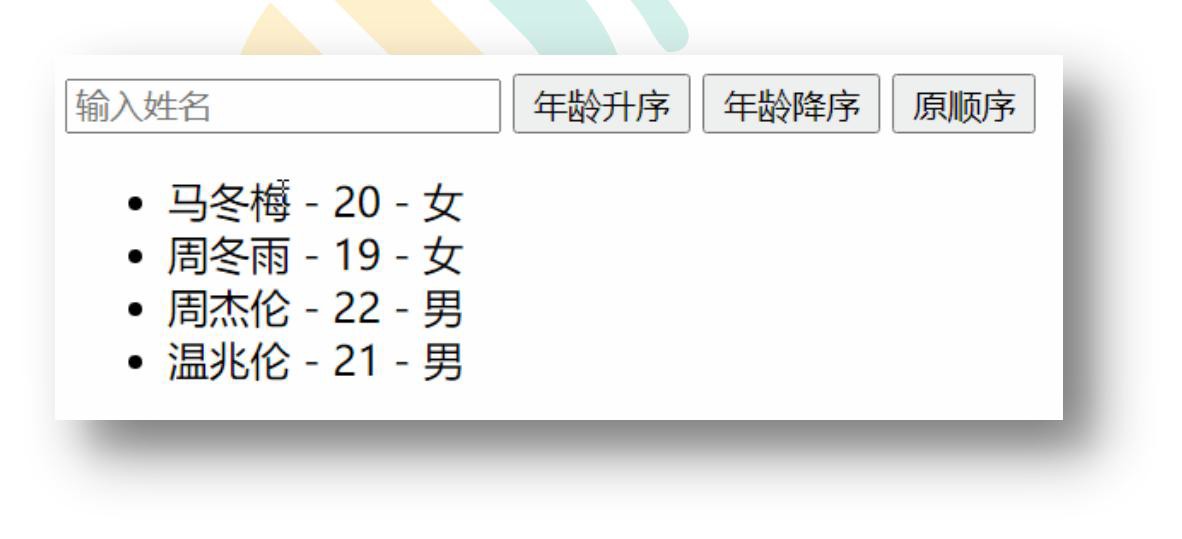
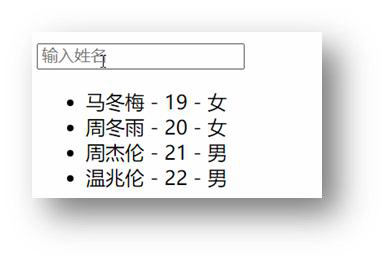
* + - 1. v-if 与v-else
      2. v-show

### 比较v-if 与v-show

* + - 1. 如果需要频繁切换 v-show 较好
      2. 当条件不成立时, v-if 的所有子节点不会解析(项目中使用)

## 列表渲染

### 效果



* + 1. **列表显示指令**

遍历数组: v-for / index 遍历对象: v-for / key

## 收集表单数据



* 1. **过滤器**

### 效果



* + 1. **理解过滤器**

1. 功能: 对要显示的数据进行特定格式化后再显示
2. 注意: 并没有改变原本的数据, 是产生新的对应的数据

## 内置指令与自定义指令

### 常用内置指令

1. v-text : 更新元素的 textContent
2. v-html : 更新元素的 innerHTML
3. v-if : 如果为true, 当前标签才会输出到页面
4. v-else: 如果为false, 当前标签才会输出到页面
5. v-show : 通过控制display 样式来控制显示/隐藏
6. v-for : 遍历数组/对象
7. v-on : 绑定事件监听, 一般简写为@
8. v-bind : 绑定解析表达式, 可以省略v-bind
9. v-model : 双向数据绑定
10. v-cloak : 防止闪现, 与css 配合: [v-cloak] { display: none }

### 自定义指令

##### 注册全局指令

Vue.directive('my-directive', function(el, binding){ el.innerHTML = binding.value.toupperCase()

})

##### 注册局部指令

directives : {

'my-directive' : { bind (el, binding) {

el.innerHTML = binding.value.toupperCase()

}

}

}

1) 使用指令

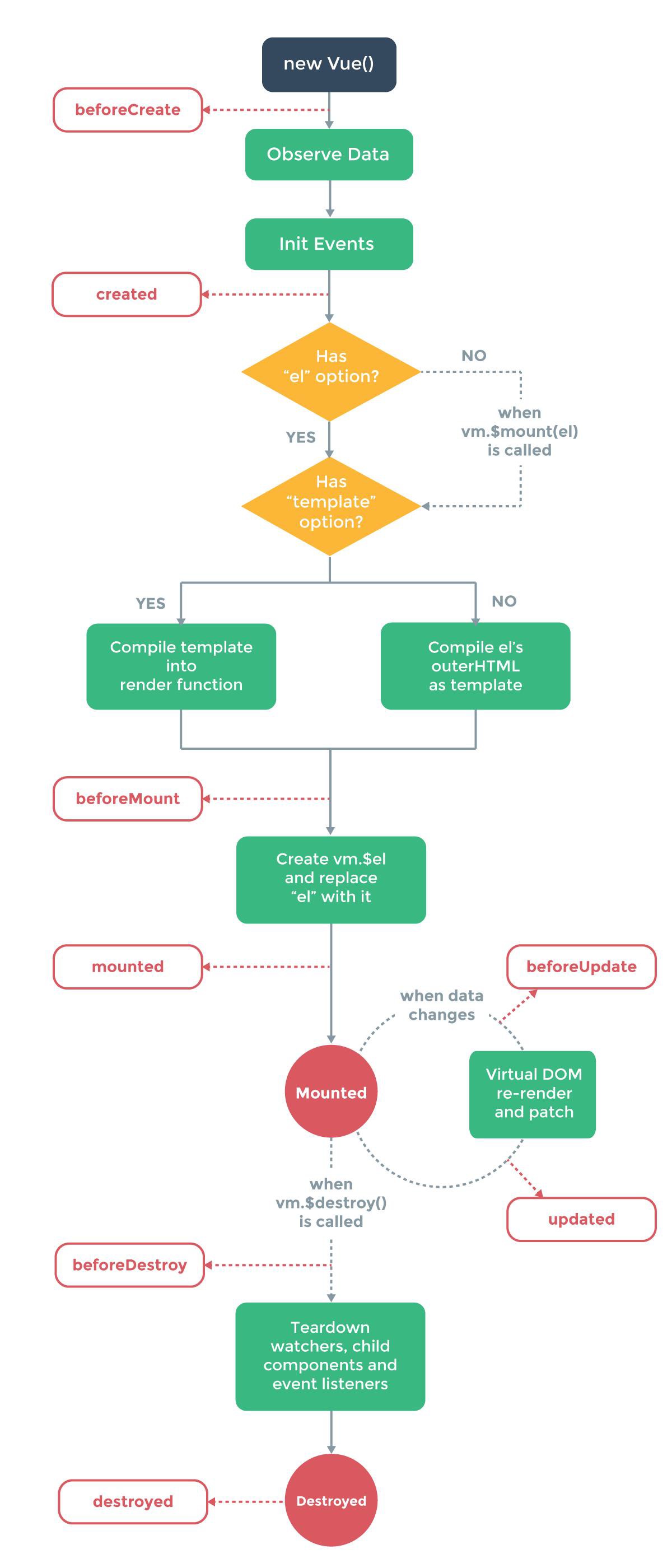
v-my-directive='xxx'

## Vue 实例生命周期

* + 1. **效果**



* + 1. **生命周期流程图**



* + 1. **vue 生命周期分析**

1. 初始化显示
   * beforeCreate()
   * created()
   * beforeMount()
   * mounted()
2. 更新状态: this.xxx = value
   * beforeUpdate()
   * updated()
3. 销毁vue 实例: vm.$destory()
   * beforeDestory()
   * destoryed()

### 常用的生命周期方法

* + - 1. mounted(): 发送ajax 请求, 启动定时器等异步任务
      2. beforeDestory(): 做收尾工作, 如: 清除定时器

# 第 2 章：Vue 组件化编程

## 模块与组件、模块化与组件化

### 模块

* + - 1. 理解: 向外提供特定功能的 js 程序, 一般就是一个js 文件
      2. 为什么: js 文件很多很复杂
      3. 作用: 复用js, 简化js 的编写, 提高js 运行效率

### 组件

* + - 1. 理解: 用来实现局部(特定)功能效果的代码集合(html/css/js/image…..)
      2. 为什么: 一个界面的功能很复杂
      3. 作用: 复用编码, 简化项目编码, 提高运行效率

### 模块化

当应用中的js 都以模块来编写的, 那这个应用就是一个模块化的应用。

### 组件化

当应用中的功能都是多组件的方式来编写的, 那这个应用就是一个组件化的应用,。

## 非单文件组件

* + 1. 模板编写没有提示
    2. 没有构建过程, 无法将ES6 转换成ES5
    3. 不支持组件的CSS
    4. 真正开发中几乎不用

## 单文件组件

### 一个.vue 文件的组成(3 个部分)

##### 模板页面

<template> 页面模板

</template>

##### JS 模块对象

<script> export default {

data() {return {}}, methods: {},

computed: {}, components: {}

}

</script>

##### 样式

<style> 样式定义

</style>

### 基本使用

1. 引入组件
2. 映射成标签
3. 使用组件标签

# 第 3 章：使用 Vue 脚手架

## 初始化脚手架

### 说明

1. Vue 脚手架是Vue 官方提供的标准化开发工具（开发平台）。
2. 最新的版本是 4.x。
3. 文档: <https://cli.vuejs.org/zh/>。

## 具体步骤

第一步（仅第一次执行）：全局安装@vue/cli。

npm install -g @vue/cli

第二步：**切换到你要创建项目的目录**，然后使用命令创建项目

vue create xxxx 第三步：启动项目

npm run serve

备注：

https://registry.npm.taobao.org

npm config set registry

1. 如出现下载缓慢请配置 npm 淘宝镜像：

2. Vue 脚手架隐藏了所有webpack 相关的配置，若想查看具体的webpakc 配置， 请执行：vue inspect > output.js

* + 1. **模板项目的结构**

**├── node\_modules**

**├── public**

**│ ├── favicon.ico: 页签图标**

**│ └── index.html: 主页面**

**├── src**

**│ ├── assets: 存放静态资源**

**│ │ └── logo.png**

**│ │── component: 存放组件**

**│ │ └── HelloWorld.vue**

**│ │── App.vue: 汇总所有组件**

**│ │── main.js: 入口文件**

**├── .gitignore: git 版本管制忽略的配置**

**├── babel.config.js: babel 的配置文件**

**├── package.json: 应用包配置文件**

**├── README.md: 应用描述文件**

**├── package-lock.json：包版本控制文件**

## ref 与 props

**ref**

1. **作用：**用于给节点打标识
2. **读取方式：**this.$refs.xxxxxx

## props

* 1. **作用：**用于父组件给子组件传递数据

##### 读取方式一: 只指定名称

props: ['name', 'age', 'setName']

##### 读取方式二: 指定名称和类型

props: {

name: String, age: Number,

setNmae: Function

}

##### 读取方式三: 指定名称/类型/必要性/默认值

props: {

name: {type: String, required: true, default:xxx},

}

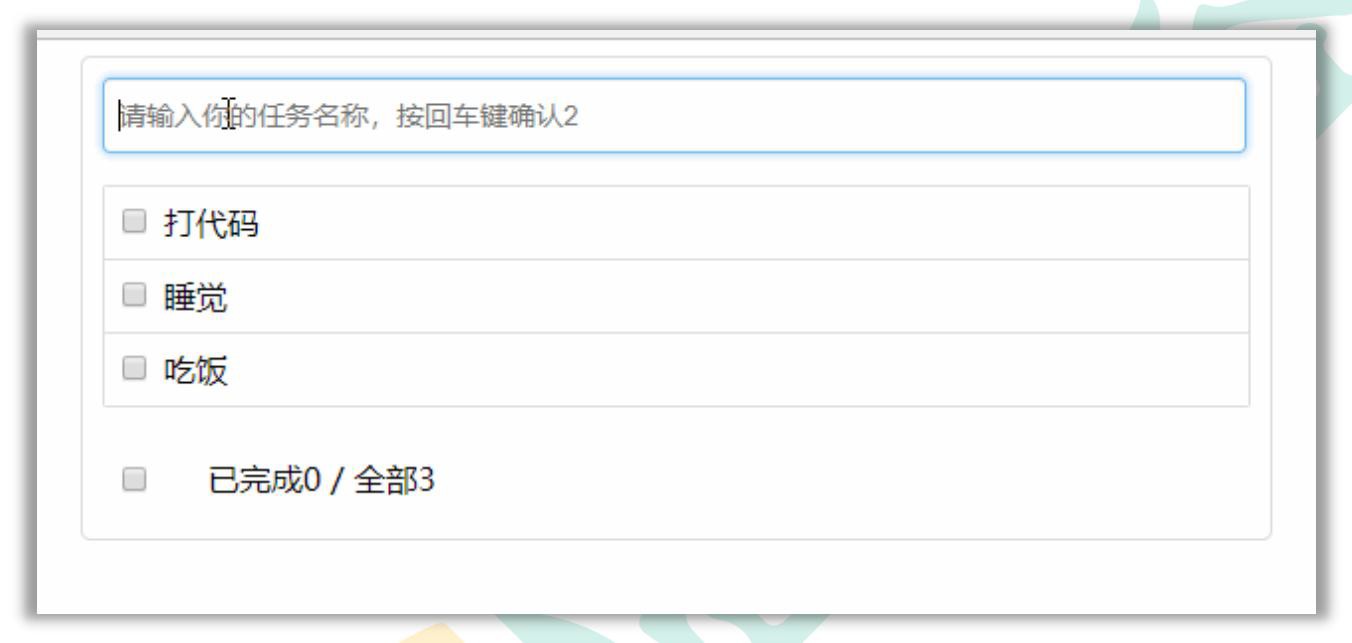
## 混入

1. Vue 插件是一个包含install 方法的对象
2. 通过install 方法给Vue 或Vue 实例添加方法, 定义全局指令等

## 插件

1. Vue 插件是一个包含install 方法的对象
2. 通过install 方法给Vue 或Vue 实例添加方法, 定义全局指令等

## Todo-list 案例



### 组件化编码流程（通用）

1. 实现静态组件：抽取组件，使用组件实现静态页面效果
2. 展示动态数据：
   1. 数据的类型、名称是什么？
   2. 数据保存在哪个组件？ 3.交互——从绑定事件监听开始

## Vue 中的自定义事件

**绑定事件监听**

<**Header @addTodo="**addTodo**"**/>

或者

<**Header ref="header"**/> **this**.**$refs**.header.$on(**'addTodo'**, **this**.addTodo)

**触发事件**

**this**.$emit(**'addTodo'**, todo)

## 全局事件总线

### 理解

1. Vue 原型对象上包含事件处理的方法
   1. $on(eventName, listener): 绑定自定义事件监听
   2. $emit(eventName, data): 分发自定义事件
   3. $off(eventName): 解绑自定义事件监听
   4. $once(eventName, listener): 绑定事件监听, 但只能处理一次
2. 所有组件实例对象的原型对象的原型对象就是Vue 的原型对象
   1. 所有组件对象都能看到Vue 原型对象上的属性和方法
   2. Vue.prototype.$bus = new Vue(), 所有的组件对象都能看到$bus 这个属性对象
3. 全局事件总线
   1. 包含事件处理相关方法的对象(只有一个)
   2. 所有的组件都可以得到
      1. **指定事件总线对象**

**new** Vue({

beforeCreate () { *// 尽量早的执行挂载全局事件总线对象的操作*

Vue.**prototype**.**$globalEventBus** = **this**

},

}).$mount(**'#root'**)

* + 1. **绑定事件**

**this**.**$globalEventBus**.$on(**'deleteTodo'**, **this**.deleteTodo)

* + 1. **分发事件**

**this**.**$globalEventBus**.$emit(**'deleteTodo'**, **this**.**index**)

* + 1. **解绑事件**

**this**.**$globalEventBus**.$off(**'deleteTodo'**)

## 消息订阅与发布

### 理解

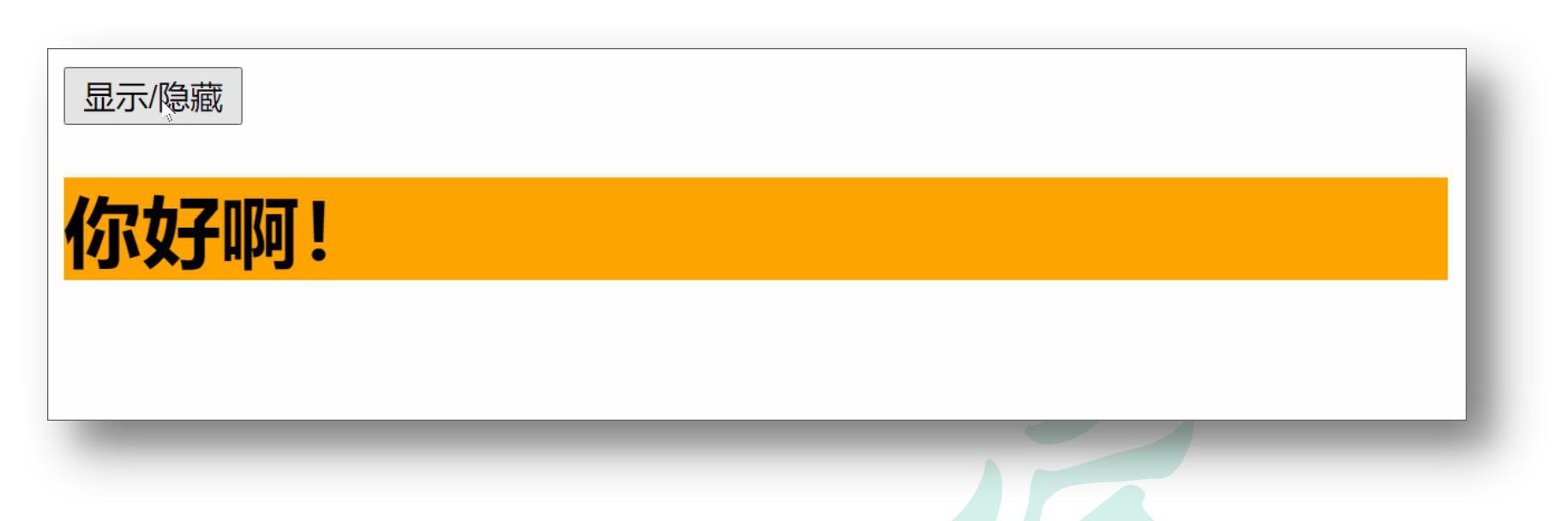
1. 这种方式的思想与全局事件总线很相似
2. 它包含以下操作:
   1. 订阅消息 --对应绑定事件监听
   2. 发布消息 --分发事件
   3. 取消消息订阅 --解绑事件监听
3. 需要引入一个消息订阅与发布的第三方实现库: [**PubSubJS**](https://github.com/mroderick/PubSubJS)

### 使用PubSubJS

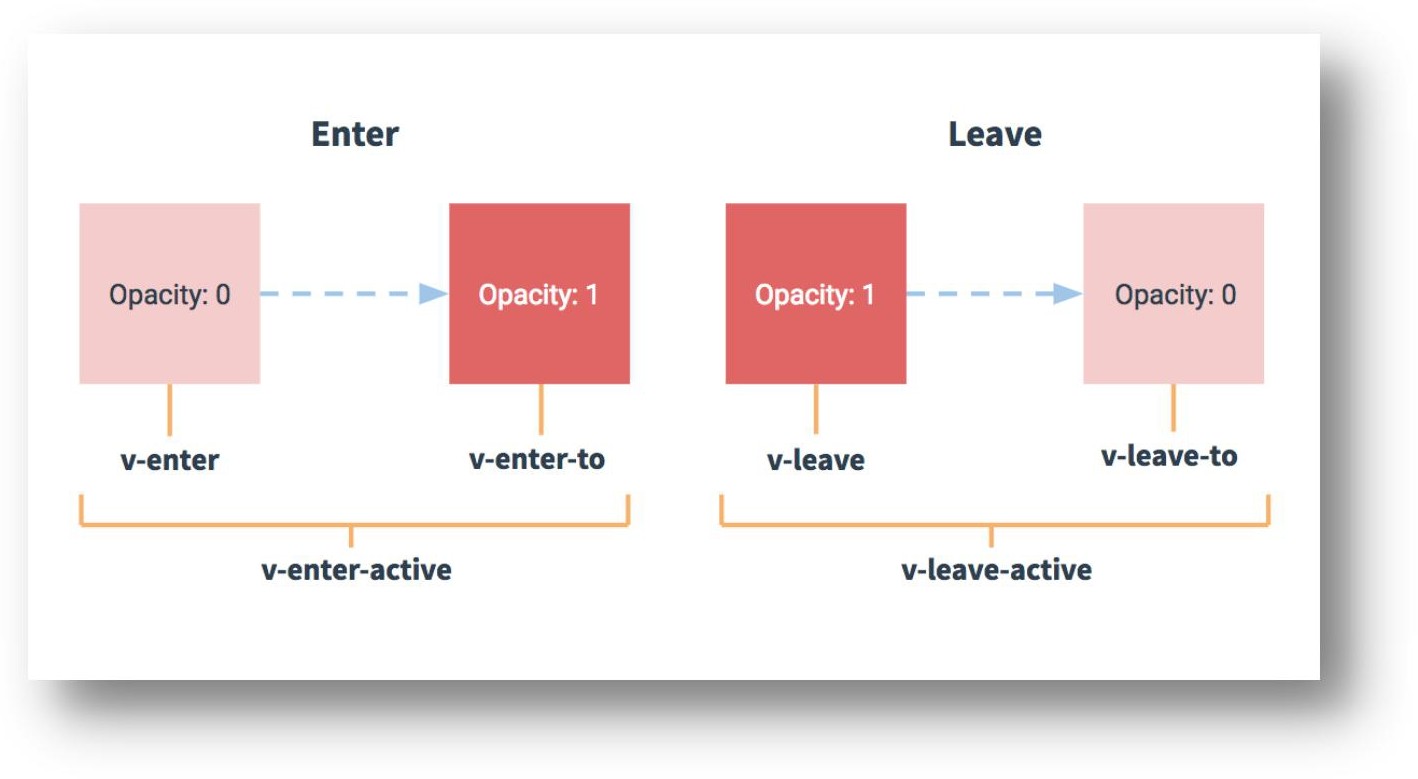
1. 在线文档: <https://github.com/mroderick/PubSubJS>
2. 下载: npm install -S pubsub-js
3. 相关语法
   1. import PubSub from 'pubsub-js' // 引入
   2. PubSub.subscribe(‘msgName’, functon(msgName, data){ })
   3. PubSub.publish(‘msgName’, data): 发布消息, 触发订阅的回调函数调用
   4. PubSub.unsubscribe(token): 取消消息的订阅

## 过度与动画

### 效果



* + 1. **vue 动画的理解**
       1. 操作css 的trasition 或animation
       2. vue 会给目标元素添加/移除特定的class
       3. 过渡的相关类名：
          1. xxx-enter-active: 指定显示的transition
          2. xxx-leave-active: 指定隐藏的transition
          3. xxx-enter/xxx-leave-to: 指定隐藏时的样式



### 基本过渡动画的编码

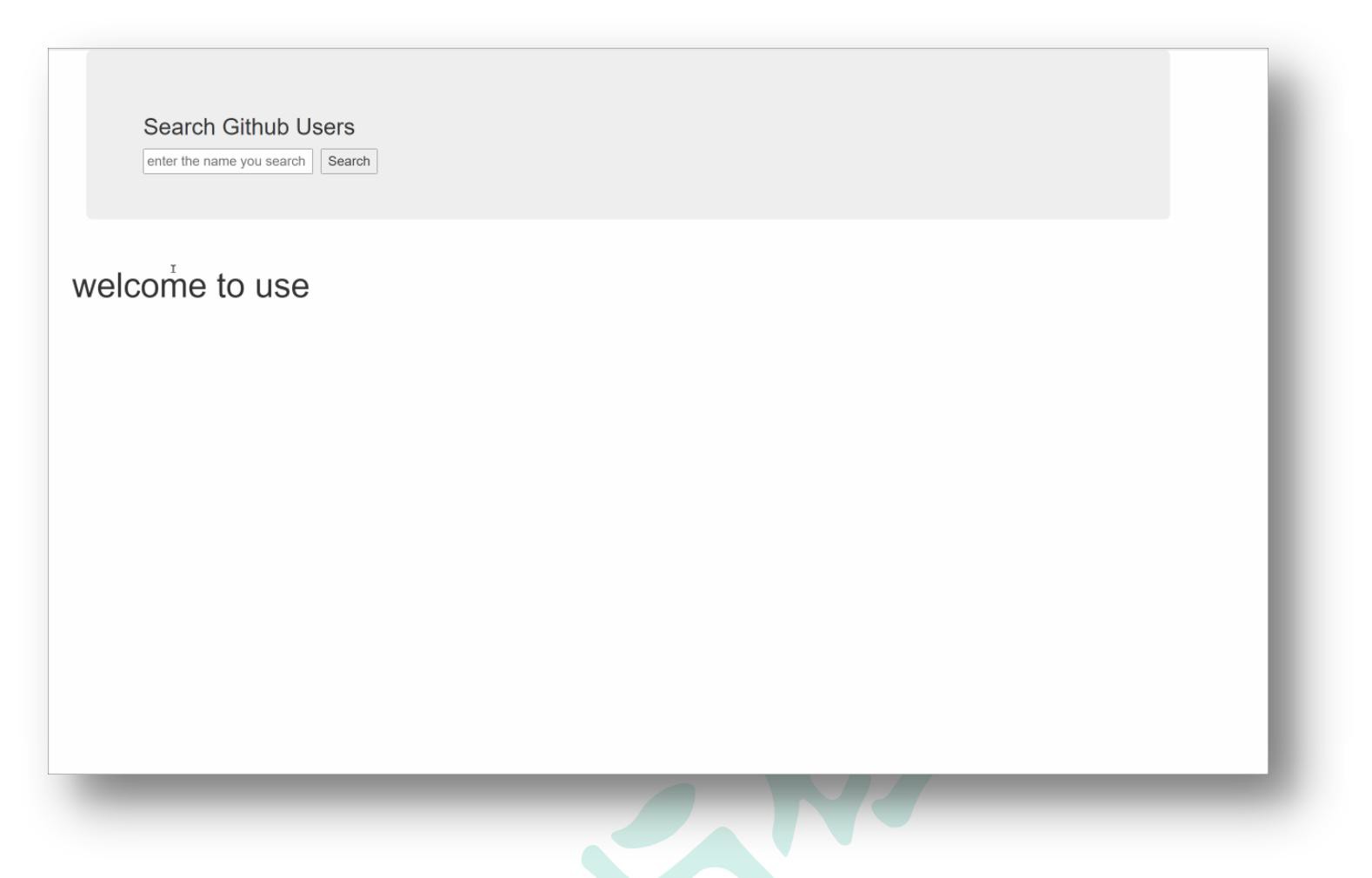
1. 在目标元素外包裹<transition name="xxx">
2. 定义class 样式
   1. 指定过渡样式: transition
   2. 指定隐藏时的样式: opacity/其它

# 第 4 章：Vue 中的 ajax

## 解决开发环境 Ajax 跨域问题

使用代理服务器

* 1. **github 用户搜索案例**
     1. **效果**



* + 1. **接口地址**

<https://api.github.com/search/users?q=xxx>

## vue 项目中常用的 2 个 Ajax 库

### axios

通用的Ajax 请求库, 官方推荐，使用广泛

### vue-resource

vue 插件库, vue1.x 使用广泛，**官方已不维护。**

## slot 插槽

### 效果

效果一（不使用插槽）：



效果二（默认插槽）：



效果三（具名插槽）：



效果三（作用域插槽）：



### 理解

父组件向子组件传递带数据的标签，当一个组件有不确定的结构时, 就需要使用slot 技术，注意：插槽内容是在父组件中编译后, 再传递给子组件的。

### 分类

1. 默认插槽
2. 命名插槽
3. 作用域插槽

# 第 5 章：vuex

## 理解 vuex

### vuex 是什么

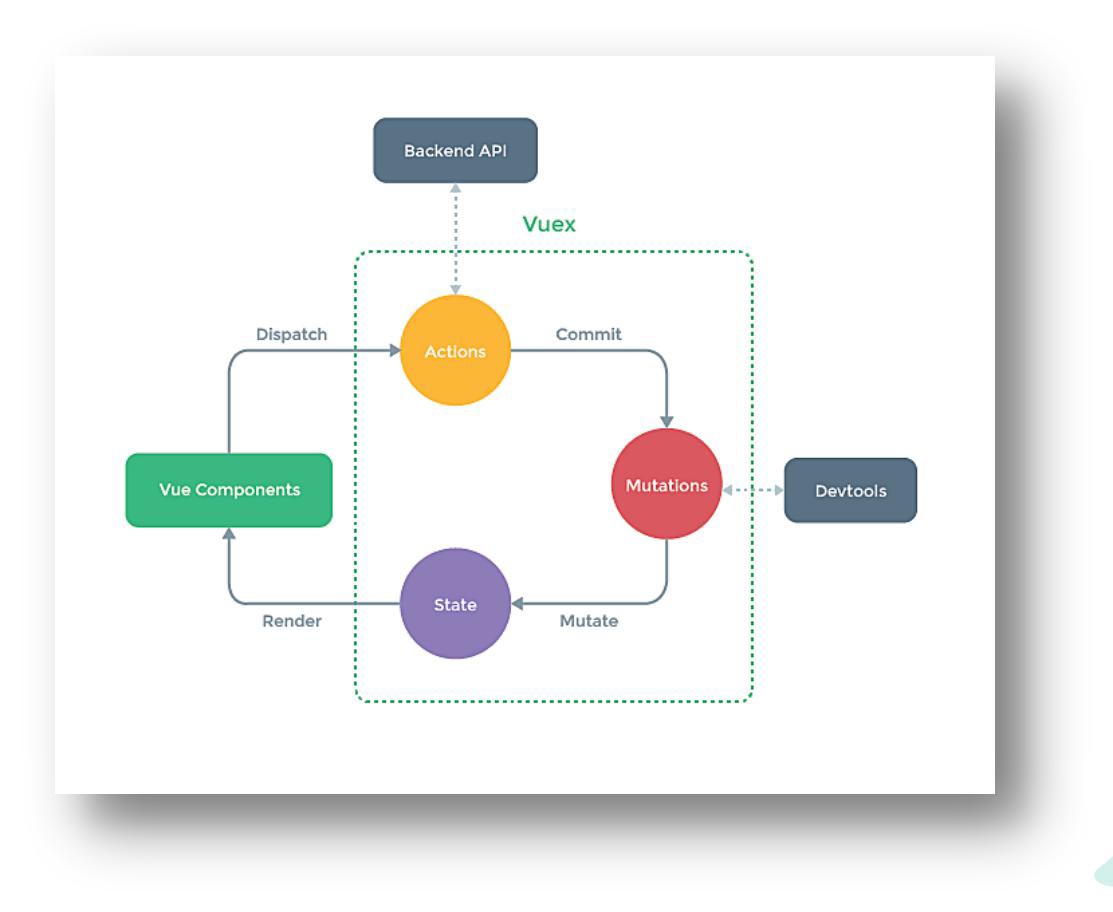
1. 概念：专门在Vue 中实现集中式状态（数据）管理的一个Vue 插件，对vue 应用中多个组件的共享状态进行集中式的管理（读/写），也是一种组件间通信的方式，且适用于任意组件间通信。
2. Github 地址: <https://github.com/vuejs/vuex>

### 什么时候使用 Vuex

* + - 1. 多个组件依赖于同一状态
      2. 来自不同组件的行为需要变更同一状态
    1. **案例**

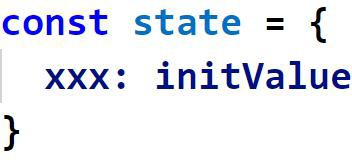


* + 1. **Vuex 工作原理图**



## vuex 核心概念和 API

* + 1. **state**
       1. vuex 管理的状态对象
       2. 它应该是唯一的
       3. 示例代码：

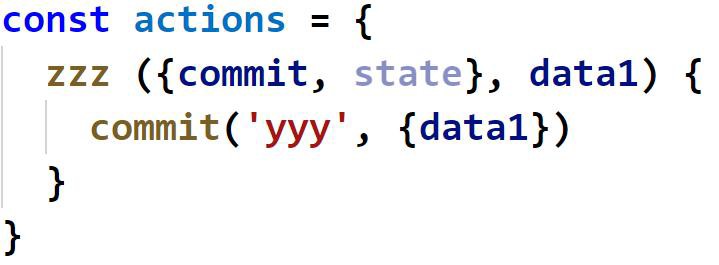


## actions

* + - 1. 值为一个对象，包含多个响应用户动作的回调函数
      2. 通过commit( )来触发mutation 中函数的调用, 间接更新state
      3. 如何触发actions 中的回调？

在组件中使用: **$store.dispatch('对应的action 回调名')** 触发

* + - 1. 可以包含异步代码（定时器, ajax 等等）
      2. 示例代码：

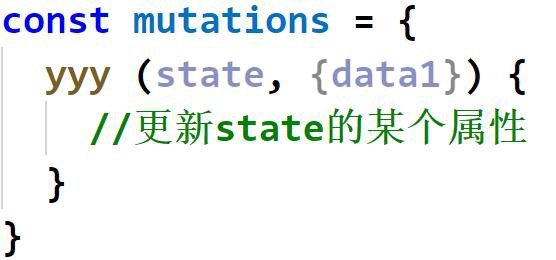


## mutations

* + - 1. 值是一个对象，包含多个直接更新state 的方法
      2. 谁能调用mutations 中的方法？如何调用？

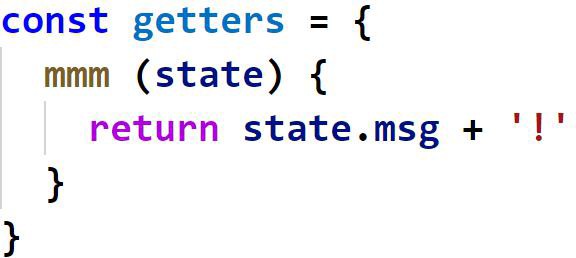
在action 中使用：**commit('对应的 mutations 方法名')** 触发

* + - 1. mutations 中方法的特点：不能写异步代码、只能单纯的操作state
      2. 示例代码：



## getters

* + - 1. 值为一个对象，包含多个用于返回数据的函数
      2. 如何使用？—— **$store.getters.xxx**
      3. 示例代码：



## modules

* + - 1. 包含多个module
      2. 一个module 是一个store 的配置对象
      3. 与一个组件（包含有共享数据）对应

# 第 6 章：vue-router

## 相关理解

### vue-router 的理解

vue 的一个插件库，专门用来实现SPA 应用

### 对SPA 应用的理解

* + - 1. 单页Web 应用（single page web application，SPA）。
      2. 整个应用只有**一个完整的页面**。
      3. 点击页面中的导航链接**不会刷新**页面，只会做页面的**局部更新。**
      4. 数据需要通过ajax 请求获取。

### 路由的理解

##### 什么是路由?

* + - * 1. 一个路由就是一组映射关系（key - value）
        2. key 为路径, value 可能是function 或component

##### 路由分类

* + - * 1. 后端路由：

理解：value 是function, 用于处理客户端提交的请求。

工作过程：服务器接收到一个请求时, 根据**请求路径**找到匹配的**函数**来处理请求, 返回响应数据。

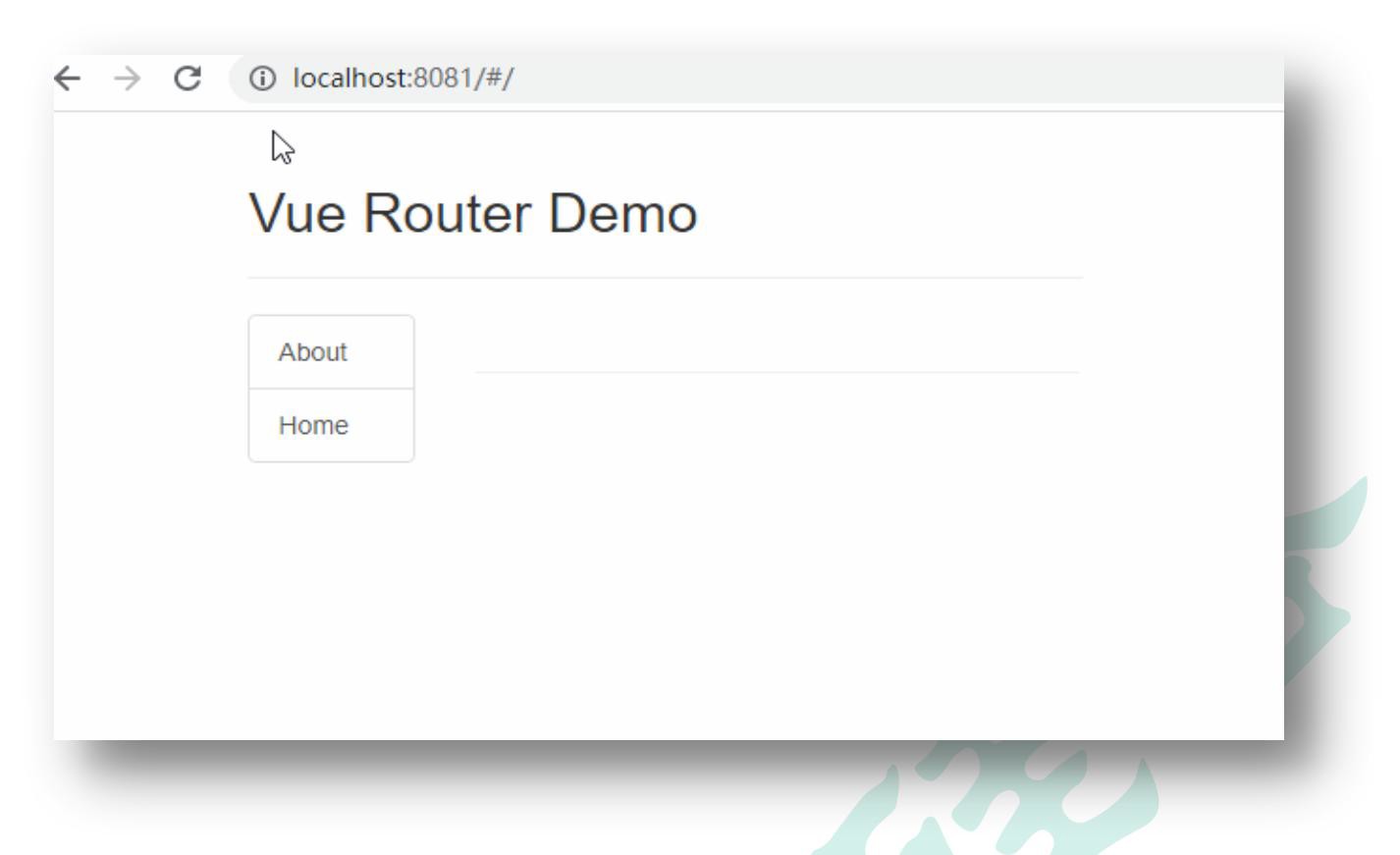
* + - * 1. 前端路由：

理解：value 是component，用于展示页面内容。

工作过程：当浏览器的路径改变时, 对应的组件就会显示。

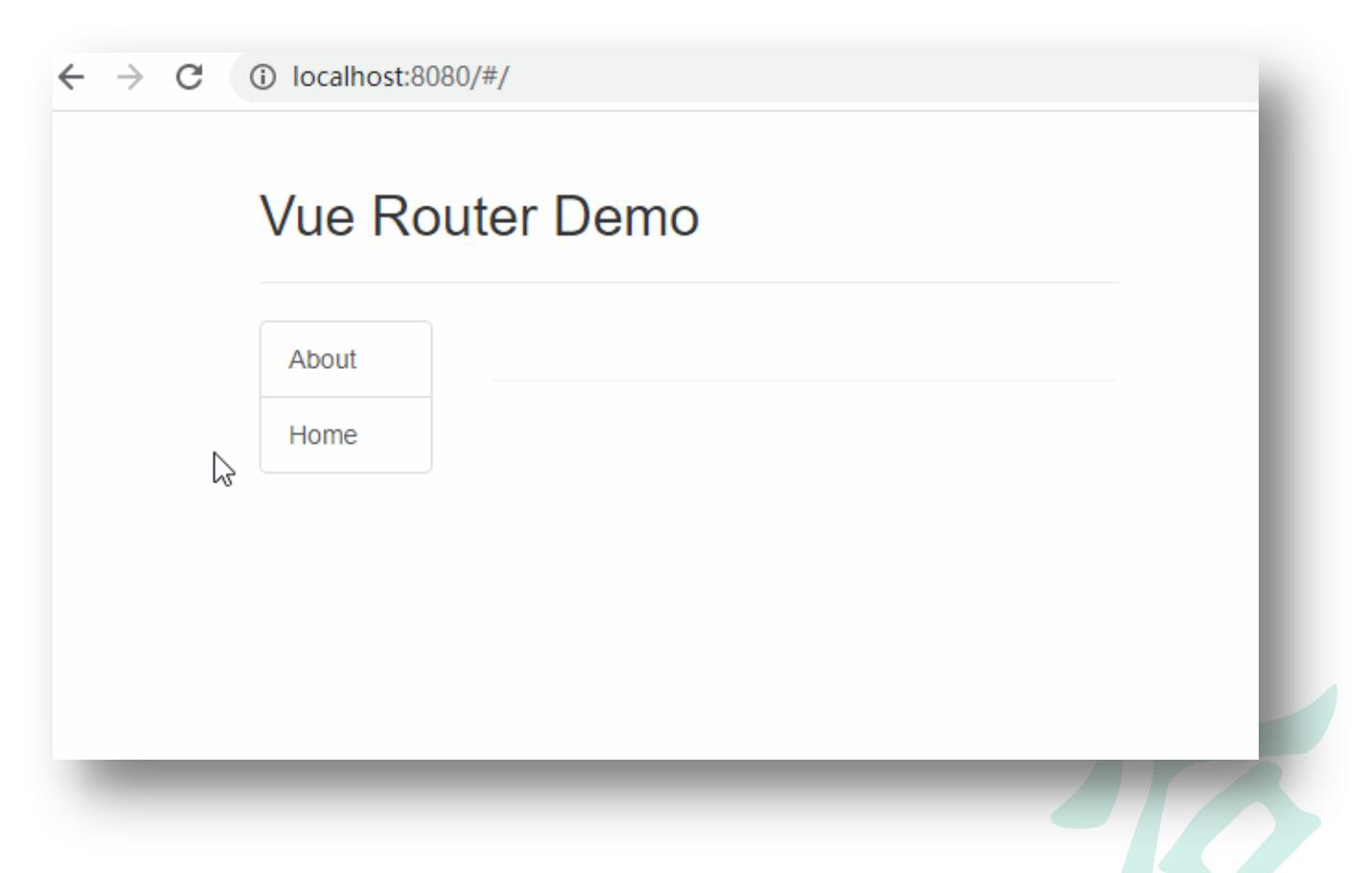
## 基本路由

### 效果

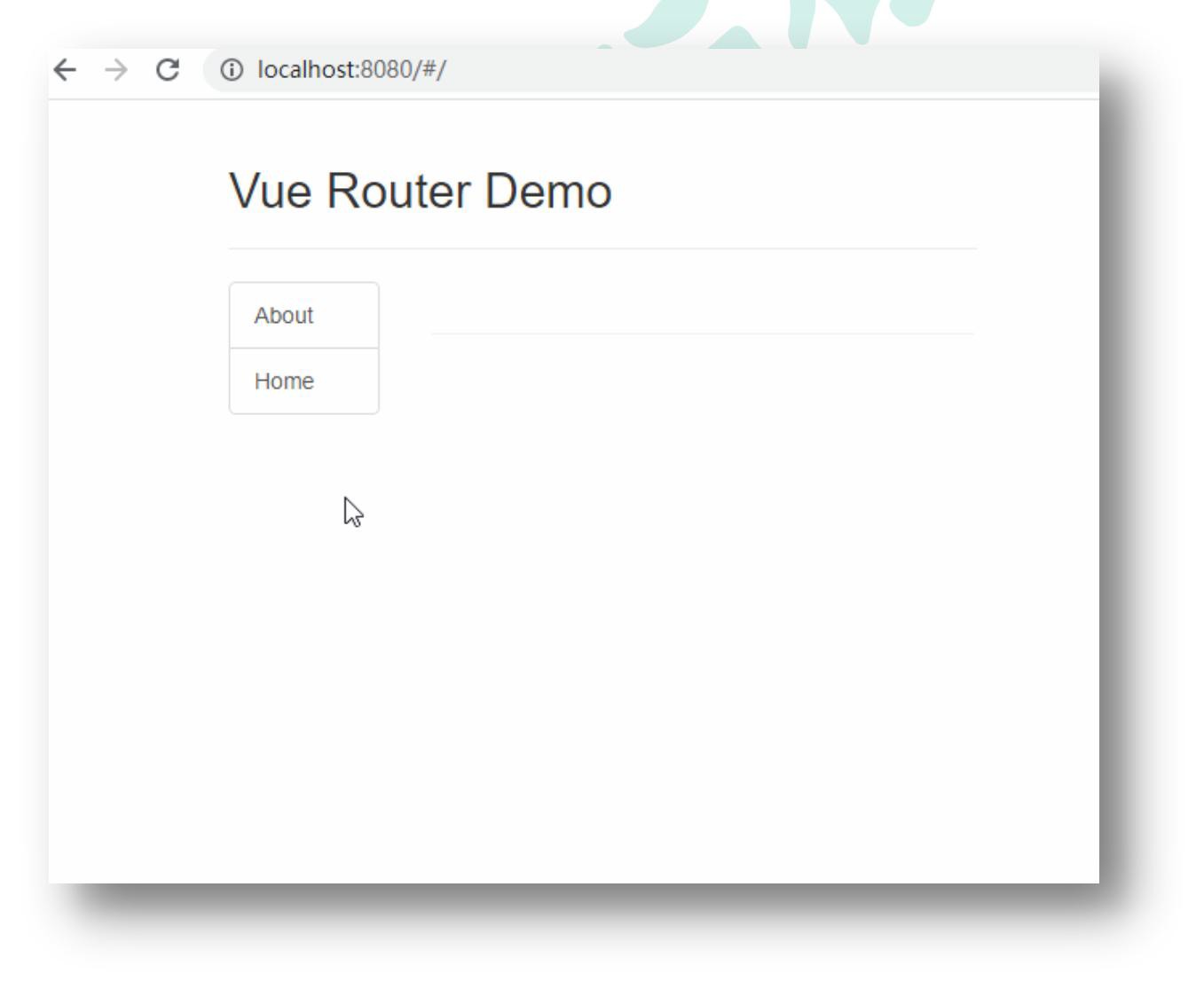


* + 1. **总结: 编写使用路由的 3 步**

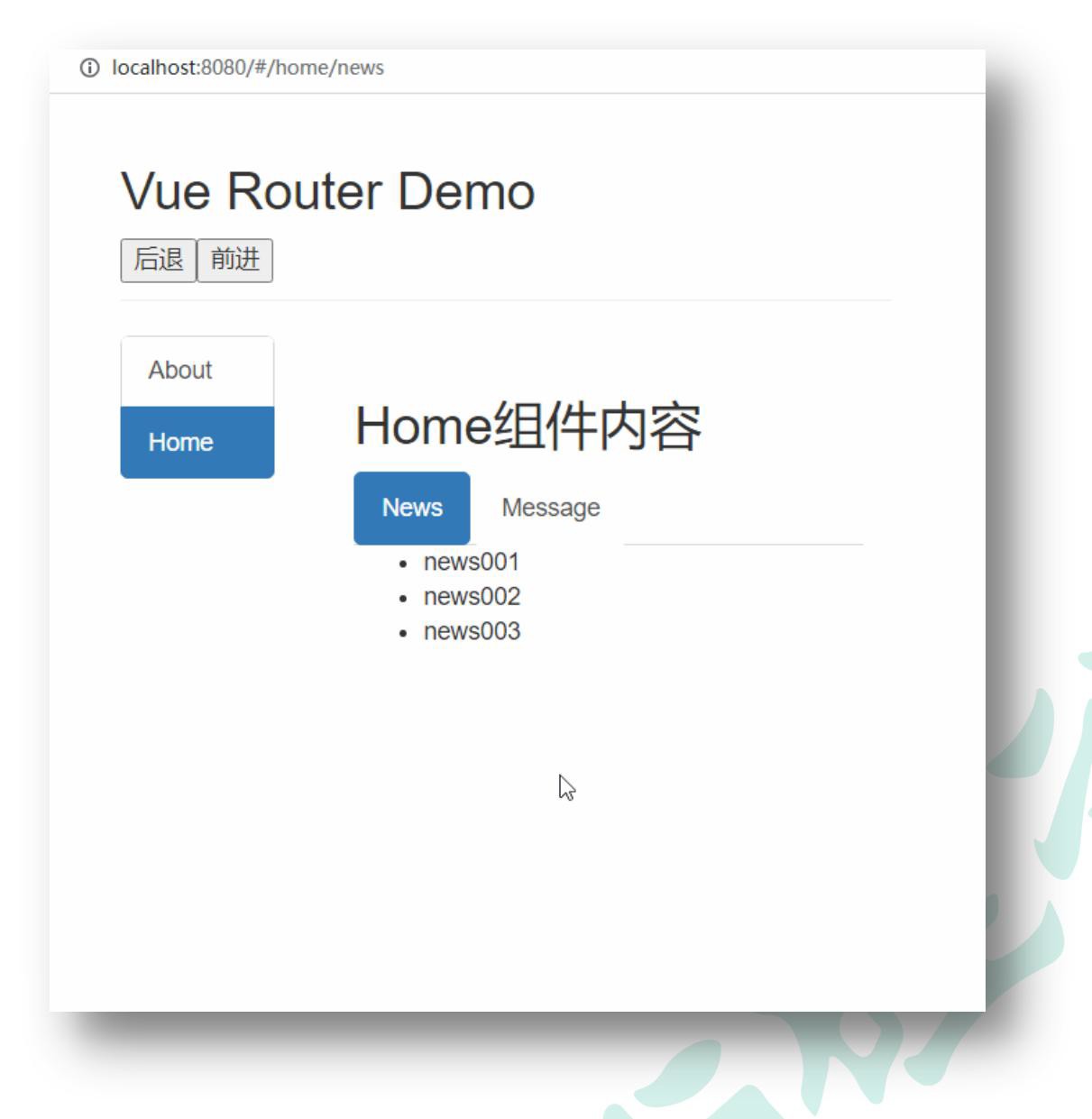
1. 定义路由组件
2. 注册路由
3. 使用路由
   1. **嵌套（多级）路由**



* 1. **路由传参**



* 1. **编程式路由导航**



#### 相关 API：

1. this.$router.push(path): 相当于点击路由链接(可以返回到当前路由界面)
2. this.$router.replace(path): 用新路由替换当前路由(不可以返回到当前路由界面)
3. this.$router.back(): 请求(返回)上一个记录路由
4. this.$router.go(-1): 请求(返回)上一个记录路由
5. this.$router.go(1): 请求下一个记录路由

# 第 7 章：Vue UI 组件库

### 移动端常用UI 组件库

1. Vant [https://youzan.github.io/vant](https://youzan.github.io/vant/)
2. Cube UI <https://didi.github.io/cube-ui>
3. Mint UI [http://mint-ui.github.io](http://mint-ui.github.io/)

### PC 端常用UI 组件库

1. Element UI [https://element.eleme.cn](https://element.eleme.cn/)
2. IView UI [https://www.iviewui.com](https://www.iviewui.com/)