# 一．项目初始化

## 1.安装vue-cli

目录

**[2. 初始化项目](#_Toc30286_WPSOffice_Level1)** **[2](#_Toc30286_WPSOffice_Level1)**

**[3. 进入项目](#_Toc4461_WPSOffice_Level1)** **[2](#_Toc4461_WPSOffice_Level1)**

**[4. 安装依赖](#_Toc1155_WPSOffice_Level1)** **[2](#_Toc1155_WPSOffice_Level1)**

**[5. 启动项目](#_Toc10464_WPSOffice_Level1)** **[2](#_Toc10464_WPSOffice_Level1)**

**[二． 项目目录结构](#_Toc9755_WPSOffice_Level1)** **[2](#_Toc9755_WPSOffice_Level1)**

**[三．模板语法：{{x+y}}](#_Toc1506_WPSOffice_Level1)** **[2](#_Toc1506_WPSOffice_Level1)**

[一个组件包含三个部分：](#_Toc4461_WPSOffice_Level2) [2](#_Toc4461_WPSOffice_Level2)

**[四．Vue基本指令：](#_Toc23766_WPSOffice_Level1)** **[3](#_Toc23766_WPSOffice_Level1)**

[v-html和模板语法的区别](#_Toc1155_WPSOffice_Level2) [3](#_Toc1155_WPSOffice_Level2)

[v-text和v-html的区别:渲染文本](#_Toc10464_WPSOffice_Level2) [3](#_Toc10464_WPSOffice_Level2)

[v-bind：绑定属性](#_Toc9755_WPSOffice_Level2) [4](#_Toc9755_WPSOffice_Level2)

[条件渲染](#_Toc1506_WPSOffice_Level2) [4](#_Toc1506_WPSOffice_Level2)

[列表渲染：](#_Toc23766_WPSOffice_Level2) [5](#_Toc23766_WPSOffice_Level2)

[事件监听：](#_Toc16141_WPSOffice_Level2) [5](#_Toc16141_WPSOffice_Level2)

[事件修饰符：](#_Toc17762_WPSOffice_Level2) [6](#_Toc17762_WPSOffice_Level2)

[数组更新检测:](#_Toc17644_WPSOffice_Level2) [6](#_Toc17644_WPSOffice_Level2)

**[五． 计算属性：](#_Toc16141_WPSOffice_Level1)** **[7](#_Toc16141_WPSOffice_Level1)**

[1.利用单行语句处理字符串](#_Toc609_WPSOffice_Level2) [7](#_Toc609_WPSOffice_Level2)

[2.利用计算属性处理](#_Toc31414_WPSOffice_Level2) [7](#_Toc31414_WPSOffice_Level2)

[3. 利用函数方法处理](#_Toc4303_WPSOffice_Level2) [7](#_Toc4303_WPSOffice_Level2)

[计算属性和函数的区别](#_Toc23977_WPSOffice_Level2) [8](#_Toc23977_WPSOffice_Level2)

**[六． 表单输入绑定](#_Toc17762_WPSOffice_Level1)** **[8](#_Toc17762_WPSOffice_Level1)**

[V-model 双向数据绑定](#_Toc25688_WPSOffice_Level2) [8](#_Toc25688_WPSOffice_Level2)

[Watch实时监测数据](#_Toc30856_WPSOffice_Level2) [8](#_Toc30856_WPSOffice_Level2)

npm install -g vue-cli

## 初始化项目

vue init webpack my-project

## 进入项目

cd my-project

## 安装依赖

npm install

## 启动项目

npm run dev

# 项目目录结构

# 三．模板语法：{{x+y}}

## 一个组件包含三个部分：

视图，template

逻辑，script

样式, style

花括号中只能写单行语句

{{“你好”}}是对的

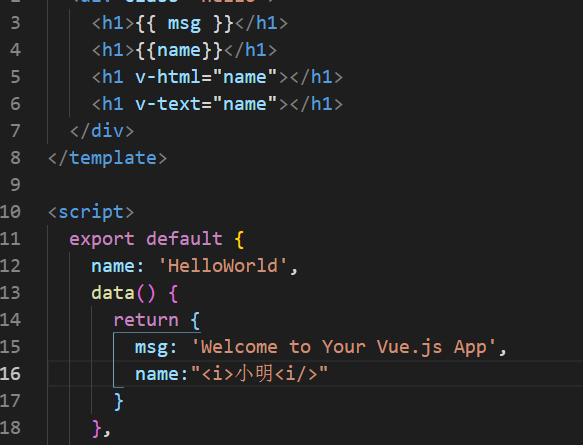
{{name = “小名”}}就是错的

{{1+3}}对的

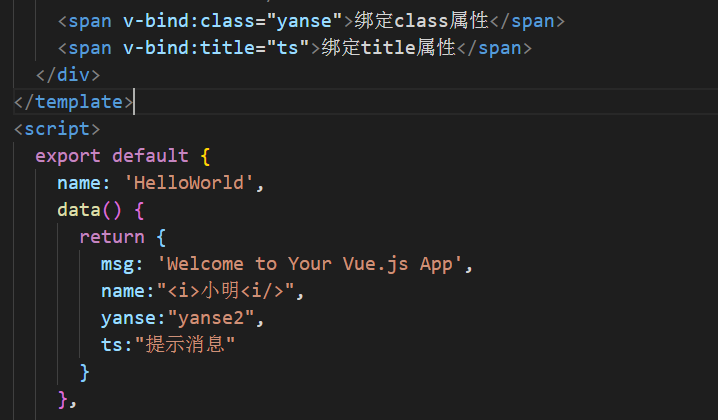
# 四．Vue基本指令：

## v-html和模板语法的区别

## v-text和v-html的区别:渲染文本



## v-bind：绑定属性



## 条件渲染

V-if

V-else



V-show

## 列表渲染：

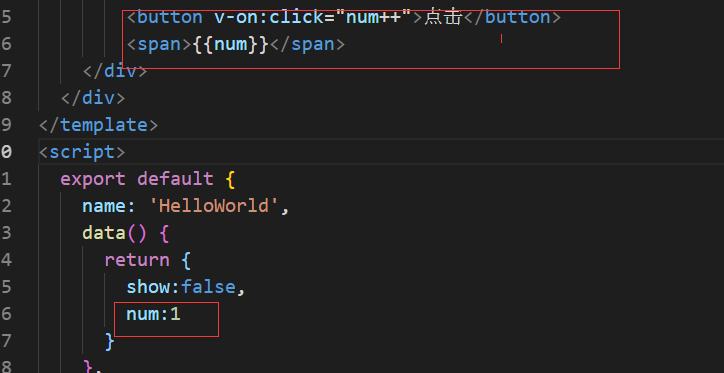
V-for

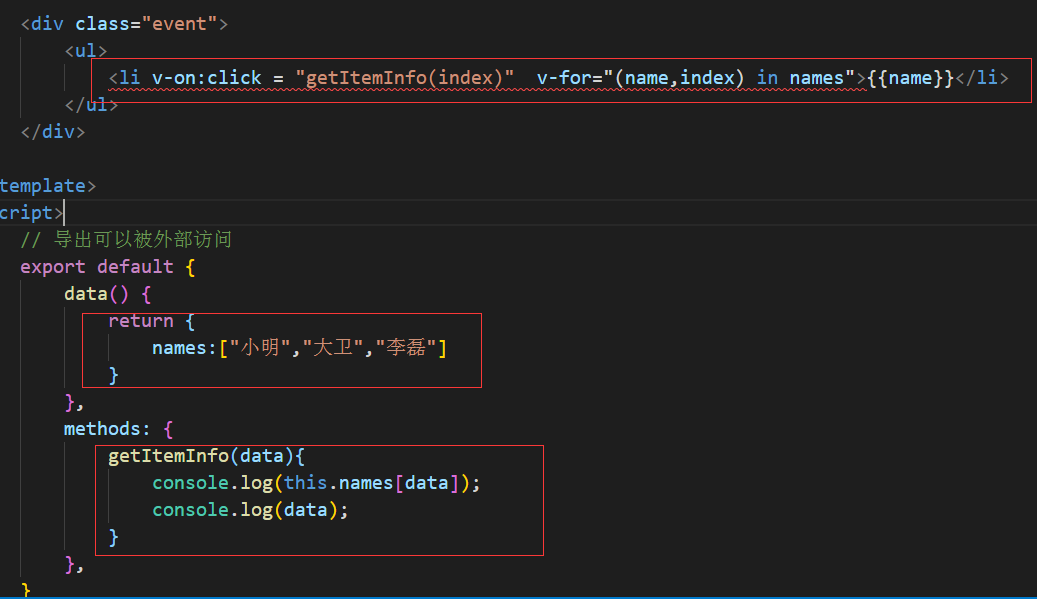
每个列表都需添加key

## 事件监听：

V-on: （简写@click=）

Methods:



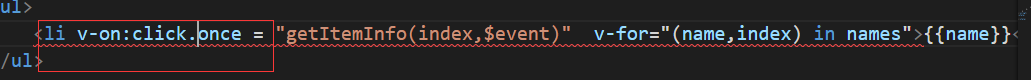


## 事件修饰符：

Stop 阻止事件冒泡

Prevent 阻止默认事件

Once 只执行一次







## 数组更新检测:

变异方法：引起视图更新（在原数组中添加）

<button v-on:click="pushName">添加人数</button>

pushName(){

this.names.push("大王");

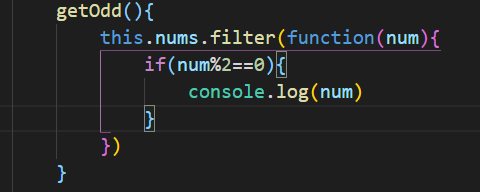
}

替换数组：不会不会引起视图更新（出现了新的数组）

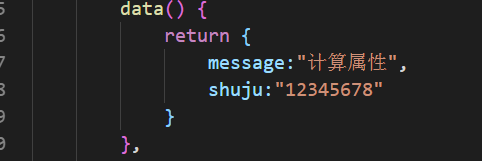
this.names.concat(["大牛"]);

显示过滤（获取偶数）

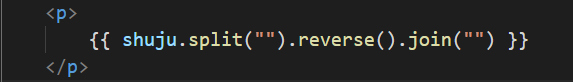




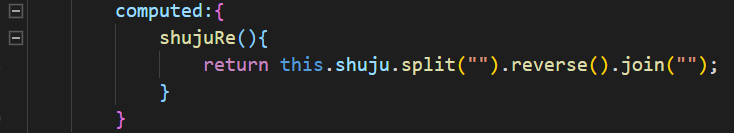
# 计算属性：



## 1.利用单行语句处理字符串

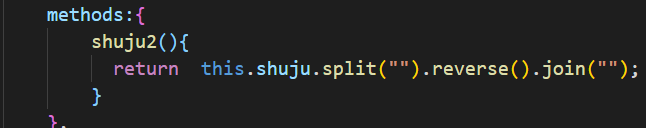


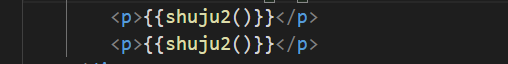
## 2.利用计算属性处理





## 利用函数方法处理





## 计算属性和函数的区别

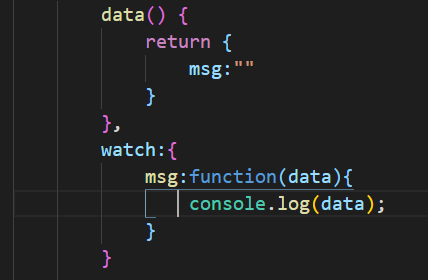
1. 计算属性是基于他们的依赖进行缓存的（除非相关属性发生改变才从新计算）
2. 方法函数每次都要进行计算

# 表单输入绑定

## V-model 双向数据绑定

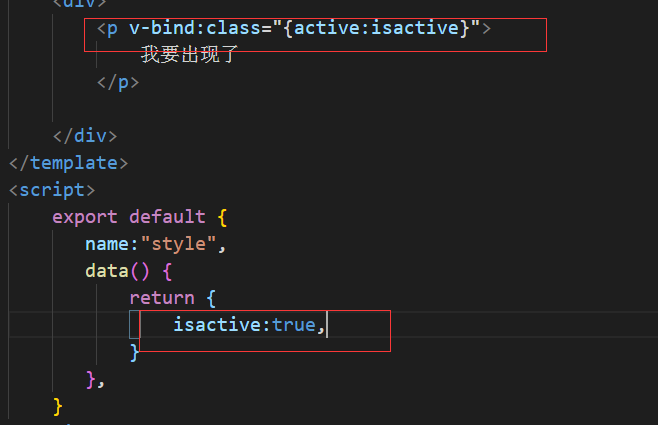


## Watch实时监测数据

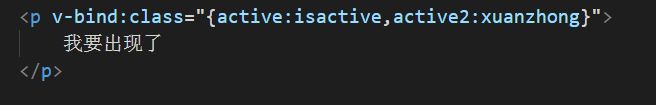


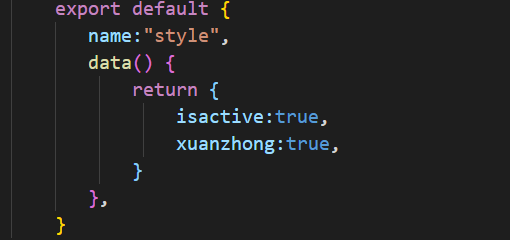
# [Class 与 Style 绑定](https://cn.vuejs.org/v2/guide/class-and-style.html)

## 绑定一个class



## 绑定两个lcass





## 对象语法把刚才的写在data中去



# 单文件组件

## 三个组成部分

1. Template

只能存在一个根元素

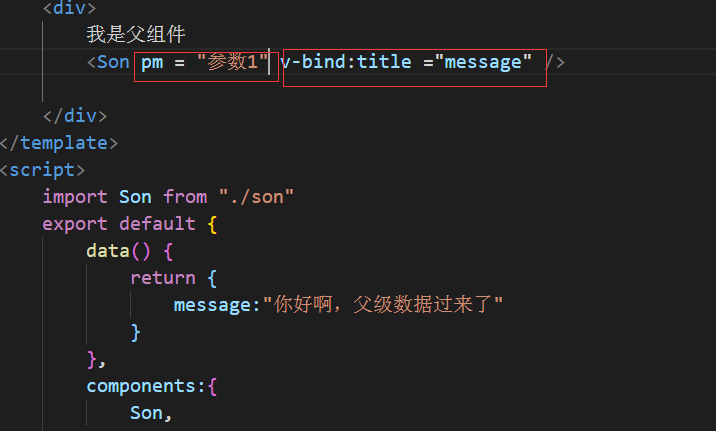
1. Script
2. Style scoped:样式只在当前组件内生效

## 子父级组件交互（通信）/组件组合

父 -> 子 ： props

子 -> 父 ： emit Event

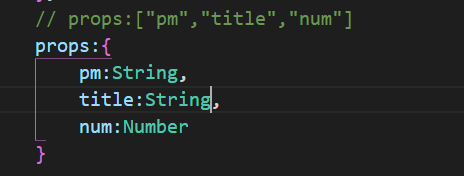
父级组件里面的内容



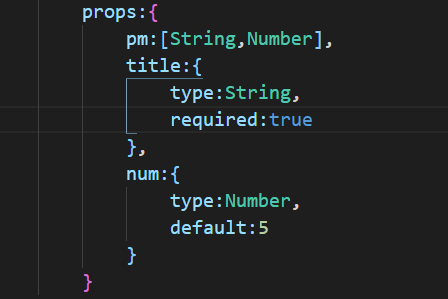
儿子组件里面的内容



## 数据传递类型限制



更多限制

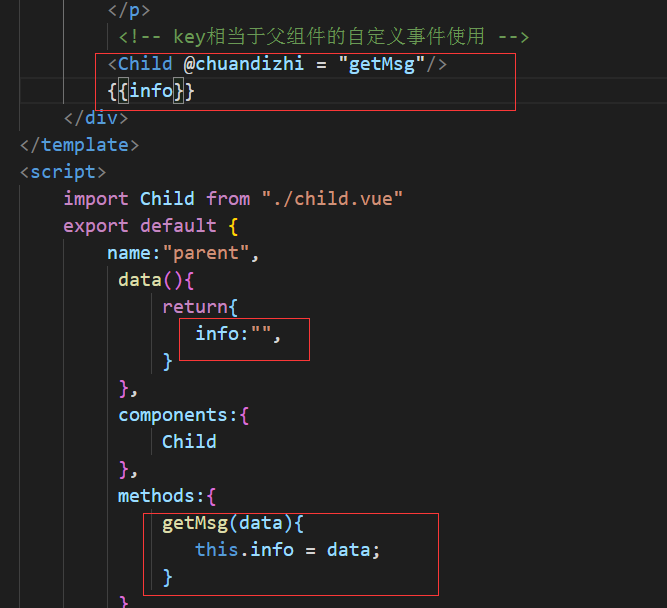


传递对象类型的



子组件向父组件传递数据 用 this.$emit(key，value)





# 插槽

**插槽可以用来相互传递东西，相当于昨天学 props 和 emit**

## 单个插槽

<slot></slot>

## 具名插槽

在子元素中

<slot name="s1"></slot>

<slot name="s2"></slot>

<slot name="s3"></slot>

在父元素中

<p slot="s1">

{{message}}

</p>

<p slot="s2">

哈哈哈 没想到吧 又传了一个

</p>

<div slot="s3">

<p>基础上结合参加爱好成绩爱好</p>

<p>曾经你撒娇刹那间刹那刹那间才能进啊</p>

</div>

## 作用域插槽

## 子-父数据传递

注意：在2.5.0之前，必须使用到template身上

Child.vue中

<slot name="s4" text="我是儿子传递过来的数据" :title="message"></slot>

Parent.vue中

<div slot="s4" slot-scope="shuju">

{{shuju.text}}

<p>

{{shuju.title}}

</p>

</div>

# 动态组件(keep-alive)



Keep-alive

<keep-alive>

<component :is="currentView"></component>

</keep-alive>

## 什么情况下使用缓存： 存取的东西一直不需要更新的，并且可以切换，要是有时间变化，有接口调用的就不能缓存