# 第一章 介绍

## 搭建vue环境

1. 下载vue.js

在浏览器输入“https://cdn.jsdelivr.net/npm/vue/dist/vue.js”，右键菜单[另存为],保存文件

1. 新建一个html文件，名称为“01搭建vue环境.html”，在文件中输入如下内容



1. 在浏览器中预览该文件

## tooltip

利用html的tooltip标记<span>. 不过span的title使用vue的变量

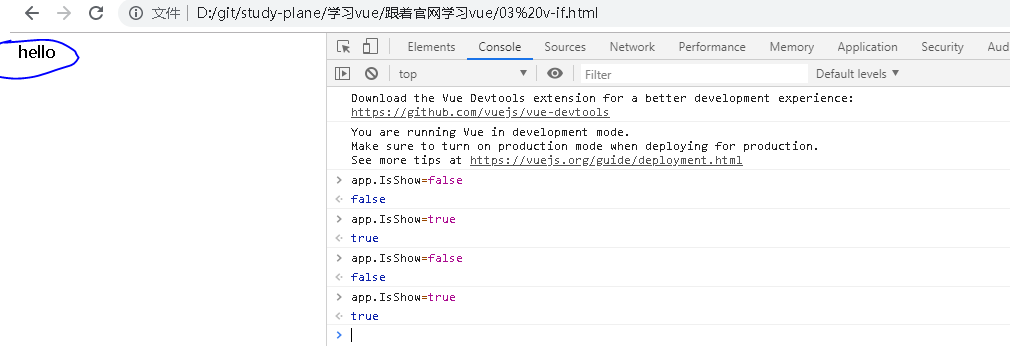


## V-if

1. 代码如下



1. 在浏览器，打开03 v-if.html文件，再按键【F12】打开开发者窗口，然后在console窗口输入：app.IsShow=false，app.IsShow=true。可以看到页面的hello消失和显示出来





1. 完

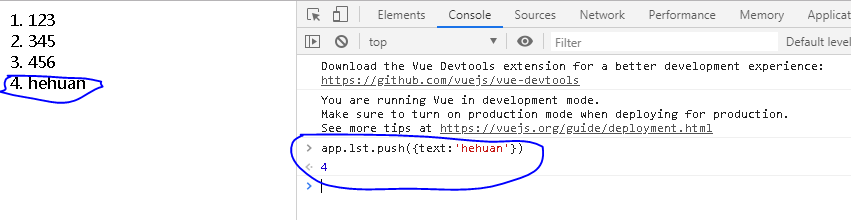
## V-for

1. 代码如下



1. 在控制台输入

app.lst.push({text:’hehuan’})

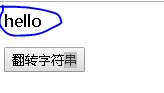


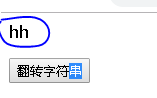
## V-on

1. 代码如下



1. 点击按钮前后，页面的变化





## V-model（在input中，直接使用v-model与变量关联）



## 组件

1. 在v-for的例子上修改



1. 显示出2个列表。着个列表使用了相同的组件，只是输入了不同的参数。达到代码重用的目标
2. 完

# 第二章 实例

这章讲了vue的new方法的参数。这个参数是1个，就是一个javascript对象。这个对象需要包含下面几个成员

1. 关联的html对象的名称

A: 字段名：el

B：字段值： #+元素名称

1. 数据

可以是多个数据，名称：值

1. 方法
2. 钩子

注意：

1. 只有在new方法定义的数据，才能够跟html中的双大括号关联。在new之后增加的data成员，仅仅是javascript的成员。不具有view的效果
2. 通过Object.freeze()释放了的对象，不再能够追踪变化，反应到html中
3. Vue 实例还暴露了一些有用的实例属性与方法。它们都有前缀 $，以便与用户定义的属性区分开来，例如：$data，$el, $watch等

生命周期与钩子

beforCreate

Created

beforMount

Mounted

beforUpdate

Updated

beforDestroy

destroyed

# 第3章 模板语法

1. 插值与 **[v-once](https://cn.vuejs.org/v2/api/" \l "v-once)**

**<p v-once=true>{{ message }}</p>**

**修改message的值，就不会发生变化**

****

1. v-html

在属性中说明是v-html，然后再使用就不会看成纯文本，而是看成html

<p v-html="message">{{message }}</p>

message: '<span title="我司123">点我显示tooltip</span>'



1. Mustache 语法不能作用在 HTML 特性上，遇到这种情况应该使用 **[v-bind 指令](https://cn.vuejs.org/v2/api/" \l "v-bind)**

**如果变未定义（**undefined**）为空（null），为false（false）。属性就被忽略了。**

1. 用 JavaScript 表达式

{{ number + 1 }}

{{ ok ? 'YES' : 'NO' }}

{{ message.split('').reverse().join('') }}

<div v-bind:id="'list-' + id"></div>

每个绑定都只能包含****单个表达式，不能是****

<!-- 这是语句，不是表达式 --> {{ var a = 1 }} <!-- 流控制也不会生效，请使用三元表达式 --> {{ if (ok) { return message } }}

1. 指令

v-if，

1. 参数

v-bind， v-on

1. 动态参数

例如：<a v-on:[eventName]="doSomething"> ... </a>， 这样程序员就可以在data中使用eventName表达要监控的动作（click， doubleClick，chenge等）

主要是v-bind，v-on2个指令使用动态参数

1. 对动态参数的值的约束

动态参数预期会求出一个字符串，异常情况下值为 null。这个特殊的 null 值可以被显性地用于移除绑定。任何其它非字符串类型的值都将会触发一个警告

1. 对动态参数表达式的约束

动态参数表达式有一些语法约束，因为某些字符，例如空格和引号，放在 HTML 特性名里是无效的。同样，在 DOM 中使用模板时你需要回避大写键名

1. 修饰符（没有看懂）
2. 缩写

v-bind:title=”xxx” :title=”xxx”

V-on:click=”xxx” @click=”xxx”

1. 完
2. 计算属性和侦听器
3. 例子



computed: 注意全部是小写，否则会有问题

{

ReverseMessage:function(){

return this.message.split('').reverse().join('')

}

}

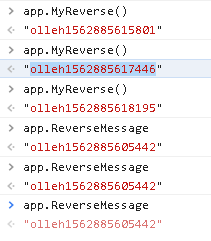
1. 计算属性与方法的比较
2. 例子



1. 在控制台操作

可以看到method多次执行后，每次返回的值是不一样的，以为这该段代码被多次调用，每次调用代码中的Date.now()都返回不同的值。

也可以看到计算属性ReverseMessage被多次调用后，值都不变。这是因为计算属性没有依赖被改变，所以计算属性的代码不会被执行，每次调用仅仅是直接返回了计算属性的缓存值。



1. 计算属性与侦听属性的比较
2. 例子



1. 在控制台，输入app.test="kk1",去修改test的值，可以看到watch函数被执行了，message的值变化了
2. 13 计算属性的seter\_geter
3. 例子



1. 在控制台，输入app.ReverseMessage=”345”

可以看到set函数打印的日志，因为我们在set函数中改变了message的值，导致get函数被重新调用，打印出get函数的日志；

1. 输入xxx=app.ReverseMessage，并不能导致get函数被调用，因为get依赖的参数没有被修改，所以该函数的值被缓存了。
2. 侦听器

这章完全没有学懂，例子做出来也有错误



1. 完
2. CSS和Style绑定
3. 绑定 HTML Class

这个例子可以成功，但是active在什么地方用？我查看百度， active是给链接在激活时设置属性用的。所以这里绑定虽然成功，但是现实不出来效果

1. 完
2. 条件渲染
3. v-if， v-else， v-else-if

在并且的“相同的html标签”之间做

1. Key

使得各个条件之间的组件，可以独立存在

1. V-for

当 v-if 与 v-for 一起使用时，v-for 具有比 v-if 更高的优先级。V-for最好不要与v-if一起使用。而是将v-if用到子元素上，或将v-if上升到一个父元素上。

1. 例子



1. 列表渲染
2. 数组
3. 、v-for="item in items"
4. v-for="(item, index) in items"
5. v-for="item of items"
6. 对象的属性
7. v-for="value in object"遍历一个对象的所有属性
8. v-for="(value, name) in object"
9. v-for="(value, name, index) in object"
10. 数组中元素顺序改变后，通过v-bind:key保证顺序也重新渲染

v-for="item in items" v-bind:key="item.id"

1. 事件处理
2. V-on
3. 直接跟表达式：v-on:click="counter += 1"
4. 跟函数，函数的参数省略（就是event）：v-on:click="greet"
5. 跟函数，函数的参数是自定义的：v-on:click="say('hi')"
6. 跟函数，函数有自定义参数，也有系统参数event

v-on:click="warn('Form cannot be submitted yet.', $event)"

1. 函数后跟修饰

<!-- 阻止单击事件继续传播 -->

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

1. 完

1. 完