# 一、Vue

## 1.定义

Vue 是一套用于构建用户界面的**渐进式框架**。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与[**现代化的工具链**](https://cn.vuejs.org/v2/guide/single-file-components.html)以及各种[**支持类库**](https://github.com/vuejs/awesome-vue#libraries--plugins)结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。（渐进式、MVVM、双向绑定、虚拟DOM）

## 2.渐进式

Vue从设计角度来讲，能够涵盖这张图上所有的东西，但是彼此独立，不必全部使用

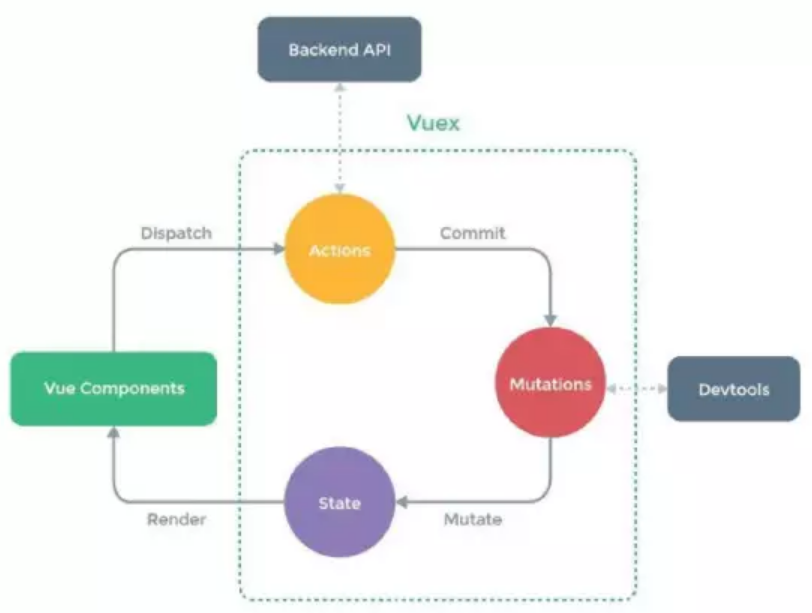


**声明式渲染：**所有的逻辑尽可能在状态的层面去进行，当状态改变的时候，View应该是在框架帮助下自动更新到合理的状态，而不需要手动设置更新。

**组件系统：**针对不同的功能，提出公共的部分做成组件，把UI结构映射到组件树。

**客户端路由：**实现多个URL之间的跳转， URL和组件树的状态之间有一个映射关系，客户端路由的职责就是让这个映射关系声明式地对应起来

**状态管理：**通过Vuex实现对组件状态的管理

****

**构建工具：**脚手架vue-cli。有多个可选模板，有简单的也有复杂的。极简的配置，更快的安装，可以更快的上手。它也有一个更完整的模板，包括单元测试在内的各种内容都涵盖

## 3.优势

**轻量级框架**

只关注视图层,是一个构建数据的视图集合,大小只有几十kb

Vue.js通过简洁的API提供高效的数据绑定和灵活的组件系统

**双向数据绑定**

也就是所谓的响应式数据绑定。指vue.js会自动对页面中某些数据的变化做出同步的响应。

也就是说，vue.js会自动响应数据的变化情况，并且根据用户在代码中预先写好的绑定关系，对所有绑定在一起的数据和视图内容都进行修改。而这种绑定关系，就是以input 标签的v-model属性来声明的，因此你在别的地方可能也会看到有人粗略的称vue.js为声明式渲染的模版引擎。

这也就是vue.js最大的优点，通过MVVM思想实现数据的双向绑定，让开发者不用再操作dom对象，有更多的时间去思考业务逻辑。

**组件化**

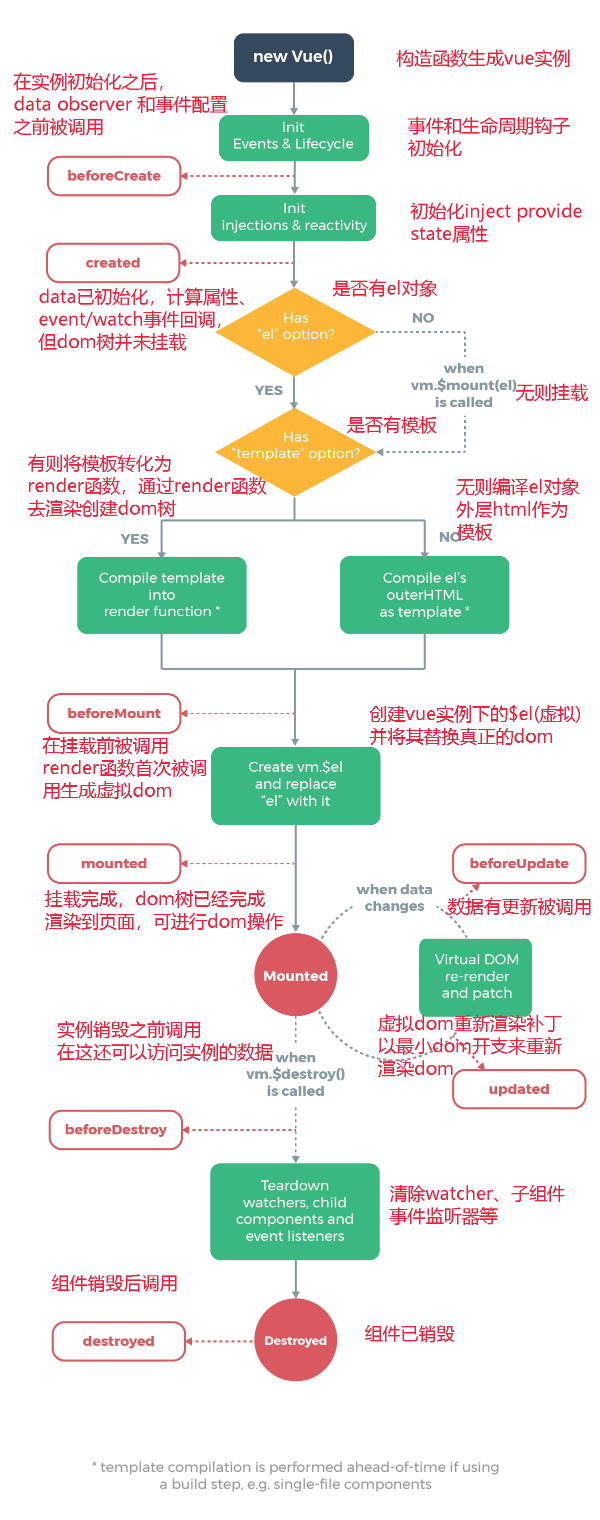
Vue.js通过组件，把一个单页应用中的各种模块拆分到一个一个单独的组件（component）中，我们只要先在父级应用中写好各种组件标签（占坑），并且在组件标签中写好要传入组件的参数（就像给函数传入参数一样，这个参数叫做组件的属性），然后再分别写好各种组件的实现（填坑），然后整个应用就算做完了。

**视图,数据,结构分离**

使数据的更改更为简单,不需要进行逻辑代码的修改,只需要操作数据就能完成相关操作

**虚拟DOM**

浏览器本身处理DOM也是有性能瓶颈的，尤其是在传统开发中，用JQuery或者原生的JavaScript DOM操作函数对DOM进行频繁操作的时候，浏览器要不停的渲染新的DOM树，导致页面看起来非常卡顿。

而Virtual DOM可以预先通过JavaScript进行各种计算，把最终的DOM操作计算出来并优化，由于这个DOM操作属于预处理操作，并没有真实的操作DOM，所以叫做虚拟DOM。最后在计算完毕才真正将DOM操作提交，将DOM操作变化反映到DOM树上。

**运行速度更快**

像比较与react而言,同样都是操作虚拟dom,就性能而言,vue存在很大的优势

**简单易学**

国人开发,中文文档,不存在语言障碍,易于理解和学习

## 4.生命周期

Vue 2.0

new Vue后

init Events & Lifecycle 初始化一个空的Vue实例,只有默认的生命周期函数、默认事件

（1）**beforeCreate**

处理data和methods

（2）**created** 需要调用methods中方法，或者操作data中数据，在此执行

判断el，没有就等待el被声明

判断template，没有使用el的外部html作为模板，有就使用

只有模板，暂时没有挂在到真正的页面

（3）**beforeMounted**

将模板挂在到真正页面

（4）**mounted** 需要初始化DOM阶段，在此执行

data中数据改变时

（5）**beforeUpdate**

利用虚拟DOM重新渲染

（6）**updated**

产生销毁操作时

（7）**beforeDestroy**

销毁实例上所有的data\methods\过滤器\指令等

（8）**Destroy**

## 5.版本 1.0与2.0

### （1）生命周期差距：



### （2）template根元素

Vue 1.0 <template>内可以有多个根元素

Vue 2.0 <template>内只能有一个根元素

### （3）for循环遍历

Vue 1.0 Vue 2.0

数组：（index,value）（value,index）

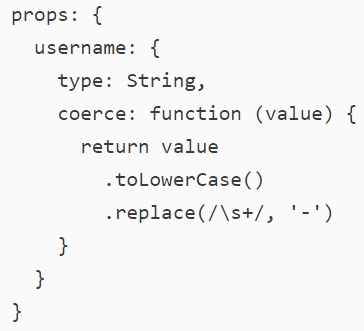
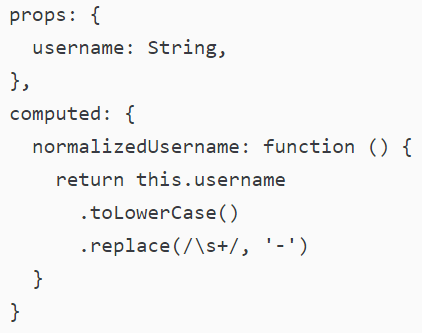
对象：（key,value） （value,key）

v-for的循环范围: v-for='item in 10'

0-9 1-10。

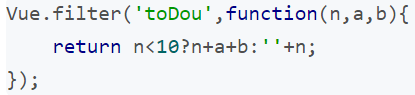
### （4）computed

Vue 2.0中，将props中的操作，放于computed中执行

### （5）过滤器

Vue 2.0需要自定义过滤器



### 5.版本2.0与3.0

### （1）项目结构

3.0移除了config、build、static文件夹。在外层vue.config.js中设置配置文件；

### （2）数据监听

3.0使用Proxy替换Object.defineProperty；

•可直接监听数组类型的数据变化  
•监听的目标为对象本身，不需要像Object.defineProperty一样遍历每个属性，有一定的性能提升  
•可拦截apply、ownKeys、has等13种方法，而Object.defineProperty不行  
•直接实现对象属性的新增/删除

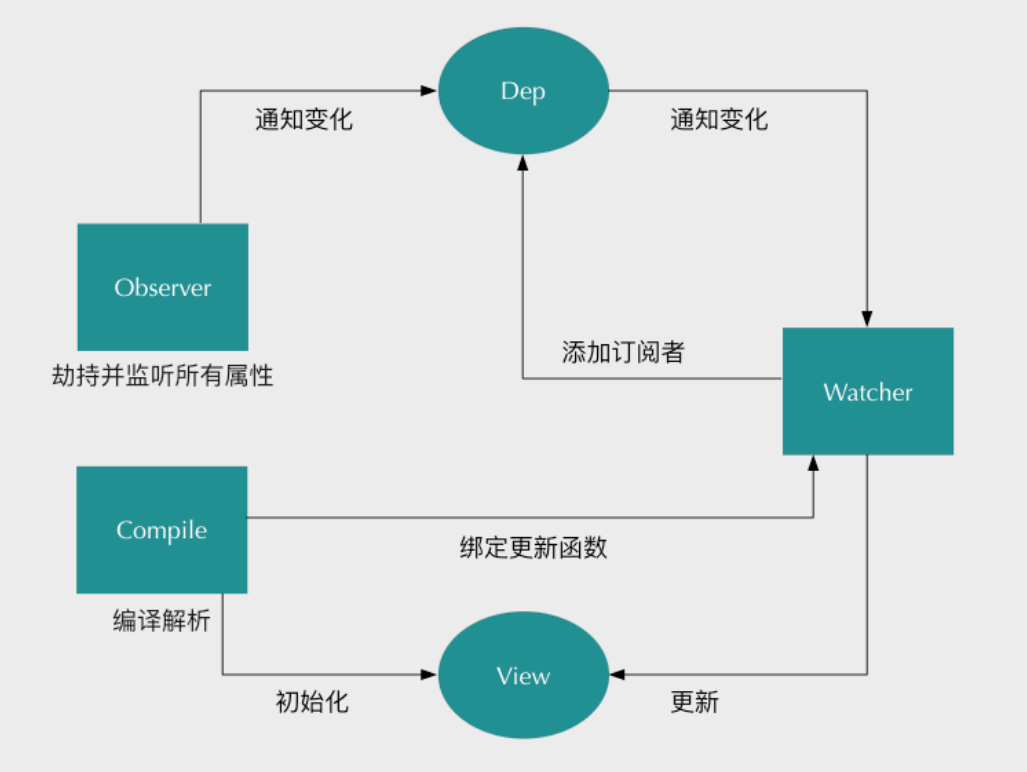
### （3）vdom性能

3.0渲染优化，令vdom操作的粒度变小，减少遍历

* 将模版基于动态节点指令切割为嵌套的区块
* 每个区块内部的节点结构是固定的
* 每个区块只需要以一个 Array 追踪自身包含的动态节点

## 6.双向绑定

### （1）原理



**Observer：**监听器，用来劫持并监听所有属性，如果有变动的，就通知订阅者。

**Watcher：**订阅者，可以收到属性的变化通知并执行相应的函数，从而更新视图。

**Compile：**解析器，可以扫描和解析每个节点的相关指令，并根据初始化模板数据以及初始化相应的订阅器。

**Dep：**消息订阅器，收集订阅者

### （2）实现

#### 重要前置方法：

**Object.defineProperty（obj，props， descriptor）：**//对象 对象的属性 操作设置

1.数据描述符

configurable: 是否可以重新定义描述符

enumerable：是否可以枚举该属性

value：初始值

writable：是否可以修改属性值

2.访问描述符

get：回调函数，根据其他属性，设置得到当前值

set：回调函数，根据当前值，设置其他值

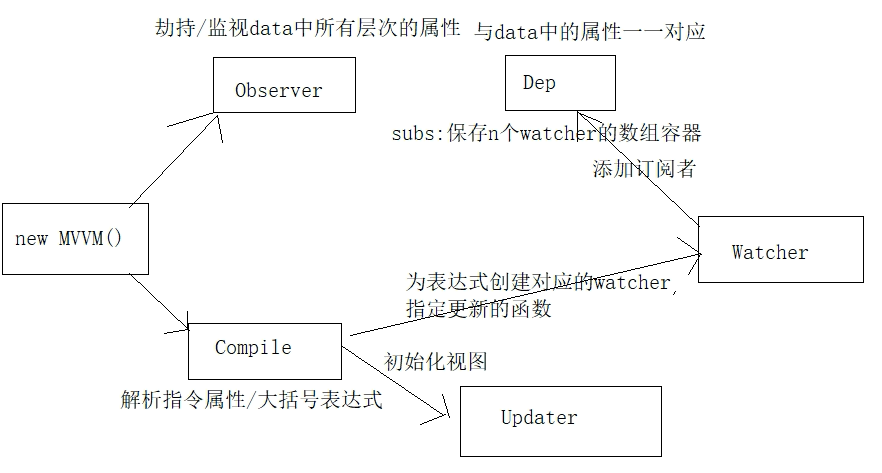
**documentFragment：**

document: 对应显示的页面, 包含n个elment 一旦更新document内部的某个元素界面更新

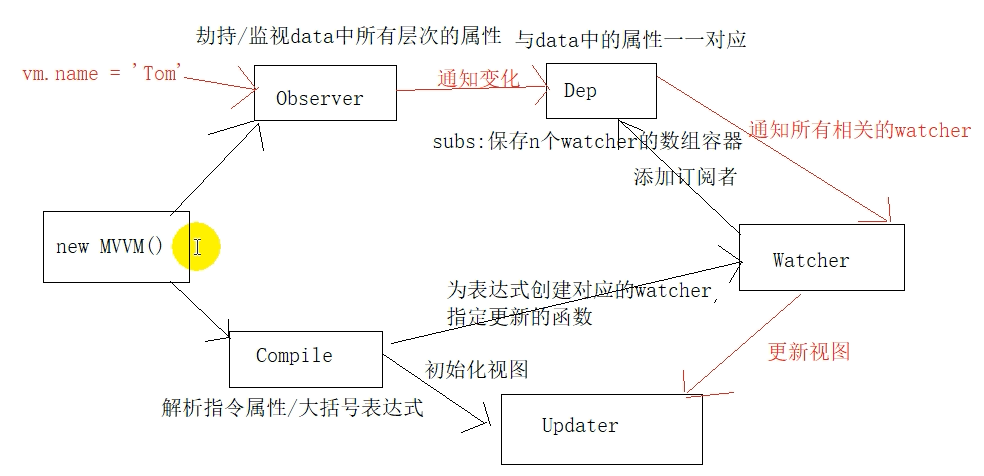
documentFragment: 内存中保存n个element的容器对象(不与界面关联), 如果更新framgnet中的某个element, 界面不变

#### 原理图：

**初始化：**（vm = new Vue({el: "#app, data: {name: 'qzy'})



**数据改变：**(vm.name = 'Tom')

****

**MVVM：**

也就是Vue，调用Observer与Compile实现初始化

**Observer:**

用来对data所有属性数据进行劫持的构造函数

给data中所有属性重新定义属性描述(get/set)

为data中的每个属性创建对应的dep对象

**Dep:**

data中的每个属性(所有层次)都对应一个dep对象

创建的时机:

在初始化define data中各个属性时创建对应的dep对象

在data中的某个属性值被设置为新的对象时

对象的结构

{

id, // 每个dep都有一个唯一的id

subs //包含n个对应watcher的数组(subscribes的简写)

}

subs属性说明

当一个watcher被创建时, 内部会将当前watcher对象添加到对应的dep对象的subs中

当此data属性的值发生改变时, 所有subs中的watcher都会收到更新的通知, 从而最终更新对应的界面

**Compile：**

用来解析模板页面的对象的构造函数(一个实例)

利用compile对象解析模板页面

每解析一个表达式(非事件指令)都会创建一个对应的watcher对象, 并建立watcher与dep的关系

complie与watcher关系: 一对多的关系

**Watcher：**

模板中每个非事件指令或表达式都对应一个watcher对象。如 {{data}}/v-text

监视当前表达式数据的变化

创建的时机: 在初始化编译模板时

对象的组成：

{

vm, //vm对象

exp, //对应指令的表达式

cb, //当表达式所对应的数据发生改变的回调函数

value, //表达式当前的值

depIds //表达式中各级属性所对应的dep对象的集合对象

//属性名为dep的id, 属性值为dep

}

**dep与watcher的关系: 多对多**

一个data中的属性对应对应一个dep, 一个dep中可能包含多个watcher(模板中有几个表达式使用到了属性) {{data}} v-text

模板中一个非事件表达式对应一个watcher, 一个watcher中可能包含多个dep(表达式中包含了几个data属性) a.b.c

## 7.Vue 扩展插件

1)

vue-cli: vue 脚手架

2)

vue-resource(axios): ajax 请求

3)

vue-router: 路由

4)

vuex: 状态管理

5)

vue-lazyload: 图片懒加载

6)

vue-scroller: 页面滑动相关

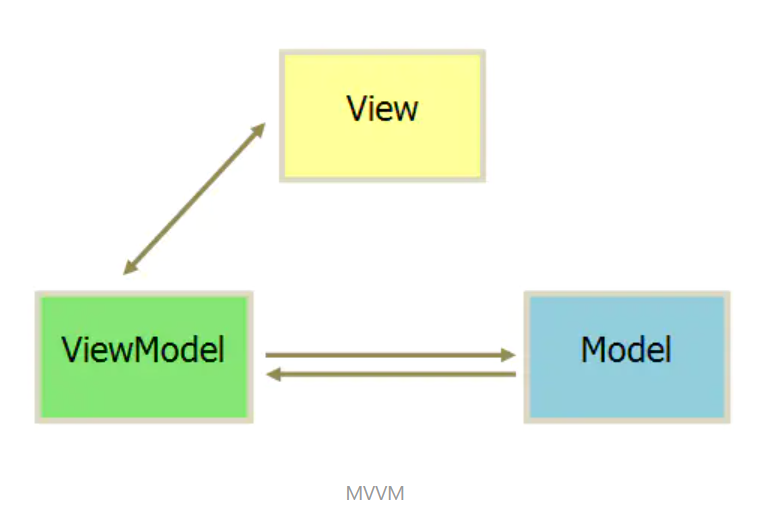
7)

mint-ui: 基于 vue 的 UI 组件库(移动端)

8)

element-ui: 基于 vue 的 UI 组件库(PC 端)

## 8.MVVM



Model: 数据层 （存储数据及对数据的处理如增删改查）

View: 视图层（UI 用户界面）

ViewModel: 业务逻辑层 （一切 js 可视为业务逻辑）负责对View与Model进行感知，并根据其改变做更新操作

## 9.模板语法

Vue.js 使用了基于 HTML 的模板语法，允许开发者声明式地将 DOM 绑定至底层 Vue 实例的数据。所有 Vue.js 的模板都是合法的 HTML，所以能被遵循规范的浏览器和 HTML 解析器解析。

在底层的实现上，Vue 将模板编译成虚拟 DOM 渲染函数。结合响应系统，Vue 能够智能地计算出最少需要重新渲染多少组件，并把 DOM 操作次数减到最少。

### （1）插值

**文本**

使用{{data}}插入动态文本，将其解释为普通文本

使用v-once指令，令data只改变一次（初始化那次）

**HTML**

data中值为html文本：myHtml: '<span>my html</span>'

作为新HTML插入某标签内部：<span v-html="myHtml"></span>

（注意XSS攻击）

**Attribute**

通过v-bind绑定属性名（id/class），内容为真时呈现

<div v-bind:id="dynamicId"></div> dynamicId是data中的数据，可为id名或css样式名称

**JS语法**

支持Math/Date/常见运算符/三元运算符

不支持赋值语句/if流控制

### （2）指令

指令 (Directives) 是带有 v- 前缀的特殊 attribute。指令 attribute 的值预期是**单个 JavaScript 表达式** (v-for 是例外情况，稍后我们再讨论)。指令的职责是，当表达式的值改变时，将其产生的连带影响，响应式地作用于 DOM。

**参数：**

<a v-bind:href="url">...</a>

**动态参数：**

<a v-bind:[attributeName]="url"> ... </a>

attributeName可为‘focus' / 'href' 等

**修饰符：**

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

.prevent 修饰符告诉 v-on 指令对于触发的事件调用 event.preventDefault()

**缩写：**

v-bind:

****

v-on:

****

## 10.计算属性和侦听器

### [计算属性](https://cn.vuejs.org/v2/guide/computed.html#%E8%AE%A1%E7%AE%97%E5%B1%9E%E6%80%A7)computed

减少data中值的计算，集中放入computed中进行计算，作为reversedMessage的getter函数



另外虽然methods也能达到相同目的，但是computed是响应式，只有相关依赖发生改变时才会计算。

### 监听属性watch

Computed: 可以关联多个实时计算的对象，当这些对象中的其中一个改变时都会出发这个属性。具有缓存能力，所以只有当数据再次改变时才会重新渲染，否则就会直接拿取缓存中的数据。

Watch：当你需要在数据变化响应时，执行异步操作，或高性能消耗的操作，自定义 watcher 的方式就会很有帮助。

## 11. Class 与 Style 绑定

字符串拼接麻烦且易错。因此，在将 v-bind 用于 class 和 style 时，Vue.js 做了专门的增强。表达式结果的类型除了字符串之外，还可以是对象或数组。

### （1）[绑定 HTML Class](https://cn.vuejs.org/v2/guide/class-and-style.html#%E7%BB%91%E5%AE%9A-HTML-Class)

#### 对象语法：

对象中的样式都将添加到属性中, 使 class="static active text-danger"

（1）可直接在HTML中添加对象



（2）也可在data中定义对象，再放入HTML中

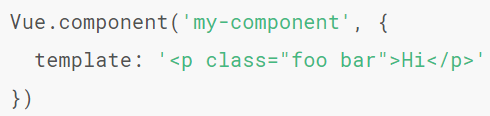


#### 数组语法：

****

#### 组件语法：

组件声明：



使用时也可声明：



最终结果：



### （2）[绑定内联样式](https://cn.vuejs.org/v2/guide/class-and-style.html#%E7%BB%91%E5%AE%9A%E5%86%85%E8%81%94%E6%A0%B7%E5%BC%8F)

#### 对象语法：

可直接在内联样式中定义：



也可在data中定义样式对象：



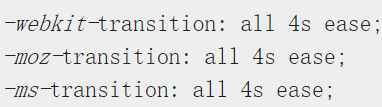
#### 数组语法：

可以将多个样式对象应用到同一个元素上



#### 自动添加前缀：

当 v-bind:style 使用需要添加[浏览器引擎前缀](https://developer.mozilla.org/zh-CN/docs/Glossary/Vendor_Prefix)的 CSS property 时，如 transform，Vue.js 会自动侦测并添加相应的前缀。



#### 多重值：

从 2.3.0 起你可以为 style 绑定中的 property 提供一个包含多个值的数组，常用于提供多个带前缀的值，例如：



这样写只会渲染数组中最后一个被浏览器支持的值。

## 12.条件渲染

### （1）v-if

**v-if**

指令用于条件性地渲染一块内容。这块内容只会在指令的表达式返回 truthy 值的时候被渲染。

想同时控制多个元素，可用template外包后，给其设置v-if

**v-else**

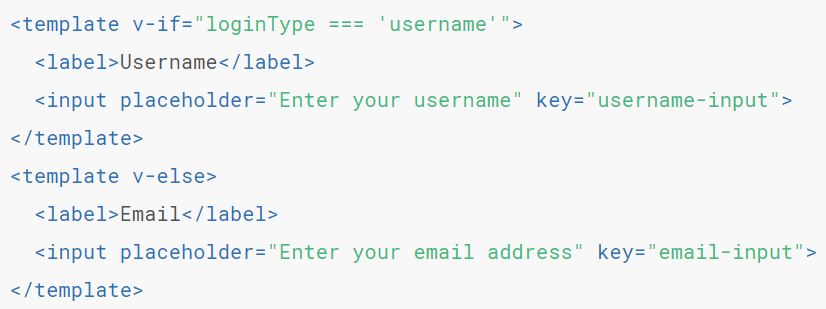
使用 v-else 指令来表示 v-if 的“else 块”，必须紧跟在带 v-if 或者 v-else-if 的元素的后面，否则它将不会被识别。

**v-else-if**

充当 v-if 的“else-if 块”，可以连续使用，也必须紧跟在带 v-if 或者 v-else-if 的元素之后。

**key**

Vue 会尽可能高效地渲染元素，通常会复用已有元素而不是从头开始渲染，因此会导致虽然切换了元素，但是元素本质没有变，已输入的内容也没有变。因此用Key来表达“这两个元素是完全独立的，不要复用它们”。



### （2）v-show

不同的是带有 v-show 的元素始终会被渲染并保留在 DOM 中。v-show 只是简单地切换元素的 CSS property display。

v-show 不支持 <template> 元素，也不支持 v-else。

### （3）[v-if 与 v-show](https://cn.vuejs.org/v2/guide/conditional.html#v-if-vs-v-show)区别

v-if根据取值会对元素进行销毁与重建，渲染条件不变时，什么都不做。切换开销高。

v-show不管初始条件如何，总是进行渲染，进行CSS切换。初始渲染开销高。

## 13.列表渲染

### （1）v-for



也可用of代替in



可循环遍历对象或数组

数组：



对象：



维护状态：

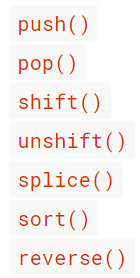
由于Vue会复用元素，只适用于不依赖子组件状态或临时 DOM 状态 。因此用key去标注每个元素

### （2）数组更新检测

**变更方法：**

直接对data中对象的属性值，不会触发视图更新

以下方法会直接改变原数组，经过包裹修改后，会导致触发视图更新



**替换数组：**

****

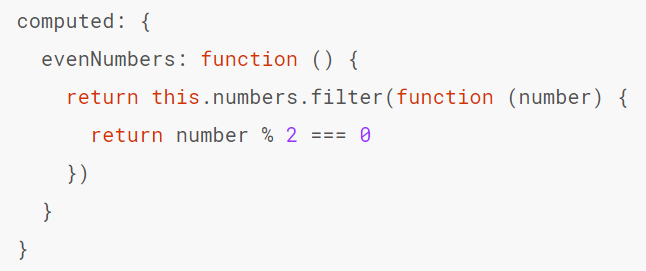
Vue 为了使得 DOM 元素得到最大范围的重用而实现了一些智能的启发式方法，所以用一个含有相同元素的数组去替换原来的数组是非常高效的操作。

**注意事项：**

由于 JavaScript 的限制，Vue 不能检测数组和对象的变化**。**

### （3）显示过滤/排序后的结果

有时，我们想要显示一个数组经过过滤或排序后的版本，而不实际变更或重置原始数据。在这种情况下，可以创建一个计算属性，来返回过滤或排序后的数组。

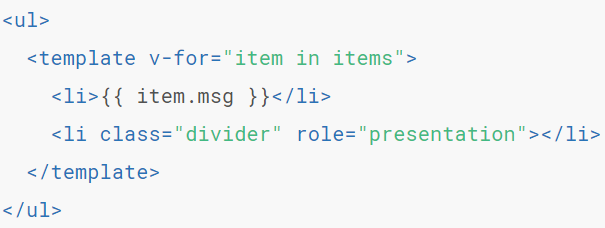


### （4）for使用值范围





### （5）template中使用v-for



### （6）v-for与v-if

当它们处于同一节点，v-for 的优先级比 v-if 更高

## 14.事件处理

### （1）监听事件

可以用 v-on 指令监听 DOM 事件，并在触发时运行一些 JavaScript 代码。



### （2）事件处理方法

method中定义方法



### （3）事件修饰符

<1>. stop：阻止冒泡

（通俗讲就是阻止事件向上级DOM元素传递）

<2>. prevent：阻止默认事件的发生

默认事件指对DOM的操作会引起自动执行的动作，比如点击超链接的时候会进行页面的跳转，点击表单提交按钮时会重新加载页 面等，使用".prevent"修饰符可以阻止这些事件的发生。

<3>. capture：捕获冒泡。

即有冒泡发生时，有该修饰符的dom元素会先执行，如果有多个，从外到内依次执行，然后再按自然顺序执行触发的事件。

<4> . self：将事件绑定到自身，只有自身才能触发，通常用于避免冒泡事件的影响

<5> . once：设置事件只能触发一次，比如按钮的点击等

<6> . passive：能够提升移动端的性能。

### （4）按键修饰符

Vue 允许为 v-on 在监听键盘事件时添加按键修饰符：



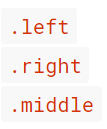
### （5）系统修饰键

**键盘：**



此时Alt+C+其他键也会执行，添加.exact后，只有完全符合才触发。

**鼠标：**

****

### （6）为什么再HTML中监听事件

1. 扫一眼 HTML 模板便能轻松定位在 JavaScript 代码里对应的方法。
2. 因为你无须在 JavaScript 里手动绑定事件，你的 ViewModel 代码可以是非常纯粹的逻辑，和 DOM 完全解耦，更易于测试。
3. 当一个 ViewModel 被销毁时，所有的事件处理器都会自动被删除。你无须担心如何清理它们。

## 15.表单输入绑定

### （1）基本用法

v-model 指令在表单 <input>、<textarea> 及 <select> 元素上创建双向数据绑定。它会根据控件类型自动选取正确的方法来更新元素

（v-model 会忽略所有表单元素的 value、checked、selected 属性的初始值而总是将 Vue 实例的数据作为数据来源。你应该通过 JavaScript 在组件的 data 选项中声明初始值，为空时不替代placeholder）

**文本：**



**多行文本：**



（在文本区域插值 (<textarea>{{text}}</textarea>) 并不会生效，应用 v-model 来代替。）

**复选框：**



单个复选框checked为布尔值，多个复选框可将checked变为数组

**单选按钮：**

****

**选择框：**

单选

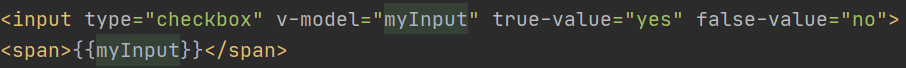
****

多选



### （2）值绑定

**复选框：**

****

选中为yes,未选中为no

**单选按钮：**

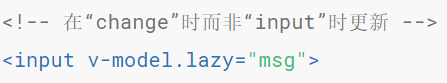
****

**下拉框：**

****

### （3）修饰符

**.lazy**



**.number**

自动将用户的输入值转为数值类型



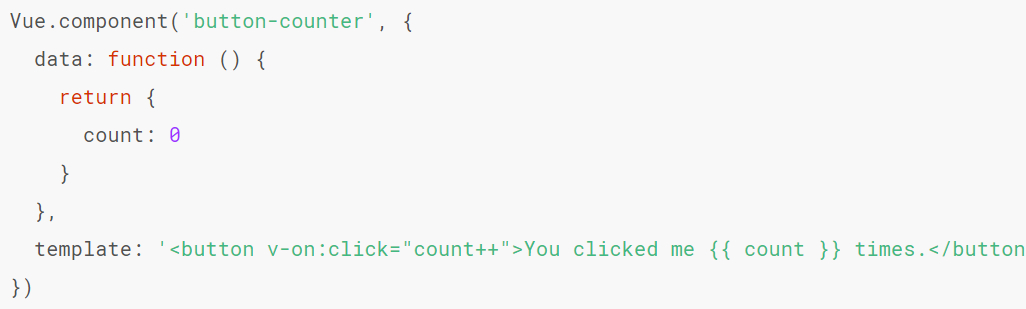
**.trim**

自动过滤用户输入的首尾空白字符



## 16.组件基础

**全局注册：**



此时data必须是一个函数，因此每个实例可以维护一份被返回对象的独立的拷贝，防止彼此影响

**向组件传值：**

****

****

## 17.双向绑定原理

（1）创建Vue对象，包含data\el

（2）创建observer，遍历data，设置其defineProperty，get时返回data[name],set时设置新data[name]。vm.data[name]变为vm[name]

（3）创建 nodeToFragment，遍历node节点，将其放入frag片段，顺便进行compile

（4）创建compile，解析当前节点的‘v-model'指令，与文本节点中的{{}}，将当前的data[name]赋值给它们；

（5）创建 Dep，每个Dep对应一个data中的name，对应其v-model。其中subs存放包含{{}}的节点，含有方法notify负责修改所有节点；

（6）创建 Watcher，每个含有{{}}的文本节点对应一个Watcher，负责对文本节点值的更新update。

（7）在observer阶段设置Dep，在compile阶段设置Watcher.

（8）监听v-model，在compile阶段，发现v-model时，设置当前节点的addEventListener，修改vm[name]，触发set中notify，改变所有{{}}的值。

步骤：

（1）创建Vue构造函数，实现nodeToFragment，对app下的节点遍历，最后将frag插回app下。

（2）nodeToFragment时，顺便对各个节点进行compile解析，查找普通节点与文本节点，分别根据data初始化。

（3）在Vue构造函数中使用Observer，首先做数据代理，其次设置get/set，set负责判断data中的值是否改变，改变后才修改data;

（4）compile中，对包含v-model的普通节点设置addEventListener，当input改变时，对vm[name]赋新值，触发set函数修改data；

（5）创建Watcher，对应每个{{}}，包含方法update（根据data，更新节点显示）

在compile解析时，遇到{{}}的文本节点时，执行Watcher，令Dep.target为当前Watcher，然后执行Update,同时触发get中addsub;

（6）创建订阅者Dep，对应每个data属性，包含方法addSub（添加sub）、notify（通知所有sub进行内容更新）。

在observe遍历data时设置Dep，

当data被get（两种情况，node为普通节点，node为文本节点，只有文本节点{{}}的watcher令Dep.target存在），如果为{{}}的文本节点，就进行addSub，将其Watcher添加。

当data被set时，执行notify令dep中所有Watcher更新视图。

简介版：

（1）双向绑定由两个遍历实现，一个是对节点树的遍历，一个·是对vue实例中data中属性进行遍历

（2）节点数遍历过程中会寻找两种节点，一种是双大括号的文本节点，给它创建一个watcher，里面定义了改变文本的方法；

另一种是含有v-model指令的普通节点，给它定义addEventListner input，令其修改vue实例中的data属性

（3）然后定义observer遍历vue实例中的data属性，首先给每个属性定义一个Dep收集者，用于收集watcher，因为一个绑定的对象可能对应多个值，然后定义对象defineProperty，用于监听每次对data的修改，这样当之前对Input事件的监听就被监听了，它修改data，然后通知Dep对运行其中所有的watcher中更新数据的方法，这样就实现了对

## 18.Vuex

### （1）定义

Vuex是专门为Vuejs应用程序设计的**状态管理工具**。它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。

### （2）构成

**1）state**

state是存储的单一状态，是存储的基本数据。

**2）Getters**

getters是store的计算属性，对state的加工，是派生出来的数据。就像computed计算属性一样，getter返回的值会根据它的依赖被缓存起来，且只有当它的依赖值发生改变才会被重新计算。（倾向于取出状态，做出计算后给其他组件使用）

**3）Mutations**

mutations提交更改数据，使用store.commit方法更改state存储的状态。（mutations同步函数）

**4）Actions**

actions像一个装饰器，提交mutation，而不是直接变更状态。（actions可以包含任何异步操作）

**5）Module**

Module是store分割的模块，每个模块拥有自己的state、getters、mutations、actions。（子vuex）

### （3）使用



### （4）原理

Vuex的设计思想，借鉴了Flux、Redux，将数据存放到全局的store，再将store挂载到每个vue实例组件中，利用Vue.js的细粒度数据响应机制来进行高效的状态更新。

1）vuex的store是如何挂载注入到组件中呢？

* 首先利用Vue的插件机制，使用Vue.use(store)定义全局上的store
* 利用Vue的混入机制，通过Vue.mixin(beforeCreate: vuexInit)，在Vue生命周期beforeCreate钩子函数前初始化vuex

2）vuex的state和getters是如何映射到各个组件实例中响应式更新状态呢？

state：创建Vue实例，将state放入data中实现

getters：借助vue计算属性computed实现数据实时监听（类似于双向绑定功能）

## 19.组件通信