Vue基础

1. Vue实例
2. 构造器
   1. 每个Vue对象都是通过构造函数Vue创造一个Vue的根实例启动的
   2. 实例化时，需要传入一个选项对象，可以包含数据，模板，挂载元素，方法，生命周期钩子等选项
   3. 可以扩展Vue构造器，从而用预定义选项创建可复用的组件构造器

var MyComponent = Vue.extend({

//扩展选项

})

var myComponentInstance = new MyComponent()

1. 属性与方法
   1. 每个Vue实例都会代理data对象里的所有属性
   2. 只有被代理的属性是响应的，在创建之后添加新的属性到实例上，不会触发视图更新
   3. 除了data属性，Vue实例暴露了一些有用的实例属性与方法。这些属性与方法都有前缀$，以便与代理的data属性区分
2. 实例生命周期

3.1 生命周期钩子诸如：created mounted updated destroyed 钩子的this指向调用它的实例

3.2 vue没有控制器的概念，组件的自定义逻辑分布在钩子中

二．模板语法

1. 插值

1.1．文本

1.1.1． 双大括号文本插值，无论何时当绑定的数据对象上的属性发生改变，插值处的内容也会发生改变

1.1.2． 通过v-once指令，可以执行一次性的插值

1.2. 纯HTML

1.2.1 双大括号会将数据解释为纯文本，而非HTML，为了输出正在的HTML需要使用v-html指令，此时被插入的内容会被当做HTML，数据绑定被忽略

1.3. 属性

1.3.1 双大括号不能在HTML属性中使用，若想绑定属性，需要使用v-bind指令

1.4. 使用JavaScript表达式

1.4.1 限制：每个绑定都只能包含单个表达式，{{var a = 1}}不会生效

2. 指令

2.1. 参数

2.1.1 参数在冒号后以冒号指明

2.1.2 <a v-bind:href=”url”> 告知该元素的href属性与表达式url的值绑定

2.1.3 v-on 用于监听DOM事件 参数是监听的事件名v-on:click

2.2. 修饰符

2.2.1 以半角句号.指明的特殊后缀，用于指出一个指令应该以特殊的方式绑定

2.2.2 .prenvent 告诉v-on 指令对于触发的事件调用event.prenventDefault()

3. 过滤器

3.1 可以自定义过滤器，过滤器的目的用于文本转换

3.2 过滤器用在两个地方mustache插值和v-bind表达式

3.3 过滤器应该被添加在JavaScript表达式的尾部，由管道符指示

3.4 为了在其他指令中实现更复杂的数据变换，应该使用计算属性

3.5 过滤函数总接受表达式的值作为第一个参数

3.6 过滤器可以串联，过滤器可以接受参数 {{ message | filterA(‘arg1’, arg2)}} arg1奖传给过滤器作为第二个参数，arg2表达式的值将被求值然后传给过滤器作为第三个参数

4. 缩写

三．计算属性

1. 计算属性：对于任何复杂逻辑，都应当使用计算属性

1.1 基础例子



vm.reversedMessage的值始终取决于vm.message的值，当vm.message发生改变时，所有依赖于vm.reversedMessage的绑定也会更新

1.2.计算缓存VS Methods

1.2.1 可以通过调用表达式method来达到同样效果

<p>reverse message:{{reversedMessage()}}</p>

1.2.2 两者的区别在于，计算属性是基于它们的依赖进行缓存的，只有在相关依赖发生改变时才会重新求值，如果依赖没有发生改变，多次访问reversedMessage计算属性会立即返回之前的计算结果，而不必再次执行函数，对于methods而言，只要发生重新渲染，method调用总会执行该函数

1.3 Computed属性vs Watched属性

1.4 计算setter

1.4.1 计算属性默认只有getter，需要时可提供setter

2.观察Watchers

2.1 当你想要在数据变化响应时，执行异步操作或开销较大的操作时，这是有用的

2.2 还可以使用vm.$watch

四 class与 style绑定

1.绑定HTML Class

1.1 对象语法

1.1.1 传递v-bind:class一个对象，以动态地切换class

1.1.2 也直接绑定数据里的一个对象

1.1.3 绑定返回对象的计算属性，这是一个常用且强大的模式

1.2 数组语法

1.2.1 把数组传给v-bind:class 以应用一个class列表



1.2.2 可以使用三元表达式，根据条件切换列表中的class

1.2.3 用在组件上

2.绑定内联样式

2.1 对象语法

2.1.1 使用v-bind:style



2.2.2 直接绑定到样式对象

2.2 数组语法

2.2.1 将多个样式对象应用到一个元素上

2.3 自动添加前缀

五. 条件渲染

1.v-if

1.1 <template>中的v-if条件组

1.1.1 把<template>元素当做包装元素，并且在上面使用v-if 最终的渲染结果不会包含<template>，用一个其他元素包含也可以如此，v-if的值为false，内部元素都不会渲染

1.2 v-else

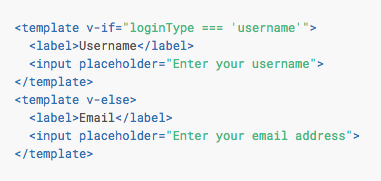
用v-else 表示 v-if的else块

1.3 v-else-if（2.1.0新增）

1.3.1 充当v-if的else-if块，可以链式的使用多次

1.3.2 类似v-else v-else-if 必须紧跟在v-if或者v-else-if元素之后

1.4 用key管理可复用的元素

1.4.1 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

切换loginType将不会清除用户已经输入的内容，因为两个模板使用了相同的元素,input不会被替换掉，仅仅是替换了它的placeholder

1.4.2 希望两个元素独立，不复用它们，需要添加一个具有唯一值的key属性

2.v-show

2.1 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属性display

2.2 v-show不支持<template>语法，也不支持v-else

3. v-if vs v-show

3.1 v-if 是真正的条件渲染 ，确保在切换过程中条件块内的事件监听器和子组件适当地被销毁和重建

3.2 v-if是惰性的，初始渲染条件为假时，什么也不做，直到条件第一次变为真时才会开始渲染条件块

3.3 v-show 不管初始条件是什么，都会被渲染，只是进行简单的css切换

3.4 v-if有更高的切换开销，v-show有更高的初始渲染开销，当切换频繁时使用v-show比较好，如果运行条件不太可能改变，使用v-if

4.v-if和v-for

v-for具有更高的优先级

六.列表渲染

1.v-for

1.1 基本用法

1.1.1. v-for需要以 item in items形式的特殊语法，items是源数据并且item是数组元素迭代的别名

1.1.2. 在v-for块中，我们拥有对父作用域属性的完全访问权限。V-for还支持一个可选的第二参数为当前项的索引

1.1.3可以 用of 替代 in 作为分隔符

1.2 Template v-for

1.2.1. 用带有v-for的<template>标签来渲染多个元素块

1.3 对象迭代v-for

1.3.1 v-for = “value in object” value是属性值

1.3.2 v-for=”(value,key) in object ” 提供第二个参数为键名

1.3.3 v-for=”(value,key,index) in object”第三个参数为索引

1.3.4 在遍历对象时，是按Object.keys()的结果遍历，但是不能保证它的结果在不同的JavaScript引擎中是一致的

1.4 整数迭代 v-for

1.4.1 v-for可以取整数，在这种情况下，它将重复多次模板

1.5 组件和v-for

1.5.1 在自定义组件里，可以像任何普通元素一样用v-for

1.5.2 组件有自己的作用域，所以不能自动传递数据到组件里，因此我们要使用props ，在调用组件的 地方，通过v-bind: xx 和props: [‘xx’] 将数据从父组件中传递到子组件中

2.Key

2.1 当v-for正在更新已渲染过的元素列表时，默认用就地复用策略，如果数据项的顺序改变，Vue不是移动DOM元素来匹配数据项的顺序，而是简单复用此处每个元素，并且确保它在特定索引下显示已被渲染过的每个元素

2.2 以上模式，只适用与不依赖子组件状态或临时DOM状态（表单输入值）的列表渲染输出

2.3 为了给Vue一个提示，以便它能跟踪每个节点的身份，从而重用和重新排序现有元素，需要为每项提供一个唯一key 属性，用v-bind:key=”xx”来帮点动态值

2.4 建议尽可能使用v-for来提供key

2.5 key是识别节点的通用机制。并不与v-for特别关联，还有其他用处

2.6 变异方法：会触发视图更新，改变这些方法调用的原始数组

2.7 重塑数组： 不会改变原始数组，返回一个

2.8 注意事项：不能检测变动的数组的情况

2.8.1 使用索引值直接设置一个项

例如: vm.items[indexOfItem] = newValue

解决方案：Vue.set(example.items,indexOfItem,newValue)

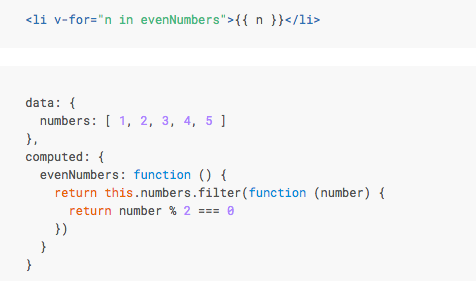
example.items.splice(indexOfItem,1,newValue)

2.8.2 修改数组的长度

例如：vm.items.length=newLength

解决方案：example.items.splice(newLength)

3.显示过滤、排序结果

3.1 显示一个数组的过滤或排序副本，不改变或重置原始数据，可以返回过滤或排序数组的计算属性

3.2 在计算属性不适用的情况下（在嵌套v-for 循环中）使用method方法

七.事件处理器

1.监听事件

1.1 使用v-on 指令监听DOM事件

2.方法事件处理器

2.1 事件处理的逻辑很复杂，直接把javascript代码写在v-on指令中不可行，可以接收一个方法

3. 内联处理器方法

3.1 除了直接绑定方法，还可以用内联javascript语句

3.2 在内联语句处理器中访问原生DOM事件，可以使用特殊变量$event把它传入方法

4.事件修饰符

4.1 在事件处理程序中调用event.preventDefault()是常见需求，但不希望在methods中去处理。希望methods只有纯粹的数据逻辑，而不是去处理细节

4.2 为解决以上问题，Vue提供了事件修饰符 .stop .prevent .capture .self .once

4.3 修饰符可以串联

4.4 .once之外的修饰符都只能对原生DOM事件起作用，.once还能被用到自定义的组件事件上

5.按键修饰符

5.1 在监听键盘事件时，需要检测常见的键值

5.2 记住所有keyCode比较困难，Vue为常用的按键提供别名

5.3 通过全局config.keyCodes自定义按键修饰符别名

八.表单控件绑定

1.基础语法：使用v-model 在表单控件元素上创建双向数据绑定

1.1 文本

1.2 多行文本

1.2.1给显示多行文本的地方加上属性style=“white-space:pre”

1.2.2 在文本区域<textarea></textarea>并不会生效应用v-model 来代替

1.3 复选框

2.绑定value

2.1对于单选按钮，勾选框及选择列表选项，v-model绑定的value通常是静态字符串，勾选框是逻辑值

2.2 我们想绑定value到Vue实例的一个动态属性上，可以用v-bind，同时这个属性值可以不是字符串

3.修饰符

3.1 .lazy v-model在input事件中同步输入框的值，添加修饰符lazy，从而转变为在change事件中同步

3.2 .number 将用户的输入类型转为Number类型

3.3 .trim 自动过滤用户输入的首尾空格

4.v-model与组件

4.1 Vue组件系统允许你创建一个具有自定义行为可复用的Input类型

八. 组件

1.什么是组件

1.1 组件可以是原生HTML元素的形式，以js特性扩展

2. 使用组价

2.1 注册

2.1.1 注册全局组件 使用Vue.component(tagName,options)

2.1.2 组件注册后可以父实例的模块中以自定义元素的形式使用，要确保在初始化实例前注册了组件

2.2 局部注册

2.2.1 通过使用实例选项注册，可以使组件仅在另一个实例组件的作用域中使用

2.2.2 该封装也适用于其他可注册的Vue功能，自定义指令

2.3 DOM模板解析说明

2.3.1 DOM模板解析时,<ul> <ol> <table> <select>限制了能被它包裹的元素

例如 <table><my-row></my-row></table>会被认为无效 使用is解决

2.3.3 使用<script type=“text/x-template”></script> javascript内联模板字符串（由反括号包含的） .vue组件 这些限制不适用

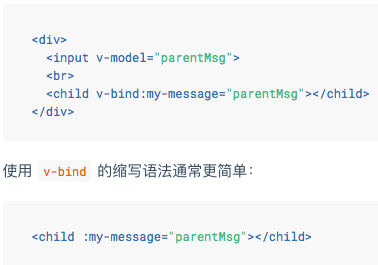
2.4 data必须是函数

2.5 构成组件 props down events up 父组件通过props向下传递数据给子组件，子组件通过events给父组件发送消息

3.Prop

3.1 使用prop传递数据

组件实例的作用域是孤立的，不能在子组件的模板内直接引用父组件的数据，子组件使用父组件的数据，需要通过子组件的props选项

3.2 动态Prop

3.3 字面量语法 vs 动态语法

3.4 单向数据流

3.4.1 prop是单向绑定的，防止子组件无意修改父组件的状态

3.4.2 每次父组件更新时，子组件的所有prop都会更为最新值

3.4.3 想在子组件中修改prop中的数组的原因：1.子组件想把它当作局部数据来用，2.prop传入后，有子组件处理成其他数据输出 正确处理方式： 1.定义局部变量，并用prop的值初始化，2.定义一个计算属性，处理prop的值并返回

3.4.5 js中对象和数组是引用类型，指向同一个内存空间，如果prop是一个对象或数组，子组件内部改变它会影响父组件的状态

3.5 Prop验证

3.5.1 为组件的props指定验证规格，传入的数据不符合规格，Vue会发出警告，当组件给其他人使用时，这很有用

3.5.2 要指定验证规格，需要用对象的形式，而不是字符串数组

4.自定义事件

子组件把数据传递回去，采用自定义事件

4.1 使用v-on绑定自定义事件

4.1.1 每个Vue实例都实现了事件接口

4.1.2 使用$on(eventName)监听事件

4.1.3使用$emit(eventName)触发事件

4.1.4 不能用$on侦听子组件抛出的事件，必须在模板里直接用v-on绑定