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

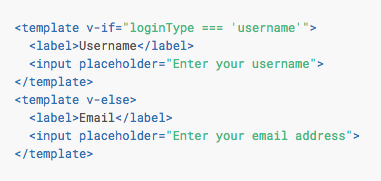
用v-else 表示 v-if的else块

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 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