响独构代插绑指计类

条列数滤监修表生过

组创传事索分动内活模复异递响

自指自过混插命

vue是一个响应的数据与DOM绑定系统

组件系统：可以用独立的可复用的小组件来构建大型应用

构造器

var vm =new Vue({

})

扩展Vue构造器：所有的Vue组件都是扩展Vue实例

var MyComponent = Vue.extend({

})

var MyComponentInstance = new MyComponent()

属性与方法

Vue实例会代理该data对象里所有的属性

Vue实例暴露了一些属性和方法，这些属性和方法都有前缀$，以便区分代理该data对象里所有的属性

数据绑定语法

插值：{{ }}将数据解析为纯文本，可用在text内

v-html指令将数据解析为html

v-once指令只处理单次插值，以后的数据变化不会引起更新

绑定表达式： 表达式，只能包含单个表达式

过滤器，{{ | }}可以串联，可以接受参数（始终以表达式的值

作为第一个参数，带引号的参数视为字符串，不带引号的参数按表

达式计算）

指令：带有v-，值限定为绑定表达式，当绑定表达式的值改变时操作DOM

有些指令可以在指令名称后加冒号和参数

修饰符是以 . 开始的后缀，用以表示指令应当以特殊方式绑定

v-bind:缩写成：

v-on：缩写成@

计算属性 conputed:{

属性名:function(){

}

}

计算属性默认只是getter

conputed:{

属性名:{

get:function(){}

set:function(){}

}

}

class与style绑定：对象语法和数组语法

条件渲染：单个元素直接用v-if，v-else，v-show=””,v-show=”!”

多个元素在外层包<template></template>并用v-if，v-else

区别：v-if局部编译或缺载，v-show切换元素display

运行时条件不太可能改变用v-if，频繁切换用v-show

v-show不支持<template>

列表渲染

v-for支持<template>

v-for操作数据数组：

v-for=”当前数据数组元素 in 数据数组”

v-for=“（当前数据数组元素 ， 索引别名） in 数据数组”

v-for操作数据对象：

v-for=”当前数据对象元素的值 in 数据对象”

v-for=”（当前数据对象元素的值 ， 键别名） in 数据对象”

v-for=”（当前数据对象元素的值 ， 键别名 ， 索引别名） in 数据对象”

v-for操作值：

v-for=”当前数据值 in 整数”

数组变动检测

变异方法：能触发视图更新，方法有pop，push，shift，unshift，sort，splice，

reverse

重塑数组：能触发视图更新，返回新数组，不改变旧数组，方法有concat，slice，

filter

就地复用为:key=””

$set方法：能触发视图更新，应用对象为数据数组对象

$get方法：能触发视图更新，应用对象为数据数组对象

想显示过滤/排序的结果，同时不实际修改或重置原始数据，可以：

1. 创建一个计算属性
2. 在计算属性不适用的情况下（如在嵌套v-for循环中）使用method方法

方法与事件处理器

v-on指令监听DOM事件

1. v-on：事件属性=”方法名”，在Vue实例中

methods：{

方法名：function（event）{

}

}

1. v-on：事件属性=”方法名（参数）”，在Vue实例中

methods：{

方法名：function（参数）{

}

}

1. v-on：事件属性=”方法名（参数，$event）”，在Vue实例中

methods：{

方法名：function（参数，event）{

}

}

事件修饰符（可以串联）

方法中event.preventDefault（）可用.prevent代替，功能是：不在重载页面

方法中event.stopPropagation（）可用.stop代替，功能是：阻止事件冒泡

.self功能是：只当事件在该元素本身（而不是子元素）触发时触发回调

按键修饰符：支持自定义按键别名，支持单字母按键别名

.up .down .left .right

.esc .tab .delete .enter .space

表单控件绑定

1. <input type=”text” v-model=”名称” />

{{名称}}对应input内容

1. <textarea v-model=”名称”></textarea>

{{名称}}对应textarea内容

1. <select v-model=”名称”>

<option value=”value1” selected></option>

<option value=”value2” ></option>

</selected>

{{名称}}对应option的value值

（4）<input type=”radio” value=”value1” v-model=”名称”/>

<input type=”radio” value=”value2” v-model=”名称”/>

{{名称}}对应radio的value值

（5）<input type=”checkbox” value=”value1” v-model=”名称”/>

<input type=”checkbox” value=”value2” v-model=”名称”/>

<input type=”checkbox” value=”value3” v-model=”名称”/>

{{名称}}对应true/false。

若Vue实例的data中有名称：[]则{{名称}}对应选中的value值

绑定值

对于select和input（radio类型）有v-bind：value=”数据中的变量名”

对于 input（checkbox类型）有v-bind：true-value=”数据中的变量名a”

对于 input（checkbox类型）有v-bind：false-value=”数据中的变量名b”

参数特性

.lazy：默认情况下v-model在input事件中同步输入框值与数据

当<input v-model.lazy=”名字”/>时，改在change事件中同步

.number:<input v-model.number=”名字”/>时，自动将用户的输入转为number类型（如果值的转换结果为NaN则返回原值）

实例生命周期

new Vue（）

beforeCreate

Observe Data

Init Events

created No

Has “el” option

Yes when vm.$mount(el) is called

Has “template” option

Yes No

Compile template Compile el`s outerHTML as template

into render function

beforeMount

Create vm.$el and replace “el” with it

mounted

when data changes beforeUpdate

Mounted Virtural Dom

re-render and patch

updated

When vm.$destroyed() is called

beforeDestroy

Teardown watchers

child components and

event listeners

destroyed Destroyed

过渡

1、过渡可以和以下资源一起用

v-if

v-show

动态组件

组件根节点上

1. 当插入或删除带有过渡的元素时
2. 查找css过渡或动画，并在合适时添加或删除css类名
3. 查找过渡钩子，若找到，将在过渡的不同阶段调用相应钩子
4. 若没有找到css过渡或动画也没有找到过渡钩子，插入或删除在下一帧立即执行
5. css过渡：<transition name=”过渡名”></trasition>

根据过渡名有.过渡名-enter和.过渡名-enter-active和.过渡名-leave和

.过渡名-leave-active四个类名

.过渡名-enter定义进入过渡的开始状态。在元素被插入时生效，在下一帧移除

.过渡名-enter-active定义进入过渡的结束状态，在元素被插入时生效，在transition或animation完成之后移除

.过渡名-leave定义离开过渡的开始状态，在离开过渡被触发时生效，在下一帧移除

.过渡名-leave-active定义离开过渡的结束状态，在离开过渡被出发时生效，在transition或animation完成之后移除

css动画：用法同css过渡，区别是在动画中v-enter类名在节点插入DOM后不会立即删除而是在animationend事件触发时删除

显式声明过渡类型：Vue.js需要给过渡元素添加事件侦听来侦听过渡何时结束，该事件要么是transitionend要么是animationend，如果使用其中一个，Vue.js能自动推测出对应的事件类型，若使用两个，需要显示声明希望Vue.js处理的动画类型。用type并设置transition或animation

1. 过渡钩子：<transition v-on：before-enter=”beforeEnter”

v-on：enter=”enter” ...>

</transition>

methods:{

beforeEnter:function(el){}

enter:function(el,done){done()}

afterEnter:function(el){}

enterCancelled:function(el){}

beforeLeave:function(el){}

leave:function(el,done){done()}

afterLeave:function(el){}

leaveCancelled:function(el){}

}

javascript过渡：可以只使用javascript钩子，不定义css规则。当只使用javascript过渡时enter和leave钩子需要调用done回调，否则它们将被同步调用，过渡立即结束。可以v-bind:css=”false”,Vue.js将跳过css检测，这样也阻止无意间让css规则干扰过渡

组件：组件可以扩展html元素，封装可重用代码。组件可以是自定义元素，Vue.js的编译器为它添加特殊功能。组件也可以是原生html元素，以is特性扩展

创建组件：var 组件名=Vue.extend({

template:””

})

选项中若有el或data则应使用函数返回一个对象，否则这个对

象将被该组件所有实例所共享

注册组件：Vue.component(“组件标签名”,组件名)

局部注册：在父组件创建时，选项中有

Component:{

“子组件标签名”,子组件名

}

注册语法糖：注册时子组件名处可以直接匿名创建组件

props：组件的作用域是孤立的，可以使用props传递数据

1. 创建组件时，选项中有props:[属性名]

在组件标签中使用v-bind:属性名=”数据中变量名”，若想传递数字则

在组件标签中使用v-bind:属性名=”数字”。

在组件标签中使用 属性名=”字符形式的值”

1. 可对传递的数据进行验证，创建组件时，props:{属性名:验证}
2. 创建组件时属性名若为驼峰式，则组件标签中属性名为短横式
3. 默认绑定类型是由父传子的单向绑定.sync绑定修饰符为双向绑定

.once绑定修饰符为单次绑定

自定义事件

每个Vue实例都实现了事件借口，即使用$on监听事件，使用$emit触发事件，另外父组件可以在使用子组件的地方直接用v-on来监听子组件触发的事件

子组件索引

<子组件标签 v-ref=”索引名”></子组件标签>

js中可用Vue实例名.$refs.索引名直接访问子组件

v-ref和v-for一起用时，ref是一个数组或对象，包含相应的子组件

slot分发内容

子组件不包含slot标签，子组件中的父组件内容被抛弃

子组件包含slot标签且没有name特性，子组件中的父组件内容将替换slot标签及其内容

具名slot：父组件slot特性，替换，子组件name特性，同值

子组件中slot标签的内容视为回退内容。回退内容在子组件的作用域编译，当宿主元素为空且没有内容供插入时显示这个回退内容

动态组件：多个组件可以使用同一个挂载点，动态地在它们之间切换。使用保留的<component>元素，动态地绑定到它的is特性

父组件创建时，选项中有:data{

currentView:”当前组件名”

}，

components:{

组件名1:{}

组件名2:{}

}

另外有：<component :is=”currentView”>

</component>

keep-alive:如果把切换出去的组件保留在内存中，可以保留它的状态或避免重新渲染

<component :is=”currentView” keep-alive>

</component>

activate钩子：在切换组件时，切入组件切入前可进行一些异步操作，为了控制组件切换时长，给切入组件添加activate钩子

创建组件时，选项中有activate:function(done){}

transition-mode：默认情况下，进入与离开平滑过渡，用这个特性可指定另外两个模式

in-out：新组件先过渡进入，等它的过渡完成之后当前组件过渡出去

out-in：当前组件先过渡出去，等它的过渡完成之后新组件过渡进入

编写可复用组件：

Vue.js组件api来自props，事件和slot

props：允许外部环境传递数据给组件

事件：允许组件触发外部环境的action

slot：允许外部环境插入内容到组件的视图结构内

异步组件：大型应用中，可能需要将应用分为小块，需要时从服务器下载，Vue.js允许组件定义为一个工厂函数，动态解析组件的定义

Vue.component(“组件标签名”,function(resolve,reject){

})

resolve收到从服务器下载的组件定义时调用

reject指示加载失败

递归组件：

组件在它的模板内可以递归地调用自己，要有name选项且递归有终止条件。

当用Vue.component()全局注册一个组件时，组件id自动设置为组件的name选项

var data={a:1}

var vm=new Vue({

data:data

})

data.b=2 //不是响应的

Vue.set(data , “b” , 2) //是响应的

异步更新队列：数据变化后用Vue.nextTick(function(){})

当前实例.$nextTick(function(){})

可完成dom更新

自定义指令

Vue.directive(“指令名”,{

//只调用一次，在指令第一次绑定到元素上时调用

bind:function(){}

//在bind之后立即以初始值为参数第一次调用，之后每当绑定值变化时调

//用，参数为新值与旧值

update:function(newValue,oldValue){}

//只调用一次，在指令从元素上解绑时调用

unbind:function(){}

})

或

Vue.directive(“指令名”,function(value){

}) //用作update

使用v-指令名

属性el：指令绑定的元素

vm：拥有该指令的上下文ViewModel

name：指令的名字

arg：指令的参数

modifiers：指令的修饰符

expression：指令的表达式

descriptor：指令的解析结果

自定义过滤器

Vue.filter(“过滤器名”,function(value){

})

用于|过滤器名

Vue.filter(“过滤器名”,function(value，参数名){

})

用于|过滤器名 “参数值”

动态参数

Vue.filter(“过滤器名”,function(value，参数名){

})

用于|过滤器名 参数名

双向过滤器

Vue.filter(“过滤器名”,{

read:function(val){ //model -> view

return

}

write:function(val,oldVal){ //view ->model

return

}

})

混合：

选项中有mixins：[要混合进来的对象]

混合策略：

同名钩子函数：都会调用，先调用混合进来的对象的，再调用自己的

选项的值为对象如methods，components和directives：将合并到一个对象，如果键冲突则使用自己的

全局混合：Vue.mixin({

})

会影响到每个创建的Vue实例

自定义混合策略：

Vue.config.optionMergeStrategies.要混合项=function(toVal,fromVal){}

对于多数选项的值为对象的，可用methods混合策略：

var strategies = Vue.config.optionMergeStrategies

strategies.要混合项 = strategies.methods

开发插件：

插件范围：

（1）添加全局方法或属性

（2）添加全局资源：指令/过滤器/过渡等

（3）添加Vue实例方法，通过把它们添加到Vue.prototype上实现

（4）一个库，提供自己的api，同时提供上面提到的一个或多个功能

插件名.install=function(Vue,options){

Vue.全局方法或属性=

Vue.directive(“全局资源如指令名”,{})

Vue.prototype.$实例方法名=

}//第一个参数是Vue构造器，第二个参数是一个可选的选项对象

使用插件：Vue.use(插件名)或Vue.use(插件名,{})

vue命名规范

导出name使用全小写短横

文件名使用开头大写驼峰

导入name使用开头大写驼峰

标签使用全小写短横

构建大型应用：

模块化

单文件组件：webpack，browserify

路由

状态管理

与服务器通信

单元测试：karma