# Vue

## MVVM

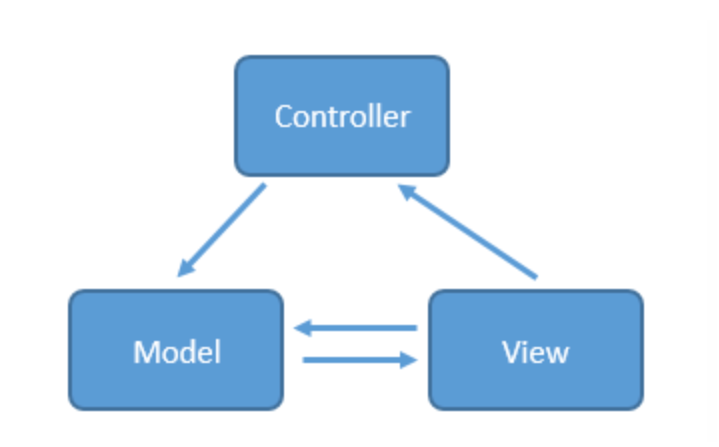
Model view view model

MVC框架 (model view controller)

通过controller的控制去操作model层的数据，返回给view层去展示

工作原理：

当用户发事件到的时候，view层会发送指令到controller层，接着controller去通知model层更新数据，model层更新完数据以后直接显示在view层上，这就是MVC的工作原理

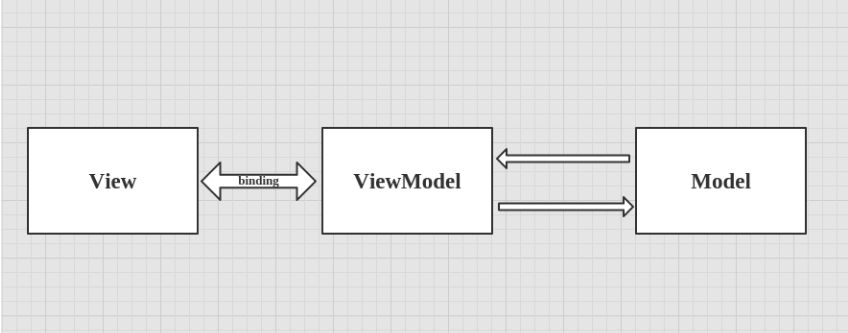


MVVM （model view viewModel）最早是微软提出来的

对数据储存 页面展示 对数据进行加工

View层和viewModel层相互绑定的,根据信息去model中更新数据，然后返回给model

ViewModel是Vue的核心，它是Vue的一个实例。Vue实例时作用域某个HTML元素上的，这个HTML元素可以是body，也可以是某个id所指代的元素。 DOM Listeners和Data Bindings是实现双向绑定的关键。DOM Listeners监听页面所有View层DOM元素的变化，当发生变化，Model层的数据随之变化；Data Bindings监听Model层的数据，当数据发生变化，View层的DOM元素随之变化。



## 指令

差值表达式：{{ a == b ? name1 : name2}}

防止闪烁：v-text = “name” v-html = “name”

V-cloak => [v-cloak]:{display:}

v-bind:value =”name”,也可以直接写 ：value = “name”

在a标签中v-bind = “{herf:‘http://baidu.com’+id}”



v-model = “name” 双向数据绑定 v-model.trim可以去掉首尾空格

v-for = “(item,index) in arr”(“(key,item) in Object”) 2.0

“(index ,item) in arr” 1.0

v-if = “true/false” 添加移除整个元素

v-show = “true/false” 显示隐藏元素

v-on:click = “getlist()” ==> @click = “getlist”

V-on:submit.Prevent 阻止默认事件

## 过滤器

filterBy v-for = “item in list | filterBy “搜索事件” in “ 匹配值” ”

Json {{obj | json}} 转换为json格式

Uppercase {{ name | uppercase}} 转换为大写

Currency {{ num | currency ￥}} 添加货币符号

自定义过滤器 自定义全局过滤器

{{ name | data}} Vue.filter(“data”,function( input ，val ){

New Vue({ 参一 input 是过滤器左边的数据

参二 val 是过滤器格式（yy-mm-dd）可不写

Filters:{ } )

Data:function(input){

Input得值为 | 左边的内容

}

}

})

## 自定义指令

1.0

<div v-”名字”></div>

Vue.directive( ”名字”，function（）{

this.vm 指定作用的区域对象

this.el 当前这个元素(可以设置样式)

this.vm[this.expression] 当前元素自定义指令的值

这里的this.expresssion指定的是dom中绑定元素，

如 v-名字=“这里对应data里的值”

})

2.0

<body>  
 <div id="app">  
 <div v-hehe="color" class="color"></div>  
 </div>  
</body>  
<script src="./pulg/vue221.js"></script>  
<script>  
 Vue.directive("hehe", {  
 bind: function (el,val) {  
 el.style.**background** =val.**value** }  
});  
Vue.directive("lala", function (el, val) {  
 el.**innerHTML** = val.**value**})  
var vm = new Vue({  
 **el**:"#app",  
 **data**:{  
 **color**:"red",  
 **data**:"为了你，而祈祷，而祝福，而感动！"  
 },  
 **methods**:{  
  
 }  
})

window.onload = function() {

// 全局指令

Vue.directive('demo',{

bind:function(el,val){

el.style.color = val.value.color

}

});

Vue.directive('focus',{

inserted: function(el,val) {

el.focus()

}

});

// 局部指令

var app = new Vue({

el: '#box',

directives: {

demo: {

bind: function(el, val) {

el.style.color = "blue";

}

},

focus: {

inserted: function(el,val) {

el.focus()

}

}

}

});

}

## Vue里的ajax

本质上vue-resource是通过http来完成AJAX请求响应的

注意 ： 必须引用vue-resource,

全局Vue对象写法

Vue.http.get('/someUrl', [options]).then(successCallback, errorCallback);

Vue.http.post('/someUrl',[body],[options]).then(successCallback, errorCallback);http来完成AJAX请求响应的

Vue中

Get this.$http.get(“url”).then(funtion(response){

Response.body 获取数据

}) 参三 是固定值

Psot this.$http.post(“url”,{参二 要传的参数}， {emulateJSON:true}).then(funtion(response){

Response.body 获取数据

})

Jsonp this.$http.jsonp(“url”).then(funtion(response){

Response.body 获取数据

})

## 键盘事件

@keydoen.键明（enter） = “函数”

查看v-on 指令中有那些预定义

查看Vue.directive(“on”).keyCodes

添加Vue.directive(“on”).kerCodes.F2 = 113

## Vue对象的生命周期

Init ： 初始化 1.0vue

Created：func（）{ 对象创建成功后自动调用这个方法 1.0 2.0都有

This.方法

}

beforeCreate:function(){ } 下面都是当触发才会执行

mounted:function(){}

## Vue动画

1 外面抱一个<transition>标签

|  |
| --- |
| /\* |
|  | 因为.show-enter-active,.show-leave-active在整个动画过程中都是存在于元素上面的，所以 |
|  | transition:all 1s ease;必须写在这个上面才能持续作用于元素，才能产生动画效果 |
|  | \*/ |
|  | .show-enter-active,.show-leave-active{ |
|  | transition:all 1s ease; |
|  | } |
|  | .show-enter：标记动画开始的位置为padding-left: 100px; |
|  | .show-leave-to:标记动画离开的结束位置为padding-left: 100px |
|  |  |
|  | \*/ |
|  | .show-enter,.show-leave-to{ |
|  | padding-left: 100px; |
|  | } |
|  | /\* |
|  | .show-leave:标记动画离开时的初始位置为：padding-left: 10px;; |
|  | .show-enter-to：标记动画开始的结束位置为：padding-left: 10px;; |
|  | \*/ |
|  | .show-enter-to,.show-leave{ |
|  | padding-left: 10px; |
|  | } |

2 结合animate.css实现动画

引入animate.css，执行动画的必须是块级元素

3 钩子函数

在标签中写入事件，在js中用方法执行

//钩子函数

beforeEnter:function(el){

el.style.transform = "translate(0,0)";

},

// 表示在动画执行过程中要触发，设定三个步骤：

// 1、设定当前动画的每一帧刷新

// 2、设定当前元素的结束位置

// 3、手动调用动画结束的方法 done()

Enter:function(el,done){

// 1、设定当前动画的每一帧刷新

el.offsetWidth;

// 2、设定当前元素的结束位置

el.style.transform = "translate(30px,430px)";

// 3、手动调用动画结束的方法 done()

done();

},

// 动画结束以后，将动画元素的v-if中的变量值重置会初始值

afterEnter:function(el){

this.isshow = !this.isshow;

},

## 定义和注册组件

1）var xxx =Vue.extend({

template:"<h2>这是第一种方法</h2>"

})

Vue.component("aaa",xxx) ======>>> <aaa></aaa>

2） Vue.component("lala",{

template:"<h2>这是第2中方法</h2>",

Data:function( ){

Return:{

Info:”123”

}

}.

Methods;{

}

})

1. 定义模板传id

<haha></haha>

<template id="com">

<h2>我是第三种方法</h2>

</template>

Vue.component("hehe",{

template:"#com" 注意如果是<script type=”x-template”></script>

})

定义子组件

Vue.component("lala",{

template:"<div><h2>这是第2中方法</h2><com></com></div>",

components: {

"com": {

template: "<h2>我是子组件</h2>"

}

}

})

组件之间相互切换

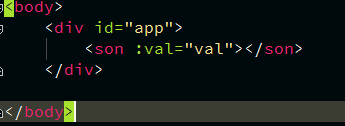
v-bind绑定 决定用一个模板

用<component :is=”xxxx”></component>

通过注册一个点击事件变化xxxx的值来实现组件之间的切换

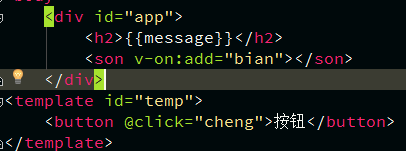
### 组件之间的父子传值

父向子传值



子向父传值

注意：子组件中的data 是一个函数 ，返回一个对象

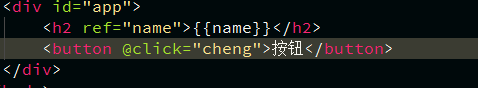


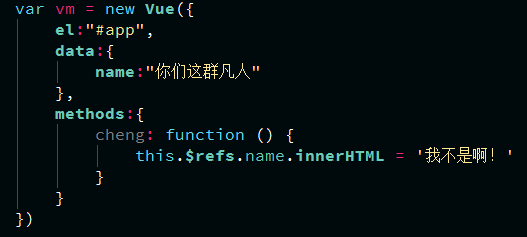
获取DOM元素中的值

2.0版本

DOM组件中获取元素 用ref= “xxx”

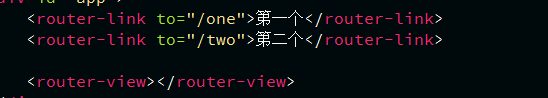
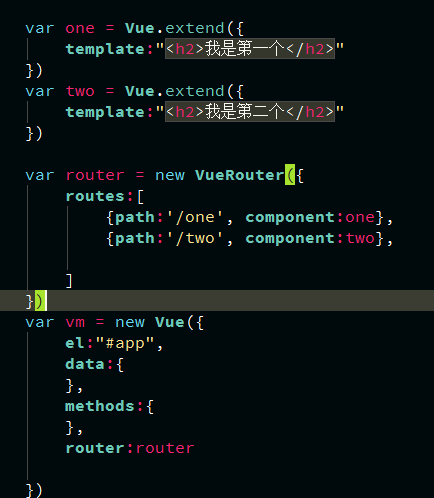
组件中获取元素 this.$refs.name.colo = 萨尼餐此案





## vue路由

首先要引入vue-router



<router-link to=”xxxx/id”>

在vue中 的{pash:”xxxx /:id”,component:xxxx}

获取这个id 可以用 this.$route.param.id

## 获取当前元素

Vue1.0

v-el 和 v-ref写在标签中 <p v-el:name v-ref:name>

Dom中获取元素 this.$el.name 可以获取该元素

组件中获取元素 this.$refs.name

Vue2.0

<v-ref=”name”>

Dom和组件中获取元素全是 thi.$refs.name

## Vue中响应对象的添加删除方法

用set (添加) delete（删除）



## 其他

要先引入require.js

Es5 导入require(“url”)，导出 module.expores{}

Es6 导入import name from “url” 导出 export default{ }r