# 一、Vue的介绍

Vue是一个渐进式的js框架，只注重视图层，结合了HTML+CSS+JS，非常的易用，并且有很好的生态系统。而且vue体积很小，速度很快，优化很到位。

SSM的整合，搞定了后端一套： 增删改查

# 二、MVVM

Vue技术是MVVM开发模式的实现者。

MVC

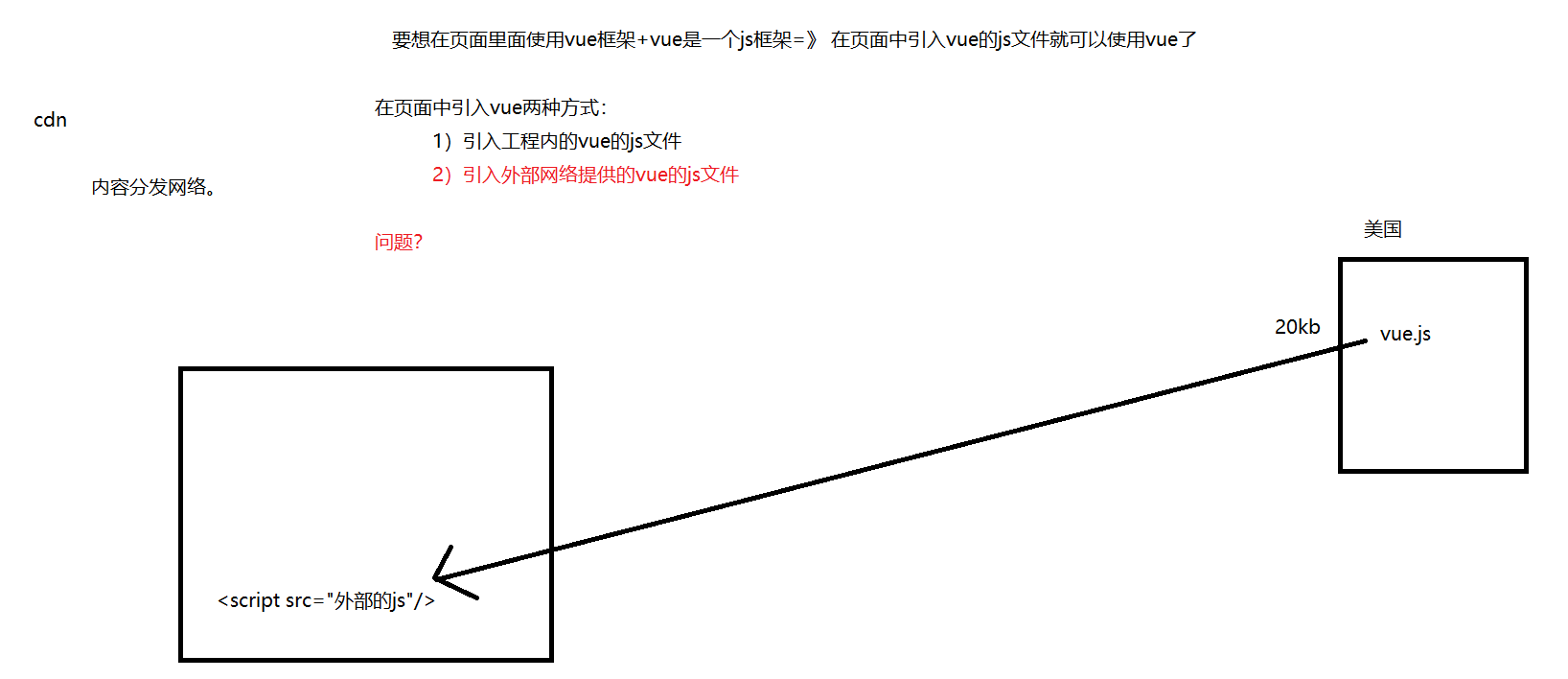
M model V view C controller

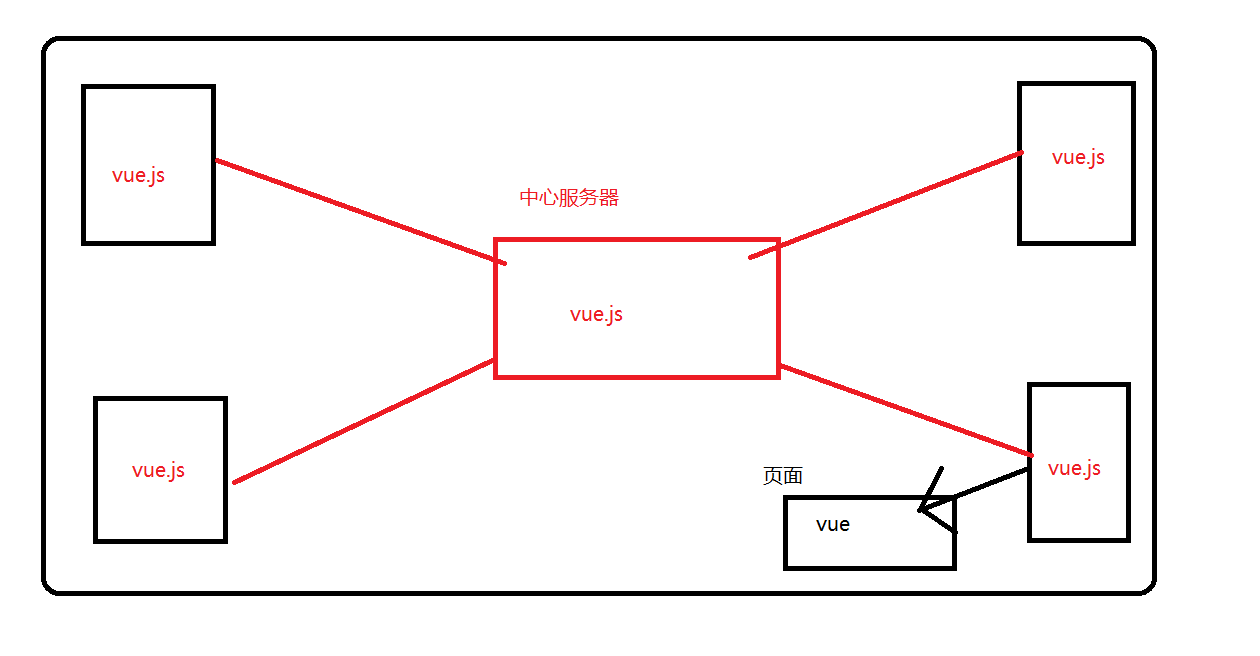
MVVM

M model V view VM viewmodel: 连接视图和数据的中间件。

## 1.cdn 内容分发网络

这是一种加速策略，能够从离自己最近的服务器上快速的获得外部的资源。





## 2.VM的实现原理

view model中内置了一个观察者，这个观察者观察两个维度

1） 观察视图的变化： 当视图变了，就通知数据进行变化

2）观察数据的变化： 当数据变了，就通知视图进行变化

—— MVVM通过VM实现了双向数据绑定。

# 三、Vue 的快速开始

## 1.如何获得vue的cdn文件

<https://www.bootcdn.cn/vue/>

## 2.如何在页面中使用vue

两个部分:

1) html： <div id="app"></div>

2)需要有一个Vue对象（实例）

## 3.vue对象里有哪些东西，分别是什么用？

new Vue({

el:'' 该vue对象绑定在哪个div上

data:{

} 提供数据的，里面存放键值对

})

## 4.在html的被vue绑定的元素中，通过插值表达式来获取vue对象中的数据。

|  |
| --- |
| <**body**>  <**div id="app"**>  欢迎你！年龄是{{age}}的{{name}}  </**div**> </**body**> <**script src="https://cdn.bootcss.com/vue/2.6.10/vue.js"**></**script**> <**script**>  **new** Vue({   **el**:**'#app'**,*//element  //数据哪里来？* **data**:{  **name**:**'小明'**,*//以后，数据就是通过发送ajax请求，来获得的。* **age**:18  }  });*//json格式的对象 使用大括号包裹，里面放了键值对，在js中键可以没有引号,多个键值对之间使用‘,’分隔。* </**script**> |

# 四、差（插）值表达式

差值表达式是用在html中被绑定的元素中的。目的是通过差值表达式来获取vue对象中的属性和方法。

语法格式： {{vue的内容}}，注意 差值表达式不能写在html的标签中，不能作为属性的值的部分。

vue对象中的属性是哪里提供？

new Vue({

data:{} <== 这个data就提供了属性

})

vue对象中的方法是哪里提供？

new Vue({

methods:{ <== 这各methods就是提供方法的

sayHi:function(){

alert("hello vue")

}

}

});

除此之外，差值表达式也能够这么使用：

|  |
| --- |
| <**div id="app"**>    {{[0,1,2,3,4][1]}}<**br**/>  {{ {name:'xiaoming',age:20}.name }}   </**div**> |

# 五、Vue中的关键字

这些关键字都是作为html页面的标签中的属性

## 1. v-model

是将标签的value值与vue实例中的data属性值进行绑定。

|  |
| --- |
| 请输入您的专业：<**input type="text" v-model="major"** />  **new** Vue({  **el**:**'#app'**,  **data**:{  **major**:**'java'** }  }); |

## 2. v-on

通过配合具体的事件名，来绑定vue中定义的函数

|  |
| --- |
| <**input type="text" v-on:click="changeMajor"** />  **new** Vue({  **el**:**'#app'**,  **data**:{  **major**:**'java'** },  **methods**:{  sayHi:**function**(){  *alert*(**"HELLO VUE!"**);  },  changeMajor:**function**(){  ***console***.log(**"change Title"**)  }  } }); |

补充知识：

（1）event.target.value:

比如在响应函数里，可以指明使用event内置的参数对象。该对象表示当前事件，可以通过event.target.value来获得当前事件对象的value的值。

（2）this的用法

this表示当前vue对象 “new Vue()”,可以通过“this.”来调用当前vue对象的属性和方法。

（3）v-on还可以简写

v-on:input="" ==> @input=""

## 3. v-bind

我们知道差值表达式是不能写在html的标签的属性内的，那如果一定要用vue中的属性作为html标签的属性的内容，就可以通过v-bind进行属性绑定。

new Vue({

data:{

link:'http://www.baidu.com'

}

});

<a v-bind:href="link"></a>

注意： v-bind也可以简写：

<a v-bind:href='link'></a>==> <a :href='link'>

## 4.v-once

此时该标签中的差值表达式，只获取一次数据。之后数据的变化不影响此差值表达式的值。

## 5. v-html和v-text

v-html会将vue中的属性的值作为html的元素来使用

v-text会将vue中的属性的值只作为纯文本来使用。

|  |
| --- |
| **new** Vue({  **el**:**'#app'**,  **data**:{  **link**:**'http://www.baidu.com'**,  **mylink**:**'<a href="http://www.qfedu.com">千锋</a>'** }  });  <**span v-html="mylink"**></**span**> <**span v-text="mylink"**></**span**> |

效果：



# 六、Vue中的事件

## 1.vue中如何使用事件

## 2.事件的参数传递

分成三个部分：

设参：

<**button type="button" @click="addbtnfn(2)"** >增加</**button**>

传参

addbtnfn:**function**(step){

接参:

this.count+=step;

## 3.vue中的事件修饰符

@click.stop 阻止点击事件的传播

@mousemove.stop 组织鼠标移动事件

@keyup.space 空格键弹起时

...

# 七、computed 计算属性

一些常用的函数，可以缓存起来，在调用时直接使用缓存中的过程（结果），以此来提高效率。

注意： computed里虽然存放的是函数。但在调用时，computed里的东西是一个属性。所以我们在调用时不能使用'()' 因为()是在调用函数，而不是在调用属性。

# 八、watch 监控属性

通过watch里给属性绑定函数，当属性的值发生变化时，该函数就会自动被调用。调用时可以接收两个参数，第一个参数时属性改变后的值，第二个参数是属性改变前的值。

|  |
| --- |
| <**div id="app"**>   {{title}}  <**input type="text" v-model="title"**>   </**div**>  </**body**> <**script src="https://cdn.bootcss.com/vue/2.6.10/vue.js"**></**script**> <**script**>   **var *v1*** = **new** Vue({  **el**:**'#app'**,  **data**:{  **title**:**"hello vue"** },  **watch**:{  title:**function**(newValue,oldValue){   ***console***.log(newValue+**":"**+oldValue);  }  }   }); </**script**> |

# 九、 Vue改变样式

1.通过给html元素的class属性绑定vue中的属性值，得到样式的动态绑定

new Vue({

data:{

temp: true

}

})

<div :class="{red:temp}"/> 如果temp是true ==>> <div class="red"/>

如果temp 是false==>> <div/>

2.通过computed返回一个对象，对象里放着多个键值对

|  |
| --- |
| <**body**>   <**div id="app"**>  <**div v-bind:class="{red:temp}" class="mydiv"**></**div**>  <**hr**/>  <**div :class="myClassStyle" class="mydiv"**></**div**>  <**hr**/>  <**div class="mydiv"**></**div**>   <**button type="button" @click="temp=!temp"**>点我</**button**>  </**div**>   </**body**> <**script src="https://cdn.bootcss.com/vue/2.6.10/vue.js"**></**script**> <**script**>  *//v-bind:class="{red:true}==> class="red"* **new** Vue({  **el**:**'#app'**,  **data**:{  **temp**:**false** },  **computed**:{  myClassStyle:**function**(){  **return** {  **red**:**this**.**temp**,  **mywidth**:**this**.**temp** }   }   }  })  </**script**> |

3.通过直接双向绑定vue中的属性来改变样式

|  |
| --- |
| <**head**>  <**meta charset="UTF-8"**>  <**title**>Title</**title**>  <**style**>   .**mydiv**{  **width**: 400**px**;  **height**: 220**px**;  **background-color**: **gray**;  }   .**red**{  **background-color**: **red**;  }  .**yellow**{  **background-color**: **yellow**;  }  .**mywidth**{  **width**:450**px**;  }  .**green**{  **background-color**: **green**;  }  .**blue**{  **background-color**: **blue**;  }   </**style**> </**head**> <**body**>   <**div id="app"**>    <**div :class="mycolor" class="mydiv"**></**div**> ==》 <div class='green'/>   <**input type="text" v-model="mycolor"**>  </**div**>   </**body**> <**script src="https://cdn.bootcss.com/vue/2.6.10/vue.js"**></**script**> <**script**>  *//v-bind:class="{red:true}==> class="red"  //div class="green"* **new** Vue({  **el**:**'#app'**,  **data**:{  **temp**:**false**,  **mycolor**:**'green'** }  }) </**script**> |

4.在html标签中使用多个样式的组合

|  |
| --- |
| .**red**{  **background-color**: **red**; } .**mywidth**{  **width**:450**px**; }  **<div :class="[mycolor,mw]" class="mydiv"**></**div**>  注意：不能这么写：**div :class="mycolor" :class="mw" class="mydiv"**></**div**>  **new** Vue({  **el**:**'#app'**,  **data**:{  **temp**:**false**,  **mw**:**'mywidth'**,  **mycolor**:**'green'** } |

5.在内嵌的css样式里指明style属性的值

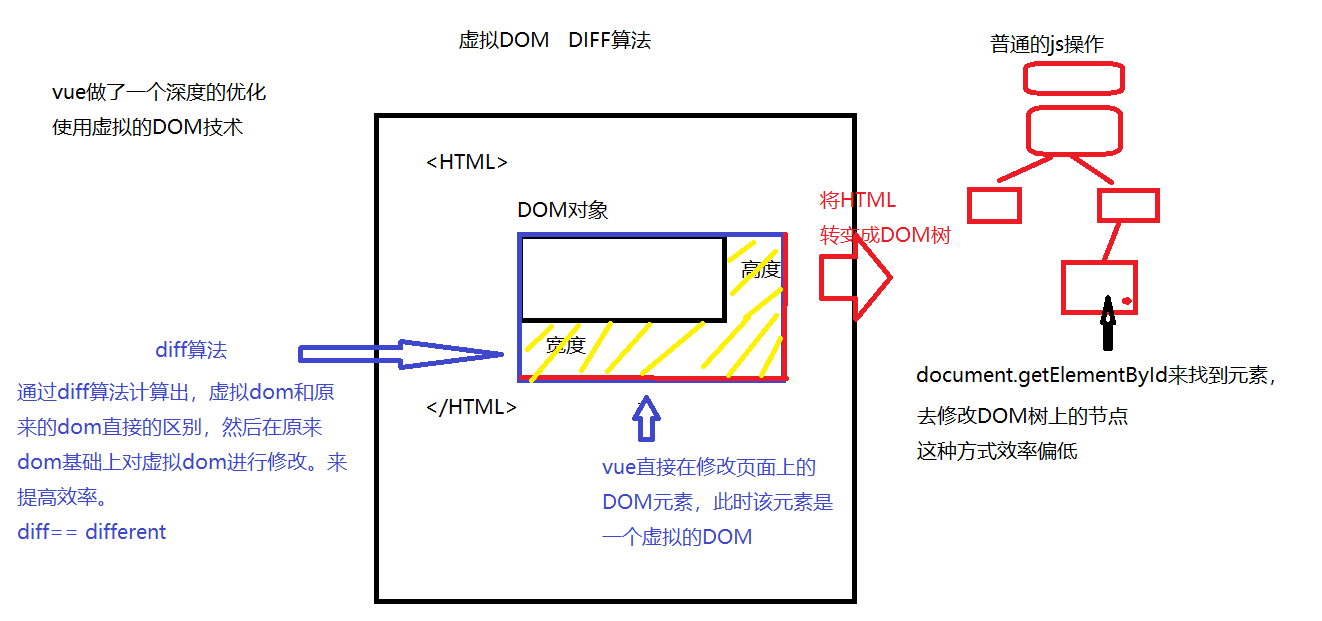
|  |
| --- |
| <**body**>   <**div id="app"**>  <**div v-bind:class="{red:temp}" class="mydiv"**></**div**>  <**hr**/>  <**div :class="myClassStyle" class="mydiv"**></**div**>  <**hr**/>  <**div :class="mycolor" class="mydiv"**></**div**>  <**hr**/>  <**div :class="[mycolor,mw]" class="mydiv"**></**div**>  <**hr**/>  <**div :style="{backgroundColor:bc}" class="mydiv"**></**div**>  注意：style引用了vue中的内容，因此是一个键值对，所以需要大括号，对象的键是不能出现“background-color”，应该改成  backgroundColor.  <**button type="button" @click="temp=!temp"**>点我</**button**>  <**input type="text" v-model="mycolor"**>  </**div**>   </**body**> <**script src="https://cdn.bootcss.com/vue/2.6.10/vue.js"**></**script**> <**script**>   *// document.getElementById("#sp").style.backgroundColor   //v-bind:class="{red:true}==> class="red"  //div class="green"* **new** Vue({  **el**:**'#app'**,  **data**:{  **temp**:**false**,  **mw**:**'mywidth'**,  **mycolor**:**'green'**,  **bc**:**'blue'** },  **computed**:{  myClassStyle:**function**(){  **return** {  **red**:**this**.**temp**,  **mywidth**:**this**.**temp** }  }  }  }) </**script**> |

6.在style里使用多个样式的组合（数组）

|  |
| --- |
| <**body**>   <**div id="app"**>  <**div :style="[myStyle,{height:myHeight+'px'}]" class="mydiv"**></**div**>  <**button type="button" @click="temp=!temp"**>点我</**button**>  颜色：<**input type="text" v-model="bc"**><**br**/>  高度:<**input type="text" v-model="myHeight"**>  </**div**>   </**body**> <**script src="https://cdn.bootcss.com/vue/2.6.10/vue.js"**></**script**> <**script**>**new** Vue({  **el**:**'#app'**,  **data**:{  **temp**:**false**,  **mw**:**'mywidth'**,  **mycolor**:**'green'**,  **bc**:**'blue'**,  **myHeight**:300  },  **computed**:{  myClassStyle:**function**(){  **return** {  **red**:**this**.**temp**,  **mywidth**:**this**.**temp** }  },  myStyle:**function**(){  **return** {  **backgroundColor**:**this**.**bc**,  }  }  }  })  </**script**> |

# 十、Vue的核心：虚拟DOM和diff算法

vue的高效的核心，就是虚拟的dom和diff算法，vue不通过修改dom树来达到修改的效果，而是直接在页面上该那个元素，此时这个元素就是一个虚拟的dom。那么vue怎么去改呢？ 通过diff算法，计算出虚拟的dom修改后和修改前的区别，然后在虚拟dom的原基础上进行修改，这样的效率就大大提升了。



# 十一、Vue中的分支语句

## 1.v-if

<div v-if="temp">!!!</div>

当temp的值是true时，div中的内容才会显示

## 2.v-else

就是v-if的对立面

## 3.v-else-if

就是在else里再嵌套一个if

|  |
| --- |
| <**div id="app"**>  <**div v-if="temp"**>看到我了！</**div**> 当temp是true时能显示    <**div v-else-if="temp1"**>!!!!前方高能!!!!</**div**> 当temp是false时，且temp1是true时能显示   <**div v-else="temp"**>看到他了！</**div**> 当temp是false时，且temp1是false时能显示   <**button type="button" @click="temp=!temp"**>点我</**button**>  </**div**> </**body**>  <**script src="https://cdn.bootcss.com/vue/2.6.10/vue.js"**></**script**> <**script**>   **new** Vue({  **el**:**'#app'**,  **data**:{  **temp**:**false**,  **temp1**:**false** }  }); </**script**> </**html**> |

## 4. v-show

用法上和v-if是相同的。也就是说 v-show="布尔值变量" 是true的时候，就会显示内容，是false的时候就不会显示内容，但是v-show改变的是元素的样式，不显示内容时样式是： display是none，而v-if是直接让元素消失和直接添加元素，效率上，v-show更高。

## 5.template模板标签的使用

在vue中经常会碰到这个标签，目前可以使用该标签配合v-if实现多个元素一起出现，或一起消失。注意，template不能和v-show一起用。

<**template v-if="temp"**>  
 <**div**>  
 <**div**>hello vue1</**div**>  
 <**p**>hello vue2</**p**>  
 </**div**>  
  
</**template**>

# 十二、Vue中的循环语句

v-for 是vue中循环的关键字

## 1.普通的for

我们需要定义数据源，然后通过v-for来遍历数据源，再使用差值表达式输出数据

|  |
| --- |
| <**body**>  <**div id="app"**>   <**ul**>  <**li v-for="a in args"**>{{a}}</**li**>  </**ul**>  </**div**> </**body**>  <**script src="https://cdn.bootcss.com/vue/2.6.10/vue.js"**></**script**> <**script**>   **new** Vue({  **el**:**'#app'**,  **data**:{  **args**:[1,2,3,4,5,6]  }  });  </**script**> |

## 2.带着索引的for

<li v-for=" (a,i) in args" :key='i'>{{i}}{{a}}</li>

此时的i就是每次循环的循环变量 ，从0开始一直到元素个数-1

## 3.遍历一个对象

<**ul**>  
 <**li v-for="(v,k,i) in student"**>{{i}}--{{k}}--{{v}}</**li**>  
</**ul**>

**new** Vue({  
 **el**:**'#app'**,  
 **data**:{  
 **args**:[1,2,3,4,5,6],  
 **students**:{  
 **name**:**'xiaoming'**,  
 **age**:20  
 }  
 }  
});

## 4.遍历一个对象数组：嵌套的for循环

<**ul**>  
 <**li v-for=" student in students"**>  
 <**span v-for="(v,k,i) in student"**>{{i}}--{{k}}--{{v}}--------------------</**span**>  
 </**li**>  
   
</**ul**>

**new** Vue({  
 **el**:**'#app'**,  
 **data**:{  
 **args**:[1,2,3,4,5,6],  
 **students**:[  
 {  
 **name**:**'xiaoming'**,  
 **age**:20  
 },  
 {  
 **name**:**'xiaowang'**,  
 **age**:21  
 }  
 ]  
 }  
});

# 十三、Vue对象的操作

## 1.可以通过一个Vue对象操作另一个Vue对象

var v1 = new Vue({

el:'',

data:{title:'hellovue'},

computed:{}

});

var v2 = new Vue({

methods:{

ct1:function(){

v1.title='hello js';

}

}

})

## 2.Vue对象操作另一个vue对象的内容，维度有两个，操作属性、操作方法

这些属性，是data或computed里定义的

## 3.Vue的实例属性

直接通过对象.的方式的调用的属性，是来自于data或computed中的属性，但是vue对象中的el

、data等等这些键也称为属性，这些属性就是vue对象的实例属性！

注意：

1)ref的使用：

在vue里面，往往使用ref属性来代替id属性的使用。那么就可以快速的通过调用ref的值来获得页面中的某个元素。

<**button type="button" ref="mybtn1" @click="showVueObject"**>点我</**button**>

this.$refs. **mybtn1.innerHTML = "hello"**

2)mount的使用：

实现了页面的元素和vue对象的动态绑定，之前都是通过el的方式来绑定，也可以通过mount实例属性进行绑定。

|  |
| --- |
| <**body**>  <**div id="app"**>   </**div**> </**body**>  <**script src="https://cdn.bootcss.com/vue/2.6.10/vue.js"**></**script**>  <**script**>   **var *v1*** = **new** Vue({  **template**:**"<h1>hello</h1>"** })  ***v1***.$mount(**"#app"**); |

# 十四、Vue的组件

万事万物皆组件

之前提到Vue的一大特性： 组件化。也就是可以将vue对象作为一个组件，被反复的使用。

要想实现组件化，需要在页面中注册组件：关于注册的方式有两种，分别是全局注册和本地注册。

## 1.注册组件：（全局注册）

Vue.component("组件名",{vue对象})

## 2.使用组件

在被vue绑定了的html元素中才能使用组件。如果一个div没有被vue绑定，那么这个div中不能使用之前注册的组件。

|  |
| --- |
| <**body**>   <**div id="app"**>   <**model1**></**model1**>  <**model1**></**model1**>  <**model1**></**model1**>    </**div**>   </**body**>  <**script src="https://cdn.bootcss.com/vue/2.6.10/vue.js"**></**script**> <**script**>    Vue.**component**(**"model1"**,{   **template**:**"<div>{{title}}<button type='button' @click='btnfn'>点我</button></div>"**,  data:**function**(){  **return** {  **title**:**"hello vue"** }  },  **methods**:{  btnfn:**function**(){  *alert*(**"hello !!!"**);  }  }  });    **new** Vue({  **el**:**'#app'** })  只有div被vue绑定，才能使用vue 的组件。   </**script**> |

## 3. 作为组件的vue对象的注意事项

### 特点1

Vue.component("组件名",{vue对象})，这个vue对象和之前的vue对象的data实例属性的写法是有区别的。

new Vue({

data:{

name:"xm",age:29

}

})

======>

{

data:function(){

return {

name:"xm",

age:29

}

}

}

### 特点2

这种组件中template的写法

template是将内容展现在页面上的一个键。值是一个字符串。一定要注意一点： template里必须有且只能有一个根元素。

这种写法是错误的：

**template**:**"<h1>{{title}}</h1><button type='button' @click='btnfn'>点我</button>"**,

应该这么写：

**template**:**"<div><h1>{{title}}</h1><button type='button' @click='btnfn'>点我</button></div>"**,

# 十五、Vue实例的生命周期

一个Vue对象会经历 初始化、创建、绑定、更新、销毁等阶段，不同的阶段，都会有相应的生命周期钩子函数被调用。参考官方文档的生命周期函数示意图。

# 十六、Vue组件的本地（局部）注册

vue的全局注册，也就意味着在页面的任意一个被vue绑定过的div中，都可以使用全局注册了的vue组件。

但是，如果是对vue组件进行本地注册，那么在其他被vue绑定的div中，不能使用该组件。

|  |
| --- |
| <**body**>   <**div id="app"**>   <**model11**></**model11**>  </**div**>  <**hr**/>  <**div id="app1"**>  <**model11**></**model11**>   </**div**>   </**body**>  <**script src="https://cdn.bootcss.com/vue/2.6.10/vue.js"**></**script**> <**script**>   *// var model1 =* **new** Vue({  **el**:**'#app'**, //只有id是app的div才能使用model11组件  **components**:{  **"model11"**:{   **template**:**"<div><h1>{{title}}</h1><button type='button' @click='btnfn'>点我</button></div>"**,  data:**function**(){  **return** {  **title**:**"hello vue"** }  },  **methods**:{  btnfn:**function**(){  *alert*(**"hello !!!"**);  }  }  }  }  })  **new** Vue({  **el**:**'#app1'//这个div不能使用model11组件** })    </**script**> |

首页的静态图

