Vue 相关内容

1. Vue
   1. 代码方式

<main id = ‘ app ‘></main>

<script>

Var vue = new Vue( {

el : ‘ #app ‘

data : { }

} )

</script>

} )

* 1. 指令

<h1 V - text = ‘ des ’ > </h1> 设置文本 = innerText = $ ( ) . text ( )

<h1 V - html = ‘ date ’ ></h1> 设置文本 = innerHTML = $ ( ) . html ( )

<Img V - bind : src = ‘ imgSrc ’ alt = ‘’ > 绑定属性

V - show ： display - none

<a : href = ‘ href ’ > 绑定属性

V - on : click = ‘ firstClick ( ) ’ 事件绑定

@ input = ‘ change ( $event ) ’ 事件绑定 $event 表示当前响应的事件

* 1. 事件阻止

@click . stop 停止注定添加的事件，也可以用来阻止冒泡/挖洞事件，

只能停止自己的事件，不能停止兄弟标签的事件

@click . stop = ‘ boxClick ( ‘ 子标签 ’ ) ’ 停止父标签的事件，同时自己添加一个新事件

@click . stop . prevent = ‘ taobao ( ) ’ 阻止事件传递 阻止默认事件 添加一个新的事件

@click . oncer = ‘ myClick( ) ’ 只能点击一次 :曾经

* 1. 条件判断

V - if = ‘ ’ V - else

V - if = ‘ ’ v - else - if = ‘ ’ v - else

* 1. for循环

V - for = ‘ book in bookList ’ {{ book . name }}

V - for = ‘ ( book , index ) in bookList ’ {{ book . name }}

* 1. 类名加载

<style> . error{ bgc = ‘ red ’ } </style>

Data : { showError : ‘ error ’ }

<div : class = ‘ showError ’></div>

* 1. methods方法 用来存放所有的方法

Methods : { click : funciton ( ) { } }

Data里面的方法比methods里面的方法的优先级要高

* 1. Watch 监听 监听某一个属性是否发生了改变，如果发生了改变，自动调用属性对应的方法

<div v - text = ‘ pro ’></div>

Pro : ‘ xxx ’

Methods : { Click :function () { this . pro = ‘ AA ’ } }

Watch : { pro :function () { console . log ( ‘ XX被改变了 ’ ) } }

* 1. Computed 当某一个因素发生改变的时候要怎么办

Computed里面写的是属性，对应的方法内部是公式，当公式的各个元素没有发生改变的时候，

公式不会重新计算，而是会使用上次缓存的结果，当任何一个元素的值法神改变的时候，结果

会重新计算，并且重新缓存

Computed : function () { var XX = this . AA + this. BB + this . CC return XX }

* 1. 双向数据绑定

V - model = ‘ ’

* 1. 过滤器

全局过滤器 ：Vue . filter ( ‘ global ’ , function ( value ) { return ‘ 全局过滤器 ’ })

<h1> {{ hero | global }} </h1>

局部过滤器 ： (只留以为小数)

<h1> {{ hero | changePrice }} </h1>

<h1> {{ gero | addDes }} </h1>

Filtesr : {

changePrice : function ( value ) {

Var priceArray : value . split ( ‘ . ’ )

Var lastChar : priceArray [ 1 ] [ 0 ]

Var fullPrice = priceArray [ 0 ] + ‘ . ’ + lastChar

Return fullprice

} ，

AddDes : function ( value ) { (条件判断)

If ( value == ‘ xx ’ ) {

Return value + ‘ aa ’

}

Return value + ‘ bb ’

}

}

* 1. 全局过滤器和局部过滤器的区别

1. 声明方式不同

局部：vue对象内部fileters 里面声明

全部：vue . filter 声明

1. 作用范围不同

局部：只对自己声明的vue对象有效

全局：对任何vue对象都有效

* 1. 组件 声明组件的时候要遵守命名规范 1.字母能大写 2.中间以 - 隔开

1. Vue . component ( ‘ globalo - com ’ , { template : ‘ <h1> 全局组件 </h1> ’ } )

<global - com >< / global - com>

1. Var obj = { template : ‘ <h1>XXX</h1> ’ }

Components ：{ ‘ first - com ’ : { template : ‘ <h1> 局部组件 </h1>} ,

‘ second - com ’ : obj }

1. <first - com></first - com>

<second - com></second - com>

1. <script id = ‘ gishi ’ type = ‘ text / html ’>

<h1>xxxx</h1>

<h1>xxxx</h1>

<h1>xxxx</h1>

</script>

Components : { ‘ gu - shi ’ : { template : ‘ # gushi ’ } }

<gu - shi></ gu - shi>

* 1. 特殊组件 tabel ul 之类

table里面如果放置非table子元素(tr)的内容和组件，那么这些内容和组件会被排除到table 以外，然后再来加载vue,js内容，但是加载的时候table内部已经没有子元素或者组件。

解决方式，在table内部使用tr，或者<tr is =’ 组件 ’>这种方式，可以确保不会被排除到 table以外，也会确保在加载vue.js的时候，table内部有内容

Components : {

‘ Tabel - content ‘ : {

Template : ‘

<tr>

<td>hello world</td>

</tr> ’

}

}

<tabel>

<tr is = ‘ tabel - content ’> <tr>

</tabel>

* 1. 组件的生命周期

设置属性 isShow = false

<input type = ‘ checkbox ’ v - model = ‘ isShow ’>

设置组件

Vue . component ( ‘ first - com ’ , {

Template : ‘ <h1> 组件创建 </h1> ’

} )

添加组件

<first - com v - if = ‘ isShow ’ ></first - com>

这里的v - if 和 v - show 的区别

V - if 当值为false的时候，元素会被删除 值为true，元素会被重新添加

V - show 元素不会被删除以及添加，知识给元素增加display : none ,让其隐藏或显示

设置方法

Vue . component ( ‘ first - com ’ , {

Template : ‘ <h1> 组件创建 </h1> ’ ，

beforeCreate ( ) { console . log ( ‘ 组件创建之前调用这个方法 ’ ) } ，

Created ( ) { console . log ( ‘ 组件创建以后调用 ’ ) } ，

Mounted ( ) { console . log ( ‘ 组件渲染数据的时候调用，经常把数据放在这个方法 ’ ) }

beforeDestroy ( ) { console . log ( ‘ 当组件将要被摧毁的时候调用 ’ ) } ，

Destroyed ( ) { console . log ( ‘ 组件被删除的时候调用，可以在这个方法里面做一些特 殊操作，如删除cookie ， 浏览痕迹等 ’ ) }

} )

Updated ( ) { console . log ( ‘ 组件的值更新了 ’ ) }

Watch 监听属性， 一旦属性值发生改变 watch里面监听到

Updated 监听事件 一旦组件的值发生改变 updated立马监听到

* 1. 组件内部使用属性

data必须为函数，函数有一个返回值，返回一个对象，所有声明的属性，都要写在对象中

Vue . component ( ‘ my - btn ’ , {

Template : ‘ <button @click = ‘ btnClick ( ) ’> { { num } } </button> ’ ,

Data : function ( ) {

Return { num = 0 } , } // 给每一个被创建的组件，单独的一个num属性

Methods : {

BtnClick : function ( ) { this . num ++ }

}

} )

* 1. 组件传值

1. data : { money : 500 }
2. <first - com :名字= ’ money ’ > </first - com>
3. Props : [ ‘ 名字 ’ ]

* 1. porps验证

Porps : {

Name : String ,

Age : Number,

Sex : [ Boolean , String ]

}

* 1. 子组件传值父组件

1. data : page
2. <first - com : myPage = ‘ page ’>< / first - com >
3. Props:[ mypage : Number ]
4. 子Data : function ( ) { return { newPage : this . mypage } }
5. 子 < span> {{ newpage }} < / span >
6. 子点击事件 preClick : function ( ) {

This. $emit ( ‘ pagechange ’ , - - this . newPage ) }

1. 父 < first - com @pagechange = ‘ refreshPage ’ >< / first - com >
2. 父 methods : { refreshPage : function ( value ) {
3. This . page = value } }
   1. 非父子组件传值
4. 创建组件1 - 设置 - 绑定 - 放主main
5. 创建组件2 - 设置 - 绑定 - 放主main
6. Watch监听组件一
7. 发生改变传代理 xxx . $emit ( ‘ pass ’ , this . firstMessage )
8. 创建代理 var xxx = new Vue()
9. 代理接收事件 mounted ( ) { xxx . $on ( ‘ pass ’ , this . getValue ) }
10. 接收后调动 methods:{ getValue : function ( value ) { } }
11. 方法内部绑定之间1 ，2