天若有情天亦老，月如无恨月长圆

Vue基础

methods中可传入方法 对应相应的生命周期会有相对的回调函数

el : '#id'

Vue指令

v-on 点击事件 <button v-on:click="btnClick">点我变色</button> 在method中定义一个btnclick事件

拓展： v-on：click="" 可以简写为 @click="" 并且 还有一些对应的修饰符

@click.stop 阻止冒泡

@click.prevent 阻止默认事件 如a标签的默认跳转

@keyup.enter="" 监听某个键帽值 例enter

@click.once="" 只执行一次

v-for 循环取出 后面要加操作代码 v-for="item in movies" {{item}}

v-for="(item,index) in movies" movies是一个数组 movies: ['海王','星际穿越','盗梦空间','大话西游'],

遍历对象： v-for="item in movies" 这样得到的是value

得到key value v-for="(value,key) in movies" <li v-for="(a,b) in obj">{{b}}-{{a}} </li>

//第三个参数可选 index a为value b为key

v-once 不可修改

v-html 解析html 例如：'<a href="http://www.baidu.com">百度一下</a>' 取出后：百度一下

v-text 相当于{{}} 一起用回覆盖{{}}

v-pre 展示原本的东西 <h2 v-pre>{{counter}}</h2> 结果： {{counter}}

v-cloak （斗篷） 解释：通常js代码是页面加载完后再加载js，如果用Vue处理，用户会看到原本的代码（{{xxx}}）后再渲染，此时如果给绑定的元素一个v-cloak，并且设置此元素display='none' 就会避免此效果。

原因：v-cloak在Vue加载完后会消失，所以display none消失。

v-bind <img v-bind:src="imgurl" alt=""> 绑定属性 可以简写为 ：

v-if 条件判断 v-else 通过给v-if添加判断 控制是否显示

<h2 v-if="isShow">{{message}}</h2>

<h2 v-else>false时显示我</h2>

<h3 v-if="score>=90">优秀</h3>

<h3 v-else-if="score>=80">良好</h3>

<h3 v-else-if="score>=60">及格</h3>

<h3 v-else="score<60">差</h3>

v-show true /false 给元素增加display none

v-model 数据的双向绑定

<input type="text" v-model="message"> //改变输入框的值。message也会改变

{{message}}

其实等同于 <input type="text" :value="message" @input="message = $event.target.value"> 监听输入并动态改变

实例：结合radio绑定可以初始化或者实现radio的互斥

对应的修饰符：

v-model.lazy="message" lazy修饰符 数据失去焦点或者回车时才会更新

v-model.number="age" 定义使用数字类型

v-model.trim="name" 拨出左右的空格

数组操作的响应式操作

movies: ['海王','星际穿越','盗梦空间','大话西游'],

this.movies.push('aqua man'); //push可以响应

this.movies.pop() //删除最后一个 可以响应式改变

this.movies.shift() //删除第一个 可以响应式改变

this.movies.unshift('shit') //第一个加 可以响应式改变

splice 作用：

1.删除元素：（start，个数）/第二个个数可选，不穿删除start到结尾所有的元素

splice(start,)

this.movies.splice(1,2) //索引1开始删除2个

2.替换元素 （start，个数，要替换的字符）

this.movies.splice(1,2,'dd','ss','op')

3.插入元素

this.movies.splice(1,0,'pp','z') //把第二个参数传为0，即可

sort()

reverse()

以上都是响应式

this.movies[0]='aqua man'; //页面不响应式改变

Vue.set(this.movies,0,'add') //set(要修改的对象，索引，修改值)

组件的使用

步骤： 1.const cpn=Vue.extends({

template：

`<div>

<h2>I am Title11111111111111</h2>

<h3>xxxxxxxxxxxxxx内容是我1xxxxxxx</h3>

</div>`

//创建模板

})

2.Vue.component({'name',cpn})

3.在元素中使用 <name></name>

简写： （即在注册时候创建，二合一）省略Vue.extend({})

Vue.component('name01',{

template: `<div>

<h2>I am Title11111111111111</h2>

<h3>xxxxxxxxxxxxxx内容是我1xxxxxxx</h3>

</div>`}) //直接在注册中创建 全局组件

components:{

name02: {

template: `<div>

<h2>I am Title11111111111111</h2>

<h3>xxxxxxxxxxxxxx内容是我1xxxxxxx</h3>

</div>`}) //局部注册中创建

}

}

template分离：组件模板的分离

1.将template中的代码写到 script中 对应id

<script type="text/x-template" id="temp">

<div>

<h2>I am Title分离分离</h2>

<h3>xxxxxxxxxxxxxx内容是我分离分离分离分离</h3>

</div>

</script>

2.将template中的代码写到 template标签中 对应id

<template id="temp">

<div>

<h2>I am Title分离分离template标签</h2>

<h3>xxxxxxxxxxxxxx内容是我分离分离分离分离</h3>

</div>

</template>

注意：

组件内部不能访问vue实例中的data数据

数据保存在组件内部 data必须为函数，并且有返回值 是一个对象

data(){

return {}

}

Vue.component('name03',{

template: '#temp',

data(){

return {

title:'我是标题'

}

}

})

最终版创建组件

<div id="app">

<cpn></cpn> //调用

</div>

<template id="temp"> //模板

<div >

<button v-for="item in categroies" @click="btnclick(item)">{{item.name}}</button>

</div>

</template>

const a = new Vue({

el:'#app',

data:{

},

components:{

cpn:{

template:'#temp'

}

},

methods:{

}

})

父传子：props 创建组件的时间添加props属性

cpn:{

template:'#temp'；

props：{

name：String, 如果props是对象表明属性类型即可 还可直接props为数组 props ：['propa','propb'] //到时候使用组件的时候传入就行 <cpn name="name" />

age:number

}

}

注意： props: {

// 1.类型限制

// cmovies: Array,

// cmessage: String,

// 2.提供一些默认值, 以及必传值

cmessage: {

type: String,

default: 'aaaaaaaa',

required: true

},

// 类型是对象或者数组时, 默认值必须是一个函数

cmovies: {

type: Array,

default() {

return []

}

}

子传父 $emit发射 后接收

父组件访问子组件的属性 父组件内使用this.$children[].属性名 （不常用）

this.$refs.aaa.属性名 （常用）

注意：其中aaa为子组件调用时加的名称 如：<cpn ref="aaa"></cpn>

子访问父 子组件内使用this.$parent.message this.$root

具名插槽

作用域插槽

创建脚手架2 vue init webpack my-project

创建脚手架3 vue create hello-world

git命令

git add. git commit -m 'shuoming ' git push

git push origin newbranch

git branch -a

覆盖本地代码：

git fetch --all //只是下载代码到本地，不进行合并操作

git reset --hard origin/master //把HEAD指向最新下载的版本 master分支名

npm run json:server

链接：

vue所有基础语法：https://blog.csdn.net/weixin\_43342105/article/details/106153672

vuex核心内容及重点细节总结：https://blog.csdn.net/weixin\_43342105/article/details/105703491

vue cli2 ，3详解：https://blog.csdn.net/weixin\_43342105/article/details/106248521

vue router全面详细知识点：https://blog.csdn.net/weixin\_43342105/article/details/106248594

axios框架核心知识：https://blog.csdn.net/weixin\_43342105/article/details/105740102

接口

http://152.136.185.210:8000/api/n3/recommend

http://152.136.185.210:8000/api/n3/home/data?type=pop&page=1