**vue基础**

1 vm.$mount('#root')

el的两种写法

        const v = new Vue({

            //el:'#root', //第一种写法

        })

        v.$mount('#root') //第二种写法

2 MVVM模型

         1. M：模型(Model) ：data中的数据

         2. V：视图(View) ：模板代码

        3. VM：视图模型(ViewModel)：Vue实例

            观察发现：

         1.data中所有的属性，最后都出现在了vm身上。

         2.vm身上所有的属性 及 Vue原型上所有属性，在Vue模板中都可以直接使用。

3 Object.defineProperty

       let obj = {x:100}; let obj2 = {y:200};

            Object.defineProperty(obj2,'x',{

                get(){

                    return obj.x

                },

                set(value){

                    obj.x = value

                }

            })

    通过Object.defineProperty()把data对象中所有属性添加到vm上。

    为每一个添加到vm上的属性，都指定一个getter/setter。

    在getter/setter内部去操作（读/写）data中对应的属性。

4 Vue中的事件修饰符：

    1.prevent：阻止默认事件（常用）；

    2.stop：阻止事件冒泡（常用）；

    3.once：事件只触发一次（常用）；

    4.capture：使用事件的捕获模式；

    5.self：只有event.target是当前操作的元素时才触发事件；

    6.passive：事件的默认行为立即执行，无需等待事件回调执行完毕；

5 Vue中常用的按键别名

            回车 => enter

            删除 => delete (捕获“删除”和“退格”键)

            退出 => esc

            空格 => space

            换行 => tab (特殊，必须配合keydown去使用)

            上 => up

            下 => down

            左 => left

            右 => right

系统修饰键（用法特殊）：ctrl、alt、shift、meta

            (1).配合keyup使用：按下修饰键的同时，再按下其他键，随后释放其他键

            (2).配合keydown使用：正常触发事件。

Vue.config.keyCodes.huiche = 13 //定义了一个别名按键

@keydown.huiche="showInfo"

6 computed:{

                fullName:{

                    get(){},

                    set(value){}

                }

            }

7 watch:{

    isHot:{

        deep:true,

        immediate:true,

        handler(newValue,oldValue){}

    }

}

8 绑定class样式

:class="mood" mood:'normal',

:class="classArr" classArr:['atguigu1','atguigu2','atguigu3'],

:class="classObj"

classObj:{

           atguigu1:false,

           atguigu2:false,

                }

9 绑定style样式

:style="styleObj"

styleObj:{

           fontSize: '40px',

           color:'red',

                }

:style="styleArr"

styleArr:[

          {

             fontSize: '40px',

             color:'blue',

              },

           {

                backgroundColor:'gray'

               }

             ]

10 条件渲染：v-if   v-show

            1.v-if

                        写法：

                                (1).v-if="表达式"

                                (2).v-else-if="表达式"

                                (3).v-else="表达式"

                        适用于：切换频率较低的场景。

                        特点：不展示的DOM元素直接被移除。

                        v-if v-else-if、v-else一起使用，结构不能被“打断”。

            2.v-show

                        写法：v-show="表达式"

                        适用于：切换频率较高的场景。

                        特点：不展示的DOM元素未被移除，仅仅是使用样式隐藏掉

            3.备注：使用v-if的时，元素可能无法获取到，而使用v-show可以获取到。

11 v-for指令:

        1.用于展示列表数据

        2.语法：v-for="(item, index) in xxx" :key="yyy" //key必须使用

        3.可遍历：数组、对象、字符串（用的很少）、指定次数（用的很少）

12 面试题：react、vue中的key有什么作用？（key的内部原理）

1. 虚拟DOM中key的作用：

        key是虚拟DOM对象的标识，Vue会根据【新数据】生成【新的虚拟DOM】,

        随后Vue进行【新虚拟DOM】与【旧虚拟DOM】的差异比较，比较规则如下：

2.对比规则：

        旧虚拟DOM中找到了与新虚拟DOM相同的key：

            ①.若虚拟DOM中内容没变, 直接使用之前的真实DOM！

            ②.若虚拟DOM中内容变了, 则生成新的真实DOM

        旧虚拟DOM中未找到与新虚拟DOM相同的key

            创建新的真实DOM，随后渲染到到页面。

3. 用index作为key可能会引发的问题：

            若对数据进行：逆序添加、逆序删除等破坏顺序操作:

            如果结构中还包含输入类的DOM：会产生错误DOM更新

13 模拟一个数据监测

let data = {

    name:'尚硅谷',

    address:'北京',

}

//创建一个监视的实例对象，用于监视data中属性的变化

const obs = new Observer(data)

//准备一个vm实例对象

let vm = {}

vm.\_data = data = obs

function Observer(obj){

    //汇总对象中所有的属性形成一个数组

    const keys = Object.keys(obj)

    keys.forEach((k)=>{

        Object.defineProperty(this,k,{

            get(){

                return obj[k]

            },

            set(val){

                console.log(`${k}被改了，我要去解析模板，生成虚拟DOM`)

                obj[k] = val

            }

        })

    })

}

14 Vue.set(this.student,'sex','男') Vue监视数据的原理

Vue监视数据的原理：

    1. vue会监视data中所有层次的数据。

    2. 如何监测对象中的数据？

        通过setter实现监视，且要在new Vue时就传入要监测的数据。

            (1).对象中后追加的属性，Vue默认不做响应式处理

            (2).如需给后添加的属性做响应式，请使用如下API：

                Vue.set(this.student,'sex','男')

                this.$set(this.student,'sex','男')

    3.在Vue修改数组中的某个元素一定要用如下方法：

        1.push()、pop()、shift()、unshift()、splice()、sort()、reverse()

        2.Vue.set() 或 vm.$set()

15 v-model的三个修饰符：

         lazy：失去焦点再收集数据

         number：输入字符串转为有效的数字

         trim：输入首尾空格过滤

16 Vue.filter(name,callback) 或 new Vue{filters:{}}

备注：

        1.过滤器也可以接收额外参数、多个过滤器也可以串联

        2.并没有改变原本的数据, 是产生新的对应的数据

{{time | timeFormater('YYYY\_MM\_DD') | mySlice}}

Vue.filter('mySlice',function(value){

            return value.slice(0,4)

        })

filters:{

           timeFormater(value,str='YYYY年MM月DD日 HH:mm:ss'){

                return dayjs(value).format(str)

                }

            }

17 自定义指令 Vue.directive directives

      //定义全局指令

         Vue.directive('fbind',{

            //指令与元素虚拟dom成功绑定时

            bind(element,binding){

                element.value = binding.value

            },

            //指令所在元素被插入页面时

            inserted(element,binding){

                element.focus()

            },

            //指令所在的模板被重新解析时

            update(element,binding){

                element.value = binding.value

            }

        })

  //定义局部指令

directives:{

                //1.指令与元素虚拟dom成功绑定时（一上来）。

                //3.指令所在的模板被重新解析时。

                big(element,binding){

                    element.innerText = binding.value \* 10

                },

                fbind:{

                    //指令与元素虚拟dom成功绑定时（一上来）

                    bind(element,binding){

                        element.value = binding.value

                    },

                    //指令所在元素被插入页面时

                    inserted(element,binding){

                        element.focus()

                    },

                    //指令所在的模板被重新解析时

                    update(element,binding){

                        element.value = binding.value

                    }

                }

            }

18 分析生命周期 mounted() {}

            beforeCreate() {}      created() {}

            beforeMount() {}       mounted() {}

            beforeUpdate() {}      updated() {}

            beforeDestroy() {}     destroyed() {}

19 Vue.component中使用组件

  Vue中使用组件的三大步骤：

            一、定义组件(创建组件)

                Vue.extend(options)

产生了VueComponent的构造函数

            二、注册组件

                1.局部注册：靠new Vue的时候传入components选项

                2.全局注册：靠Vue.component('组件名',组件)

            三、使用组件(写组件标签)

                <School></School>

相当于new VueComponent(options)

组件名字写法

<School></School>

<MySchool></MySchool>

<MySchool/>

定义组件(创建组件)可以简写，

因为Vue.component('组件名',组件)注册组件,内部包含了定义组件

        const school = Vue.extend(options)

        可简写为：const school = options

1.重要的关系：VueComponent.prototype.\_\_proto\_\_ === Vue.prototype

2.为什么要有这个关系：让组件实例对象（vc）可以访问到 Vue原型上的属性、方法。

**vue-cli**