***一、初识Vue***

1. 想让Vue工作就必须创建一个Vue实例，且要传入一个配置对象；
2. 容器里的代码依然符合html规范，只不过混入了一些特殊的Vue语法；
3. 容器里的代码被称为Vue模板。



1. Vue实例和容器一一对应；
2. 真实开发中只有一个Vue实例，并且会配合着组件一起使用；
3. {{xxx}}中的xxx要写js表达式，且xxx可以自动读取到data中所有属性；
4. 一旦data中的数据发生改变，那么页面中用到的该数据的地方也会自动更新；
5. 区分js表达式和js代码（语句）
   1. 表达式：一个表达式会产生一个值，可以放在任何一个需要值得地方；
   2. Js代码(语句)：if(){};for(){}
6. 开发版本得vue.js能在控制台提示一些详细得错误信息，生产版本不会。

***二、模板语法(两大类)***

1、插值语法

a） 功能：用于解析标签体内容；

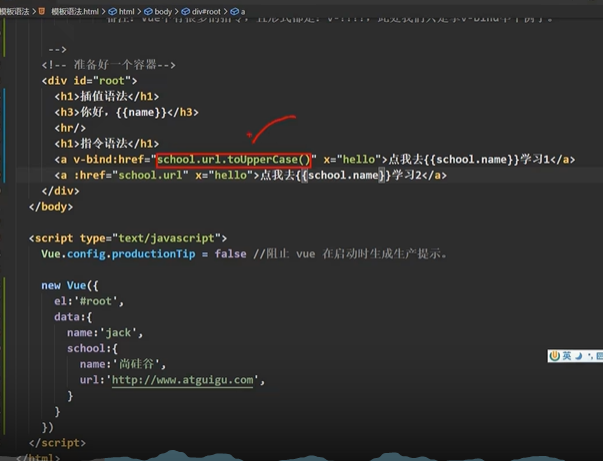
b） 写法：{{xxx}} xxx是js表达式，且可以直接读取到data中得所有属性

2、指令语法

a） 功能：用于解析标签(标签属性、标签体内容、绑定事件……)

b） 举例：v-bind:href=”xxx” 或简写 :href=”xxx” xxx同样要写js表达式

c) 备注：Vue中有很多得指令，且形式都是:v-???, 此处我们只是用v-bind举例



***三、数据绑定(两种方式)***

1、单向数据绑定(v-bind):数据只能从data流向页面

2、双向数据绑定(v-model):数据不仅能从data流向页面，还能从页面流向data。

a） 双向绑定一般都应用在表单类元素上（如：input、select等）

b） v-model:value可以简写为v-model,因为v-model默认收集得就是value值

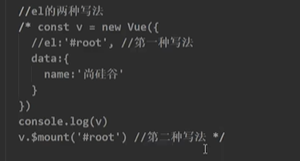


***四、el与data的两种用法***

1、el的两种写法

a） new Vue时候配置el属性

b） 先创建Vue实例，随后再通过v.$mount(‘#root’)指定el的值



2、data有两种写法

a） 对象式

b） 函数式

如何选择：正常页面用那哪种都可以，组件必须要用函数式



3、一个重要原则

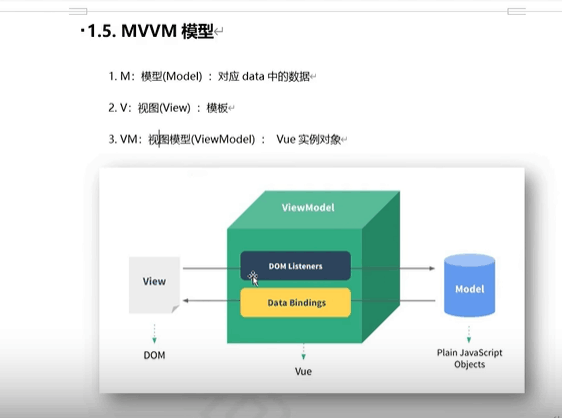
由Vue管理的函数，一定不要写箭头函数，一旦写了箭头函数，this指向的就不是Vue实例，而是window

***五、MVVM***

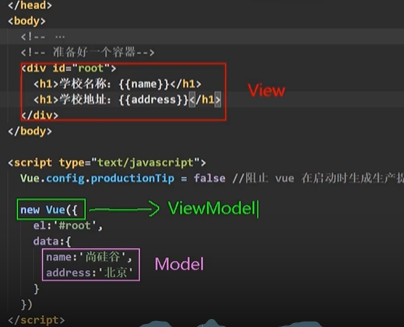
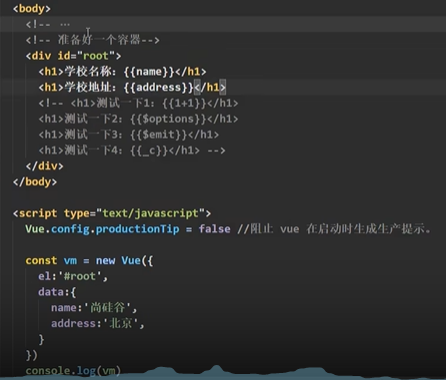
M：model

V：view

VM：viewmodel



1. data中所有属性，最后都出现在vm身上
2. vm身上所有的属性以及Vue原型上所有的属性，在Vue模板中都可以直接使用。



***六、数据代理***

1、Object.defineproperty(object,’属性’,{

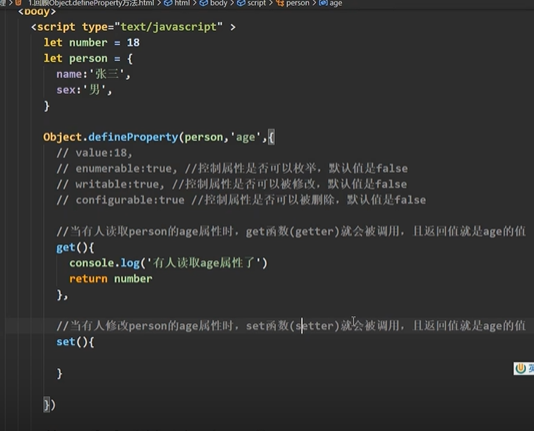
value:属性值

enumrable:true,//控制属性是否可以被枚举，Object.keys(object) 默认false:不可以

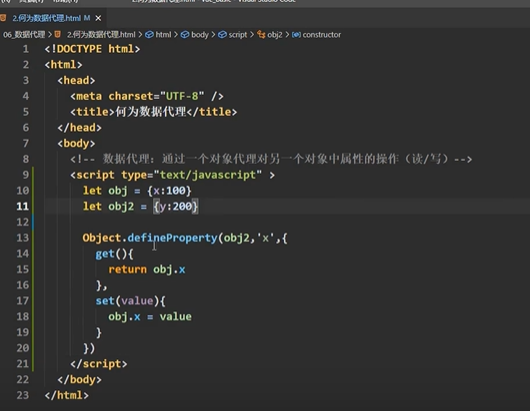
writable:true, //控制属性是否可以被修改，默认false:不可以

configurable:true,//控制属性是否可以被删除，默认false:不可以

});



2、理解数据代理 eg:



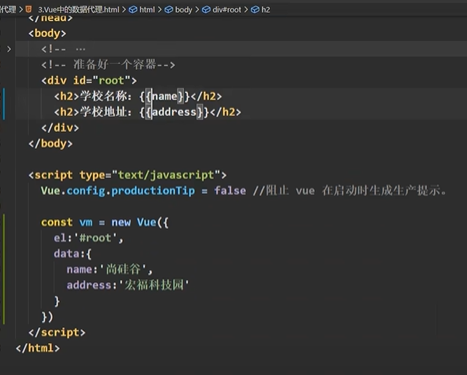
3、Vue中的数据代理

a） vue中的数据代理：通过vm对象代理data对象中属性的操作(读/写)

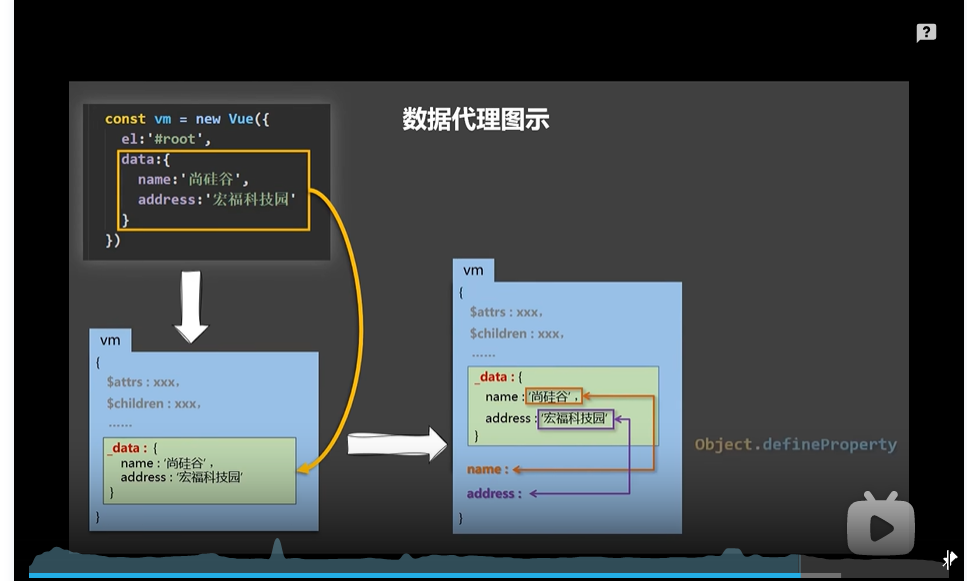
b） Vue中的数据代理的好处：更加方便操作data中的数据

c） 基本原理：1、通过Object.defineProperty()把data对象中的所有属性附加到vm上

2、为每一个添加到vm上的属性，都指定一个getter/setter。3、在getter/setter内部取操作(读/写)data中对应的属性



4、原理图



***七、事件处理***

1、事件的基本使用

a） 使用v-on:xxx或者 @xxx绑定事件，其中xxx是事件名

b） 事件的回调需要配置在methods对象中，最终会在vm上。

c） methods中的配置函数，不要用箭头函数！否则this指向的就是windows而不是vm了

d）methods中配置的函数，都是被VUE所管理的函数，this的指向是vm或者组件实例对象。

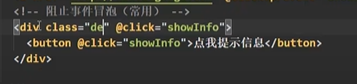
e）@click=“demo” 和 @click=”demo($event)”效果一致，但是后者可以传参

2、事件修饰符

a） prevent:阻止默认事件



b） stop:阻止事件冒泡



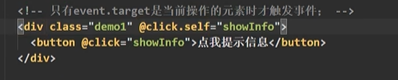
c） once:事件只触发一次



d） capture:使用事件的捕获模式



e） self:只有event.target是当前操作元素时才触发

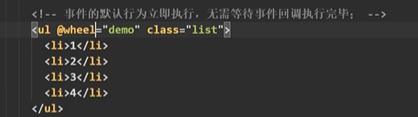


f） passive:事件的默认行为立即执行，无需等待事件回调执行完毕

滚动条



鼠标滚轮



1. 键盘事件

一、Vue中常用的按键别名

* + 1. 回车=> enter
    2. 删除=>delete(插获“删除”和“退格”健)
    3. 退出=>esc
    4. 空格=>space
    5. 换行=> tab(特殊,必须配合keydown去使用)
    6. 上=>up
    7. 下 => down
    8. 左=>left
    9. 右=>right

二、Vue未提供别名的按健，可以使用按键原始的key值去绑定，但注意要转为kebab-case(短横线命名)

三、系统修饰健（用法特殊）: ctrl、alt、 shift、 meta(windows键)

i. 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，事件才被触发

ii. 配合keydown使用:正常触发事件。

四、也可以使川keyCode去指定具休的按健（不推荐)

五、Vue.config . keyCodes.自定义健名=键码,可以去定制按键别名

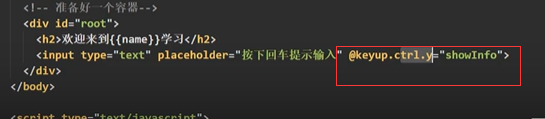
****

4、键盘事件使用小技巧

a） 修饰符可以联系写

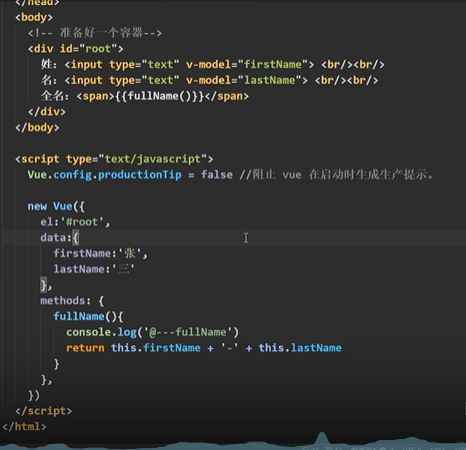


b） 修饰符可以固定按键

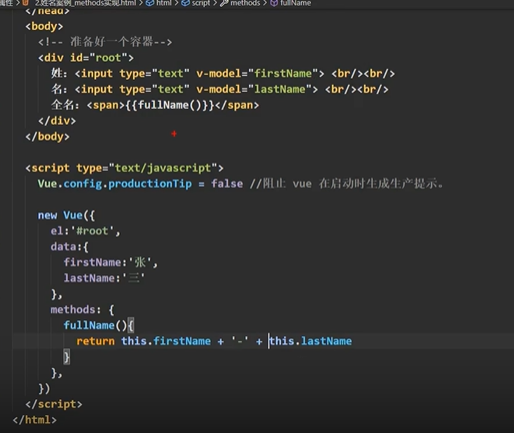


***八、计算属性***

1、插值语法



2、methods写法



3、计算属性写法

a） get有什么作用?当有人读取fullName时，get就会被调用，且返回值就作为fullName的值

b） get什么时候调用?1.初次读取fullName时。2.所依赖的数据发生变化时。

4、计算属性的定义

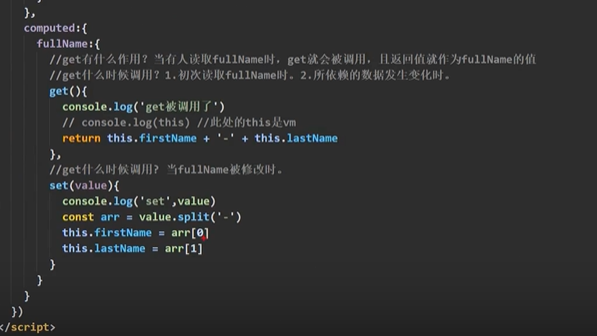
a） 定义:要用的属性不存在,要通过已有属性计算得来。

b） 原理:底层借助了objcet.defineproperty方法提供的getter和setter.

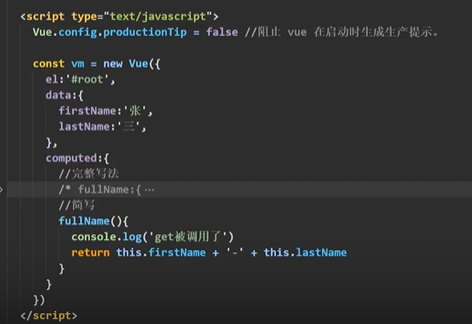
c） get的函数什么时候执行：初次读取时会执行一次。

d） 优势:与methods实现相比，内部有缓存机制（复用），效率更高，调试方便.5.备注:

f） 1.计算属性最终会出现在vm上，直接读取使用即可；2.如果计算属性要被修改，那必须写set函数去响应修改，且set中要引起计算时依赖的数据发生改变.



5、计算属性简写

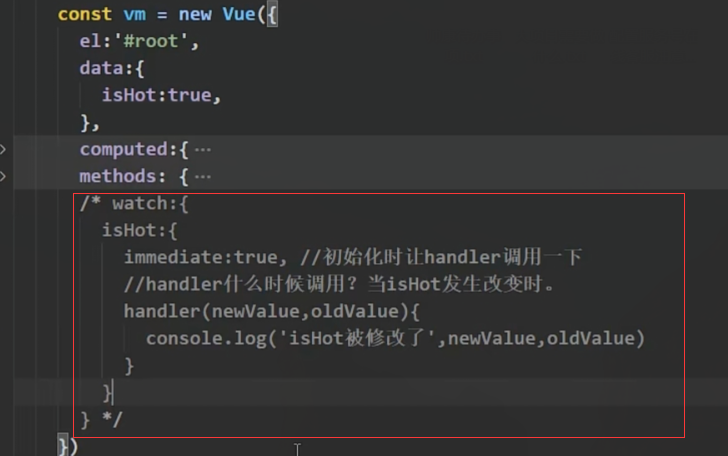


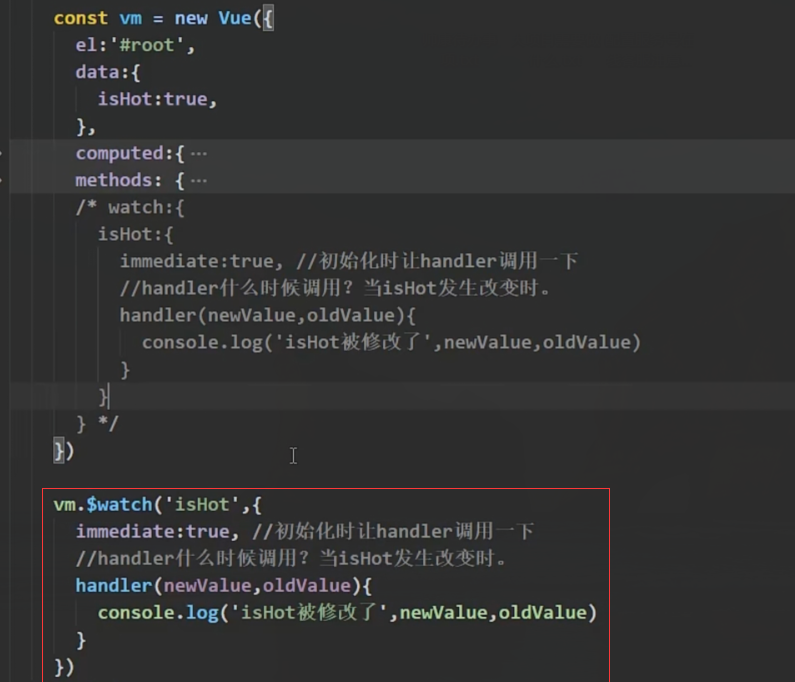
***九、监视属性***

1、watch的一些配置

immediate:true,//初始化时让handler调用一下

handler什么时候调用?当{被监听对象}发生改变时。





2、监视属性watch

当被监视的属性变化时，回调函数自动调用，进行相关操作

监视的属性必须存在，才能进行监视

监视的两种写法：1、new Vue()时传入watch配置；2、通过vm.$watch监视

3、深度监视

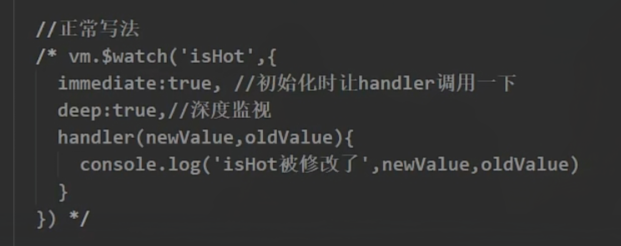
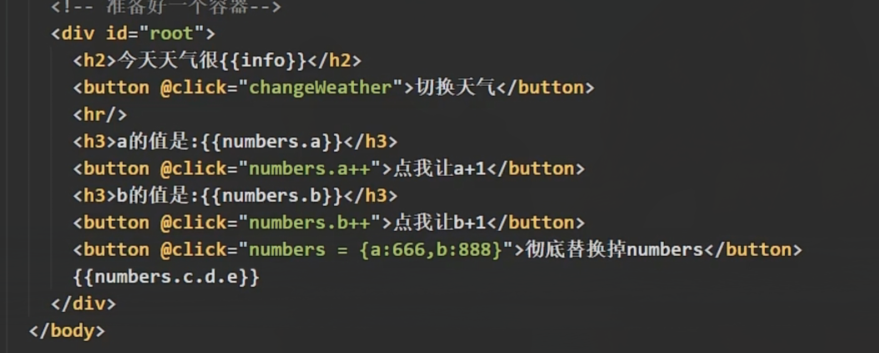
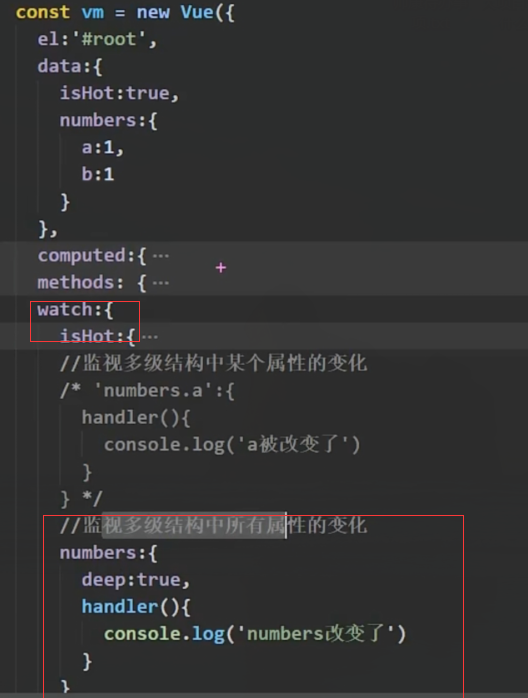
Vue中的watch默认不监测对象内部值的变化(一层)

配置deep:true可以监测对象内部值得改变

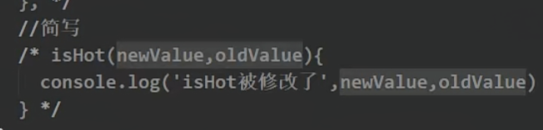
注：

Vue自身可以监测对内部值得改变，但是Vue提供的watch默认不可以

使用watch时根据数据的具体结构，决定是否采用深度监视。



4、监听属性的简写



***十、监视属性与计算属性的区别***

1、computed能完成的功能，watch都可以完成

2、watch能完成的功能，computed不一定能完成，例如：watch可以进行异步操作

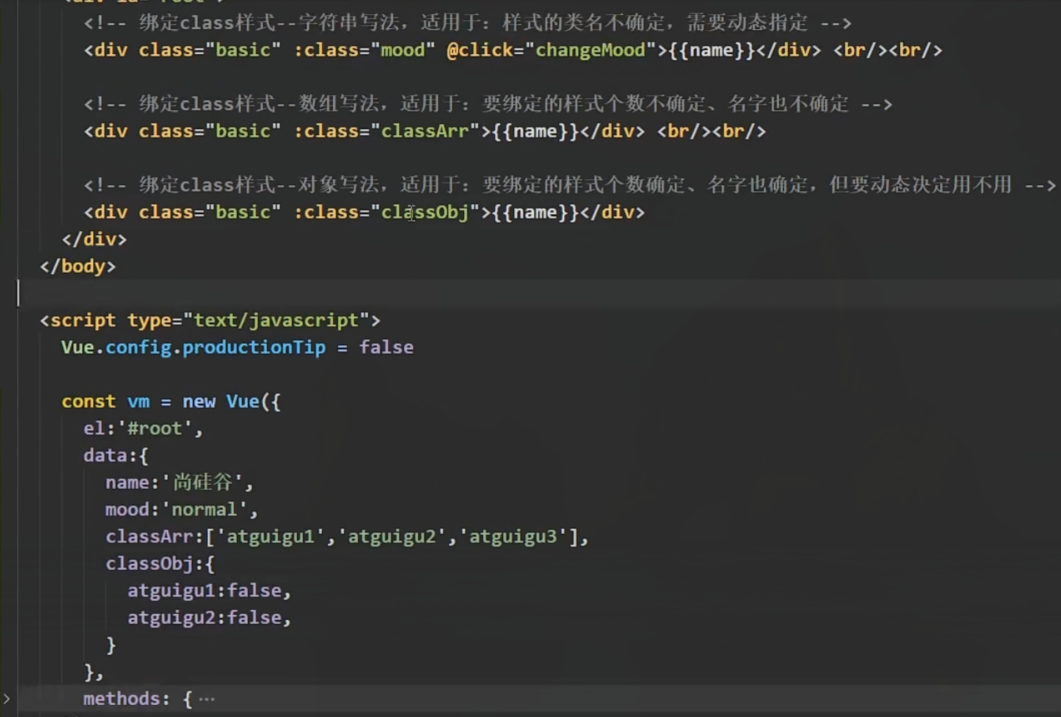
3、两个重要原则：

所有被Vue管理的函数，最好写成普通函数，这样this的指向才是vm或组件实例对象

所有不被Vue管理的函数(eg:定时器的回调函数、ajax的回调函数等)，最好写成箭头函数，这样this的指向才是vm或者组件实例对象。

***十一、绑定class***

1、三种样式绑定：1、字符串绑定、数组绑定、对象绑定

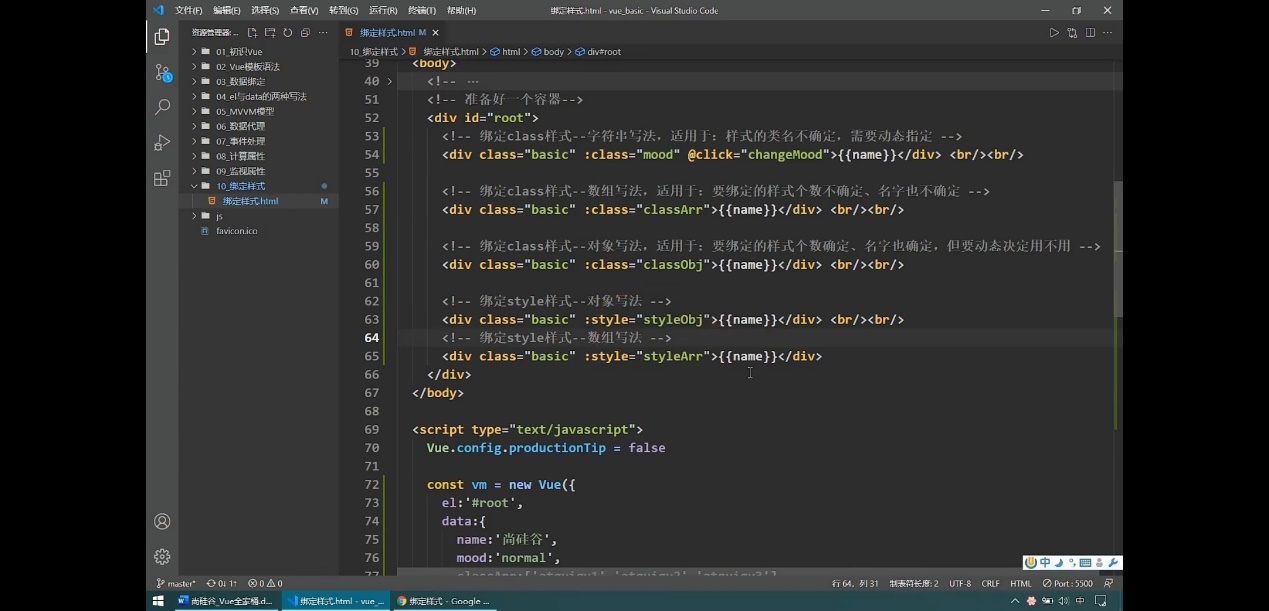


2、绑定class样式--字符串写法，适用于:样式的类名不确定，需要动态指定

3、绑定class样式--数组写法，适用于:要绑定的样式个数不确定、名字也不确定

4、绑定class样式--对象写法，适用于:要绑定的样式个数确定、名字也确定，但要动态决定用不用

***十二、绑定style***

1、

***十三、条件渲染 v-if v-show***

1、v-if：写法：v-if=”表达式” v-else-if=”表达式” v-else=”表达式”

适用于：切换频率较低得场景

特点：不展示的DOM元素直接被移除

注意：v-if可以和:v-else-if、v-else一起使用，但要求结构不能被“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一定可以获取到。

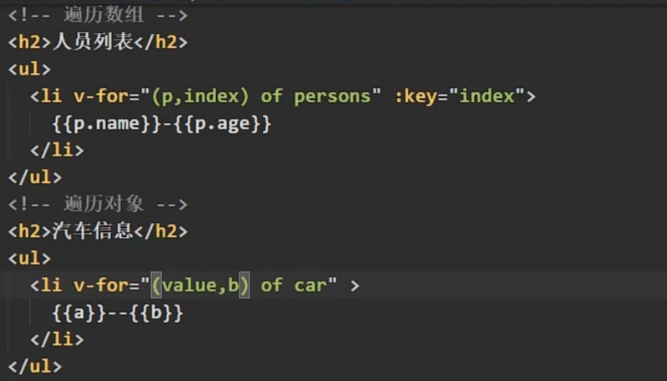
4、v-if可以和template配合使用 v-show不可以

***十四、列表渲染 v-for***

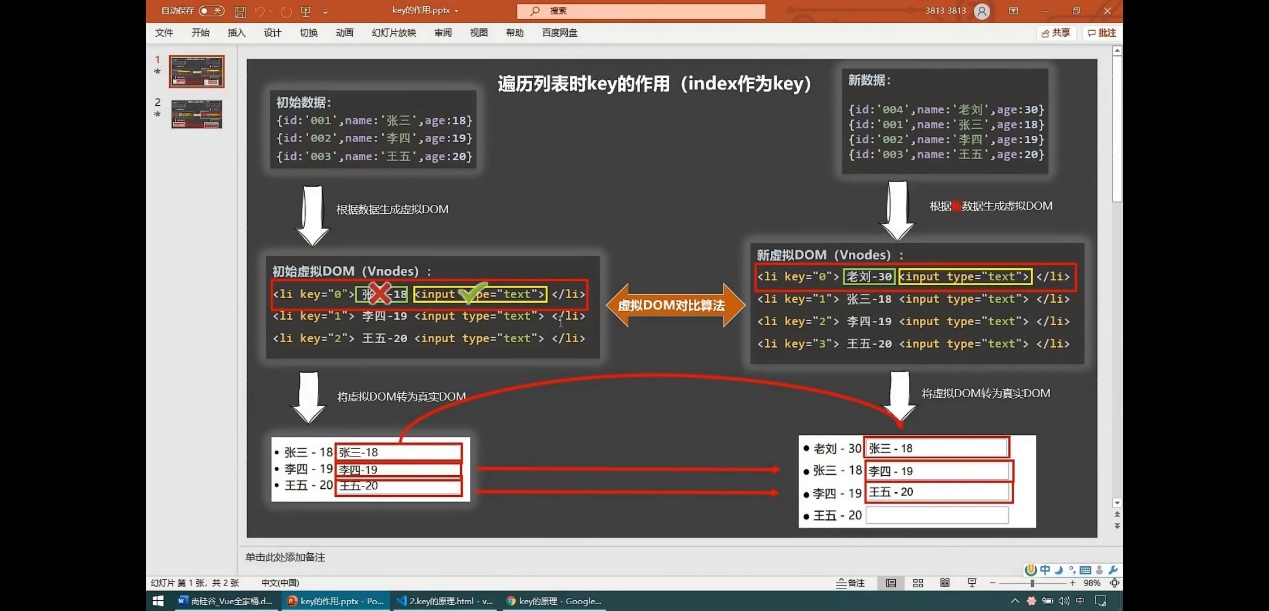
1、用于展示列表数据

2、语法:v-for="(item,index) in xxx" :key="yyy" v-for="(item,index) of xxx" :key="yyy"

3、遍历对象、遍历数组、遍历字符串（用的很少）、遍历指定次数（用的很少)



4、遍历时key的作用 （react和vue的key的作用）



虚拟DOM中key的作用：

key是虚拟DOM对象的标识，当状态中的数据发生变化时，Vue会根据【新数据】生成【新的虚拟DON】, 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:

对比规则：

1. 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，随后替换掉页面中之前的真实DOM.
2. 旧虚拟DOM中未找到与新虚拟DOM相同的key：
   1. 创建新的真实DOM,随后渲染到到页面。

用index作为key可能会引发的问题：

1. 若对数据进行:逆序添加、逆序删除等破坏顺序操作：会产生没有必要的真实DOM更新==>界面效果没问题,但效率低。
2. 如果结构中还包含输入类的DOM：会产生错误DOM更新==>界面有问题。

开发中如何选择key：

1. 最好使用每条数据的唯一标识作为key,比如id、手机号、身份证号、学号等唯一值。
2. 如果不存在对数据的逆序添加、逆序删除等破坏顺序操作，仅用于渲染列表用于展示，使用index作为key是没有问题的。

***十五、数据监测***

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?
   1. 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(target. propertyName/index. value)或vm.$set(target. propertyName/index. value)
3. 如何监测数组中的数据?
   1. 通过包寝数组更新元素的方法实现,本质就是做了两件事:
      1. 调用原生对应的方法对数组进行更新。
      2. 重新解析模板，进而更新页面。
4. 在Vue修改数组中的某个元素一定要用如下方法:
   1. 1.使用这些API:lush()、pop()、shift()、unshift()、splice()、sort()、reverse()
   2. Vue.set()或vm.$set()

5、特别注意:Vue.set() 和vm.$set()不能给vm或vm的根数据对象添加属性!!!

***十六、收集表单数据***

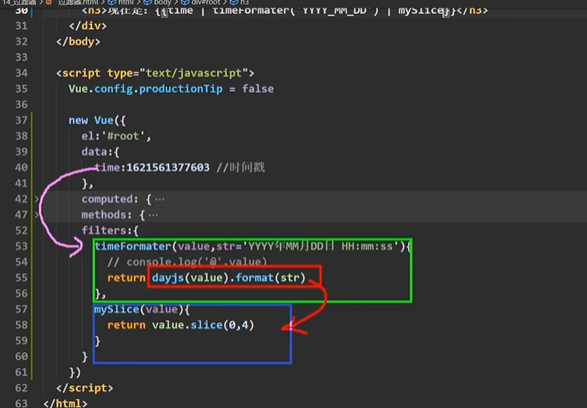
1. 若:<input tvpe=text"/>，则v-model收集的是value值,用户输入的就是value值。
2. 若:input type="radio"/>，则v-model收集的是value值,且要给标签配置value值。
3. 若:<input type="checkbox" />
   1. 没有配置inputvalue属性，那么收集的就是checked（勾选or未勾选，是布尔值）
   2. 配置input的value属性:
      1. v-model的初始值是非数组，那么收集的就是checked（勾选 or未勾选，是布尔值)
      2. v-model的初始值是数组，那么收集的的就是value组成的数组
4. 备注: v-model的三个修饰符:
   1. lazy:失去焦点再收集数据
   2. number:输入字符串转为有效的数字
   3. trim:输入首尾空格过滤

***十七、过滤器***

1. 网站：bootcdn.cn/dayjs/
2. 写法



1. 执行顺序



1. 全局过滤器用法



1. 定义总结
   1. 定义:对要显示的数据进行特定格式化后再显示（适用于一些简单逻辑的处理).
   2. 语法:
      1. 注册过滤器:vue.filter(name, callback）或new Vue{filters:{}}
      2. 使用过滤器:{{ xxx│过滤器名}} 或v-bind :属性=“xxx│过滤器名"

c） 备注:

i. 1.过滤器也可以接收额外参数、多个过滤器也可以串联。

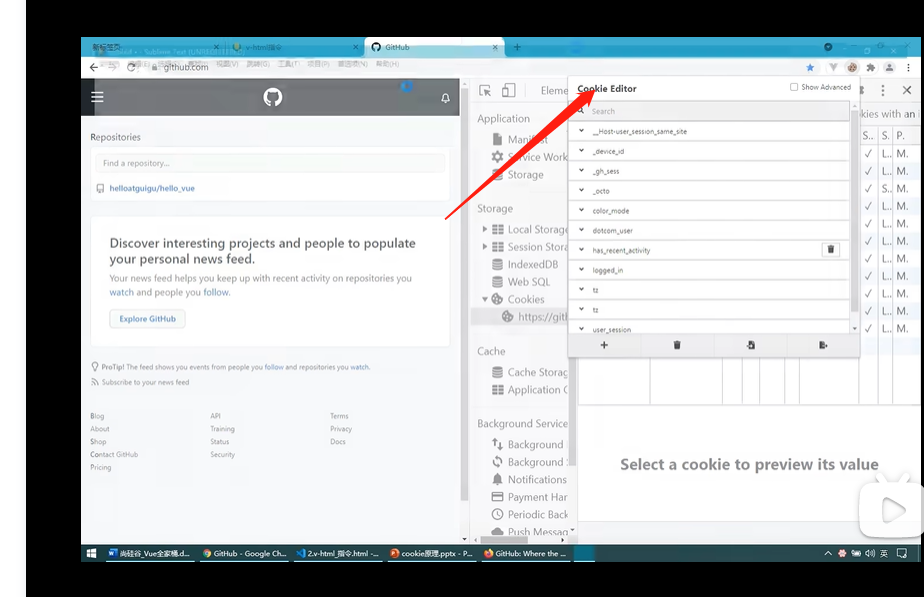
ii. 2.并没有改变原本的数据，是产生新的对应的数据。

***十八、指令***

1. 部分指令



1. Cookie Editor Google扩展程序



3、v-html **document.cookie**

a） 作用:向指定节点中渲染包含html结构的内容。

b） 与插值语法的区别:

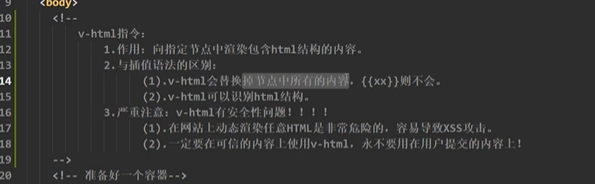
i. v-html会替换拉节点中所有的内容，{{xx}}则不会。

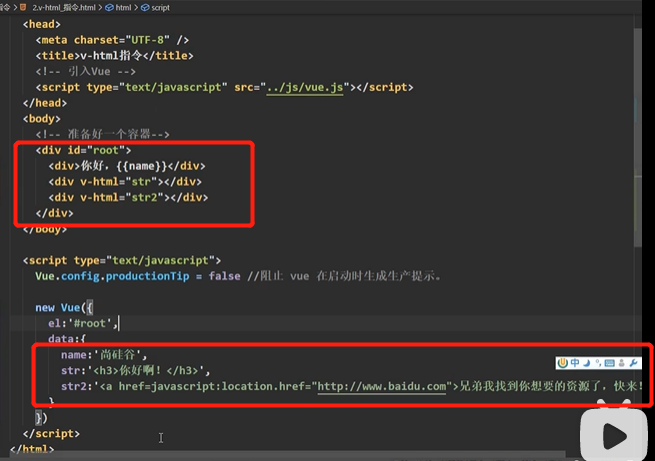
ii. (2) .v-html可以识别htm1结构。

c） 严重注意: v-html有安全性问题!!!!

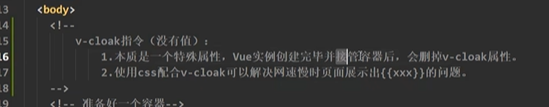
i. 在网站上动态演染任意HTML是非常危险的,容易导致XSS攻击.

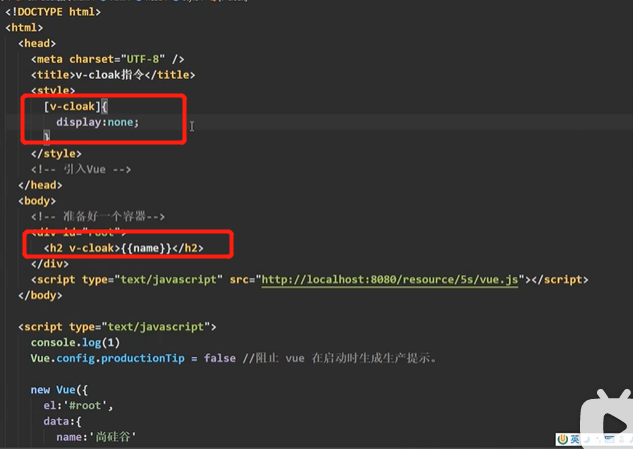
ii. 一定要在可信的内容上使用v-html，永不要用在用户提交的内容上!





4、v-clock

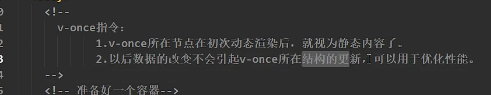


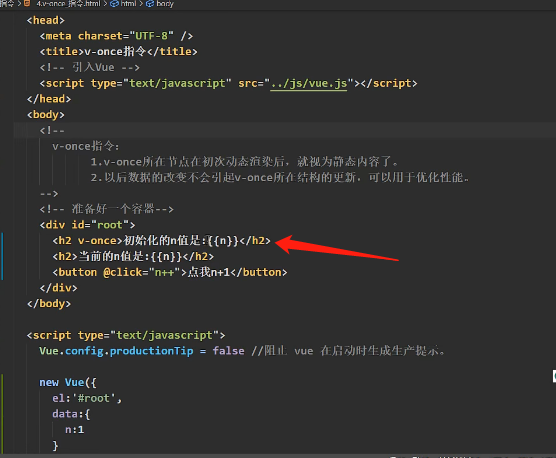


5、v-text

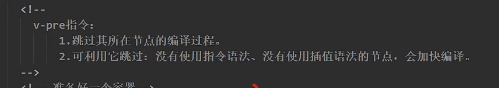


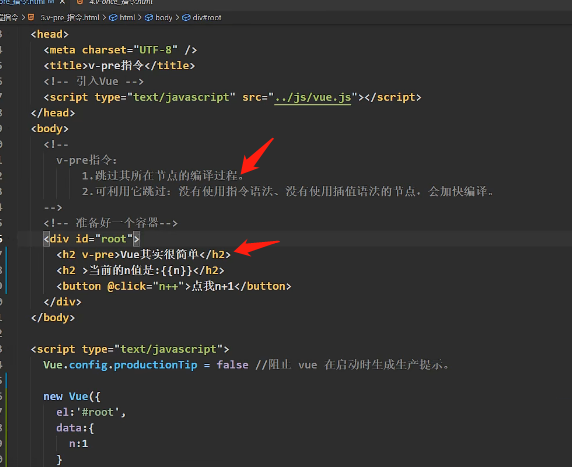
6、v-once





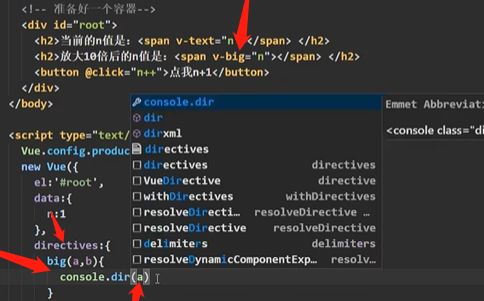
7、v-pre



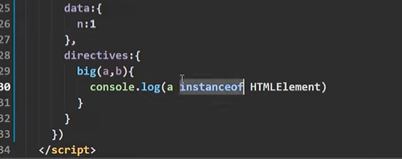


***十九、自定义指令***

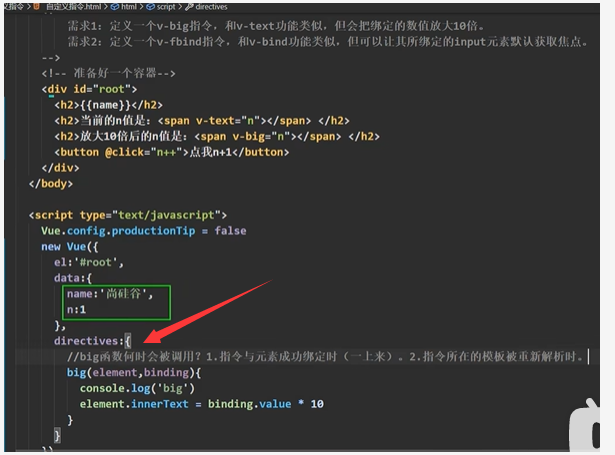
1、console.dir(a) 可以看到元素得全部属性



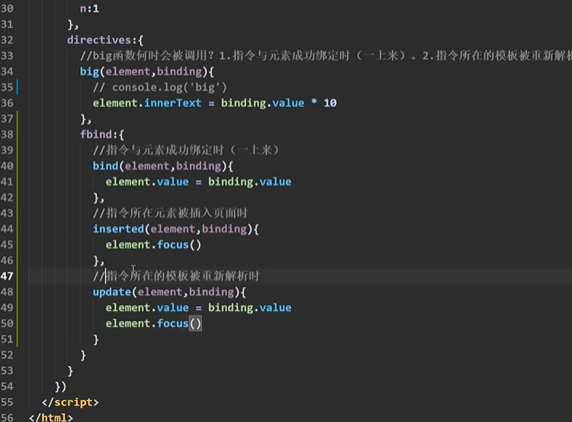
2、判断元素是否是真实dom元素



3、指令得使用：big函数何时会被调用?1.指令与元素成功绑定时（一上来)。2.指令所在的模板被重新解析时



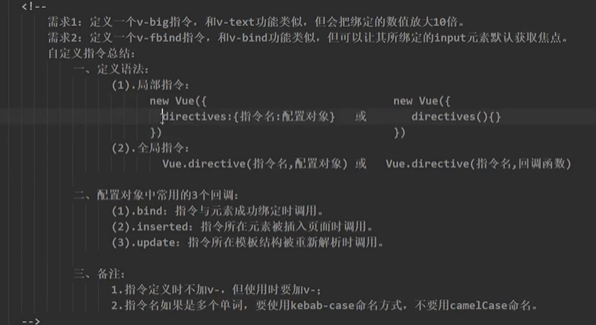
2、指令得两种使用模式



3、指令得全局使用



4、指令得总结



***二十、***

1、