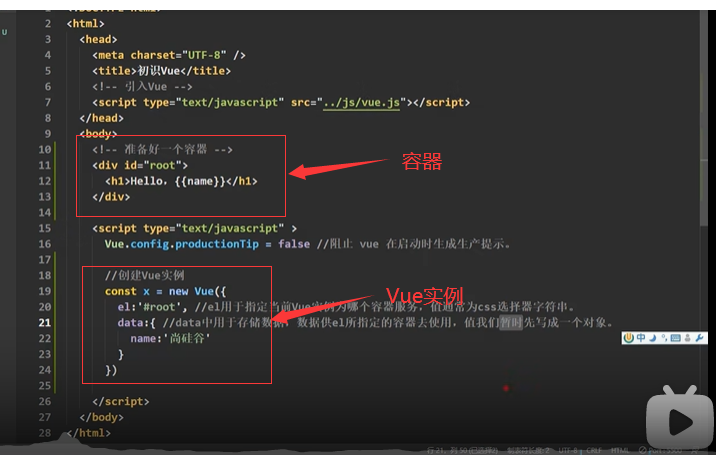
***一、初识Vue***

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. 真是开发中只有一个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；

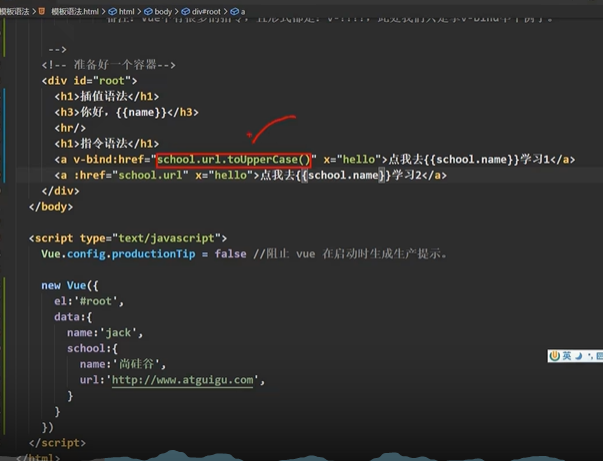
b） 写法：{{xxx}} xxx是js表达式，且可以直接读取到data中得所有属性

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

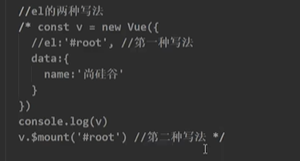


***四、el与data的两种用法***

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vm.$mount(‘#root’)指定el的值

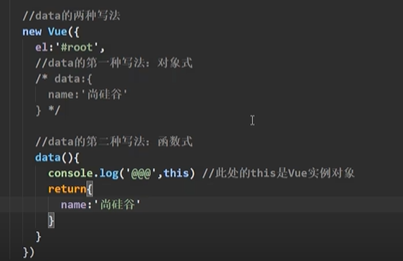


2、data有两种写法

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

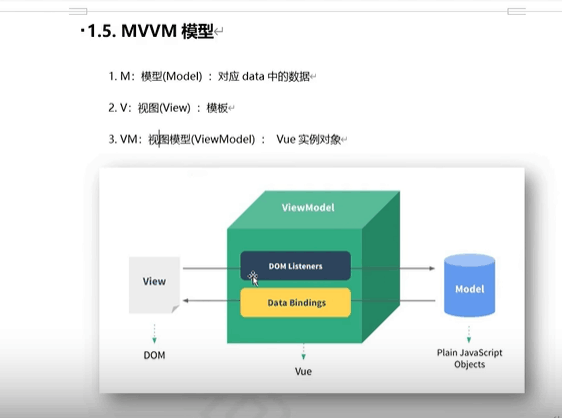
由Vue管理的函数，一定不要写箭头函数，一旦写了箭头函数，this指向的就不是Vue实例，而是window

***五、MVVM***

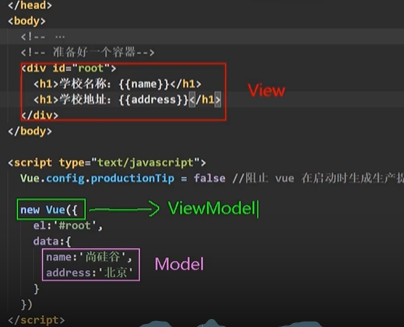
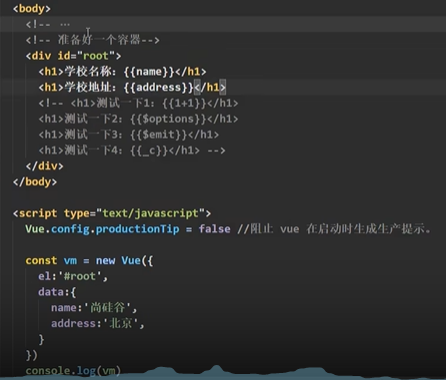
M：model

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’,{

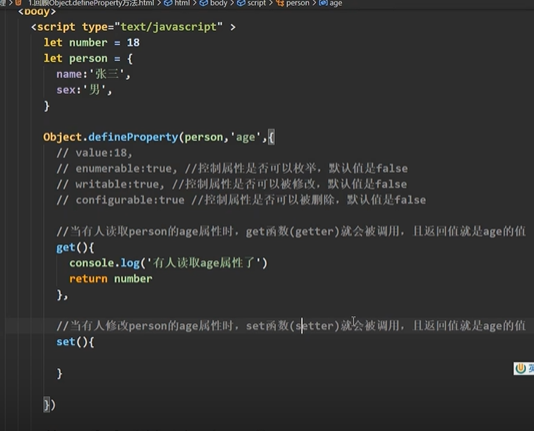
value:属性值

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

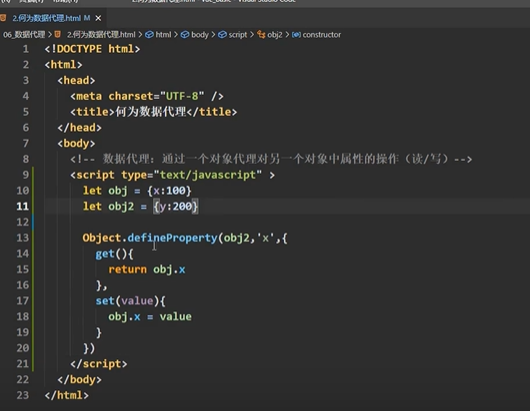
writable:true, //控制属性是否可以被修改，默认false:不可以

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:



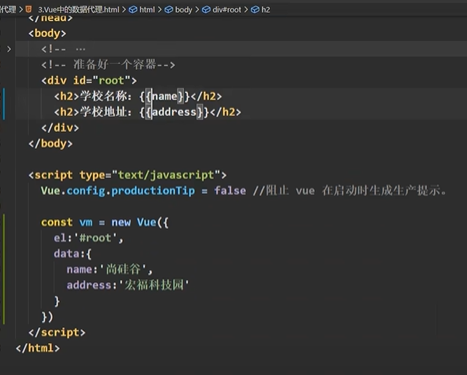
3、Vue中的数据代理

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)

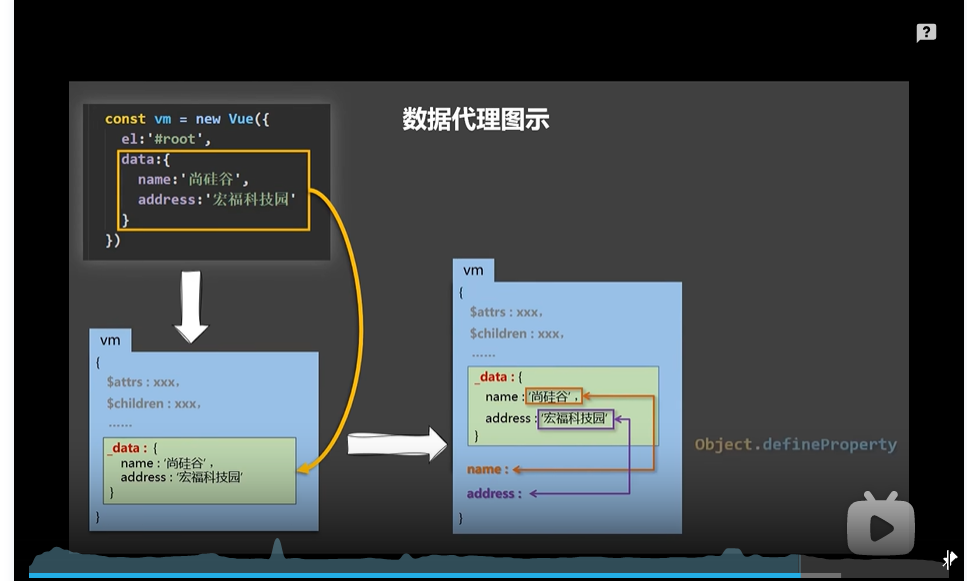
b） Vue中的数据代理的好处：更加方便操作data中的数据

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。

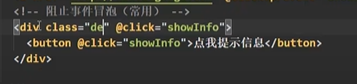
e）@click=“demo” 和 @click=”demo($event)”效果一致，但是后者可以传参

2、事件修饰符

a） prevent:阻止默认事件



b） stop:阻止事件冒泡



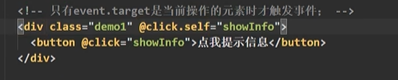
c） once:事件只触发一次



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

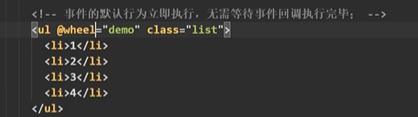


f） passive:事件的默认行为立即执行，无需等待事件回调执行完毕

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

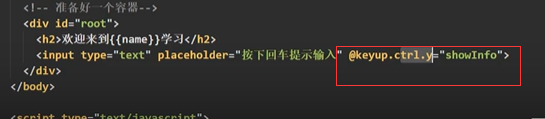
****

4、键盘事件使用小技巧

a） 修饰符可以联系写

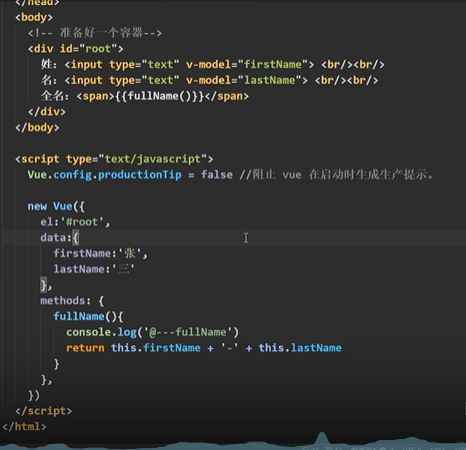


b） 修饰符可以固定按键

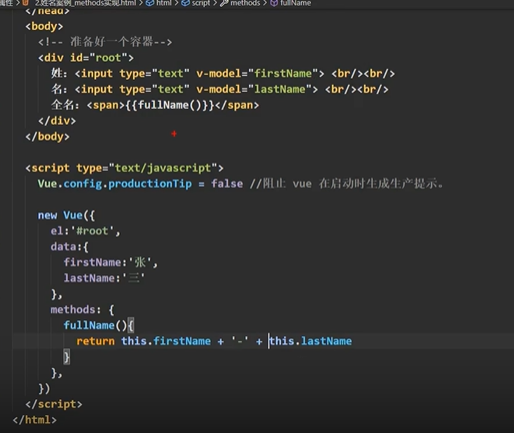


***八、计算属性***

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

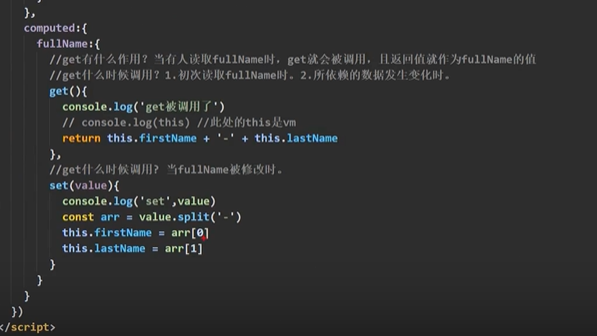
a） 定义:要用的属性不存在,要通过已有属性计算得来。

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

