1. 在vs code中按Ctrl+/，出现的不是注释，而是{% comment %}？

答: 禁用插件中的django template

2. 在vs code中用boot没有提示？

答: 安装bootstrap3 snippets 或bootstrap4 snippets插件

3. 如果调用cook(。。。)函数，但是没有用变量wan接，是不是就没有执行return？

答：函数只要调用，都会完整执行。return也是语句，当然也会被执行。这和外部用不用变量接无关。

就好像现实中，家人煮好面条了，问你吃还是不吃。你可以选择不吃，也可以选择拿碗吃。但是，不会因为你不吃面，家里人煮面的步骤就会少一步。这是不可能的。

必须把晦涩的程序和现实结合起来。因为程序也是人根据现实中的经验创造的出来的。都离不开现实。

4. vscode中，按tab键不会自动补全？

答：可以点菜单"编辑"->Emmet展开缩写，效果是一样的

也可能是右下角不是html的问题。

5. var fun=100; 是将地址值改为100，所以找不到函数才报错吗？

答？无论是普通的number数值，还是一个对象的地址值，对程序来说，都是无差别的一个数据。只要是一个数据就可以保存在变量中

fun=100只是用一个整数100数据，替换了原来保存在变量中的地址数据。

报错，是因为，语法规定，如果fun中保存的是一个数字，则数字后是不能跟()，像函数一样调用的。

fun=100替换后，函数对象的地址是不会改变的，一个对象的地址，一旦创建，就不会改变了。就像现实中的手机号。

6. 为什么声明两个同名函数，函数定义不同，也会覆盖？

答：js中，函数名其实就是一个普通的变量。函数体其实是一个对象。函数名中保存着函数对象的地址值。函数名变量通过函数对象的地址值，找到函数对象。就像现实中打电话找人一样。

声明两次同名函数，意味着，先创建一个函数名变量，然后将第一个函数对象的地址保存到函数名变量中。此时，函数名变量暂时通过第一个函数对象的地址值指向第一个函数对象。

第二次声明同名函数时，不会重复创建函数名变量，而是将新函数对象的地址，覆盖到函数名变量中现有旧函数对象的地址。就像手机通讯录里，将一条通讯录中的地址换成另一个人的手机号。从此，函数名变量，就和旧函数再无关系。而是用新函数对象的地址，引用着新函数对象。

7. var a=10

function foo(){

console.log(a); //10

a=20;

}

foo();  
 console.log(a) //20

答：10和20, 因为程序中，自始至终只有一个变量a，无论函数内，还是函数外，用的都是一个全局变量a。对a做任何修改，都是修改的一处。都会影响后续操作a的值。

如果换成:

var a=10

function foo(){

console.log(a); //undefined

var a=20;

}

foo();  
 console.log(a) //10

结果变为undefined和 10

首先：其实程序中不止一台hoist起重机。其实每个函数内的顶部，都各自带一个起重机。

在调用函数时，都会先再函数内声明提前，再执行。上面程序会变为:

var a=10

function foo(){

//hoist

// |

// |

// J

var a; //undefined

console.log(a); //undefined

a=20; //这里因为函数中有局部变量a了

//优先使用局部变量，所以，不会影响全局。

//且局部变量，在函数调用后，就释放了。

}

foo();  
 console.log(a) //10

//所以，除了函数，全局a还是10

8. console.log(fun()); //2 undefined，为啥输出带一个undefined

答：因为函数fun(){}内部没有return，就没有返回值，返回到函数外部。所以，凡是没有return的函数，都不能直接打印执行结果，打印输出的都是undefined。只有函数内部有return的函数，才能直接输出函数的调用结果。

9. var fun=100;  
 console.log(fun()) 结果为什么不是100？

答：程序有严格的语法规定。如果想使用变量中的值，就不能加()。如果想调用变量引用的函数，执行一段任务逻辑，就必须加()。

所以写程序前必须先要知道变量中存的是什么，然后再想想打算怎么用这个变量里的东西。

如果变量中引用的是一个函数，想调用函数，就必须fun()

如果变量中保存的是一个值，想输出值，就不能加()。

这里，如果想输出变量的值，就应该console.log(fun)

如果想输出函数调用后执行的结果，就要console.log(fun())

执行时，先调用fun()，如果有返回值，会放到console.log()中输出。

如果没有返回值，console.log()没有接到结果，就输出undefined。

10. 函数和方法的差别?

答: 函数和方法，本质都是function，也就是都是函数

严格来说:

不属于任何对象的独立的function称为函数

比如: parseInt() parseFloat() isNaN()

保存在对象内的函数，称为方法，也称为method

比如: arr.sort() reg.test() str.toUpperCase()

数组的下标和length，不是方法，也不是函数，而是属性。属性不做事儿，只存值。

11. ajax四步能不能包装在匿名函数中？

答：凡是可能反复使用的函数，就不能用匿名函数！

比如ajax四步，将来一定会封装在一个叫ajax的有名字的函数中被反复调用。

12. (function(){

function add(){

...

}

})()  
 </script>  
 <button onclick="add()">

为什么绑定不上事件处理函数？

答：function add(){}已经不是全局函数了，而是局部函数。

不能直接用onclick=""方式找到。

必须在匿名函数内，通过DOM查找到元素，动态绑定。

今后，不会用onclick="js语句"这种方式。可以忘掉了。

12. do{

var reg=/明明|静静/g;

var arr=reg.exec(str);

console.log(arr)

}while(...);

永远只能找第一个。

但是:

var reg=/明明|静静/g;

do{

var arr=reg.exec(str);

console.log(arr)

}while(...);

就可以自动向后找所有

答: /正则/是创建一个正则表达式对象的意思。

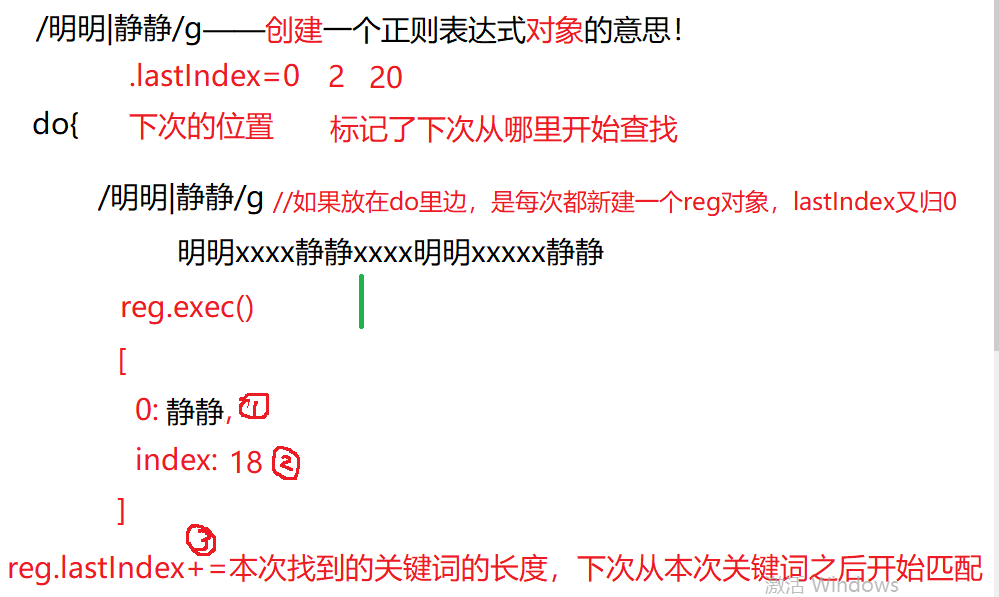
/正则/放在do外部，只创建一个，反复使用

/正则/放在do内部，每循环一次，都创建新的reg对象

又因为exec之所以能向后找，是因为reg对象上有一个lastIndex属性指定累加下次开始的位置。

如果只创建一个reg对象，则reg.lastIndex可反复累加。而如果每次创建新的reg对象，都会导致新reg.lastIndex重新回到0.

所以，放在外边，可自动向后找。放在内部，只能不断找第一个关键词。



13. 如果arguments中参数个数相同，但是参数内容不同，怎么区分？比如: str.replace(/正则/g,"新值")和str.replace(/正则/g,function(keyword){ return 加工后新值; })

答: 如果.length相同，区分不出来

可以用typeof判断arguments[i]的值的类型

replace中，如果第二个值是字符串，则直接替换回字符串中

replace中，如果第二个值是函数，则先调用函数，再将函数的返回值替换回字符串中。

14. function cook(dan,cai,mian){ ... }如果只是部分参数不确定，而且参数值赋值给参数时，又要求对应关系。怎么解决？

比如: cook( , 黄瓜, ) 只缺西红柿，换成黄瓜，别的都有

cook( 火腿肠, , ) 只缺鸡蛋，换成火腿肠，别的都有

cook( , , 方便面) 只缺面条，用方便面代替，别的都有

答: 以上传参方式，都会报语法错误！因为调用函数时的实参值列表中间，不允许出现空值

当遇到多个参数不确定有没有，又要求对应关系时，只能用对象语法解决！

2步:

1. 定义函数时:

function 函数(一个形参变量obj){

//先判断obj对象中包含哪些属性，不包含哪些属性。缺少的属性用默认值代替。

//函数执行过程中，都从对象里，取实参值使用！

}

但是，我们规定，将来调用时，所有实参值都要放在一个对象中传入。

2. 调用函数时:

函数名({属性1: 实参值1, 属性2:实参值2, ...})

优点: 任意参数都可以缺少！都不会报错！

function cook(obj){

/\*obj{

dan:"火腿肠",

cai:undefined,

mian:undefined

}\*/

//如果一个形参变量没有传值，则给一个默认值顶替。

if(obj.dan===undefined){ obj.dan="鸡蛋"}

if(obj.cai===undefined){ obj.cai="西红柿"}

if(obj.mian===undefined){ obj.mian="面"}

console.log(`炒${obj.dan}`)

console.log(`炒${obj.cai}`)

console.log(`炒${obj.cai}和${obj.dan}`)

console.log(`煮${obj.mian}`)

console.log(`把${obj.mian}和${obj.cai}${obj.dan}拌在一起`)

return `热腾腾的${obj.cai}${obj.dan}${obj.mian}`;

}

//周一:

var wan=cook({}); console.log(`获得一碗:${wan}`);

//周二:

wan=cook({dan:"辣条",cai:"黄瓜",mian:"米线"}); console.log(`获得一碗:${wan}`);

//周三:

wan=cook({dan:"火腿肠"})

// ↓ undefined undefined

// cook(dan, cai, mian)

console.log(wan);

//周四:

wan=cook({cai:"黄瓜"})

console.log(wan);

//周五:

wan=cook({cai:"榴莲",mian:"米饭"})

console.log(wan);

15. do{

var reg=/正则/g; //死循环

var arr=reg.exec(str);  
 if(arr!=null){  
 输出arr  
 }else{

break;

}

}while(true);

答: exec之所以可以向后找下一个，是因为修改了正则表达式对象reg中的一个属性lastIndex。

但是，如果把var reg=/正则/g。放在循环内，意思就是每循环一次，都重新创建一次新的正则表达式对象。之前辛辛苦苦修改的lastIndex属性丢了，新的reg的lastIndex归0

16. 看着明明内容相同的两个对象/数组，用==作比较，却返回false？

var a=/\d{6}/;

var b=/\d{6}/;

a==b

false

[]==[]

false

var a={x:1};

var b={x:1};

a==b

false

答: 因为// [] {} 其实都是new RegExp(), new Array(), new Object()的简写。虽然看不见new，但是都有new的作用！都是创建一个新对象的意思。

== 在比较两个对象时，既不比较内容，又不做任何转换，而是比较两个对象的内存地址值。只要两次new出的两个对象，即使内容看着相同，地址肯定是不一样的。所以都返回false

复习思维导图: 运算符和表达式->所有隐式转换

17. 为什么var reg=/1[3-9]\d{9}/就正确，var reg=new RegExp("1[3-9]\d{9}")，就不正确？

答: 正则中的\d，放在js的""中，js误以为\d是自己的转义字符，冲突了。js会自作主张去掉\或\d，导致正则表达式早就不是自己想要的。所以，如果用""创建正则表达式，其中所有\d都必须换成\\d，意思是不让js随便改动前一个\后的\，保持\d原样。

18. 函数的形参不需要声明是语法规定吗？

答: 是的，函数的形参变量虽然没有写var，但是都等效于自动var。所以，形参等效于在函数内var出的局部变量。

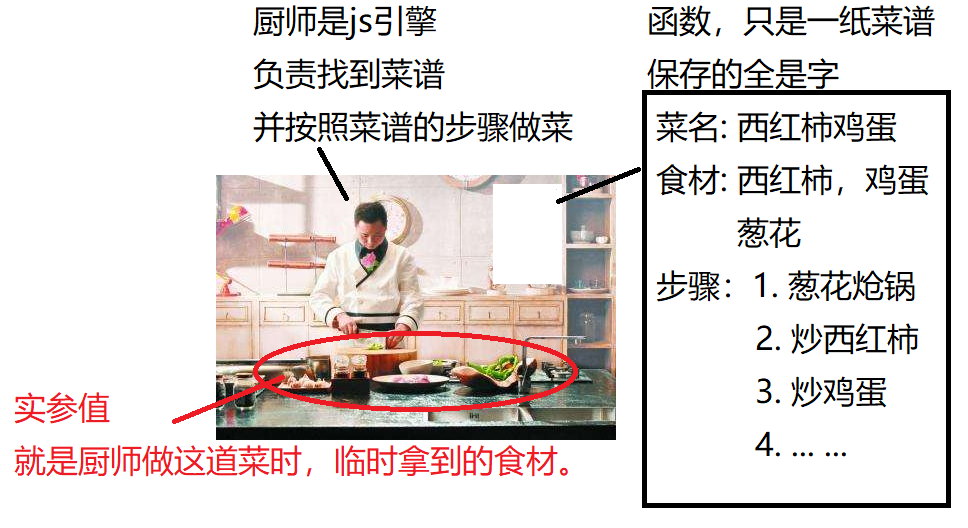
19. 第一次调用函数传入的参数赋值给形参，第二次调用函数不传入形参，形参就没有值，是函数内部将形参的赋值给处理掉了吗？还是形参会隐式覆盖？

答: 函数其实内部只存步骤，不存参数值

调用函数，其实是派js引擎，在内存中，找到函数，让js引擎按照函数内的步骤执行代码。

就好像，函数只是一张记录做菜步骤的菜谱。菜谱自己不能做菜，需要有厨师按照菜谱的步骤执行操作，才能做出菜来。js引擎，就好像现实中的厨师。

而每次调用函数时，传入的实参值，其实并没有保存在函数内部，而是js引擎将他们临时保存在一个地方。等效于厨师今天做这一道菜时的食材。食材并没有保存在菜谱中，而是保存在厨师做菜时临时的案板上。



一个常识，这道菜做完了，剩下的鸡蛋皮，菜叶等下脚料，默认厨师是不会留着的，就倒进垃圾桶不要了。

所以，js引擎在执行完一个函数后，也会将这些临时使用的实参值，释放掉。但是函数定义(菜谱)仍然贴在内存中，可以被js引擎反复使用。

下次，调用函数时，如果没有传入实参，就等于厨师要做饭，你没提供食材，巧妇难为无米之炊，自然就没有值可用。

20. 引用类型都包含哪些:

答: 除了五种原始类型，都是引用类型

21. []-[], []+[] 分别输出多少

答: - 法计算永远试图将左右两边的值都转为数字，因为只有数字可以做减法。所以，自动调用Number()

Number([]) = 0 记住。

[]-[]等于0, []-1等于-1

+ 法则完全不同。两个对象相加，会自动调用两个对象的toString方法，将两个对象转为字符串，然后执行字符串拼接。

[].toStirng() = ""，因为数组没有值，所以转为空字符串

[]+[]，等于""+""，返回""

[1,2,3]+[4,5,6], 等于"1,2,3"+"4,5,6" 结果:

"1,2,34,5,6"

22. function fun(a){  
 var a=10;  
 ...  
 }  
 var a和形参a是什么关系？

答: 调用函数时，先将形参变量a创建为一个局部变量保存在函数的临时函数作用域对象中。然后var a发现函数作用域对象中已经有局部变量a了，所以var a什么也不做，不会覆盖。

但是a=10，会覆盖形参变量原先接住的传入的实参值。是等号覆盖了a变量中的值，而不是var覆盖了形参变量。

23. 明明验证密码强度的正则写对了: ^(?![A-Za-z]+$)(?![a-z0-9]+$)[A-Za-z0-9]{6,}$，但是，在猫头鹰里却匹配不了密码: Password1？

答: 猫头鹰里如果希望区分大小写，必须选择case sensitive。否则，正则中的大小写一视同仁，就起不到判断大写字母的作用。

24. var object={

name:"obj",

getName:function(){

var that=this;

return function(){

console.log(that.name)

}

}

}

object.getName()();

答: 这个题，考的是闭包。

object.getName()，先调用getName函数:

getName()函数中有一个局部变量that，保存住this，此时this和that都指当前.前的对象object本身。

因为getName()中又有一个内层函数，用到了getName()的局部变量，且getName()将这个内层函数返回到外部，所以这里形成了闭包。内层函数中的that永久保存住了object的地址。今后，只要在内层函数中使用that，都相当于使用object对象。

所以, getName()返回的就是:

function(){ (that->object) console.log(that.name) }

这个函数再加()，等效于执行console.log(object.name)

所以，结果是object对象的name属性值"obj"

25. 猫头鹰是否支持零宽断言?

答: 猫头鹰选javascript只支持(?=)向后零宽断言，不支持向左零宽断言(?<=)。改成选java，就只吃(?<=)了

26. js中，明明负向零宽断言写对了(?=:\/\/)，但是却报错说正则表达式不正确？

答: 旧浏览器js不支持负向零宽断言(?=:\/\/)，需要升级一下浏览器。