# JS\_CORE

## day01

1. 我[槽，草，艹]也能匹配“我草”

答： “我，”也会被误判为敏感词

因为“，”放在[]也会作为一个普通的字符备选字

2. 我艹[槽草]，不能匹配“我槽”

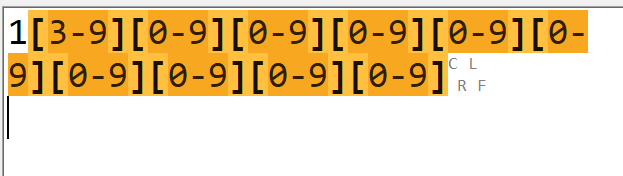
答：必须匹配3个字的词

| 槽 |

| 我 艹 |

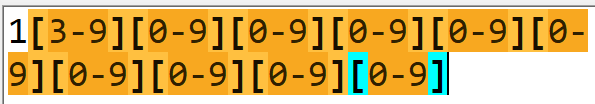
| 草 |

3. 正则明明写对了，却匹配不出敏感词或验证不正确

答: 正则表达式，开头，结尾，或中间不要有任何多余字符：空格，逗号，回车。比如  
 

结尾的CLRF表示换行，会影响匹配的结果。

所以，见到多余字符必须删除



4. 我[艹草操]不能匹配“草泥马”？

答：因为规则中规定必须是我字开头

| 艹 |

| 我 草 |

| 操 |

5. 想防住我草中间的空格：我·+草

答：问题1： 为啥输入空格，出现点：RegExpBuddy工具中，空格都是用中点来表示的，怕咱们看不到。所以，上方规则框里的中点其实是空格。

问题2： 其实正则中防守空格，不能直接输入空格必须用\s，因为用户完全有可能按tab键。所以，今后，正则中空格一律用\s

问题3： 我草之间的空格，屌丝可能输入，也可能不输入，也可能输入多个。所以，应该是可有可无，多了不限，才能防守住不确定个数的空格。所以，应该改为\s\*

6. ..·[0-9A-Z]{5} 也能匹配：京A·12345？

答：正则不能以能不能匹配某一个词看是否正确。应该广泛测试，暴力测试。

.虽然可以匹配京和A，但是同样也可以匹配其它任何子，比如aa·12345也能匹配。范围太大了。所以，能匹配只是最低要求。是否能够完全覆盖相匹配的，屏蔽不想匹配的，才是正则是否正确的评判标准。

7. \d{4}(\d{2})? 为什么?前要用()括起来?

答: \d{2}已经是一个完整的“字符集+量词”的组合了。不能再直接接量词。必须用()括起来才能接下一个量词。因为量词前即可接一个字符集，也可接一个分组来修饰分组出现的次数。

8. 用\b是否可以匹配单词之间的所有空格？

答：不行。因为\b代表位置，而位置，不等于字符。

比如: |no| |zuo| |no| |die|

空格↑ ↑空格↑ ↑空格↑ ↑空格↑ ↑空格

\s \b \b \s \b \b \s \b \b \s \b \b \s

这个例子中即包含多个边界位置，又包含多个空格，不是同一种东西。

9. 正则表达式不是只能匹配一个词吗？为什么可以匹配开头和结尾的空字符呢？

答: 正则表达式匹配时，是从字符串的开头开始匹配，一直匹配到结尾。发现几处符合要求的，就匹配几处。

^\s+|\s+$， 即可匹配开头，又可匹配结尾

^\s+|\s+$ -> ... ^\s+|\s+$ ->

↓ ↓

" no zuo no die "

但是，已经匹配过的敏感词，就不能参与下次匹配。下次匹配必须从本次敏感词之后才开始匹配。

比如: \d{4}|\d{6}

↓ ↓

|1234|56

10. 或不是一旦匹配就不向后执行了吗？

答: |在本次匹配中，如果前一个条件满足则后一个条件不执行。但是这并不能阻止整个正则继续想后续字符串再次匹配。

^\s+~~|\s+$~~ -> ... ~~^\s+|~~\s+$ ->

↓ ↓

" no zuo no die "

11. 即想匹配每个单词首字母，又想匹配每个单词末尾字母:

\b[a-z]\b

答: 不对的。这个正则的意思是，匹配两个单词边界之间只有一个字母的情况。

如果想即匹配单词开头的字母，又匹配单词结尾的字母：\b[a-z]|[a-z]\b

12. 用live server打开网页，发现端口变成5501或5502，怎么办？

答: 现阶段不会有影响。如果非要改，5501和5502是因为端口冲突造成。可能开了多个vs code和多个live server。只要关闭所有vs code甚至重启电脑，再开vs code，可能就会恢复成5500。

13. 希望的效果没出来？

答: 打开F12->Console，看问题！

14. document.write("xxx")?

答: document.write()

网页 的 写()

可以向网页的内容中写一条HTML片段

仅初级调试使用。今后，不用。

15. 为什么用模板字符串拼接字符串？和+拼接有什么差别？

答：今后开发中，都是用模板字符串拼接字符串。因为+会和程序中的加法计算产生歧义。功能上没有差别。

16. 猫头鹰软件中，为什么有的敏感词是黄色，有的是蓝色高亮?

答: 蓝色和黄色是一样的！只不过万一两个敏感词紧挨着时，可以用两种颜色区分是两个敏感词，不是一个。

17. 为什么匹配一位字母，不区分大小写：[A-Za-z]

而不是: [A-z]

答: A-z之间不都是字母！

ASCII: A-Z a-z

65-90 97-122

91-96

[ \ ] ^ \_ `

所以，如果从A-z，匹配范围会超出字母的范围，包含部分特殊符号！

18. \d{4,6} 既能匹配4位数字，又能匹配6位数字，那么: 652730，是先匹配4位？还是先匹配6位？

答: 正则表达式，大部分情况下，默认采用贪婪模式匹配。总是尽量匹配最长的那个关键词。能匹配到6位，就不止匹配4位。

19. \d{4,6} 为什么在一个字符串中可以匹配多个敏感词？

比如: 1234561234561234561234

答: 猫头鹰软件中，匹配时，都是用正则从字符串开头，一致匹配到字符串结尾。只要找到符合条件的就高亮显示。

20. 我.\*草 可不可以？

答: .\*防守范围过大。比如: 我去了她家。草草的收拾了物资。

.\*表示任意字符，任意个。很有可能误判

21. (我|w|wo) 为什么只能匹配w，匹配不到wo

答: |不遵守贪婪模式，而是采用类似短路逻辑的模式。

如果前一个规则已经匹配，则后一个规则，不再匹配！

总结: |不到万不得已，不用！

短路逻辑可复习: javascript.mmap中

运算符和表达式->包括->逻辑运算->短路逻辑

同理: 微|w|wei，也只能匹配w，无法匹配wei

但是，为什么^\s+|\s+$不短路。因为|左右是包含关系。^\s+是匹配开头的空字符，\s+$是匹配结尾的空字符。二者没有交集。

22. ^\s+|\s+$和^\s\*|\s\*$ 有什么差别？

答: 在开头或结尾没有空字符时，^\s+|\s+$，可能一次都不匹配。但是^\s\*|\s\*$，即使没有空格也会多匹配两次。

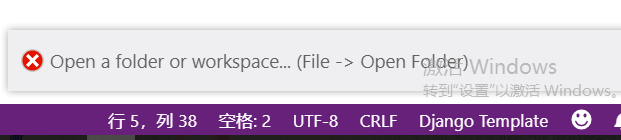
将来要删除时，如果开头结尾没有空格，则用^\s+|\s+$，一次都不删除。而用^\s\*|\s\*$，即使没有，也要白白执行两次没用的删除操作。

23. 规则: ,\b 字符串: ,no

答: 规则:边界前的逗号

字符串: , | no

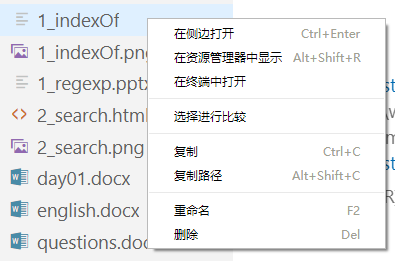
逗号 边界 单词no

24. 

答: vs code不能只打开一个文件，要打开，就必须打开这个文件所在的文件夹！

25. 右键看不到open with live server后缀:

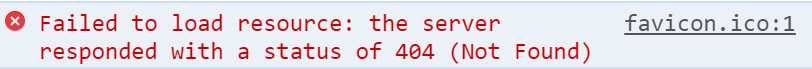
答: 1. 可能网页文件忘记写.html后缀了



26. Ctrl+/ 本来应该注释代码，但是没有注释，却出来{% comment %} xxx {% endcomment %}

答: 插件冲突: 卸载:



27. 

答: live server 试图查找一个正式网站的ico图标文件，但是，咱们现在做的都是示例网页，没有这个图片。所以这个错误可以忽略。实在看着不舒服，可多刷新一次，就消失了。

28. 如果live server运行出现浏览器错误或无法打开应用程序

答: 都是因为操作系统的默认浏览器配置错误：

在操作系统左下角找到开始按钮，点开选设置按钮，再选应用->默认应用->Web浏览器，换成Chrome

如果还报应用程序错误，就要装chrome软件。

29. str.indexOf()和str.search()为什么返回-1

答: 用别人的函数，只能遵守别人的规定！

30. ${}什么意思，忘了!

答: 思维导图: ES6->模板字符串！

31. vs code是英文版

答: 需要安装chinese simple扩展

但是，只能在线安装，需要开外网。下载vsix安装会有版本兼容性错误。

32. live server默认打开是IE浏览器，怎么办？

答: 修改操作系统中设置里的默认浏览器为chrome

33. live server显示无法打开页面？

答: 需要关闭防火墙

34. 如何防住一个单独的草字？

答: (([我卧]|wo)\s\*)?([草艹槽操]|cao)

但是，单独匹配一个草字，打击面太大，容易误伤。所以没意义。

35. \d\d，可不可以写成\d{2}

答: 可以。但是，\d比{2}，少一个字，我比较懒，愿意用\d。不愿意多写一个字。

36. [0-9x]能不能写成[\dx]？

答：可以，但是尽量不要嵌套。可读性差

37. live server提示找不到应用程序？

答: 配置操作系统的默认浏览器:

windows10操作系统: 左下角点开始菜单，选左下角齿轮图标->应用->默认应用->浏览器->选chrome

38. live server提示Error on port 5500？

答: 可能是因为同时开了多个vscode，导致5500端口冲突。解决办法是，关闭所有vscode，再只打开一个vscode。再运行live server。

## day02

1. 电子邮件的正则表达式:

答: \w+ 1个以上的合法字符

@ 必须有一个@

\w+ 1个以上的合法字符

\. 必须有一个. 但是.是通配符，要转义为\.

\w+ .后至少有一个后缀是一个以上的合法字符

\.cn 最后的.cn可有可无

\w+@\w+\.\w+(\.cn)?

2. 防守微信的正则表达式:

答: (微|w(ei)?) 汉字微或拼音wei，但是ei可有可无

\s\* 可能有空格，也可能没有空格

(信|x(in)?) 汉字信或拼音xin，但是in可有可无

(微|w(ei)?)\s\*(信|x(in)?)

3. 学了很多，记不住，好像什么也不会做？

答：没有做个人项目。

4. 数据库不会，nodejs不会，怎么做项目？

答：一点儿关系没有！

先做"原型"：就是和成品项目一模一样的静态页面

静态界面上的数据去原网站copy，或者将来可以找一些开放接口。

5. 为什么学nodejs?

答: 为了了解后端，为了自保！当功能出现问题时，可以判断出是那一端的责任。

6. 第一阶段学的不好，第二阶段boot记不住？

答: 第一阶段，只有JSBASIC 那9天是重点

第二阶段，只有CSS基础，CSS高级，CSS3动画，媒体查询。

7. 做项目可以仿写别人模板的样式吗？

答：现在大家正处在照猫画虎的阶段，就像达芬奇画蛋一样。不可能学完立刻原创。所以，现阶段，做静态页面，本就应该以模仿和参考别人优秀的模板为主。时间有限，精力有限，最忌讳事事亲力亲为，从头写到尾。但是，仿写别人的网站，要吸取别人的优秀做法和技巧。效果上，要尽量达到原来的要求。最好能够有超过原网站的效果作为亮点。所以，应该多参考一些优秀的模板。

8. 多个页面中相同的部分，比如页头，页尾怎么做？

答：多个页面中共用的部分，单独做成一个.html文件，比如,header.html，单独调试样式和内容。

但是！千万不要用iframe引入！

在我这个阶段，会教大家用正确的ajax和vue两种方式引入公共部分。

9. 如果git命令行有问题，怎么办？

答：下载有界面的github desktop

10. match返回的数组中，房间号必须叫index吗？可以自己改名吗？

答：不可以。必须用index。

虽然关联数组的下标名，可以自定义，但是，那是在你自己创建关联数组的情况下。

现在这个数组是人家match函数返回的，不是你自己创建的。所以，人家返回什么样，就只能按照人家的要求用。

11. 然哥教我们用的iframe，为什么你不让用？

答：那是因为你们当时学的少，然哥也没办法。

今后做项目就是不用iframe的。然哥也不用。

每个阶段都有每个阶段的局限。

iframe会让一个页面变成两个页面，两个页面之间数据几乎无法共享和传递，所以，今后页头加载，要么在服务端生成，要么在客户端用ajax拼接。绝不用iframe

12. 为什么输入0就可以拿到找到的敏感词呢？

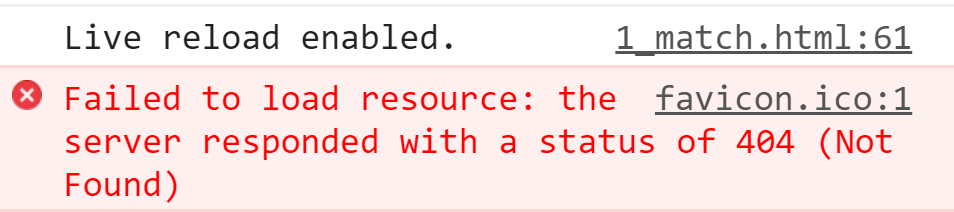
答：match在找到敏感词后，会自动在内部创建一个数组，然后将敏感词放入数组的0位置。然后再把数组交给我们。所以，人家放在哪个位置，我们就只能从哪个位置取。

就好像中午点盒饭，人家把肉菜放在第一个饭盒里，把素菜放在第二个饭盒里。那我们想吃肉菜，只能从第一个饭盒里取。

13. match返回的arr数组

长度应该是6个房间呀？为什么length为3？

答：因为length默认只能统计数字下标的房间个数。字符串下标的房间不参与统计。记住即可。没有为什么。

14. ？

答：这个右侧显示favicon.ico:1的错误不用管。是live servre运行时报的一个无关紧要的错误。

重新刷新页面就没有了。

15. vs code下弹出错误: open a folder or workspace

答：vs code要用Live server运行，必须先打开一个文件夹，文件夹中再包含网页文件。不能直接打开网页文件运行。

16. arr!=0 也可以？

答: !=是关系运算，关系运算中，左右默认隐式转为数字

如果arr返回null，也会被Number()隐式转为数字

Number(null)=0，所以可以和0作比较。

如果arr返回的是一个有内容个数组，也会被转为数字：Number([1,2,3])=NaN，NaN!=0 返回true条件成立。

但是，按照语义，必须写成arr!=null。

写程序，语义优先。还有，不要瞎改程序！

17. 返回的敏感词一定放在0位置吗？

答：一定的！不会例外！因为浏览器自带的原生函数，都是ECMAScript标准固定好的，必须这么实现。不会有例外。

还有，写程序要有基本的人和人之间的信任。

18. ajax四步或五步不会，怎么办？

答：根本没关系:

DOM: 我给你封装好了ajax()

jQuery: 人家也封装好了$.ajax()

VUE: 人家也封装好了axois.get()

19. == === != !==

答: == !=是带隐式转换的等于比较

交给他们的两个值，都要先被隐式转换一下，再比较。可能和你的本意就不一致了。

=== !==是不带隐式转换的严格等于比较

绝对不会对给的两个值擅自加工。而是保持原样进行比较。多数情况下，都要用===和!==代替==和!=

但是，如果必须在不同类型数据之间作比较，需要先强制转换为一致类型，再用===和!==比较。

如果确实对js的隐式转换了如指掌，可以使用==和!=

20.\*\*\*\*\* Cannot read property 'xxx' of null或undefined

不能 读取 属性←xxx←的←null 或 undefined

答: 凡是看到这个句式的错误，不是属性名'xxx'写错了，也可能不是这句话写错了。很可能是xxx属性.前的变量，该拿到的值，没拿到！

解决: 往前找！找这个变量在哪里赋的值，为什么没有拿到值

21. replace中的function，换成箭头函数，也能用

答: 暂时禁止使用箭头函数。因为箭头函数有坑，不是所有情况都能换。所以，必须等我讲完再用！

22. replace(/xxx/,function(){...})中的匿名函数是因为那个()才被调用的？

答：回调函数，是一种我们只负责定义，不负责调用，而交给其他函数内部去自动调用的函数。所以，我们不可能看到回调函数被自动调用时的()。

这里replace()后的()是开始调用replace函数，不是调用内部匿名函数的()

function后的()，是定义函数的语法要求，不是函数调用。

所以，我们现在用肉眼看到的()，都和内部匿名函数自动调用无关。

23. console.log(ltrim(str)) ltrim是回调函数?

答: ltrim绝不是回调函数，而是主动调用过了一次！ltrim(str)已经构成了标准的函数调用语法。

为什么ltrim(str)直接放在console.log()里？因为我想将ltrim()执行的结果，直接交给console.log()去打印

console.log(<- 结果 <- ltrim(str))

24. ltrim()不加str，也能执行?

答: 每个函数都是一个独立的功能体，函数内部的变量应该尽量避免直接使用外部的变量名。

这里ltrim()不加str参数也可使用，是因为巧合。函数内的变量和全局变量str同名了。所以，就算你不给str，函数执行时会自动去全局找str变量用。

但是这样是不对的。因为外部变量名一旦变化，函数也连带着不能使用了。

所以，将来函数需要什么数据，必须通过形参从外部获得。而就不能直接使用外部的变量。

25. var reg=/^\d{6}$/; reg.test(str)验证的是数字吗？

答: 验证的不是数字。而是字符串中的数字字符。

比如: 程序中123和"123"

虽然在页面上输入的是123456，但是所有从页面传入的数据，都自动加上了""变成了字符串

年龄:

|  |
| --- |
| 26 |

↓

程序中

var age=input.value;

age:"26"

26. 将/^\d{6}$/换成new RegExp("^\d{6}$")为什么不对？

答： 正则的\d和js中字符串本身的\转义字符冲突了！解析时，js先将\d看做自己的转义字符，但是js中又没有\d这个转义字符，所以js擅自将\d前的\去掉了！

解决: 如果用new RegExp()其中的\d都要转为\\d，第一个\是在告诉js，后边的东西不要动，不需要你解析！

但是\u4e00和\u9fa5不用保护。因为js认识\u

27. if(arr!=null)  
 ...arr.index ... arr[0] ...

就正确

如果改为

...arr.index ... arr[0] ...

}while(arr!=null); 报错

答: 如果arr有可能是null的话应该先判断不是null，才能使用。如果将条件放在后边的while中，则顺序变为先使用，后判断，就失去了对arr=null的防护。

28. arr["index"]可以反问"index"位置的元素，也可以用.访问arr.index也可以。arr["length"]可反问数组长度，arr.length也可访问数组长度。为什么？

答: js中一切对象底层，其实都是关联数组！

任何对象和数组的成员，都可用["下标"]访问！

只要是非数字的下标，都可用.简写！

任何对象和数组都可在任何时候随意添加新成员

访问任何对象和数组的不存在位置，都不会报错，而是返回undefined！

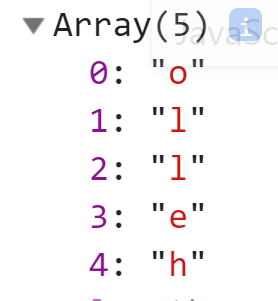
所以，其实arr["index"]可简写为arr.index。

其实arr.length也可改为arr["length"]

但是，数字下标不能简写！arr.0和小数点冲突了！

只能用[]方式访问！

29. 为什么arr.reverse()调用前输出的arr，却是已经翻转过的！



答: console.log(对象/数组)其实不是给对象拍照！而是输出对象的地址！

当程序执行完，我们去点地址旁边的时，才临时去内存中找到这个对象，显示对象的内容。

因为此时程序已经执行完了，且内存中只有一个arr数组。所以，无论何时点开三角，看到的对象内容都是一样的，且都是最终的结果！

解决: 如果非要看到数组中间瞬时的状态: 2个办法

1. 刷新页面，会在控制台，显示数组在中途执行时的照片。但是也不要展开。

2. 可以将数组或对象转为字符串，再输出

String(数组) vs toString(数组)

其实两个是极其相似的。

30. (微|w|wei)单独用，会有短路现象，总是匹配w，而不会匹配wei，但是(微|w|wei)\s\*(信|xin|x)，就能匹配weixin？为什么没有发生短路呢？

答: 因为正则表达式不可能只匹配第一个字母后后三个字母weixin, 要匹配，就会匹配整个词。正式因为后边的xin，才顺带匹配上了ei。

31.( (微| wei | w))，既能匹配wei，又能匹配w?

答: 因为先比较的是wei，所以如果wei先满足，则优先选中wei，而不会再单独匹配w。

32. var name=xxx，经常无效或出错？

答: 禁止使用name当做变量名，最好也不要当做属性名。

因为在浏览器中全局下，已经有一个内置的属性name，用于标识当前窗口的名称。是浏览器自用的。不允许任何人替换！

33. replace中回调函数里的形参变量keyword可不可以改名？

答: 几乎所有的形参变量都可以随便改名，但是只要保证变量名见名知意即可。

比如: keyword可以改为kword，kw都行。

34. 形参变量keyword能否在replace外输出或使用？如果return keyword了，能不能在函数外使用？

答：不行！因为形参变量属于所在函数的局部变量。

如果return keyword，尝试将keyword返回到函数外部，也只是将keyword变量的值返回到外部，变量依然出不来。也必须用另一个外部的变量接住keyword的值。接住的也不是keyword变量。

35. 何时使用回调函数：

答: 1. 如果别的函数需要我们提供回调函数参与执行时

2. \*主动使用回调函数\*: 3\_callback.wmv

36. `搜索 ${arr} 相关的商品...`，为什么arr会变成逗号分隔的字符串？

答: 模板字符串本质，是字符串拼接，代替的是"xxx"+"xxx"

在字符串拼接中"xxx" + "xxx"，任何变量碰到字符串，都会被自动转为字符串。但是转字符串不是凭空转的。自动转字符串其实都是自动调用了该对象/数组的toString()函数。

上例中

`搜索 ${arr} 相关的商品...`

等效于: `搜索` + arr +`相关的商品...`

等效于: `搜索` + arr.toString() +`相关的商品...`

因为arr.toString()，会返回所有数组内容用逗号相连形成的字符串: 华为,i7,8g，所以

上例最终等效于：

等效于: `搜索` + "华为,i7,8g " +`相关的商品...`

37. console.log(ltrim(str))为什么ltrim可以放在console.log()中:

答: 因为ltrim()也是有返回值的。只有ltrim()有返回值，就可以将返回值直接交给console.log()打印。

38. function ltrim(oldStr){} 和 ltrim(str)有什么关系？

答: function ltrim(oldStr){}是定义一个函数。oldStr，是ltrim函数的形参，表示调用ltrim()函数时必须的数据是原字符串。只有传入原字符串，ltrim()才能将原字符串去掉空字符，返回新的处理结果。

ltrim(str)是调用函数，就是要开始执行函数的处理过程。str是传给函数形参变量的实参值。调用函数时，必须按照函数定义时的要求传入实参值，ltrim()函数才能正常执行。传入的实参str的值，就是赋值给oldStr形参变量，用于在函数内执行操作之用。



39. arr.reverse()为什么不需要用变量=接住

答: 数组都是可变类型。所有数组家函数都可以直接修改原数组，所以不需要用变量=接住。

40. 用身份证号判断男女中，pid[16]为什么不能写成pid[-2]?

答: 用身份证号判断男女的方式: 倒数第二位是奇数，就是男士，倒数第二位是偶数，就是女士。

var pid="110102199312262151"

012345678901234567

var sex=pid[16]%2==0?"女":"男"

其中: pid[16]不能写成pid[-2]，因为数组中一个元素，只有一个下标。就是从0开始递增的数字下标。数组中，没有负数下标。

但是: 为什么arr.slice()中可以用-2等负数下标？因为arr.slice(6,-2)，实际上会被自动翻译为arr.slice(6,arr.length-2)。你没有使用arr.length，但是人家arr.slice，为了减轻你的负担，自动调用了arr.length，才可以用负数下标，其实是自动做减法。所以，只有arr.slice()可以用负数位置。

另外，如果不愿意，从0开始向后数，如果位置离结尾近，也可以写成pid[pid.length-2]。这样就不用从头开始数了。也就是虽然没有人帮我们自动调用pid.length，我们可以自己主动调用。来为自己创造条件使用倒数位置。

41. vs code 开发环境不能换行的可以:

答: 在左下角，打开设置，在右侧编辑窗口，添加"editor.wordWrap": "on",

42. arr.slice()也修改原数组？

答: 数组家两个函数特殊，不修改原数组，只返回新数组:

拼接数组

var 新数组=arr.concat(xxx,xxx)

截取子数组

var 子数组=arr.slice(开始位置,结束位置)

数组家还有一批不返回数组的函数，比如arr.toString()和arr.join()，返回的是字符串，所以，无所谓修改还是不修改原数组，肯定返回新字符串，原数组不变。所以，也需要变量接住。

其余函数，都是直接修改原数组的。

复习思维导图里数组API一栏下的所有内容。

43. var dong=["包","包","包","包"]

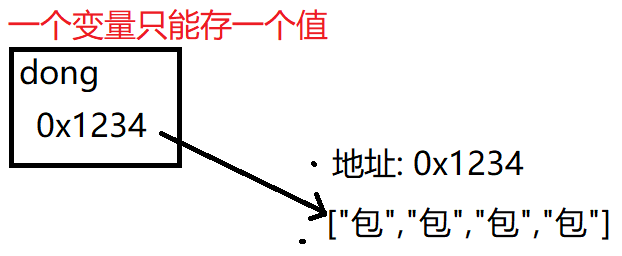
var ran=dong;

ran.length--;

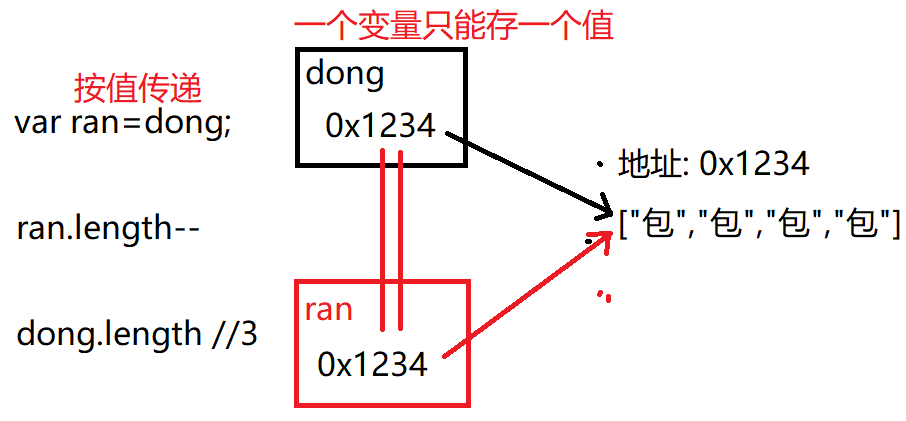
console.log(dong.length);为什么是3？

答: 第一句话: var dong=["包","包","包","包"]，在内存中创建了一个数组对象，数组对象在内存中有唯一的地址值。然后创建一个变量dong，将数组对象的地址值保存到变量dong中。

因为js规定，一个变量规定只能存一个值，而数组里包含多个值，所以数组无法直接保存到变量中。只能放在变量外部。然后，变量里只能保存一个数组的地址值，用来指向数组实际存储位置。这称为引用。就好像现实中我们手机里不能把别人直接装进手机，只能把别人的手机号保存在手机里，想找人家时，通过手机号找到这个人。

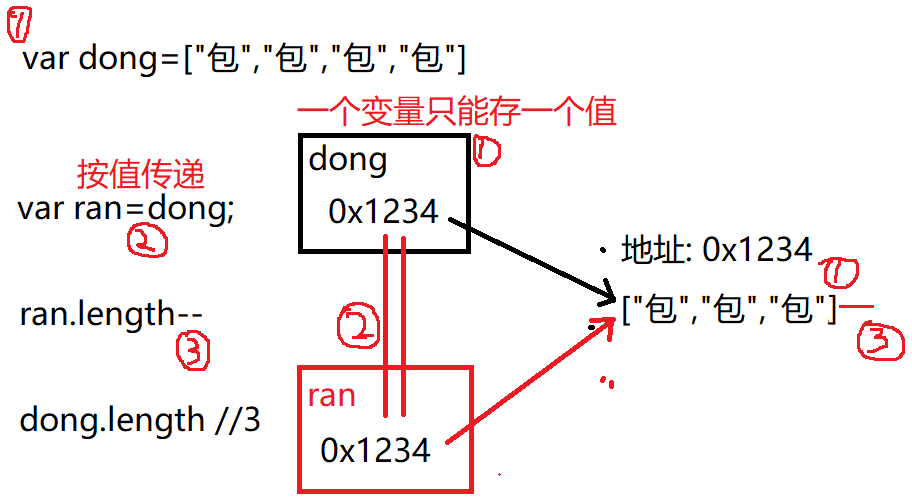


第二句话: var ran=dong; 两个变量间赋值，发生按值传递。也就是将原变量中的值赋值一个副本到新变量中。这里，因为dong中保存的只是数组的地址值，所以，东只能将自己的地址值复制给ran，数组并不会多一个。所以形成了以下格局，dong和ran两个变量使用相同的地址值，指向同一个数组对象。就好像现实中，很多人都可以保存你的手机号，然后都可以通过手机号找到你这一个人是相同的道理:



第三句话: ran通过地址，将数组长度-1，导致4个包子剩3个。

最后，输出dong.length，因为dong和ran看的的同一屉包子，所以，dong看到的包子，也只剩3个了。



44. 如果不小心把test点没了

答: 可以在，右上角选View按钮(调色板)，在下拉列表里选test。

45. ([微|w(ei)?])\s([信|x(in)?])，为什么可以匹配i xin?

答: 因为放入[]中的一切都变成单个的普通字符。i在[]里也是一个备选字符。所以可以单独匹配i字母。解决: 不要放在[]里

46. 一个网页中后一个<script>中的程序为什么可以访问前一个<script>中的变量？

答: 因为一个网页只有一个统一的巨大的对象window。无论在哪个<script>中声明的全局变量，都集中存放在统一的一个window对象内。

但是<script>会阻断声明提前。每个script中的变量，就算提前，也只能提前到当前<script>的顶部。绝不可能超出当前<script>范围。所以前一个<script>中的程序，无法访问后一个<script>中的变量。

47. 引入一个网页中的多个js文件内的变量，可以互相访问吗？

答: 只要引入一个网页内的js文件的内容，都属于同一个网页的window对象内。所以，后引入的js文件中的程序，就可以访问先引入的js文件中的内容。但是，先引入的js文件中的程序，就无法访问后引入的js文件中的变量。

## day03

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。

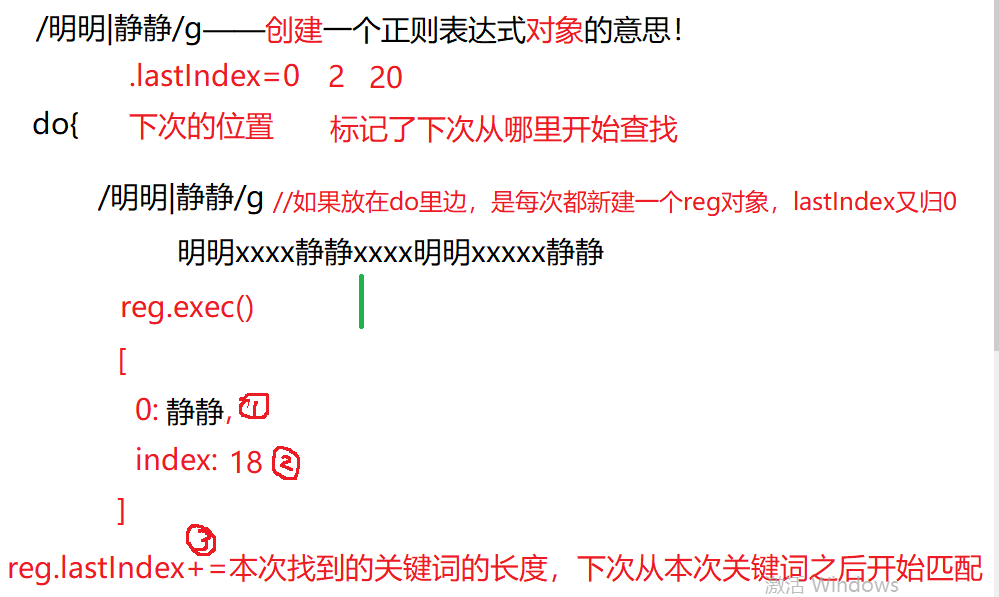
/正则/放在do外部，只创建一个，反复使用

/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。

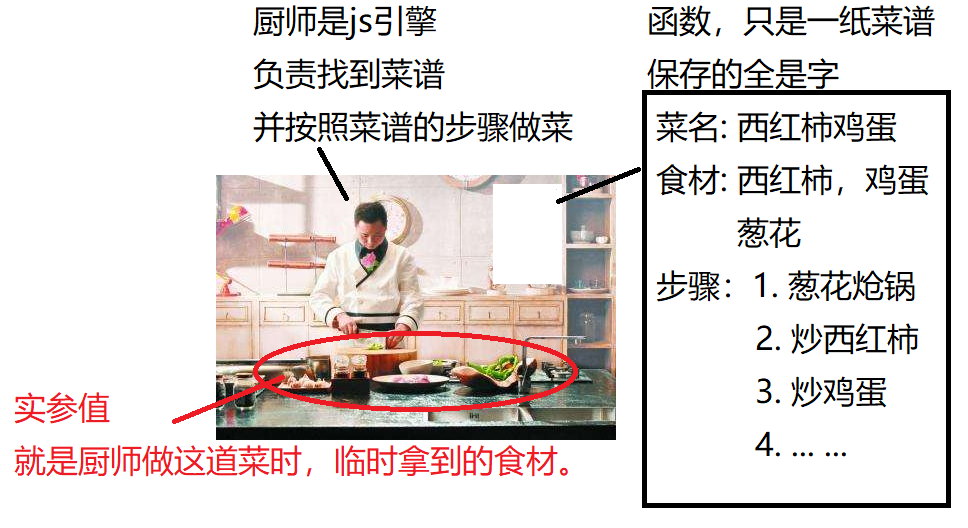
19. 第一次调用函数传入的参数赋值给形参，第二次调用函数不传入形参，形参就没有值，是函数内部将形参的赋值给处理掉了吗？还是形参会隐式覆盖？

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。

## day04

1. console.log(window.a)输出undefined，而console.log(a)，就报错，难道window.a是全局，a就不是全局？

答: 所有全局变量一定是保存在window中的

保存在window中的变量其实有两种访问方式:

可以以变量语法方式: 变量名

可以以window对象的成员形式使用: window.变量

比如: var a=10;

console.log(a) //10

因为所有没有前缀的，js都会自动加上window.前缀——仅限于全局作用域中执行的程序

比如: alert() -> 执行时被补全为window.alert()

再比如: parseInt() -> 执行时被补全为window.parseInt()

console.log(window.a) //10

如果没有声明a，执行console.log(a)和console.log(window.a)，一个报错，一个不报错？

语法规定:

如果直接访问一个不存在的变量，会报错！ReferenceError: a未定义

如果强行访问对象中一个不存在的属性，或访问数组中一个不存在的位置，不报错！而是返回undefined。

全局变量a，其实都有两个角色，一个是单纯的变量a，同时也是window对象中的成员。所以，以单独变量方式访问不存在的变量，则参考第一条规则，会报错。如果以window.a成员的方式访问变量a，永远不会报错！而是返回undefined。

补: var a和 window.a

1. 如果未声明变量a，则单独使用a会报错，而使用window.a不报错。

2. var a的变量不能用delete彻底删除

而用window.a强行添加的变量，才能被delete彻底删除

总结: 今后即使用全局变量，也要用window.变量。不要用var。因为如果不想用了，还可以用delete彻底删除

2. pay=null后，为什么变量pay还在？

答: 给pay变量赋值为null，只是清空变量的内容，而变量在window中不会被删除。

如果希望彻底删除变量: delete window.变量

但是: 删除不掉！

原因: pay是var pay创建的。如果想用delete彻底删除变量，创建变量时，必须用window.pay

3. console.log(a) //报错，为什么不是undefine?

a=10;

console.log(a)

答？因为a=10前，没有var。没有var，就不是声明语法，就不会会被声明提前。不会被声明提前，第一次使用时就没有a可用，就报错。a=10，只是一个赋值语句。赋值语句会强行创建变量a，但是，也只能在语句所在位置才创建a。因为声明提前规定，“”赋值留在原地“

4. 为什么加入overflow:hidden或添加元素设置为display:table，可防止高度坍塌？BFC？

答: Block Formatting Context

以块的形式，格式化显示元素的子内容

为什么: 启用BFC模式都是为了避免子元素被隐藏或遮挡。

如何启用BFC：

浮动元素：float 除 none 以外的值。

绝对定位元素：position (absolute、fixed)。

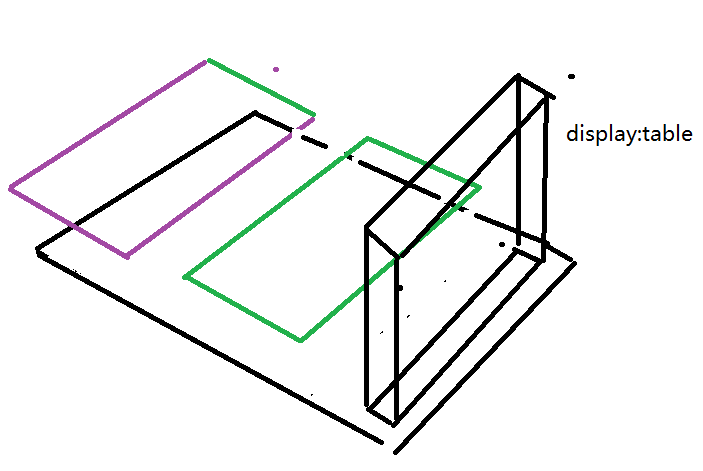
display 为 inline-block、table-cells、flex。

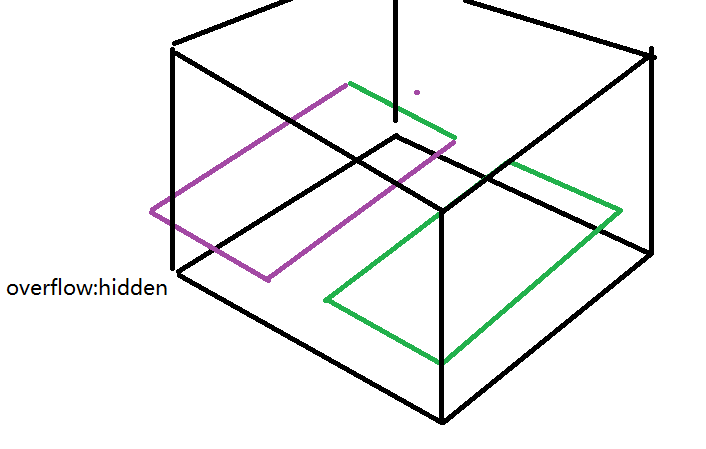
overflow 除了 visible 以外的值 (hidden、auto、scroll)

后果:

1. 父元素启用BFC，则范围必须包含所有子元素的范围。

2. 平级子元素启用BFC，则其他兄弟子元素不能覆盖在它之上。





5. 作用域到底是什么？

答: 作用域其实就保存变量的对象。只有两种作用域对象:

全局作用域对象: window  
 函数作用域对象：一般看不见，也没有。只有在调用函数时，在内存中临时创建。函数作用域对象中保存的是本次函数调用使用的局部变量。

6. 反复调用外层函数parent()，为什么变量值不变？和调用内层函数pay有什么差别？

答: 调用内层函数pay是一个小孩儿，反复花自己的红包

反复调用父母parent()，是反复生小孩儿，反复包红包。

7. 如何从外部访问到闭包中的变量:

答: 不让访问！如果能从外部随便访问，不叫闭包！

其实，是有办法看到的！

输出函数在内存中的结构： console.dir(pay)

dir可输出对象在内存中的详细存储结构

每个函数输出后都可看到一个属性[[scopes]]，这就是还有列表。里边在不调用时，暂时只有window或window和父母的作用域。

8. 为什么在修改之前输出对象的内容和修改后输出的内容都是修改之后的新内容？

答: console.log()和console.dir()不是给对象拍照的意思，不是留住对象瞬间的状态。而是输出对象的引用地址。

当程序结束后，我们点击三角展开对象时，控制台才临时去内存中找对应的对象的内容。此时，程序已经结束，所以无论点开哪个三角，看到的只能是最后的修改值。  
 解决: 分布注释，解开注释，调试程序，查看阶段的输出结果。不要最后一股脑都输出。

9. 为什么调用外层函数必须用变量接住返回值？

答: 如果想获得外层函数的返回结果，必须自己用变量接一下。如果不接，第一，无法获得内容。第二，不能重复使用。

因为只要重复使用的东西，都要保存在变量中

10. 如果外层函数只用一次，怎么办？

答: 绝大多数情况下，父母函数确实只要产生一次闭包，就不再使用。

解决: 如果一个函数只用一次，都可用匿名函数自调

比如: var pay=(function(){

var total=1000;

return function(money){

total-=money;

console.log(`本次花了${money}元，还剩${total}元`)

}

})();

pay(100);

结果: 父母函数调用一次后释放，但是孩子函数和父母函数的作用域对象都被保留下来。

11. getN(), nAdd(), getN() 为什么只有两个输出？

答: nAdd没有输出，也没有返回值。

12. 给nAdd前加上var，会怎样?

答: 报错，nAdd未定义。因为nAdd成了局部变量。函数外不可使用。

13. 在nAdd中改成console.log(n++)为什么输出999?

答: 因为是后++，先输出n旧的值，然后再对n+1

14. 禁止使用"name"作为变量名?

答：因为window中有一个固定的常量叫name，不可替代。

15. 为什么关联数组ym里明明有三个成绩，但是ym.length是0？

答：数组的length属性，紧紧记录数字下标的元素个数。自定义下标名称的元素，是不参与length统计的。

16. var lilei2=clone(lilei) 是什么意思？

答：调用clone函数，并将lilei对象作为参数，传入clone函数中处理。将clone函数的返回值，赋值给lilei2变量保存。

就好像，把玉米粒，放到爆米花机里，爆出爆米花，再将爆出的爆米花，装到袋子里保存起来。

17. nAdd=function(){ n++ }既然是全局变量，为什么可以访问fun(){}中的局部变量n呢？

答: nAdd=function(){ n++ }其实是2件事:

1.先创建一个函数——在fun内的——孩子是fun生的。

2. 然后才把fun生的孩子赋值给了全局变量，才从fun中出来的。

所以，因为function是fun生的，所以，使用自己父母的局部变量是情理之中的。

18. 既然对象不是作用域，那么对象中的属性，应该是全局变量呀？

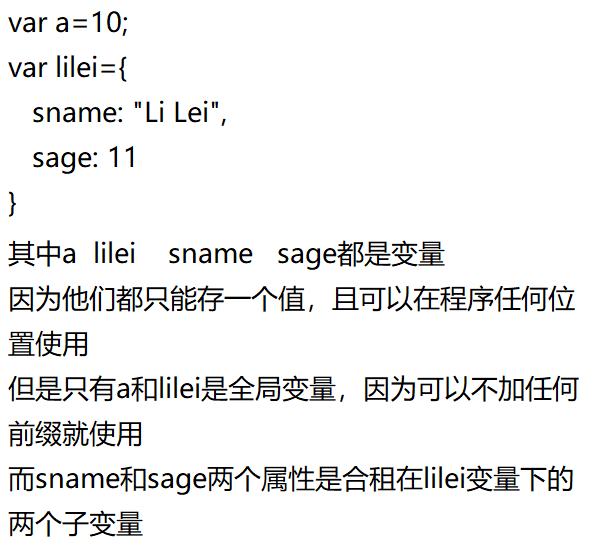
答: 对象中的属性，不是全局变量，是因为，直接使用属性名，不加对象名.前缀，是访问不到属性变量的

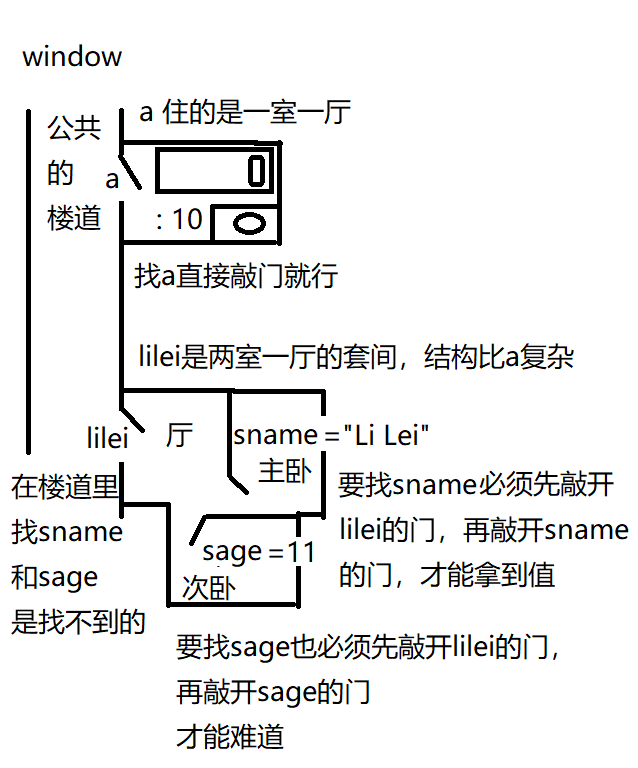
对象中的属性，也不是局部变量，因为，对象中的属性可以通过加对象名.前缀随时访问。而局部变量，只能在函数内访问，无论加什么，都不能在函数外访问。

那么对象中的属性，到底是什么变量？

答: 对象中的属性是一种套在对象变量中的子变量。可以当做变量一样使用，但是查找时，必须先找到对象，进入对象，才能找到属性变量。

就好像租房子： 有整租一室一厅，有合租两室一厅，合租三室一厅。





但是，sname和sage不是局部变量，更不是私有。因为，只要你可以敲开lilei的门，sname和sage你随便用，没有限制。

19: lilei中的方法intr()的好友列表中为什么只有两级作用域

答: 因为intr()方法外层的程序结构不是函数，只是一个普通的变量引用的一个普通对象，只不过结构比一般的只存一个值的变量结构复杂而已。所以不是一级作用域对象。不能作为好友列表中的一员。

函数的好友列表中，只能保存作用域对象，只要不是作用域的对象，都没有资格加入函数的好友列表中。

而闭包中，内层函数的外部，是一个外层函数，是一级作用域，既然是一级作用域对象，就有资格作为内层函数的好友列表中的一员。

20. var char=str[i]; arr[char]中的char为什么不加""？

答: char是不确定的，是从str中临时获取的变量值。只要是变量，就不应该加""。如果不加引号，程序在执行时，会动态将char变量的位置替换成char中保存的值。如果加了""，就无法动态改变变量值了。

21. arr["h"]===undefined，就可以判断数组中没有名为"h"的元素

答: 访问数组中任何一个位置的元素时，如果下标位置存在，就返回元素的值，如果下标位置不存在，也不会报错，而是返回undefined，所以经常用强行访问某个位置的元素，是否返回undefined来判断是否存在这个位置的元素。

22. 为什么===undefined，换成==有什么不同?

答: 因为==是自带隐式转换的等于比较，会将null也转为undefined。所以如果用==很容易误判null和undefined。而===是不带隐式转换的等于比较，所以可以区分null和undefined。因为null和undefined首先类型都不同。

23. 为什么数组中不存在arr["h"]，也能强行赋值？

答: js规定所有数组和对象，可在任何时候强行添加新属性/新成员，而不会报错！

24. for(var i=0;i<3;i++){} 循环结束后，i不应该是2吗？为什么是3？

答: 循环退出，一定是因为循环条件不满足。如果i还是2，那么2<3，循环条件还满足，怎么会退出循环呢？之所以循环退出了，一定是因为i<3条件不满足了。i只有=3时，循环条件才不满足了，循环才退出。

25. 为什么每个对象都有sname和sage属性，为什么不放在原型对象中公用呢？

答: 要公用的属性，必须连值都要完全相同，但是sname和sage是每个子对象各自不同的，所以不应该公用。

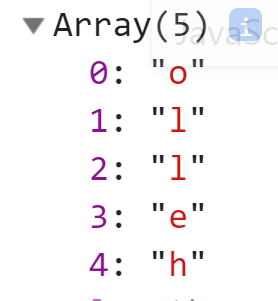
26. parent()()和var a=parent(); a()，两个输出结果是一样的，有什么差别？  
 答: parent()()，只能调用一次返回的内层函数，调用后，立刻释放。因为没有变量接住。

后边用a接住内层函数对象，就可反复使用。

27. var getN=fun(); fun()(); getN(); nAdd(); getN(); 结果是两个999，为什么？

答: nAdd是全局变量，第二次fun()时，又生成了一个nAdd把第一个fun()生成的nAdd覆盖了。又因为第二次fun()的nAdd和第一次fun()生的getN()不是一次生的，所以各自使用各自的红包，毫无关系。新的nAdd()影响不了旧的getN()的值。

28. 为什么arr.reverse()调用前输出的arr，却是已经翻转过的！



答: console.log(对象/数组)其实不是给对象拍照！而是输出对象的地址！

当程序执行完，我们去点地址旁边的时，才临时去内存中找到这个对象，显示对象的内容。

因为此时程序已经执行完了，且内存中只有一个arr数组。所以，无论何时点开三角，看到的对象内容都是一样的，且都是最终的结果！

解决: 如果非要看到数组中间瞬时的状态: 2个办法

1. 刷新页面，会在控制台，显示数组在中途执行时的照片。但是也不要展开。

2. 可以将数组或对象转为字符串，再输出

String(数组) vs toString(数组)

其实两个是极其相似的。

29. 函数为什么不能保存在变量中，而只能把函数地址地址保存在变量中？

答: 变量规定只能存1个值！只有简单的原始类型(number, string, boolean, null, undefined)值，才可直接保存在变量中。

而复杂的引用类型(数组，函数，对象等)，无法保存进变量中，只能在变量外开辟存储空间。将对象的地址保存回变量中。因为地址只是一个值，所以不违反变量的规定。

使用变量时，是通过变量中保存的地址找到远处的对象，来使用。

30. function add(){ ... } 和 Array.prototype.sum=function(){}都可以累加任意多个数的和，有什么区别？

答: function add(){ ... }，是所有程序都可随便使用的一个函数，且多个数值以零散的方式传入函数内求和。

Array.prototype.sum()，只允许数组家孩子使用，且只能遍历当前正在调用sum()函数的.前的数组中的元素，求和。无法随便传入sum()中参数值。

31. Array.prototype.sum=function(){}中的this.length，能不能换成arguments.length?

答: 不行。因为sum()函数的用法，是arr1.sum()，arr2.sum()，是sum()中没有参数，而是去遍历.前的数组中的内容。所以arguments在这里没用。因为arguments只能获得参数值，无法获得点前的数组。只有this，才能自动获得.前的对象。

## day05

1. 如何从ftp上下载html和js文件?

答: 右键->链接另存为

2. 闭包的外层函数向外抛出内层函数共有几种方式?

答: 3种:

1. return function(){ ... }

2. 全局变量=function(){ ... }

3. return [ function, ...]

return { 方法: function(){ ... } }

3. 为什么lilei.intr()可以调用到原型对象中的intr()函数?

答: 因为原型对象是lilei的父级对象。程序中规定，调用函数时，先在对象本地查找。如果对象本地没找到，就自动延\_ \_proto\_ \_向父对象找。

4. 到底怎样创建一种类型?

答: 2步:

1. 先定义构造函数:

function Student(sname,sage){

this.sname=sname;

this.sage=sage;

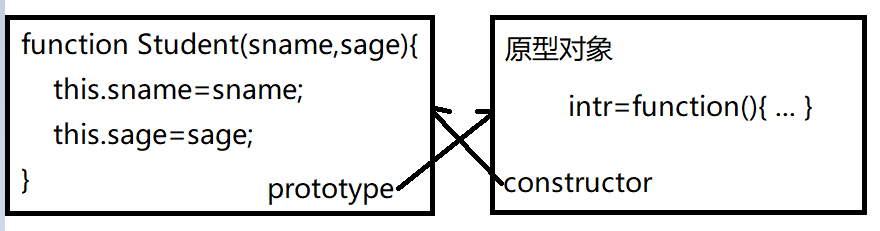
}

2. 为构造函数的原型对象强行添加共有方法:

Student.prototype.intr=function(){

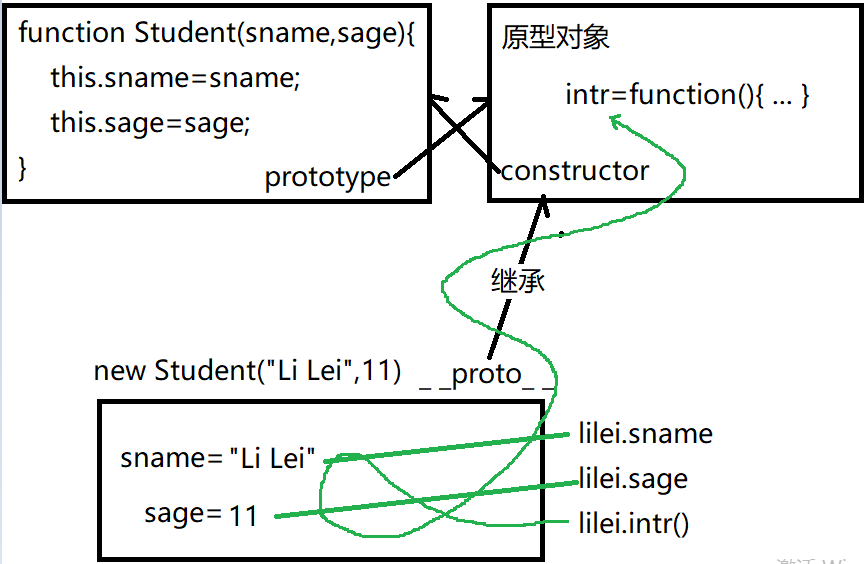
console.log(`我是 ${this.sname}, 我今年 ${this.sage} 岁`)

}



创建类型后，如何用类型，创建子对象:

var lilei=new Student("Li Lei",11)



5. 都指向原型对象，为什么一个属性叫prototype，一个属性叫\_ \_proto\_ \_?

答: prototype和\_ \_proto\_ \_指的是同一个对象，都是构造函数的原型对象。

和现实一样，以不同的角色去访问原型对象时，必须用与自己身份相符的称呼。必须有所区分。

比如: 妈妈称呼爸爸，就叫老公

孩子称呼爸爸，就应该叫爸爸！

6. 原型对象中的this->?

答: 判断this一定不好看定义在哪里，保存在哪里

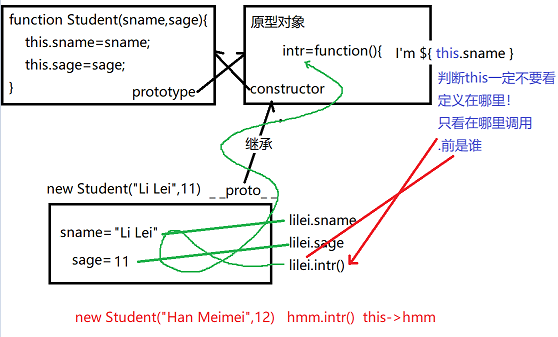
只看在哪里调用！看.前是谁

Student.prototype.intr=function(){ ... this.sname ... }

虽然存在原型对象中，但是不是在原型对象中被调用的

而是在稍后，由子对象调用: lilei.intr() hmm.intr();

所以，原型对象中的方法中的this，指将来调用该函数的.前的子对象。



7. 一共讲了几种this的情况?

答: 4种:

1. obj.fun() -> this->obj

2. fun() 或(function(){ ... })() this->window

3. new fun() -> this->新对象

4. Student.prototype.intr=function(){ ... this.sname ...}

this-> 将来调用intr()的.前的子对象

8. function Student()是一个函数，为什么能.prototype属性？

答：js中一切都是对象，函数也是对象。只要是对象，都可以用.访问属性。都可以添加属性。其实，函数和数组，和lilei一样都是存东西的对象。只不过，函数中，存的是代码段，而数组和lilei中存的是数据而已。

9. 何时使用prototype，何时使用\_ \_proto\_ \_?

答: 1. 向原型对象中添加共有属性和方法时，必须通过构造函数妈妈找到原型对象爸爸才能，向其中添加共有属性和方法。

类型名.prototype.共有属性=值

2. 如果通过儿子，查看自己爹里的内容时，可以通过子对象.\_ \_proto\_ \_获得儿子的爹。

其实，向原型对象中添加共有属性，也可以通过子对象.\_ \_proto\_ \_.共有属性=值，强行添加，但是通常，向原型对象中添加共有成员时，还没有子对象呢。所以，很少用子对象.\_ \_proto\_ \_去操作父对象。

10. 如果孩子擅自修改了共有属性，导致在自己身上添加了同名自有属性，那么，如果调用原型对象中的方法时，方法中刚好用到这个共有属性的话，方法使用的是共有的还是自有的属性？

答：方法使用的是自有的同名属性。因为，访问属性使用的是this.属性名。 this查找属性时，都是按先自有，再共有的顺序查找的。

11. 何时使用console.log(), 何时使用console.dir()

答: 需要详细了解对象在内存中的存储结构和内容时，就用console.dir()

如果只是一般的输出对象的普通内容，就用console.log()。

有时console.log会自动调用对象的toString但是不是总是。

12. 为什么new Student()生出的lilei和new Array()生出的数组，以及new Date()生出的日期对象，内部存储结构不一样？

答：子对象的存储结构具体是什么样，其实是由构造函数妈妈决定的。构造函数妈妈肚子里规定了哪些属性，将来孩子身上就会长出哪些属性。

比如，function Student(){}妈妈中规定，所有学生孩子都有sname属性和sage属性两个属性，不是三个属性，也不是四个属性，属性名不叫name，也不叫age。这些都是构造函数妈妈肚子里规定的。

同理，别人家的构造函数妈妈肚子里也规定了别人家的孩子的属性结构和类型。不同的构造函数妈妈肚子里规定的数据结构，属性名，属性个数，都是不一样。所以，用不同构造函数妈妈生出的孩子，长的都不一样。

但是，因为Array()和Date()，是浏览器内置的，使用浏览器底层代码实现的。所以，我们看不到源代码。只能想象。

13. 构造函数和原型对象之间的关系是什么意思，为什么构造函数和原型对象之间要有关系?

答: 构造函数通过prototype属性指向自己的老公——原型对象

原型对象通过constructor属性指向自己的老婆——构造函数

因为开发时，很可能需要随时找到构造函数和原型对象

如何找原型对象？原型对象本身没有名字，不能直接找到。要找原型对象，只能通过先找到构造函数，再通过构造函数的prototype属性，才能找到隐藏的原型对象

万一不知道构造函数的情况下，想找到一个孩子的妈妈:

先找到孩子，再找到还的爸爸: 孩子.\_\_proto\_\_

通过孩子的爸爸，才能找到孩子的妈妈：

孩子.\_\_proto\_\_.constructor

14. 将Array.prototype.sum=function(){ ... } 换成

arr1.\_\_proto\_\_.sum=function(){ ... } 为什么报错？

答: 这个时候有arr1吗？通常添加原型对象方法时，还没有孩子呢！

15. 上午讲的子对象强行修改父对象中共有属性，导致自动在子对象添加自有属性，是不是多态？

答: 不是。多态，专指函数在不同情况表现出不同的状态。和属性无关。

16. 匿名函数自调前为什么用+，!，~，都能代替()呢？

比如: +function(){ ... }()

!function(){ ... }()

~function(){ ... }()

都能匿名函数自调？

答：+ ! 和~都是运算符，运算符后跟表达式时，js引擎会先试图执行运算符后的表达式，获得结果，然后在将结果和之前的运算符组合，进行下一步计算。所以，当匿名函数前跟+ ! 或跟~时，其实是在诱导js引擎去执行后边的函数表达式，就相当于调用了。

17. Array.prototype.sum=function(){ ... }中, Array是爷爷？还是爸爸？还是类型？

答：看图：Array是构造函数妈妈： function Array(){ ... }  
 Array.prototype是原型对象爸爸，是Array的老公。

类型：是指由妈妈+爸爸组成的一个“家庭”称为一个类型。反过来说，每一个类型都是由构造函数妈妈和原型对象爸爸组成。

18. arguments是不是多态？

答：js中实现重载，要借助于arguments。而重载属于多态的一种情况。所以重载这种现象是多态。而arguments本身只是实现多态所借助的一个对象。

19. var a=10;  
 function fun (){

console.log(a)

}

fun();

这里函数内可以使用外部的a是不是继承？

答：不是。一个函数执行时，可用的作用域对象，都保存在一个scopes数组（好友列表）中。查找变量时，是按顺序遍历scopes数组中的每个作用域对象，查找变量的。属于遍历。

而继承是通过\_ \_proto\_ \_属性，逐级查找父级对象中的属性。而不是变量。继承必须先有继承关系，才能通过\_ \_proto\_ \_查找父对象的属性。

20. 为什么Array.prototype.sum=function(){}, 放在上边的<script>就能用。而Student.prototype.toString=function(){} 放在上边的<script>就不能用？

答：执行顺序问题！

Array类型是内置类型，你不用创建function Array(){}，内存中已经有了。所以你可以直接使用Array.prototype.sum=function(){}。

而Student类型是你自己定义的，你不定义，内存中，就没有。所以，Student.prototype，只能在创建function Student(){}之后才能使用。

21. Cannot set property 'sname' of undefined ?

答: 不是sname写错了，而是sname .前的那个值，本应该是一个对象，但是因为各种原因，没拿到，是undefined。而undefined加任何后缀都会报错！

解决: 看sname前的变量是谁！往前找，哪里给这个变量赋的值。

22. 子对象继承来的属性，会不会变成自己的

答: 子对象继承来的属性和方法，不会变成自己的属性和方法。只不过，当自己没有该属性和方法时，可以自动去父对象中找到该方法或属性使用。用完，这个共有的属性和方法依然保存在父对象中。不会成为对象自有。

23. in为什么只找到爹就停止了？爷爷的toString和那么多方法，都找不到？

答: in一定会一直查找到顶级父对象中。

但是，各级对象中都有浅紫色的方法或属性，这些浅紫色的方法或属性是半隐藏的：

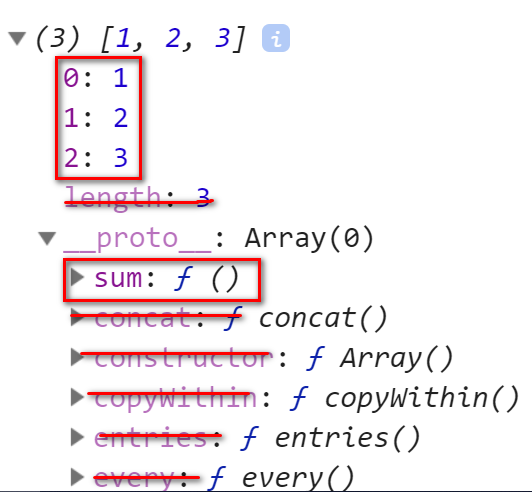
只有in看不到！用.却可以随时访问！

24. 为什么Array.prototype.sum中的for循环改成for in，就错了？

答: for in循环专门遍历自定义下标名称的关联数组或对象。绝对不应该用for in 循环来遍历索引数组！

因为, in这个词，不仅遍历当前对象的数字下标，而是遍历当前对象的所有成员以及原型链上的所有成员。除了那些浅紫色的属性不遍历（浅紫色的属性屏蔽in），别的都会被in遍历到。

比如下图中，用in遍历这个数组，就会遍历0 1 2 和原型对象里的sum。



但是，如果求和计算时，将sum函数也一起求和，就会出错。

所以，今后，只要遍历数字下标的索引数组和类数组对象，必须只能用for循环，不能用for in循环。

只要遍历自定义下标的关联数组和对象时才能用for in

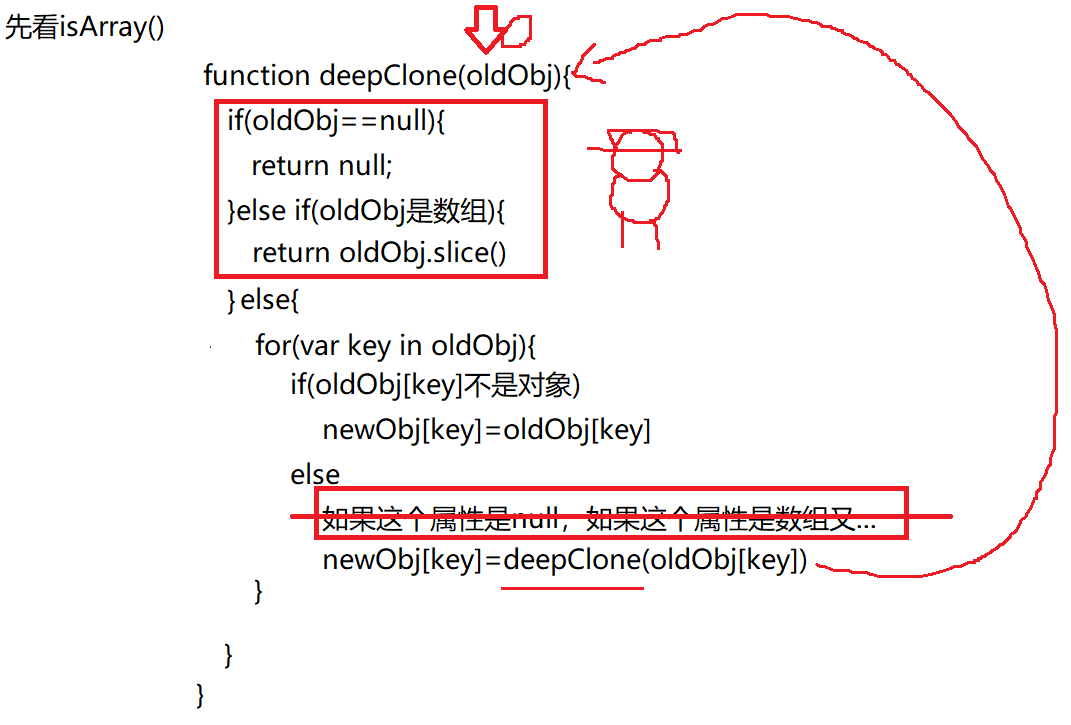
25. Student.prototype.className可以修改所有子对象的班级名，从lilei.\_\_proto\_\_.className也可以修改所有子对象的班级名？今后如何选择？

答: 在软件设计过程中，都是先设计原型对象，再创建子对象。所以，在设计原型对象内容时，还没有子对象呢！所以，通常Student.prototype.className用的绝对多。从另一个角度来说，应该尽量避免用子对象修改父对象的成员。

26. 深克隆中，为什么在函数一开始就判断oldObj==null，为什么不是在遍历到某个属性时判断？

答: 如果在深克隆函数开头就判断是不是null和数组，既可防止首次调用就传入不好的对象的情况，又可防住之后递归时，传入的不好的属性值。因为递归调用还会回到函数顶部执行。

如果只在遍历时，判断某个属性，则无法防住第一次就给不好的对象的情况



27. Object.prototype.toString.call()和{}.toString.call()有啥差别？

答: 本来抢的就是同一个函数。内存中只有一个toString保存在Object.prototype里。所以Object.prototype.toString()完全可以访问到顶级原型对象中的toString。但是{}是创建一个对象的意思，{}直接继承自Object.prototype，所以{}.toString()也表示顶级父对象Object.prototype中的toString()，因为孩子可直接使用爹中的方法。那为什么用{}.toString()。因为简单！

比喻，本来想抢爹的东西，但是打不过爹，于是就从他家小孩儿下手。让小孩儿把他爹的东西拿出来给咱们用！

28. function Student(sname){

this.sname=sname;

this.className="初一2班"

}

var lilei=new Student("Li Lei");

var hmm=new Student("Han Meimei");

这样也能实现lilei和hmm同班的效果，和Student.prototype.className="初一2班"，有什么差别？

答: 构造函数中的所有属性，最终都会加入到每个子对象内部成为每个子对象的自有属性。而自有属性无法实现，一次修改多个对象同时应用的效果。这里的this.className虽然在一开始时，多个子对象相同，但是无法实现过一年大家一起升一级的效果。因为this.className是每个子对象的自有属性。如果希望一次更改，多个对象都应用时，只能将属性放在原型对象中变成共有属性。只有Student.prototype.className="初一2班"，才能实现过一年大家一起升一级的效果！一次更改，多个对象同时应用！

## day06

1. 如何从ftp上下载html和js文件?

答: 右键->链接另存为

2. 闭包的外层函数向外抛出内层函数共有几种方式?

答: 3种:

1. return function(){ ... }

2. 全局变量=function(){ ... }

3. return [ function, ...]

return { 方法: function(){ ... } }

3. 为什么lilei.intr()可以调用到原型对象中的intr()函数?

答: 因为原型对象是lilei的父级对象。程序中规定，调用函数时，先在对象本地查找。如果对象本地没找到，就自动延\_ \_proto\_ \_向父对象找。

4. 到底怎样创建一种类型?

答: 2步:

1. 先定义构造函数:

function Student(sname,sage){

this.sname=sname;

this.sage=sage;

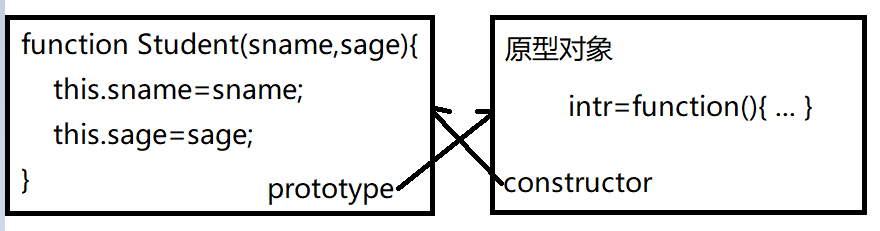
}

2. 为构造函数的原型对象强行添加共有方法:

Student.prototype.intr=function(){

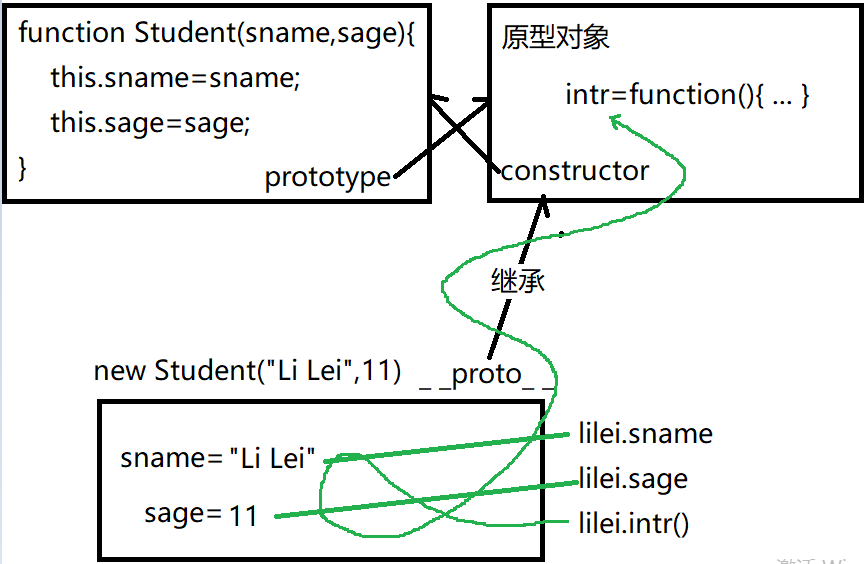
console.log(`我是 ${this.sname}, 我今年 ${this.sage} 岁`)

}



创建类型后，如何用类型，创建子对象:

var lilei=new Student("Li Lei",11)



5. 都指向原型对象，为什么一个属性叫prototype，一个属性叫\_ \_proto\_ \_?

答: prototype和\_ \_proto\_ \_指的是同一个对象，都是构造函数的原型对象。

和现实一样，以不同的角色去访问原型对象时，必须用与自己身份相符的称呼。必须有所区分。

比如: 妈妈称呼爸爸，就叫老公

孩子称呼爸爸，就应该叫爸爸！

6. 原型对象中的this->?

答: 判断this一定不好看定义在哪里，保存在哪里

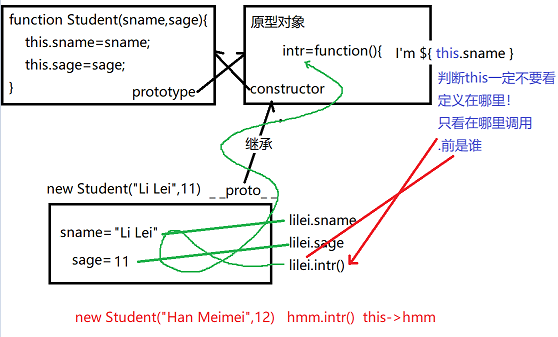
只看在哪里调用！看.前是谁

Student.prototype.intr=function(){ ... this.sname ... }

虽然存在原型对象中，但是不是在原型对象中被调用的

而是在稍后，由子对象调用: lilei.intr() hmm.intr();

所以，原型对象中的方法中的this，指将来调用该函数的.前的子对象。



7. 一共讲了几种this的情况?

答: 4种:

1. obj.fun() -> this->obj

2. fun() 或(function(){ ... })() this->window

3. new fun() -> this->新对象

4. Student.prototype.intr=function(){ ... this.sname ...}

this-> 将来调用intr()的.前的子对象

8. function Student()是一个函数，为什么能.prototype属性？

答：js中一切都是对象，函数也是对象。只要是对象，都可以用.访问属性。都可以添加属性。其实，函数和数组，和lilei一样都是存东西的对象。只不过，函数中，存的是代码段，而数组和lilei中存的是数据而已。

9. 何时使用prototype，何时使用\_ \_proto\_ \_?

答: 1. 向原型对象中添加共有属性和方法时，必须通过构造函数妈妈找到原型对象爸爸才能，向其中添加共有属性和方法。

类型名.prototype.共有属性=值

2. 如果通过儿子，查看自己爹里的内容时，可以通过子对象.\_ \_proto\_ \_获得儿子的爹。

其实，向原型对象中添加共有属性，也可以通过子对象.\_ \_proto\_ \_.共有属性=值，强行添加，但是通常，向原型对象中添加共有成员时，还没有子对象呢。所以，很少用子对象.\_ \_proto\_ \_去操作父对象。

10. 如果孩子擅自修改了共有属性，导致在自己身上添加了同名自有属性，那么，如果调用原型对象中的方法时，方法中刚好用到这个共有属性的话，方法使用的是共有的还是自有的属性？

答：方法使用的是自有的同名属性。因为，访问属性使用的是this.属性名。 this查找属性时，都是按先自有，再共有的顺序查找的。

11. 何时使用console.log(), 何时使用console.dir()

答: 需要详细了解对象在内存中的存储结构和内容时，就用console.dir()

如果只是一般的输出对象的普通内容，就用console.log()。

有时console.log会自动调用对象的toString但是不是总是。

12. 为什么new Student()生出的lilei和new Array()生出的数组，以及new Date()生出的日期对象，内部存储结构不一样？

答：子对象的存储结构具体是什么样，其实是由构造函数妈妈决定的。构造函数妈妈肚子里规定了哪些属性，将来孩子身上就会长出哪些属性。

比如，function Student(){}妈妈中规定，所有学生孩子都有sname属性和sage属性两个属性，不是三个属性，也不是四个属性，属性名不叫name，也不叫age。这些都是构造函数妈妈肚子里规定的。

同理，别人家的构造函数妈妈肚子里也规定了别人家的孩子的属性结构和类型。不同的构造函数妈妈肚子里规定的数据结构，属性名，属性个数，都是不一样。所以，用不同构造函数妈妈生出的孩子，长的都不一样。

但是，因为Array()和Date()，是浏览器内置的，使用浏览器底层代码实现的。所以，我们看不到源代码。只能想象。

13. 构造函数和原型对象之间的关系是什么意思，为什么构造函数和原型对象之间要有关系?

答: 构造函数通过prototype属性指向自己的老公——原型对象

原型对象通过constructor属性指向自己的老婆——构造函数

因为开发时，很可能需要随时找到构造函数和原型对象

如何找原型对象？原型对象本身没有名字，不能直接找到。要找原型对象，只能通过先找到构造函数，再通过构造函数的prototype属性，才能找到隐藏的原型对象

万一不知道构造函数的情况下，想找到一个孩子的妈妈:

先找到孩子，再找到还的爸爸: 孩子.\_\_proto\_\_

通过孩子的爸爸，才能找到孩子的妈妈：

孩子.\_\_proto\_\_.constructor

14. 将Array.prototype.sum=function(){ ... } 换成

arr1.\_\_proto\_\_.sum=function(){ ... } 为什么报错？

答: 这个时候有arr1吗？通常添加原型对象方法时，还没有孩子呢！

15. 上午讲的子对象强行修改父对象中共有属性，导致自动在子对象添加自有属性，是不是多态？

答: 不是。多态，专指函数在不同情况表现出不同的状态。和属性无关。

16. 匿名函数自调前为什么用+，!，~，都能代替()呢？

比如: +function(){ ... }()

!function(){ ... }()

~function(){ ... }()

都能匿名函数自调？

答：+ ! 和~都是运算符，运算符后跟表达式时，js引擎会先试图执行运算符后的表达式，获得结果，然后在将结果和之前的运算符组合，进行下一步计算。所以，当匿名函数前跟+ ! 或跟~时，其实是在诱导js引擎去执行后边的函数表达式，就相当于调用了。

17. Array.prototype.sum=function(){ ... }中, Array是爷爷？还是爸爸？还是类型？

答：看图：Array是构造函数妈妈： function Array(){ ... }  
 Array.prototype是原型对象爸爸，是Array的老公。

类型：是指由妈妈+爸爸组成的一个“家庭”称为一个类型。反过来说，每一个类型都是由构造函数妈妈和原型对象爸爸组成。

18. arguments是不是多态？

答：js中实现重载，要借助于arguments。而重载属于多态的一种情况。所以重载这种现象是多态。而arguments本身只是实现多态所借助的一个对象。

19. var a=10;  
 function fun (){

console.log(a)

}

fun();

这里函数内可以使用外部的a是不是继承？

答：不是。一个函数执行时，可用的作用域对象，都保存在一个scopes数组（好友列表）中。查找变量时，是按顺序遍历scopes数组中的每个作用域对象，查找变量的。属于遍历。

而继承是通过\_ \_proto\_ \_属性，逐级查找父级对象中的属性。而不是变量。继承必须先有继承关系，才能通过\_ \_proto\_ \_查找父对象的属性。

20. 为什么Array.prototype.sum=function(){}, 放在上边的<script>就能用。而Student.prototype.toString=function(){} 放在上边的<script>就不能用？

答：执行顺序问题！

Array类型是内置类型，你不用创建function Array(){}，内存中已经有了。所以你可以直接使用Array.prototype.sum=function(){}。

而Student类型是你自己定义的，你不定义，内存中，就没有。所以，Student.prototype，只能在创建function Student(){}之后才能使用。

21. Cannot set property 'sname' of undefined ?

答: 不是sname写错了，而是sname .前的那个值，本应该是一个对象，但是因为各种原因，没拿到，是undefined。而undefined加任何后缀都会报错！

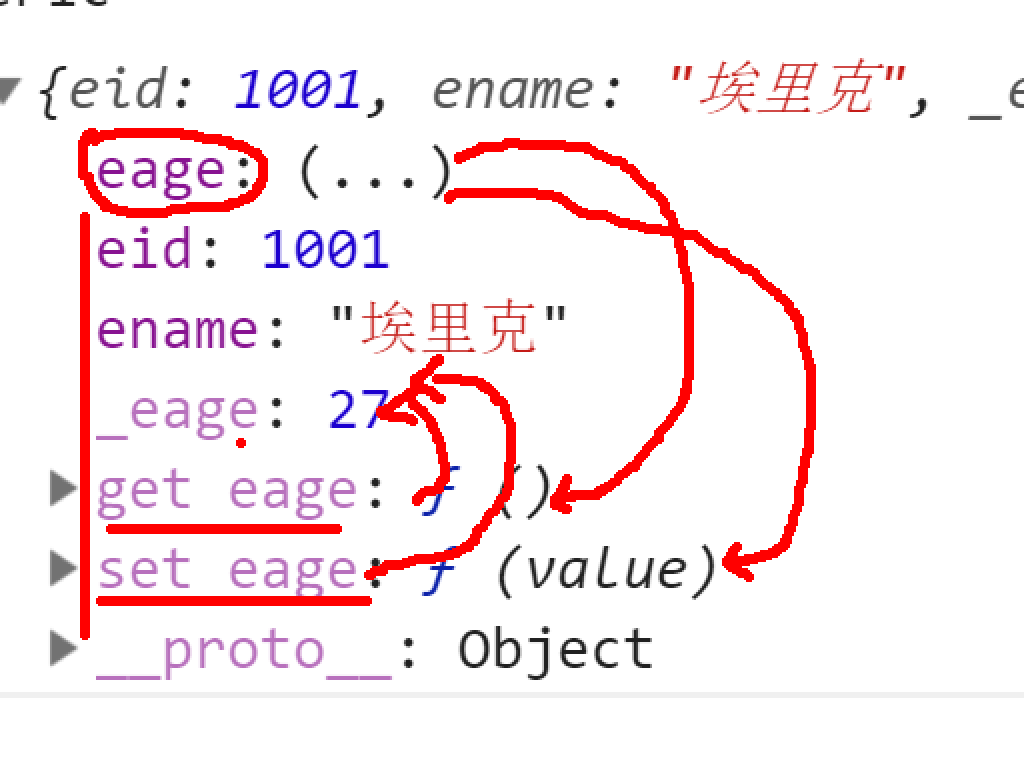
解决: 看sname前的变量是谁！往前找，哪里给这个变量赋的值。

22. 访问器属性中的this，到底指谁？为什么必须加this？

答： 访问器属性中的this，指的依然是当前对象自己。

之所以访问器属性中，访问受保护的数据属性，必须加this.，是因为:

get()和set()两个函数，编写时，虽然写在访问器属性内部，但是，js编译时，悄悄将get()和set()，编译成了和\_eage平级的方法。那么，\_eage平级的方法中，要访问\_eage属性，都必须加this.



23. eage属性怎么知道自己保护的是\_eage属性？

答: 在get()和set()内，我们写的都是this.\_eage，也就是从取值是从\_eage里取的。改值也是改回\_eage中。

24. bind()，能不能只绑定第二个形参变量

答: 不能跳过第一个形参变量，只能从第一个形参变量开始向后绑定。如果非要绑定第二个形参变量，可以将第二个形参变量提前到第一个形参的位置。因为形参变量本身是没有顺序规定的。

25. bind() 能不能绑定多个形参变量:

答: 只要从头开始按顺序绑定，可以绑定多个形参变量的

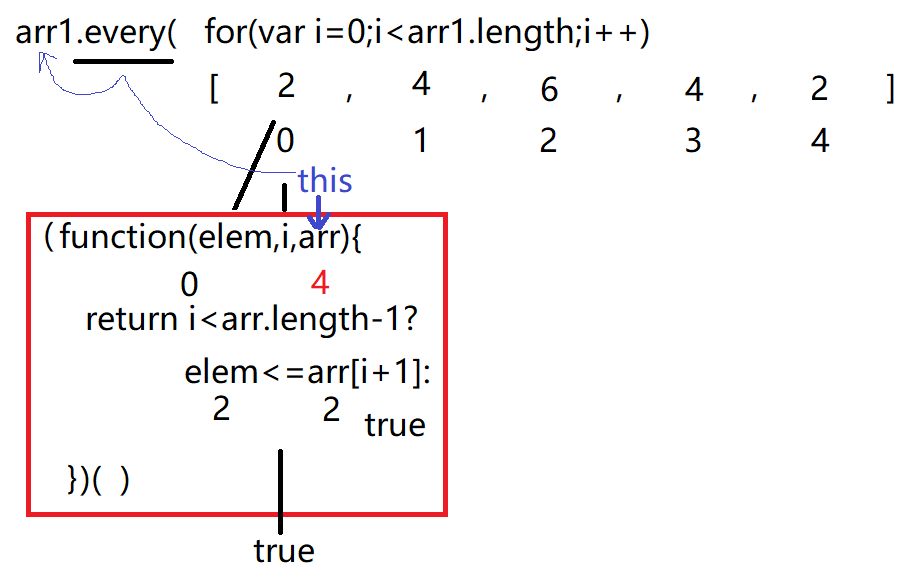
26. arr1.every(function(elem,i,arr))中，回调函数要想获得every.前的当前数组对象，为什么不能写死arr1或用this

答: 形参变量注定是局部变量，形参变量的值完全依靠外部传入。就算把形参arr，改成arr1，也无法找到外部的arr1数组。

所以，改成arr1和arr都是一样的效果

又因为回调函数中的this->window，所以在回调函数内不可能用this获得外部主函数.前的arr1对象。

但是，因为every()也是arr1对象的一个函数，所以every中的this是自动指向.前的arr1对象的。而我们可以想象作为主函数的every()，一定在内部自动将every的this，临时传递给了回调函数的形参arr，才让arr变量获得了every().前的arr1



27. var eric={ ename:"埃里克" }  
 Object.defineProperties(eric,{  
 \_eage:{ … },  
 eage:{ … }  
 })  
 为什么\_eage和eage最后都跑到eric对象里了？

答: Object.definePropeties()函数时重定义对象的一个属性的意思。如果要重定义的属性已经包含在对象中，则直接修改现有属性。如果要重定义的属性没有包含在对象中，则Object.defineProperties()函数会为对象自动添加该属性。

28. 为什么eric.\_eage=-2，就能绕过访问器，成功修改年龄为-2？

答: js的访问器属性其实并不完善，因为\_eage的enumerable:false，只能防住for in, 防不住点。所以，外人如果知道有\_eage的话，还是可以直接通过eric.\_eage=-2来篡改。

29. 为什么\_eage中writable:true, enumerable:false, configurable:false里true和false的属性都要写？开关属性值不是默认是true吗？

答：开关属性不总是默认为true，有时是true，有时是false。所以，为了不受制于人，干脆，所有开关的值都自己来定义。

其实， 用{}直接添加的属性，开关默认值都是true，比如：var eric={ \_eage:25 }  
 而用Object.defineProperty()添加的新属性，开关默认值都是false。

所以，最好都自己写开关值，不要信默认值。

## day07

1. arr1.every(function(elem,i,arr){ })内，如果想使用.前的arr1，为什么不能用this直接找到.前的对象，而只能用形参中的arr呢？

答：所有回调函数的执行，都是自动执行的，且没有.前缀。所以，绝大多数回调函数中的this，都指window。

再比如:

var lilei={

sname:"Li Lei",

intr:function(){ console.log(`I'm ${this.sname}`)}

};

lilei.intr(); //this->.前的lilei，所以可以正常输出lilei.sname

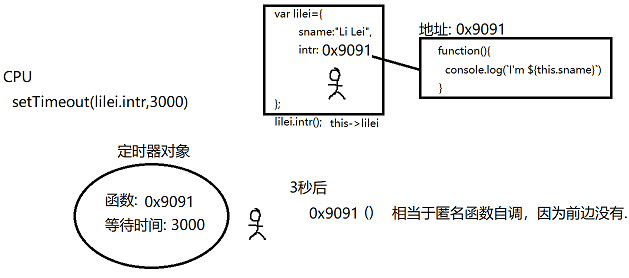
但是:

setTimeout(lilei.intr,3000); //3秒后输出的却是undefined

1. 为什么lilei.intr不加()？因为定时器不是立刻调用。而仅仅是给定时器拿走，等3秒后，由定时器自动调用。

2. 为什么输出undefined，而不是lilei.sname?

答: 因为定时器中的函数，也是回调函数。定时器在拿走函数时，只是单纯的拿走了函数的地址，而没有同时拿走.前的对象lilei。3秒后执行函数时，是直接用函数的地址加()调用的函数。早就不知道函数的所有者是谁了。仅相当于匿名函数自调。匿名函数自调中的this->window，而window.sname当然返回undefined。



2. 什么是回调函数(callback)?

答: 咱们自己定义的函数，却不由咱们自己亲自调用，而是交给别的函数去调用。

比如: setTimeout(function(){},3000)里的function(){}，虽然是咱们定义的，但是调用时setTimeout等三秒后调用的。

再比如: arr.every(function(elem,i,arr){})里的function(){}，虽然是咱们定义的，但是，调用几次咱们不知道，每次传入什么实参也不知道。完全由every()在内部自动调用。

还有arr.sort(function(a,b){}), str.replace(/正则/g,function(kword){}), xhr.onreadystatechange=function(res){}，这些都是回调函数。

3. 何时加()，何时不加()?

答: 主动想调用一次函数时，就要加()

不调用函数，只是将函数给别人拿走，将来由别人负责调用时。不加()!

4. 0.1+0.5==0.6 为true，而 0.1+0.2==0.3为什么为false？

答：舍入误差: 计算机中底层采用二进制计算，也有计算不尽的数值。就好像现实中10/3得几？3.3333333333....说几个3也无法精确。因为根本除不尽。计算机中的二进制更过分，有加不尽，减不尽，乘不尽。比如:

2-1.6=0.3999999999999999...

555.55\*100=55554.99999999999

等等，不确定什么时候，这种舍入误差就会出现。

只要是浮点数参数算数计算都有可能产生舍入误差，但是和十进制一样，不是所有情况都舍入误差。比如: 10/5是可以除尽的。10/2也是可以除尽的。所以，舍入误差不可预期。

现实中解决舍入误差：计算式保留过精度小数位数，比如保留8位小数进行计算。包括数据库中存储数字，也用八位小数。只有要显示给用户看时，再用toFixed()，按2位小数或者按整数四舍五入。

5. [{x:1},{x:2},...]数组中保存的实际上是对象的地址。但是[1,2,3]里保存的也是1,2,3的地址吗？

答: 其实数组中的一个元素，也是一个变量，只不过是放在数组中的一个变量而已。只要是变量，都只能存一个值。如果是原始类型的值，可直接保存在变量内。如果引用类型的对象，则只能保存在变量外部，通过地址值引用对象。

同理，对象的属性，其实也是一个变量，只不过保存在对象内部而已。

[{x:1},{x:2},...] 里边的对象，其实是存在数组外部的，每个元素只保存对象的地址。

[1,2,3]里边的原始类型的数值，其实是直接保存元素内部的。

6. 匿名函数和回调函数有什么差别和关系?

答: 回调函数不一定是匿名函数，匿名函数也不一定是回调函数。

只有当回调函数只使用一次时，才会定义为匿名函数。如果回调函数可能反复使用，也要定义为有名称的函数。

比如: setTimeout(function(){ ... }, 3000)

如果这句话需要反复使用:

不好的做法: 将setTimeout(连同里边的function(){}定义，反复复制多次！

就应该: 只定义一次函数，然后再定时器中用函数名反复引用函数:

function task(){ ... }

setTimeout(task,3000)

setTimeout(task,2000)

setTimeout(task,1000)

总结: 是否使用匿名函数，取决于将来这个函数被几处使用。如果只被一处使用，则就用匿名函数。如果多处都需要使用这个函数，则必须起名！

另一方面: 匿名函数也有主动自调的情况，就不属于回调。

总结: 之所以大多数回调函数都是匿名函数，只是因为多数回调函数只在一个位置使用一次！所以，没必要起名。

7. 闭包中的内层函数，算不算回调函数?

答: 不算！外层函数生出的内层函数，需要我们自己主动调用。调用几次，就反复执行几次。每次调用时，也需要我们主动传入实参值。

8. 数组新函数何时只用一个elem形参，何时三个形参都要用?

答: 首先，无论你写几个形参，人家一定会在调用时，自动给你三个实参值。如果你不接，大不了就不要了！

具体定义几个形参，取决于本次操作所需几个数据才能正常执行。

如果只靠elem一个值就可正常执行全部逻辑，就不要定义后两个形参。

如果只靠elem不能完成所有操作，必须还依赖位置i和当前数组对象arr，则必须提供三个形参。

9. 两次统计都用total作为变量名，难道不冲突吗？能不能起两个变量名？

答: 从语法上来说，是不冲突的。无非就是给一个变量，先后赋了两次值而已。

从用法上来说：如果整个程序只有一个捐款箱，最好只起一个名字。如果两个班的捐款分开统计，也就是有两个捐款箱时，反而应该起两个名字。

10. arr.reduce(function(prev,elem){

...

return 的是什么东西？

},

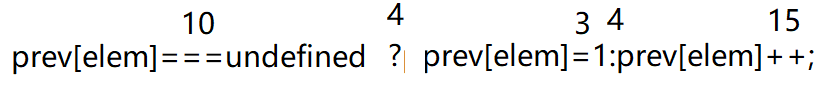
{}

)

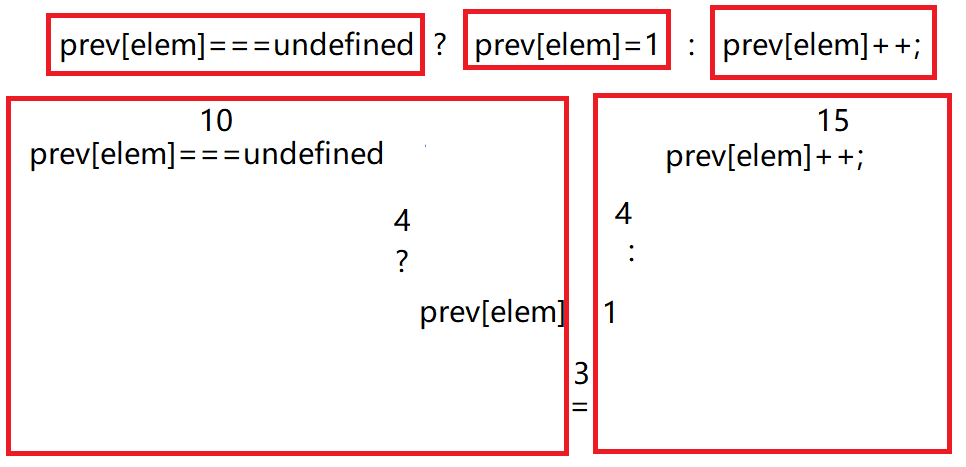
答: return的必须是修改后的整个捐款箱替换回去。而不应该是return一个表达式。所以 return的只能是prev

11. prev[elem]===undefined?prev[elem]=1:prev[elem]++;是错的？

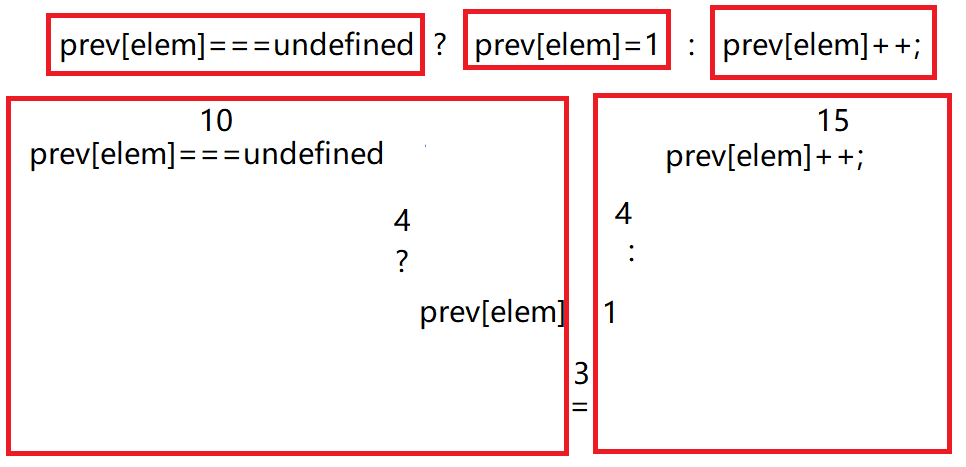
答: 因为运算符优先级导致程序自动断句错误！



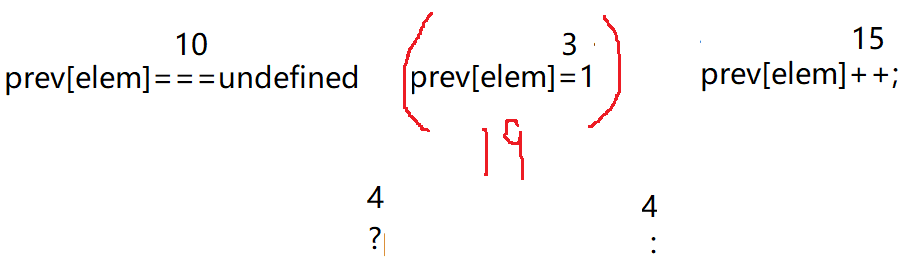
会被断句成:



但是我们希望



解决: 给希望整体优先执行的表达式加()，因为()优先级永远最高！

如何: 

12. prev.elem不对？

答: 复习！复习对象克隆！

13. 为什么reduce的第二个参数要用{}?

答: 这是由最终希望获得的结果的数据结构定的。

最终希望获得: h:1, e:1, l:3, o:2, w:1, r:1, d:1

在程序中保存键值对的集合，最好的办法是{}，其次可以用[]，因为在js中{}底层就是[]，两个的用法是完全一样的！

14. let t=0后，其余var s=10; 或function task1(){}，会不会也和t一样变成局部呢？

答：不会！let t一定会尽量保证不影响其他无关的变量或函数。比如，可以自动通过将var s=10，换成window.s=10，或者自动将function task1(){}改为window.task1()=function(){}等来保证其他非let的变量或函数，依然属于全局。

15. 箭头函数中，无法使用arguments

答: 解决: rest语法：

var add=(...nums)=>{

return nums.reduce((prev,elem)=>prev+elem,0)

}

16. arr.forEach(function(elem){ ... this ... })中的this为什么指window，而不是forEach.前的arr?

答: this是在回调函数中的，回调函数时在forEach内部被自动调用，调用时，回调函数之前什么.都没有，所以，this找不到要指的对象。按照规定，默认只能指向window

17. arr.forEach(function(elem){ ... this->window ... })，如果替换回调函数中的this为想要的对象，为什么不能用call，只能用bind？

答: call()是主动调用一次函数。而回调函数，恰恰是不由自己主动调用，且反复调用的次数也无法确定，也不止一次。所以不符合call的应用场景，更不符合apply的应用场景。

bind()不是调用函数的意思，而是创建并返回一个新的函数副本。用.bind()完全可以基于原函数先创建一个副本，再将函数副本给主函数forEach去反复使用。

结论: 今后只要希望替换回调函数中的this，都用bind()，而不用call()

18. 回调函数和匿名函数有什么差别:

答: 回调函数是指一切由我们创建的函数，却传给别的函数中，由别的函数自动调用的函数都称为回调函数。

回调函数既可以是有名的函数，也可以是匿名函数

比如: arr.sort(function(a,b){ return a-b; });

也可以写成: var fun=function(a,b){return a-b}

arr.sort(fun)

两种方式都是将一个函数，传入arr.sort()内，由arr.sort()自动按需反复调用。只不过，匿名函数方式是直接插入，而arr.sort(fun)是先将函数保存在变量中，再将变量传给arr.sort()

而匿名函数，是指一切创建函数时，没有指定函数名的函数。回调函数推荐以匿名函数方式传入，是因为没有名称的函数用完就释放，节约内存。但是，匿名函数不止用于回调函数。还可用于匿名函数自调来划分临时作用域，避免产生全局变量。比如，我们要求所有js代码都应该放在一个匿名函数自调中，来避免产生全局变量。这个匿名函数自调就不是回调函数，而是我们自己主动调用的一个函数。

19. fromi=fromi||0，按理说，逻辑运算应该返回bool值才对呀？为什么会返回fromi的值或者数字0呢？

答: 逻辑运算有一个特殊的情况，就是，如果逻辑运算前后，都是值时，就不再返回bool值，而是先尝试转bool类型做判断，然后，决定具体返回左边还是右边的值

比如: 1||0， 返回1，而不是true

1&&0 返回0, 而不是false

20. total>=500&&(total\*=0.8)，可不可以换成三目？

答: 按语义分支结构分为三种情况:

1. 一件事，满足条件就执行，不满足条件就不执行

if实现: if(条件){ 操作 }

比如: if(total>=500){ total\*=0.8 }

可简写为: total>=500&&(total\*=0.8)

2. 两件事，根据条件二选一执行

if实现: if(条件){操作1}else{操作2}

比如: if(score>=60){ 输出及格 }else{ 输出不及格 }

可用三目简写: score>=60?输出及格:输出不及格

3. 多件事，根据条件多选一执行

if实现: if(条件1){操作1}

else if(条件2){ 操作2 }

...

else { 默认操作 }

比如: if(score>=90){ 输出A }

else if(score>=80){ 输出B }

else if(score>=60){ 输出C }

else { 输出D }

可用三目简写:

score>=90?输出A:

score>=80?输出B:

score>=60?输出C:

输出D

所以，选择何种简写，需要根据其语义来选择。如果只有一件事，选择做与不做，最好使用短路，而不是三目。三目至少要有两件事，二选一执行时，才适用。

21. arr.forEach((elem,i,arr)=>arr[i]\*=2) elem没用到，能不能省略？

答: 不行。因为回调函数是自动调用。每次调用时外围的主函数都会按约定顺序传入三个值: 元素值，元素位置，数组对象。对应传给elem, i和arr三个形参。如果省略elem，则传参会发生错位:

元素值, 位置, 数组

( i , arr )

但是，如果后两个形参不想要可省略:

元素值, 位置, 数组

( elem 不要 不要 )

22. 为什么prev[elem]!==undefined 可判断对象中是否包含当前字母的记录?

答: str.split("").reduce(function(prev, elem){

//每次遍历时:

//prev得到保存临时汇总值的结果对象{ h:1, e:1,... }  
 //elem得到字符串中每个字符: "h" "e" "l" "l"...

//需求中要求判断在prev对象中，是否包含以elem所存字符为属性名的记录.——如何判断对象中是否包含某个属性?

//强行访问: obj["属性名"]

//如果obj中包含该属性名则一定不会返回undefined

//如果obj中不包含该属性名则一定返回undefined

//所以: obj["属性名"]!==undefined，就成为判断该对象中是否包含该属性的判断条件。

//放在本利中: 要判断prev对象中是否包含elem中所存字符的属性名: prev[elem]!==undefined。

//elem不加引号，是因为elem是一个变量。

//优雅的判断对象中是否包含某个属性的办法:

// var bool="属性名" in 对象

//本例中: var bool=elem in prev

//in 会在当前对象内部，以及当前对象的原型链上查找是否包含该属性名。如果包含，返回true，如果不包含，返回false。

}, {} )

23. intr:function(){

//this->lilei  
 this.friends.forEach(

function(){  
 this->window  
 }.bind(this)

)  
 }

蓝色this也在forEach内，为什么指外边的lilei？

答: this.friends.forEach(

内←外 function(){  
 内←外 this->window  
 内←外 }.bind( 内 ) ← 外this

)  
24. Array.prototype.slice.call(arguments)为什么就是抢slice?

答: arguments不能直接使用slice，因为arguments不是数组家人。

call()的作用域是替换slice中不想要的this，而this指的是.前的对象。.slice.call(argumetns)虽然是在换slice里的this，等效于将.slice 点前的对象同时换掉。所以，临时等效于arguments.slice(...)

加Array.prototype，是为了找到保存在数组家原型对象中的slice函数。只有找到，才可能去抢。

## day08

1. 在ES5中，如何实现多个形参不确定，还要求参数值与形参变量一一对应?

答: function order(menu){

//menu: { zs:xxx, tang:xxx, td:xxx, rc:xxx, sc:xxx}

menu.zs...

menu.tang...

}

order({ zs:xxx, tang:xxx, td:xxx, rc:xxx, sc:xxx})

2. function Fun(){

var i=0;

return function(){

console.log(i++);

}

}

var f1=Fun();

var f2=Fun();  
 f1(); //0

f1(); //1

f2(); //0 不是2?

f2(); //1 不是 3?

答: 妈妈分两次分别生了两个孩子，每次生孩子都给孩子包一个专属的红包。所以，f2和f1用的不是一个红包。他们分别有自己专属的红包。

3. .then()中的函数为什么不加()也能执行？谁调用的？

答：liang().then(ran)，其实还是回调函数的方式，先把ran放入.then中留着备用。只有当前一个liang中调用了door()打开门后，才自动找到.then()中提前放好的ran，此时才自动加()调用。

4. liang().then(ran).then(dong).then(liang)为什么不会形成死循环?

答: 每次调用一个函数，都是创建并返回一个新的格子间。新的格子间，如果后边没有接then,就不会有下一步后续任务。

这里liang() ... .then(liang)等于先后调用了两次liang，返回过两个格子间。而第二次调用liang返回的格子间绝对不等于第一个格子间，且第二个格子间后边没有.then。所以，第二次调用.then(liang)后，就没有后续步骤了。

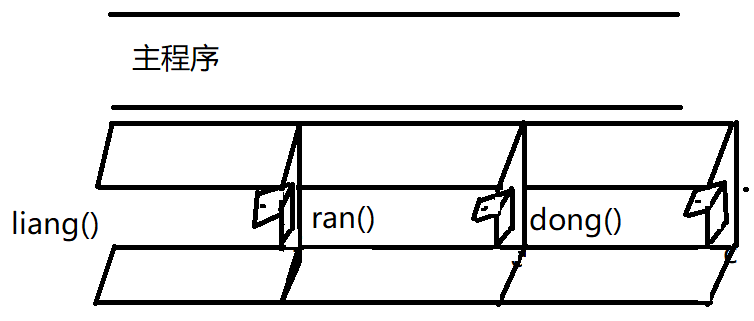
5. ran不加()也能调用，是.then()的功劳吗？

答？不只是then。

所有.then,其实起到的是收集函数并排队的作用。比如: liang().then(ran).then(dong)，等效于: [liang, ran, dong]顺序存储好。然后，依次自动调用每一个函数，如果前一个函数调用了door，才i++，自动调用第二个。如果第二个调用了door，才i++，自动调动第三个。所以ran的执行，是因为被.then()收齐起来，再由上一步调用door()共同触发执行的。

6. console.log(“比赛结束”)，不放在then中，依然会在所有人到达终点前就执行？

答：是的！看图



Promise格子间本身都是异步的。第一个格子间是异步的，在主程序之外执行的，所以，就算后续格子间与第一个格子间是串联的关系，无非也是在主程序之外，串联起一个更长的异步执行过程而已。所以，由promise串起来的多个异步函数，组成的是一个大的异步任务，但与主程序相比，依然是异步的。

7. var bang=await liang(); 可不可以不加await

答: 不加虽然不会报错，但是后续程序不会等待当前函数执行完，就提前执行了。执行顺序是不是颠倒了！

8. return new Promise(

function(resolve, reject){

}

)中，resolve和reject，必须叫这两个名字吗？

答: 只要是形参，永远没有固定的名称

但是，顺序是固定的！

将来只要调用第1个开关，都表示当前函数执行成功，都走.then()

将来只要调用第2个开关，都表示当前函数执行错误，进入.catch()

9. Uncaught (in promise) 说明是人家Promise函数中用reject()抛出了错误！但是你没接住:

答: ES6: .catch(e=>console.log(e))

ES7: try{

...

}catch(e){ console.log(e) }

10. 异步是不是脱离开~~全局~~主程序，没有和主程序在一起? 如果加上await，是不是就等待了呢？

答: 问题一: 异步的确是不和主程序一起执行的

问题二: await不能随便用:

1. 这个异步函数，必须支持Promise  
 2. await外围必须套上(async function(){...})();

11. 到目前为止有哪些函数是异步的:

答: setTimeout() setInterval() 可以自己控制结束的时间

xhr.onreadystatechange=function(responseText){

}

将来:

ajax({url, type, data, dataType}).then(res=>{...})

jquery:

$.ajax({url, type, data, dataType}).then(res=>{...})

vue:

this.axios.get(url,{ params: ... }).then(res=>{ ... })

nodejs:

pool.query(sql,[...],(err,result)=>{//异步

... ...

pool.query(sql,[...],(err,result)=>{//异步

... ...

pool.query(sql,[...],(err,result)=>{//异步

... ...

res.send(result)

})

})

})

12. 如果等待多个异步任务都完成后，才执行后续操作，用ES7怎么写？如果Promise.all中抛出异常，如何处理?

答: var arr=await Promise.all([liang(), ran(), dong() ])

console.log("比赛结束")

其中: arr中: [101,102,103]

如果Promise.all中抛出异常，则依然使用try{ ... }catch(e){ ... }处理:

try{

var arr=await Promise.all([liang(),ran(),dong()])

}catch(e){

//错误处理代码

}

await代替了.then,

try{ ... }catch(e){ ... }代替了.catch()

13. 三种遍历，那种效率最高？

答：一样高！因为其实只有一种遍历，就是for循环。其它都是封装的for循环而已。

所以，ES5的其它遍历函数,indexOf, every, some, forEach, map, filter和reduce，都没有效率的提升，因为内部都是for循环而已。只是语法简单而已。

14. arguments可不可以通过认数组的原型对象当干爹来使用数组家的函数

答: 只要是数字下标的对象，都可以通过继承数组的原型对象来使用数组家的函数。

其实，还可以用.call()抢。

比如:

Array.prototype.reduce.call(arguments,(prev,elem)=>prev+elem,0)

15. reduce()去掉0，也能正常执行？

答: 标准写法都要带第2个参数

只是在极特殊情况下，如果数组中要统计的元素都是数字，则捐款箱的默认值为0——了解。

16. 是不是有了ES6，就不用ES5了？

答: 要看浏览器是否兼容ES6.

如果使用框架和微信开发，基本上可以使用最新的语法。因为框架中和微信开发中都自带翻译工具（Babel），在项目打包时，会将ES6代码翻译成ES5代码，保证浏览器兼容性。

如果没使用框架，使用原生或jquery开发，则尽量不要使用ES6。

17. ...不是打散的意思吗？这里怎么成了收集？

答: 定义函数时，形参列表中的...是收集不确定个数的实参值的意思

调用函数时，实参值列表中的...是打散数组实参为多个实参值，分别传入的意思。

18. ...作为打散数组放在实参列表中时，不一定放在结尾？

答: 是的！可以可其他单独的实参值混搭！

19. function liang(callback){ ... }

liang(function(){ ran() })

liang()调用时，实参可不可以使用函数名？

答: 可以，但是只能接一个函数，下一个函数就接不了了

比如: liang(ran) 此时callback=ran callback()=ran()

但是，因为ran不是立刻调用，不能加()，所以ran无法接下一个dong函数了。

20. liang(

function(){

ran();

}

)为什么必须套一层function()，为什么不能写成liang(ran())

答: liang(ran())，程序先执行ran()，将ran()的返回值，给liang()作为参数。但是，因为function ran(){} 是一件事儿，没有return。意味着ran()，返回undefined。就说明liang()得到的参数是undefined。而liang()要的是一个可执行的函数对象。

liang(ran)，是对的。ran没有加()，说明不是立刻调用执行。而是将ran这个函数对象本身交给liang()作为参数。这符合liang()的要求。但是，这样做的缺陷是，ran不能加()，也就无法接第三个函数。

解决: liang(

function(){

下一步要执行的js语句

ran(); //好处: ran()中可接下一个函数

}

)

21. 都已经百米赛跑了，为什么每个人还需要door()开门呢？

答: 整个比赛结束的标准是每个人都比赛结束了

而其中仅任何一个人开门，都无法代表别人也结束比赛了。所以，只有每个人都打开了自己的门，才说明整个比赛结束了。

22. door(bang)能不能用return代替

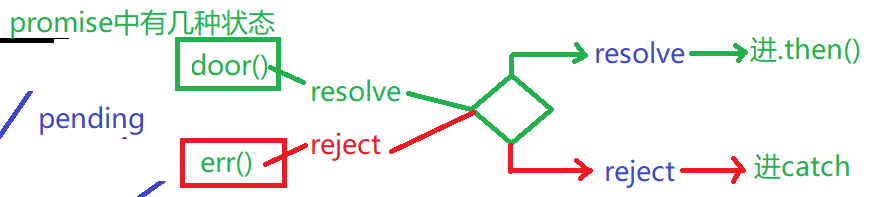
答: 不行！因为door()不仅有返回值的作用，而且，还有修改promise状态的作用。而return只有返回值的作用，无法改变promise对象的状态触发.then或.catch

其实一个promise对象有三个状态:

1. pending: 挂起: 异步任务正在执行，但还没执行完。

2. resolve: 同意: 当异步任务执行完，且执行成功，主动调用door()函数，会将Promise对象的状态，改为resolve。当promise对象的状态被改为resolve后，就会自动执行.then()中的函数

3. reject: 拒绝: 当异步任务执行完，且执行失败了，主动调用err()函数，会将Promise对象的状态，改为reject。当promise对象的状态一旦被改为reject后，就会自动调用catch()中的函数。



23. Promise.all如何处理错误？

如果Promise.all中一个异步函数出错，结果是怎样？

答: Promise.all()也是用.catch()处理函数某一个异步任务错误。如果其中一个异步任务出错，则其它异步任务继续执行完。但是.then()不再执行，而是转到.catch()中执行错误处理代码。因为all中其它异步任务不知道哪个同伴出错了，所以依然继续执行。而.then因为没有收到all中所有异步函数都转为resolve状态的通知，认为不是所有异步任务都成功，就不能继续执行.then了。

24. //function fun(sname){

let fun=(sname)=>{

this.sname=sname

}

var lilei=new fun("Li Lei"); //报错

console.log(lilei.sname);

答: 箭头函数不能new

25. var objTest={  
 '2':'3',

'3':'4',

length:'2',

splice:Array.prototype.splice,

push:Array.prototype.push  
 }

objTest.push(1);

objTest.push(2);

console.log(objTest)

结果为啥是: [empty x 2, 1, 2, length:4, ...]

答: push本意是在数组末尾添加一个新元素。而数组末尾，就是length属性值所指的位置。（length属性值通常总是等于最大下标+1，且length指向的位置，就是数组中最后一个元素之后的位置）

上例中, length:2，所以，push(1)时会将第一个值放在数组中下标为2的位置，并且将length+1变为3，然后再push(2)，会将2放在数组中3的位置，并将length再+1，变为4。这就是push()方法内部固有的原理。

所以，1和2代替了原[2] [3]两个位置的值，且length变为4。

至于为什么输出数组，应该是浏览器控制台误判。因为从输出结果来看，这个objTest对象的\_\_proto\_\_依然是Object，所以，这还是一个对象。而不是一个数组。所以，数据类型没有变化。

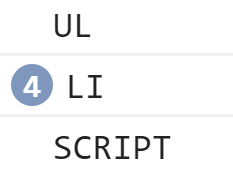
# DOM

## day01

1. Promise.all怎么能即使某个任务出错，也能接受其他执行正确的任务的返回值呢？

答：不能。因为all中的任务和then是前置条件和后置步骤的关系。前置条件，只要有一个不满足，表示整件事儿，办不了。比如，去办手续，人家说要提供身份证，户口本，照片。那么，这三样，缺一样，这个手续的手续步骤都是办不成的。所以，前置条件只是部分成立，是没有意义的。

从程序角度说，.then和.catch()是二选一的关系。只要有人开了err()，状态变为reject，整个任务就会进入.catch()，.then就不再执行。不可能又执行.then()，又执行.catch()。

2. 为什么第一次输出

刷新后变成: 

答: 控制台有个习惯，相邻的多个相同的输出结果，只显示一个，在左侧圆圈中标出重复的次数。

但是，本例中，我们不应该关心控制台的显示格式。而应该只关心总次数对不对。比如: 根据深度优先遍历，ul到script之间，总共应该输出4个li。所以，以上两种方式都是正确的！

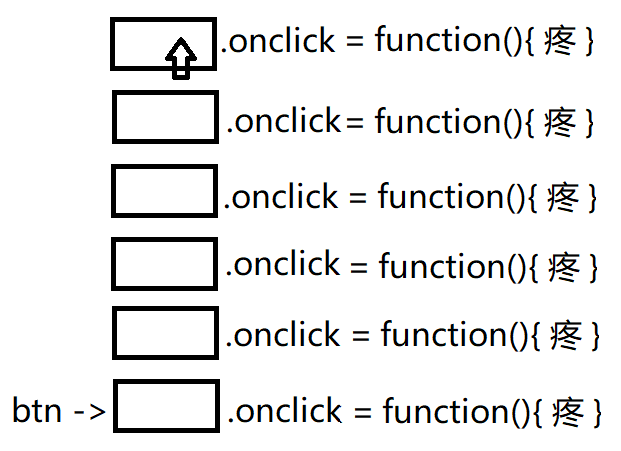
3. for(var btn of btns){  
 btn.onclick=function(){

alert("疼")

}  
 }

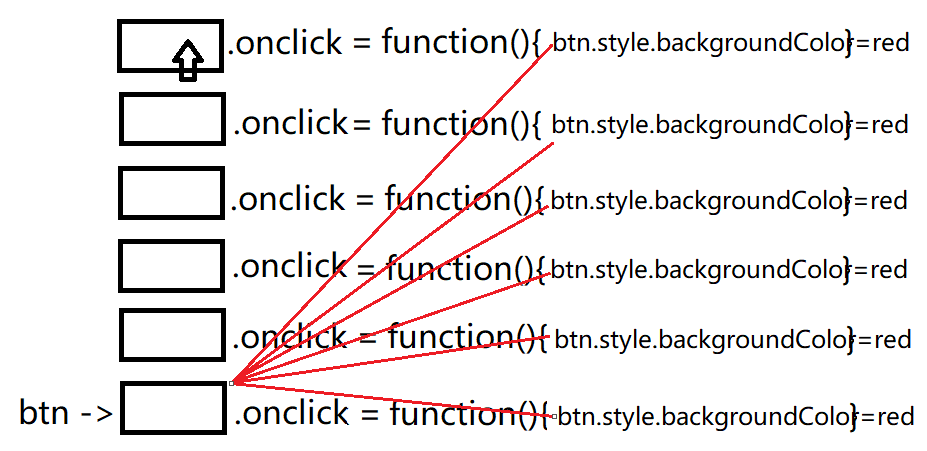
是不是也只能最后一个按钮喊疼？

答: 不是。每个按钮都能喊疼！

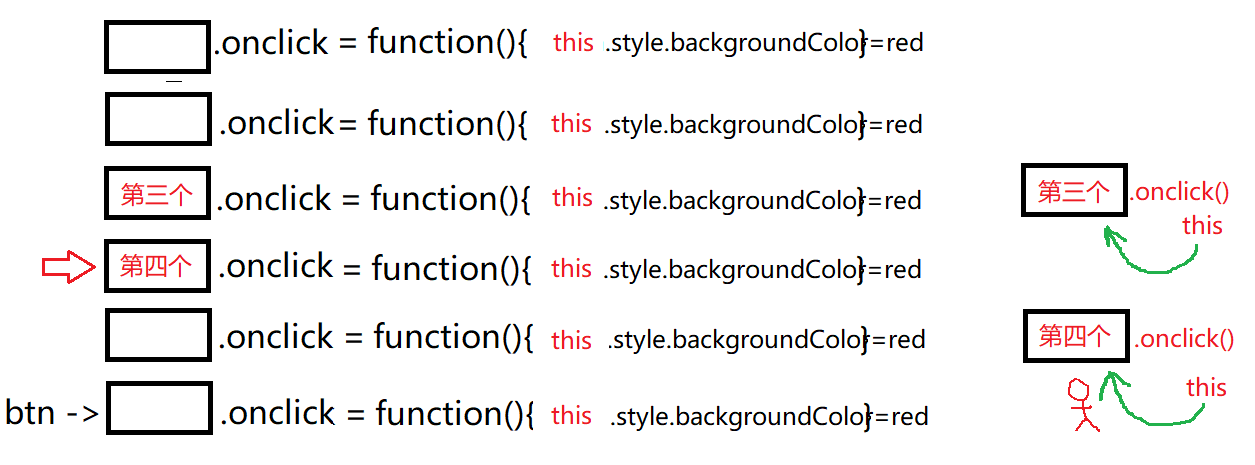


为什么onclick中改成 btn.style.backgroundColor=red就只有最后一个按钮才能变红呢？

因为处理函数中没有局部变量，所有函数中的btn都指向全局变量btn，也就是最后一个按钮：



为什么用this又对了呢？



4. var n=parseInt(span.innerHTML)为什么不用Number()转换

答: parseInt()和parseFloat()是专门将字符串转为数字类型的函数。且parseInt()和parseFloat()可自动去掉数字后非数字字符，比如px单位。而Number()无此功能，比较弱。所以，今后，能用parseInt()或parseFloat()都用parseInt()和parseFloat()。

5. if(btn.innerHTML=="+"){

n++

}else if(n>1){

n--

}  
 为什么else if中不写btn.innerHTML=="-"

答: 因为if else结构中，如果if条件不满足，则已经隐含了btn.innerHTML不是+，在这个例子中，是有两种情况: btn是+或btn是-，那么只要不是+，就意味着一定是-。所以，else if中不用重复写btn.innerHTML=="-"。

6. 到目前为止所有this的用法:

答: this的汇总:

1. 普通函数或匿名函数自调中的this->window

严格模式下: this->undefined

2. obj.fun() fun中的this->obj

3. new Fun() Fun中的this->正在创建的新对象

4. 原型对象中的函数里的this->将来调用当前函数的.前的当前类型的子对象。

5. btn.onclick=function(){... } this->btn

6. 回调函数:

arr.forEach(function(){ ... })

arr.map(function(){ ... })

setInterval(function(){ ... })

setTimeout(function(){ ... })

$.ajax({

...

success:function(){ ... }

}).then(function(){

...

})

this->window 所有回调函数，真正被调用时，前边是没有任何"对象."前缀

所以，通常如果希望回调函数中的this不指window，而是跟外部的this保持一致，都要改为箭头函数

7. 索引数组，关联数组，类数组对象的差别

答: 索引数组: 是指自己没有指定下标名称，而自动生成0 1 2 3数字下标的数组，称为索引数组。

比如: var arr=["a", "b", "c"];

结果: arr中的结构是: [0:"a", 1:"b", 2:"c"]

关联数组: 是指自己定义了有意义的下标名称，不用数字下标的数组

比如: var ym=[]

ym["数学"]=79

ym["语文"]=65

ym["英语"]=89

结果: ym中的结构是:

[ "数学":79, "语文":65, "英语":89 ]

(控制台显示时，虽然不显示双引号，但是自定义下标都是字符串，毋庸置疑)

类数组对象: 是指原本不是数组，而是一个对象{}，但是对象中的属性名，有人故意定义成0 1 2 3数字下标，这种长得像数组的对象，称为类数组对象

比如: var obj={}

obj["0"]="a";

obj["1"]="b";

obj["2"]="c";

(这里[]里双引号可省略，因为即使你省略了，人家也会在添加时，自动加"")

结果: obj中的结构是: { "0": "a", "1":"b", "2":"c" }

(控制台显示时，属性名的""是不显示的，但是任何对象的属性名都是字符串，这毋庸置疑)

8. var table=document.getElementsByTagName("table")

var btns=table.getElementsByTagName("button")

报错: table.getElementsByTagName不是一个函数

答: 因为getElementsByTagName永远返回类数组对象，即使只找到一个，也会放在类数组对象中返回，而不可能直接返回一个元素对象。所以这里table拿到的不是元素，而是一个类数组对象。所以，不能用类数组对象继续调用getElementsByTagName()，因为这个函数只能在元素对象上调用。

解决: var table=document.getElementsByTagName("table")[0]

这样table获得的才是找到的唯一的那个元素对象

才可以继续调用查找方法。

var btns=table.getElementsByTagName("button")

9. 类数组对象到底是保存在[]中，还是{}中？

答: 类数组对象是长得像数组的对象，本质是对象，所以，应该是{}表示。但是，因为类数组对象的用法和数组几乎相同，所以chrome控制台或者笔记中，有时为了直观会将类数组对象显示成[]。但是这些都不应该成为使用类数组对象的障碍。因为无论是对象还是数组，底层都是一样的，只是表现稍微不同而已。所以，无论{}还是[]，都不重要。重要的是，拿到类数组对象或者数组后，都可访问下标，都可有length属性，都可for循环遍历。这才是本质。不要只看表象。

10. 将事件处理函数改成箭头函数，原来正确的功能，就不正确了？

答: 事件处理函数中用到了this，要求是指当前正在触发事件的元素对象，而不能指到函数外其它对象。所以，只要希望函数内外this不同，就不能用箭头函数简化。

## day02

1. var tds=document.querySelectorAll("tbody td:last-child")

var total=0;

for(var td of tds){

total+=parseFloat(td.innerHTML)

}

既然td已经找到了小计的单元格，直接累加到total上就行，为什么必须加innerHTML？

答: 每个元素都是一个元素对象，而元素对象不等于元素内容。元素内容，仅仅是元素对象中众多属性中的一个。比如元素对象有id属性，title属性，还有onclick属性等。内容属性innerHTML，只是一个元素对象中众多属性中的一个而已。所以，往total变量中累加的不是整个td对象。累加的应该仅仅是td对象的innerHTML属性中保存的元素的内容。所以，必须加innerHTML

2. 哪些函数找不到返回-1？哪些函数找不到返回null？哪些函数找不到返回空集合，什么时候返回undefined？

答: 总结:

凡是返回值为下标的函数，找不到都返回-1

比如: indexOf

凡是返回值为一个对象或数组的，找不到都返回null

比如: querySelector() getElementById() match()

凡是返回值为一个集合(类数组对象)，找不到都返回长度为0的空数组: [].length=0

比如: querySelectorAll()

getElementsByXXX()

没有函数返回undefined

3. 在浏览器中如何测试简单的HTML代码和样式:

答: F12->elements 右键单击要修改内容的元素，选edit as html，就可修改。改完后，点编辑区域以外的空白处，就可应用修改。一定小心，刷新后，就没有了！

4. 父元素不是靠内容撑起高度吗？为什么不能不定死高度呢？

答: transition要求必须有明确的变化前高度，和变化后高度，transition才会计算并过渡变化。如果少任何一个值，transition都无法工作。

5. 定死高度不好，如何动态获得高度？

答: 只能用js来每次都动态计算高度了。

//开始时，让第一个二级菜单显示:

//找到第一个二级菜单

var ul1=document.querySelector(

"ul.tree>li:nth-child(1)>span+ul"

);

//二级菜单的高，等于二级菜单ul下li的个数\*每个li的高。

ul1.style.height=ul1.children.length\*22+"px";

//每次展开菜单时:

//先获得当前span旁边的ul

var ul=this.nextElementSibling

//修改当前span旁边的ul的高为ul下li的个数\*每个li的高22

ul.style.height=ul.children.length\*22+"px";

//每次关闭菜单时

//清除当前span旁边的ul的高，使其同步隐藏

this.nextElementSibling.style.height=0;

但是，js动态计算的方式，到vue框架中思路完全不同，所以还要重新写。所以，决定使用动态高度时，一定考虑有没有必要。如果菜单内容变化极其不频繁，就没有必要动态改变，承担风险。

6. //如果当前chb的checked为false，说明这个chb未选中，则全选一定不选中

if(this.checked==false){

chbAll.checked=false;

}else{//否则，也不应定全选，因为即使只有当前一个chb选中，也无法保证其他chb是选中的

//再判断，在tbody下找不到未选中的chb了，才能让全选选中

//先尝试查找tbody下未选中的chb

var unchecked=

document.querySelector(

"tbody>tr input:not(:checked)"

);

这里不写if，也能实现，为什么必须写if？

答: 查找也不是天上掉馅饼，也需要耗费时间，尤其是选择器查找藏的很深的元素。所以，能不查找，就不查找。能让计算机少查找一次，就少查找一次。既然已经知道，如果当前chb未选中，则上方的全选肯定不选中，就没有必要耗费时间去遍历别的chb。提高效率。

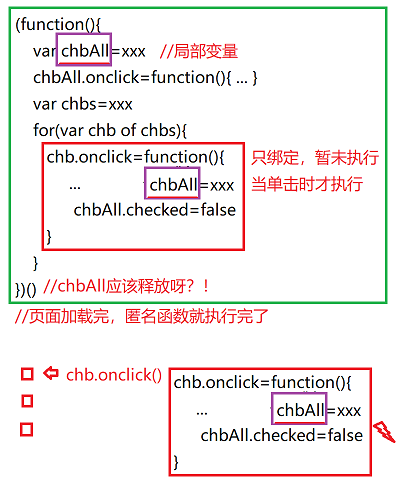
7. 在selectAll中有的查找执行了两次，能不能共用一次?

答: 如果这个元素或这批元素，在操作过程中始终保持数量和属性不变！则可以共用一次查找。

但是，即使共用，也不能用全局变量保存

可以将整段代码包裹在匿名函数自调中。

被匿名函数自调包裹的代码和不用匿名函数自调包裹的代码执行效果完全一样的！且不会产生全局变量。



如果查找的结果在操作过程中数量或对象可能发生增减或变化，则及时反复查找也是必要！因为保证每次找到的结果都是最新的。

## day03

1. 标签是不是最小的节点?

答: 一个<标签名>是一个标签，一个标签不会成为节点对象。只有由开始标签+结束标签联合表示的一个元素才是一个元素节点对象。

节点不分大小，只区分上下级。而元素节点不一定就是最底层的节点。还可能包含子元素或子文本。

但是，在元素树中，最底层节点一定都是元素节点。

2. <table class="data">

var table=document.getElementsByClassName("data");

table.getElementsByTagName("button") 报错！getElementsByTagName不是一个函数?

答: document.getElementById()只返回一个元素对象，所以可以直接使用这个元素对象，调用后续函数

但是其余三个getElementsByXxx()都返回的是集合。集合中装的才是找到元素对象。即使只找到一个元素，也是装在集合中返回的！如果要取出集合中唯一找到的这一个元素使用，应该:var 一个元素对象= getElementsByXxx() [0]

3. 为什么把<script>放在<head>中，或<body>开头就错了?

答: 网页是顺序执行。如果<script>放在网页内容的前边，则<script>中查找元素时，什么也找不到！

4. for(var btn of btns){  
 btn.onclick=function(){ btn.style.backgroundColor="red" }  
 }为什么无论点哪个按钮，总是最后一个按钮变红?

答: btn是循环变量，在循环过程中，绑定事件处理函数时，临时指一下当前元素。然后，btn会随着循环，向之后的元素移动。直到循环结束，btn只能指向最后一个按钮。

而事件处理函数虽然是在循环过程中绑定，但是并不是立刻执行！事件处理函数是在今后用户点击按钮时，才自动执行。而此时，循环早就结束了！btn只能是最后一个按钮。

所以在处理函数中不能用循环变量来当做当前元素使用的。应该用this->当前元素。因为时间发生时，浏览器自动执行事件处理函数的语句是: 当前元素.onclick(); this->当前元素。

5. for(var btn of btns){

btn.onclick=function(){

var btn=this; //可不可以去掉，因为外边已经有 变量btn了，为什么不能直接用？是否冲突？

btn.style.backgroundColor="red";

}

}

答: var btn=this;是必须的！因为外边的btn是全局的！极容易被污染！而之所以在函数内var btn，就是想在函数内定义一个专属的局部变量btn，不受外部btn的影响。名字相同，纯属巧合！

6. if(btn.innerHTML=="+"){//如果点的是+

n++;//就+1

}else if(n>1){//否则 如果点的是-，且内容>1

//才能-1

} else if中为什么不写btn.innerHTML=="-" ？

答: 一共只有两种按钮，+ 或 -。如果程序能执行到else if，说明前一个条件已经不满足了，就暗含着btn.innerHTML一定不是"+"，那就只能是"-"

7. var n=span.innerHTML，修改n，为什么页面上的span不变？

答: 按值传递。页面上span.innerHTML是一个原始类型的值——"1"，原始类型的值赋值给另一个变量时，是复制副本给对方。修改副本，不影响原值！所以，要想改变页面内容，必须再赋值回去！

8. 页面上span的内容显示没有"1"，只是一个数字1，为什么还要强制转换？

答: 凡是从页面上获得的，哪怕看着是数字的！其实一律都是字符串！因为页面上只能显示字符串！在算术计算前，都要强制转为数字，才能用！

9. for(var i=0;i<btns.length;i++){

btns[i].onclick=function(){

btns[i].backgroundColor=”red”;

}

}为什么也不对？

答：i是循环变量，i只有在遍历过程中，才临时指向某一个具体按钮在集合中的下标。而在遍历过程中，仅仅是对按钮的onclick属性赋值一个事件处理函数，并没有调用执行这个函数。因为事件处理函数，在任何情况下只有单击才执行。只要没单击，就不执行，也不会读取函数中的内容。函数中的i，只能是i，不会变成具体数组。

而当循环结束后，整段程序结束后，用户才单击按钮。单击按钮时，才开始读取处理函数的内容，而此时读到btns[i], 根据最基本的循环的原理，i已经变成刚好等于btns.length了。而btns中是没有btns.length这个下标的。Btns中的下标，最多才到btns.length-1。所以btns[i]返回undefined，所以undefined后边只要加.就会报错！

10. 事件处理函数为什么不能简化为箭头函数？

答：事件处理函数中的this，要求必须和外边的this不一样！所以，不能简化为箭头函数。如果非要简化箭头函数，也有一个比较别扭的方法:

btn.onclick=e=>{

var btn=e.target;

。。。 。。。

}

其中, e.target多数情况下等效于this。但是有不同的时候。所以，也只是临时的做法。

11. <span>1</span>这个1，到底是保存在哪里了？内存？网页上？

答：先写在程序中，<span>1</span>

再被浏览器解析到DOM树上，保存在对象属性中：span:{ nodeName: “SPAN”, innerHTML: “1” , parentNode: td, … }

最后，被浏览器画成界面，显示在网页中: 1

12. 为什么var n=parseInt(span.innerHTML), n++, 就能计算，而span++，就不行？

答: 因为span是一个元素对象，元素的内容是保存在对象的属性中的值。

所以，可以对元素对象中的某个属性+1，但是不能对元素对象整体+1。因为元素对象中包含很多属性值，到底要对那个属性+1呢？无法确定。

就好像，可以lilei.sage++，但是不能对Lilei++一样，没意义。你到底是想给lilei的身高+1，还是给lilei的年龄+1呀？

13. Uncaught DOMException: Failed to execute 'querySelectorAll' on 'Element': 'tbody td:last-chlid' is not a valid selector. ?

答: 'tbody td:last-chlid' 不是一个可用的选择器

凡是看到这个错，都是选择器写错了。

14. Uncaught DOMException: Failed to set the 'display' property on 'CSSStyleDeclaration': These styles are computed, and therefore the 'display' property is read-only.

答: 计算后的所有属性都是只读的。因为来源不确定！怕牵一发而动全身！所以，禁止修改！

总结: 今后，只要修改样式: 元素.style.css属性="值"

今后，只有获取样式时:

var style=getComputedStyle(元素);

style.css属性 ——只读

15. 有id的元素，不用查找，就可直接用id作为变量名，使用该元素，这样不是更简单吗？

答：浏览器确实可以将所有id属性值变为全局变量，并指向该元素对象：

<button id="btn1">

btn1.onclick=function(){ ... }

这里的btn1名，不用查找，也可以使用！而且，btn1指的就是<button>对象。这是浏览器自动转换的一个小福利。

但是，因为浏览器自动将id转为的是全局变量，非常不靠谱，极容易被篡改，所以，禁止直接使用id名来操作元素。所有元素使用之前，必须去DOM上根据特征或选择器找到，才更稳妥。

16. var style=getComputedStyle("d1")，程序出错了

报错! Uncaught TypeError: Failed to execute 'getComputedStyle' on 'Window': parameter 1 is not of type 'Element'

答: getComputedStyle()有规定，参数必须是一个实实在在的DOM元素对象！不能只是一个id名或class名

17. ，四个style不是说变量可以随意改名吗？为什么都改成style\_d1，却报错:

不能访问undefined的display属性?

答: 变量可以随意改名，所以前两个自己var出来的style可以改名： var style\_d1=getComputeStyle(d1)

if(style\_d1.display==="block")

但是，访问对象的属性，不能随便改名，因为属性时在对象中，提前定义好的！我们必须遵照执行

比如: 后两个d1.style，是访问DOM元素中专门保存内联样式的固定属性"style"。是DOM标准中规定的，必须叫这个名字。

18. 空连接: <a href="#"> 和 <a href="javascript:;"> 有什么差别?

答: href="#" 不是不调转，而是在当前页面内跳转。而且只写一个#，表示回到顶部的意思。如果页面很长，且发生了滚动，用href="#"，会导致页面跳回顶部。

href="javascript:;"，其实href中有两种选择:

1. href="url或#锚点" 执行跳转操作

2. href="javascript: js语句;" 当href中以javascript:开头时，浏览器不再执行跳转操作。而是执行javascript:后规定的js语句。但是，多数情况下，不会这么写，因为不符合内容与行为分离的原则。但是javascript:;经常用来表示什么也不做！“;”本来是js语句的结束，如果分号单用，表示空语句，什么也不做！

总结: 今后，只要将一个a当做普通按钮用时，而不执行跳转时，都将href="javascript:;"

19. href=""能不能表示空连接？

答：不行，href=""，表示没有href属性，a也就不是一个超链接元素了。鼠标放上去，是变不成手指的。因为，""是所有无效属性的默认值。href=""，相当于让href属性失效。

20. 是否只要添加自定义属性，data-前缀都是必须的？

答: 是的。今后，只要添加自定义扩展属性都要加data-前缀

21. :hover好像也能实现这个功能？和这个js实现有什么差别？

答: :hover只能实现鼠标悬停，一旦鼠标移走，状态保持不住。

如果需要单击显示，以及鼠标移走，依然保持状态，只能js实现。

22. 能不能用箭头函数保持内外this一致?

答: 这里不行。因为function vali(){}是独立于另外两个事件处理函数外的函数。和事件处理函数不是内外的包含关系，而是平级之间调用的关系。所以不能使用箭头函数保持this相同。只能通过打针(call)方式注射进去

23. Uncaught TypeError: cts is not iterable ?

答: 1. cts变量中确实获得了集合，但是因为浏览器版本太低，导致部分集合还不支持for of遍历。

2. 升级浏览器之后，如果还报这个错，说明变量cts本该获得一个集合，但是因为之前程序的错误，导致没有拿到集合。所以，应该输出变量cts中获得的到底是什么。

24. DOM中: className="类名"。如果原来元素上已经有一个固定不变的class，指向追加一个新类名，怎么做？如果元素上，已经有很多类名，只想移除其中一个，保留其余类名，怎么做？

答: DOM的className操作，非常的死板，只能用字符串整体替换。如果希望仅操作多个类中的一个，可使用字符串操作:

比如: 为按钮添加一个class btn-success

btn.className+=" btn-success"

移除按钮上的class btn-success

btn.className=btn.className.replace("btn-success", "");

但是，依然要小心谨慎。

## day04

1. <table>

<tr>...</tr>

<tr>...</tr>

</table>

table.children找不到tr？

答: 因为html标准规定，<table>下必须先包含行分组，在行分组内，再包含<tr>。即使写HTML代码时，没有加行分组，直接用<table>包含<tr>，为了符合HTML标准规定，浏览器也会自动将所有<tr></tr>包含在一个<tbody>中。

<table>

<tbody>

<tr>...</tr>

<tr>...</tr>

</tbody>

</table>

所以，今后只要写table，必须写行分组，然后在行分组中，再添加tr。

2. function fun(){

... ...

return 3

和

return

3

}

有什么差别?

答: 本来一个js表达式中间是可以换行的

但是: return后如果有返回值的话，不能随意换行

如果换行，js会在return后自动添加; 即使return下方还有其他表达式，也不再执行。

总结: return后只要有返回值，都不要换行

3. var lilei={}

(function(){

})()

(function(){

})()

问题1: 报错 {}不是一个函数

答: js将{}和之后的匿名函数的()，断句成了一句话，所以报错。

解决: 今后只要匿名函数之前必须加;

var lilei={} ;

(function(){

})()

(function(){

})()

问题2: 依然报错: TypeError: (intermediate value)(...) is not a function

答: js再次断句错误，将前一个匿名函数自调的(function(){})()和下一个匿名函数自动的()断句成一句

解决: 今后匿名函数自调结尾，也要加;

var lilei={} ;

(function(){

})();

(function(){

})();

总结: 今后只要使用匿名函数自调，前后都要加“；”

4. <i></i>能不能触发事件?

答: 所有的元素上都有onxxx属性，只要有onxxx属性的元素，都可以触发事件，绑定事件处理函数。

5. btnShoot.onclick=function shoot1(){ ... }

btnShoot.removeEventListener("click",shoot1)报错: shoot1未定义

答: =function shoot1(){}这是一种特殊的语法。称为函数签名。函数签名中的函数名shoot1，属于一种仅限于当前函数内才可使用的变量（类似于闭包），绝不等同于全局变量——记住。

所以，在function shoot1(){}之外，使用shoot1，都是未定义。

同时，即使改成function shoot1(){ ... }

btnShoot.onclick=shoot1;也是错误的

因为DOM规定，使用onclick绑定的事件处理函数，不能用removeEventListener移除。只能重新赋值为null: .onclick=null。

6. btnShoot.addEventListener("click",shoot2)，即使执行多次，也只能获得一个跟踪导弹

答: 因为浏览器规定，监听对象队列中不允许添加完全相同的两个监听对象。完全相同指: 元素对象相同，事件名相同，且处理函数地址还是同一个函数。

7. close() 出现以下错误:

VM48:1 Scripts may close only the windows that were opened by it.

答: Chrome浏览器个别版本要求，close()函数只能关闭用程序打开的窗口: 比如用open()打开的新窗口，才可用close()关闭。如果是手动打开的浏览器窗口，禁止使用close()关闭。

但是不是所有Chrome都报这个错误，只限于个别版本。

8. btnShoot.addEventListener("click",shoot2)，shoot2后为什么不加()？

答: 事件处理函数，属于回调函数，只有在将来用户点击时，才自动调用。如果用户不点击，就不会调用。而()是立刻调用函数，且只调用这一次的意思，事件处理函数绝不是立刻调用函数，更不是只执行一次。所以不加()。

# JQUERY

## day01

1. TypeError: $btns is not iterable ?

答: $btns不能被for of遍历

可能1: $btns本该拿到数组或类数组对象，但是没拿到

可能2: 使用的jQuery版本不是3.x。只有jquery 3.x的类数组对象，才开始支持for of遍历。jquery 1.x都不支持for of遍历

解决: 改为for循环  
 其实，不用循环。因为jQuery所有函数都自带遍历效果。只要对整个jQuery对象调用一次简化版函数，就等于自动给jQuery中的每个DOM元素都调用了一次相同的函数。

2. 既然$btns.click()只写一次，就可以给所有按钮绑定单击事件，那么今后是否就不需要事件委托了？

答: 依然需要事件委托。因为虽然对整个$btns只调用了一次click()函数，但是，实际效果，却是自动在内部给每个button都.onclick了一次，也就说，依然创建了多个监听对象。监听对象的个数，并没有少。

所以，今后只要多个子对象，都需要响应相同的事件时，还是要用事件委托

3. n="0" n+1 返回 "01" n++ 返回 1?

答: + 碰上字符串，会变成字符串拼接

++ 会强转为数字，再+1

4. 到底谁自带遍历？是$()带遍历？

答: $()只是负责找到DOM元素保存起来，并不对DOM元素做任何操作！

只有$(...).简化版函数()，才会对jquery对象内的DOM元素执行操作。所以，其实是每个简化版函数内都自动包含一个for循环。

5. DOM中学的事件对象e，在jquery中能不能使用？

答: 完全相同

而且不止jQuery中。今后，框架里用到e，也是原生的DOM中的e。

6. 使用btn.click(function(){ ... }) 事件处理函数，能不能加形参e，获得事件对象？

答: 可以！ btn.click(function(e){ ... })

因为jQuery其实就是DOM，和DOM原理完全一样

新瓶装旧酒

7. 使用btn.click(function(){ ... }) 能不能将事件处理函数改为箭头函数

答: 不能！因为事件处理函数中的this，要求指向当前触发事件的元素本身，就要与外部的this不一致。所以，不能改为箭头函数

8. 事件处理函数能不能改成箭头函数

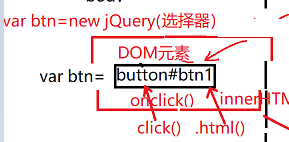
答: 如果事件处理函数内涉及this，就不能修改成箭头函数。因为事件处理函数中的this要求和外部应该不一样！

但是，如果this可用e.target代替，就与this无关了！就可以改成箭头函数！

9. $("#btn1").click(function(){ ... this ...})，.前已经是$("#btn1")了，this应该已经指向$("#btn1")了，为什么还要再$(this)呢？

答: $(...).click(function(){ ... })中的.click()只是原生方法.onclick的一个简化版外号，便有使用。但是，当我们调用.click()时，jquery对象还会自动将简化版外号，翻译为原生的onclick。所以，$("#btn1").click(function(){ ... this ...})在底层实际执行的还是: btn1.onclick=function(){ ... this ...}。真正作用在btn1上的还是onclick事件，所以this还是指DOM的btn1。

一定要透过现象看本质，透过简写的外号和别名，看到原理，才能应付自如。不能只看表面。



## day02

1. $(...).attr({ "src":"img/1.jpg"}), 也可以写{src:"img/1.jpg"}

答: 其实{}中:前的名字都是"属性名"。在对象语法中，属性名本质都是字符串，但是""可省略！

但是2种情况不能省略:

1. json字符串中，属性名必须加""

2. 属性名中包含特殊符号的:

.css({ "background-color": "red" })

也可以不加引号，但是要变为驼峰命名

backgroundColor:"red"

2. $("img.my-small").click(function(){

this -> click的.前的主语

↓

$("img.my-big").attr("src",$(this).attr("data-target"))

})

答: 指click前的$("img.my-small")

.attr中的$(this)是作为实参值传入到attr中，还没进去呢！是从.attr()外部传入的！所有this跟外部的this。

3. css transition 是不是放在开始状态和结束状态的class中都行？

答: transition，必须放在常驻元素上的class上，保持对元素持续的监控。而不能一会有，一会儿没有

比如: 如果.fade{} 一直在元素保留，而是反复切换.in{}class，则transition必须定义在.fade{}中.

@keyframes 动画名{}

animation 动画名 ——应该放在那个不断变化的class上

比如: 如果.fade{} 一直在元素保留，而是反复切换.in{}class，则animation必须定义在.in{}中.

如果animation放在.fade{}中，页面一加载就会自动播放。

4. 对话框居中时，为什么margin-left和margin-top要再减掉padding？

答: 因为盒子模型中，width和height不包含padding，而用户却希望整体居中。

解决: 可将box-sizing:border-box，结果width和height中包含padding了。

5. $("#add-block").click(function(){

添加新元素div.block

});

var $xs=$("div.block>span.close");

$xs.click(function(){

... ...

})

为什么代码没写错，但是，单击x，没反应？

答: 执行顺序问题: 添加div.block的代码，写在了单击事件内部。只要不单击，就不执行。意为着，页面加载时，一个block也没有！所以，在主程序中，查找div.block，一个页面找不到，一个处理函数也没绑上。

当之后，单击按钮添加div.block时，主程序中的代码不会重复执行。只是反复调用处理函数而已。所以，新增的元素，无法自动获得事件处理函数。

不好的解决: 在每次单击时，每次创建div.block时，都反复为新元素绑定单击事件。——缺: 增加事件监听的个数

好的解决: 利用事件委托，将单击事件绑定在父元素上。——优: 1. 事件监听只有一个！2. 新添加的元素可自动获得事件处理函数.

6. $("div.container").prepend($block);

但是换成.prependTo("div.container")

答: 其实.prependTo()中可以是三种情况都行:

1. $("div.container") 是一个jquery对象

2. 也可以是一个DOM对象

3. 最简单的可以只是一个选择器！

扩展: 很多参数，都可以即写选择器，又可以写jquery对象

7. <script> var i=0</script>  
 ...

<script>console.log(i)</script>  
 第二个<script>里为什么可以直接使用第一个<script>中的i

答: 一个网页只有一个全局作用域对象window。一个网页内的所有<script>中的代码其实共享一个window对象。所以，不同<script>之间的变量，可以互相访问。但是，必须注意先后加载的顺序。比如:

<script>console.log(i)</script>

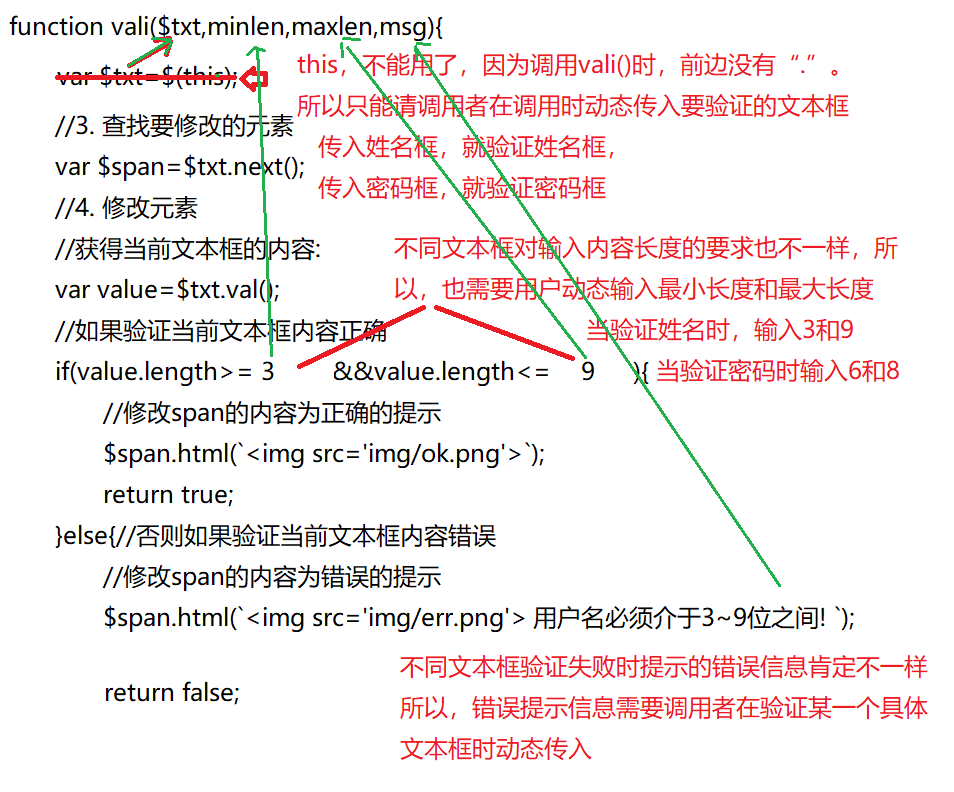
... ...

<script> var i=0</script>

第一个<script>中就报错，因为执行第一句话时，还没有var i=0呢。即使声明提前，也不能把语句从一个<script>中提前到另一个<script>中。所以，第一句话报错。

8. function vali($txt, minlen, maxlen, msg){} 为什么定义四个形参？不是三个形参？不是五个形参？

答: 一个函数需要几个形参，取决于函数体内，有哪些值不能写死，需要动态获得。比如:



9. .toggleClass()可不可以做鼠标划过表格行，高亮显示？

答: 可以，但是最好别用。因为鼠标滑过表格行只是一种临时的样式变化，用CSS的:hover足够了，兼容性也好。

今后，只有那些鼠标离开后，或者点击后，需要将样式保持住的样式变化，才考虑用js+class做。

10. unsupported pseudo: check, 或 unsupported pseudo: unme或 unsupported pseudo: psasword错误

答: $()中的选择器写错了。pseudo是伪类的意思。重点查看:后的单词是否写错

11. css中:

table>tbody>tr:hover{

background-color:lightYellow

}

jq中:

$('table>tbody>tr:odd')

.css('background-color',"lightBlue")

结果: 鼠标悬停在偶数行(:odd行)上时，背景不高亮显示？

答: 程序中.css()函数代替的就是.style.css属性的操作，相当于修改内联样式。所以，程序中.css()修改的样式，优先级最高。

解决: 其实，这种设置背景和悬停变化的操作，就都应该用css做，而不应该用jquery程序做。咱们是处于让大家练习选择器的目的采用jQuery做的。今后，能用css写的都优先用css写。

12. var $a=$(e.target);  
 if(a.nodeName=="A"){ ... }

为什么不行？

答: nodeName是DOM的属性，不是jQuery家的属性。而这个案例中的$a=$(e.target)，是被$()包裹后的jQuery家子对象。所以，被$()包裹后的jQuery家孩子$a，就无法使用nodeName属性。

13. 如果有选择器，是不是最好就不用.parent()或.next()这样的函数？

答: 如果首次查找第一个元素时，只能用选择器。

只有在已经获得元素时，想找周围附近的元素时，才用按节点间关系查找！

从效率上来说，一定优先使用按节点间关系查找！因为按节点间关系查找，根本就没有找！而是直接读取元素对象上早就保存好的属性值！

## day03

1. $(function(){ ... }) 能否改为箭头函数?

答: 可以: $(()=>{ ... })

2. 是否还需要匿名函数自调:

答: 如果所有的jquery代码都放在了$(function(){ ... })中，就不再需要匿名函数自调

如果$(function(){ ... })之外，还有变量等其他程序，就需要再在外围包一个匿名函数自调

3. $(function(){ alert("疼!") })

和$(function(){

btn1.onclick=function(){ alert("疼!")}

})

答: 第一个是在DOM内容加载后，就立刻执行alert，弹出警告框，不需要用户点击

第二个是在DOM内容加载后，仅仅给按钮绑定一个单击事件，当用户单击按钮时，才执行alert()

4. css中： .content.in{ display: block }

js中: $(".content").addClass("in")

css中没有in呀？怎么可以addClass("in")

答: .content.in

一个复杂选择器的类名，其实就是最后一个.的名字

之前所有内容，只是限制这个类名使用的前提条件

比如: .accordion>.content.in

类名最后还是in

只不过前边的限定词规定: 必须是class为accordion下的class为content的且class为in的元素才起作用

5. 如何避免组件间样式冲突？

比如: /\*\*\*下拉菜单的样式\*\*\*\*/

.dropdown{ ... }

.dropdown>.dropdown-menu{ ... }

.fade{ ... }

.in{ ... }

/\*\*\*手风琴组件的样式\*\*\*/

.accordion{ ... }

.accordion>.title{ ... }

.accordion>.content{ ... }

.fade{ ... }

.in { ... }

答: 一个组件下的样式类，必须以这个组件统一的class名为开头编写！

/\*\*\*下拉菜单的样式\*\*\*\*/

.dropdown{ ... }

.dropdown>.dropdown-menu{ ... }

.dropdown>.dropdown-menu.fade{ ... }

.dropdown>.dropdown-menu.in{ ... }

/\*\*\*手风琴组件的样式\*\*\*/

.accordion{ ... }

.accordion>.title{ ... }

.accordion>.content{ ... }

.accordion>.content.fade{ ... }

.accordion>.content.in { ... }

6. 万一每个.content高度不一样，怎么办？

答: 在css中提前设置预案

比如: /\*给每个div单独定义不同的目标高度\*/

.content:nth-child(2).in{

/\*display:block;\*/

height:95px;

opacity:1;

}

.content:nth-child(4).in{

/\*display:block;\*/

height:65px;

opacity:1;

}

.content:nth-child(6).in{

/\*display:block;\*/

height:155px;

opacity:1;

}

7. 如何实现持续不断的动画效果，比如一直闪烁

答: 错误: 死循环！——主程序，且会阻塞主程序

正确: 周期性定时器！

8. <script>何时放在顶部，何时放在底部？

答：学了$(function(){ ... })和window.onload之后，放哪儿都无所谓了！

虽然不一定放在<head>中，但是，顺序还是要注意:

任何情况下必须先引入基础库:

<script src="js/jquery-1.11.3.js">

然后，再引入自己对库的加工或自己封装的函数:

<script src="js/myquery.js">

<script src="js/ajax.js">

最后，再自己的js代码:

<script src="js/index.js"

9. 表单中用<input type="submit">时，会自动刷新页面，错误提示的效果一闪而过，怎么解决:

答: 表单元素<input type="submit">和<button></button>都有默认行为，是整个页面刷新并提交表单。这种整个页面刷新提交表单的方式，现在已经不再使用。今后，所有表单都是用咱们手动编写ajax方式提交表单中的数据。

如果不希望自动整个页面刷新方式提交表单，而是用自己写ajax方式提交表单，就不应该用<input type="submit">和<button></button>。只能用<input type="button">。因为type="button"的按钮，不带任何默认行为，不会自动提交表单。我们可在<input type="button">的单击事件中，编写ajax请求，获取表单中表单元素的值，并提交表单。

10. 笔试题:

function multiFun(a,b,c){

return a\*b\*c;

}

var multi=exec(multiFun);

multi(2,3,4);

要求: 实现exec()函数，实现输出24

答: 从调用exec()函数方式看出，exec只有一个参数，就是一个回调函数，所以: function exec(callback){}

exec()返回一个函数，题目要求，返回24，说明exec()返回的函数，执行的内容，和传入的回掉函数执行的内容完全一样。都是三个数相乘，所以exec()只要将传入的回调函数返回即可。

function exec(callback){

return callback

}

## day04

1. jquery有了$(function(){ ... })，是否还需要包裹匿名函数自调？

答: 如果$(function(){ ... })之外，没有其它变量或函数定义，就不需要再包一层匿名函数自调了。

但是，如果外部还有其他全局变量或函数，则依然需要再包一层匿名函数自调。

2. var [p1, p2,p3]=result 报错: result is not iterable

答: 浏览器版本太低，不支持解构。重新安装高版本Chrome浏览器。

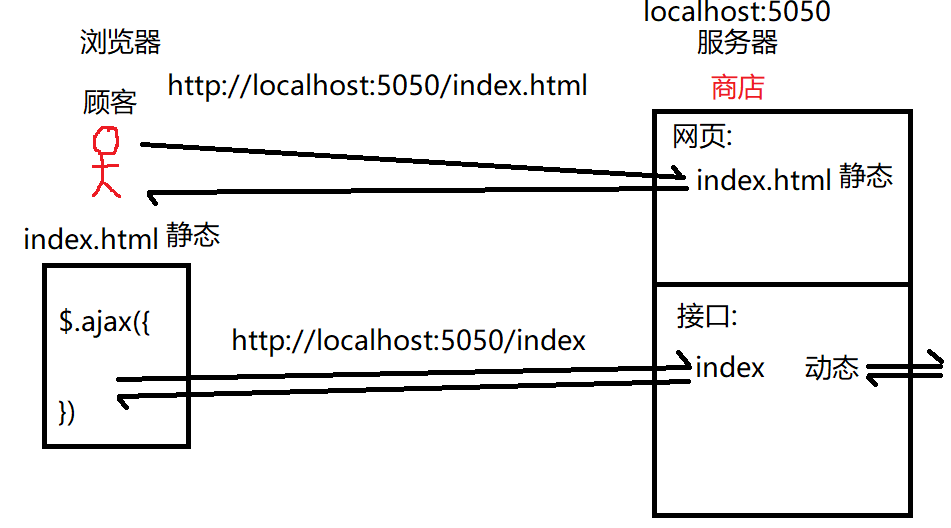
也可能是忘开数据库。

3. 为什么index.js中$.ajax({url: "http://localhost:3000/index"})的url属性值就不加.html结尾，而header.js中$({url:"header.html"})的url属性必须带url结尾呢？

答: 因为一切地址都是服务端规定好的

服务端规定，请求静态的HTML文件必须xxx.html 才能找到想要的html文件

服务端规定，接口地址必须是/index，客户端请求时就不能加.html，就只能用/index



服务端不可能规定程序接口地址以.html结尾，因为会和静态的.html文件地址冲突。

4. 只允许ul下的li触发事件: $ul.on("click","li",function(){ ... })

其中的第二个参数"li"，为什么改为"ul>li"，就不对呢？

答: .on()在事件委托情况下，第二个选择器参数，是相对于父元素的相对条件:

比如: 写$ul.on("click","li",...

判断条件: ul li

写$ul.on("click","ul>li",...

判断条件: ul ul>li