提前准备:

1. 今天的历届师兄师姐问过的经典的高频的问题都已经整理在小程序->首页->JSCORE->day03里

2. 今天的课程中所需的单词列表都已经整理在小程序->单词->JSCORE->day03里

3. 今天的重点商品、扩展高频鄙视题视频已经整理在小程序->在线->JSCORE->day03里

如果不知道问什么: 谁问的越多！谁培训费花的越值！

1. 笔记中或画图中那句话或者那个位置不明白，立刻问！

2. 案例中哪行代码或注释不明白，立刻问！

3. 课下自己做项目，功能不会做，立刻问！

4. 刷笔试题时遇到不会的笔试题，立刻问！——公众号: 前端大全 定期总结大厂笔试题

5. 看小程序问题清单时，遇到看不懂的问题，立刻问！

所有带注释的案例: 每个案例至少写三遍！

1. 注释+代码抄一遍！

2. 保留注释，删掉代码，自己试着把代码填回来！

3. 删除注释和代码，用自己的话把注释和代码一起写回来！

标准: 直到自己可以用人话把注释写出来！(代码记不住都没关系！可以翻笔记！)

/未来6天，将学习内功心法和基本功，不要考虑功能怎么做！/

正课:

1. 函数

2. 重载

3. 匿名函数

4. 作用域和作用域链

5. \*\*\*\*\*闭包

一. 函数:

1. 什么是函数: 程序中专门保存一段可反复使用的代码段的程序结构，再起一个名字

2. 为什么使用函数: 重用!

3. 何时: 只要一段代码片段，可能被反复使用时，都要保存在一个函数中，然后反复调用函数。

4. 如何定义函数: 3种:

(1). 用声明方式定义函数:

a. function 函数名(形参列表){

要反复使用的代码片段

return 返回值;

}

b. 形参:

1). 什么是: 专门接受函数执行时必须的参数值的变量

2). 为什么:

i. 因为有些函数必须提供某些数据才能正常执行！

ii. 为了让函数变的更灵活！可根据不同的实参值，执行不同的操作。

3). 何时:

i. 如果一个函数必须提供某些数据才能正常执行时，必须提供形参！必须几个值，就要定义几个形参变量

ii. 如果一个函数根据不同的实参值，执行不同的操作时，也可以定义形参。

4). 如何:

function 函数名(形参变量1, 形参变量2, ...){

... 使用形参变量 ...

}

强调: 形参变量即使没有var！也能自动创建该变量

5). 如果一个函数不需要外界传入值，就没必要定义形参

c. 返回值:

1). 什么是: 一个函数的执行结果！

2). 为什么: 因为函数内的执行结果，默认函数外不可使用！

3). 何时: 如果将来调用这个函数的人，希望获得函数的执行结果时，才用返回值!

4). 如果调用函数的人不需要知道函数的执行结果时，就不用返回值！

5). 如何:

function 函数名(){

...要重用的代码段...

return 返回值 //小提示: return和返回值之间一定不要加换行

}

6). 强调:

i. return和返回值之间一定不要换行，因为return后如果没东西，js会自动给return后加; 表示什么都不返回

ii. return不能用于三目内部:

~~条件?return 值1: return 值2~~

return 条件?值1:值2

d. 示例: 定义一个可以做饭的函数，接受不同的食材，做不同的饭

|  |
| --- |
| //想定义一个函数可以做饭  function cook(dan,cai,mian){    console.log(`炒${dan}`);    console.log(`炒${cai}`);    console.log(`${cai}和${dan}一起炒`);    console.log(`煮${mian}`);    console.log(`将炒好的${cai}${dan}和煮好的${mian}搅拌在一起`)    return `一碗热腾腾香喷喷的${cai}${dan}${mian}`  }  //星期一想吃西红柿鸡蛋面:  var wan=cook("鸡蛋","西红柿","面条");  console.log(`吃 ${wan}`);  //星期二没西红柿了！只有黄瓜！  var wan=cook("鸡蛋","黄瓜","面条");  console.log(`吃 ${wan}`);  //星期三没有鸡蛋了，只有辣条!  var wan=cook("辣条","黄瓜","面条");  console.log(`吃 ${wan}`); |

(2). 用赋值方式定义函数:

a. var函数名=function (形参列表){

要反复使用的代码片段

return 返回值;

}

b. vs 声明方式比较:

1). 将来调用时，用法完全一样！

2). 声明提前:

i. function 函数名(形参列表){ ... } 会整体被声明提前

ii. var 函数名=function(){ ... } 只有=前的函数名会被声明提前，=后的function(){ ... } 会留在原地.

c. 总结: 基本上都会用赋值方式，代替声明方式来定义函数，避免被声明提前，打乱顺序!

d. 揭示了: js中函数的本质:

1). 函数其实也是一个对象，只不过对象中保存的不是一个数字，不是一个字符串等值，而保存的是一段代码片段！

2). 函数名其实仅仅是一个普通的变量而已！

3). 函数名中通过保存函数对象的内存地址值，来引用着函数对象

4). 当调用函数时，js引擎会先找到函数名变量，再通过变量保存的函数地址找到函数对象，然后才执行函数对象内保存的代码片段！

更多声明提前的知识: 下课后做作业:

看小程序->在线->JSCORE->day03: 1. 声明提前和3. 声明提前...

e. 示例: 比较声明方式创建函数和赋值方式创建函数的差别

|  |
| --- |
| //var fun=function(){  function fun(){    console.log(1);  }  fun(); //2  //var fun=function(){  function fun(){    console.log(2);  }  fun(); //2 |

(3). 用new来创建函数:

a. var 函数名=new Function("形参变量1","形参变量",...,"函数体; return 返回值");

b. 很少使用！

c. 更揭示了: js函数的本质: 函数也是一个对象

其实function 只是 new Function()的简写而已！

function等效于new Function()

d. 示例: 两个函数做比较:

|  |
| --- |
| var a=function(){console.log(1)};      //new Function() 地址: 0x1234  var b=function(){console.log(1)};      //new Function() 地址: 0x9091  console.log(a==b); //false  //     0x1234==0x9091 |

5. 如何调用函数:

(1)var 变量=函数名(实参值列表);

(2)结果:

a. js引擎先找到指定名称的函数

b. 将调用函数时()中的实参值列表，逐一赋值给定义函数时()中的形参变量

c. 按函数定义中的代码片段，执行任务

d. 将函数返回的执行结果，保存到=左边的变量中

二. 重载(overload):

1. 什么是: 相同函数名，不同形参列表的多个函数，在调用时，可根据传入实参值的不同，自动匹配对应的函数调用执行.

2. 何时: 今后只要一件事，希望根据传入实参值的不同，动态执行不同的操作时

3. 为什么: 为了减少函数名的个数，减轻调用者的负担！

4. 如何:

(1). 问题: Js语法默认不支持重载效果！

因为js中不允许多个同名函数同时存在

如果同时存在多个同名函数，则后定义的同名函数会覆盖先定义的同名函数

(2). 解决: js中是通过一个特殊arguments对象来变通实现重载效果

a. 什么是arguments:

1). 每个函数中自带的，无需自己创建，就可直接使用

2). 自动接收所有传入函数内实参值，

js中即使没有定义形参变量，将来调用时，依然可以所以传入实参值！因为有arguments自动接着呢！

3). 类数组对象:

i. 什么是: 长得像数组的对象

ii. vs 数组:

相同: 1. 下标i, 2. length, 3. 可用for循环遍历

arguments: { 0: 值1, 1:值2,... , length:元素个数 }

不同: 类数组对象不是数组家孩子！无权使用数组家任何函数！

b. 如何arguments变通实现重载:

1). 只定义一个函数，且不要定义形参！

2). 在函数内部，判断arguments对象内容或元素个数不同，来选择执行不同的操作

5. 示例: 定义一个pay函数，支持三种付款方式:

|  |
| --- |
| //想定义一个函数可以付款:  //但是现实中有三种付款方式:  //1. 手机支付: 不需要提供任何东西，直接用手机扫描即可支付  //2. 现金支付: 需要将现金交给收款员手中  //3. 刷卡支付: 既需要刷卡识别卡号，又需要输入密码  function pay(          ){   //arguments{          }.length   //           0  1  2    console.log(arguments);    if(arguments.length==0){//说明没传入任何实参值      console.log("手机支付...");    }else if(arguments.length==1){//说明只传入一个实参值      console.log(`现金支付...,收您:${arguments[0]}元现金`)    }else{//说明传入了1个以上的实参值      console.log(`刷卡支付...从你卡号:${arguments[0]}中扣款成功！`)    }  }  //将来调用时:  pay();//手机支付  pay(100);//现金支付  pay("6553 1234","123456")//刷卡支付 |

总结:

1. 创建正则表达式对象: 2种:

(1). 如果正则表达式是固定不变的，不需要在程序运行时动态生成:

var reg=/正则/ig;

(2). 如果正则表达式不是固定的，需要在程序运行时根据其他内容动态生成：

var reg=new RegExp("正则","ig")

(3). 冲突: 无论用//还是用new RegExp()，只要正则中的内容和关键词或关键标点符号冲突了！都要用\转义！

2. 查找敏感词: 4种:

(1). 查找一个固定的敏感词出现的位置: var i=str.indexOf("敏感词",fromi)

(2). 用正则查找一个敏感词的位置: var i=str.search(/正则/i);

(3). 获得敏感词的内容: 2种:

a. 只获得一个敏感词的内容和位置: var arr=str.match(/正则/i)

b. 获得所有敏感词的内容(不包含位置): var arr=str.match(/正则/ig);

(4). 既获得每个敏感词的内容，又获得每个敏感词的位置:

do{

var arr=reg.exec(str);

if(arr!=null){

console.log(arr)

}

}while(arr!=null);

(5). 返回值:

a. 如果函数返回下标位置i，则找不到返回-1

b. 如果函数返回数组arr，则找不到返回null

3. 替换敏感词: 2种:

(1). 将所有敏感词都替换为统一的新值:

str=str.replace(/正则/ig, "新值");

(2). 根据每次找到的敏感词不同，动态选择不同的新值替换

str=str.replace(/正则/ig, function(kw){

return 根据本次kw接住的敏感词不同，动态返回不同的新值

} )

(3). 删除敏感词: 其实就是将敏感词替换为""

4. 切割字符串:2种

(1). 切割符是固定不变的:

var arr=str.split("固定的切割符");

(2). 切割符不是固定的

var arr=str.split(/正则/i);

(3). 打散字符串为字符的数组:

var arr=str.split("");

5. 验证字符串格式:

var bool=reg.test(/^正则$/)

作业：

1. (必须)看小程序视频复习声明提前:

看小程序->在线->JSCORE->day03: 1. 声明提前和3. 声明提前...

2. (必须)看小程序问题清单实现add函数，能够计算任意多个数值的和

小程序->首页->搜"重载"

3. (必须)看小程序问题清单学习作用域笔试题:

小程序->首页->搜"foo" 结果: 笔试题 var a=10 foo...