JavaScript基础入门

第3天课堂笔记（本课程共3天）

# 目录

[目录 2](#_Toc24537)

[一、复习 3](#_Toc15128)

[1.1 if 3](#_Toc3939)

[1.2 for 3](#_Toc17497)

[二、几道算法练习 4](#_Toc9249)

[2.1 题目1：报7游戏的安全数 4](#_Toc3131)

[2.2 题目2：水仙花数 4](#_Toc29513)

[2.3 题目3：求1~100的和 5](#_Toc32511)

[2.4 求阶乘 6](#_Toc31856)

[2.5 用户输入一个数，输出因数的个数。 6](#_Toc14548)

[2.6 判断质数 6](#_Toc25137)

[三、函数 7](#_Toc5229)

[3.1 初步认识函数 7](#_Toc21745)

[3.2 函数的参数 9](#_Toc10030)

[3.3 函数的返回值 10](#_Toc26828)

# 一、复习

昨天学习到了程序的两种控制流向语句，原来的程序，只能一条一条的执行，不能有别的分支。

“流程控制语句”：if、for。

## 1.1 if

选择语句，给程序添加了多种执行路线。

|  |
| --- |
| 1. if(){ 2. 语句1 3. }else if(){ 4. 语句2 5. }else if(){ 6. 语句3 7. }else{ 8. 语句4 9. } |

有且仅有一条出路。注意跳楼现象。

所以我们发现，计算机的两个基本能力：1）计算能力 2）流程控制能力

## 1.2 for

循环语句，顾名思义，就是将结构类似的语句重复执行。

|  |
| --- |
| 1. for(var i = 0 ; i <= 100 ; i++){ 2. console.log(i); 3. } |

for语句能够简化程序的书写，不用大量的ctrl+C、ctrl+V了；

**for语句充分体现计算机的“奴隶性”。**

比如题目：寻找1~1000之内，所有能被5整除、或者能被6整除的数字

|  |
| --- |
| 1. **for(var i = 1 ; i <= 1000 ; i++){** 2. if(i % 5 == 0 || i % 6 == 0){ 3. console.log(i); 4. } 5. **}** |

上面这个算法，我们有一个术语**“穷举法”**。

穷： 完整。欲穷千里目，更上一层楼。穷尽。

举： 列举，推举。

穷举法：就是一个一个试。我们现在要寻找1~1000之内，所有能被5整除、或者能被6整除的数字。

我们的思路，就是将1、2、3、4、……998、999、1000依次去试验。看看这个数字，能不能被5或者6整除。

这时候你说，老师，那计算机太辛苦了。你看，13这个数字，为什么还要试呢？打眼一看就不能被5、6整除。

计算机就是一个奴隶。它没有思维，它就是一个不吃饭，有着极强计算力的东西。

# 二、几道算法练习

## 2.1 题目1：报7游戏的安全数

大家从小到大，都玩儿过的一个庸俗的游戏：

游戏玩儿法就是，大家轮流报数，如果报到能被7整除的数字，或者尾数是7的数字，都算踩地雷了。就应该罚唱歌。

请在控制台输出1~60之间的所有“安全数”。

比如：

1、2、3、4、5、6、8、9、10、11、12、13、15、16、18、19、20、22、23、24、25、26、29、30……

答案见案例

## 2.2 题目2：水仙花数

水仙花数是一种特殊的三位数，它的特点就是，每个数位的立方和，等于它本身。

比如153就是水仙花数。因为：

13+53+33 = 153

100~999之内，只有4个水仙花数，请找出来。

特别经典的算法，是每个学习编程的人，都要会做。

答案见案例，我们只列出数值答案： 153、370、371、407

## 2.3 题目3：求1~100的和

求和的题目，涉及到了新的一种算法思想，叫做“累加器”。

1+2+3+4+5+6……

正确的：

|  |
| --- |
| 1. var sum = 0; //累加器 2. //遍历1~100，将所有的数字扔到累加器里面 3. for(var i = 1 ; i <= 100 ; i++){ 4. sum = sum + i; 5. } 6. console.log(sum); |

初学者常见的错误：

错误1：不声明sum，不行的，因为所有的变量都要声明：

|  |
| --- |
| 1. //遍历1~100，将所有的数字扔到累加器里面 2. for(var i = 1 ; i <= 100 ; i++){ 3. sum = sum + i; 4. } 5. console.log(sum); |

错误2： sum不能在for里直接声明：

|  |
| --- |
| 1. //遍历1~100，将所有的数字扔到累加器里面 2. for(var i = 1 ; i <= 100 ; i++){ 3. **var** sum = sum + i; 4. } 5. console.log(sum); |

错误3： 每次for循环都要var一次，是不正确的：

|  |
| --- |
| 1. //遍历1~100，将所有的数字扔到累加器里面 2. for(var i = 1 ; i <= 100 ; i++){ 3. **var sum = 0; //累加器** 4. sum = sum + i; 5. } 6. console.log(sum); |

## 2.4 求阶乘

所谓的阶乘，比如6的阶乘，就是1\*2\*3\*4\*5\*6 = 720。

现在，计算13的阶乘。

|  |
| --- |
| 1. var result = 1; //累乘器 2. for(var i = 1 ; i <= 13 ; i++){ 3. result = result \* i; 4. } 5. console.log(result); |

## 2.5 用户输入一个数，输出因数的个数。

昨天晚上的作业3，就是用户输入一个数，输出所偶遇能够整除它的数字。

比如，

用户输入48，此时输出1、2、3、4、6、8、12、16、24、48 。 共**10**个数字。

用户输入21，此时输出1、3、7、21.共4个数字。

今天的这道题目，和昨天的这个题目非常像，不过不输出完成序列，只输出个数。

也就是说，用户输入48，弹出**10**。

用户输入21，弹出**4**

用户输入11，弹出**2**

提示：此题用到累加器。

## 2.6 判断质数

质数：就是只能被1和自己整除。

翻译过来：它的因数个数是2。

比如：2、3、5、7、11、13、17、19、23、29、31、37……

用户输入一个数字，弹出这个数字是否是质数。

提示，用2.5的思路。

答案见案例。

# 三、函数

## 3.1 初步认识函数

|  |
| --- |
| 1. <script type="text/javascript"> 2. console.log("你好"); 3. **sayHello();** //调用函数 4. //定义函数： 5. function sayHello(){ 6. console.log("欢迎"); 7. console.log("welcome"); 8. } 9. </script> 10. </body> |



**函数，是一种封装。就是将一些语句，封装到函数里面。通过调用的形式，执行这些语句。**

函数的使用，是两个步骤，**第一步，函数的定义**：

语法：

|  |
| --- |
| 1. function 函数名字(){ 3. } |

function就是英语“函数”、“功能”的意思。顾名思义，将一些功能封装到函数里面。

function是一个关键字，和var、typeof一样，都是关键字，后面要加空格。

函数名字的命名规定，和变量的命名规定一样。只能是字母、数字、下划线、美元符号，不能以数字开头。

后面有一对儿空的小括号，里面是放参数用的，下午介绍。大括号里面，是这个函数的语句。

常见错误：

不能小括号包裹大括号：

|  |
| --- |
| 1. function sayHello({ 3. }) |

不能忘了小括号对儿：

|  |
| --- |
| 1. function sayHello{ 3. } |

**第二步，函数的调用。**

**函数如果不调用，等于白写。**

调用一个函数，太简单了，就是这个函数的名字后面加小括号对儿。

语法：

|  |
| --- |
| 1. 函数名字(); |

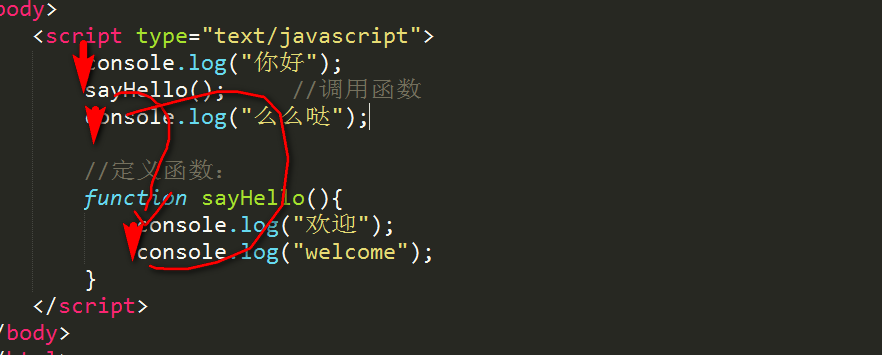
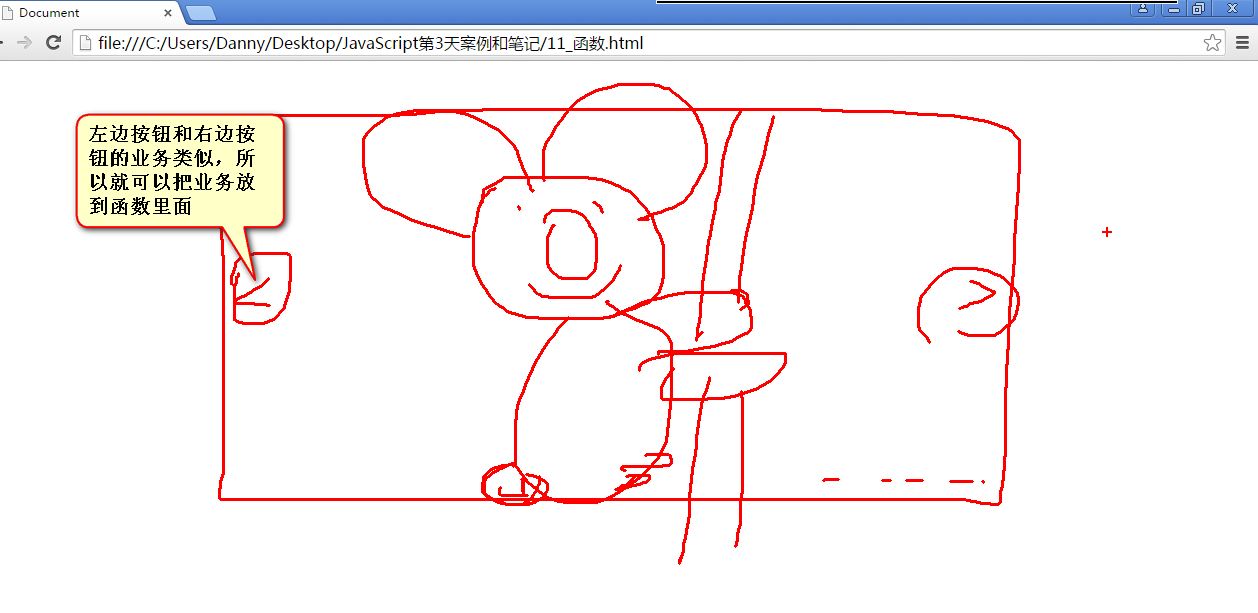
定义函数，可以在调用的后面：这是JS的语法特性，函数声明头的提升。知道就行了。

|  |
| --- |
| 1. console.log("你好"); 2. sayHello(); //调用函数 3. //定义函数： 4. function sayHello(){ 5. console.log("欢迎"); 6. console.log("welcome"); 7. } |

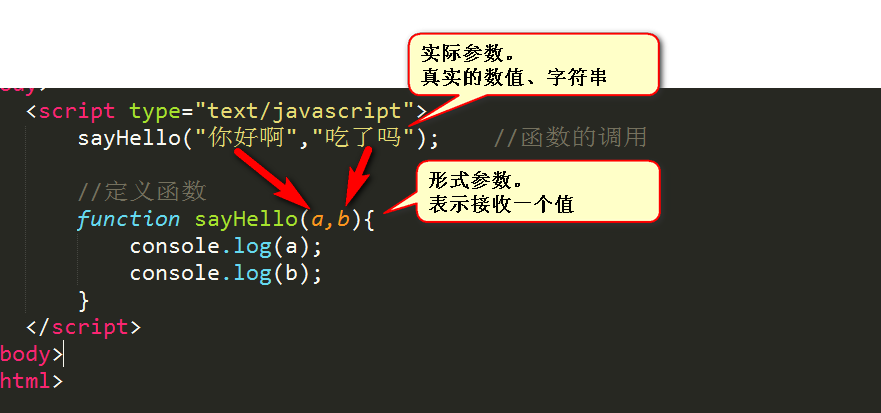
函数的功能、好处：

1） 将会被大量重复的语句写在函数里面，这样以后需要这些语句的时候，直接调用函数，不用重写那些语句。

2） 简化编程，让编程变的模块化。



## 3.2 函数的参数



实际参数和形式参数的个数，要相同。

|  |
| --- |
| 1. qiuhe(3,4); 2. qiuhe("3",4); 3. qiuhe("我爱你","中国"); 5. function qiuhe(a,b){ 6. console.log(a + b); 7. } |



## 3.3 函数的返回值

|  |
| --- |
| 1. <script type="text/javascript"> 2. console.log(**qiuhe(3,4)**); 3. function qiuhe(a,b){ 4. **return** a + b; 5. } 6. </script> |

return就是英语“返回”的意思，那么就表示此时这个“函数调用的表达式”（红色部分），值就是这个a+b。

● 函数里面可以没有return，如果有，只能有一个。不能有多个return；

● 函数里面，return后面不允许书写程序了，也就是说写在后面的程序无效；