# JavaScript基础入门

第3天课堂笔记（本课程共10天）

# 目录

[JavaScript基础入门](#_Toc15504)

[目录](#_Toc8534)

[一、上次课知识复习](#_Toc18515)

[1.1 晨早测验题目](#_Toc29722)

[1.2 昨天作业](#_Toc27770)

[1.3 运算符的复习](#_Toc13937)

[二、条件分支语句](#_Toc3261)

[2.1 if……else……](#_Toc17649)

[2.2 switch case语句](#_Toc9210)

[2.3 三元运算符](#_Toc1317)

[三、循环语句](#_Toc26435)

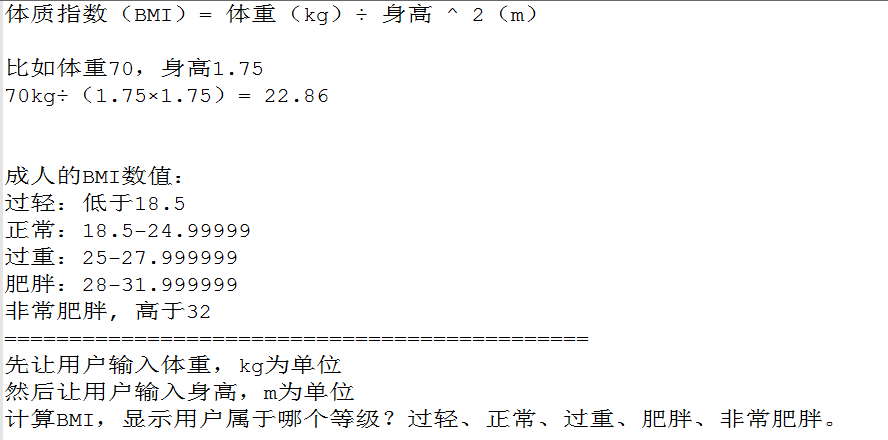
[3.1 for循环语句](#_Toc28344)

[3.2 穷举思想](#_Toc21208)

# 一、上次课知识复习

我们先来看看晨早测验的题目，和昨天的作业。

## 1.1 晨早测验题目



这道题最最美的地方，就是对跳楼现象的理解的考察。

|  |
| --- |
| 1. //第一步，用户输入两个数字 2. var weight = parseFloat(prompt("请输入体重(kg)",60)); 3. var height = parseFloat(prompt("请输入身高(m)",1.70)); 4. //第二步，计算BMI 5. var bmi = weight / Math.pow(height,2); 6. //第三步，盖if的楼进行判断，输出 7. if(bmi < 18.5){ 8. alert("过轻"); 9. }else if(bmi < 25){ 10. alert("正常"); 11. }else if(bmi < 28){ 12. alert("过重"); 13. }else if(bmi < 32){ 14. alert("肥胖") 15. }else{ 16. alert("非常肥胖"); 17. } |

## 1.2 昨天作业

**第1题：用户输入一个数字，然后判断这个数字能否被5、6整除的情况。**

解决一个事情，如何判断a是否能被5整除？

|  |
| --- |
| 1. a % 5 == 0 |

如果是真，那么就表示a能够被5整除。否则，不能的。

|  |
| --- |
| 1. //用户输入一个数字，然后判断这个数字能否被5、6整除的情况 2. //第一步，用户输入一个数字 3. var a = parseInt(prompt("请输入一个整数")); 4. //第二步，判断 5. if(a % 5 == 0 && a % 6 == 0){ 6. alert("这个数字同时能被5、6整除"); 7. }else if(a % 5 != 0 && a % 6 ==0){ 8. alert("这个数字能被6整除，但是不能被5整除"); 9. }else if(a % 5 == 0 && a % 6 !=0){ 10. alert("这个数字能被5整除，但是不能被6整除"); 11. }else if(a % 5 != 0 && a % 6 !=0){ 12. alert("这个数字不能被5、6整除"); 13. } |

**第2题：用户输入一个数字，然后再输入一个数字，判断两者的奇偶性是否相同。**

解决一个事情，如果判断奇数还是偶数？

|  |
| --- |
| 1. a % 2 == 0 |

结果是真，a就是偶数。

还有一种方法：

|  |
| --- |
| 1. a / 2 == parseInt(a / 2) |

结果是真，a就是偶数

**方法1：**

老师出这个作业的目的，是为了让大家练习复杂的“与、或”逻辑

|  |
| --- |
| 1. if(a % 2 == 0 && b % 2 == 0 || a % 2 != 0 && b % 2 != 0){ 2. alert("这两个数字的奇偶性相同"); 3. }else{ 4. alert("这两个数字的奇偶性不同"); 5. } |

**方法2：**

**如果a除以2的余数，等于b除以2的余数，那么两个数字同奇同偶**

|  |
| --- |
| 1. if(a % 2 == b % 2){ 2. alert("这两个数字的奇偶性相同"); 3. }else{ 4. alert("这两个数字的奇偶性不同"); 5. } |

**方法3：**

我们发现，偶数+偶数结果是偶数。偶数+奇数结果是奇数，奇数加奇数结果是偶数。

所以，两个数字相加，如果得到偶数，那么这两个数字同奇同偶。

|  |
| --- |
| 1. if((a + b) % 2 == 0){ 2. alert("这两个数字的奇偶性相同"); 3. }else{ 4. alert("这两个数字的奇偶性不同"); 5. } |

方法越来越简单，告诉我们一个道理：

算法为王！

**第3题：计算是否是闰年。**

**[闰年？](http://baike.baidu.com/link?url=ba77dwdVHUkQUfmbQUw3stLM1qRb57ymqopubGmiU5jZhsr222Lor8ABsahMW0Navcbqmp0cImrEdxOq4mImqvoZ5VIff80E9Iiwxcn1cSS)判断规则：**

如果这一年是100的整数倍，就是年费以00结尾的，那么除以400，整除就是闰年，有余数不是闰年；

如果这一年不是100的整数倍，那么除以4，整除就是闰年，有余数不是闰年。

**方法1：**

根据我们的题面的意思，分成两个大类；先判断是否是100的整数倍，然后分别进行判断。

伪代码：

|  |
| --- |
| 1. **if(这一年是100的整数倍){** 2. **if(年份除以400能整除){** 3. **alert("是闰年");** 4. **}else{** 5. **alert("不是闰年");** 6. **}** 7. **}else{** 8. **//这年不是100的整数倍** 9. **if(年份除以4能整除){** 10. **alert("是闰年");** 11. **}else{** 12. **alert("不是闰年");** 13. **}** 14. **}** |

**方法2：**

利用巧妙算法，连续书写“与”、“非”逻辑

如果（年份是100的整数倍并且能被400整除）或者（不能被100整除且能被4整除的年份），就是闰年。

|  |
| --- |
| 1. if(year % 100 == 0 && year % 400 == 0 || year % 100 != 0 && year % 4 == 0){ 2. alert("是闰年"); 3. }else{ 4. alert("不是闰年"); 5. } |

这里可以用一个布尔公式：(a && b) || (a && c) 等价于 a && (b || c) 简化**第4题：年终奖的计算**

某个公司要给员工发年终奖，为了奖励老员工，所以工作时间越长，发的越多，规则如下：

工作满0年 发月薪的1倍月薪年终奖，如果月薪大于8000，那么就是发1.2倍

工作满1年 发月薪的1.5倍月薪年终奖，如果月薪大于10000，那么就是发1.7倍

工作满2年，甚至更多 发月薪的3倍月薪年终奖，如果月薪大于12000，那么就是发3.2倍

用JS编写程序，让用户输入工作了几年，可以输入0，然后输入月薪。然后算年终奖。

|  |
| --- |
| 1. if(year == 0){ 2. //工作0年的 3. if(salary > 8000){ 4. xishu = 1.2; 5. }else{ 6. xishu = 1.0; 7. } 8. }else if(year == 1){ 9. //工作1年的 10. if(salary > 10000){ 11. xishu = 1.7; 12. }else{ 13. xishu = 1.5; 14. } 15. }else{ 16. //工作2年的 17. if(salary > 12000){ 18. xishu = 3.2; 19. }else{ 20. xishu = 3; 21. } 22. } 23. alert(xishu \* salary); |

## 1.3 运算符的复习

就不写为什么了，直接快速说答案：

|  |
| --- |
| 1. 1 + 2 \* 6 % 3  **//1** 2. false + true \* null **//0 ， 注意undefined是不能再数学运算中转的 NaN** 3. 6 + undefined **//NaN** 4. 0 / 0 **//NaN** 5. 6 / 0  **//Infinity** 6. Infinity - Infinity  **//NaN** |

|  |
| --- |
| 1. 5 == "5" **//t** 2. 5 === "5" **//f** 3. 5 != "5" **//f** 4. 5 !== "5" **//t** 5. "66" < "8" **//t** 6. 66 < "8" **//f ， 因为有一个数字，就按数字比了，”8”就转为8了** 7. "66" < 8 **//f** 8. false == 0 **//t** 9. true == 1 **//t** 10. NaN == NaN **//f** 11. NaN != NaN  **//true** 12. null == 0 **//false , 因为null是一个对象，引用类型值，和基本类型值比，这个你着重记忆一下。** 13. 3 > 2 > 1 **//false** |

|  |
| --- |
| 1. false || true && !!!false  **//true** 2. false && 8 **//false** 3. 3 && 4 **//4** 4. "" && 6 **//””** 5. 6 && undefined **//undefined** 6. null && undefined **//null** 7. 3 || 4 **//3** 8. "" || 18  **//18** |

|  |
| --- |
| 1. var a = 1; **//a的值1** 2. var b = a++;  **//b的值是1，a的值是2 。++写在后面，是先用再加** 3. a += ++b;  **//先把b加1，b就是2了，a原来是2，加上2，就是4** 4. console.log(a);  **//4** 5. console.log(b); **//2** |



下面这个很难：

|  |
| --- |
| 1. var a = 3; **//a的值是3** 2. console.log(a++ + a++);  **//计算从左到右的，所以第一次a++时用的是3，a变为4。** 3. **//第二次遇见a++的时候，a用的是4，变为5** 4. **//3 + 4 所以输出7** 5. console.log(a); **//输出5** |



|  |
| --- |
| 1. var a = 3; 2. console.log(a++ + ++a); **//8** 3. console.log(a); **//5** |



综合顺序：贴身的、数学、关系、逻辑、赋值

|  |
| --- |
| 1. var a = 4; 2. 1 && 2 + 3 \* a++ % 5 || 6 > 7 == 8 / !false |

解：原式 = 1 && 2 + 3 \* a++ % 5 || 6 > 7 == 8 / !false

= 1 && 2 + 3 \* 4 % 5 || 6 > 7 == 8 / true **先算贴身的**

= 1 && 2 + 2 || 6 > 7 == 8  **算数学的**

= 1 && 4 || 6 > 7 == 8 **算数学的**

= 1 && 4 || false == 8  **算关系的**

= 1 && 4 || false **算关系的**

= 4 || false **算逻辑的**

= 4 **算逻辑的**



# 二、条件分支语句

条件分支语句，也叫作条件判断语句，就是根据某种条件执行某些语句，不执行某些语句。

JS中有三种语法是可以表示条件分支的

## 2.1 if……else……

条件分支的主力语法，这个主力语法能够书写所有的条件分支语句。也就是说，一会儿我们学switch case语句，switch case语句能干的，if else语句一定也能干。

昨天已经讲过了，复习一下：

标准的完整结构：

|  |
| --- |
| 1. if(){ 3. }else if(){ 5. }else if(){ 7. }else{ 9. } |

**else只能有一个，并且一定要放在最后。表示所有的分支都不满足的时候执行的事情**。

可以没有else：

|  |
| --- |
| 1. if(){ 3. }else if(){ 5. }else if(){ 7. } |

可以没有分支：

|  |
| --- |
| 1. if(){ 3. } |

看准楼的位置：

|  |
| --- |
| 1. if(8 > 5){ 2. console.log("哈哈"); 3. } 4. console.log("嘻嘻"); //这行语句不在if里，所以不管if成功没有，都执行 |



如果只有一行语句，那么就可以省略大括号：

|  |
| --- |
| 1. if(8 > 15) 2. console.log("哈哈"); 3. console.log("嘻嘻"); |

跳楼现象，要会合理利用，比如今天早上的bmi晨早测验！

|  |
| --- |
| 1. var a = 4; 2. if(a < 8){ 3. a++; //a变为5，然后跳楼，不会进行下面的a==5的比较了 4. }else if(a == 5){ 5. a++; 6. } 7. console.log(a); //5 |

## 2.2 [switch case语句？](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Statements/switch)

先来看一个简单的例子：

|  |
| --- |
| 1. **switch**(**xingzuo**){ 2. **case "白羊座"** : 3. alert("今天要注意调整自己的状态，有些人的睡眠质量比较差"); 4. **break**; 5. **case "金牛座"** : 6. alert("今天会在感情方面遇到一些小小的问题"); 7. **break**; 8. **case "双子座"** : 9. alert("要做好心理准备会在工作上被他人挑剔找茬，这是你避免不了的事情"); 10. **break**; 11. **case "巨蟹座"**: 12. alert("工作上要打醒十二分精神，要不然很大可能会经常要返工哦"); 13. **break**; 14. **case "狮子座"**: 15. alert("恋爱中的人今天的心有点摇摆不定，有人可能会有精神上的出轨"); 16. **break**; 17. **default** : 18. alert("对不起，我们没有收录你的星座运势！"); 19. **break**; 20. } |

语法：

|  |
| --- |
| 1. **switch**(待检测值){ 2. **case** 值1 : 3. 值1 与 待检测值 相同时做的事情 4. **break**; 5. **case** 值2 : 6. 值2 与 待检测值 相同时做的事情 7. **break**; 8. **case** 值3 : 9. 值3 与 待检测值 相同时做的事情 10. **break**; 11. **default** : 12. 默认要做的 13. **break**; 14. } |

switch case语句的哲学是什么？存在的意义，简化了if……else……的书写：

你会发现，下面的if语句的分支中，都是在判断xingzuo是不是和某一个字符串相等，显得臃肿：

|  |
| --- |
| 1. if(xingzuo == "白羊座"){ 2. alert("今天要注意调整自己的状态，有些人的睡眠质量比较差"); 3. }else if(xingzuo == "金牛座"){ 4. alert("今天会在感情方面遇到一些小小的问题"); 5. }else if(xingzuo == "双子座"){ 6. alert("要做好心理准备会在工作上被他人挑剔找茬，这是你避免不了的事情"); 7. }else if(xingzuo == "巨蟹座"){ 8. alert("工作上要打醒十二分精神，要不然很大可能会经常要返工哦"); 9. }else if(xingzuo == "狮子座"){ 10. alert("恋爱中的人今天的心有点摇摆不定，有人可能会有精神上的出轨"); 11. }else{ 12. alert("对不起，我们没有收录你的星座运势！"); 13. } |

所以，这种要将变量依次和不同的值比较，区分情况的时候，用switch case语句。

比较一下：

|  |  |
| --- | --- |
|  |  |

我们经过测试，发现switch进行相同判断，底层是===的比较，连类型一起比较

|  |
| --- |
| 1. <script type="text/javascript"> 2. var a = **"5"**; 3. switch(a){ 4. case **5** : 5. alert("哈哈"); **//不会弹出！！因为类型不一样** 6. break; 7. } 8. </script> |

如果不写break，那么switch语句就表现的非要有意思，除了执行这个case里面的语句之外，还将无条件的执行下面的case的语句，直到遇见一个break拦住它。

|  |
| --- |
| 1. var a = 5; 2. switch(a){ 3. case 4: 4. alert("我是4"); **//没有通过验证，不执行** 5. case 5: 6. alert("我是5"); **//弹出** 7. case 6: **//无视这行case，直接直接这行case里面的语句** 8. alert("我是6"); **//弹出，** 9. break; **//挡住了，就不能继续往下运行了** 10. case 7: 11. alert("我是7"); 12. break; 13. default: 14. alert("我是默认"); 15. } |

//用户输入月份，告诉它这个月有几天

|  |
| --- |
| 1. switch(month){ 2. **case** 1: 3. **case** 3: 4. **case** 5: 5. **case** 7: 6. **case** 8: 7. **case** 10: 8. **case** 12: 9. alert("这个月有31天"); 10. **break**; 11. **case** 4: 12. **case** 6: 13. **case** 9: 14. **case** 11: 15. alert("这个月有30天"); 16. break; 17. **case** 2: 18. alert("这个月一般来说是28天，可能29天"); 19. } |

初学者最最可爱的错误是：

|  |
| --- |
| 1. switch(month){ 2. case **~~1 || 3 || 5 || 7 || 8 || 10 || 12~~** : 3. alert("这个月有31天"); 4. break; 5. case **~~4 || 6 || 9 || 11~~** : 6. alert("这个月有30天"); 7. break; 8. default : 9. alert("我是默认"); 10. } |

这样写不行的！因为1 || 3 || 5 || 7叫做表达式，计算机一遇见表达式，就会计算求值

|  |
| --- |
| 1. 1 || 3 || 5 || 7 || 8 || 10 || 12 |

表达式的值是1，等价于case 1:

switch这个语句，就只能干这种事儿。就是把一个值，看情况得几，执行不同的事儿。

如果要在某一个范围内，做某件事儿，也行，不过不建议这么写。此时应该用if……else if……

|  |
| --- |
| 1. switch(**true**){ 2. **case** score < 60 : 3. alert("不及格"); 4. **break**; 5. **case** score < 75 : 6. alert("及格"); 7. **break**; 8. **case** score < 85 : 9. alert("良好"); 10. **break**; 11. **case** score <= 100 : 12. alert("优秀"); 13. **break**; 14. **default** : 15. alert("你输入的成绩错误"); 16. } |

## 2.3 三元运算符

? :是一组运算符，这是JS中唯一一个需要三个元素参加的运算符。

|  |
| --- |
| 1. 条件 ? val1 : val2 |

表达式的值，要看条件是true还是false。如果条件是true，那么表达式的值就是val1。如果条件是false，表达式的值就是val2。

简单的例子：

|  |
| --- |
| 1. console.log(true ? 3 : 8); //3 2. console.log(false ? 3 : 8); //8 |



三元运算符，经常这么使用：

员工的年终奖倍数，跟月薪有关。月薪大于8000，系数就是1.2。否则是1。

|  |
| --- |
| 1. var **xishu** = salary > 8000 **?** 1.2 **:** 1; |

等价于：

|  |
| --- |
| 1. if(salary > 8000){ 2. **xishu** = 1.2; 3. }else{ 4. **xishu** = 1; 5. } |

现在总结一下，条件分支的语句一共有3种，但是&& ||短路写法也有一种选择的感觉，总共算4种。

if……else if…… （必须立即想到！这个东西无敌）

switch case （某一方面很擅长，体现在了代码的清晰度上）

三元运算符 （赋值的时候，根据条件有两个可能的值，马上想到三元运算符）

&&短路写法 （经验问题，用的不熟悉没事）

# 三、循环语句

JS中流程控制的语句，就两个：条件分支、循环语句。靠这两种语句，就能完成所有的程序。

循环语句是一系列反复执行直到符合特定条件的命令。

循环语句能够解决大量的类似语句书写。

## 3.1 f[or循环语句？](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Statements/for)

整体感知：

|  |
| --- |
| 1. for(var i = 1 ; i <= 99 ; i++){ 2. console.log("第" + i + "次说我爱你"); 3. } |



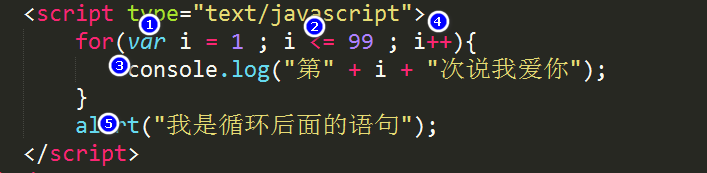
你就发现了，for循环就是一个壳子，让变量i依次、轮流的值为1、2、3、4、……99。

变量i我们称为循环变量，这个变量可以任意设置，我们习惯用i来表示循环变量

语法上，最最容易错误的，最后没有分号：

|  |
| --- |
| 1. for(var i = 1 ; i <= 99 ; i++**;**){ ← 错误的写法 2. console.log("第" + i + "次说我爱你"); 3. } |

for循环的本质，必须搞清楚。

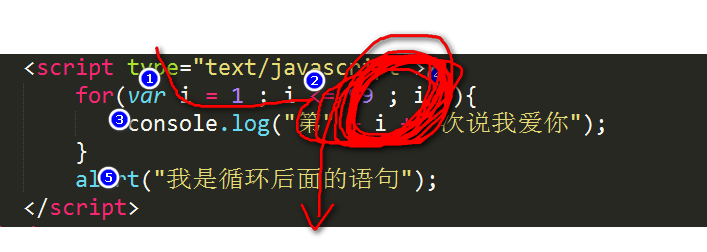


系统遇见了for循环结构，会立即执行语句①，此时声明了一个变量i，赋值为1。

系统会立即检测，是否满足②这个条件表达式，如果是真，则执行③；如果②是假，则跳出循环，执行语句⑤。

执行完③之后，系统会立即执行语句④，然后再次检测语句②，如果为真，则做③，如果为假，则做⑤；

执行完③之后，系统会立即执行语句④，……



for循环的本质，决定了一个事儿，就是你能否预测循环的结果。用20个题目训练这个知识点。

|  |
| --- |
| 1. for(var i = 3 ; i **<** 11 ; i+=4){ 2. console.log(i); 3. } |



|  |
| --- |
| 1. for(var i = 4 ; i **<=** 16 ; i+=4){ 2. console.log(i); 3. } |



|  |
| --- |
| 1. for(var i = 4 ; i **!=** 10 ; i+=2){ 3. } |



|  |
| --- |
| 1. for(var i = 6 ; i < 4 ; i++){ 2. console.log(i); 3. } |

寂寞



系统遇见了for循环，会执行语句var i = 6; ，立即进行一次i<4的判断，结果是假。所以，一次也不执行里面的语句，直接退出循环。

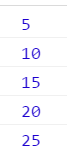
|  |
| --- |
| 1. for(var i = 2 ; i < 12 ; i+=3){ 2. i += 4; 3. console.log(i); 4. } |



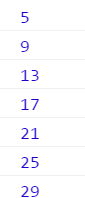
|  |
| --- |
| 1. for(var i = 1 ; i < 10 ; i++){ 2. //里面没有语句 3. } 4. console.log(i); |



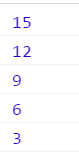
|  |
| --- |
| 1. for(var i = 2 ; i < 30 ; i++){ 2. if(i % 5 == 0){ 3. console.log(i); 4. } 5. } |



|  |
| --- |
| 1. for(var i = 2 ; i <= 26 ; i++){ 2. if(i % 2 == 0){ 3. i += 3; 4. }else{ 5. i += 2; 6. } 7. console.log(i); 8. } |



|  |
| --- |
| 1. for(var i = 15 ; i > 2 ; i-=3){ 2. console.log(i); 3. } |



当循环变量已经在外部定义了，并不需要在①号位上重新定义，写一个空的;

|  |
| --- |
| 1. var i = 8; 2. for(**;** i<12;i++){ 3. console.log(i); 4. } |

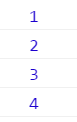


|  |
| --- |
| 1. for(var m = 1 , n = 2 ; m + n < 12 ; m++){ 2. n++; 3. console.log(m); 4. } |



|  |
| --- |
| 1. var i =1; 2. for(alert("你好");i<5;i++){ 3. console.log(i); 4. } |

先弹出你好，然后输出：



## 3.2穷举思想

[暴力破解法？](https://zh.wikipedia.org/wiki/%E6%9A%B4%E5%8A%9B%E7%A0%B4%E8%A7%A3%E6%B3%95)

我们的for没有讲完，for里面有continue、break的小关键字，我们下节课讲。

我们初学者，最最关键的，就是掌握for循环的穷举思想。

比如，现在要寻找48的所有约数。

什么是约数： a ÷ b 没有余数，能整除。那么我们称a是b的倍数，称b是a的约数。

计算机是笨笨的，不能立即告诉你48能整除哪些数，所以，我们就要让计算机从1开始，一个一个实验。

我们试试1是不是48的约数

我们试试2是不是48的约数

我们试试3是不是48的约数

……

我们试试48是不是48的约数

把所有的可能性一一列出，然后筛选。全举法，举就是列举的意思。中文比较文绉绉，穷举法。

欲穷千里目，更上一层楼。

比如，我们现在要寻找水仙花数。水仙花数，是三位数，每个数位的立方和等于它本身。

比如153， 因为13+53+33 = 1 + 125 + 27 = 153。所以153就是水仙花数。

寻找100~999的所有水仙花数。

穷举法，从100~999，一个一个试，

我们试试100是不是水仙花数

我们试试101是不是水仙花数

我们试试102是不是水仙花数

我们试试103是不是水仙花数

……

我们试试999是不是水仙花数

for循环，还有一个小应用，叫做**累加器**。

比如，我们现在想计算1+2+3+4+……100

|  |
| --- |
| 1. var sum = 0; **//累加器** 2. for(var i = 1 ; i <= 100 ; i++){ 3. sum = sum + i; 4. } 5. console.log(sum); |

计算6！

|  |
| --- |
| 1. var sum = 1;  **//累乘器，累乘器的初值是1** 2. for(var i = 6 ; i >= 1 ; i--){ 3. sum = sum \* i; 4. } 5. console.log(sum); |

现在我们想计算一下48的约数的个数。注意，我们不关心它的约数是什么，我们只关心多少个。

当一个数字的约数个数，是2个，也就是只有1和它自己，我们称呼它为质数，也叫做[素数](http://baike.baidu.com/link?url=PDeIt9lWQEhmYrZ-u97A3tbS0OO8JQQawN7JAMW5wvUt4vu9oQk-_-lwdxWwQRIPHIWurR4v_cup9NVyVhDwnaGugoZetACLJ1IxQdk-3VG)。

2、3、5、7、11、13、17、19、23、29、31、37、41……

寻找1~100的所有质数