# 复习

运算符:

数学运算符：

纯数字字符串，true,false,null参与运算隐式转换。

纯数字字符串参与+,仍然是拼接作用

比较运算符：结果一定是布尔值

纯数字字符串，true,false,null参与运算隐式转换

NaN,Infinity不会发生隐式转换。

|  |
| --- |
| 1. console.log(NaN == NaN); //false |

两侧都是字符串比较时，不是比较长短，编码越靠越大（数字，大写字母，小写字母）

逻辑运算符：

隐式真：非0数字，非空字符串

隐式假：“”， 0， NaN，undefined, null

短路语法：

a && b : 如果a真，结果是b;如果a假，结果a

a || b : 如果a真，结果是a；如果a假，结果b

赋值运算符：

=：

条件分支语句：根据条件让某些语句先执行，某些语句不执行

if (条件表达式) {

满足条件表达式，if语句后面的结构体才会执行

}else {

不满足条件表达式，else语句后面的结构体才会执行。

}

# 条件表达式

## 三元表达式

三元表达式（三元运算符）：

语法：

|  |
| --- |
| 1. 条件表达式 ？ 值1 : 值2; |

当条件表达式为真时，三元表达式结果是值1；

当条件表达式为假时，三元表达式结果是值2；

|  |
| --- |
| 1. var sum = false ? 1 : 0; |

当结果是二选一时使用三元表达式更简单。

当工作年限不满1年，工资小于8000，年终奖为工资的1倍，否则是1.2倍；

当工作年限不满2年，工资小于10000，年终奖为工资的1.5倍，否则是1.7倍；

当工作年限等于超过2年，工资小于13000，年终奖为工资的2.3倍，否则是3倍；

用户输入年限和工资，告诉他对应的年终奖

|  |
| --- |
| 1. // 的得到年限,0。9年 取整0 2. var year = parseInt(prompt("请输入工作年限")); 3. // 工资 4. var salary = parseInt(prompt("请输入工资")); 5. // 倍数 6. var beishu = 0; 7. // 外层 使用多分支if语句，根据年限划分条件 8. if (year === 0) { 9. // 倍数1,1.2 10. if (salary < 8000) { 11. beishu = 1; 12. }else { 13. beishu = 1.2 14. } 15. }else if (year === 1) { 16. beishu = salary < 10000 ? 1.5 : 1.7; 17. }else if (year >= 2) { 18. beishu = salary >= 13000 ? 3 : 2.3 ; 19. } 20. // 输出，字符串拼接 21. console.log("该用户工作年限是" + year + "年，工资是" + salary + "元，对应的年终奖是" + (beishu \* salary) + "元。"); |

## 1.2 if语句嵌套

|  |
| --- |
| 1. /\* 2. 男性大于等于22岁可以结婚；女性大于等于20可以结婚 3. 用户输入年龄，性别判断是否可以结婚 4. \*/ 5. // 性别 6. var sex = prompt("请输入性别"); 7. // 年龄 8. var age = parseInt(prompt("请输入年龄")); 9. // 外层使用性别 10. if (sex === "男性") { 11. if (age >= 22) { 12. // 该结构体执行： 同时满足两个条件 sex===男性 age >= 22 13. console.log("该男性可以结婚"); 14. }else { 15. // age <22 16. console.log("该男性不可以结婚"); 17. } 18. console.log("该用户是男性"); 19. }else if (sex === "女性") { 20. if (age >= 20) { 21. console.log("女性可以结婚"); 22. }else { 23. console.log("女性不可以结婚"); 24. } 25. console.log("女性查询完毕"); 26. } 27. console.log("查询结束"); |

## 1.3 switch语句

开头语句，允许程序去计算一个表示式的值，然后让这个值去匹配case标签。匹配成功执行该case标签的执行体。

|  |
| --- |
| 1. switch （表达式）{ 2. case 值1： 3. 执行体1; 4. break; 5. case 值2： 6. 执行体2; 7. break; 8. case 值3： 9. 执行体; 10. break; 11. …… 12. default: 13. 执行体; 14. break; 15. } |

|  |
| --- |
| 1. // 用户输入星座，输出对应的运势 2. // 获取星座 3. var xingzuo = prompt("请输入星座"); 4. switch (xingzuo) { 5. case "白羊座": 6. console.log("白羊座"); 7. console.log("白羊座今日运势普通，注意处理好自己的负面情绪，不要让它影响你的生活、工作。感情方面运势一般，你可能会希望得到伴侣更多地关注和关心，但你传递给他/她很大地压力感让人"); 8. break; 9. case "金牛座": 10. console.log("金牛座"); 11. console.log("金牛座今日运势稍弱，遇事不要慌张，越紧张头脑越不清楚，会影响你的判断力。感情方面运势一般，伴侣的一个无心举动都可能会惹得你不高兴，情绪不佳时不妨独处一会儿。"); 12. break; 13. case "双子座": 14. console.log("双子座"); 15. console.log("双子座今日运势较好，抗压能力强，面对困难不会轻易气馁的你适合尝试完成高难度的事情。"); 16. break; 17. default: 18. console.log("没有你输入的星座换一个吧"); 19. break; 20. } 21. console.log("查询结束"); |

break: 匹配case执行后面的break语句，不再继续向下执行后面的语句，立即跳出switch语句，执行switch后面其他语句。

default: 当以上case都没有匹配时，执行default后面的语句。

default语句中的break可以省略，当执行default后面的语句结束，也会跳出switch语句

|  |
| --- |
| 1. // 获取数字 2. var num = parseInt(prompt("请输入数字")); 3. switch (num) { 4. case 1: 5. console.log(num); 6. break; 7. case 2: 8. console.log(++ num); 9. case 3: 10. console.log(num ++); 11. case 4: 12. console.log(num \*= 2); 13. break; 14. default: 15. console.log(num + "没有匹配结果"); 16. } 17. /\* 18. num = 2; 19. 匹配case 2执行后面的语句 20. console.log(3); num=3 21. 没有break，语句不会结束，继续向下执行，虽然不匹配case 3，也会执行， 22. 输出console.log(3) num=4 23. 没有Break继续执行 24. console.log(8) 25. 执行break，立即跳出switch语句 26. \*/ |

利用break案例：

|  |
| --- |
| 1. /\* 2. 1,3,5,7,8,10,12 31 3. 4,6,9,11 30 4. 2 28 5. \*/ 6. // 获取月份 7. var month = parseInt(prompt("请输入月份")); 8. switch (month) { 9. case 2: 10. console.log(month + "月有28天"); 11. break; 12. case 4: 13. case 6: 14. case 9: 15. case 11: 16. console.log(month + "月有30天"); 17. break; 18. default: 19. console.log(month + "月有31天"); 20. } 21. console.log("查询结束"); |

另类写法：

switch后面直接书写true,匹配case 后面书写的条件表达式

|  |
| --- |
| 1. //分数案例 2. // 的到分数 3. var score = parseInt(prompt("请输入分数")); 4. // 计算机遇到所有的表达式,都是先计算 5. switch (true) { 6. case score >= 90: 7. console.log("优秀"); 8. break; 9. case score >= 80: 10. console.log("良好"); 11. break; 12. case score >= 70: 13. console.log("还行"); 14. break; 15. default: 16. console.log("加油，超越自己"); 17. } 18. console.log("查询结束"); |

总结：if语句是使用最广泛。

三元表达式当结果是二选一时，使用较简单

switch语句当值有多种可能性时。

# 二、循环语句

循环语句就是重复执行一段代码直到遇见某个指令时结束该循环。

直白解释：就是重复执行一段代码直到，条件为假时跳出循环。

for循环，do while，while 。

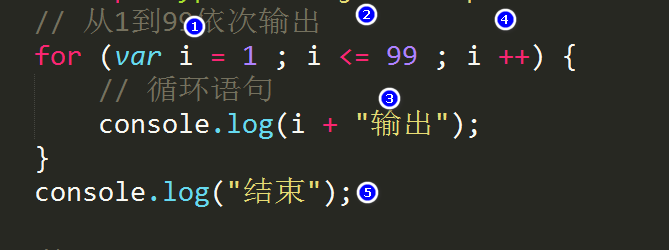
## 2.1 for循环

for循环，前测试循环语句。先判断是否满足循环条件，当满足条件，循环语句才会执行。

for语句不固定

|  |
| --- |
| 1. for (循环变量初始值; 循环条件; 步长) { 2. 循环语句; 3. } |

for机理：for循环判断条件表达式，当条件表达式为真时执行循环语句，直到条件为假时跳出循环。结束循环后再执行for循环后面的语句。



for循环的过程：

for循环第一次执行时，首先执行一次语句1，然后执行语句2，判断是否满足条件，当结果为真，向内执行循环语句3；如果为假，直接跳出for循环，执行语句5。

如果能进入语句3，然后执行语句4之后。再执行语句2，判断是否满足条件，当条件为真是继续向内执行语句3；如果条件为假时，跳出for循环。执行语句5.……

当条件为真时，重复执行语句2，语句3，语句4.直到条件为假时，跳出for循环，执行语句5.

我们了解for循环的过程可以宏观看for循环，可以不用输出直接根据过程得出for循环的结果。

1for语句中，循环变量i是全局变量。可以书写在for语句外面。一般循环变量i,j,k

|  |
| --- |
| 1. // i全局变量，可以书写在for语句外面 2. var i = 3; 3. for (; i < 8 ; i += 2) { 4. console.log(i); 5. } |

2 循环条件可以书写成任何形式，结果一定是布尔值。

|  |
| --- |
| 1. for (var i = 4; i != 10 ; i += 3) { 2. console.log("输出"); 3. } 4. console.log(i); |

3 步长可以自加，可以自减

|  |
| --- |
| 1. // 4步长，可以自加可以自减 2. for (var i = 15; i > 5; i -= 4) { 3. console.log(i); 4. } |

4循环语句有可能一次都不执行

|  |
| --- |
| 1. // 循环语句有可能一次都不执行 2. var i = 3; 3. for (console.log(++ i); i < 1 ; i ++) { 4. console.log(i + "输出"); 5. } 6. console.log(i + "结束"); |

5循环语句可以一直做，条件一直为真，死循环

|  |
| --- |
| 1. for (var i = 1; i > 0 ; i ++) { 2. console.log("输出"); 3. } 4. console.log("结束"); |

6for语句可以嵌套if语句

|  |
| --- |
| 1. for (var i = 3; i < 20 ; i += 4) { 2. // 当i能够被5整除才输出i 3. if (i % 5 === 0) { 4. console.log(i + "能被5整除"); 5. } 6. } 7. console.log("for循环结束"); |

7 for循环可以嵌套for循环，循环变量不能相同

|  |
| --- |
| 1. for (var i = 1; i <= 3 ; i ++) { 2. for (var j = 1; j <= 3 ; j ++) { 3. console.log(i,j); 4. } 5. console.log(j); 6. } 7. console.log(i); 8. console.log(j); |

## 2.2 穷举思想

概念：我们经常想得到一组数据，有一些特定的要求，计算机没法自己帮我们输出这些数据。我们人为的需要编写一段程序，让计算机去帮我们实现程序。将所有的可能情况，一一的列举出来，然后我们人为限定判断条件，把符合条件的数据就给它输出，不满足的就跳过，继续验证下一个数据是否满足条件，直到把所有可能情况都验证一个遍。这个方法就叫做全举法，也叫穷举法。

外层：一一列举，for循环。

内层：进行判断，if语句。

案例1：在控制台输出一个数的所有约数。

约数概念：a %b ==0，那么a叫做b的倍数，b叫做a的约数或者因数。

一个数的约数，最小是1，最大是它本身。

|  |
| --- |
| 1. var num = parseInt(prompt("请输入数字")); 2. for (var i = 1 ; i <= num ; i ++) { 3. // 控制台只想要num约数 4. // 判断i如果是num的约数，才输出i 5. if (num % i === 0) { 6. console.log(i); 7. } 8. } 9. console.log(i + "结束"); |

案例2：找三位数里的“水仙花数”，在控制台输出。

条件1：三位数，得全部列举出来100,~999

条件2：水仙花数，每一个位数的三次方，他们的和，还等于原来的三位数通过判断找到符合条件的水仙花数

|  |
| --- |
| 1. // 声明变量 2. var ge; 3. var shi; 4. var bai; 5. // 求和 6. var sum; 7. for (var i = 100 ; i < 1000; i ++) { 8. // 限制i = 134 9. ge = i % 10; 10. shi = parseInt(i / 10) % 10; 11. bai = parseInt(i / 100); 12. sum = Math.pow(ge, 3) + Math.pow(shi, 3) + Math.pow(bai, 3); 13. if (sum === i) { 14. console.log(i + "是水仙花数"); 15. } 16. } |

## 2.3 累加器和累乘器

累加器：我们不想输出所有的可能值，只想输出一个最后满足条件的总数。

1 累加器是变量，初始值0，累加器变量声明必须书写在for循环外面。

2 累加器可以自加，有一个满足条件的结果就+1

3 最终结果在for循环外面输出

用户输入一个数字，输出该数字约数总个数。

|  |
| --- |
| 1. // 1的到用户输入的数字6 2. var num = parseInt(prompt("请输入数字")); 3. // 1声明累加器，必须书写在for循环外面，初始值0 4. var sum = 0; 5. // 穷举思想一一列举 6. // 约数1-本身 7. for (var i = 1; i <= num ; i ++) { 9. // 找到一个满足条件的数据累加器就自加1 10. // 2如果i是num的约数，累加器自加1 11. if (num % i === 0) { 12. sum ++; 13. } 14. } 15. // 3累加器在for循环外面输出 16. console.log(num + "约数总个数是" + sum); |

累乘器：保存我们前面乘数的积。

阶乘：8！= 8 \* 7 \* 6 \* 5 \* 4 \* 3 \*2 \*1

注意：

①累乘器的变量必须写在循环外面。

②初始值必须是1.

③输出也必须是在for循环完毕之后。

|  |
| --- |
| 1. // 的到用户输入的数字5 2. var num = parseInt(prompt("请输入数字")); 3. // 阶乘5! = 5\*4\*3\*2\*1 4. // 注意1：累乘器声明书写在for循环外，初始值1 5. var cheng = 1; 6. for (var i = num ; i >= 1 ; i --) { 7. // 注意2 累乘器使用赋值语句 8. cheng \*= i; 9. } 10. // 注意3结果在for循环外输出 11. console.log(num + "阶乘是" + cheng); |

用户输入一个数字，判断是不是质数。

质数：一个数除了1和他本身之外没有其他的约数

1不是质数；2是质数；7是质数

|  |
| --- |
| 1. // 的到用户输入的数字 2. var num = parseInt(prompt("请输入数字")); 3. // 判断该数是不是质数，该数约数总个数是2就是质数 4. // 注意1声明必须书写在for外面 5. var sum = 0; 6. // 找约数一定是穷举思想 7. // 一一列举1-本身之间所有的数 8. for (var i = num ; i >= 1 ; i --) { 9. // 如果i是num的约数，sum累加器自加1 10. if (num % i === 0) { 11. // i是num的约数 12. sum ++; 13. } 14. } 15. // 注意sum累加器输出也是在for循环结束 16. if (sum === 2) { 17. console.log(num + "是质数"); 18. } |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |