# 第13天【运算符和分支、循环结构】

## 主要内容

1. 算数运算符
2. 比较运算符
3. 赋值运算符
4. 逻辑运算符
5. 分支结构
6. 循环结构
7. Break和continue

## 学习目标

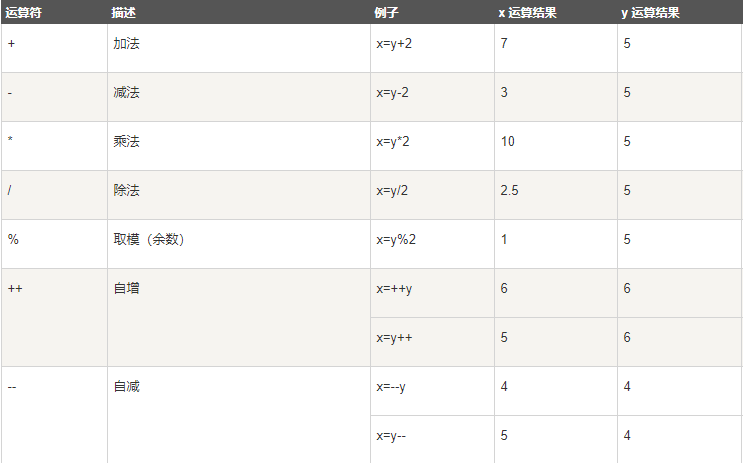
|  |  |  |
| --- | --- | --- |
| 节数 | 知识点 | 要求 |
| 第一节 | 算数运算符 | 掌握 |
| 第二节 | 比较运算符 | 掌握 |
| 赋值运算符 | 掌握 |
| 逻辑运算符 | 掌握 |
| 第三节 | 单分支结构 | 掌握 |
| 双分支结构 | 掌握 |
| 多分支结构 | 掌握 |
| 分支结构嵌套 | 掌握 |
| 第四节 | 循环结构 | 掌握 |
| 循环结构嵌套 | 了解 |
| Break和continue | 了解 |

1. **运算符**

运算符叫做operator，也可以叫做操作符。运算符有很多种，一元运算符、二元运算符；数学运算符、逻辑运算符……

**1.1算数运算符**

y=5，下面的表格解释了这些算术运算符：



|  |
| --- |
| console.log(1 + 2);//3  console.log(8 - 5);//3  console.log(1.5 \* 2);//3  console.log(12 / 4);//3  console.log(13 % 5);//3 要余数 |

取余数这个运算，实际上也是除，要的是余数：

|  |
| --- |
| console.log(12 % 3); //0  console.log(121 % 11); //0  console.log(5 % 8); //5  console.log(8 % 5); //3  console.log(5 % 5); //0 |

默认的计算顺序，先乘除，后加减。**乘除取余是平级，先遇见谁，就算谁**。

|  |
| --- |
| console.log(1 + 2 \* 3); //7  console.log(1 + 2 \* 3 % 3); //1  console.log(1 + 2 % 3 \* 3); //7 |

我们可以用小括号来改变计算先后顺序，注意没有中括号和大括号，一律用小括号。

|  |
| --- |
| var a = 4 \* (3 + (1 + 2) \* 3);  alert(a);//48 |

隐式转换，所有带有字符串的运算都会尽可能的转为数字进行计算，加号比较特殊，遇到字符串直接进行字符串拼接。

数学运算符的正统，是number和number的数学运算，结果是number。出于面试的考虑，有一些奇奇怪怪的数学运算。

数学运算中，只有纯数字字符串、布尔值、null能够帮你进行隐式转换。

|  |
| --- |
| //隐式转换：就是没有写parseInt()、parseFloat()自己帮你转格式  console.log(3 \* "8"); //24  console.log("3" \* "8"); //24  console.log("48" / "2"); //24  console.log("24" % 55); //24 |

|  |
| --- |
| console.log(3 \* null); //0 隐式转换的时候null将被转为0  console.log(3 \* false); //0 隐式转换的时候false将被转为0  console.log(3 \* true); //3 隐式转换的时候true将被转为1 |

不纯的字符串和undefined是不能帮你进行隐式转换的，结果都是NaN

|  |
| --- |
| console.log(3 \* "8天"); //NaN 数学运算中，不纯的字符串没法隐式转换  console.log(3 \* undefined); //NaN 数学运算中，undefined不能进行隐式转换 |

加法比较特殊，因为+同时是加法和连字符的符号，所以加法在面对字符串的时候没有隐式转换。

|  |
| --- |
| //加法没有隐式转换,遇到字符串直接进行字符串拼接  console.log(3 + "8"); //38  console.log(3 + undefined); //NaN  console.log(3 + null); //3  console.log(3 + false); //3  console.log(3 + true); //4  console.log(3 + "true"); //3true |

总结：

1、无论哪种运算，只要出现了undefined参与运算，结果都是NaN。

2、然后"4"、false、true、null都能进行隐式转换，true转换为1，false和null转换为0进行计算。

3、加号比较特殊，遇到字符串没有隐式转换，直接进行字符串拼接

**单目运算符（自增自减运算符）**

单目运算符只需要一个运算子即可计算，表示对自身进行增加或者减少。使用时需要注意：

a.单目运算符放在运算子之前，先改变运算子本身，在参与计算结果。

|  |
| --- |
| var i = 1;  console.log(++i);//2 |

b.单目运算符放在运算子之后，先计算运算结果，语句执行完毕后在改变运算子本

|  |
| --- |
| var i = 1;  console.log(i++);//1 |

|  |
| --- |
| var a = 10;  a++;  console.log(a); //11 |
| var a = 10;  ++a;  console.log(a);//11 |

总结：不管前++还是后++，对于变量本身来说没有区别

|  |
| --- |
| var a = 10;  console.log(a++);//10  console.log(a);//11 |
| var a = 10;  console.log(++a);//11  console.log(a);//11 |

总结：后++是先返回再执行+1，前++ 是先执行+1，再返回

|  |
| --- |
| var a = 10;  console.log(a++);//10  console.log(a++);//11 |
| var a = 10;  console.log(++a);//11  console.log(++a);//12 |
| var a = 10;  console.log(++a);//11  console.log(a++);//11 |
| var a = 10;  console.log(a++);//10  console.log(++a);//12 |

|  |
| --- |
| var a = 10;  var b = a++ + a++;  console.log(a);//12  console.log(b);//21 |
| var a = 10;  var b = ++a + ++a;  console.log(a);//12  console.log(b);//23 |
| var a = 10;  var b = ++a + a++;  console.log(a);//12  console.log(b);//22 |
| var a = 10;  var b = ++a + a++ + a++;  console.log(a);//13  console.log(b);//34 |

**1.2比较运算符**

比较运算符比较两个值，然后返回一个布尔值，表示是否满足比较条件。

给定x=5，下面的表格解释了比较运算符：

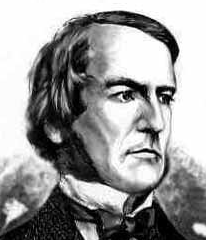


关系运算符的正统，number和number进行数学运算，得到的答案boolean。

|  |
| --- |
| console.log(8 > 5);//true  console.log(7 < 4);//false |

关系运算的结果，是boolean类型的。

true和false叫做布尔值。boolean类型只有两个值，就是true和false。表示真、假。



== 我们叫做“相等判断”，只判断数值。

|  |
| --- |
| console.log(5 == "5"); //true |

===我们叫做“全等判断”，不仅仅比较数值是否相等，还比较类型是否相等

|  |
| --- |
| console.log(5 === "5"); //false |

!= 是==的反面，如果==运算是true，那么!=就是false

!==是===的反面，如果===运算是true，那么!==就是false

|  |
| --- |
| console.log(5 != "5"); //false ， 脑子要反着想一下，5=="5"结果是t，所以就是f  console.log(5 !== "5"); //true ， 脑子要反着想一下，5==="5"结果是f，所以就是t |

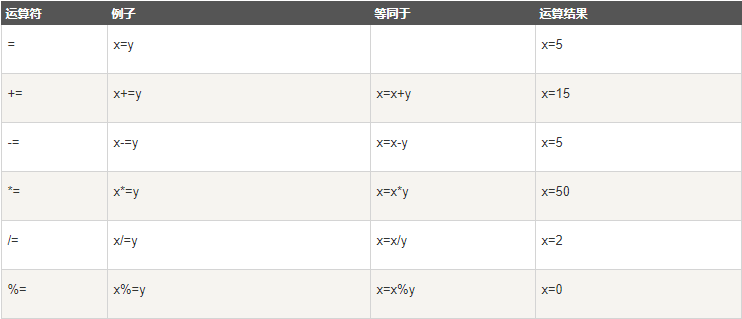
|  |
| --- |
| console.log(undefined == null);//true  console.log(undefined === null);//false |

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

**1.3赋值运算符**

赋值运算符用于给 JavaScript 变量赋值。

给定 **x=10**和**y=5**，下面的表格解释了赋值运算符：



|  |
| --- |
| var a = 1;  a += 2; //这行语句等价于a = a + 2;  console.log(a); //3 |

|  |
| --- |
| var b = 6;  b /= 3; //等价于b = b / 3  console.log(b); //2 |

|  |
| --- |
| var c = 100;  c %= 10; //等价于c = c % 10;  console.log(c); //0 |

**1.4逻辑运算符**

逻辑运算符用于测定变量或值之间的逻辑。

给定 x=6 以及 y=3，下表解释了逻辑运算符：



正统来说，参与逻辑运算的是boolean和boolean，得到的结果也是boolean

值按照真值表来定

&& 逻辑与，“且”

|  |  |  |
| --- | --- | --- |
| a && b | | |
| a | b | 结果 |
| 真 | 真 | 真 |
| 真 | 假 | 假 |
| 假 | 真 | 假 |
| 假 | 假 | 假 |

“都真才真”，“有假就假”。

命题1：“地球是圆的” 真的

命题2：“宋仲基很帅” 真的

命题1 且 命题2 真的

命题1：“1+1=3” 假的

命题2：“地球是方的” 假的

命题1 且 命题2 假的

|  |
| --- |
| //逻辑运算符  console.log(true && true); //t  console.log(true && false); //f  console.log(false && true); //f  console.log(false && false); //f |

|| 逻辑或，“或者”的意思

|  |  |  |
| --- | --- | --- |
| a || b | | |
| a | b | 结果 |
| 真 | 真 | 真 |
| 真 | 假 | 真 |
| 假 | 真 | 真 |
| 假 | 假 | 假 |

“有真就真”，“都假才假”

命题1： 1 + 1 = 2

命题2：“宋仲基很帅”

命题1 或者 命题2 总结果是真

|  |
| --- |
| console.log(true || true); //t  console.log(true || false); //t  console.log(false || true); //t  console.log(false || false); //f |

!就是“逻辑非”，相反的

|  |
| --- |
| console.log(!true); //f  console.log(!false); //t  console.log(!!!!!!!!!false); //t |

运算顺序是非、与、或

1. **分支结构**

通常在写代码时，您总是需要为不同的决定来执行不同的动作。您可以在代码中使用条件语句来完成该任务。

在 JavaScript 中，我们可使用以下条件语句：

**if 语句** - 只有当指定条件为 true 时，使用该语句来执行代码

**if...else 语句** - 当条件为 true 时执行代码，当条件为 false 时执行其他代码

**if...else if....else 语句**- 使用该语句来选择多个代码块之一来执行

**switch 语句** - 使用该语句来选择多个代码块之一来执行

**2.1单分支结构**

只有当指定条件为 true 时，该语句才会执行代码。

|  |
| --- |
| if (condition)  {      当条件为 true 时执行的代码  } |

请使用小写的 if。使用大写字母（IF）会生成 JavaScript 错误！

当时间小于 20:00 时，生成问候 "Good day"：

|  |
| --- |
| if (time<20) {  console.log("Good day");  } |

请注意，在这个语法中，没有 ..else..。您已经告诉浏览器只有在指定条件为 true 时才执行代码。

**2.2双分支结构**

请使用 if....else 语句在条件为 true 时执行代码，在条件为 false 时执行其他代码。

|  |
| --- |
| if (condition){  当条件为 true 时执行的代码  }else  {  当条件不为 true 时执行的代码  } |

当时间小于 20:00 时，生成问候 "Good day"，否则生成问候 "Good evening"。

|  |
| --- |
| if (time<20)  {  console.log("Good day");  }  else  {  console.log("Good evening");  } |

如果成绩大于等于60，输出及格了，否则输出不及格。

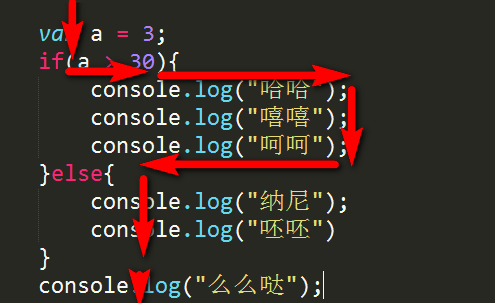
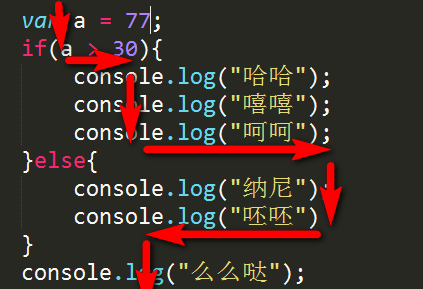
|  |
| --- |
| var a = 80;  if(a >= 60){  alert("及格了");  }else{  alert("不及格");  } |

如果要执行的语句，只有一行语句，那么就是单行if，就可以省略大括号。

|  |
| --- |
| var a = 112;  if(a >= 60)  alert("及格了");  else  alert("不及格"); |

注意if语句是一个结构体，注意哪些语句是在结构体中，哪些语句不是结构体：

|  |
| --- |
| var a = 355;  if(a > 30){  console.log("哈哈");  console.log("嘻嘻");  console.log("呵呵");  }else{  console.log("纳尼");  console.log("呸呸")  }  console.log("么么哒"); /\*一定会执行，在if结构体外面\*/ |



小练习：用户输入一个年龄，判断用户能不能考取驾照。交规说： 大于等于18，小于等于70才能考取驾照。

|  |
| --- |
| //让用户输入年龄  var age = parseInt(prompt("请输入年龄"));  //判断  if(age >= 18 && age <= 70){  alert("可以考取驾照");  }else{  alert("年龄不符合要求");  } |

**2.3多分支结构**

使用 if....else if...else 语句来选择多个代码块之一来执行。

|  |
| --- |
| if (condition1)  {  当条件 1 为 true 时执行的代码  }  else if (condition2)  {  当条件 2 为 true 时执行的代码  }  else  {  当条件 1 和 条件 2 都不为 true 时执行的代码  } |

如果时间小于 10:00，则生成问候 "Good morning"，如果时间大于 10:00 小于 20:00，则生成问候 "Good day"，否则生成问候 "Good evening"：

|  |
| --- |
| if (time<10)  {  document.write("<b>早上好</b>");  }  else if (time>=10 && time<16)  {  document.write("<b>今天好</b>");  }  else  {  document.write("<b>晚上好!</b>");} |

选择一个分支执行，选择这个分支了，就暗含了上一个分支没有满足：

|  |
| --- |
| var score = 76;  if(score >= 85){  alert("优秀"); //不满足条件，所以不执行  }else if(score >= 70){  alert("良好"); //走到这的，一定暗含小于85  }else if(score >= 60){  alert("及格"); //不执行  }else{  alert("不及格"); //不执行  } |

**2.4分支结构嵌套**

用户先输入自己的性别，比如男、女。然后输入自己的年龄。判断能否结婚。

男的22以上

女的20以上

if语句可以嵌套，如果里面再套一层如果：

|  |
| --- |
| //判断  **if(sex == "男"){**  **//男**  **if(age >= 22){**  **alert("男同志你好，可以结婚");**  **}else{**  **alert("你还是个小正太，不能结婚！");**  **}**  **}else{**  **//女**  **if(age >= 20){**  **alert("女同志你好，可以结婚");**  **}else{**  **alert("你还是个小萝莉，不能结婚！");**  **}**  **}** |

|  |
| --- |
| if(sex == "男" && age >= 22 || sex == "女" && age >= 20 ){  console.log("可以结婚");  }else{  console.log("不可以结婚");  } |

楼层中else只能有一个。 else if可以有多个。

1. **循环结构**

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。

如果您希望一遍又一遍地运行相同的代码，并且每次的值都不同，那么使用循环是很方便的。

**3.1for循环**

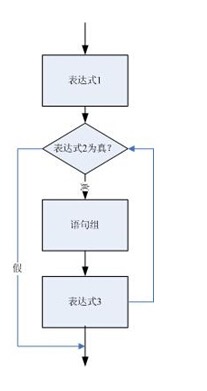
语法：

|  |
| --- |
| **for (语句 1; 语句 2; 语句 3) {     被执行的代码块 }** |

语句1：初始化循环变量

语句2：循环条件

语句3：执行规律



整体感知：

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



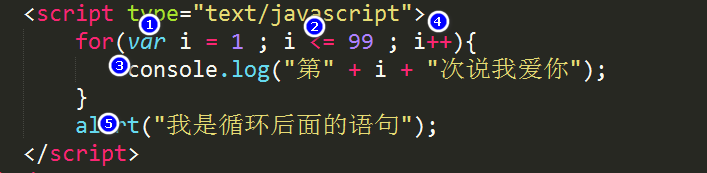
你就发现了，for循环就是一个壳子，让变量i依次、轮流的值为1、2、3、4、……99。

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：

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

for循环的本质，必须搞清楚。

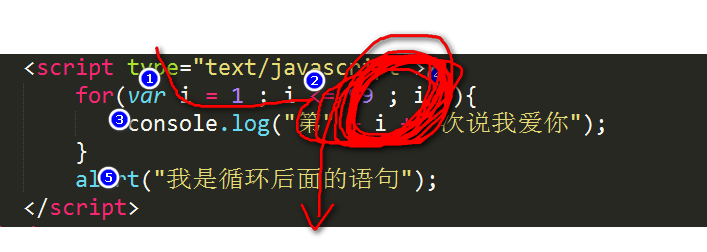


系统遇见了for循环结构，会立即执行语句①，此时声明了一个变量i，赋值为1。

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，……



【实例1】

控制台输出1-100

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

【实例2】

输出1-100的和

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

【实例3】

求1-10的阶乘

|  |
| --- |
| var ji = 1;  for(var i = 1; i <= 10; i++){  ji \*= i;  }  console.log(ji); |

**3.2循环注意事项**

1、for循环表达式1可以不写。如果不写表达式1，则需要在循环结构外为循环变量赋初值。

|  |
| --- |
| var num = 0;  for(;num<10; num++){  console.log(num);  } |

2、for循环表达式2可以不写。如果不写表达式2，则表示循环条件恒成立。(死循环)

|  |
| --- |
| for(var num = 0; ;num++){  console.log(num);  } |

3、for循环表达式3可以不写。如果不写表达式3，则需要在循环结构内部为循环变量增加改变条件

|  |
| --- |
| for(var num = 0; num<10;){  console.log(num);  num++;  } |

4、其实for循环三个表达式都可以不写。但是括号中的分号不能省略！！

|  |
| --- |
| for（; ;）{  console.log(“hello javascript！”);  } |

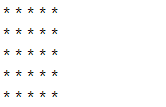
**3.3循环嵌套**

循环和条件语句可以进行嵌套使用来进行更复杂的逻辑处理。

|  |
| --- |
| for (语句 1; 语句 2; 语句 3){  for (语句 1; 语句 2; 语句 3){  被执行的代码块  }  } |

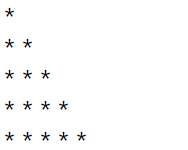
原理：外层循环走一下，内层循环走一圈

【实例1】



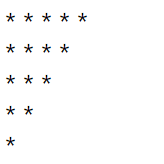
|  |
| --- |
| for(var i = 1; i <= 5; i++){  for(var j = 1; j <= 5; j++){  document.write(" \* ");  }  document.write("<br/>");  } |

【实例2】



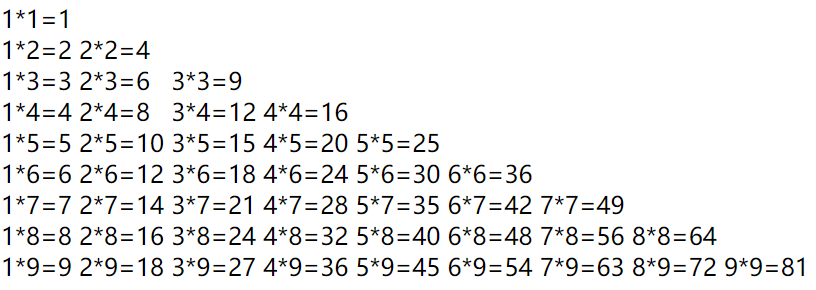
|  |
| --- |
| for(var i = 1; i <= 5; i++){  for(var j = 1; j <= i; j++){  document.write(" \* ");  }  document.write("<br/>");  } |

【实例3】



|  |
| --- |
| for(var i = 1; i <= 5; i++){  for(var j = 1; j <= 6-i; j++){  document.write(" \* ");  }  document.write("<br/>");  } |

【实例4】



|  |
| --- |
| for(var i = 1; i <= 9; i++){  for(var j = 1; j <= i; j++){  var result = i\*j;  if(result < 10 && j >1){  result = result + "&nbsp;&nbsp;"  }  document.write(j+"\*"+i+"="+result+"&nbsp;");  }  document.write("<br/>");  } |

**3.4break和continue**

break语句和continue语句都具有跳转作用，可以让代码不按既有的顺序执行。

break语句用于跳出代码块或循环，循环终止。

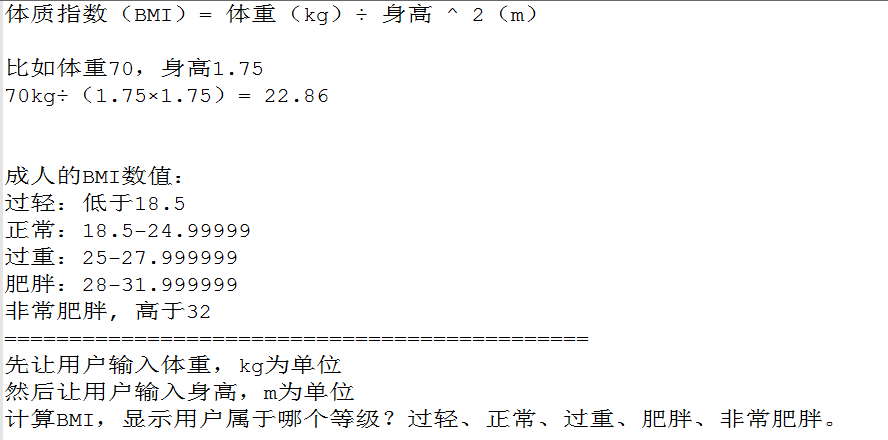
continue语句用于立即终止本轮循环，返回循环结构的头部，开始下一轮循环，循环不终止。

|  |
| --- |
| for(var i = 1; i <= 10; i++){  if(i == 5){  break;  }  console.log(i);  } |

|  |
| --- |
| for(var i = 1; i <= 10; i++){  if(i == 5){  continue;  }  console.log(i);  } |

1. **作业**

1、计算BMI



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。

3、如果a除以2的余数，等于b除以2的余数，那么两个数字的奇偶性相同

4、计算是否是闰年。

5、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