

目录

目录

作业讲解

学习目标

比较运算符

逻辑运算符 重要

三目运算符

运算符的优先级

变量数据类型的转换 了解

强制转换

其他类型转换成字符串类型

其他类型转换成数值类型

其他类型转换成布尔类型

自动转换

流程控制

顺序结构

分支结构

循环结构

分支结构 重点

单分支

双分支

多分支

循环结构 重点

while循环

js调试工具的使用 掌握了有利于理解循环

for循环 重点,必须掌握

今日总结

今日作业

作业讲解

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>Document</title>
6 <script type="text/javascript">
7 // document.write输出的内容会显示在body中,所以可以解析HTML标签
8 // document.write("HTML标签");
9
10 // document.write("<h3 style='color:red;'>我是标题3</h3>");
11
12
13 // 最简单输出表格方法
14 // document.write('<table border="1" rules="all" width="500">');
15 // document.write('<tr>');
16 // document.write('<td colspan="4" align="center">');
17 // document.write('<h3 style="color:red;">张三疯的个人简历</h3>');
18 // document.write('</td>');
19 // document.write('</tr>');
20 // document.write('<tr>');
21 // document.write('<td align="right"><b>姓名:</b></td>');
22 // document.write('<td>张三疯</td>');
23 // document.write('<td align="right"><b>性别:</b></td>');
24 // document.write('<td>男</td>');
25 // document.write('</tr>');
26 // document.write('<tr>');
27 // document.write('<td align="right"><b>年龄:</b></td>');
28 // document.write('<td>34岁</td>');
```

```

29 // document.write('<td align="right"><b>学历:</b></td>');
30 // document.write('<td>大专</td>');
31 // document.write('</tr>');
32 // document.write('<tr>');
33 // document.write('<td align="right"><b>毕业时间:</b></td>');
34 // document.write('<td>2005年7月</td>');
35 // document.write('<td align="right"><b>毕业学校:</b></td>');
36 // document.write('<td>北京科技大学</td>');
37 // document.write('</tr>');
38 // document.write('</table>');
39
40 // 进阶版一
41 // var str = "abc";
42 // str = str + "zhangsan";// str = "abc" + "zhangsan"; str = "abczhangsan"
43 // str = str + "李四";// str = "abczhangsan" + "李四"; str = "abczhangsan李四"
44 // str = str + "王五"// str = "abczhangsan李四" + "王五"; str = "abczhangsan李四王五"
45 // console.log( str );
46
47
48 /*var str = "早上好,";
49 // += 是赋值运算符
50 // 等同于 str = str+"同学们"; str = "早上好,"+"同学们"; str="早上好,同学们"
51 str += "同学们";
52
53 // 等同于 str = str+",今天也要加油鸭~"; str = "早上好,同学们"+" ,今天也要加油鸭~"; str="早上好,同学们,今天也要加油鸭~"
54 str += ",今天也要加油鸭~";
55 console.log( str );*/
56
57
58 /*var str = '<table border="1" rules="all" width="500">';
59 str += '<tr>';
60 str += '<td colspan="4" align="center">';
61 str += '<h3 style="color:red;">张三疯的个人简历</h3>';
62 str += '</td>';
63 str += '</tr>';
64 str += '<tr>';
65 str += '<td align="right"><b>姓名:</b></td>';
66 str += '<td>张三疯</td>';

```

```

67  str += '<td align="right"><b>性别:</b></td>';
68  str += '<td>男</td>';
69  str += '</tr>';
70  str += '<tr>';
71  str += '<td align="right"><b>年龄:</b></td>';
72  str += '<td>34岁</td>';
73  str += '<td align="right"><b>学历:</b></td>';
74  str += '<td>大专</td>';
75  str += '</tr>';
76  str += '<tr>';
77  str += '<td align="right"><b>毕业时间:</b></td>';
78  str += '<td>2005年7月</td>';
79  str += '<td align="right"><b>毕业学校:</b></td>';
80  str += '<td>北京科技大学</td>';
81  str += '</tr>';
82  str += "</table>";
83  // str = '</table>';// 因为=号是赋值运算符,把右边的值赋给左边的变量
84  // console.log( str );
85  // 刷新页面能否看到表格?不能,为什么看不到呢,因为str只是一个变量,这个变量没有被输出
86  // document.write( "str" );// 这样输出正确吗?不正确,因为str被加了引号,就是一个普通字符串,如果想得到变量值,直接使用变量名即可,不需要加引号
87  document.write( str );*/
88
89  // 进阶版二 个人的信息使用变量保存
90  var username = "张三疯";
91  var sex = "男";
92  var age = 34;
93  var xueli = "大专";
94  var biyeshijian = "2005年7月";
95  var school = "北京科技大学";
96
97  var str = '<table border="1" rules="all" width="500">';
98  str += '<tr>';
99  str += '<td colspan="4" align="center">';
100  str += '<h3 style="color:red;">'+username+'的个人简历</h3>';
101  str += '</td>';
102  str += '</tr>';
103  str += '<tr>';
104  str += '<td align="right"><b>姓名:</b></td>';
105  str += '<td>'+username+'</td>';

```

```

106 str += '<td align="right"><b>性别:</b></td>';
107 str += '<td>'+sex+'</td>';
108 str += '</tr>';
109 str += '<tr>';
110 str += '<td align="right"><b>年龄:</b></td>';
111 str += '<td>'+age+'岁</td>';
112 str += '<td align="right"><b>学历:</b></td>';
113 str += '<td>'+xueli+'</td>';
114 str += '</tr>';
115 str += '<tr>';
116 str += '<td align="right"><b>毕业时间:</b></td>';
117 str += '<td>'+biyeshijian+'</td>';
118 str += '<td align="right"><b>毕业学校:</b></td>';
119 str += '<td>'+school+'</td>';
120 str += '</tr>';
121 str += "</table>";
122 document.write( str );
123 </script>
124 </head>
125 <body>
126 <!-- HTML的表格标签在HTML第二天学的 -->
127 <!-- table标签代表最大的表格,tr代表行,td,th代表列 -->
128 <!-- colspan是左右合并单元格 -->
129 <!-- <table border="1" rules="all" width="500">
130 <tr>
131 <td colspan="4" align="center">
132 <h3 style="color:red;">张三疯的个人简历</h3>
133 </td>
134 </tr>
135 <tr>
136 <td align="right"><b>姓名:</b></td>
137 <td>张三疯</td>
138 <td align="right"><b>性别:</b></td>
139 <td>男</td>
140 </tr>
141 <tr>
142 <td align="right"><b>年龄:</b></td>
143 <td>34岁</td>
144 <td align="right"><b>学历:</b></td>
145 <td>大专</td>

```

```

146 </tr>
147 <tr>
148 <td align="right"><b>毕业时间:</b></td>
149 <td>2005年7月</td>
150 <td align="right"><b>毕业学校:</b></td>
151 <td>北京科技大学</td>
152 </tr>
153 </table> -->
154 </body>
155 </html>

```

张三疯的个人简历			
姓名:	张三疯	性别:	男
年龄:	34岁	学历:	大专
毕业时间:	2005年7月	毕业学校:	北京科技大学

学习目标

- 能够掌握使用比较运算符
- 能够掌握使用逻辑运算符
- 能够掌握使用三目运算符
- 能够了解运算符的优先级
- 能够了解变量数据类型的转换
- 掌握并应用分支结构if语句
- 能够掌握理解和熟练应用while循环
- 能够理解和熟练应用do,while循环

- 能够掌握和熟练应用for循环

学习目标

算术,赋值运算符,以及自操作运算符等等

- 能够掌握使用比较运算符
- 能够掌握使用逻辑运算符
- 能够掌握使用三目运算符
- 能够了解运算符的优先级
- 能够了解变量数据类型的转换
- 掌握并应用分支结构if语句
- 掌握switch语句
- 能够掌握理解和熟练应用while循环
- 能够理解和熟练应用do,while循环
- 能够掌握和熟练应用for循环

5*2+3 = 25?
5*2+3 = 13?

手动的强制转换
js帮我们自动转换
两种转换的结果是一样

流程控制
顺序结构:代码从上到下,按顺序执行
分支结构:代码有选择的执行
循环结构:满足一定条件以后,不断重复执行相同的代码

比较运算符

比较运算符执行的是比较运算。每个比较运算符都返回一个布尔值。结果为：
true或者false

假设：y=5 x=10，下面的表格解释了比较运算符的结果：

运算符	描述	例子	结果
>	大于	x>y	true
<	小于	x<y	false
>=	大于等于	x>=y	true
<=	小于等于	x<=y	false
==	等于 只需要比较值是否相等	x==y	false
!=	不等于	x!=y	true
===	全等于 (要比较数据类型相等和值相等)	x===y	false
!==	不全等于 (判断值跟类型是否都不相等,值或者类型有一个不相等,才可以得到true)	x!==y	true

注意：比较运算符它最终得到的结果是布尔值：**true和false**

举例：

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  <meta charset="UTF-8">
5  <title>Document</title>
6  <script type="text/javascript">
7  // 比较运算符有 < > <= >= == === != !==
8
9  // 比较运算符执行的是比较运算。每个比较运算符都返回一个布尔值,也就是结果只有两种true或者false,正确就是true,错误就是false
10
11  var x = 5;
12  var y = 10;
13  console.log( x < y );// true
14  console.log( x > y );// false
15  console.log( x <= y );// true
16  console.log( x >= y );// false
17
18  // 注意:
19  // 在js语言中, = 是赋值符号
20  // 在js语言中, == 才是比较两个值是否相同,如果值相等,得到true,不相等得到false,不管数据类型的
21  console.log( x == y );// false
22  console.log("");
23
24  // === 是全等的意思,需要变量的值跟变量的类型都相同才能得到true,其他情况得到的都是false
25  var a = 10;
26  var b = 10;
27  var c = "10";
28  console.log( a == b );// true
29  console.log( a === b );// true
30  console.log( a == c );// true
31  console.log( a === c );// false
32  console.log("");
33
34  // != 判断值是否不相等,如果不相等,得到true;如果相等得到false
35  console.log( a != b );// false
36  console.log( a != c );// false
```



```

37
38 // !== 判断值跟类型是否都不相等,值或者类型有一个不相等,就可以得到true
39 console.log( a !== b );// false
40 console.log( a !== c );// a跟c值相等,但是类型不相等,得到true
41 console.log("");
42
43 var i = 5;
44 var j = 10;
45 console.log( i !== j );// true
46 console.log( i !== j );// true
47
48 </script>
49 </head>
50 <body>
51
52 </body>
53 </html>

```

逻辑运算符 重要

在形式逻辑中，逻辑运算符或逻辑联结词把语句连接成更复杂的复杂语句。例如，假设有两个逻辑命题，分别是“正在下雨”和“我在屋里”，我们可以将它们组成复杂命题“正在下雨，并且我在屋里”或“没有正在下雨”或“如果正在下雨，那么我在屋里”。一个将两个语句组成的新的语句或命题叫做复合语句或复合命题。逻辑运算符用来表示日常交流中的“并且”，“或者”，“除非”等思想。

符号	说明	功能
&&	逻辑与 并且	&&符号两边的表达式的结果同时为真才为真
	逻辑或 或者	符号两边的表达式的结果只要有一边为真就为真
!	逻辑非 取反	取反操作 将true变为false 将false变为true

小结:

- 1 && 两个操作数同时为true，结果为true，否则都是false
- 2 || 两个操作数有一个为true，结果为true，否则为false
- 3 ! 取反

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <script type="text/javascript">
7     // 逻辑运算符
8     // && 逻辑与,表示"并且"的关系 &&符号两边的表达式的结果同时为真才为真
9     console.log( true && true );// true
10    console.log( true && false );// false
11    console.log( false && false );// false
12    console.log("");
13
14    // || 逻辑或,表示"或者"的关系 ||符号两边的表达式的结果只要有一边为真就为真
15    console.log( true || true );// true
16    console.log( true || false );// true
17    console.log( false || false );// false
18    console.log("");
19
20    // ! 逻辑非,也可以叫取反 取反操作 将true变为false 将false变为true
21    console.log( !true );// false
22    console.log( !false );// true
23    console.log( false );// false
24    console.log("");
25
26    console.log( 3>2 && 10<3 );// true && false => false
27    console.log( 3>2 || 10<3 );// true || false => true
28    console.log("");
29
30    console.log( !!false );// !true => false
31    console.log( !!!false );// !!true => !false => true
32  </script>
33 </head>
34 <body>
35
36 </body>
37 </html>
```

三目运算符

语法格式：

条件语句 ? 语句1 : 语句2;

说明：

问号前面的位置是判断的条件，判断结果为boolean型，为true时执行语句1，为false时执行语句2。

其逻辑为："如果为真执行第一个，否则执行第二个。"

举例：

```
4      <meta charset="UTF-8">
5      <title>Document</title>
6      <script type="text/javascript">
7          // 三目运算符
8          // 语法如下
9          // 条件表达式 ? js语句1 : js语句2;
10         // 结构说明:如果条件表达式的结果为true,
           执行js语句1;如果条件表达式的结果为false,执行js语句2
11
12         // 补充知识点:什么是条件表达式?条件表达式就是可以得到布尔值的式子,
           比如2>3,比如 10==5
13
14
15         console.log( 2>3 ? 2 : 3); // 3
16         console.log( true ? "成立" : "不成立"); // "成立"
17     </script>
18 </head>
19 <body>
20
```

运算符的优先级

运算符	描述
[] 0	字段访问、数组下标、函数调用以及表达式分组
++ -- ~ ! delete new typeof void	一元运算符、返回数据类型、对象创建、未定义值
* / %	乘法、除法、取模
+ - +	加法、减法、字符串连接
<< >> >>>	移位
< <= > >= instanceof	小于、小于等于、大于、大于等于、instanceof
== != === !==	等于、不等于、严格相等、非严格相等
&	按位与
^	按位异或
	按位或
&&	逻辑与
	逻辑或
?:	条件
= o/=	赋值、运算赋值
,	多重求值

- 1 优先级从高到底
- 2 1. () 优先级最高
- 3 2. 一元运算符 ++ -- !
- 4 3. 算数运算符 先* / % 后 + -
- 5 4. 关系运算符 > >= < <=
- 6 5. 相等运算符 == != === !==
- 7 6. 逻辑运算符 先&& 后||

- 1 为什么会有运算符的优先级这个问题？
- 2 在一个式子里面出现了多种运算符 需要考虑哪一个运算符优先运行。

小学数学学过：有小括号先算小括号里面的 然后再乘除后加减 如果需要提升某一个运算符的优先级 请加小括号

```
1 // 练习1:
2 console.log ( 4 >= 6 || '人' != '阿凡达' && !(12 * 2 == 144) && true
);
3 // 练习2:
4 var num = 10;
5 console.log( 5 == num / 2 && (2 + 2 * num) === '22' );
```

举例:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>Document</title>
6 <script type="text/javascript">
7 // 运算符的优先级就是多个运算符同时存在的时候,先执行哪个运算符,后执行哪个运算符,这些运算符执行有个顺序,这个顺序我们就叫做运算符的优先级
8 console.log( 5+2*3 );// 5+6 = 11
9
10 var y = 20;
11 // 因为赋值运算是最后执行的
12 console.log( y = y+10 ); // y = 20+10; y =30;
13 console.log("");
14
15 // 我们可以给运算符添加(),提高优先级
16 console.log( (5+2)*3 );
17 console.log("");
18
19 /*
20 优先级从高到底
21 1. () 优先级最高
22 2. 一元运算符(自操作) ++ -- !
```

23 3. 算数运算符 先* / % 后 + -

24 4. 关系运算符 > >= < <=

25 5. 相等运算符 == != === !==

26 6. 逻辑运算符 先&& 后||

27 7. 赋值运算符

28

29 也不要要求大家记忆,想提高优先级,就加(),因为()的优先级是最高的

30 赋值运算符的优先级是最低

31 */

32

33 // 如果优先级相同,那么就从左到右运算

34 console.log(10 + 2 - 5);// 7

35 console.log(10 - 2 + 5);// 3

36 console.log("");

37

38

39

40 // 练习1:

41 console.log (4 >= 6 || '人' != '阿凡达' && !(12 * 2 == 144) && true);// true

42

43

44 // 练习2:

45 var num = 10;

46 console.log(5 == num / 2 && (2 + 2 * num) === '22');// false

47 console.log("");

48

49 // 这题比较难 迎难而上 难上加难

50 // var x = 10;

51 // console.log(x + ++x + x++);// 32

52

53

54 // 第一步:先看以下代码

55 // var x = 10;

56 // console.log(x + ++x);// 21

57 // 分析

58 // console.log(10 + ++10);

59 // console.log(10 + 11);

60 // console.log(21);

61

62

```

63
64 // var y = 20;
65 // console.log( ++y + y); // 42
66 // 分析
67 // console.log( ++20 + 21);
68 // console.log( 21 + 21);
69 // console.log( 42 );
70
71
72 var x = 10;
73 console.log( x + ++x + x++); // 32
74 // 分析
75 // console.log( 10 + ++10 + x++);
76 // console.log( 10 + 11 + 11++);
77 // console.log( 10 + 11 + 11);
78 // console.log( 21 + 11);
79 // console.log( 32);
80 </script>
81 </head>
82 <body>
83
84 </body>
85 </html>

```

变量数据类型的转换 了解

什么是数据类型的转换:

就是将一种数据类型转换为其它的数据类型!

在JS中变量数据类型的转换分成两种:**自动转换**、**强制转换**!

自动转换是JS代码根据上下文环境将A类型转换成B类型,是系统自己完成的! 我们看不到系统转换的过程!

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">

```

```

5 <title>Document</title>
6 <script type="text/javascript">
7   var v1 = 1;
8   //条件语句 本省要得到一个true或者false 但是我们现在给的是number类型的数据
9   //JS它就会自动根据代码的需求,将数据类型转成了boolean类型
10  var res = v1 ? "好的" : "不好";
11  document.write(res);
12 </script>
13 </head>
14 <body>
15
16 </body>
17 </html>

```

强制转换是程序员通过JavaScript中的几个系统函数来完成的。

注意: 自动转换与强制转换的结果是一样的!

强制转换

其他类型转换成字符串类型

- 1 变量名.`toString()`
- 2 格式: 变量名.`toString()` 将其它的类型转换为string类型

- 1 `String`(要转换的变量名)
- 2 `String()`函数存在的意义:有些值没有`toString()`,这个时候可以使用`String()`。比如:`undefined`和`null`

- 1 拼接字符串方式
- 2 当 `+` 两边一个操作符是字符串类型,一个操作符是其它类型的时候,会先把其它类型转换成字符串再进行字符串拼接,返回字符串

- 1 还可以在其它的数据类型的外面包裹一个引号

以上4种方法,用的最多的是`toString`

举例:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <script type="text/javascript">
7     // 其他类型转换成字符串类型
8     // 方法一: 变量名.toString()
9     var a = 10;
10    console.log( a );// 10
11    console.log( typeof(a) );// number
12    console.log("");
13
14    // 转过以后别忘记赋值
15    a = a.toString();
16    console.log( a );// "10"
17    console.log( typeof(a) );// string
18    console.log("");
19
20    var b = true;
21    console.log( typeof(b) );// boolean
22    b = b.toString();
23    console.log( typeof(b) );// string
24    console.log("");
25
26
27    // 有些数据类型是没有toString()方法的
28    /*var c = undefined;
29    console.log( typeof(c) );
30    c = c.toString();
31
32    var d = null;
33    console.log( typeof(d) );
34    d = d.toString();*/
35
36    // 方法二: String(要转换的变量名)
37    // String()函数存在的意义:有些值没有toString(), 这个时候可以使用String()。比
    如:undefined和null
38
```



```

39  var c = undefined;
40  console.log( typeof(c) );// undefined
41  c = String(c);
42  console.log( typeof(c) );// string
43  console.log("");
44
45
46  var d = null;
47  console.log( typeof(d) );// object
48  d = String(d);
49  console.log( typeof(d) );// string
50  console.log( "" );
51
52
53
54  // 方法三：拼接字符串方式 +号两边的运算数有一个是字符串类型,+号就会变成字符串连接符
55  var e = "123" + 1;
56  console.log( e );// 1231
57  console.log( typeof(e) );// string
58  console.log( "" );
59
60  // 方法四：还可以在其它的数据类型的外面包裹一个引号
61  var f = 10;
62  console.log( typeof(f) );// number
63
64  var g = "10";
65  console.log( typeof(g) );// string
66
67  </script>
68  </head>
69  <body>
70
71  </body>
72  </html>

```

其他类型转换成数值类型

```

1  Number(要转换的变量)

```

- 2 `Number()` 可以把任意类型转换成数值型，如果要转换的字符串中有一个不是数值的字符，会返回NaN
- 3
- 4 使用格式：
- 5 `Number`(要转换的变量)

作用:将其它的数据类型强制的转换为数值型

在数值型数据中有一个特殊的数据叫NaN(Not a Number)这不是一个数，为什么会出现NaN呢？当将其它的变量的数据类型转换为Number的时候 如果不能直接的转换过来 就会变成NaN

死记:

数据格式	转换后的结果
"数字"	数字
"数字字符"	NaN
"字符数字"	NaN
"字符"	NaN
" " 和 "空字符串和有空格的字符串"	0
true	1
false	0

小结:只要是在字符串中含有非数字的都会转换为NaN;

```
// 其他类型转成数值类型,有以下方法
// 第一种: Number(变量名)可以把任意类型转换成数值型, 如果要转换的字符串中有一个不是数值的字符, 会返回NaN

// NaN => Not a number 不是一个数字

var v1 = "123";
var v2 = "456abc";
var v3 = "abc789";
var v4 = "abc";
var v5 = "";
var v6 = " ";
var v7 = true;
var v8 = false;

// 使用 Number(变量名)将其他类型转成数值型
// 结论:只要是在字符串中含有非数字的都会转换为NaN;
console.log( Number(v1) );// 123
console.log( Number(v2) );// NaN
console.log( Number(v3) );// NaN
console.log( Number(v4) );// NaN
console.log( Number(v5) );// 0
console.log( Number(v6) );// 0
console.log( Number(v7) );// 1
console.log( Number(v8) );// 0
```

parseInt(要转换的变量)

```
1 //从其它类型中提取整数
2 // parse 解析的意思
3 // Int 整数的意思
4 // parseInt() 从字符串中提取整数 将字符串转换为number类型
5 var v1 = "12.34abcd";
6
7 // 结论:
8 // 如果该字符串的第一个字符不是数字 就会得到 NaN
9 // 如果该字符串的第一个字符是数字 接下来从它里面将整数提取出来 遇到了非数字的以后就停止提取
10
11 var res = parseInt(v1);
12 document.write(res);
```

```
// 第二种方法: parseInt(要转换的变量)
//从其它类型中提取整数
// parse 解析的意思
// Int 整数的意思
// parseInt() 从字符串中提取整数 将字符串转换为number类型

// parseInt()转换规则如下:
// 如果该字符串的第一个字符不是数字 就会得到 NaN
// 如果该字符串的第一个字符是数字 接下来从它里面将整数提取出来 遇到了非数字的以后就停止提取

var str1 = "123abc";
var str2 = "abc456";
var str3 = "789abc123";
var str4 = "123.456";
console.log( parseInt(str1) );// 123
console.log( parseInt(str2) );// NaN
console.log( parseInt(str3) );// 789
console.log( parseInt(str4) );// 123
console.log("");
```

parseFloat(要转换的变量)

```
1 // parseFloat() 从字符串中提取小数 将字符串转换为number类型
```

```
2 // parseFloat() 从一个字符串提取小数 如果第一个字符不是数字会得到NaN 如果是会进行提取
3 // 提取规则 如果遇到了除了第一个.以外的非数字 就会停止提取
4 var v1 = "12.3.4abcd";
5 var res = parseFloat(v1);
6 document.write(res);
```

```
// 第三种方法: parseFloat(要转换的变量)
// parseFloat() 从字符串中提取小数 将字符串转换为number类型
// parseFloat() 从一个字符串提取小数 如果第一个字符不是数字会得到NaN 如果是会进行提取
// 提取规则 如果遇到了除了第一个.以外的非数字 就会停止提取
var str5 = "123.456.789";
var str6 = "abc123.45";
var str7 = "123.45abc789.2";
console.log( parseFloat(str5) );// 123.456
console.log( parseFloat(str6) );// NaN
console.log( parseFloat(str7) );// 123.45
```

在JavaScript中，如果想让某一个类型的数据自动转化为数字类型，可以结合加号+、减号-、或者-0操作。

```
1 var str = '500';
2 console.log(+str); // 取正
3 console.log( typeof(+str) );
4
5 console.log(-str); // 取负
6 console.log( typeof(-str) );
7
8 console.log(str - 0); // 减0
9 console.log( typeof(str-0) );
```

举例:

```
// 第四种方法:在JavaScript中,如果想让某一个类型的数据自动转化为数字类型,可以结合加号+、减号-、或者-0操作。
var str8 = "10";
console.log( str8 );// "10"
console.log( typeof( str8 ) );// string

console.log( +str8 );// 10
console.log( typeof( +str8 ) );// number
console.log("");

var str9 = "45";
console.log( str9 );// "45"
console.log( typeof( str9 ) );// string

console.log( -str9 );// -45
console.log( typeof( -str9 ) );// number
console.log("");

var str10 = "78";
console.log( str10 );// "78"
console.log( typeof( str10 ) );// string

console.log( str10-0 );// 78
console.log( typeof( str10-0 ) );// number

// 注意: +0容易出问题
console.log( str10+0 );// "780"
console.log( typeof( str10+0 ) );// string
```

其他类型转换成布尔类型

Boolean(要转换的变量) 可以实现将其它的类型转换为布尔类型

数据格式	转换后的结果
null和undefined	false
0和" " 和NaN	false
"0"	true
1	true
" " 有空格的字符串	true
"有具体内容的"	true

总结:

- 将number类型转换为boolean类型 只有0与NaN是false 其它的都是true
- null与undefined 转换为boolean类型 是false
- 将String转换为boolean类型 只有空字符串是false 其它的字符串都是true 只要是这个字符串里面有内容 就是true

举例:

```

<script type="text/javascript">
  // 其他类型转换成布尔类型
  // Boolean(要转换的变量)    可以实现将其它的类型转换为布尔类型

  // Boolean(要转换的变量)转换规律如下
  // 将number类型转换为boolean类型 只有0与NaN是false 其它的都是true
  // null与undefined 转换为boolean类型 是false
  // 将String转换为boolean类型 只有空字符串是false 其它的字符串都是true
  只要是这个字符串里面有内容 就是true

  var v1 = null;
  var v2 = undefined;
  var v3 = 0;
  var v4 = "";
  var v5 = NaN;
  var v6 = "0"; // 字符串0
  var v7 = 1; // 非0数字
  var v8 = " "; // 有空格字符串
  var v9 = "123abc"; // 有具体内容的字符串
  console.log( Boolean(v1) ); // false
  console.log( Boolean(v2) ); // false
  console.log( Boolean(v3) ); // false
  console.log( Boolean(v4) ); // false
  console.log( Boolean(v5) ); // false
  console.log( Boolean(v6) ); // true
  console.log( Boolean(v7) ); // true
  console.log( Boolean(v8) ); // true
  console.log( Boolean(v9) ); // true
</script>

```

自动转换

自动转换我们主要指的是**布尔类型的隐式转换**

流程控制语句会把后面的值隐式转换成布尔类型

- 1 转换为true : 非空字符串 非0数字 true 任何对象
- 2 转换成false : 空字符串 0 null undefined

举例:

// 结果是什么?

var a = !!'123';

```

var a = !!'123';
console.log( a ); // true    因为"123"是非空字符串,所以会被转换为true

```

练习:

- 1 // 结果是什么?
- 2 !!10<4
- 3
- 4 // 结果是什么?
- 5 !(10<4)

```
// console.log( !!10<4 );
// 分析 !跟<两个运算符, !优先级高?
// console.log( !!10<4); // !取反,真变假,假变真,
// 10这个时候,10需要先被自动转换为布尔类型的true

console.log( !10 ); // false
console.log( !!10 ); // true
console.log( true<4 ); // true, 因为<是比较运算符,
// 所以比较的结果一定是布尔类型的,什么东西才能比较大小?? 数值才可以比较大小
// ,所以 true会被转成数值1
console.log( 1<4 ); // true
console.log("");

console.log( !!10<4 ); // true
console.log( !!10<0 ); // false
console.log("")

console.log( !(10<4) ); // false
```

流程控制

什么是流程控制?

流程控制

控制流程（也称为流程控制）是计算机运算领域的用语，意指在程序运行时，个别的指令（或是陈述、[子程序](#)）运行或[求值](#)的顺序。不论是在声明式编程语言或是函数编程语言中，都有类似的概念。

流程控制分为三种结构:顺序结构、分支结构、循环结构

顺序结构

从上到下执行的代码就是顺序结构

程序默认就是由上到下顺序执行的

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <script type="text/javascript">
7     // 从上到下执行的代码就是顺序结构,程序默认就是由上到下顺序执行的
8     document.write("aaaa");
9     document.write("bbbbbb");
10  </script>
11 </head>
12 <body>
13
14 </body>
15 </html>

```



The screenshot shows a web browser window titled 'Document'. The address bar shows the file path 'F:/GZ20H5直播6班/Javascript基础/02-Java...'. The page content displays 'aaaabbbbbb', which is the result of the sequential execution of the JavaScript code in the provided HTML snippet.

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <script type="text/javascript">
7     // 从上到下执行的代码就是顺序结构,程序默认就是由上到下顺序执行的
8     document.write("bbbbbb");
9     document.write("aaaa");
10  </script>
11 </head>
12 <body>
13
14 </body>
15 </html>

```



The screenshot shows a web browser window titled 'Document'. The address bar shows the file path 'F:/GZ20H5直播6班/Javascript基础/02-Java...'. The page content displays 'bbbbbaaaa', which is the result of the sequential execution of the JavaScript code in the provided HTML snippet, where 'bbbbbb' is written first, followed by 'aaaa'.

分支结构

根据不同的情况，执行对应代码

虽然有多种选择，但是最终只会选择一个。

在javascript中,分支结构语句有两种 **if语句和switch语句**

if语句分为三种分支，单分支,双分支,多分支

循环结构

循环结构:满足一定的条件，重复做一件事情

在javascript中，循环语句有三种，**while、do..while、for循环。**

分支结构 重点

单分支

语法:

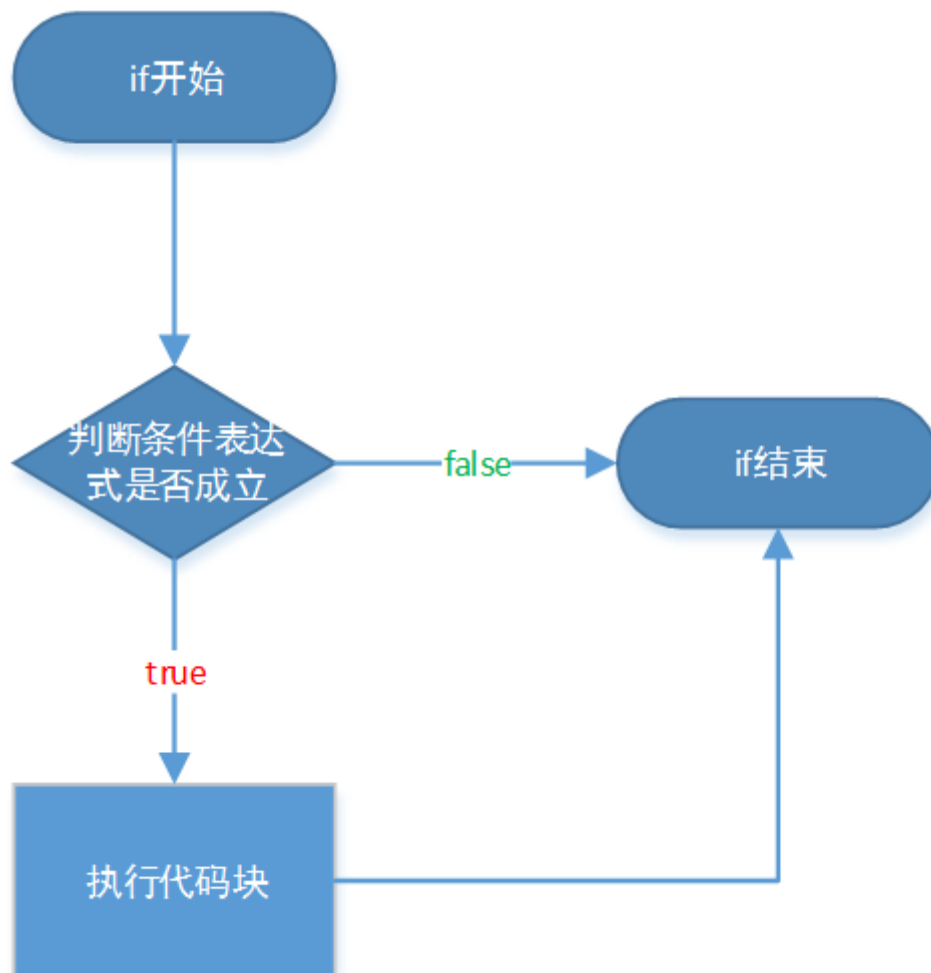
```
1 if(条件语句){  
2   要执行的代码块  
3 }
```

结构说明:

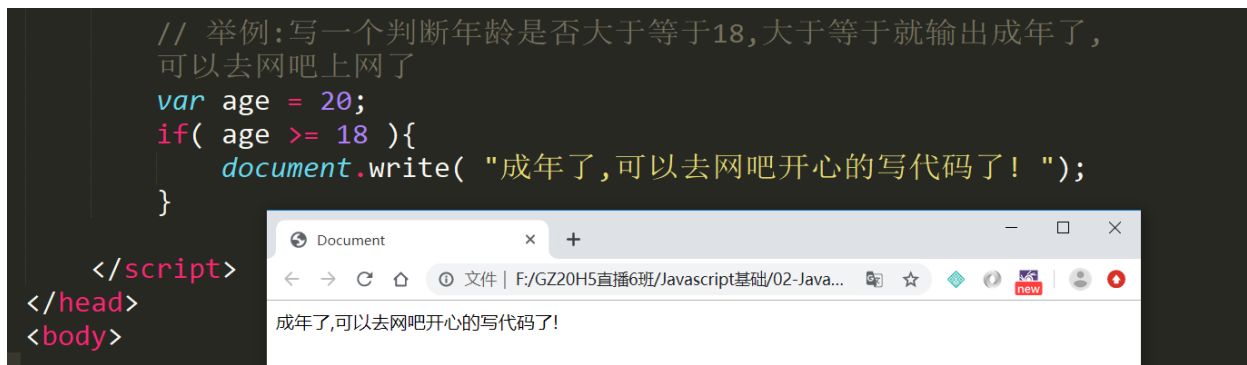
当条件语句成立时:才会执行代码块。

条件表达式成立指的是:得到布尔类型的值为true 就表示成立。

流程图:



举例:写一个判断年龄是否大于等于18,大于等于就输出成年了,可以去网吧上网了



双分支

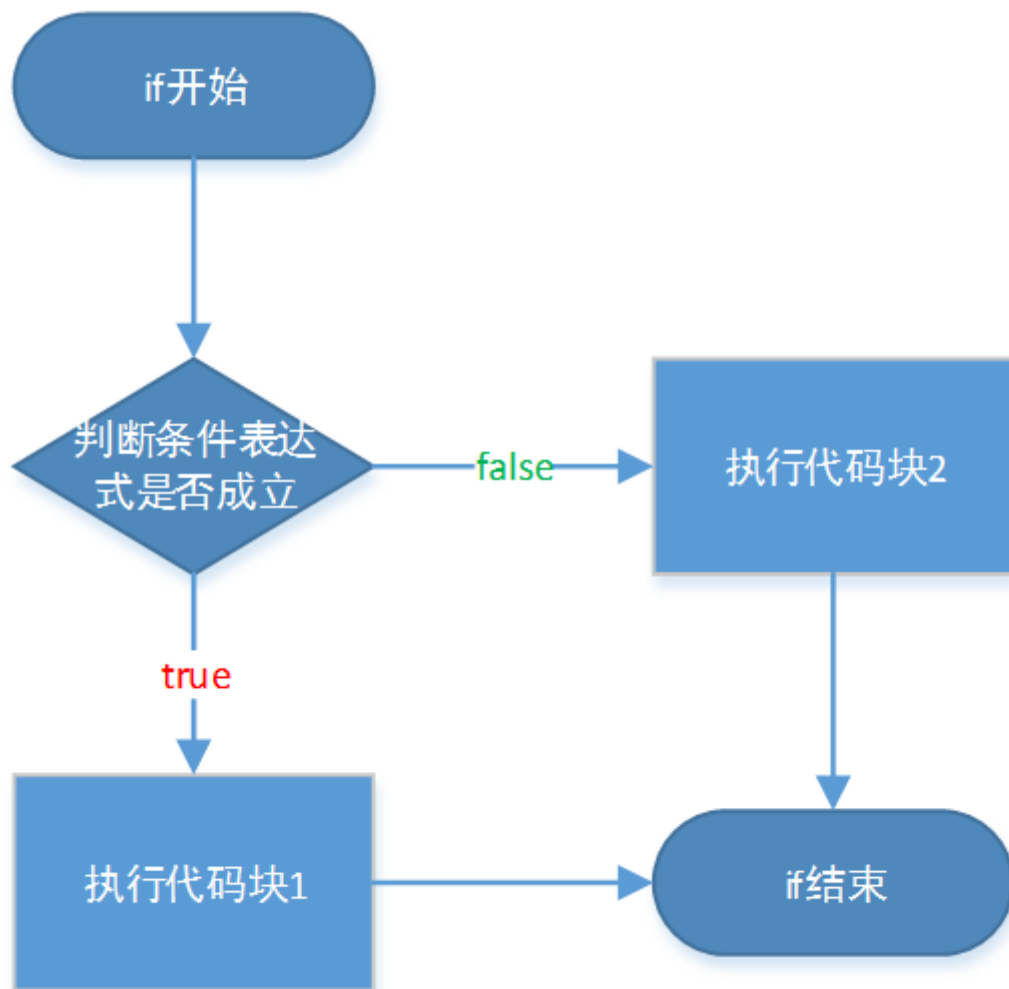
语法:

```
1 if(条件表达式){
2   执行代码块1
3 }else{
4   执行代码块2
5 }
```

结构说明:

如果条件表达式成立的话就执行代码块1, 反之如果不成立就执行代码块2

流程图:



举例: 如果分数大于或者等于60分 就输出及格 否则 分数 小于60 就输出 不及格

```
22     var score = 70;
23     if(score >= 60){
24         document.write("及格");
25     }else{
26         document.write("不及格");
27     }
28 </script>
29 </head>
30 <body>
31
32 </body>
33 </html>
```

The screenshot shows a web browser window with the title 'Document'. The address bar displays the file path 'F:/GZ20H5直播6班/Javascript基础/02-Java...'. The browser content area shows the text '及格' (Pass), which is the output of the JavaScript code provided in the adjacent code block.

```
22     var score = 55;
23     if(score >= 60){
24         document.write("及格");
25     }else{
26         document.write("不及格");
27     }
28 </script>
29 </head>
30 <body>
31
32 </body>
33 </html>
```

练习: 判断一个数是偶数还是奇数

```
4 <meta charset="UTF-8">
5 <title>Document</title>
6 <script type="text/javascript">
7     // 练习：判断一个数是偶数还是奇数
8     // 如何判断偶数还是奇数 有的同学用/
9     // 有的同学用%
10
11     // %是求余数 一个数%2的结果就两种，
12     // 一个是0一个1,如果能被2整除,%2的结果是0
13     // 如果不能被2整除,%2的结果是1
14     var num = 2;
15     if(num%2 == 0){
16         document.write(num+"是偶数");
17     }else{
18         document.write(num+"是奇数");
19     }
20 </script>
21 </head>
22 <body>
```

多分支

多分支也称之为多条件判断

语法:

```
1 if(条件表达式1){
2     代码块1
3 }else if(条件表达式2){
4     代码块2
5 }else if(条件表达式3){
6     代码块3
7 }else if(条件表达式n){
8     代码块n
9 }else{
```

```
10  默认代码块
```

```
11  }
```

结构说明:

第一步:先去判断条件表达式1是否成立 如果成立就执行代码块1 然后就结束if语句 如果不成立

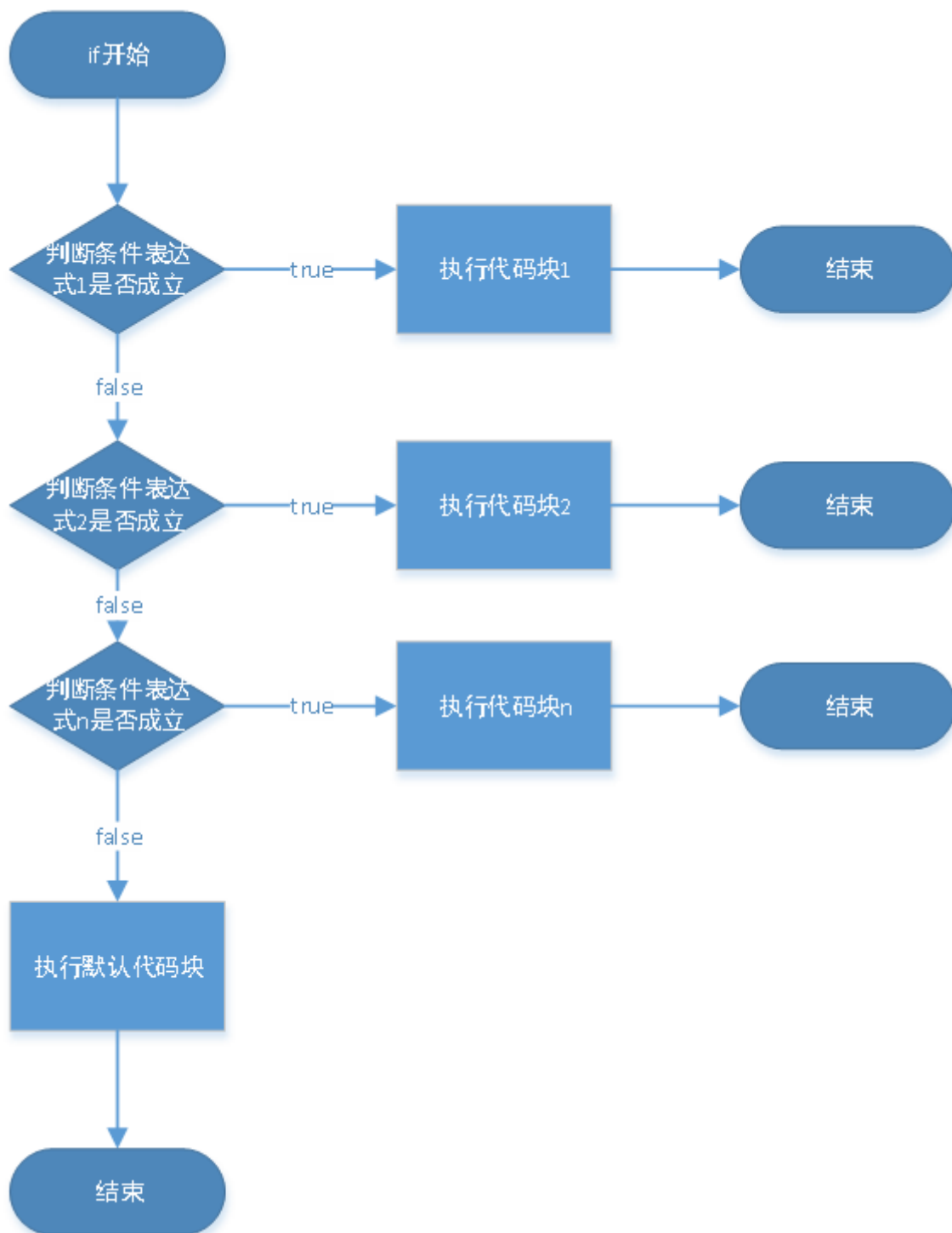
第二步:判断条件表达式2是否成立 如果成立就执行代码块2 然后就结束if语句 如果不成立

第三步:判断条件表达式3是否成立 如果成立就执行代码块3 然后就结束if语句 如果不成立

第四步:判断条件表达式n是否成立 如果成立就会执行代码块n 然后就结束if语句 如果不成立

第五步:上面的所有的条件表达式都不成立 就会执行else语句里面的代码块(这里的else语句可以省略不写,不写就代表没有默认代码块)

流程图:



举例:

如果分数大于或者等于90 优秀
如果分数大于或者等于80 并且 小于90 良好
如果分数大于或者等于70 并且 小于80 一般
如果分数大于或者等于60 并且 小于70 及格
如果分数小于 60 渣渣辉

```
// 举例：
// 如果分数大于或者等于90 优秀
// 如果分数大于或者等于80 并且 小于90 良好
// 如果分数大于或者等于70 并且 小于80 一般
// 如果分数大于或者等于60 并且 小于70 及格
// 如果分数小于 60 渣渣辉

var score = 75;
if( score>=90 ){
    document.write("优秀");
}else if( score>=80 ){
    document.write("良好");
}else if( score>=70 ){
    document.write("一般");
}else if( score>=60 ){
    document.write("及格");
}else{
    document.write("渣渣辉");
}
```

循环结构 重点

在满足一定的条件下 重复执行某些代码！

在javascript中，循环语句有三种，while、do..while、for循环。

while循环

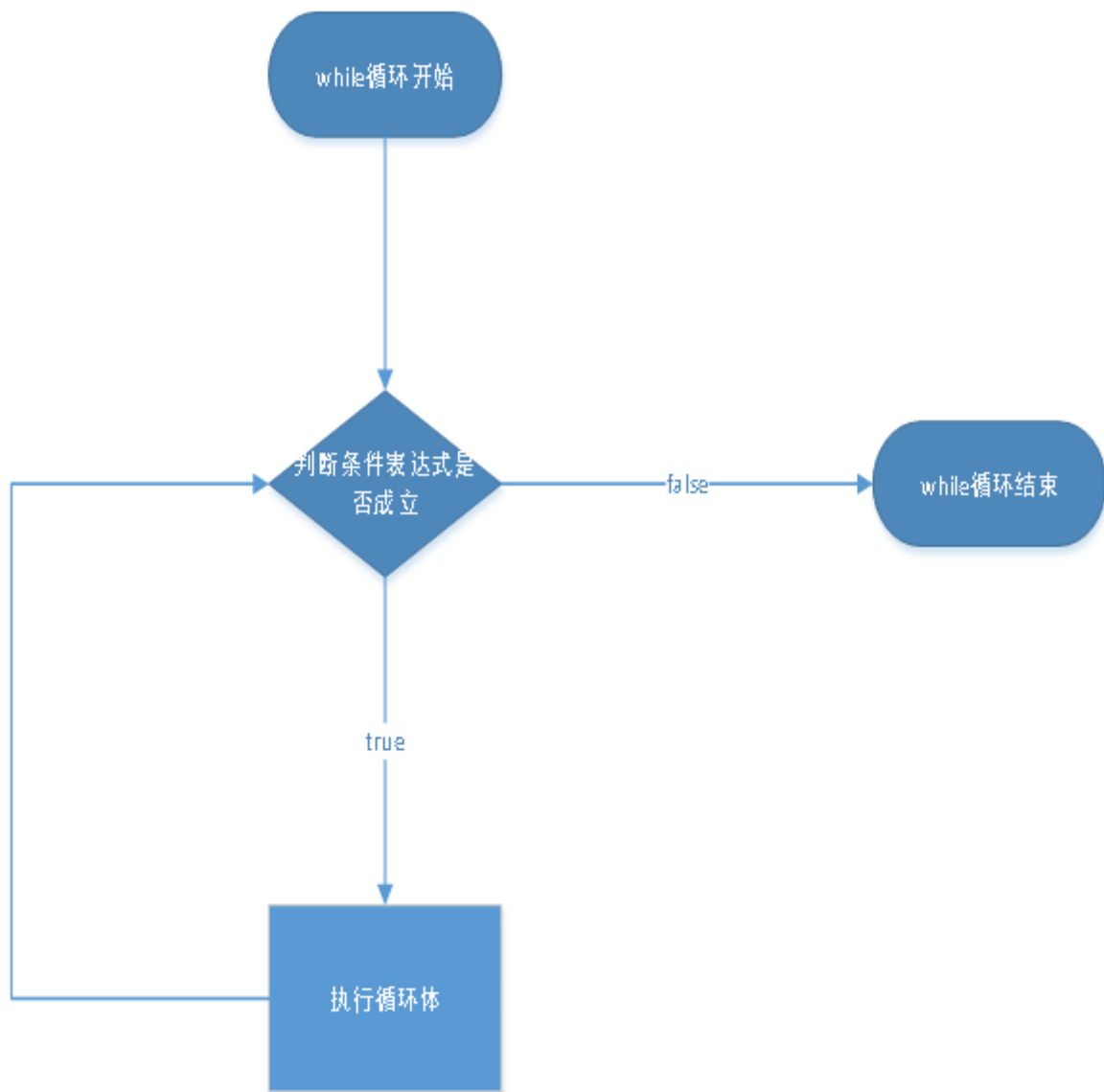
语法：

```
1  初始化变量；
2  while(条件表达式){
3    循环体；
4    变量更新；
5  }
```

结构说明：

当条件表达式成立的时候就执行循环体，反之如果条件表达式不成立就结束while循环！

流程图：



举例:输出10行hello

```
// 先执行变量初始化var i = 1; 接着判断i<=10, 成立进入循环体, 执行循环体,接着变量更新; 变量更新以后 i = 2; 接着判断i<=10, 成立进入循环体,执行循环体,接着变量更新....
```

```
var i = 1;
while( i<= 10){
    document.write("hello<br/>");
    i++;
}
```

注意:当条件一直成立的时候,就会出现死循环,这种情况,我们应该避免!

js调试工具的使用 掌握了有利于理解循环

- 过去调试JavaScript的方式

- `alert()`
- `console.log()`

- 断点调试

断点调试是指自己在程序的某一行设置一个断点，调试时，程序运行到这一行就会停住，然后你可以一步一步往下调试，调试过程中可以看各个变量当前的值，出错的话，调试到出错的代码行即显示错误，停下。

- 调试步骤

1 浏览器中按 **F12** --> `sources` --> 找到需要调试的文件 --> 在程序的某一行设置断点

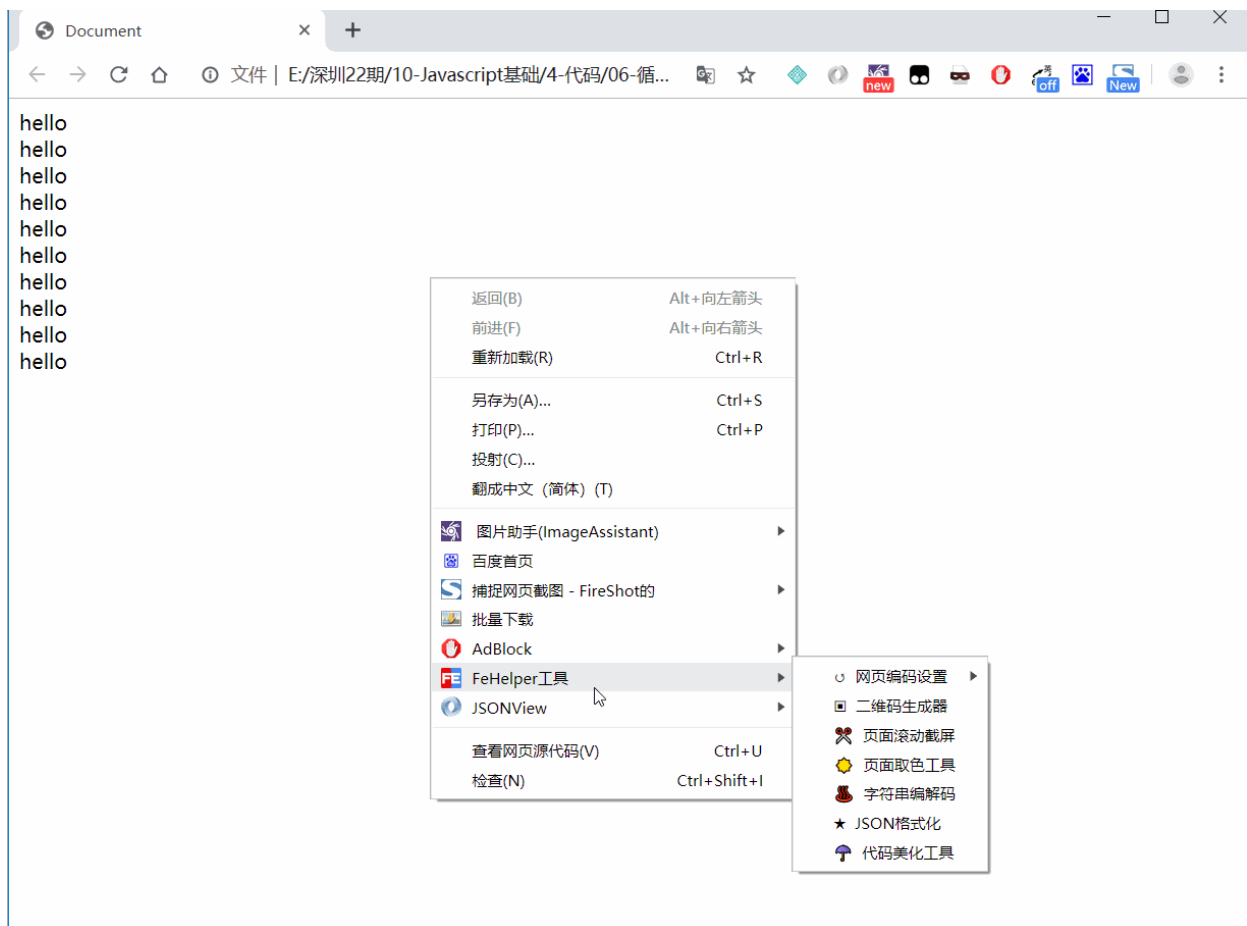
- 调试中的相关操作

- 1 **Watch**: 监视，通过 `watch` 可以监视变量的值的变化，非常的常用。
- 2 **F10**: 程序单步执行，让程序一行一行的执行，这个时候，观察 `watch` 中变量的值的变化。
- 3 **F8**: 跳到下一个断点处，如果后面没有断点了，则程序执行结束。

提示: 监视变量，不要监视表达式，因为监视了表达式，那么这个表达式也会执行。

1. 代码调试的能力非常重要，只有学会了代码调试，才能学会自己解决 bug 的能力。初学者不要觉得调试代码麻烦就不去调试，知识点花点功夫肯定学的会，但是代码调试这个东西，自己不去练，永远都学不会。
2. 今天学的代码调试非常的简单，只要求同学们记住代码调试的这几个按钮的作用即可，后面还会学到很多的代码调试技巧

实际操作如下：动图演示



for循环 重点,必须掌握

do...while明天讲

while和do...while一般用来解决无法确认次数的循环。for循环一般在循环次数确定的时候比较方便

语法:

```
1 for (初始化变量; 条件表达式; 变量更新) {  
2   循环体;  
3 }
```

注意:for循环的表达式之间用的是英文状态下分号分隔的,千万不要写成逗号,

结构说明:

- 第一步:初始化变量 定义一个变量并给其赋值 它只会执行一次
- 第二步:判断条件表达式是否成立 如果成立就执行第三步 如果不成立就会结束for循环
- 第三步:假设条件表达式成立 执行循环体
- 第四步:对变量进行更新 变量更新完毕以后接下来就会重复第二步=>第三步=>第四步

一直到条件表达式不成立了 然后for循环才会结束

执行顺序:

1234 ---- 234 -----234(直到循环条件变成false)

说明:

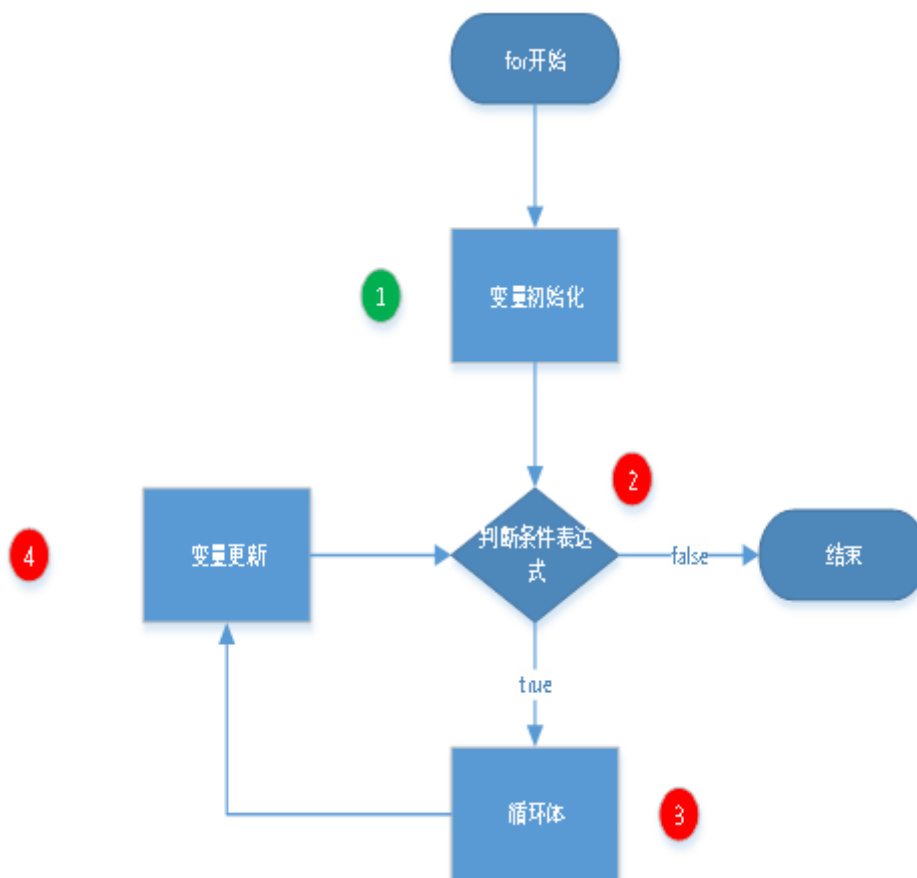
1代表初始化变量

2代表条件表达式

3代表循环体

4代表变量更新

流程图:



举例:使用for循环,输出1~100之间所有的数,包括1跟100

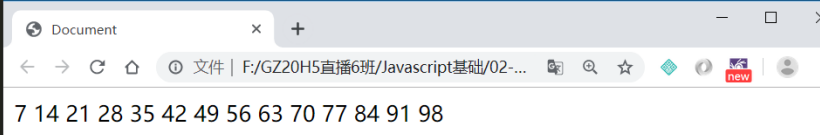
```
// 举例:使用for循环,输出1~100之间所有的数,包括1跟100
for(var i = 1; i <= 100; i++){
    document.write(i+" ");
}
```

举例:输出1~100之间所有7的倍数,比如7 14 21...

```

29 // 举例:输出1~100之间所有7的倍数,比如7 14 21 28 35...
30 // 实现步骤:
31 // 第一步:先输出1~100
32 for(var i = 1;i <= 100; i++){
33     // 第二步:再对输出的结果进行筛选,筛选要使用if分支结构
34
35     if(i%7 == 0){// 如果是7的倍数,才输出i
36         document.write( i + " ");
37     }
38 }
39
40 </script>
41 </head>
42 <body>
43
44 </body>
45 </html>

```

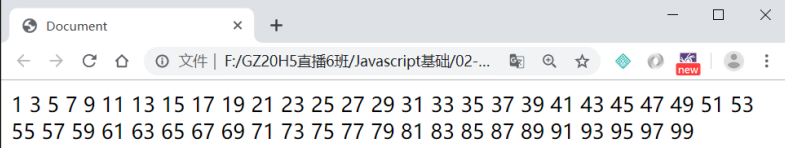


举例:输出1~100之间所有的奇数(不能被2整除的数),比如 1 3 5 7 9 ...

```

42 // 举例:输出1~100之间所有的奇数(不能被2整除的数),比如 1 3 5 7 9 ...
43 for(var i = 1;i <= 100; i++){
44     // 第二步:再对输出的结果进行筛选,筛选要使用if分支结构
45
46     if(i%2 != 0){// 如果不是2的倍数,才输出i
47         document.write( i + " ");
48     }
49 }
50
51 </script>
52 </head>
53 <body>
54
55 </body>
56 </html>

```



举例:输出1~100之间所有偶数(偶数就是能被2整除的数)的和

```
// 举例:输出1~100之间所有偶数(偶数就是能被2整除的数)的和
// 第三步:定义一个变量保存和
var sum = 0;

// 第一步:还是输出1~100
for(var i=1; i <= 100; i++){
    // 第二步:还是输出1~100之间所有的偶数
    if( i%2 == 0){ // 如果能被2整数,那么这数就是偶数
        // sum = sum + i;
        sum += i;
    }
}

document.write( sum );
```

今日总结




xmind,你们懂的

今日作业

上课不懂的代码敲一敲,代码是敲出来,不是看出来

接着再做作业,强化巩固知识

电脑 > 文档 (F:) > GZ20H5直播6班 > Javascript基础 > 02-JavaScript基础 > 5-作业 > 参考答案

名称	修改日期	类型	大小
 01-选做作业-打印九九乘法表.html	2018/11/25 19:25	Chrome HTML D...	1 KB
 02-必做-三五替换.html	2018/11/25 19:37	Chrome HTML D...	2 KB
 03-必做-输出1-100之间能被3整除的数.html	2018/11/25 19:39	Chrome HTML D...	1 KB
 04-必做-鸡兔同笼.html	2018/11/25 19:44	Chrome HTML D...	1 KB
 05-必做-安全数.html	2019/12/17 17:47	Chrome HTML D...	1 KB
 06-选做作业-菱形表格.html	2020/2/6 21:25	Chrome HTML D...	2 KB