> 部门：千锋H5学科部
>
> 作者：芮栋(Hal)

一、流程控制语句

> 前提介绍：
>
> 1. 顺序结构语句 js默认由上至下执行
> 2. 分支结构语句 js会根据条件的判断,决定是否执行某段代码
> 3. 循环结构语句 js会根据条件的判断,反复的执行某段代码

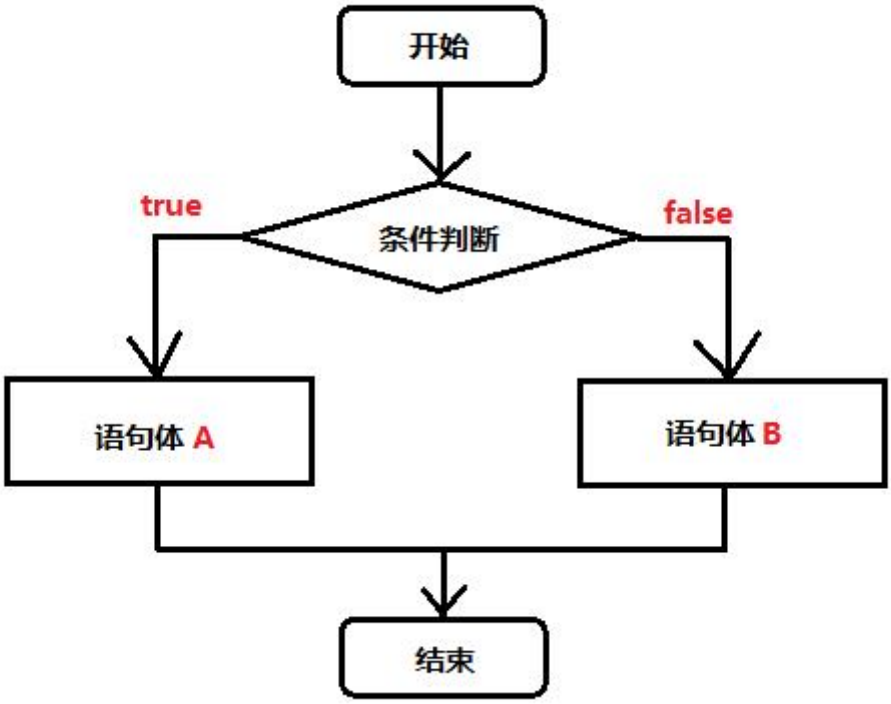
1. prompt函数

prompt()方法用于显示可提示用户进行输入的对话框。

这个方法返回用户输入的字符串。

```
var age = parseInt(prompt('请输入您的年龄:'));
alert(age);
```

2. if语句



- 书写格式：`if (判断式){代码块}`
- **if 语句** - 只有当指定条件为 true 时，使用该语句来执行代码
- if (*condition*){

  当条件为 true 时执行的代码

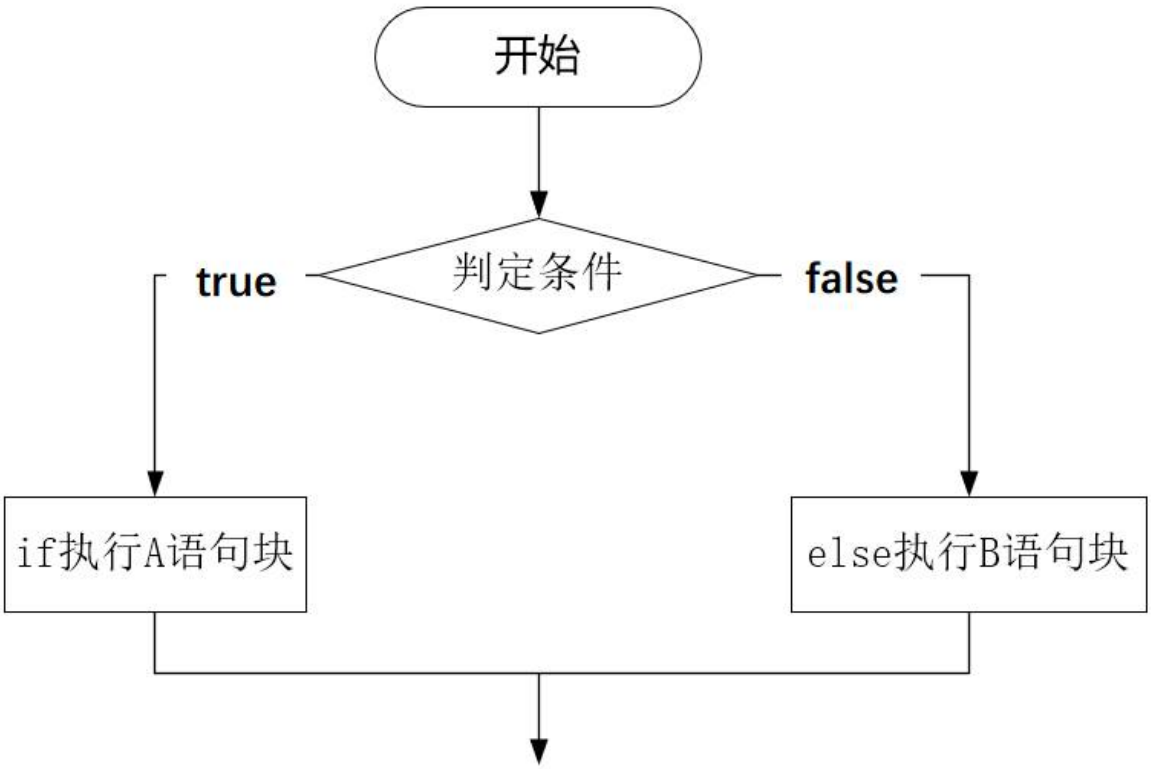
  }
- 请使用小写的if。使用大写字母（IF）会生成 JavaScript 错误!

练习案例：在弹框中输入您的年龄，如果年龄大于18，弹出欢迎来到红浪漫，如果年龄不大于18，则不显示

```
<!DOCTYPE html>
<html lang="en">
      <head>
          <meta charset="UTF-8">
          <title>Title</title>
      </head>
      <body>
          <script>
              var age = parseInt(prompt('请输入您的年
龄:'))
```

```
                if(age > 18) {
                    alert("欢迎来到红浪漫，男宾2位.");
                }
            </script>
        </body>
</html>
```

3. if else语句



- 书写格式：`if（判断式）{代码块1}else{代码块2}`
- 如果判断式为真，则执行语句A，否则将执行语句B
- 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*

  }

练习案例：在弹框中输入您的年龄，如果年龄大于18，弹出欢迎来到红浪漫，否则弹出未成年不能进入。

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Title</title>
    </head>
    <body>
        <script>
            var age = parseInt(prompt('请输入您的年
龄:'))
            if(age > 18) {
                alert("欢迎来到红浪漫");
            }else{
                alert("未成年禁止访问");
            }
        </script>
    </body>
</html>
```

4. if...else if...else 语句

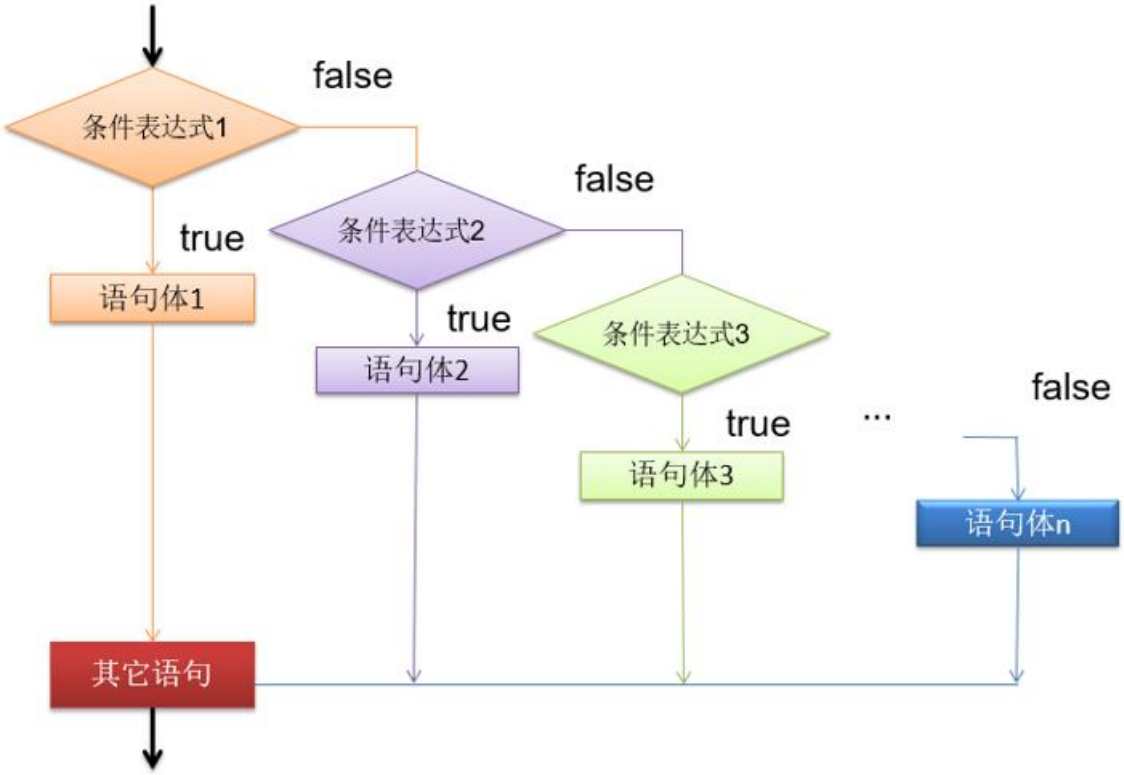
思考：在弹框中输入您的成绩，如果分数少于60，弹出努力吧少年，

如果分数在60到80之间，弹出及格，

如果成绩在80到90之间弹出良好，

如果分数在90到100之间弹出优秀，

如果分数是100，则弹出你可以毕业了，告辞!

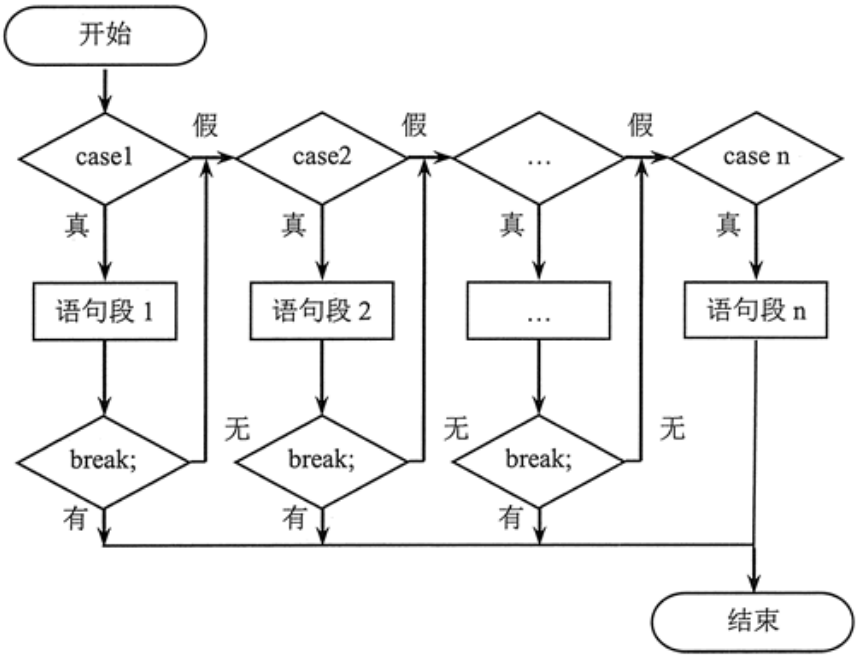



```
<!DOCTYPE html>
<html lang="en">
      <head>
          <meta charset="UTF-8">
          <title>Title</title>
      </head>
      <body>
          <script>
              var score = parseInt(prompt('请输入你的
成绩:'))

              if (score < 60 && score >= 0) {
                  alert('努力吧少年');
              } else if (60 <= score && score < 80)
{
                  alert('及格');
              } else if (80 <= score && score < 90)
{
                  alert('良好');
              } else if (90 <= score && score < 100)
{
```

```
                alert('优秀');
            } else if (score == 100){
                alert('毕业了,你可以起飞了');
            }
        </script>
    </body>
</html>
```

5. switch...case语句



- switch 语句用于基于不同的条件来执行不同的动作。
- 书写格式：

```
switch(n)
{
   case 1:
      执行代码块 1
      break;
   case 2:
      执行代码块 2
      break;
   default:
```

```
        与 case 1 和 case 2 不同时执行的代码
    }
```

- 首先设置表达式 *n*（通常是一个变量）。随后表达式的值会与结构中的每个 case 的值做比较。如果存在匹配，则与该 case 关联的代码块会被执行。请使用 **break** 来阻止代码自动地向下一个 case 运行。

案例练习：在弹框中输入操作序号，如果输入1，则弹出非常满意，

如果输入2，则弹出满意，

如果输入3，则弹出一般，

如果输入4，则弹出不满意，

如果输入5，则弹出非常不满意，

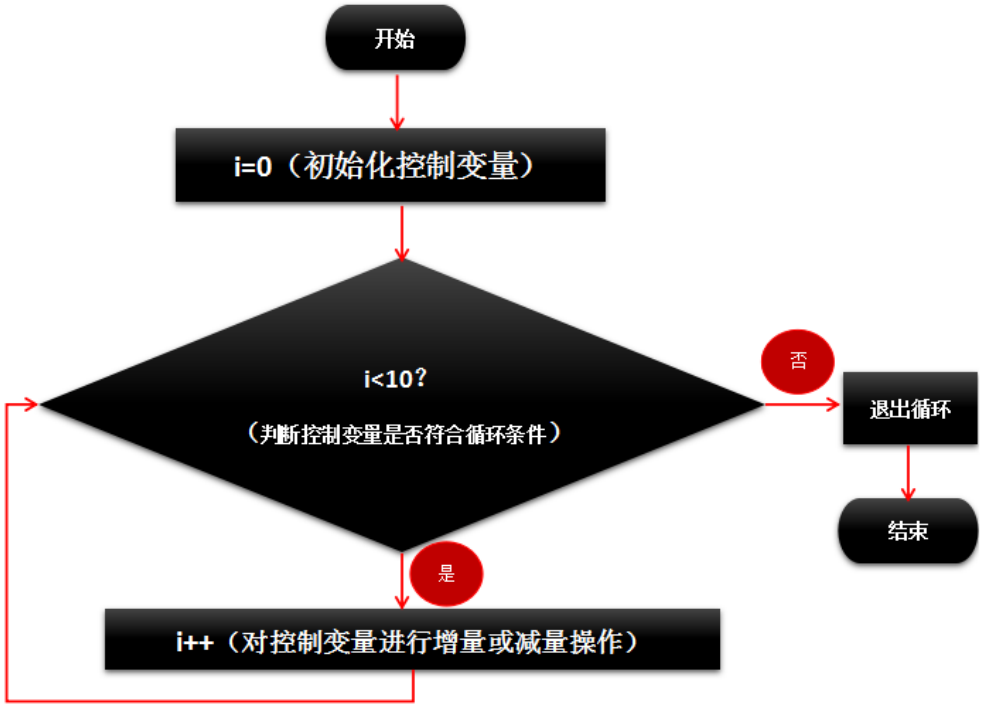
否则弹出关门吧，

```
<!DOCTYPE html>
<html lang="en">
      <head>
          <meta charset="UTF-8">
          <title>Title</title>
      </head>
      <body>
          <script>
              var operation = prompt('请输入你的操作序号:')
              switch(operation){
                  case '1':
                      console.log('非常满意');
                      break;
                  case '2':
                      console.log('满意');
                      break;
```

```
                    case '3':
                        console.log('一般');
                        break;
                    case '4':
                        console.log('不满意')
                        break;
                    case '5':
                        console.console.log('非常不满
意');
                        break;
                    default:
                        console.log('关门吧');
                        break;
                }
            </script>
        </body>
</html>
```

6. for循环

思考：在控制台上打印1到10可以吗？打印1到100呢？1000？10000?1000000000?



- 假如您需要运行代码多次，且每次使用不同的值，那么循环相当方便使用。
- for循环支持多次遍历代码块
- ```
  for (语句 1; 语句 2; 语句 3) {
      要执行的代码块
  }
  语句 1 在循环（代码块）开始之前执行。
  语句 2 定义运行循环（代码块）的条件。
  语句 3 会在循环（代码块）每次被执行后执行。
  ```

1. for循环

   案例练习：打印1到100的数字

   ```
   <!DOCTYPE html>
   <html lang="en">
       <head>
           <meta charset="UTF-8">
           <title>Title</title>
       </head>
       <body>
           <script>
               for(i = 1;i <=100;i++) {
                   console.log(i);
               }
           </script>
       </body>
   </html>
   ```

   作业：

   1. 求1到10的和
   2. 求1到100的和
   ```

3. 求1-100中的偶数

4. 求1-100中的偶数的和

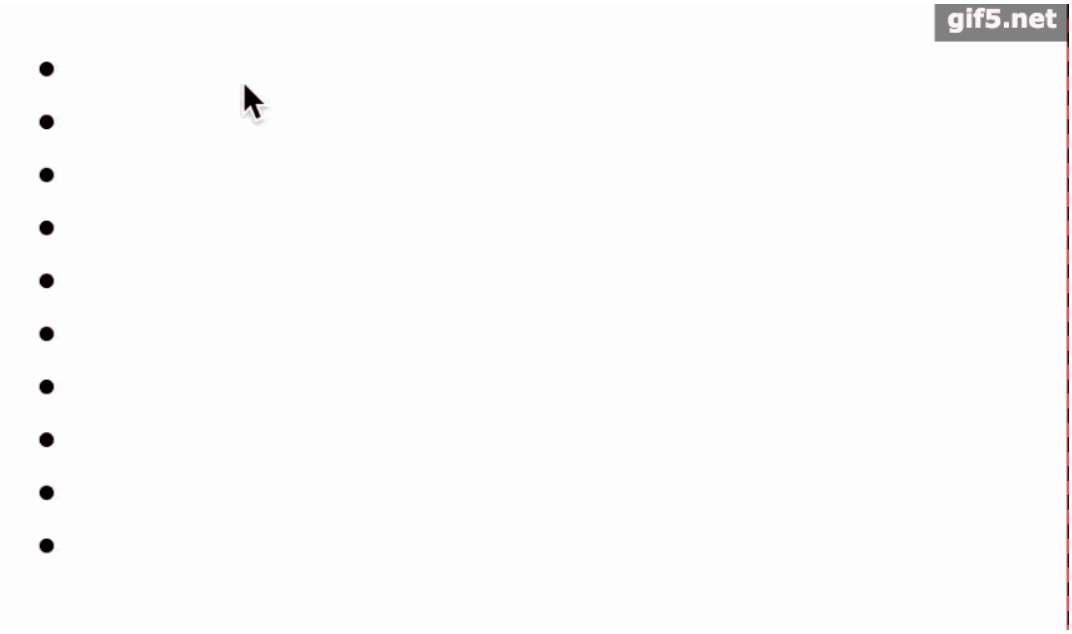
5. 求1-100中是3的倍数的数字和

6. 判断12是不是质数

7. 在页面中输入一个数判断是不是质数

8. 求2-100中所有的质数

9. 点谁谁绿




```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>for循环应用</title>
    </head>
    <body>
        <ul id="box">
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
```

```
            <li></li>
            <li></li>
            <li></li>
            <li></li>
        </ul>
        <script>
            var oLis =
document.getElementsByTagName("li");

            for (var i = 0; i < oLis.length; i
++) {
                oLis[i].onclick = function () {
                    this.style.backgroundColor
= "green";
                }
            }
        </script>
    </body>
</html>
```

10. 九九乘法表

1*1=1
1*2=2 2*2=4
1*3=3 2*3=6 3*3=9
1*4=4 2*4=8 3*4=12 4*4=16
1*5=5 2*5=10 3*5=15 4*5=20 5*5=25
1*6=6 2*6=12 3*6=18 4*6=24 5*6=30 6*6=36
1*7=7 2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49
1*8=8 2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64
1*9=9 2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81

```
<!DOCTYPE html>
<html lang="en">
        <head>
            <meta charset="UTF-8">
            <title>九九乘法算表</title>
```

```
        </head>
        <body>
        <div id="box">
        </div>
        <script>
            var oBox =
document.getElementById("box");
            for (var i = 1; i < 10; i++) {
                for (var j = 1; j <= i; j++) {
                    var math = j + "*" + i + "="
+ j*i+" ";
                    oBox.innerHTML =
oBox.innerHTML + math;
                }
                oBox.innerHTML += "<br>";
            }
        </script>
        </body>
</html>
```

JS打印三角形*我是谁的博客-CSDN博客*js打印三角形

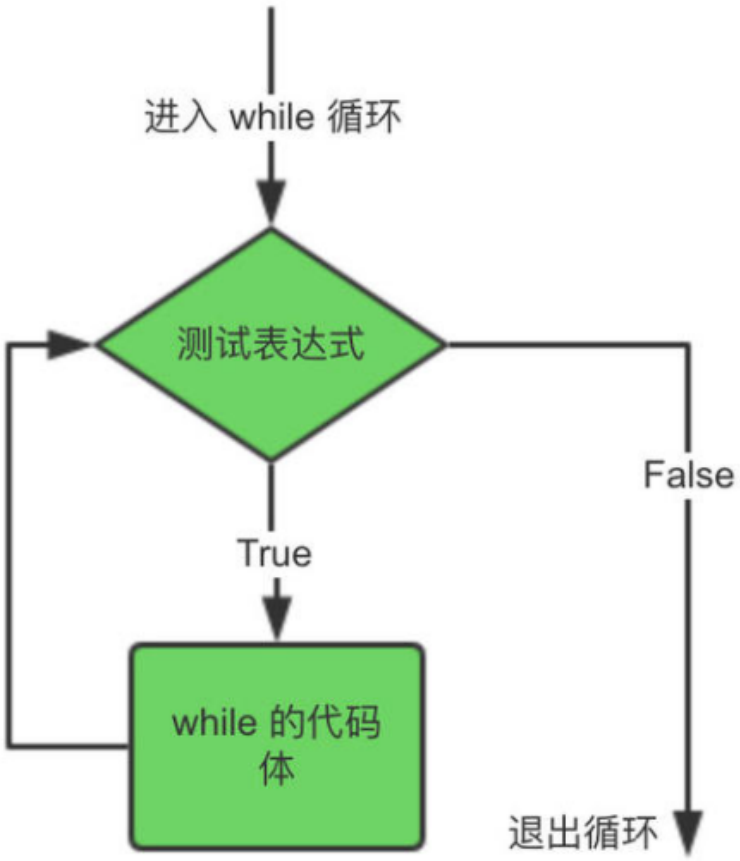
2. for in循环

   for/in 语句循环遍历对象的属性

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>for in循环</title>
    </head>
    <body>
        <script>
            var person=
{fname:"Bill",lname:"Gates",age:56};
            for (x in person){
```

```
                console.log(x + ' ' +
person[x]+'\n');
            }
        </script>
    </body>
</html>
```

7. while循环



while 循环的运行

- while 循环会一直循环代码块，只要指定的条件为 true。
- 语法结构：while (条件) {

  要执行的代码块

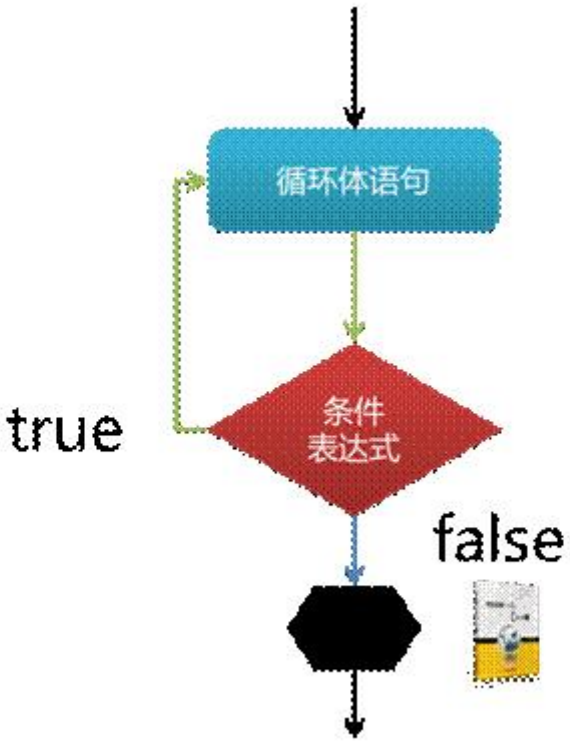
  }

  案例练习：打印1到100的所有数字

```
<!DOCTYPE html>
```

```
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>for in循环</title>
    </head>
    <body>
        <script>
            var i = 1;
            while(i <= 100){
                console.log(i);
                i++;
            }
        </script>
    </body>
</html>
```

8. do while循环



- do/while 循环是 while 循环的变体。在检查条件是否为真之前，这种循环会执行一次代码块，然后只要条件为真就会重复循环

- 语法结构：do {

  要执行的代码块

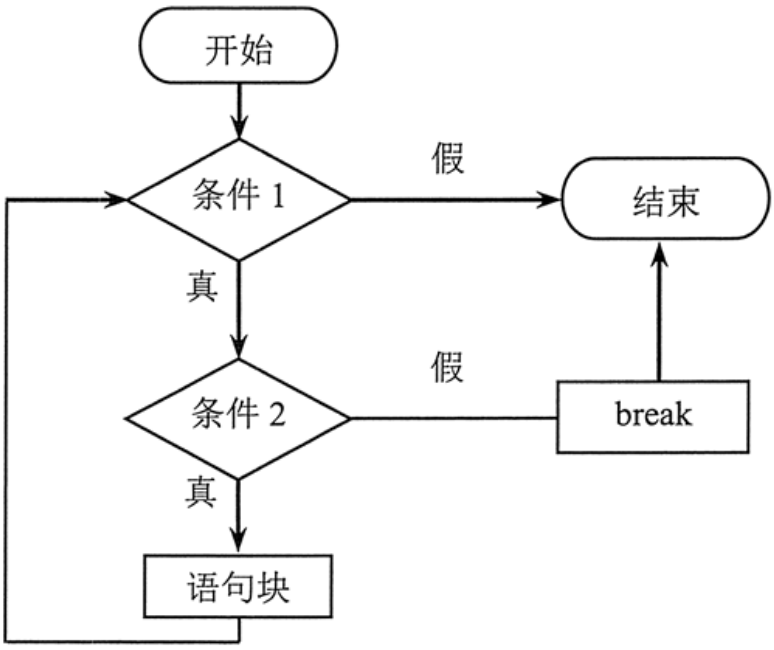
  }while (条件);

案例练习：测试do while循环

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>do while循环</title>
    </head>
    <body>
        <script>
            do{
                console.log('你好'); //先执行一次do里的
代码，然后再判断条件
            }while(1 > 2);
        </script>
    </body>
</html>
```

9. break和continue关键字

break 语句流程控制示意如图所示。



1. break关键字

   - break语句能够结束当前for、for/in、while、do/while或者switch语句的执行。同时break可以接受一个可选的标签名，来决定跳出的结构语句。
   - 如果没有设置标签名，则跳出当前最内层结构。

   案例练习1：for循环遍历范围是1到10，遍历的时候只想打印到3.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>这是一个break的测试文件</title>
    </head>
    <body>
        <script>
            for ( var i=0 ; i<5 ; i++ ) {
                if ( i == 3 ) {
                    break;
                }
                console.log(i);
            }
```

```
        </script>
    </body>
</html>
```

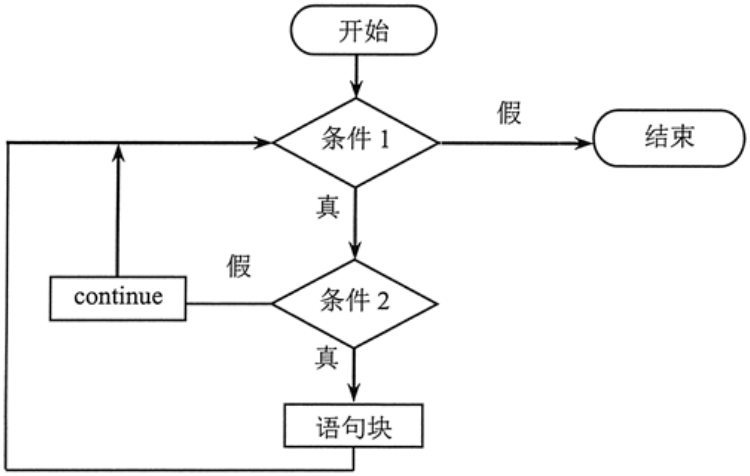
思考：break只能跳出最内层的循环，那么如果有多层的循环，怎么跳出指定哪层或者最外层的for循环？

案例练习2：两层for循环嵌套，每一个都打印0到9，最终打印5，5。

```
<!DOCTYPE html>
<html lang="en">
      <head>
          <meta charset="UTF-8">
          <title>这是一个break的测试文件</title>
      </head>
      <body>
          <script>
              outermost:
              for(var i=0; i<10;i++){
                  for(var j=0; j<10; j++){
                      if(i==5 && j==5){
                          break outermost;
                      }
                  }
              }
              console.log(i,j);
          </script>
      </body>
</html>
```

2. continue关键字

continue语句流程控制示意如图所示。



- continue语句用在循环结构内，用于跳出本次循环中剩余的代码，并在表达式的值为真的时候，继续执行下一次循环。
- 可以接受一个可选的标签名，来决定跳出的循环语句。

案例练习1：遍历0到4的过程中，如果遇到3则跳过。

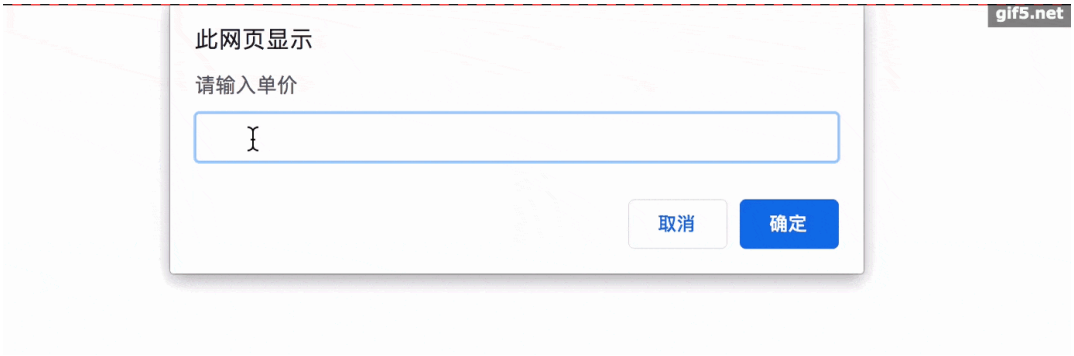
```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>这是一个break的测试文件</title>
    </head>
    <body>
        <script>
            for ( var i=0 ; i<5 ; i++ ) {
                if ( i == 3 ) {
                    continue;
                }
                console.log(i);
            }
        </script>
    </body>
</html>
```

案例练习2：两层for循环嵌套，每一个都打印0到9，当遇到5，5之后不终止当前循环执行下一次循环。

```
<!DOCTYPE html>
<html lang="en">
      <head>
          <meta charset="UTF-8">
          <title>这是一个break的测试文件</title>
      </head>
      <body>
          <script>
            outermost:
            for(var i=0; i<10;i++){
                for(var j=0; j<10; j++){
                    if(i==5 && j==5){
                        continue outermost;
                    }
                    console.log(i,j);
                }
            }
          </script>
      </body>
</html>
```

3. break和continue练习
    - 求整数1～100的累加值，但要求碰到个位为3的数则停止累加
    - 求整数1～100的累加值，但要求跳过所有个位为3的数
    - 求1-100之间不能被7整除的整数的和（用continue）
    - 求200-300之间所有的奇数的和（用continue）
    - 求200-300之间第一个能被7整数的数（break）
    - 收银程序

      输入单价和数量，计算总价。如果总价大于500 则打八折。然后用户输入付钱，最终弹出找零。




```
<!DOCTYPE html>
<html lang="en">
        <head>
            <meta charset="UTF-8">
            <title>收银程序</title>
        </head>
        <body>
        <script>
            //输入商品的单价和数量
            var price = prompt("请输入单价");
            var count = prompt("请输入数量");

            //计算总价
            var sumPrice = price * count;

            //判断是否打折
            if (sumPrice >= 500){
                sumPrice *= 0.8;
            }

            //给用户说需要付多少钱
            var money = prompt("您本次消费"+sumPrice+"元，请付款(输入付钱的面额即可，稍等给您找零)");

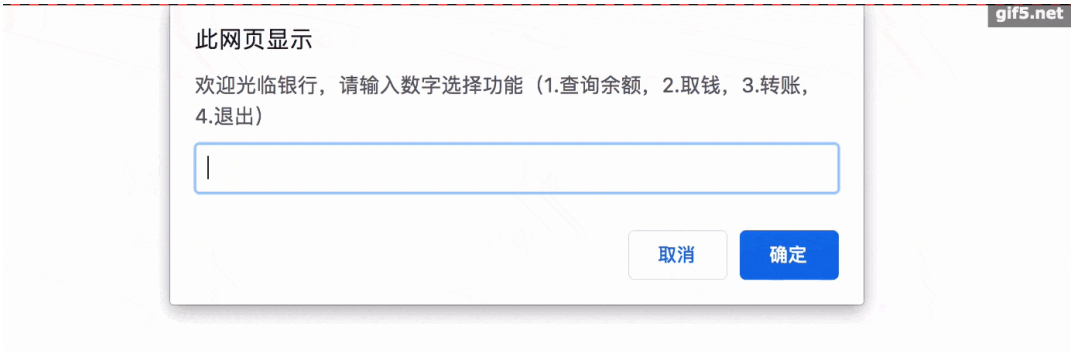
            // 判断用户缴费是否足够，并找零
            if (money > sumPrice) {
                // 计算找零
```

```
                var reduceMoney = money -
sumPrice;
                alert("找零" + reduceMoney +
"元，请收好")
            }else{
                alert("钱不够！！！！")
                //此时可以重新递归调用，代码省略
            }

            alert("欢迎下次光临");
        </script>
        </body>
</html>
```

- ATM取钱

  输入相应数字，执行相应功能



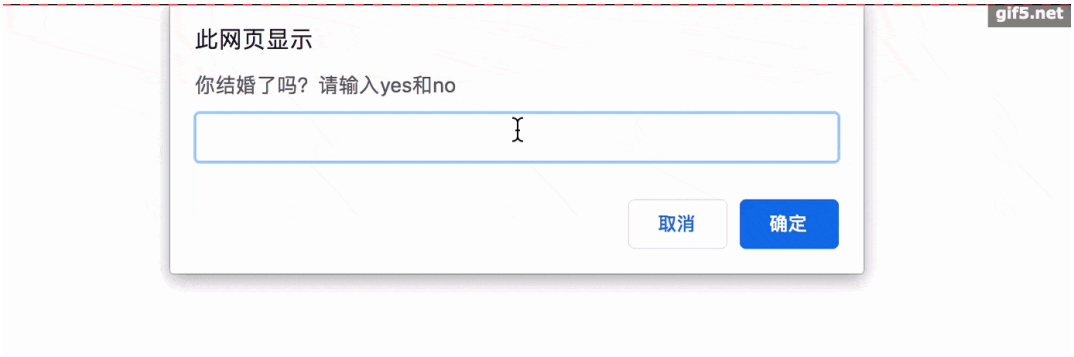

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>ATM取钱</title>
    </head>
    <body>
    <script>
        var userPress = prompt("欢迎光临银行，请输
入数字选择功能（1.查询余额，2.取钱，3.转账，4.退
出）");
        switch (parseInt(userPress)) {
```

```
            case 1:
                search();
                break;
            case 2:
                drag();
                break;
            case 3:
                change();
                break;
            case 4:
                exit();
                break;
            default:
                alert("你丫看不懂提示么");
        }

        function search() {
            alert("正在查询余额...");
        }
        function drag(){
            alert("正在取钱啊...");
        }
        function change() {
            alert("正在转账...")
        }
        function exit() {
            alert("正在退出,告辞!")
        }
    </script>
    </body>
</html>
```

- 买保险

公司给员工买保险（用户通过此程序查询自己是否符合条件）： 1.只要结婚的都买;2.没有结婚的男人 25岁以下不买;3.没有结婚的姑娘 22岁以下不买.




```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>买保险</title>
    </head>
    <body>
    <script>
        /*
        * 公司给员工买保险（用户通过此程序查询自己是否符合条件）：
        * 只要结婚的都买
        * 没有结婚的男人 25岁以下不买
        * 没有结婚的姑娘 22岁以下不买
        */
        //输入是否结婚
        var isMarry = prompt("你结婚了吗？请输入yes和no");
        var age = prompt("您今年高寿啊");
        var sex = prompt("您性别是什么(man or woman)");
        if (isMarry === "yes"){
            alert("您放心做吧，公司给您提供免费的保险");
```

```
        }else if ((sex === "man" && age < 25)
|| (sex === "woman" && age < 22)){
            alert("自己回家买去吧");
        }else{
            alert("公司也给您买");
        }
    </script>
    </body>
</html>
```

10. 死循环

死循环:在循环中,没有结束条件的循环是死循环,程序中要避免的,否则会造成内存溢出

- 第一种:while(true);
- 第二种:for(;true;);
- 第三种;for(;;);

11. 扩展

1. js中if和switch该如何选择?

   if 和 switch 都可以设计多重分支结构，一般情况下 switch 执行效率要高于 if 语句。但是也不能一概而论，应根据具体问题具体分析。简单比较如表所示。

   if 和 switch 的比较

语句	If 语句	Switch 语句
结构	通过嵌套结构实现多重分支	专为多重分支设计
条件	可以测试多个条件表达式	仅能测试一个条件表达式
逻辑关系	可以处理复杂的逻辑关系	仅能处理多个枚举的逻辑关系
数据类型	可以适用任何数据类型	仅能应用整数、枚举、字符串等类型

   相对而言，下面情况更事宜选用 switch 语句。

   - 枚举表达式的值。这种枚举是可以期望的、平行的逻辑关系。
   - 表达式的值具有离散性，是不具有线性的非连续的区间值。

- 表达式的值是固定的，不会动态变化。
- 表达式的值是有限的，不是无限的，一般应该比较少。
- 表达式的值一般为整数、字符串等简单的值。

下面情况更事宜用 if 语句。

- 具有复杂的逻辑关系。
- 表达式的值具有线性特征，去对连续的区间值进行判断。
- 表达式的值是动态的。
- 测试任意类型的数据。

案例1：根据学生分数进行等级评定：如果分数小于 60，则不及格；如果分数在 60 与 75 之间，则评定为合格；如果分数在75与 85 之间，则评定为良好；如果分数在 85 与 100 之间，则评定为优秀。

根据上述需求描述，确定检测的分数是一个线性区间值，因此选用 if 语句会更合适。

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>优化</title>
    </head>
    <body>
        <script>
            if (score < 60) {
                console.log("不及格");
            }else if (score < 75) {
                console.log("合格");
            }else if (score <85) {
                console.log("良好");
            }else {
                console.log("优秀");
            }
        </script>
```

```
    </body>
</html>
```

如果使用 switch 结构，则需要枚举 100 种可能，如果分数值还包括小数，这种情况就更佳复杂了，此时使用 switch 结构就不是明智之举。

案例2： 设计根据性别进行分类管理。这和案例属于有效枚举条件，使用 switch 会更高效。

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>优化</title>
    </head>
    <body>
        <script>
           switch (sex) {  //离散值判断
                case 1 :
                    console.log("女士");
                    break;
                case 2 :
                    console.log("男士");
                    break;
                default :
                    console.log("请选择性别");
            }
        </script>
    </body>
</html>
```

2. js中for和while如何选择?

   for 和 while 语句都可以完成特定动作的重复性操作。不过，使用时不可随意替换。

for 语句是以变量的变化来控制循环进程的，整个循环流程是计划好的，可以事先知道循环的次数、每次循环的状态等信息。

while 语句是根据特定条件来决定循环进程的，这个条件时动态的，无法预知的，存在不确定性，每一次循环时都不知道下一次循环的状态如何，只能通过条件的动态变化来确定。

因此，for 语句常用于有规律的重复操作中，如数组、对象等迭代。while 语句更适用于特定条件的重复操作，以及依据特定事件控制的循环操作

一般来说，在循环结构中动态改变循环变量的值时，建议使用 while 结构，而对于静态的循环变量，则可以考虑使用 for 结构。简单比较 while 和 for 语句，区别如表所示。

while 语句和 for 语句的比较

语句	while 语句	for 语句
条件	根据条件表达式的值决定循环操作	根据操作次数决定循环操作
结构	比较复杂，结构相对宽松	比较简洁，要求比较严格
效率	存在一定的安全隐患	执行效率比较高
变种	do/while 语句	for/in 语句