循环结构

循环：指事物周而复始地运动或变化。

在实际问题中有许多具有规律性的重复操作，因此在程序中就需要重复执行某些语句。

特征：
1.有规律性的重复操作
2.重复执行的代码极其相似

如：输出10次'hello world'

```
console.log('hello world 1');
console.log('hello world 2');
console.log('hello world 3');
......
console.log('hello world 10');
```

这样处理起来非常的费时费力，同时也会有非常多的冗余代码!

假如我要输出 100次 1000次 'hello world' 呢? ?

FOR 循环

语法形式为：

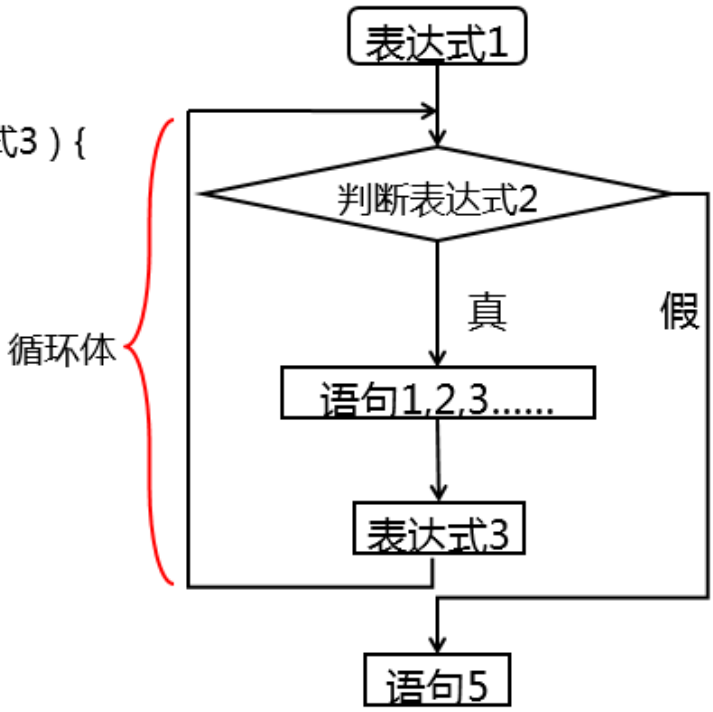
```
for (表达式1; 表达式2; 表达式3) {
    循环体;
}
```

表达式1：为不参与循环的单次表达式，用来给循环控制变量赋初值
表达式2：一般是一个关系表达式，作为循环条件(设置终止值)
表达式3：一般为循环变量增量或减量(步长)
循环体：需要重复执行的代码

```
for (var i = 0; i < 5; i++) { // 增量循环
    console.log(i);
}

for (var i = 5; i >= 1; i--) {// 减量循环
    console.log(i);
}
```

```
for（表达式1；判断表达式2；表达式3）{
    语句1；
    语句2；
    语句3；
    语句……；
}
语句5
```



注意：

- for()括号中的表示式皆可以省略，但分号不可省略。
- 省略了表达式2(循环条件), 若不做其它处理则成为死循环。
- 死循环：没有终止条件并一直执行的循环即为死循环。
- for循环的嵌套，可以简单的理解为行和列的关系。

WHILE 循环

- `while`，中文叫 当...时，其实就是当条件满足时就执行代码，一旦不满足了就不执行了
- 语法 `while（条件）{ 满足条件就执行 }`
- 因为满足条件就执行，所以我们写的时候一定要注意，就是设定一个边界值，不然就一直循环下去了

```
// 1. 初始化条件
var num = 0;
// 2. 条件判断
while (num < 10) {
  // 3. 要执行的代码
  console.log('当前的 num 的值是 ' + num)
  // 4. 自身改变
  num = num + 1
}
```

  - 如果没有自身改变，那么就会一直循环不停了

DO WHILE 循环

- 是一个和 `while` 循环类似的循环
- `while` 会先进行条件判断，满足就执行，不满足直接就不执行了
- 但是 `do while` 循环是，先不管条件，先执行一回，然后在开始进行条件判断
- 语法： `do { 要执行的代码 } while (条件)`

```
// 下面这个代码，条件一开始就不满足，但是依旧会执行一次 do 后面 {} 内部的代码
var num = 10
do {
  console.log('我执行了一次')
  num = num + 1
} while (num < 10)
```

for，while do while三个循环的区别

for 循环一般用在循环次数可以确定的情景。

while 循环一般用在循环次数未知的情景。

do while：先执行一次，在判断，也是不知道循环次数

案例解析

```
面试题：
var k=0;
for(var i=0, v=0; i<6, v<9; i++, v++){
    k = i + v;
}
console.log(k);

var k=0;
for(var i=0, v=0; i<9, v<6; i+=2, v++){
k = i + v;
}
console.log(k);
```

BREAK 终止循环

- 在循环没有进行完毕的时候，因为我设置的条件满足，提前终止循环
- 比如：我要吃五个包子，吃到三个的时候，不能在吃了，我就停止吃包子这个事情
- 要终止循环，就可以直接使用 `break` 关键字

```
for (var i = 1; i <= 5; i++) {
  // 没循环一次，吃一个包子
  console.log('我吃了一个包子')
  // 当 i 的值为 3 的时候，条件为 true，执行 {} 里面的代码终止循环
  // 循环就不会继续向下执行了，也就没有 4 和 5 了
  if (i === 3) {
    break
  }
}
```

CONTINUE 结束本次循环

- 在循环中，把循环的本次跳过去，继续执行后续的循环
- 比如：吃五个包子，到第三个的时候，第三个掉地下了，不吃了，跳过第三个，继续吃第四个和第五个
- 跳过本次循环，就可以使用 `continue` 关键字

```
for (var i = 1; i <= 5; i++) {
  // 当 i 的值为 3 的时候，执行 {} 里面的代码
  // {} 里面有 continue，那么本次循环后面的代码就都不执行了
  // 自动算作 i 为 3 的这一次结束了，去继续执行 i = 4 的那次循环了
  if (i === 3) {
    console.log('这个是第三个包子，掉地下了，我不吃了')
    continue
  }
  console.log('我吃了一个包子')
}
```