Javascript03_循环结构

循环结构

循环语句可以让我们反复多次的执行同一段代码，他们工作原理是：只要给定的条件可以满足，包含在循环语句里的代码就将重复地执行下去，一旦给定条件的求值结果不再是true，循环也就到此为止。

循环结构分类

一. for循环

for (循环变量初始化表达式; 循环条件表达式; 循环变量变化表达式)
{
    循环体
}
注：for循环一般用来将循环体中的代码重复执行指定的次数。

```
for (var i = 1; i <= 10; i++) {
    console.log(i);
}
```

用for循环来控制代码的好处是循环控制结构更加清晰。与循环有关的内容都在for后的()里。

- for循环结构适用于明确知道循环次数的循环
- for循环语法格式

```
for(循环变量初始化;循环条件判断;循环变量递增){
  循环体;
}
```

  - for循环的执行流程

```
(1) --循环变量初始化
(2) --循环条件判断
(3) --循环变量递增
(4) --循环体
循环执行的流程： (1) -> (2) -> (4) -> (3) -> (2) -> (4) -> (3) ...-> (2)
```

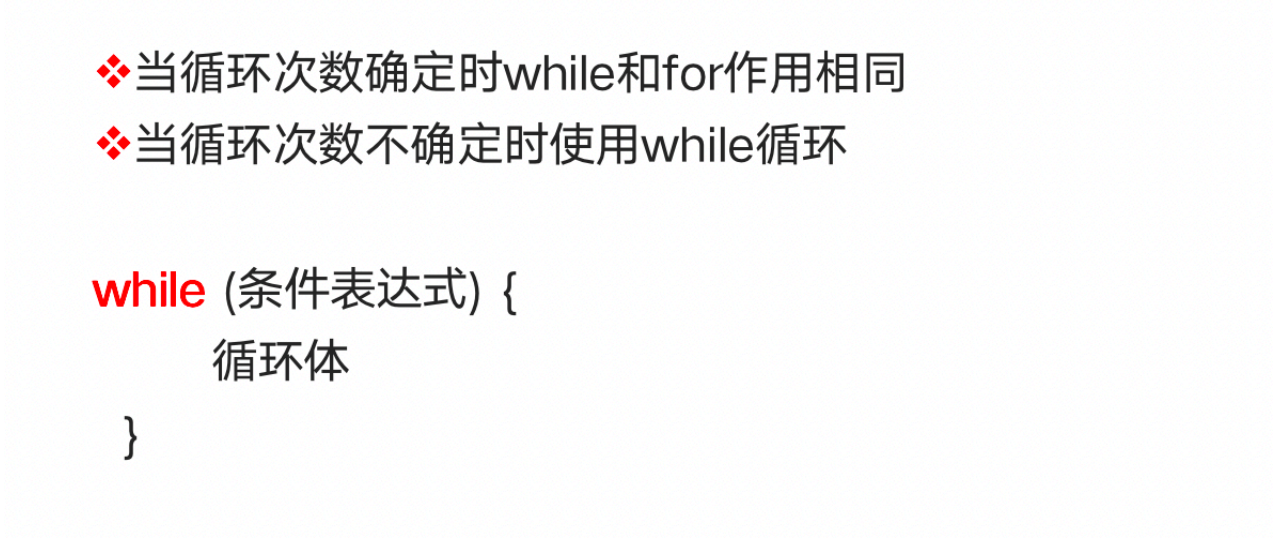
- for循环的使用

```
// 使用for循环输出1-100所有的数字
for(var i = 0; i < 100; i++){
  console.log(i+1);
}
/*
 var i = 0:该操作即为循环变量初始化操作，目的是定义一个变量充当计数器，用来标明循环执行的次数
 i < 100:该操作即为循环条件判断，目前是判断本次循环是否能够执行
 console.log(i+1):该操作即为循环体
 i++:操作为循环变量递增
*/
```

- for循环可以将需要重复书写的代码进行缩减，有效减少代码的冗余
- 循环嵌套
  - 循环内部再嵌套一个循环，注意循环嵌套中经常用到双层for循环嵌套，一般在双层for循环中外层循环控制输出的行数，内层循环控制输出的列数

```
//输出电影院所有的座位号码
for(var i = 0; i < 6; i++){
  for(var j = 0; j< 30; j++){
    console.log("第"+(i+1)+"排第"+(j+1)+"号");
  }
}
```

二. while循环



- while循环适用于不知道循环次数，但是明确知道循环结束的条件。注意for循环可以被while循环替代，但是while循环不可以被for循环替代。
- while循环的语法结构

```
//condition代表循环的条件
while (condition){
      循环体
}
```

```
var count=1;
while(count<=11)
  {
    alert(count);
    count++;
  }
```

分析：

首先创建变量count并赋值为1

while后的小括号中count<=11表示只要count的变量的值不超过11就一直保持执行循环（一直执行alert(count)和count++这两句）也就是{}中的内容。

在循环体中用count++对这个变量做每次执行都+1的操作。直到加到12不再满足while的循环条件。

- 注意
  - while循环在替代for循环时，一定要保证存在一个变量记录循环执行的次数，并且每次循环结束一定保证循环变量递增，否则就会出现死循环。
  - 死循环99%情况下是有害的

```
//使用while循环输出1-100的所有的偶数
var i = 1; //循环变量初始化
while (i <= 100){
  if(i % 2 == 0){
    console.log(i);
  }
  //循环变量递增
  i++;
}
```

三. do...while循环

但是while循环有个弊端，那就是先判断了循环条件，如果循环条件不满足那么一次都不会执行循环体。我们如果循环体需要先执行一次再来判断，那我们需要使用do...while。

❖do...while 循环是 while 循环的变体。

❖该循环会先执行一次循环体，然后再判断循环条件，决定是否进行下一次循环

```
do {
    循环体
} while (条件表达式)
```

- do...while循环的语法结构

```
循环变量初始值;
do {
  循环体;
  循环增量;
}while(循环条件)
```

- 注意:
  - do...while与while循环的区别

    do...while会先执行循环体再去判断条件是否成立, 而while循环先判断循环条件是否成立再决定是否执行循环体. 所以, 如果循环条件一开始就为假, do...while也会执行一遍, 而while循环一次都不执行; 如果循环条件一开始成立, 那么do...while与while的执行次数和结果没有区别

break关键字和continue关键字

- break：用在循环中，用来结束break关键字所在的循环结构。注意代码执行break以后，不管后面是否还有未完成的循环，循环都不再执行。
- continue：用在循环中，用来结束本次循环。注意代码执行continue之后，continue后面的代码不会执行，循环直接跳到循环变量递增之后继续执行后面的循环

```
 //输出1-100中所有7的倍数
     for(var i = 0; i < 100; i++){
       if((i+1) % 7 != 0){
           //结束本次循环，循环直接跳转到循环变量递增
         continue;
       }
       console.log(i+1);
     }
//输出1-100中第一个7的倍数
     for(var i = 0; i < 100; i++){
       if((i+1)%7 == 0){
         console.log(i+1);
         //结束break当前所处的循环
```

```
            break;
        }
    }
```

四. 随机数

1. 产生一个[0, 1)的随机小数 Math.random()
2. 产生一个[0, n)的随机小数 Math.random()*n
3. 产生一个[0, n)的随机整数(向下取整) Math.floor(Math.random()*n)
4. 产生一个[0, n]的随机整数 Math.floor(Math.random()*(n+1))
5. 产生一个[m, n]的随机整数Math.floor(Math.random()*(n-m+1)+m)

```
var ran1 = Math.random();
console.log(ran1);

var ran2 = Math.random()*100;
console.log(ran2);

console.log(Math.floor(99.0001));

var ran3 = Math.floor(Math.random()*100);
console.log(ran3);

var ran4 = Math.floor(Math.random() * (70 - 30 + 1) + 30);
console.log(ran4);
```

```
0.6162150283285797                01_随机数r.html:24
81.86287146718145                 01_随机数r.html:27
99                                01_随机数r.html:29
6                                 01_随机数r.html:32
46                                01_随机数r.html:36
>
```