普通for循环

for循环最常用的地方是利用索引来遍历数组：

var arr = ['Apple', 'Google', 'Microsoft'];

var i, x;

for (i=0; i<arr.length; i++) {

x = arr[i];

console.log(x);

}

for循环的3个条件都是可以省略的，如果没有退出循环的判断条件，就必须使用break语句退出循环，否则就是死循环：

var x = 0;

**for** (;;) { // 将无限循环下去

**if** (x > 100) {

**break**; // 通过**if**判断来退出循环

}

x ++;

}

for ... in

for循环的一个变体是for ... in循环，它可以把一个对象的所有属性依次循环出来：

**var** o = {

name: 'Jack',

age: 20,

city: 'Beijing'

};

**for** (**var** key **in** o) {

console.log(key); *// 'name', 'age', 'city'*

}

要过滤掉对象继承的属性，用hasOwnProperty()来实现：

**var** o = {

name: 'Jack',

age: 20,

city: 'Beijing'

};

**for** (**var** key **in** o) {

**if** (o.hasOwnProperty(key)) {

console.log(key); *// 'name', 'age', 'city'*

}

}

由于Array也是对象，而它的每个元素的索引被视为对象的属性，因此，for ... in循环可以直接循环出Array的索引：

**var** a = ['A', 'B', 'C'];

**for** (**var** i **in** a) {

console.log(i); *// '0', '1', '2'*

console.log(a[i]); *// 'A', 'B', 'C'*

}

请注意，for ... in对Array的循环得到的是String而不是Number。

### while

for循环在已知循环的初始和结束条件时非常有用。而上述忽略了条件的for循环容易让人看不清循环的逻辑，此时用while循环更佳。

while循环只有一个判断条件，条件满足，就不断循环，条件不满足时则退出循环。比如我们要计算100以内所有奇数之和，可以用while循环实现：

**var** x = 0;

**var** n = 99;

**while** (n > 0) {

x = x + n;

n = n - 2;

}

x; *// 2500*

在循环内部变量n不断自减，直到变为-1时，不再满足while条件，循环退出。

### do ... while

最后一种循环是do { ... } while()循环，它和while循环的唯一区别在于，不是在每次循环开始的时候判断条件，而是在每次循环完成的时候判断条件：

**var** n = 0;

**do** {

n = n + 1;

} **while** (n < 100);

n; *// 100*

用do { ... } while()循环要小心，循环体会至少执行1次，而for和while循环则可能一次都不执行。

·反引号