JavaScript基礎-第3回資料

◇代入演算子

例1

```
1  var num = 1 ;
2  num += 1 ; //num=num+1
3  num -= 1 ; //num=num-1
4  num *= 1 ; //num=num*1
5  num /= 1 ; //num=num/1
6  num %= 1 ; //num=num%1
7  var str = "1+1=" ;
8  str += 2 ; //str=str+2
```

<例1>における各文の意味はそれぞれのコメントと同様である
このように書くことによって演算はかなり簡潔になる
8行目は算術演算子の+（2行目）とは異なる連結演算子（文字の結合）である点に注意

◆for/while

例2

```
1  for ( var count = 0 ; count <5 ; count ++){
2          print ( count + '¥n' );
3  }
```

例3

```
1  var count = 0 ;
2  while ( count <5 ){
3          print ( count + '¥n' );
4          count ++;
5  }
```

for/whileはいずれも条件をチェックし真の間、繰り返し（ループして）続くブロックを実行する
文法は"for(初期化式;条件式;処理後に実行される式){処理}"と"while(条件式){処理}"となっている
ここでの"count<5"はcountが5よりも小さいという条件で、<例2>と<例3>の実行結果は同じである
一般的にはfor内でカウンタとして使用する変数が必要な場合、わざわざwhileを使って書くことは少ない
一方、forブロックの外側ですでに存在する変数などがループの条件になる場合はwhileを使うことが多い

この他にdo while文が存在するが、その構文は"do{処理}while(条件);"である
条件の真偽を処理後に判定するため、whileとは違って最低1回は処理が実行される
実のところ、これはほとんど使わないと思うのでこれ以上説明はしない

例4

```
function inc(){
	i++;
}
for(var i=0;i<5;inc()){
	print(i+'¥n');
}
```

forの処理後に実行される式はインクリメントかデクリメントが使われることが多いが、どのような命令でも1ステートメントであれば入れることができる<例4>

例5

```
for(var i=0;i<5;i++){
	if(i==3)break;
	print(i+'¥n');
}
```

switchの時に紹介したbreakをforまたはwhileの中で使うことによって、そのレベルのfor/whileをスキップできる（→ループが2重になっているときは注意）
また、breakの代わりにcontinueを使うと、for/whileをスキップするのではなく、そのブロックのcontinue以降のみをスキップし、for/while自体はスキップしない

◇余計な話
他の言語を知っている人は「JavaScriptはbreak/continue時にラベルを指定できないのか！」と思っただろう
もちろん、JavaScriptにもラベルは存在し、break/continue時にラベルを指定することはできる
ただ、私はラベルを使ったことがないし、他人のコード中でもラベルを使っているのを見たことがない
それぐらい使わないのでここで言及するだけにとどめる
さらに言っておくとJavaScriptにはgotoがない（ただし、それ以外はC言語と同等の制御構文があると記憶している）