

JS语法

语句块

JS使用大括号构成语句块。

ES6 之前语句块是没有作用域的，从ES6开始支持块作用域，let只能在块作用域内可见

```
1  function hello() {
2      let a = 1;
3      var b = 2;
4      c = 3
5  }
6
7  //let d = 100
8  if (1)
9  {
10     let d = 4; // 去掉let会怎么样?
11     var e = 5;
12     f = 6
13     if (true) {
14         console.log(d)
15         console.log(e)
16         console.log(f)
17         console.log('-----')
18         g = 10
19         var h = 11
20     }
21 }
22
23 //console.log(a) // 不可见
24 //console.log(b) // 不可见
25 //console.log(c) // 不可见吗?
26
27 //console.log(d) // 块作用域使用let, 不可见;但是块外的d可见
28 console.log(e) // 块作用域使用var, 可见
29 console.log(f) // 块作用域隐式声明, 可见
30 console.log(g) // 可见
31 console.log(h) // 可见
```

流程控制

条件分支

```

1  if (cond1){
2
3  }
4  else if (cond2) {
5
6  }
7  else if (cond3) {
8
9  }
10 else {
11
12 }

```

```

1  条件的False等效
2      false
3      undefined
4      null
5      0
6      NaN
7      空字符串
8
9  其它值都将被视为True

```

switch...case分支语句

```

1  switch (expression) {
2      case label_1:
3          statements_1
4          [break;]
5      case label_2:
6          statements_2
7          [break;]
8      ...
9      default:
10         statements_def
11         [break;]
12 }

```

这里最大的问题，就是穿透问题，一定要在case中恰当的使用break语句，否则就会继续顺序向下执行。

```

1  let x = 5 // 换成1试一试
2  switch (x) {
3      case 0:
4          console.log('zero')
5          break;
6      case 1:
7          console.log('one');
8      case 2:
9          console.log('two');
10     case 3:
11         console.log('three');
12         break;
13     case 5:
14     case 4:

```

```

15     console.log('four');
16     default:
17         console.log('other')
18         // break;
19 }

```

switch...case语句都可以写成多分支结构。

for循环

```

1 // C风格for循环
2 for ([initialExpression]; [condition]; [incrementExpression])
3 {
4     statement
5 }

```

```

1 for (let i=0;i<10;i++){
2     console.log(i)
3 }
4 console.log('~~~~~')
5
6 for(var x=0,y=9;x<10;x++,y--){
7     console.log(x*y)
8 }
9 console.log('~~~~~')
10
11 for (let i=0;i<10;i+=3){ // 步长
12     console.log(i) // i++ ++i
13 }

```

while循环 和 do...while循环

```

1 while (condition)
2     statement

```

条件满足，进入循环，条件为真，继续循环

```

1 do
2     statement
3 while (condition);

```

先进入循环，然后判断，为真就继续循环

```

1 let x = 10;
2 while (x-->0) {
3     console.log(x);
4 }
5 console.log('~~~~~')
6 do {
7     console.log(x);
8 }while(x-->0)
9 // 分析这个程序的打印结果

```

练习

九九乘法表，使用JS实现

```
1 for (let i=1;i<10;i++){
2   line = '';
3   for (let j=1;j<=i;j++){
4     line += `${j}*${i}=${i*j} `;
5   }
6   console.log(line)
7 }
```

for...in循环

对象操作语句for...in用来遍历对象的属性

```
1 for (variable in object) {
2   statements
3 }
```

```
1 // 数组
2 let arr = [10, 20, 30, 40];
3
4 console.log(arr[1]) // 20
5
6 for (let x in arr)
7   console.log(x); // 返回索引
8
9 for (let index in arr)
10  console.log(`${index} : ${arr[index]}`); // 插值
11
12 // C风格
13 for(let i=0;i<arr.length;i++)
14   console.log(arr[i]);
15
16 // 对象
17 let obj = {
18   a:1,
19   b:'magedu',
20   c:true
21 };
22
23 console.log(obj.a);
24 console.log(obj['b']); // 对象属性当索引访问
25 console.log(obj.d); // undefined
26 console.log('~~~~~')
27
28 for (let x in obj)
29   console.log(x); // 属性名
30
31 for (let key in obj) // 返回属性名，如同key
32   console.log(`${key} : ${obj[key]}`);
```

for in 循环返回的是索引或者key，需要间接访问到值。

数组反正返回的是索引，C风格for循环操作可能方便点。根据个人喜好选择。

对象用for in合适。

for...of 循环

ES6的新语法

```
1 // for of
2 let arr = [1,2,3,4,5]
3 let obj = {
4   a:1,
5   b:'magedu',
6   c:true
7 }
8
9
10 for (let i of arr) { // 返回数组的元素
11   console.log(i)
12 }
13
14 for (let i of obj) { // 异常，不可以迭代
15   console.log(i)
16 }
```

注意：for ... of 不能迭代一个普通对象。

原因是，of后面必须是一个迭代器（TypeError: obj[Symbol.iterator] is not a function）

可类比python中的for in，例如for x in []

break、continue

break 结束当前循环

continue 中断当前循环，直接进入下一次循环

for迭代的差别

```
1 function sum(arr){
2   for (let x in arr){ // 遍历index或对象属性
3     console.log(x, typeof(x), arr[x]);
4   }
5   for (let x of arr){ // 遍历元素
6     console.log(x, typeof(x));
7   }
8   for (let x=0;x<arr.length;x++){ // 自己定义索引数值遍历
9     console.log(x, typeof(x), arr[x]);
10  }
11 }
12
13 sum([3,6,9]);
```