**流程控制**

1. 什么流程?

计算机代码执行的顺序。

1. 流程控制？

控制计算机代码执行的顺序。

1. 流程控制的分类：
   1. 顺序结构：
      1. 代码自上而下执行的过程。
   2. 分支/选择结构：

在不同的情况下执行的不同代码块。

* 1. 循环结构：

重复执行某段代码

顺序结构：

顺序结构就是js默认执行的顺序，代码自上而下不需要人为控制

分支/选择结构：

使用大括号包裹起类的称为代码块，内部可以放多条js语句

判断条件之能是布尔值

单项分支

If(判断条件)

一条js语句；

双项分支： if() else

If(判断条件)

一条js语句；

else

一条js语句；

多向分支：else if()

If(判断条件)

一条js语句；

else if(判断条件)

一条语句

else if(判断条件)

一条语句

else if(判断条件)

一条语句

else if(判断条件)

一条语句

else if(判断条件)

.......

switch(变量){

case:

js代码

case:

js代码

case:

js代码

case:

js代码

}

注意：

1. case的值必须是具体的数值而不能是一个范围，
2. case值可以是浮点数。
3. break 在代码中的作用就是，中断分支结构的执行，一旦break分支就会结束，不再执行其他内容。

循环结构：

console.log(); //向控制台输出

document.write() //向页面中输出

while 循环

格式：

声明一个增量初始值；

While（判断循环执行条件）{

循环体； // 就是需要循环的内容

[ 自增自减的条件；]

}

var i=0;

while( i<100 ) {

console.loh ( i );

i++;

}

向控制台中输出0~100；

例子：

循环一百以内的奇数：

var i=0;

while( i<100 ) {

if ( i%2 ==1 ) {

console.log ( i );

}

i++;

}

过滤一百以内不包含包含1的数：

var i = 0;

while ( i<100 ) {

if (! (( i%10 == 1 ) || ( i>=10 && i <20 )) ){

console.log( i )

}

i++;

}

从1 到 100 的累加：

var i = 0,s=0 ;

while( i<100 ){

s+= i;

i++;

}

Console.log( s );

用js向页面输出表格

document.write('<table width="800" border="1">');

document.write('<tr>');

document.write('<td>1</td>');

document.write('<td>1</td>');

document.write('<td>1</td>');

document.write('<td>1</td>');

document.write('<td>1</td>');

document.write('</tr>');

document.write('</table>');

第二种：

document.write('<table width="800" border="1" >');

//循环出5行

var tr= 0,i=0; //行的自增量

while (tr<6){ //行的自增区域

document.write('<tr>');//开始循环

//循环输出5列

var td=0;//列的自增量

while (td<5){//列的自增区域

//开始循环列

i++; //表示循环的次数

document.write('<td>'+i+'</td>');

//循环自增条件

td++;

}

document.write('</tr>');

tr++; //循环自增条件

}

document.write('</table>');

第三种：

document.write('<table width="800" border="1" >');

//循环输出5列

var td = 0;//列的自增量

while (td < 50) {//列的自增区域

//开始循环列

if (td % 10 == 0) { //判断在哪段加入tr的开始标签

document.write('<tr>');

console.log('<tr>')

}

console.log('<td>' + td + '</td>');

document.write('<td>' + td + '</td>');//循环的所有td

//循环自增条件

if (td % 10 == 9) {//判断f在哪段加入tr的结束标签

document.write('</tr>');

console.log('</tr>');

}

td++;

}

document.write('</table>');

do{}while() 循环

格式：

var i=0;

do{

循环体；

自增自减条件；

}while( 判断循环条件 )

while和do..while 的差别就是，前者先进行判断，而后者是先执行一次循环，然后再进行判断。

for（）{} 循环

格式：

for（初始化变量；循环的判断条件；自增或自减条件）{

循环体；

}

圆括号内的顺序不要乱，声明不要放外面

break；

在分支中： 在switch中，破坏分支选择；

在循环中： 终止，并且跳出循环；

continue：

在循环中：跳过本次循环，开始下一次循环，一般是用来过滤的

注意：continue一般放在循环体之前