使用JS实现 **入栈出栈**，要求使用2套方案

方案一：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>JS栈</title>

</head>

<body>

<script type="text/javascript">

function Stack(count){

var top=-1;//top头指针

this.myArray=new Array();

if(count!=undefined){

this.count=count;

this.myArray=new Array(this.count);

}else{

this.count=0;

}

//入栈

this.In=function(value){

if(top==this.count){

return false;

}else{

++top;

this.myArray[top]=value;

return true;

}

return false;

}

//出栈

this.Out=function(){

if(top==-1){

return false;

}else{

var removeValue=this.myArray[top];

this.myArray[top]=null;

top--;

return removeValue;

}

}

this.Clear=function(){

this.top=-1;

}

//遍历栈

this.tostring=function(){

for(var i=0;i<this.myArray.length;i++){

document.write(this.myArray[i]+'<br>');

}

}

}

Stack(3);

In(1);

In(2);

In(3);

tostring();//1 2 3

Out();

Out();

tostring();//1 null null

In(4);

tostring();//1 4 null

</script>

</body>

</html>

方案二：

//使用数组dataStore保存站内元素，构造函数将其初始化为一个空数组。

//变量top定义栈顶的位置，构造时初始化为0，表示栈的起始位置为0

function Stack(){

this.dataStore = [];

this.top = 0;

this.push = push;

this.pop = pop;

this.peek = peek;

this.clear = clear;

this.length = length;

//注意++操作符的位置，它放在this.top的后面，这样新入栈的元素就被放在top的当前位置对应的位置，同时top自加1，指向下一个位置

function push(element){

this.dataStore[this.top++] = element;

}

//返回栈顶元素，同时top的位置减1

function pop(){

return this.dataStore[--this.top];

}

//peek()方法返回数组的第top-1个位置的元素，即栈顶元素

function peek(){

return this.dataStore[this.top-1];

}

//将top的值设置0，即清空一个栈

function clear(){

this.top = 0;

}

//返回变量top的值即为栈内元素的个数

function length(){

return this.top;

}

}