一、细节记录

a、chrome浏览器console换行“Shift+Enter”；

二、数组

a、声明数组简洁方式：var arr = []; 若赋值则 var arr = [1,2,3];

b、斐波那契数列: f(n) = f(n-1)+f(n-2);

c、输出数组简洁方式：console.log(arrayName) arrayName 为数组名；

1、增删元素

a、JavaScript数组可以动态增加。

array[length ] = xx; length为自定义长度；

array.push(value); 把值添加到数组的末尾；

b、把数组直接增加到数组首位 array.unshift(value);

c、删除数组最后的元素 array.pop(); 删除第一个元素，array.shift();

2、数组方法参考

a、迭代

every()不断迭代，直到返回false；

some()不断迭代，直到返回true；

forEach()迭代所有；

三、栈

1、创建栈

function Stack(){}

2、选择数组保存栈中元素。

var items = [];

3、声明方法。

push(elements):添加元素到栈顶

this.push = function(elements){items.push(elements);}

pop()：移除栈顶元素，同时返回被移除元素。

this.pop = function(){return itmes.pop();}

peek():返回栈顶元素，不对栈进行修改。

this.peek = function(){return items[items.length-1];}

isEmpty():检查栈是否为空。

this.isEmpty = function(){return items.length==0;}

clear():清空栈中元素。

this.clear = function(){items = [];}

print():输出栈中元素到控制台。

this.print = function(){console.log(items.toString());}

5、例子

从十进制到二进制

function divideBy2(decNumber)

{

var numberContainer = new Stack();

var temp = "";

while(decNumber>0)

{

//把余数放入栈中

numberContainer.push(Math.floor(decNumber%2));

decNumber = Math.floor(decNumber/2);

}

//反向输出余数

while(!numberContainer.isEmpty())

{

temp += numberContainer.pop().toString();

}

return temp;

}

从十进制到任意进制

function divideByNumber(decNumber,number)

{

var numberContainer = new Stack();

var temp = "";

var digitals = "0123456789ABCDEF";

while(decNumber>0)

{

//把余数放入栈中

numberContainer.push(Math.floor(decNumber%number));

decNumber = Math.floor(decNumber/number);

}

//反向输出余数

while(!numberContainer.isEmpty())

{

temp += digitals[numberContainer.pop()];

}

return temp;

}

四、队列

1、创建队列

function Queue(){}

2、选择数组保存队列中的元素

var items = [];

3、声明方法

enqueue(elements):向队尾增加一个（多个）元素。

this.enqueue = function(elements){ items.push(elements);};

dequeue():移除队列中排在最前面的一项，并返回移除项。

this.dequeue = function(){return items.shift();};

front():返回队列中排在最前面的一项。

this.front = function(){return items[0];};

isEmpty()、size()和栈相同。

4、知识点（splice）

splice() 方法

splice() 方法向/从数组中添加/删除项目，然后返回被删除的项目。

**注释：**该方法会改变原始数组。

语法

arrayObject.splice(index,howmany,item1,.....,itemX)

a、增加但不删除

<script type="text/javascript">

var arr = new Array(6)

arr[0] = "George"

arr[1] = "John"

arr[2] = "Thomas"

arr[3] = "James"

arr[4] = "Adrew"

arr[5] = "Martin"

document.write(arr + "<br />")

arr.splice(2,0,"William")

document.write(arr + "<br />")

</script>

结果：

George,John,Thomas,James,Adrew,Martin

George,John,William,Thomas,James,Adrew,Martin

b、增加且删除

document.write(arr + "<br />")

arr.splice(2,1,"William")

document.write(arr)

结果：

George,John,Thomas,James,Adrew,Martin

George,John,William,James,Adrew,Martin

document.write(arr + "<br />")

arr.splice(2,3,"William")

document.write(arr)

结果：

George,John,Thomas,James,Adrew,Martin

George,John,William,Martin

删除一个 删除多个

五、链表

1、单链表

创建链表

function linkedList()

{

//节点

var Node = function(element)

{

this.element = element;

this.next = null;

};

var length = 0;

var head = null;

//方法

……

}

增加节点

追加

/\*--------向尾部增加节点--------\*/

this.append = function(element)

{

var node = new Node(element);

var current;

if(head==null)//链表第一个节点{head = node;}

else

{

current = head;

//循环链表，直到最后一项

while(current.next){current = current.next;}

current.next = node;

}

length++;//链表长度加1

};

按位置插入

/\*--------按位置插入节点--------\*/

this.insert = function(position,element)

{

//检查是否越界

if(position>-1&&position<length)

{

var node = new Node(element);

var current = head;

var previous;

var index = 0;

if(position==0)

{

node.next = head;

head = node;

}

else

{

while(index++<position)

{

previous = current;

current = current.next;

}

previous.next = node;

node.next = current;

}

length++;

return true;

}

else

{

return false;

}

};

移除节点

/\*--------按位置移除节点--------\*/

this.removeAt = function(position)

{

//检查是否越界

if(position>-1&&position<length)

{

var current = head;

var previous;

var index = 0;

//移除第一项

if(position==0)

{

head = current.next;

}

else

{

while(index<position)

{

previous = current;

current = current.next;

index++;

}

previous.next = current.next;

}

length--;

}

else

{

return null;

}

};

查看元素索引值

/\*--------查看节点元素的索引值--------\*/

this.indexOf = function(element)

{

var current = head;

var index = 0;

while(current)

{

if(current.element==element)return index;

else

{

current = current.next;

index++;

}

}

return -1;

};

按元素值移除节点

/\*--------按元素值移除元素--------\*/

this.remove = function(element)

{

var index = this.indexOf(element);

return this.removeAt(index);

};

打印链表

this.toString = function()

{

//把lickedList转化为字符串

var current = head;

var string = "";

while(current)

{

string += current.element + " ";

current = current.next;

}

return string;

};

六、集合

集合由一组无序且唯一（即不能重复）的项组成。