*// 自执行函数*

            (*function*(*a*){

                document.querySelectorAll('#app1 ul li')[0].innerHTML = a;

            })('利用括号将函数编程表达式，达到自执行A');

            (*function*(*a*){

                document.querySelectorAll('#app1 ul li')[1].innerHTML = '利用括号将函数编程表达式，达到自执行B';

            }());

            0+*function*(*a*){

                document.querySelectorAll('#app1 ul li')[2].innerHTML = a;

            }('利用0+将函数编程表达式，达到自执行');

            true&&*function*(*a*){

                document.querySelectorAll('#app1 ul li')[3].innerHTML = a;

            }('利用布尔值将函数编程表达式，达到自执行');

            0,*function*(*a*){

                document.querySelectorAll('#app1 ul li')[4].innerHTML = a;

            }('利用0,将函数编程表达式，达到自执行');

            ~*function*(*a*){

                document.querySelectorAll('#app1 ul li')[5].innerHTML = a;

            }('利用二进制运算符~将函数编程表达式，达到自执行');

            +*function*(*a*){

                document.querySelectorAll('#app1 ul li')[6].innerHTML = a;

            }('利用+将函数编程表达式，达到自执行');

            -*function*(*a*){

                document.querySelectorAll('#app1 ul li')[7].innerHTML = a;

            }('利用-将函数编程表达式，达到自执行');

            !*function*(*a*){

                document.querySelectorAll('#app1 ul li')[8].innerHTML = a;

            }('利用！将函数编程表达式，达到自执行');

            typeof *function*(*a*){

                document.querySelectorAll('#app1 ul li')[9].innerHTML = a;

            }('利用typeof将函数编程表达式，达到自执行');

            new *function*(*a*){

                document.querySelectorAll('#app1 ul li')[9].innerHTML = a;

            }('利用new将函数编程表达式，达到自执行');

*// 闭包闭包的实例*

*// 1*

*function* closure\_a(){

*function* closure\_b(*a*){

                    document.querySelectorAll('#app1 ul li')[10].innerHTML = a;

                }

                closure\_b('闭包的实例1');

            }

            closure\_a();

*// 2*

*function* closured1(){

*function* closured2(){

*function* closured3(*d3*){

                        document.querySelectorAll('#app1 ul li')[11].innerHTML = d3;

                    }

                    closured3('闭包的实例2');

                }

                closured2();

            }

            closured1();

*// 获取时间*

*var* Dte = new Date();

            document.querySelectorAll('#app1 ul li')[12].innerHTML =Dte;

        </script>

shift：删除原数组第一项，并返回删除元素的值；如果数组为空则返回undefined   
var a = [1,2,3,4,5];   
var b = a.shift(); //a：[2,3,4,5] b：1

unshift：将参数添加到原数组开头，并返回数组的长度   
var a = [1,2,3,4,5];   
var b = a.unshift(-2,-1); //a：[-2,-1,1,2,3,4,5] b：7   
注：在IE6.0下测试返回值总为undefined，FF2.0下测试返回值为7，所以这个方法的返回值不可靠，需要用返回值时可用splice代替本方法来使用。

pop：删除原数组最后一项，并返回删除元素的值；如果数组为空则返回undefined   
var a = [1,2,3,4,5];   
var b = a.pop(); //a：[1,2,3,4] b：5

push：将参数添加到原数组末尾，并返回数组的长度   
var a = [1,2,3,4,5];   
var b = a.push(6,7); //a：[1,2,3,4,5,6,7] b：7

concat：返回一个新数组，是将参数添加到原数组中构成的   
var a = [1,2,3,4,5];   
var b = a.concat(6,7); //a：[1,2,3,4,5] b：[1,2,3,4,5,6,7]

splice(start,deleteCount,val1,val2,...)：从start位置开始删除deleteCount项，并从该位置起插入val1,val2,...   
var a = [1,2,3,4,5];   
var b = a.splice(2,2,7,8,9); //a：[1,2,7,8,9,5] b：[3,4]   
var b = a.splice(0,1); //同shift   
a.splice(0,0,-2,-1); var b = a.length; //同unshift   
var b = a.splice(a.length-1,1); //同pop   
a.splice(a.length,0,6,7); var b = a.length; //同push

reverse：将数组反序   
var a = [1,2,3,4,5];   
var b = a.reverse(); //a：[5,4,3,2,1] b：[5,4,3,2,1]

sort(orderfunction)：按指定的参数对数组进行排序   
var a = [1,2,3,4,5];   
var b = a.sort(); //a：[1,2,3,4,5] b：[1,2,3,4,5]

slice(start,end)：返回从原数组中指定开始下标到结束下标之间的项组成的新数组   
var a = [1,2,3,4,5];   
var b = a.slice(2,5); //a：[1,2,3,4,5] b：[3,4,5]

join(separator)：将数组的元素组起一个字符串，以separator为分隔符，省略的话则用默认用逗号为分隔符   
var a = [1,2,3,4,5];   
var b = a.join("|"); //a：[1,2,3,4,5] b："1|2|3|4|5"