一**、属性、事件、添加事件、自定义函数**

1. 属性

获取元素

（1）静态方法ById(找一个,必须是html里有的，必须是document)

var oUl=document.getElementById(‘list’);依据id=’list’

（2）动态方法ByTagName(可以不是document)

➀一堆中的某一个：var oUl=document.getElementByTagName(‘ul’)[0];切记必须加中括号[]

➁找一堆元素的集合 var aLi=oUl.getElementByTagName(‘li’);(相当于一个数组)

属性的读操作

元素.属性名 如：oBtn.id oBtn.vlaue

元素.innerHTML 如：oP.innerHTML

属性的写操作

元素.属性名=新的值 如：oBtn.vlaue=”按钮”； oBtn.vlaue=oText.vlaue;

元素.innerHTML=新的值 如：oP.innerHTML=oText.vlaue;

2、事件（鼠标事件、键盘事件、系统事件、表单事件、自定义事件……）

onclick鼠标点击，ondbclick鼠标双击，onmouseover鼠标移入，onmouseout鼠标移出，onmousemove鼠标移动，onmouseup鼠标按键被松开，onmousedown鼠标按键被按下

onkeydown某个键盘的键被按下

onload某个页面或图象被加载完成

……

3、添加事件

元素.事件 如：oBtn.onclick

1. 自定义函数

Function abc(){

……}

(1)直接调用：abc();

(2)事件调用：oBtn.onclick=abc;

或 oBtn.onclick=function abc(){……}；

1. **for应用、this关键字、cssText行间样式**

1、for

例子一：

for (var i=0;i<aLi.length;i++) {}

每执行一次for循环，需要去计算一下aLi.length，因此出于性能方面的考虑，将代码更改成如下：

var len = aLi.length

for (var i=0;i<len;i++) {}

例子二：

为body添加3000个按钮

for (var i=0;i<3000;i++) {

document.body.innerHTML += '<input type="button" value="按钮" />'

}

每执行一次，需要去计算document.body.innerHTML的内容，找到后往里添加一个按钮，

出于性能方面的考虑，先用一字符串存起来，最后再添加到document.body.innerHTML中，修改如下：

var str = '';

for (var i=0;i<3000;i++) {

str += '<input type="button" value="按钮" />'

}

document.body.innerHTML ＝ str;

2、this关键字

（1）alert(this) = window.alert(this) //this=>window

（2） function fn1() {

this

}

fn1(); //this=>window

（3）oDiv.onclick = fn1; // this=>oDiv

（4）oDiv.onclick = function() {

this //this=>oDiv

}

（5） oDiv.onclick = function() {

fn1(); fn1()里的this=>window

}

（6）<div onclick=' this '></div> this=>div

<div onclick=' fn1() '></div> fn1()里的this=>window

总结：

1、直接调用函数，比如：fn1() ,this=>window

2、谁调用this就是指谁

1. cssText行间样式（不能存在”-”）

oDiv.style.width = '200px';

oDiv.style.marginleft = '200px';(magin-left中的“-”得去除)

oDiv.style.cssText ='width:200px;height:200px;'cssText是覆盖原先的行间样式

1. **数据类型、类型转换（JS书P23）**
2. 数据类型：number数字(NaN)、string字符串、boolean布尔值、函数类型、object对象（obj、[]、{}、null）、undefined未定义

typeof用来判断基本数据类型，instanceof判断引用数据类型

1. 强制转换Number()、parseInt()、parseFloat()

Number()用于任何数据类型：Boolean值、数字值、null值返回0、unfined返回NaN、字符串（只包含数字或十六进制，将其转换为对应的十进制；包含有效的浮点格式，如：“1.1”，将其转换为为对应的浮点数值；字符串为空，转换为0；除以上格式的字符，转换为NaN）、object。

parseInt()将字符串转换为数值：第一个是数字字符开始解析直到遇到非数字的字符停止，只返回数字部分；以0开头的数字字符作为八进制，返回对应的十进制；以0x开头的数字字符为十六进制，返回对应的十进制

parseFloat()将字符串转换为数值：与parseInt()区别一：字符串中的第一个字符串有效的，如:“22.34.5”转换为22.34；区别二：始终会忽略开头的0，所以十六进制格式的转化为0，只解析十进制。

1. **作用域**
2. 局部环境的变量（只能在此内部中使用）
3. 内部环境可以访问所有的外部环境，但外部的不能访问局部的任何变量和函数
4. **Return**
5. fn1();

function fn1(){

return function(){

alert(666);//呵呵哒}

}

fn1()=>alert(666);//呵呵哒

1. fn1(){

function fn1(){

return 100;

alert(666);

}

}

fn1()=>100(return后面的语句不会执行)

1. **定时器**
2. setInterval(函数，毫秒) =>重复执行

clearInterval(函数) =>遇到事件时，先要清除

1. setTimeout(函数，毫秒) =>等待毫秒数后，只执行一次

clearTimeout(函数)

1. **日期对象、时钟倒计时**

1、new Date() =>当前系统的时间对象

　　数字类型：getFullYear()//返回4位数，例如：2014年返回2014

　　 getUTCFullYear()//返回UTC的年份，也是四位数

　　 getMonth() //月+1；

　　 getDate() //日

　　 getDay() //星期，0为星期日，1为星期1，需要判断一下

　　 getHours() //时

　　 getMinutes()//分

　　 getSeconds()//秒

　　还需要对时分秒，不足10的需要前面补0，即：

　　function toTwo (n){

　　　　return n<10 ? '0'+n : ''+ n

}

2、倒计时new Date()

数字形式：new Date(2014,0,21,22,17,01); //0代表1月，如果希望是5月，需设置成4

字符串: new Date('January 21,2014 22:17:01')

时间转换公式

t = Math.floor((未来的时间-当前的时间)/1000) 秒（1秒=1000毫秒）

　　　　 1天＝1\*24\*60\*60秒=86400

天：Math.floor(t/86400)

时：Math.floor(t%86400/3600)

分：Math.floor(t%86400%3600/60)

秒：t%60

时间戳：getTime()

返回从1970年1月1日0点0分0秒0毫秒

八、**字符串**

1、var str = '2014年新春快乐哈'

（1）str.length =>10 字符串长度

（2） str.charAt(5) =>新 获取指定位置的字符

（3） str.charCodeAt(5) =>‘新’的Unicode是26032 获取指定位置的字符Unicode

0-9 48-57

A-Z 65-90

a-z 97-122

（4） String.fromCharCode(26032,26149,24555,20048) =>新春快乐 根据Unicode编码转换成文字

通过charCodeAt转换成编码与数字相减，再通过fromCharCode转换成文字，实现了一个简单的加密

2、位置

（1）indexOf 查找子字符串出现的位置和次数，从前往后

没有找到 -1

indexOf('m') 只有一个参数，从0开始找

indexOf('m',-30) 负数,从0开始找

（2）lastIndexOf 从后往前

lastIndexOf('m',-30) 负数，从0开始找

alert('1000'>2) true(数字比较)

alert('1000'>'2') false（字符串比较）

3、截取类

（1）substring

str.substring(5,9) => 新春快乐,从每5位开始到第8位

str.substring(9,5) => 前面参数>后面参数，颠倒位置

str.substring() => 未截取

str.substring(-3,5) => 2014年,负数当0处理

（2）slice

str.slice(9,5) => 空值,不交换位置

str.slice(-5,-1) => 新春快乐

4、大小写转换

str.toUpperCase() 大写

str.toLowerCase() 小写

5、与数组之间的转换

（1）split 分成好几段，变成数组对象

str.split() => 不分割，是数组

str.split('') => 一个字符，一个字符分割

str.split('',2) => 只取前面2段，后面的丢弃

（2）join 数组变成字符串

var arr = [ 'aa', 'bb', 'cc' ];

arr.join(); => 字符串 aa,bb,cc

arr.join(''); => 字符串 aabbcc

arr.join('-'); => 字符串 aa-bb-cc

1. **数组**

1、jason只能用for in遍历，不能用for，且没有length和下标

var json={'name':'abc',’age’:3,’fun’:’前端’} //属性加分号为安全的写法

for(var atter in jason){

alert(attr); //name //age //fun

alert(jason[attr])//abc //3 //前端

}

1. 数组

（1）长度

var arr1 = [14]; //长度为1

var arr2 = new Array(14); //长度为14

（2）添加

arr.push() 从尾部添加，返回值为数组新的长度

arr.unshift() 从头部添加,返回值为数组新的长度，IE6和IE7不支持返回值

（3）删除

arr.pop() 从尾部删除，返回值为删除的那个

arr.shift() 从头部删除，返回值为删除的那个

(4)**splice()** 删除、替换、添加，返回值为删除的数据

var arr1=[‘AB’,’CD’,’EF’]

删 arr.splice(0,2) //从第一个开始删两个，此次被删“AB”和“CD”

替 arr.splice(1,2,’前端’) //从第二个开始替换掉两个 变成[’AB’,’前端’]

添 arr.splice(0,0,’前端’) //第一个开始删除0个，在第一个前加上“前端”

（5）重排序

➀sort() 按照字符串的规则来排序

var a=[1,5,2,4,3];

a.sort(); //1,2,3,4,5

sort中的比较函数compare，来进行数字的排序，规则：正数交换位置，负数不交换，相等返回

var b=[1,5,10,15,0];

function compare(b1 ,b2){

if(b1<b2){

return 1; //从大到小排

} else if(b1>b2){

return -1;

}else{

return 0;

}

}

b.sort(commpare); //15,10,5,1,0

➁reverse()反转数组原来的顺序

1. 两数组相连用concat()

var c=arr1.concat(a); //AB,CD,EF,1,5,2,4,3