# 1.形状变换

<button>与<input>的区别

JS中变换css属性，div.style[width]=

用this.index代替遍历中的i，打破闭包的限制

不要忘记先使用window.onload

# 2.网页换肤

body{font: 12px/1.5 Tahoma;} //像素12px行高1.5

Zoom的使用方法：   
zoom :﻿ normal | number   
normal :　 默认值。使用对象的实际尺寸   
number :　 百分数 | 无符号浮点实数。浮点实数值为1.0或百分数为100%时相当于此属性的 normal 值用白话讲解就是zoom:后面的数字即放大的倍数，可以是数值，也可以是百分比。如：zoom:1，zoom:120%。而这个属性只要在IE中才起作用，所以很少用到它的实际用途，而最经常用到作用是清除浮动等，如：

\_zoom是CSS hack中专对IE6起作用的部分。IE6浏览器会执行zoom:1表示对象的缩放比例，但这里   
overflow:hidden;和\_zoom:1;是连起来用的，作用是清除border内部浮动。

text-indent 属性规定文本块中首行文本的缩进。 注释:允许使用负值。如果使用负值,那么首行会被缩进到左边

head里的link标签也是可以提取的

# 4.div循环变色

center标签的使用

for(var i in oDiv) oDiv[i].style.background="red" 可以实现效果，但是会提示错误

# 5.onmouseover,onmouseout

# 6

vertical-align:center 垂直居中

# 第二课

# 1.

this.value.replace(/[^(\d)|(,)]/,"")

var oInput = document.getElementsByTagName("input")[0].value.split(",");

for(var i in oInput){

sum += parseInt(oInput[i]);

}

4.弹出层

很多朋友问过我absolute与relative怎么区分，怎么用？我们都知道absolute是绝对定位，relative是相对定位，但是这个绝对与相对是什么意思呢？绝对是什么地方的绝对，相对又是相对于什么地方而言的呢？那他们又有什么样的特性，可以做出什么样的效果呢？关于两者之间又有什么样的技巧呢？下面我们就来一一解读。

Absolute，CSS中的写法是：position:absolute; 他的意思是绝对定位，他是参照浏览器的左上角，配合TOP、RIGHT、BOTTOM、LEFT(下面简称TRBL)进行定位，在没有设定TRBL，默认依据父级的做标原始点为原始点。如果设定TRBL并且父级没有设定position属性，那么当前的absolute则以浏览器左上角为原始点进行定位，位置将由TRBL决定。

一般来讲，网页居中的话用Absolute就容易出错，因为网页一直是随着分辨率的大小自动适应的，而Absolute则会以浏览器的左上角为原始点，不会应为分辨率的变化而变化位置。很多人出错就在于这点上出错。而网页居左其特性与Relative很相似，但是还是有本质的区别的。

Relative，CSS中的写法是：position:relative;  他的意思是绝对相对定位，他是参照父级的原始点为原始点，无父级则以BODY的原始点为原始点，配合TRBL进行定位，当父级内有padding等CSS属性时，当前级的原始点则参照父级内容区的原始点进行定位。

**父元素设置为 relative ,但是不设置TRBL.。子元素设置为absolute，那子元素就是相对于父元素的TRBL**

## [offsetLeft与style.left的区别](http://www.cnblogs.com/woshilee/articles/1951457.html)

offsetLeft 获取的是相对于父对象的左边距

left 获取或设置相对于 具有定位属性(position定义为relative)的父对象 的左边距

如果父div的position定义为relative,子div的position定义为absolute，那么子div的style.left的值是相对于父div的值，  
这同offsetLeft是相同的，区别在于：  
1. style.left 返回的是字符串，如28px，offsetLeft返回的是数值28，如果需要对取得的值进行计算，  
还用offsetLeft比较方便。  
2. style.left是读写的，offsetLeft是只读的，所以要改变div的位置，只能修改style.left。  
3. style.left的值需要事先定义，否则取到的值为空。而且必须要定义在html里，我做过试验，如果定义在  
css里，style.left的值仍然 为空，这就是我刚开始碰到的问题，总是取不到style.left的值。

offsetLeft则仍然能够取到，无需事先定义div的位置。

offsetParent：该属性返回一个对象的引用，这个对象是距离调用offsetParent的元素最近的（在包含层次中最靠近的），已进行过CSS定位的容器元素。 如果这个容器元素未进行CSS定位, 则offsetParent属性的取值为body元素的引用。 当容器元素的style.display 被设置为 "none"时（译注：IE和Opera除外），offsetParent属性 返回 null。

 top：该属性一般对用过css定位的元素有效（position为“static”时为auto，不产生效果），定义了一个top属性有效的元素（即定位元素）的上外边距边界与其包含块上边界之间的偏移。

clientTop：元素上边框的厚度，当没有指定边框厚底时，一般为0。

scrollTop：位于对象最顶端和窗口中可见内容的最顶端之间的距离，简单地说就是滚动后被隐藏的高度。

offsetTop：获取对象相对于由offsetParent属性指定的父坐标(css定位的元素或body元素)距离顶端的高度。

clientHeight：内容可视区域的高度，也就是说页面浏览器中可以看到内容的这个区域的高度，一般是最后一个工具条以下到状态栏以上的这个区域，与页面内容无关。

scrollHeight：IE、Opera 认为 scrollHeight 是网页内容实际高度，可以小于 clientHeight。FF 认为 scrollHeight 是网页内容高度，不过最小值是 clientHeight。

offsetHeight：获取对象相对于由offsetParent属性指定的父坐标(css定位的元素或body元素)的高度。IE、Opera 认为 offsetHeight = clientHeight + 滚动条 + 边框。FF 认为 offsetHeight 是网页内容实际高度，可以小于 clientHeight。offsetHeight在新版本的FF和IE中是一样的，表示网页的高度，与滚动条无关，chrome中不包括滚动条。

 \*诸如left、clientLeft、offsetLeft、clientWidth、scrollWidth等，和top、height类似，不再赘述。

============================分割线=====================================

clientX、clientY：相对于浏览器窗口可视区域的X，Y坐标（窗口坐标），可视区域不包括工具栏和滚动条。IE事件和标准事件都定义了这2个属性。

pageX、pageY：类似于event.clientX、event.clientY，但它们使用的是文档坐标而非窗口坐标。这2个属性不是标准属性，但得到了广泛支持。IE事件中没有这2个属性。

offsetX、offsetY：相对于事件源元素（target或srcElement）的X,Y坐标，只有IE事件有这2个属性，标准事件没有对应的属性。

screenX、screenY：相对于用户显示器屏幕左上角的X,Y坐标。标准事件和IE事件都定义了这2个属性

**1、offsetWidth width+padding+border)**

当前对象的宽度。

style.width也是当前对象的宽度(width+padding+border)。

区别：1)style.width返回值除了数字外还带有单位px；

           2)如对象的宽度设定值为百分比宽度,则无论页面变大还是变小，

              style.width都返回此百分比,而offsetWidth则返回在不同页面中对象的宽度值而不是百分比值；

           3)如果没有给 HTML 元素指定过 width样式，则 style.width 返回的是空字符串；

**2、offsetHeight :(Height+padding+border)**

当前对象的高度。

style.height也是当前对象的高度(height+padding+border)。

区别：1)style.height返回值除了数字外还带有单位px；

           2)如对象的高度设定值为百分比高度,则无论页面变高还是变矮，

              style.height都返回此百分比,而offsetHeight则返回在不同页面中对象的高度值而不是百分比值；

           3)如果没有给 HTML 元素指定过 height样式，则 style.height返回的是空字符串；

**3、offsetLeft :**

当前对象到其上级层左边的距离。

不能对其进行赋值.设置对象到其上级层左边的距离请用style.left属性。

style.left当前对象到其上级层左边的距离。

区别：1)style.left返回值除了数字外还带有单位px；

           2)如对象到其上级层左边的距离设定值为百分比，

              style.left返回此百分比,而offsetLeft则返回到其上级层左边的距离的值；

           3)如果没有给 HTML 元素指定过 left样式，则 style.left返回的是空字符串；

**4、offsetTop :**

当前对象到其上级层顶部边的距离。

不能对其进行赋值.设置对象到上级层顶部边的距离请用style.top属性。

**offsetTop就是当前对象border的外顶点到父元素border的内顶点。不带px。**

**style.top就是当前对象的margin的外顶点到父元素border的内顶点。 带px。**

**没有Margin的时候两者相等。**

style.top当前对象到其上级层顶部边的距离。

区别：1)style.top返回值除了数字外还带有单位px；

           2)如对象到其上级层顶部边的距离设定值为百分比，

              style.top返回此百分比,而offsetTop则返回到其上级顶部边的距离的值；

           3)如果没有给 HTML 元素指定过 top样式，则 style.top返回的是空字符串；

注意：如果上级层为body，由于IE、FF对padding、margin的解释不一样所以要明确规定处理不是下列的区别就不成立了。

IE   1)如果Div的上级层是body，而div与body之间有个div，如body->div->divo；divo的offsetTop=div的padding+margin+boder；

       2）如果Div的上级层是body，如body>divo；divo的offsetTop=div的padding+margin+boder；

             这divo的offsetTop=divo的margin >body.padding则为divo的margin，否则为body.padding谁大是谁？

FF  上述两种情况：offsetTop=margin+padding ;

(IE与FF中的body默认padding为10)在IE6.0 FF3.6.13

**5、scrollWidth:**获取对象的滚动宽度 。

**6、scrollHeight:** 获取对象的滚动高度。

**7、scrollLeft:**设置或获取位于对象左边界和对象中目前可见内容的最左端之间的距离(width+padding为一体)

**8、scrollTop:**设置或获取位于对象最顶端和对象中可见内容的最顶端之间的距离；(height+padding为一体)

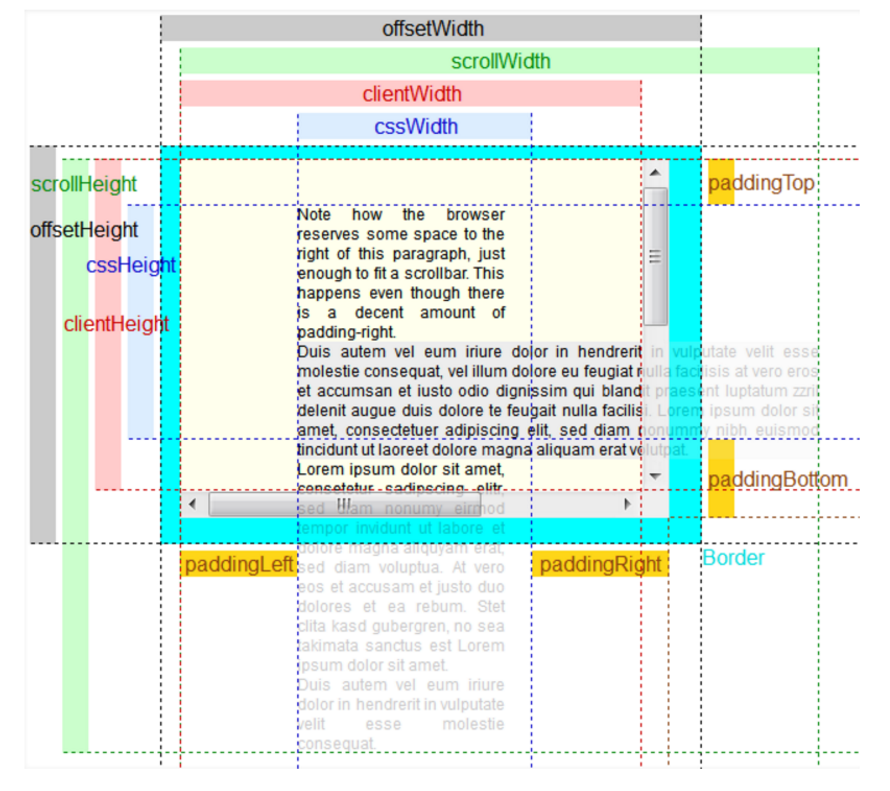
**9、clientWidth:** 获取对象可见内容的宽度，不包括滚动条，不包括边框；

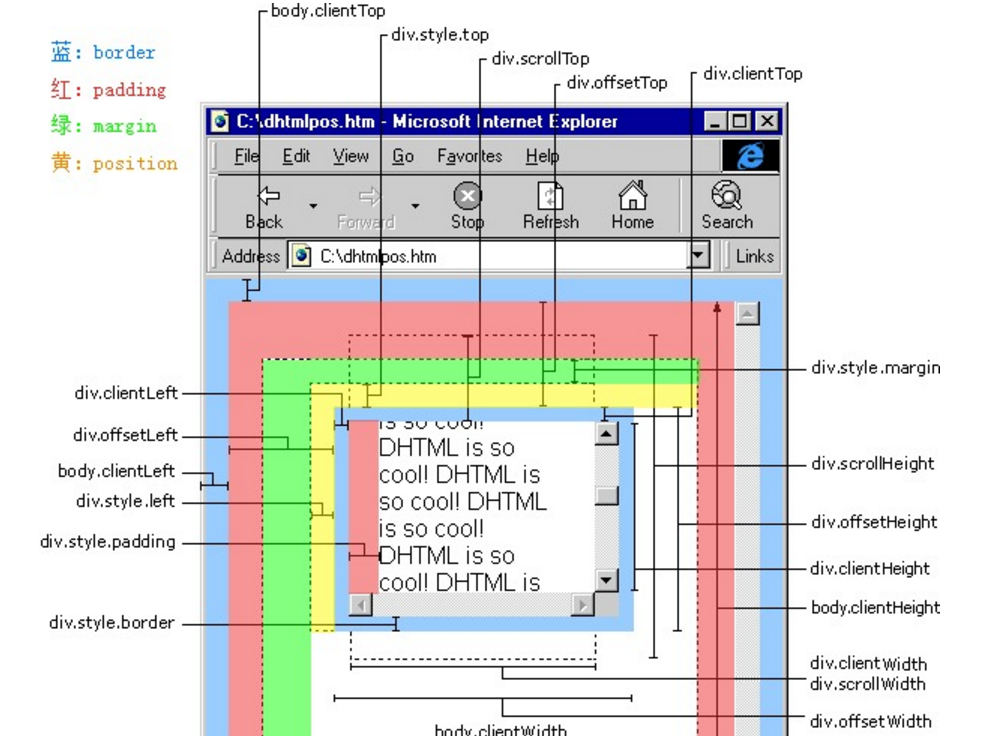
**10、clientHeight:** 获取对象可见内容的高度，不包括滚动条，不包括边框；

**11、clientLeft:** 获取对象的border宽度

**12、clientTop**：获取对象的border高度

**13、offsetParent :**当前对象的上级层对象.





# 5.函数传参

oDiv.removeAttribute("style");不能用style=””。

# 6.移入透明

padding顺序：上右下左顺时针。

opacity: 1;filter: alpha(opacity=100)为什么要一起写？ie低版本不支持opacity的  只支持filter  。为了兼容

# 7.简易选择卡

this.className

for(var n in oTab) oTab[n].className = "";

this.className = "current" //可以用for in

for(var x=0; x<oUl.length;x++) oUl[x].style.display = "none"; //不可以用 for in

# 10.提示框效果

CSS 伪类(Pseudo-classes)

CSS伪类是用来添加一些选择器的特殊效果。



使用[content](http://www.runoob.com/cssref/pr-gen-content.html) 属性来指定要插入的内容。

# 11.图片显示

在父元素上使用 overflow:hidden; zoom:1; 可以清除子元素浮动造成的体积影响。

可以插入伪元素 :after (content = “ ..”) 来实现同样的效果

要达实现加载过程中的过度动画效果。可以创建一个新的img对象。利用 img.complete? 来判断是否加载成功。

var img = new Image();

oDiv.style.display = "block";

img.src = oImg[0].src = this.src.replace(/small/,"big");

img.complete?oDiv.style.display = "none":(oImg[0].onload=function(){oDiv.style.display = "none"});

# 12.复选框

**不要忘记！！！！**

//根据每个复选框状态更新全选框状态。。。。

for(var i=1;i<oInput.length;i++){

oInput[i].onclick = function(){

isCheckAll();

}

}

# 第三课

1.typeof

2.parseInt

提取text文本框内容要用value属性。

首先，对限定文本框内容进行检查。利用正则表达式和value.replace()

然后，检查文本框是否为空。如果有空要提醒用户填写。

# 5.页面自计数

var updateNum = function(){

oBody.innerHTML = ++i;

}

updateNum(); //先调用一次，保证从1开始，前面没有空白时间

setInterval(updateNum,1000);

var 定义的函数，要在使用函数之前定义。

function updateNum (){

oBody.innerHTML = ++i;

}

function定义的函数可以放在函数使用后面

# 6.检查两位数

var x = /^\d{2}$/.test(parseInt(oInput.value));

x?alert("√ 是两位数"):alert("这是" + oInput.value.length + "位数");

正则表达式。

7.简易计算器

eval() 函数可计算某个字符串，并执行其中的的 JavaScript 代码。

语法

eval(string)

|  |  |
| --- | --- |
| **参数** | **描述** |
| string | 必需。要计算的字符串，其中含有要计算的 JavaScript 表达式或要执行的语句。 |

该方法只接受原始字符串作为参数，如果 string 参数不是原始字符串，那么该方法将不作任何改变地返回。因此请不要为 eval() 函数传递 String 对象来作为参数。

如果试图覆盖 eval 属性或把 eval() 方法赋予另一个属性，并通过该属性调用它，则 ECMAScript 实现允许抛出一个 EvalError 异常。

## 逻辑运算符 ||， && 的短路规则：

**|| 从左向右开始计算，当遇到为真的条件时停止计算，整个表达式为真；所有条件为假时表达式才为假。**

**&& 从左向右开始计算，当遇到为假的条件时停止计算，整个表达式为假；所有条件为真时表达式才为真。**

最后，我们做以下总结：  
对于“与”（**&&**） 运算： **x && y** 当**x**为**false**时，直接跳过，不执行**y**；相当于 if(x==true){y;}  
对于“或”（**||**） 运算 ：   **x||y** 当**x**为**true**时，直接跳过，不执行**y**。

if(x==false){y;}

onfocuse="this.blur()"   
onfocuse是聚焦的意思，当你把光标放在文本框上输入的时候，就是聚焦，但这里添加了"this.blur()"，blur的作用就是去除聚焦，也就是你不能把光标放在这个文本框上，换句话说就是你不能输入文本了 .

在网上看到有的人用onFocus="this.blur()"来消除虚线框，

## 定义和用法

NaN 属性是代表非数字值的特殊值。该属性用于指示某个值不是数字。可以把 Number 对象设置为该值，来指示其不是数字值。

**提示：**请使用 isNaN() 全局函数来判断一个值是否是 NaN 值。

定义和用法

slice() 方法可从已有的数组中返回选定的元素。

语法

arrayObject.slice(start,end)

参数 描述

start 必需。规定从何处开始选取。如果是负数，那么它规定从数组尾部开始算起的位置。也就是说，-1 指最后一个元素，-2 指倒数第二个元素，以此类推。

end 可选。规定从何处结束选取。该参数是数组片断结束处的数组下标。如果没有指定该参数，那么切分的数组包含从 start 到数组结束的所有元素。如果这个参数是负数，那么它规定的是从数组尾部开始算起的元素。

本身有逻辑漏洞

# 8.简易时钟

function format(a)   
{   
return a.toString().replace(/^(\d)$/, "0$1")   
}   
这个函数实现啥功能

在只有数字的一行前面加0.

# 9.倒计时器

this.className = this.className=="" ? "cancel" : "";

this.className=="cancel" ? (timer = setInterval(updateTime,1000),updateTime()) : (clearInterval(timer));

# 第四课

# 4.arguments

## 类数组对象：arguments

在js中在调用一个函数的时候，我们经常会给这个函数传递一些参数，js把传入到这个函数的全部参数存储在一个叫做arguments的东西里面。

在js中万物皆对象，甚至数组字符串函数都是对象。所以这个叫做arguments的东西也是个对象，而且是一个特殊的对象，它的属性名是按照传入参数的序列来的，第1个参数的属性名是’0’，第2个参数的属性名是’1’，以此类推，并且它还有length属性，存储的是当前传入函数参数的个数，很多时候我们把这种对象叫做类数组对象。类数组对象和数组都是对象这个妈生的，但是数组是大哥比类数组对象多了很多其他的玩具(方法)，类数组对象只是长得很像数组的弟弟而已。

慢着，刚刚不是说数组也是对象吗，现在这个类数组对象又是什么？ 没办法，js就是这么的灵活。这个类数组对象不仅存储给函数传入的参数，也具有一些其他的属性，等下会一一道来。

因为类数组对象和数组有很多的共性，所以我们经常可以用call方法，让类数组对象也使用的数组的一些方法，就是让这个弟弟去玩哥哥的玩具，比如……，还是不扯远了，这篇文章只是说什么是arguments，想知道更多关于对象如何借调数组方法的话，请参考这篇文章。

有了这个对象我们以后写函数的时候，就不用给所有的形参指定参数名，然后通过参数名的方式获取参数了，我们可以直接使用arguments对象来获取实参，这样是不是方便了很多呢。   
有些语言在我们给函数指定了参数名之后，当调用函数时，会判断当前传入的参数是否与函数定义的参数个数相等，不相等就会报错，但是灵活的js（不是我说，js是真的灵活）并不会验证传递给函数的参数个数是否等于函数定义的参数个数。所以为了装逼（代码的简洁度），我们使用arguments调用参数可以不混淆不同函数之间的参数名。另外为了装逼（代码的严整度），我们也能用arguments来判断当前传入参数的个数是否与我们需要的数量一致。

最后我们还可以看到arguments还有一个叫做callee的属性，这个属性是表示的是当前函数的一个引用，简单点说，这个属性里面存储的我们调用的这个函数的代码，实在无法理解的时候，又到了console.log大显身手的时候了。

function showcallee() {

var a = '这里是代码';

var b = '这是另一段代码';

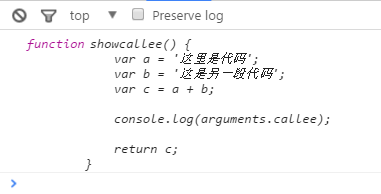
var c = a + b;

console.log(arguments.callee);

return c;

}

showcallee();



看到结果的你是不是和我一样惊呆了呢，这不就是我写的代码吗，arguments.callee完完整整的把这个函数的这段代码返回了。

## arguments的一些妙用

**1.利用arguments实现方法的重载**

下面我们利用arguments对象来实现一个参数相加的函数，不论传入多少参数都行，将传入的参数相加后返回。

function add() {

var len = arguments.length,

sum = 0;

for(;len--;len<0){

sum += arguments[len];

}

return sum;

}

console.log( add(1,2,3) ); //6

console.log( add(1,3) ); //4

console.log( add(1,2,3,5,6,2,7) ); //26

由于js是一种弱类型的语言，没有重载机制，当我们重写函数时，会将原来的函数直接覆盖，这里我们能利用arguments，来判断传入的实参类型与数量进行不同的操作，然后返回不同的数值。

**1.利用arguments.callee实现递归**

先来看看之前我们是怎么实现递归的，这是一个结算阶乘的函数

function factorial(num) {

if(num<=1) {

return 1;

}else {

return num \* factorial(num-1);

}

}

但是当这个函数变成了一个匿名函数时，我们就可以利用callee来递归这个函数。

function factorial(num) {

if(num<=1) {

return 1;

}else {

return num \* arguments.callee(num-1);

}

}

这个方法虽然好用，但是有一点值得注意，ECMAScript4中为了限制js的灵活度，让js变得严格，新增了严格模式，在严格模式中我们被禁止不使用var来直接声明一个全局变量，当然这不是重点，重点是arguments.callee这个属性也被禁止了。不过这都不是事儿，ES6为我们新增了很多好用的变量声明方式和新的语法糖，作为一个时髦的前端，我们赶紧学习一些ES6的新语法吧。

### 剩余参数, 默认参数 和 解构赋值参数

arguments 对象可以与剩余参数、默认参数和结构赋值参数结合使用。

function foo(...args) {

return args;

}

foo(1, 2, 3); // [1,2,3]

在严格模式下,剩余参数剩余参数、默认参数和结构赋值参数的存在不会改变 arguments 对象的行为,但是在非严格模式下就有所不同了。

当在非严格模式函数中 没有包含 剩余参数、默认参数或结构赋值参数,那么 arguments 对象将会追踪参数的变化,反之亦然。看下面的代码:

function func(a) {

arguments[0] = 99; // 更新了arguments[0] 同样更新了a

console.log(a);

}

func(10); // 99

并且

function func(a) {

a = 99; // 更新了a 同样更新了arguments[0]

console.log(arguments[0]);

}

func(10); // 99

当在非严格模式函数中 包含有 剩余参数、默认参数或结构赋值参数,那么 arguments 对象将不会追踪参数的变化,反之亦然。相反, arguments 反映了调用时提供的参数 :

function func(a = 55) {

arguments[0] = 99; // 提供了默认参数,更新 arguments[0] 不会更新 a

console.log(a);

}

func(10); // 10

并且

function func(a = 55) {

a = 99; // 提供了默认参数,更新 a 不会更新 arguments[0]

console.log(arguments[0]);

}

func(10); // 10

并且

function func(a = 55) {

console.log(arguments[0]);

}

func(); // undefined

//上面函数调用的时候没有传递参数,所有 arguments[0] 值为 undefined

# 7.数组

split() 字符串拆分成数组

join() 数组转变成字符串

shift() 删除第一项

pop() 删除最后一项

push() 在数组最后插入

reverse() 颠倒顺序

arrayObject.slice(start,end) 选择特定顺序的值，包含从 start 到 end （不包括该元素）  
arrayObject.sort(sortby) 排序

\s 任何空白字符

aArr.splice(0,3);删除从0开始，后面的3项，但是返回的是被删除的数组，会改变实际对象

aArr.splice(1,0,"orange","purple");第二个参数为0，不删除，是必须参数，从第2项开始插入



# 第六课

## 理解SetCapture、ReleaseCapture、GetCapture

**正常情况下，鼠标指针位于哪个窗口区域内，鼠标消息就自动发给哪个窗口。如果调用了SetCapture，之后无论鼠标的位置在哪，鼠标消息都发给指定的这个窗口，直到调用ReleaseCapture或者调用SetCapture设置另一个窗口为止。**  
很多时候，窗口或控件在鼠标按下后，需要知道鼠标移动和放开的情况，例如按钮在鼠标按下后会变为“按下”状态，鼠标移出按钮区域时“弹起”，所以按钮控件需要在鼠标按下时SetCapture，鼠标放开后ReleaseCapture。  
GetCapture只是检查一下当前是哪个窗口捕获了鼠标，通常不需要调用。

拖曳过程有一个更新参数的监听函数。要注意阻止冒泡函数的写法。

(event || window.event).cancelBubble = true;

图像的左侧和右侧均不允许出现浮动元素：

img

{

float:left;

clear:both;

}

正则表达式中 ？ 允许匹配一个或者0个前面的

//鼠标移出样式,active前面允许有一个空格或tab或回车

oThis.className = oThis.className.replace(/\s?active/,"")

//鼠标移入样式..一定要注意，这里有一个空格！！！多个className之间要有一个空格键

oThis.className += " active";

规定段落中的文本不进行换行：

p

{

white-space: nowrap

}

<http://www.zhangxinxu.com/wordpress/2012/05/getcomputedstyle-js-getpropertyvalue-currentstyle/>

# 6.2模仿微博

## 实例

规定可以由用户调整 div 元素的大小：

div

{

resize:both;

overflow:auto;

}

# 提取现在的style属性

 IE中使用的是obj.currentStyle方法，而FF是用的是getComputedStyle 方法  。“DOM2级样式”增强了document.defaultView，提供了getComputedStyle()方法。这个方法接受两个参数：要取得计算样式的元素和一个伪元素字符串（例如“:after”）。如果不需要伪元素信息，第二个参数可以是null。getComputerStyle()方法返回一个CSSStyleDeclaration对象，其中包含当前元素的所有计算的样式。以下面的HTML页面为例：

# 设置css格式函数理解

function css(obj,attr,value){

//检验给函数的参数个数

switch(arguments.length){

case 2:

//例如css(oLi, {"opacity" : "0", "height" : "0"});设置了对象obj,第二个对象类型是object,批量设置属性。style[i]=attr[i]。attr[opacity] = 0。

if(typeof argument[1] == "object"){

for(var i in attr) i == "opacity" ?(obj.style["filter"] = "alpha(opacity=" + attr[i] + ")") : obj.style[i] = attr[i];

//例如css(oLi, "height");第二个对象类型是string,读取属性

}else{

//IE用currentStyle方法，没有的话，用getComputedStyle方法

return obj.currentStyle ? obj.currentStyle[attr] : getComputedStyle(obj,null)[attr]

}

break;

//有三个参数，单一设置属性。

case 3:

attr == "opacity" ? (obj.style["filter"] = "alpha(opacity=" + value + ")",obj.style[attr] = value/100) : obj.style[attr] = value;

break;

}

};

# 重置表单

reset() 方法可把表单中的元素重置为它们的默认值。

charAt() 方法可返回指定位置的字符。

请注意，JavaScript 并没有一种有别于字符串类型的字符数据类型，所以返回的字符是长度为 1 的字符串。

### 语法

stringObject.charAt(index)

# 第7课

关于setInterval(function(){…},300);

如果有个函数不带参数，function next(){}.可以写成setInterval(next,300);

如果这个函数带参数，function next(a){}.一定要写成setInterval(function(){next(a)},300);

事情是这样的

function move(){} <=> move = function(){}

也就是说 move是个变量，存储的是函数本身

你把函数本身传给setInterval，他就会每隔100毫秒执行一次 这个函数

而加了括号以后，函数直接就运行了，所以你把move()运行后的值传给了setInterval,函数move又没有返回值，那setInterval每100毫秒执行一次“空”，所以啥都没干