float发胆的原意是用来解决文字环绕图片的，让图片脱离文档流，文字自动围绕图片排列，可以换行补齐（超过图片高度时)

不用float用inline-block会解析空格，在父级加font-size:0解决 或者父级letter-space 或者word-space。。。vertical-align:middle来解决inline-block自带竖起方向高度

box-sizing: border-box;

设置他以后，相当于以怪异模式(margin+width)解析，border和padding全会在你设置的宽度内部，比如手机端设置两行并且的布局，宽度各为50%,如果不用这个属性，设置border后右边的div会下来错位，设置这个属性，宽度还是50%而不是50%+\*px,两行可以并列显示.

[常见浏览器兼容性问题与解决方案（面试题目）](http://www.cnblogs.com/iceflorence/p/6646344.html)

1，浏览器兼容问题一：不同浏览器的标签默认的margin和padding不同

问题症状：随便写几个标签，不加样式控制的情况下，各自的margin 和padding差异较大。碰到频率:100%

解决方案：可以使用Normalize来清除默认样式

body,h1,h2,h3,ul,li,input,div,span,a,form …… { margin:0; padding:0; }

 2，浏览器兼容问题二：块属性标签float后，又有横行的margin情况下，在IE6显示margin比设置的大

问题症状:常见症状是IE6中后面的一块被顶到下一行

碰到频率：90%（稍微复杂点的页面都会碰到，float布局最常见的浏览器兼容问题）

解决方案：在float的标签样式控制中加入 display:inline;将其转化为行内属性

备注：我们最常用的就是div+CSS布局了，而div就是一个典型的块属性标签，横向布局的时候我们通常都是用div float实现的，横向的间距设置如果用margin实现，这就是一个必然会碰到的兼容性问题。

 3，兼容问题三：设置较小高度标签（一般小于10px），在IE6，IE7，遨游中高度超出自己设置高度

问题症状：IE6、7和遨游里这个标签的高度不受控制，超出自己设置的高度碰到频率：60%

解决方案：给超出高度的标签设置overflow:hidden;或者设置行高line-height 小于你设置的高度。

备注：这种情况一般出现在我们设置小圆角背景的标签里。出现这个问题的原因是IE8之前的浏览器都会给标签一个最小默认的行高的高度。即使你的标签是空的，这个标签的高度还是会达到默认的行高。

4，浏览器兼容问题四：行内属性标签，设置display:block后采用float布局，又有横行的margin的情况，IE6间距bug 问题症状：IE6里的间距比超过设置的间距 碰到几率：20%解决方案：在display:block;后面加入display:inline;display:table;

备注：行内属性标签，为了设置宽高，我们需要设置display:block;(除了input/img标签比较特殊)。在用float布局并有横向的margin后，在IE6下，他就具有了块属性float后的横向margin的bug。不过因为它本身就是行内属性标签，所以我们再加上display:inline的话，它的高宽就不可设了。这时候我们还需要在display:inline后面加入display:talbe。

5，浏览器兼容问题五：图片默认有间距

问题症状：几个img标签放在一起的时候，有些浏览器会有默认的间距，通配符清除间距也不起作用。

碰到几率：20% 解决方案：使用float属性为img布局

备注：因为img标签是行内属性标签，所以只要不超出容器宽度，img标签都会排在一行里，但是部分浏览器的img标签之间会有个间距。去掉这个间距使用float是正道。（也可使用负margin，虽然能解决，但负margin本身就是容易引起浏览器兼容问题的用法，所以尽量不要使用）

6，浏览器兼容问题六：标签最低高度设置min-height不兼容

问题症状：因为min-height本身就是一个不兼容的CSS属性，所以设置min-height时不能很好的被各个浏览器兼容 碰到几率：5% 解决方案：如果我们要设置一个标签的最小高度200px，需要进行的设置为：{min-height:200px; height:auto !important; height:200px; overflow:visible;}

备注：在B/S系统前端开发时，有很多情况下我们有这种需求。当内容小于一个值（如300px）时。容器的高度为300px；当内容高度大于这个值时，容器高度被撑高，而不是出现滚动条。这时候我们就会面临这个兼容性问题。

 7，浏览器兼容问题七：各种特殊样式的兼容，比如透明度、圆角、阴影等。特殊样式每个浏览器的代码区别很大，所以，只能现查资料通过给不同浏览器写不同的代码来解决。

 8，清除浮动：

[复制代码](javascript:void(0);)

.clearfix::after {

content: "";

display: table;

clear: both;

}

.clearfix {

\*zoom: 1;

}

[复制代码](javascript:void(0);)

  9，盒模型

Element {

box-sizing: border-box;

/\*box-sizing: content-box;\*/

}

/\* CSS hack\*/

我很少使用hacker的，可能是个人习惯吧，我不喜欢写的代码IE不兼容，然后用hack来解决。不过hacker还是非常好用的。使用hacker我可以把浏览器分为3类：IE6 ；IE7和遨游；其他（IE8 chrome ff safari opera等）

◆IE6认识的hacker 是下划线\_ 和星号 \*

◆IE7 遨游认识的hacker是星号 \*

比如这样一个CSS设置：

div { height: 300px; \*height: 200px; \_height:100px; }

IE6浏览器在读到height:300px的时候会认为高时300px；继续往下读，他也认识\*heihgt， 所以当IE6读到\*height:200px的时候会覆盖掉前一条的相冲突设置，认为高度是200px。继续往下读，IE6还认识\_height,所以他又会覆盖掉200px高的设置，把高度设置为100px；

IE7和遨游也是一样的从高度300px的设置往下读。当它们读到\*height200px的时候就停下了，因为它们不认识\_height。所以它们会把高度解析为200px，剩下的浏览器只认识第一个height:300px;所以他们会把高度解析为300px。因为优先级相同且想冲突的属性设置后一个会覆盖掉前一个，所以书写的次序是很重要的。

做兼容页面的方法是：每写一小段代码（布局中的一行或者一块）我们都要在不同的浏览器中看是否兼容，当然熟练到一定的程度就没这么麻烦了。建议经常会碰到兼容性问题的新手使用。很多兼容性问题都是因为浏览器对标签的默认属性解析不同造成的，只要我们稍加设置都能轻松地解决这些兼容问题。如果我们熟悉标签的默认属性的话，就能很好的理解为什么会出现兼容问题以及怎么去解决这些兼容问题。

实战是解决问题的最佳途径，也是遇到问题的唯一途径，大家多多亲自制作才能更快更好的成长，另外多去借鉴别人的经验也是进步的捷径。

clear:both 左右二边都不允许有浮动元素。规定元素的哪一侧不允许其他浮动元素。

.class hearder.class1

#id header#id1

div1,p 选择所有div1和p

div1 p 选择div1里的p元素

div>p选择父元素为div的所有P元素

div+p选择紧接在div元素之后 的p元素

[target]选择带有target属性所有元素

a：link未被访问 a:visited访问过的链接

a:active活动链接 a:havor鼠标指向时

&nbsp空格h

block继承你元素宽度，换成inline-block不会了，根据内容来定宽 inline用法



html样式和JS调用方法

<!DOCTYPE html>

<html lang="en-US">

<head>

<meta http-equiv="Content-Type" content="text/html;charset=utf-8">

<link rel="stylesheet" type="text/css" href="style11.css">引用css

<script type="text/javascript" src="js/jqmobile/jquery.mobile-1.4.5.min.js"></script>引用js

<title>Layout Test</title>

</head>

head里写添加js和css方法

<script type="text/javascript">

document.write("Hello World!")

</script>

<style type="text/css">

p

{

border: medium double rgb(250,0,255)

}

</style>

<body>

<a style="text-decoration:underline;" onclick="javascript:alert('dddd');" href="#">元素上定位CSS和js</a>

</body>

</html>

如果一个盒子300px宽 三个子盒子各是100px但有宽度，左浮动后你元素一行显示不下，要显示在一行上的话可以在三个子元素再套一个你元素306px就能显示下三个盒子。 然后300px顶级父盒子overflow：hidden 这样的话右边第三个盒子超出部分不显示，少一点点没所谓。

backgroun:rgba(0,0,0,0.3) 0.3是透明度，直接用opacity是盒子半透明，里面的内容也透明了，不好。

影响脱标的几点display postion float BFC是块级格式化上下文 block formatting context 大概意思是这块区域内随便怎么写不影响这块区域外其他元素的布局。比如 float不为none的浮动框;postion为absolute或者fixed;display为inline-block,table-cell,flex,,table-caption,inline-flex非块框的块容器;overflow不为visible的块框;

只有定位的盒子才有z-index 定位的对象默认层级是0

div里加一个a a要加宽度 宽度是继承父元素的，高度是不继承的。

行高要在字体大小后面才行 比如line-height：40px;font:400 14px "microsfot yahei"这样不行，要调换顺序

padding会不会撑开盒子要看有没有写width有就会撑开，没有就不会撑开。

子元素超出父部分虽然显示，但还是要加overflow:visible适应不同浏览器。

P里面不能放DIV A里面可以放很多包括DIV A里面不要放a input

input类表单比较特殊，二个表单元素之前有空间，如果要挨着，就用浮动或者定位。

#fff在css里支持，在行内样式里不支持要用#ffffff

div.aaa a:link{color:red;text-decoration:none;font-size:14px;}

a.bbb:link{color:red;text-decoration:none;font-size:14px;}

<div class="aaa"><a href="#">在a标签外的标签加class</a></div><br />

<a href="#" class="bbb">在a标签加</a>

div的话先要找到div再用div里的元素。。非div可以直接引用元素。a和div前缀可以省略

js的话

html引用js文件<script src="js文件路径"></script>

<script>f1();</script>调用函数

另外单独的js文件要写功能函数 function f1(){}

html标记用的是id的话，css用aaa#id,用class的话，用aaa.id 通用的话可以用class,特殊写规则的建议用id，方便代码 重构

thml注释 <!-- -->

<meta http-equiv="Content-Type" content="text/html;charset=utf-8"> 这个不设置容易乱码 charset="utf-8"

背景图片大小设置

width: 45px;

height: 40px;

background: url("./1.png") no-repeat; /\*background: url(./1.png) 0 -40 no-repeat;下面就不用了 0-40代表图片左上角相对于圆点0的坐标，长和宽取值就是上面的width和height\*/

background-position: 0 -40； url里一定要包含""或者''包着文件路径 ，不然ie不识别空格有些文件名包含空格就会出错

图片裁剪，要配合position: absolute;不然容易出问题

img{

position: absolute;

clip:rect(0 45px 40px 0);

}

权重分析

假使元素在未应用任何样式前它的权重值为0，那么，每条样式的权重值就是该样式所包含的所有选择器相对应的权重值之和：

元素，伪元素：+1 类，伪类，属性：+10 ID：+100 内联样式：+1000

！important !important允许开发人员强制应用某样式，他的用法是写在该样式的某属性值后，结束分号前，以便强制应用该样式，如：

img { width: 150px !important;}

居中完全方法

先设置绝对定位，再把上下左右值设为0，再把margin设置为outo，这样就不用去算值了。

也可以用绝对定位，left和top设置为center，再margin-left和margin-top元素半长和半宽

transition:时间2s 延迟1s 效果对一个元素设置后。

然后其他事件如hover时改变属性动画transform:retate(180deg) scale(1.5);来设置动画，就是动画执行上面设置的时间

为了避免重复性比如加一些多余的标签影响性能，可以考虑after before在这个元素内部元素的后和前分别加一些内容。比如三个旋转的广块。设置一个方法后，再在前后各加一个，旋转一下就可以形成五边形。

定位（考虑二个问题一个是占不占位置，另一个是相对于谁定位而已）和浮动

postion:absolute 绝对定位，不占位置，根据z-index来定层次。跟据父元素来定位，如果父元素没有定位就按BODY来。。相对定位占位置。通常先把你元素相对定位，再绝对定位子元素，精确定位。背景图方式如果背景图500\*500内容是300\*700可以把背景图绝对定位不占位置，再写内容（相对位置）就会占位置，互不影响，这样处理是个好方式，比如京东主体背景图示例。

float:left 元素浮动起来，不占位置，直到它的外边缘碰到包含框或另一个浮动框的边框为止，浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。但非块框就要当他们存在。

当一个父DIV里加了子DIV浮动的话，需要设置你元素的高度，要算精确，1.加div并设置为clear:both清除浮动，来使父DIV自动适应子DIV高度，就不用精确算你DIV高度了。2.父DIV设置overflow:hidden方法来使你DIV自动适应子DIV高度(这种坏处是有些定位会被遮挡)。3.伪元素加clearfix类 .clearfix:after{ content:'';visibility:hidden;height:0,clear:both;display:block;} .clearfix{\*zoom:1;} ie6才需要这个\*zoom 4.双伪元素.clearfix:before,.clearfix:after{display:table;content:''} .clearfix:after{clear:both;} .clearfix{\*zoom:1;}

;

margin和top

如果设置了盒子的宽和高,padding会将盒子撑大,如果没设置盒子的高和宽，而设置了盒子的父级标签的宽度,padding不会讲该盒子撑大，而只会压缩内容。margin不管父盒子有没有确定大小，margin不会撑大盒子，而会压缩本盒子的内容。

JS

js 原生JS文档加载完成

window.onload = function(){

//要执行的js代码段

}

引入jq后文档加载完成 以下二种可以

$(function(){

//要执行的js代码段

});

$(document).ready(function(){

//要执行的js代码段

});

bodyItem['0'].setAttribute('class', 'w1190');设置属性

display 隐藏不占位置 visibility:hidden隐藏占位置 。

在函数内部用return来设置返回值，一个函数只能有一个返回值。同时，终止代码的执行。

变量提升 如果在函数里定义了变量相当于 var 变量; 贬值操作具体看位置

onfocus onblur onclick 点击 onmouseover onmouseout

this指的是本身 This 主要是指事件的调用者 。

className 类名

innerHTML

更换 盒子里面的内容 文字 标签都换.

表单更换内容

Input.value

isNaN nan 不是一个数字 is 是 是 不是一个数字

isNaN(“12”) 如果里面的不是个数字 返回 true 否则返回false

法和属性的区别：

方法一律带有小括号 。 isNaN( ); 动词

方法给值： isNaN（“值”）;

Iphone.color = “red”; 属性给值一定是等号

Iphone.tel(); 方法

Txt.focus(); 方法 select() 选择功能方法

Onfocus 事件

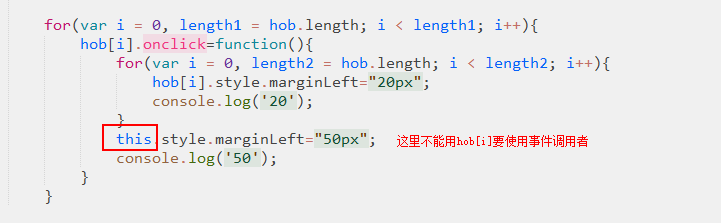
数组

var arr=[1,3,4,5,7];

var arr=new Arrray(1,4,6);

行内元素权重style.backgrount为1000 再加current类名100覆盖不了之前行内元素的属性 所以要给 .current的加权重 ！important

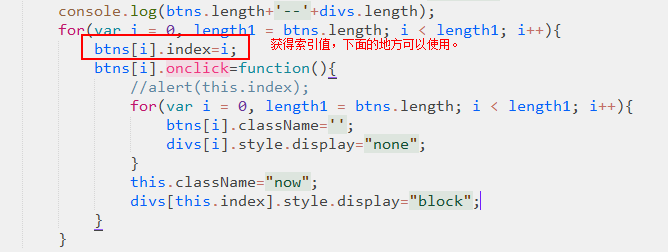




btn.onclick=fun();引用函数加了（）代表执行函数了。如果不执行则换成btn.onclick=function(){ fun();}这样来解决，点击了才执行函数。

input checkbox checked="checked" js checked=false/true;

获得索引值，值相对应的标签使用。for循环时使用。示例标签切换，相对应的标签内容也要对应标签索引值，这时使用。



数组操作 push尾部添加 unshift前面添加 pop尾部删除 shift首部删除 join数组转成字符串 split字条串转成数组 concat合并数组

DOM



封装自己的getclass类。因为ie8及以上不支持getElementsByClassName



child 和parent关系访问

var ul=document.getElementById("ul1");

console.log(ul.parentNode);//获得父亲元素 div

var bronext=ul.nextElementSibling||ul.nextSibling;//获得下一个兄弟元素。前者是普通浏览器（要写在前面），后者是IE678

console.log(bronext);

var bropre=ul.previousElementSibling||ul.previousSibling;//获得前一个兄弟元素。前者是普通浏览器，后者是IE678

console.log(bropre);

var firchi=ul.firstElementChild||ul.firstChild;//第一个孩子

console.log(firchi);

var laschi=ul.lastElementChild||ul.lastChild;//最后一个孩子

console.log(laschi);

console.log(ul.childNodes.length);//孩子节点，包括了换行.输出11 ，用下面方式解决nodeType==1

for(var i = 0, length1 = ul.childNodes.length; i < length1; i++){

if (ul.childNodes[i].nodeType==1) {

if (i%3==0) {

console.log(i);

ul.childNodes[i].style.backgroundColor="red";

}else {

ul.childNodes[i].style.backgroundColor="yellow";

}

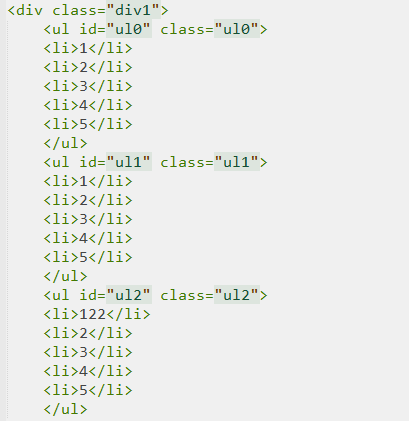
}

}

// 上面childnode包括了节点，用children不包括节点，不过高版本浏览器注释会算进去，处理访求是去掉注释，无伤大雅

ul2=ul.nextElementSibling||ul.nextSibling;//获得下一兄弟元素

console.log(ul2.children[0].innerHTML);//获得下一兄弟元素的第一个孩子的html内容//122



创建，添加，删除，克隆元素

var div10=$("div1");

var firdiv=document.createElement("div");//创建一个节点

div10.appendChild(firdiv);//添加到父元素末尾

var insdiv=document.createElement("div");

div10.insertBefore(insdiv, $("div11"));//参数，插入的元素，插入的位置

div10.removeChild($("div11"));

var div10clo=$("div10").cloneNode(true);

div10.insertBefore(div10clo, $("div10"))

获取节点属性getAttribute("属性名");

设置节点属性 setAttribute("属性名","属性值"); 删除属性 removeAttrbute("属性");

var date=new Date();//当前时间var date=new Date("2012/12/12 17:30:30");固定时间

date.getTime()//时间戳还有很多get时间相关的函数

间隔时间执行函数setInterval(fun名字, 时间间隔);

parseInt()取整

setTimeout(fun,1000)设置多少时间后执行函数，只执行一次

css3旋转

ele.style.webkitTransform(degree); google浏览器

ele.style.mozTransform(degree); google浏览器

ie

&& || a&&b 如果a为真则为b a为假则为a a||b a为真则为a,a为假则为b &&优先级大于||

a= 1&&3 //a=3 a= 1||3 //a=1 a=1||0&&3||0 为1先 0&&3为0 1||0为1 1||0 //1

console.log(txt.charAt(3));//该索引值所有的字符是哪个

console.log(txt.charCodeAt(3));//该索引值所有的字符的编码

console.log(txt.indexOf("好"));//第一个字符所在索引

console.log(txt.lastIndexOf("好"));//最后一个字符所有索引

console.log(1||2&&0||4&&3||0);//1

console.log(0||2&&0||4&&3||1);//3

console.log(arguments.length);//参数长度

console.log(arguments.callee);//函数所有内容打印出来

decodeURICompenet 解码

encodeURICompenet 加码

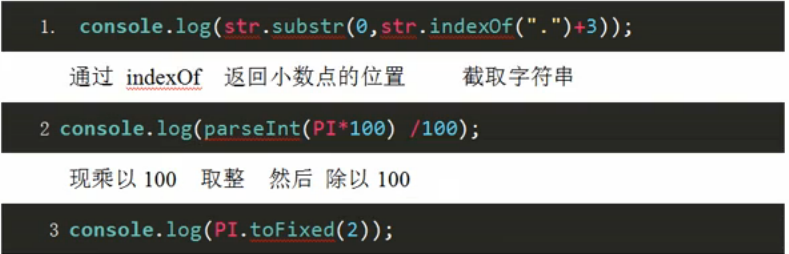
slice(start,[end]) 起始位置和结束位置索引值取字符串，end可选，支持负号，从末尾开始算

subStr(start,length)起始位置取某长度字符串，length可选，startie678不支持负号,其他高级浏览器支持，可以用length来代替。只取到末尾，比如从最后一个字符索引取3个，也只能取到最后一个字符。

var k=3.2323; //typeof number

k=k+""; //数字不能进去字条串操作，这样处理后就是字条串了typeof string

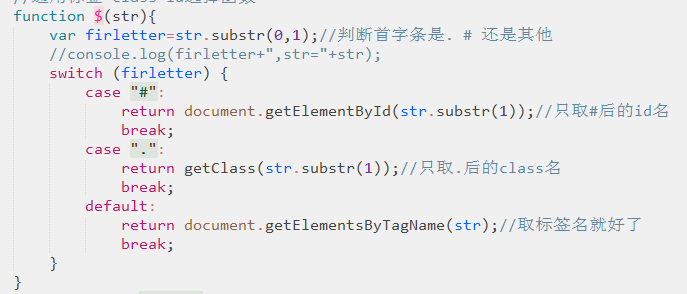
取二位小数三种方法 parseInt向下取整 aa.toFixed(2) aa要是数字型



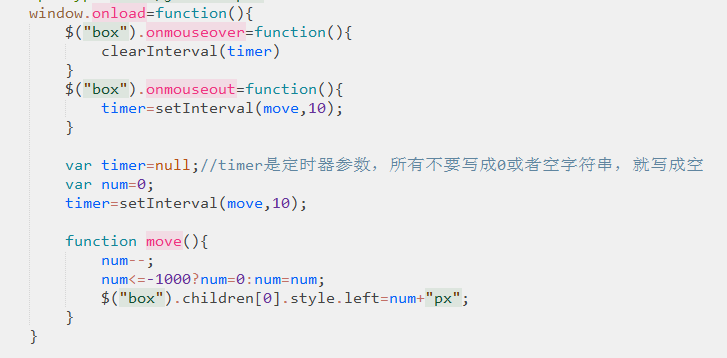
toUpperCase()转换成大写

vertical-align: top;对于图片会有下面多余3象素间隔，用这个方法去除掉,也可以用display：block来解决





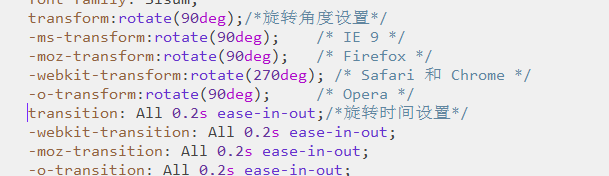
匀速移动图片 大盒子500px的话。图片250px 就把图片包在等于400%另一个大盒子里，让这个盒子移动，不是图片移动。用六张图片。第四张图片走完，第五张和第六张完全显示在的时候，就把left调为0肉眼看不出这种差别。



缓动动画公式

start = start +(target-start)/20; start是初始值 target是上限值 20步长决定速度,越大速度越小

旋转角度和时间设置



day8 offset

offsetWidth/offsetHeight得到对象的高度和宽度（自己的，与他人无关）=width+borde+padding不含margin....style.width只能获得写在行内式的高度样式。不能得到行外样式高度和宽度。

offsetLeft/offsetTop返回距离上级盒子（带定位）左边/上边的位置。如果父级没有定位，那就向上找定位的父级，直到找到定位的父级，不然就一直到body为止的距离。父级有定位就以最近的父级为准。从父级的padding开始算，不算父级的border。就是子盒子到定位父级边框到边框的距离。

同样是绝对定位的元素，后面的会压着前面的元素。如果要让前面元素背景显示，可以把后面的元素ul背景设置为透明(取不设置颜色)。如惊斗云背影与文字ul li效果.

offsetParent返回带有定位的父级（不一定是亲父亲），如果没有就一直到body.parentNode是返回亲父亲

offsetLeft/top与style.left/top值区别

1.offsetLeft可以返回没有定位的盒子，style.left只能返回带有定位的盒子的left

2.offsetLeft返回的是数字，style.left返回的是字符串带px

3.offsetLeft只读，style.top可读写

4.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

5.offsetLeft可以得到非行内式left，style.left只能得到行内式样式。

事件对象

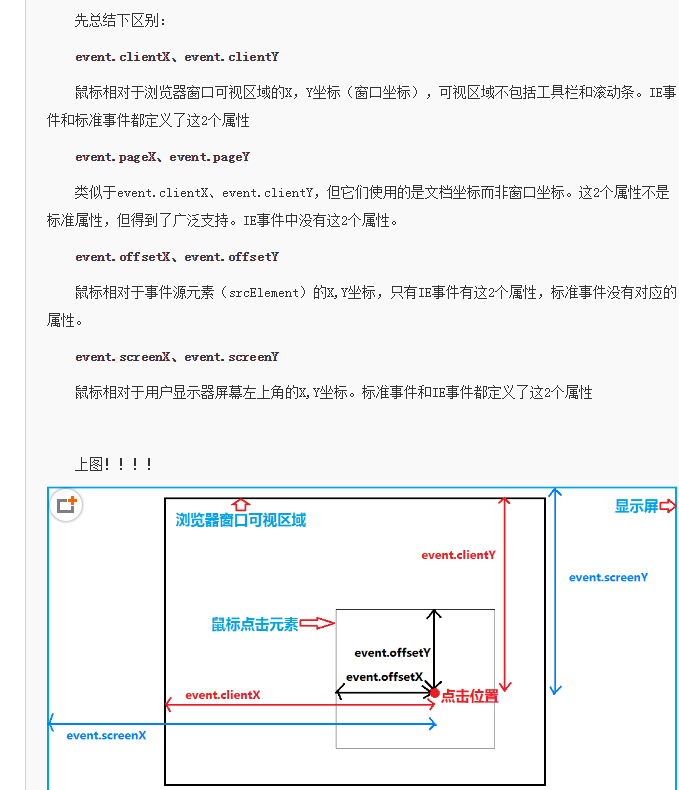
btn.onclick=function(event){} event是对象，即前面onclick事件的对象

clientX clientY对象相对于文档左和上的距离

ie678 window.event.clintX;

其他浏览器 event.clientx

兼容写法var event=event||window.event;

如果前面写了mousemove方法，鼠标松开时就用这样清楚mousemove方法，不然一直移动鼠标就会一直刷



window.innerWidth获得当前窗口内部宽度和高度

文档基本访问方法 document.title/head/body 没有document.html只有document.documentElement代替

scrollTop scrollLeft滚动轴

页面滚动事件window.onscroll=function(){ 滚动事件 }



所以要得到scrollTop值可以如下方法得到

var scrollTop=window.pageYOffset||docuement.documentElement.scrollTop||document.body.scrollTop||0;

兼容left top整理自己的scroll类

function scroll() {

if (window.pageYOffset !=null) {

return {

left:window.pageXOffset,

top:window.pageYOffset

}

} else if(document.compatMode=="CSS1Compat"){//声明了doctype的非怪异模式html文档

return {

left:document.documentElement.scrollLeft,

top:document.documentElement.scrollTop

}

}

//没有声明doctype的怪异模式html文档

return {

left:document.body.scrollLeft,

top:document.body.scrollTop

}

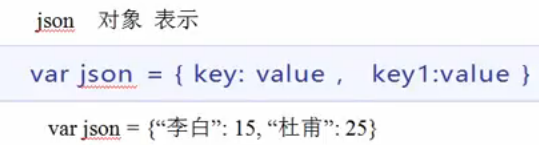
}

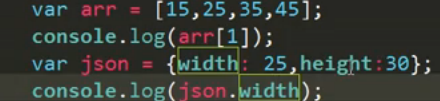
window.onscroll=function(){

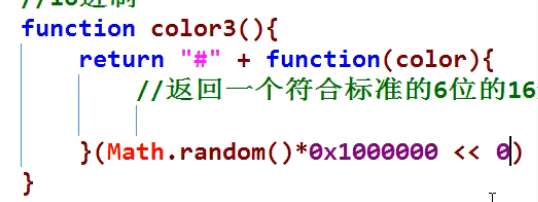
//alert('dd');

console.log("left:"+scroll().left+',top'+scroll().top);

}





匿名函数的执行,在函数后面加个()括号里可以带参数

window.onscroll=function(){}滚动鼠标 window.onresize=function(){}改变窗口大小

scrollTo(x,y) 去往滚动到页面中某个位置 window.scrollTo

offsetWidth border+padding+width

clientWidth padding+width

scrollwidth 内容部分大小scroll 不常用

window.screen.width 屏蔽的分辨率

冒泡机制

比如点孙子按钮，事件会在父DIV中触发，也会在爷DIV中触发。比如在btn加onclick事件，也在document上加onclick事件，如果点击空白处就只有document的onclick事件触发，但点击btn就会触发btn的onclick事件和document的onclick事件。

不冒泡的事件有blur focus load unload等

阻止冒泡兼容写法：

event.stopPropagation?event.stopPropagation():event.cancelBubble=true;

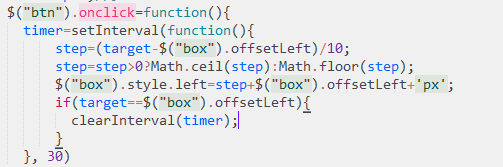
判断当前对象，判断鼠标点击的源目标的id是否是某个id值event.target.id==''||event.srcElement.id==""//ie678;

ar targetid=event.target?event.target.id:event.srcElement.id;

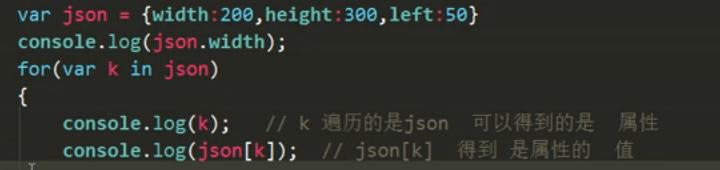
2、禁止默认事件  
//阻止浏览器的默认行为   
function stopDefault( e ) {   
    //阻止默认浏览器动作(W3C)   
    if ( e && e.preventDefault )   
        e.preventDefault();   
    //IE中阻止函数器默认动作的方式   
    else  
        window.event.returnValue = false;   
    return false;   
}

取整函数Math.round()四舍五入 Math.ceil()向上取整 Math.floor()向下取整

真正的缓动公式 之前用的那种不会停止 ，一直会执行



json遍历方法 for (变量in json){执行语句;｝in必须是字符串或者可以转换成字符串的数据烦死，json要是对象或者数组



zIndex backgroundColor js写css属性没有- 前面小写，后面不要-再加后面的首字母大写

/\* function fun(){console.log("ddd")};//定义一个函数、

var aaa=fun;//aaa赋值这个函数并并不执行

aaa();//加了()才是执行这个函数\*/

//闭包函数的使用，简化成 ,但不能直接这样定义函数，前面 要有类似click事件才可以这样使用。不然象下面,没有over事件就自动执行完了btn3亮了

/\* btns[i].onmouseover= function (i) {

return function(){

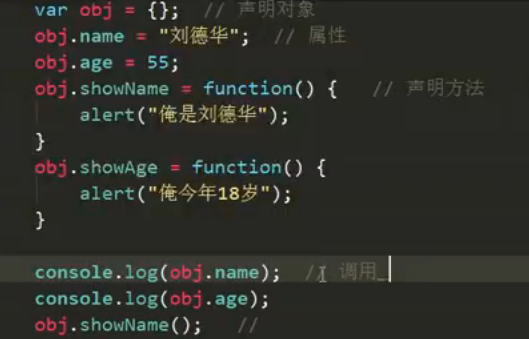
console.log('ddd'+i);

}

}(i);\*/

对象的声明和赋值和使用

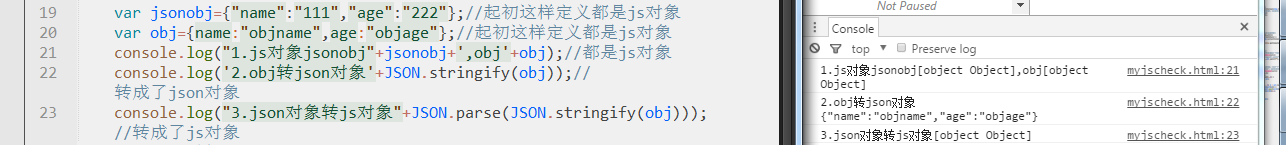
var ojb=new Object();推荐下面这种方式



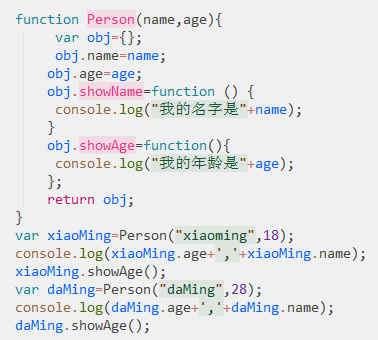
**JSON和JS对象区别对比表**

|  |  |  |
| --- | --- | --- |
| **区别** | **Json** | **Javascript对象** |
| 含义 | 仅仅是一种数据格式 | 表示类的实例 |
| 传输 | 可以跨平台数据传输，速度快 | 不能传输 |
| 表现 | 1,键值对方式，键必须加**双**引号  2,值不能是方法函数,不能是undefined/NaN | 1,键值对方式，键不加引号  2,值可以是函数、对象、字符串、数字、boolean 等 |
| 相互转换 | Json字符串转化为json：  1,JSON.parse(jsonstring); (不兼容ie7)  2,Jsobj=eval("("+jsonstring+")")；  (兼容所有浏览器，但不安全，会执行json里面的表达式?) | Js对象转换为Json：  JSON.stringify(jsobj);(不兼容ie7) |
| 其他 | 调用json官网的js，实现parse 和 stringify 在各个浏览器的兼容：  https://github.com/douglascrockford/JSON-js/blob/master/json2.js | |

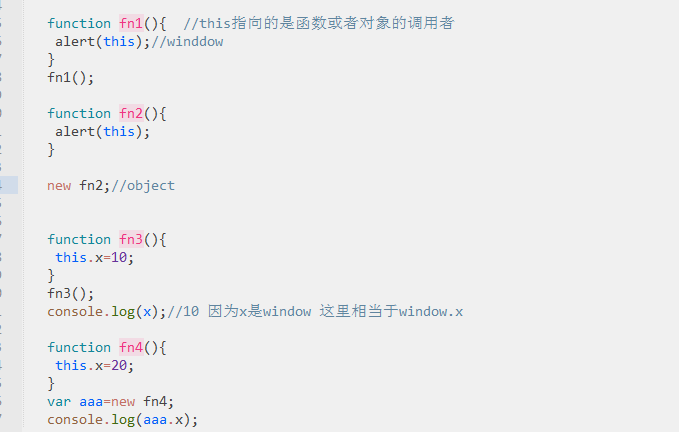
总而言之你可以理解为JSON是JS下的一种数据格式，他从属于JS，并且在处理JSON数据时可直接使用JS内置API

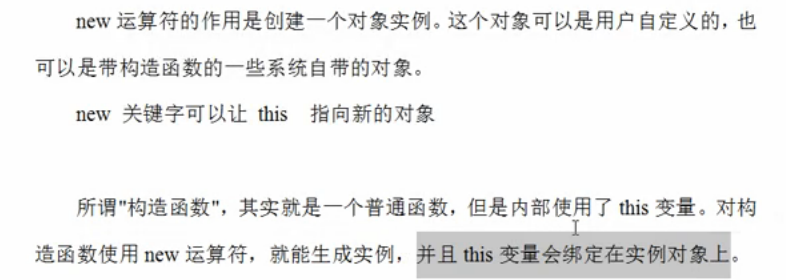






函数的this理解



构造函数

person前加New

console.log(xiaoMing.showName==daMing.showName);//false不相等。因为xiaoMing和daMing属性不同的对象实例，里面的this分别是xiaoMing和daMing

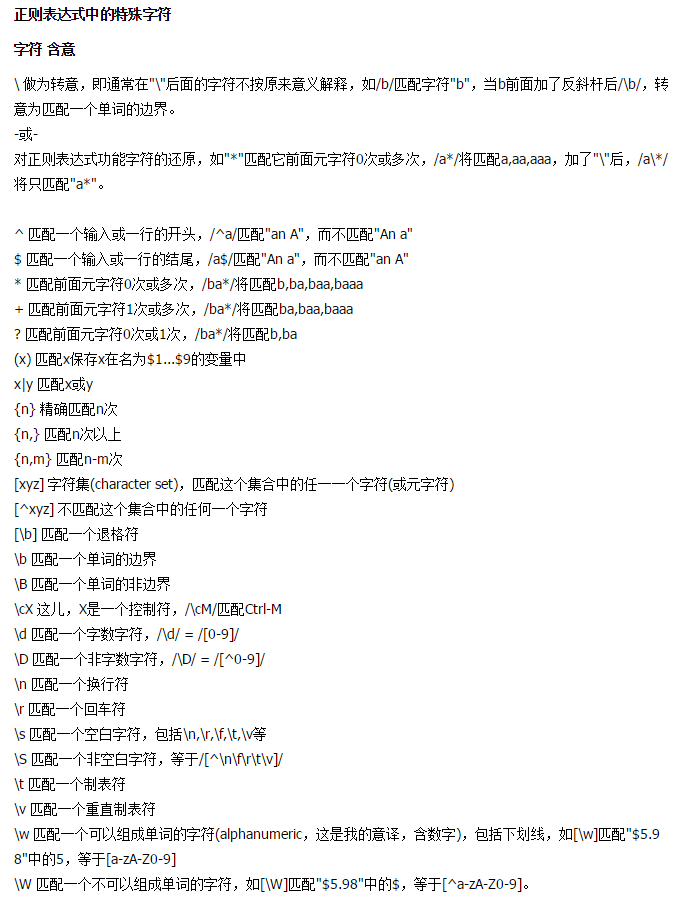


protytype属性的利用的好处如上

深入理解面向对象的二级菜单 方法

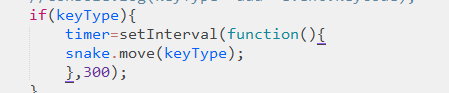


正则表达式规则

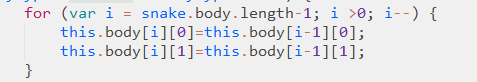


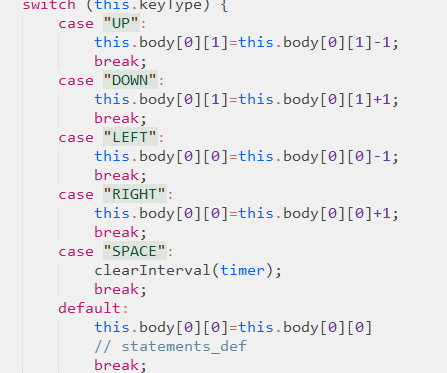
eval(3+9)计算表达式的值 match匹配 reg1=/^[/\*|\/].+/;//以\*|/开始 res.math(reg);

贪吃蛇,这里不能直接setInterval(snake.move(keyType),200);不然只执行一次



贪吃蛇除第一蛇头外，移动的时候都是后一个蛇身是前一个蛇身的坐标就可以了，蛇头根据上下左右来判断





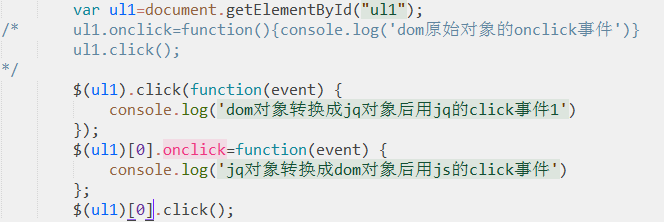
贪吃蛇多维数组的使用，要理解null的作用，null为空是没创建，不为空的时候只需要改坐标



dom对象与jq对象互相转换

jq对象转js对象 $("id")[0] 或者$("id").get(0);

js对象转jq对象 js原始对象ul=document.getElementById("ul") 转换成jq对象 ul=$(ul);

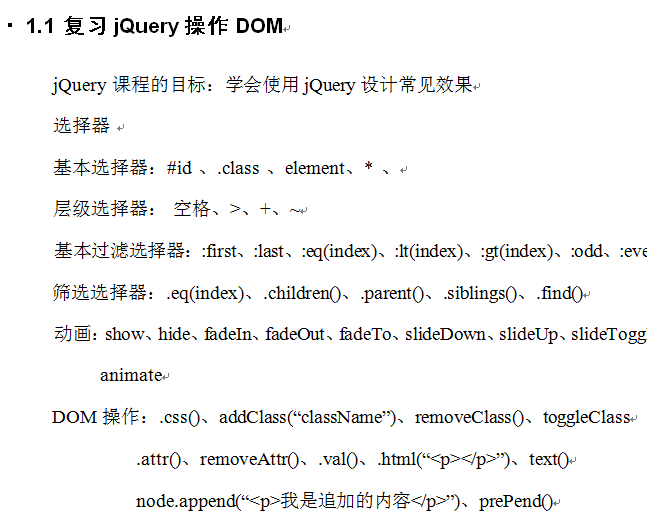


对象与创建对象

jq





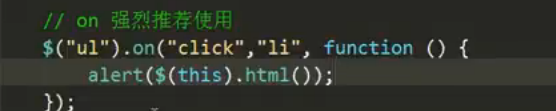


bind与one



下面第一种方式元素太多，每个元素要绑定一个事件，比较费时间，可以用第二种方式，在父元素上用delegate方式，还是推荐on方法





函数的理解:

// 测试1 i不属性函数，相当于全局变量，循环完后i为3停止循环了所以输入都是3 3 3

/\* arr=[]

for (var i = 0; i < 3; i++) {

arr[i]=function(){

console.log(i);

}

}

arr[1]();

arr[2]();

arr[0]();\*/

// 使用闭包来解决这个问题让输入分别是123

/\* arr=[];

for (var i = 0; i < 3; i++) {

arr[i]=(function(k){

return function(){

console.log(k);

}

})(i)

}

arr[0]();

arr[1]();

arr[2]();\*/

//test3 函数创建，属性以及原始链属性

function Test(name){

this.name=name;

this.show=function(){

console.log(this.name);

}

}

Test.brand="九阳";

console.log(Test.brand);//九阳 这样是可以的，这个函数的属性

Test.prototype.brand2="九阳2";

Test.prototype.brand3=function(){

console.log(this.name+'3333');

}

var test=new Test("aa");

console.log(test.brand);/\*undefined 因为test是构造函数,他的属性在创建new的时候就在函数内用this来指定了。外部的不属于New这个个体的属性了。\*/

test.show();//aa

console.log(test.brand2);//如果用Test.prototype.brand就可以，应该是这个函数的原型链属性了。查的资料：当一个函数被用作构造函数来创建实例时，该函数的prototype属性值将被作为原型赋值给所有对象实例（也就是设置实例的\_\_proto\_\_属性）

test.brand3();//aa3333

原生ajax方式

function ajax() {

//先声明一个异步请求对象

var xhr = null;

if (window.ActiveXObject) {//如果是IE

xhr = new ActiveXObject("Microsoft.XMLHTTP");

} else if (window.XMLHttpRequest) {

xhr = new XMLHttpRequest(); //实例化一个xmlHttpReg

}

//如果实例化成功,就调用open()方法,就开始准备向服务器发送请求

if (xhr != null) {

xhr.open("get", "reginAjaxdemo.php", true);

xhr.send(null);

xhr.onreadystatechange = doResult; //设置回调函数

}

//回调函数

//一旦readyState的值改变,将会调用这个函数,readyState=4表示完成相应

//设定函数doResult()

function doResult() {

if (xhr.readyState == 4) {//4代表执行完成

if (xhr.status == 200) {//200代表执行成功

//将xmlHttpReg.responseText的值赋给ID为resText的元素

document.getElementById("resText").innerHTML = xhr.responseText;

}

}

}

}

jquery ajax get post xml json 方式

//示例1 get html返回值形式

/\* $.ajax({

url:"jqueryAjax.php",

dataType:"html",

type:"get",

data:"aaa=1&bbb=2",

success:function(data){

$("#div1").append(data);

},

error:function(e){

console.log(e);

}

})\*/

/\* //示例2 get json返回值形式

$.ajax({

url:"jqueryAjax.php",

dataType:"json",

type:"get",

data:"aaa=1&bbb=2",

success:function(data){

$("#div1").html(data.username+'姓名，密码是:'+data.password);

},

error:function(e){

console.log(e);

}

})

})\*/

//示例3 post形式

/\* $.ajax({

url:"jqueryAjax.php",

dataType:"html",

type:"post",

data:"aaa=1&bbb=2",

success:function(data){

$("#div1").append(data);

},

error:function(e){

console.log(e);

}

})

})\*/

//示例4 json post形式

/\* $.ajax({

url:"jqueryAjax.php",

dataType:"json",

type:"post",

data:"aaa=1&bbb=22",

success:function(data){

$("#div1").append(data.username+'姓名，密码是:'+data.password);

},

error:function(e){

console.log(e);

}

})

})\*/

//示例5 get xml形式

/\* $.ajax({

url:"jqueryAjax.xml",

dataType:"xml",

// type:"get",

// data:"aaa=1&bbb=22",

success:function(data){

console.log(data);//返回的对象是文档xml格式

console.log($(data).children().children("title"));//文档转换成jq对象，再寻找或者循环里面的子元素

var book=data.getElementsByTagName('book');

$("#div2").append("<table><tr><th>"

+$(book).children("title").text()

+"</th><th>"+$(book).children("auther").text()

+"</th><th>"+$(book).children("price").text()

+"</th></tr></table>");

},

error:function(e){

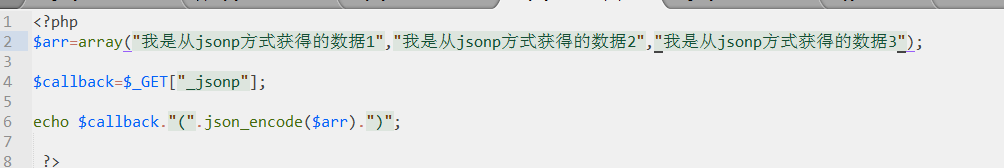
console.log(e);

}

})\*/

原始从第三方接口方式jsonp方式传递本地函数名及数据方式\_jsonp从而 获得远程php数据并调用本地方法进行解析





jquery ajax jsonp跨域请示数据

/\* $.ajax({

url:"day3ajaxjsonp.php",

type:"get",

dataType:"jsonp",

jsonp:"cb",//传递给请求处理程序或页面的，用以获得jsonp回调函数名的参数名(默认为:callback)

jsonpCallback:"cbfunction",//自定义的jsonp回调函数名称，默认为jQuery自动生成的随机函数名

success:function(json){

for(i=0;i<json.length;i++){

$("#div").append(json[i]+"<br>");

}

},

error:function(e){

console.log(e);

}

})\*/

数据库链接php



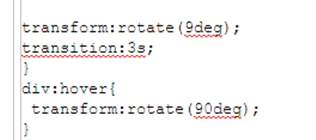
其他文件require("connect.php");写法可以再查一下

php数据处理



函数的调用要理解，箭头指向二种方式是相同的效果



第一行可以直接控制元素二维旋转9度，hover下旋转到90度，持续时间3秒

下面是自动执行动画，animation:动画名称 时间 循环次数

{

animation:mymove 5s infinite;

}

@keyframes用法@keyframes动画名｛ from ｛属性｝ to｛属性｝｝

@keyframes mymove

{

from {top:0px;}

to {top:200px;}

}

@keyframes mymove

{

0% {top:0px;}

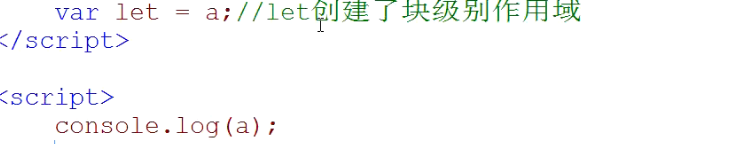
25% {top:200px;}

50% {top:100px;}

75% {top:200px;}

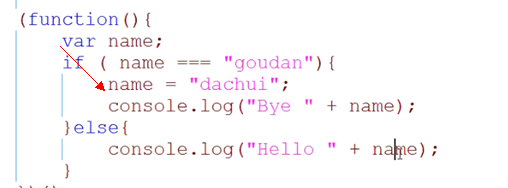
100% {top:0px;}

}



a is not defined is var a a is consoled

hello da chui 下面的var name="dachui"档当于 函数中先声明 var name .if里才是赋值。。函数找变量会先在函数内部查到是否有这个变量，有的话不会取外面的全局变量





timer eslapes test



封装简易框架

类似jquery