**04/12**

0 JavaScript学前预热课程-1

1. 126记住密码提示框

在JavaScript中，双引号之中只能用单引号，单引号之中只能用双引号。

1. 如果用JavaScript修改HTML元素的class属性时，由于class属于JavaScript元素的关键字，需在class单词之后加上Name。

例如：document.getElementById('pid').className=''

1. 网页换肤：准备几套CSS样式，点击按钮，改<link>标签的href属性值。
2. 在书写CSS样式时，id选择器的优先级高于class选择器。
3. 用JavaScript更改HTML元素的属性值或者元素样式的属性值时，属性值需要用引号包裹。（在javaScript中，string数据类型需要用引号包裹）。

0 JavaScript学前预热课程-2

1. 在实际工作之中，JavaScript代码一般不通过HTML行间事件添加，而是采用DOM0级事件处理程序或者DOM2级事件处理程序，实现行为、样式、结构三者分离。
2. Window.onload事件：在网页加载完成之后执行一个功能。

<script>

window.onload=function (){

alert("a");

};

</script>

注意：function(){}为一个匿名函数。

1. 行为(JavaScript)，样式(CSS)，结构(HTML)三者分离的写法

例子：

<script>

window.onload=function(){

var div=document.getElementById("div");

div.onmouseover=getBlack;//这里不能加()。

div.onmouseout=getRed;//这里不能加()。

}

//以上具代码一定要放在window.onload=function(){}函数中，即在页面加载完成之后执行该函数，否则按照从上到下的文档执行顺序，在执行var div=document.getElementById("div")此行代码时，页面中还没有id为“div”的标签，变量div为undefined。

function getBlack(){

var div=document.getElementById("div");

div.style.width='200px';

div.style.height='200px';

div.style.backgroundColor='black';

}

function getRed(){

var div=document.getElementById("div");

div.style.width='100px';

div.style.height='100px';

div.style.backgroundColor='red';

}

</script>

</head>

<body>

<div id="div"></div>

</body>

注意：事件函数有两种写法，一种是事件等于完整的函数，一种是事件等于函数名无括号。即：括号与函数体不可分离。（与句柄相同）

例子：<button id="btn">按钮</button>

<script>

var btn=document.getElementById("btn");

btn.onclick=function(){ //第一种写法，有括号有函数体。

alert(123);

};

btn.onclick=demo //第二种写法，无括号无函数体。

function demo(){

alert(123);

};

</script>

1. 收缩展示菜单

例子： <script>

window.onload=function (){

var p=document.getElementById("pid");

var u=document.getElementById("uid");

p.onclick=function (){

if(u.style.display=="block"){

u.style.display="none";

}else{

u.style.display="block";

};

};

};

</script>

<p id="pid">输入法</p>

<ul id="uid">

<li>手写</li>

<li>拼音</li>

<li>其他</li>

</ul>

错误例子：<script>

window.onload=function (){

var p=document.getElementById("pid");

var u=document.getElementById("uid");

if(u.style.display=="block"){

p.onclick=function (){

u.style.display="none"}

}else{

p.onclick=function (){

u.style.display="block"}

}

}

</script>

<p id="pid">输入法</p>

<ul id="uid">

<li>手写</li>

<li>拼音</li>

<li>其他</li>

</ul>

注释:if...else条件判断语句只执行一次，函数也只能绑定一个，故无法实现切换效果。

**04/14**

1. 复选框全选效果

问题：①如何定位复选框？②如何使用JavaScript控制复选框的选中与否？

For循环的组成部分：A初始化（只执行一次）、B条件、C程序语句、D自增。执行顺序：A→B→C→D→B→C→D→B→C→D→B。即先执行程序语句再自增。

for**循环**可以实现全选的功能。

1. this表示当前发生事件的那个元素。**（当前方法所属于的对象）**

例子：

<script>

window.onload=function(){

var abtn=document.getElementsByTagName("button")

for (n=0;n<abtn.length;n++) {

abtn[n].onclick=function(){

alert(this.value)

//alert(this.getAttribute("value"))定位属性值的另一方法，不常用

};

};

};

</script>

</head>

<body>

<button value="妙味课堂">妙味课堂</button>

<button value="妙味茶馆">妙味茶馆</button>

<button value="妙味视频">妙味视频</button>

</body>

0 JavaScript学前预热课程-3

1. **选项卡**

所有div的display都为none，让当前div 的display为block。

<style>

button{background-color: white;}

.active{background-color: yellow;}

div{width: 150px;height: 150px;background-color: #555;display: none;}

</style>

<script>

window.onload=function(){

var abtn=document.getElementsByTagName("button");

var adiv=document.getElementsByTagName("div");

for (n=0;n<abtn.length;n++) {

abtn[n].index=n; //利用循环给每个btn自定义一个索引值属性，属性值等于下标值，index属性名可换为其他名字。

abtn[n].onclick=function(){

for (m=0;m<abtn.length;m++) {

abtn[m].className="" //此处是一个循环而不是一个函数，应写abtn[m]，而不是this。

adiv[m].style.display="none"//为了简便，可写在一个循环里。}

this.className="active";

**adiv[this.index]**.style.display="block"//将当前按钮的序号为adiv的顺序。

};

};

};

</script>

</head>

<body>

<button class="active">按钮1</button>

<button>按钮2</button>

<button>按钮3</button>

<div style="display: block;">111</div>

<div>222</div>

<div>333</div>

</body>

8. 简易日历

<script>

var arr=["一月活动","二月活动","三月活动","四月活动","五月活动","六月活动","七月活动","八月活动","九月活动","十月活动","十一月活动","十二月活动"]

var ali=document.getElementsByTagName('li');

var div=document.getElementById("div");

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

ali[i].index=i //利用循环给每个li自定义一个索引值属性，属性值等于下标值，index属性名可换为其他名字。

ali[i].onmouseover=function(){

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

ali[i].className=''

};

this.className="active"

div.innerHTML="<h1>"+"第"+(this.index+1)+"月活动</h1><br/><p>"+arr[this.index]+"</p>"

//设置div的innerHTML为当前li的索引值。innerHTML识别字符串内的标签。

};

};

</script>

注意：Object对象数据类型**可以自定义属性**：数组new Array、函数function、类Array（等价于构造函数）、元素。基本数据类型**不可以自定义属性**：undefined、boolean、number、string。例外情况：null为Object数据类型但不可以自定义属性。

var t1=10;——t1是一个Number数据类型，不是一个对象。

var t2=new Number(10);——此时t2是Object数据类型，是一个对象。

var t3=“abc”;——t3是一个String数据类型，不是一个对象。

var t4=new String(“abc”);——此时t4是Object数据类型，是一个对象。

即：一切事物皆对象，但不是一切对象都可以**自定义**属性和方法。

如果给对象的原型添加属性和方法，该对象一定可以调用该属性和方法。包括：number、string、undefined、boolean等。

9. 定时器

var timer=setInterval(show,1000)

//这样写的话，timer为局部变量，定时器无法停止。

clearInterval(timer);

timer=setInterval(show,1000)//正确写法。

clearInterval(timer);

var oDate=new Date()

注：new是用来创建一个新的对象。

例子：

<body style="background: blueviolet;font-size: 50px;color: gold;text-align: center;">

<img src="img/0.png" />

<img src="img/0.png" />

点

<img src="img/0.png" />

<img src="img/0.png" />

分

<img src="img/0.png" />

<img src="img/0.png" />

秒

<script>

var imgs=document.getElementsByTagName("img");

function upDateTime(){

var oDate=new Date();

function toDouble(num){

if(num<10){

return '0'+num;

}else{

return ''+num;//这一句不可省略。否则当num大于等于10时，函数返回值为undefined。

};

};

var str=toDouble(oDate.getHours())+toDouble(oDate.getMinutes())+toDouble(oDate.getSeconds());

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

imgs[i].src='img/'+str.charAt(i)+'.png' //str.charAt(i)获得字符串中下标值为i的字符。

};

};

upDateTime();

setInterval(upDateTime,1000)

</script>

</body>

注意：①数字相加转化为字符串相加的方法——起始位置加空字符串””。②当函数不设置返回值时，函数的返回值为undefined。③计时器setInterval(函数名,1000)有两个含义，第一次执行此函数为页面加载一秒后，此后每间隔一秒执行一次此函数。

例如： <script>

function demo(a){

}

alert(demo(11)) //弹出undefined

</script>

10. <input id="ipt" type="text" />文本框内的文字既不是文本框的innerHTML，又不是文本框的innerTEXT，而是文本框的value属性值。

<input type="button" value="按钮" /> //这种写法中，按钮上的文字为按钮的value属性值。

<button value="anniu">按钮</button> //这种写法中，按钮上的文字为按钮的inerHTML值，即使设置value值也不发生作用。

0 JavaScript学前预热课程-4

1. 延时提示框

例题：

<style>

#div1{

width: 250px;

height: 50px;

background-color: burlywood;

}

#div2{

width: 200px;

height: 50px;

background-color:crimson;

margin-top: 20px;

display:none;

}

</style>

</head>

<body>

<div id="div1"></div>

<div id="div2"></div>

<script>

var odiv1=document.getElementById("div1")

var odiv2=document.getElementById("div2")

odiv1.onmouseover=function(){

odiv2.style.display="block";

clearTimeout(timer2);

//关闭计时器timer2，解决了鼠标从div2回到div1时，div2隐藏的问题。

}

odiv1.onmouseout=function(){

timer1=setTimeout(function(){

//设置计时器解决了鼠标无法从div1移动到div2上的问题。

odiv2.style.display="none";

},300)}

odiv2.onmouseover=function(){

// odiv2.style.display="block";

//这里的odiv2可换为this。

clearTimeout(timer1)

//完成当鼠标放入div2时，div2不隐藏有两步操作：①设置div的display为block。②关闭计时器timer1。第①步可以省略。

}

odiv2.onmouseout=function(){

timer2=setTimeout(function(){

odiv2.style.display="none";

},300)

}

</script>

</body>

代码简化：

<div id="div1"></div>

<div id="div2"></div>

<script>

var odiv1=document.getElementById("div1")

var odiv2=document.getElementById("div2")

function show(){

odiv2.style.display="block";

clearTimeout(timer);

}

function hide(){

timer=setTimeout(function(){

odiv2.style.display="none";

},300)}

odiv1.onmouseover=show;

odiv2.onmouseover=show;

odiv1.onmouseout=hide;

odiv2.onmouseout=hide;

</script>

注意：鼠标移入函数：①display="block"②取消计时器。鼠标移出函数：计时器中display="none"。

二次简化：

<div id="div1"></div>

<div id="div2"></div>

<script>

var odiv1=document.getElementById("div1")

var odiv2=document.getElementById("div2")

odiv1.onmouseover=odiv2.onmouseover=function(){

odiv2.style.display="block";

clearTimeout(timer);

}

odiv1.onmouseout=odiv2.onmouseout=function(){

timer=setTimeout(function(){

odiv2.style.display="none";

},300)}

</script>

1. div滚动

<body>

<button id="btn1">向右移动</button>

<button id="btn2">停止移动</button>

<div id="div" style="width: 100px;height: 100px;background-color: #A9A9A9;position: absolute;"></div>

<script>

var obtn1=document.getElementById("btn1");

var obtn2=document.getElementById("btn2");

var odiv=document.getElementById("div");

obtn1.onclick=function(){

timer=setInterval(function(){

//**timer前不可以加var，否则timer为局部变量**，在函数外无法调用，计时器无法停止。

odiv.style.left=odiv.offsetLeft+5+'px';

},30)

}

obtn2.onclick=function(){

clearInterval(timer);

}

</script>

</body>

注意：①元素.offsetleft表示元素的外边**框**距离相对定位元素内边**框**的距离（包括元素左外边距），并且以一个数字的形式存在，无px。绝对定位元素.style.left表示元素外边**距**的外边缘距离相对定位元素内边**距**的内边缘的距离（不包括元素左外边距）。②绝对定位元素基本公式：绝对定位元素.offsideleft+‘px’=绝对定位元素.style.left+绝对定位元素.margin-

left。当绝对定位元素的margin-left等于0时，基本公式：绝对定位元素.offsideleft+‘px’=绝对定位元素.style.left。③移动公式：绝对定位元素.left=绝对定位元素.offsetleft+数字+’px’。（前提：绝对定位元素的margin-left等于0）其中第一个加号有数字相加，数字绝对不能加引号，第二个加号为数字与字符串的字符串相加。

3. 无缝滚动

注意：offsetLeft左边距、offsetTop上边距、offsetWidth元素宽度offsetHeight元素高度。（四者都为数字，无px。）

div设置overflow:hidden，ul设置绝对定位，在计时器函数中改变ul的left值。

0 JavaScript学前预热课程-5

案例待学

1 JavaScript基础教程-1

1. JavaScript组成：①ECMAScript：解释器、翻译。②DOM③BOM

ECMAScript：将程序代码与0、1之间相互翻译转化。 几乎没有兼容性问题。

DOM：Document Object Model。Document指网页。有一些兼容性问题。

DOM使JavaScript拥有操作页面的能力。

BOM：Browser Object Model。 没有兼容性问题——几乎不兼容。

BOM使JavaScript拥有操作浏览器的能力。

1. HTML元素的数据类型为Object。

例子：<div id="div"></div>

<script>

var div=document.getElementById("div");

alert(typeof div); //弹出Object，为对象数据类型。

</script>

变量的数据类型由变量所存储数据的类型决定。

undefined数据类型：①没有被声明的变量：is not defined。②虽然声明了，但是没有被赋值的变量：undefined。

1. 对象

基本类型（不可被分解为其他数据类型）：数字、字符串、布尔

复合类型（可以被分解为基本数据类型）：对象

例如：div id=“字符串” width=“数字”

基本类型与复合类型的对比类似水滴与汽车的对比。

除了数字、字符串、布尔值和undefined之外都为对象，都是复合类型。

例如：标签、数组、日期……都为对象。

1. JavaScript为弱类型语言，一个变量可以存储不同类型的数据，但一个变量应该只存放一种类型的数据。

1 JavaScript基础教程-2

1. 数据类型转换：

例子： <input type="text"/>//文本框内的数据类型为string。

<input type="text"/> //文本框内的数据类型为string。

<button id="btn">求和</button>

<script>

var ipt1=document.body.children[0];

var ipt2=document.body.children[1];

var btn=document.body.children[2];

btn.onclick=function(){

alert(parseFloat(ipt1.value)+parseFloat(ipt2.value))

}

</script>

parseInt(“字符串/数字”)——从字符串或数字起始位置开始提取整数，若无整数则返回NAN。

parseFloat(“字符串/数字”)——从字符串或数字起始位置开始提取浮点数（整数加小数），若无浮点数则返回NAN。

注意：①parseInt和parseFloat是显式类型转换（强制类型转换）②NAN也不等于NAN。isNAN是一个函数，用来检测NAN是否是NAN。

例子： alert(NaN==NaN) //弹出false

alert(isNaN(NaN)) //弹出true

1. 隐式类型转换

加号：①数字相加 ②字符串连接。

减法、乘法、除法、取余：数字运算。

1. 变量作用域和闭包

变量作用域：全局变量、局部变量。

局部变量：定义在函数中，只能在本函数中用。

全局变量：不定义在任何一个函数中，可以在任何地方去用。

1. 闭包

**子函数可以使用父函数的局部变量。**

1 JavaScript基础教程-3

1. 命名规范及必要性

可读性——能看懂；规范性——符合规范

匈牙利命名法：①类型前缀；②首字母大写。例如：var oDivUserLogin=

变量存储不同数据类型时的前缀命名规则：数组a；布尔值b；浮点数f；函数fn；整数i；对象o；正则表达式re；字符串s；变体变量（不限定数据类型）v。

注意：该命名规则，既适用于变量，又适用于函数，但是函数无类型前缀。例如：function showUserName(){}

1. 运算符

var oul=document.getElementById("oul");

var oli=oul.getElementsByTagName("li"); //规范性写法。

1. 改变运算符优先级：括号
2. 程序流程控制：

判断：if、switch、？：

循环：while、for

跳出：break、continue

什么是真：true、非零数字、非空字符串、空格、非空对象。

什么是假：false、数字零、空字符串、空对象、undefined、NaN。

1. Json(JavaScript Object Notation)

<script>

var obj={a:4,b:[1,2,'fg'],c:"abc"};

alert(obj.b) //弹出1,2,fg

var arr=[{a:6,b:3},{a:9,b:0}]

alert(arr[0].b);//弹出3

</script>

注意：既可以把数组放在Json里，又可以把Json放在数组里。

例子：使用for in遍历Json：

<script>

var obj={a:4,b:3,c:6};

var arr;

for (arr in obj) {

alert(arr+"="+obj[arr]) //弹出a=4；弹出b=3；弹出c=6。

};

</script>

1 JavaScript基础教程-4

1. 函数返回值：函数返回值为函数的执行结果。函数可以没有return，此时返回undefined。一个函数应该只返回一种类型的值。

例题：<script>

function demo1(){}

alert(demo1()) //弹出undefined。

function demo2(){

return;

}

alert(demo2()) //弹出undefined。

function demo3(){

return 2;

}

alert(demo3()) //弹出2。

</script>

1. 函数传参

可变参（不定参）argument：参数个数可变，参数数组。

argument：一个存储所有函数参数的数组。

例题：<script>

function demo(){

alert(arguments.length)//弹出2

alert(arguments[1])//弹出5

}

demo(3,5)

</script>

注意：只要函数的参数不固定，就需要使用argument。

例题：<script>

var result=0;

function sum(){

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

result+=arguments[i]

};

alert(result);

};

sum(3,5,1,2); //弹出11

</script>

1. CSS函数

注意：JavaScript中style只能获取和修改行间样式。

准备：

<div id="div" style="width: 100px;height: 100px;"></div>

<script>

var div=document.getElementById("div");

div.style['background']="blue";

//JavaScript修改CSS行间样式的第二种方法，无点有括号有分号。

// div.style.background="blue";

//JavaScript修改CSS行间样式的第一种方法，有点无括号无分号。

</script>

注意：在获取或修改CSS样式时，点属性名可以改为方括号属性名加引号。

例题1：<button id="btn" >样式</button>

<div id="div" style="width: 200px;height: 200px;background:

aquamarine;"></div>

<script>

var btn=document.getElementById("btn");

var div=document.getElementById("div");

function demo(){

if (arguments.length==2) { //获取

return arguments[0].style[arguments[1]];

} else if(arguments.length==3){ //设置

arguments[0].style[arguments[1]]=arguments[2]

}

}

btn.onclick=function(){

demo(div,'background','green'); //在JavaScript中，所有变量都归属于一种数据类型，需符合书写规范。**background只能是字符串类型，必须加引号。**

alert(demo(div,'width')) ;

}

</script>

如何获取非行间样式？

语法：

①元素.currentStyle.属性名

第二种写法：元素.currentStyle[“属性名”]

**只有在IE浏览器中**可以获取当前CSS样式属性值（无论是行间样式、内部样式还是外部样式）。

②getComputedStyle(元素,false).属性名;

第二种写法：getComputedStyle(元素,false)[“属性名”]);

**在IE高版本以及其他浏览器中**获取计算后的CSS样式属性值（无论是行间样式、内部样式还是外部样式）。

注意：**只可以获取，不可以修改。**

<style>

#div{width: 100px;

background: red;

}

</style>

</head>

<body>

<div id="div" style="height: 100px;"></div>

<script>

var div=document.getElementById("div");

if(div.currentStyle){

alert(div.currentStyle.width);

}else{

alert(getComputedStyle(div,false).width);

};

</script>

4. 封装currentstyle

<style>

#div{width: 100px;

background: red;

height: 150px;

}

</style>

</head>

<body>

<div id="div"></div>

<script>

function getStyle(obj,attr){

if (obj.currentStyle) {

return obj.currentStyle[attr]

} else{

return getComputedStyle(obj,false)[attr]

};

};

var div=document.getElementById("div");

alert(getStyle(div,"background"))

</script>

</body>

注意：元素.currentStyle.属性名以及getComputedStyle(元素,false).属性名);**无法获取复合样式**，只能获取基本样式。

1. 完整版CSS函数

<style>

#div{width: 100px;

height: 150px;

background: red;

}

</style>

</head>

<body>

<button id="btn">按钮</button>

<div id="div"></div>

<script>

var btn=document.getElementById("btn");

var div=document.getElementById("div");

function getstyle(obj,attr){

if (obj.currentStyle) {

return obj.currentStyle[attr];

} else{

return getComputedStyle(obj,false)[attr]

}

}

function demo(obj,attr,value){

if (arguments.length==2) {

return getstyle(obj,attr);

} else if(arguments.length==3){

obj.style[attr]=value;

}

}

btn.onclick=function(){

demo(div,"background","yellow")

alert(demo(div,"width")) //等号左边是事件，则等号右边是包裹JavaScript代码的函数。

};

</script>

注意：事件函数有两种写法，一种是事件等于完整的函数，一种是事件等于函数名无括号。即：括号与函数体不可分离。（与句柄相同）

1 JavaScript基础教程-5

1. 一个变量应该只存储一种类型数据。一个函数应该只返回一种类型数据。一个数组应该只存一种类型变量。
2. 数组排序：

<script>

var arr=[2,5,76,12,9,34];

arr.sort(function(a,b){

return a-b;

});

alert(arr);

</script>

1. 数组.splice(起始底，**删个数**，增素1，增素2，....)

可以实现数组任何位置**删除**、**插入**、以及**替换**功能。

2 DOM-1

1. 文档碎片

文档碎片可以提高DOM操作性能（理论上）

文档碎片原理:避免页面多次回流。

document.creatDocumentFragment()

例题：

<button id="btn">普通</button>

<ul id="ul"></ul>

<script>

var btn=document.getElementById("btn");

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

var frag=document.createDocumentFragment()//创建文档碎片。

btn.onclick=function(){

var time=new Date().getTime()

for (n=1;n<100000;n++) {

var li= document.createElement("li")

frag.appendChild(li); //将创建的元素存入文档碎片

}

ul.appendChild(frag);//将文档碎片加入ul

alert(new Date().getTime()-time)

}

</script>

2 DOM-2

1. DOM基础

DOM可以使JavaScript操作页面元素。

DOM也是一个标准、规范。现行DOM标准为：DOM1、DOM2。

1. DOM——节点（元素标签属于一类节点）

**元素.offsetParent：定位此元素作为定位基准的父级元素。**

1. 元素CSS样式 属性操作：

第一种：oDiv.style.display=”block”; 点

第二种：oDiv.style[“display”]=”block”; 中括号分号

第三种：Dom方式

①getAttribute(“属性名”)

②setAttribute(“属性名称”，“新的属性值”)

③removeAttribute(“属性名”)

1. DOM元素灵活查找

document.getElementsByClassName()

2 BOM

1. document归属于window。即：网页归属于浏览器。

document是DOM根节点，而window不是节点。

例子： <textarea id="tex" rows="20" cols="40"></textarea><br />

<button id="btn">按钮</button>

<script>

var tex=document.getElementById("tex");

var btn=document.getElementById("btn");

btn.onclick=function(){

var neWindow=window.open("about:block");

neWindow.document.write(tex.value);

//在新窗口中输出文档的方法。

}

</script>

2. window.open()

火狐浏览器：不允许脚本关闭(window.close())非脚本打开（用户手动打开）的页面，只允许脚本关闭(window.close())脚本打开(window.open())的页面。

1. 获得浏览器版本：

window.navigator.useAgent;

注意：可以帮助理解“BOM使JavaScript拥有操作浏览器的能力。”

1. window.location.href不仅可以输出地址，而且可以修改地址。
2. 广告侧边栏

IE6不支持position：fixed。

**回到顶部 不懂 待学！！**

2 字符串详解

1. 获取类

charAt(下标值) 指定位置字符 注：char字符

charCodeAt(下标值) 指定位置字符的ASCII码 字符→编码

**String**.formCharCodeAt(ASCII码) 编码→字符

1. 查找类

indexOf(“索引的内容”)

lastIndexOf(“索引的内容”)

search(“索引的内容”) 适用于正则表达式

1. 比较类

字符串1.localeCompare(字符串2) ——按照本地习惯排序。

例如： alert("不".localeCompare("佛")) //弹出-1。

1. 截取类

字符串.slice(起始底，起始底+截个数) //负数会变成倒数第几个。

字符串.slice(起始底) //从起始底一直取到结束为止。

字符串.subString(起始底，起始底+截个数) //负数直接变成零。

字符串.subString(起始底) //从起始底一直取到结束为止。

字符串.subStr(起始底，结束底) //负数直接变成零。

字符串.subStr(起始底) //从起始底一直取到结束为止。

|  |  |  |
| --- | --- | --- |
| slice | 不取结束位置 | 负数变成倒数 |
| substring | 不取结束位置 | 负数变成零 |
| substr | 取结束位置 | 负数变成倒数 |

1. 其他

split(‘分隔为数组’)

3 事件详解-1

1. event为事件对象，通过它可以获取所有的事件相关信息,在火狐浏览器中不兼容。

<script>

document.onclick=function(){

//给document添加点击事件函数,而不是给document.body添加点击事件函数，因为如果body内无内容时，body的高为0px，而document为DOM文档根节点，在网页任何位置都会被点击。

alert(event.clientX+','+event.clientY)

//弹出点击事件发生时，鼠标的X、Y坐标位置，无Px。

}

</script>

注意：clientX与clientY属于**事件**（鼠标点击、鼠标移入、鼠标移出、鼠标移动……）**的属性，**属性值等于**事件发生时（不仅仅是鼠标点击时），光标的坐标数值**，无px。

例子： <div id="div1" style="background-color: aqua;"></div>

<div id="div2" style="background-color:#DC143C;"></div>

<div id="div3" style="background-color:bisque;"></div>

<div id="div4" style="background-color:chartreuse"></div>

<script>

var div1=document.getElementById("div1");

var div2=document.getElementById("div2");

var div3=document.getElementById("div3");

var div4=document.getElementById("div4");

div1.onclick=function(ev){

alert(ev.clientX)

}

div2.onmouseover=function(ev){

alert(ev.clientX)

}

div3.onmouseout=function(ev){

alert(ev.clientX)

}

div4.onmousemove=function(ev){

alert(ev.clientX)

}

</script>

1. 在IE高版本以及其他浏览器中事件函数内的参数为事件本身。

<script>

document.onclick=function(a){

alert(a)

//弹出Object MouseEvent鼠标点击事件，在IE高版本以及其他任何浏览器中都兼容。

// alert(a.clientX+','+a.clientY)

//弹出点击时，鼠标的X、Y坐标。

}

</script>

1. 用if 解决事件对象的浏览器兼容问题（麻烦）

<script>

document.onclick=function(a){

if (a) { //在IE高版本以及其他浏览器执行此语句。

alert(a.clientX+';'+a.clientY)

} else{ //在IE低版本中执行此语句。

alert(event.clientX+';'+event.clientY)

}

}

</script>

1. 用或||解决浏览器兼容问题：

<script>

document.onclick=function(a){

var oEvent=a||event;

alert(oEvent.clientX+';'+oEvent.clientY)

}

</script>

||的原理：若第一个为真，只执行||之前；若第一个为假，则执行||之后。

1. 事件流：

事件冒泡：

例子1：<div style="width: 300px;height: 300px;background-color: #DC143C;" onclick="alert(this.style.height)">

<div style="width: 200px;height: 200px;background-color:aqua;" onclick="alert(this.style.height)">

<div style="width: 100px;height: 100px;background-color:cadetblue;" onclick="alert(this.style.height)"></div>

</div>

</div>

例子2：<button id="btn">按钮</button>

<div id="div" style="width: 100px;height: 150px;background-color: #DEB887;display: none;"></div>

<script>

var btn=document.getElementById("btn");

var div=document.getElementById("div");

btn.onclick=function(ev){

div.style.display="block";

var oEvent=ev||event //浏览器兼容。

oEvent.cancelBubble=true; //阻止事件冒泡。

}

document.onclick=function(){

div.style.display="none"

}

</script>

3 事件详解-2

6. **scroll top滚动条滚动距离，即可视区的滚动距离，即可视区距离body顶端的距离。**

实例1： <body style="height: 2000px;">

<script>

document.onclick=function(){

alert(document.documentElement.scrollTop)

//谷歌浏览器不兼容。

}

</script>

</body>

实例2： <body style="height: 2000px;">

<script>

document.onclick=function(){

alert(document.body.scrollTop)

//只适用于谷歌浏览器。

}

</script>

</body>

实例3： <body style="height: 2000px;">

<script>

document.onclick=function(){

var scrollTop=

document.documentElement.scrollTop||document.body.scrollTop;

alert(scrollTop)

//兼容写法，适用于任何浏览器。

}

</script>

</body>

7. 鼠标事件

clientX页面可视区鼠标横坐标

clientY页面可视区鼠标纵坐标（最大值为可视区高度，而不是body的高度。）

鼠标移动事件：onmousemove

<body style="height: 1000px;">

//body设置1000px高度之后，此时body高度大于页面可视区高度。

<div id="div" style="width: 100px;height: 100px;background-color: #FF0000;position: absolute;">

</div>

<script>

document.onmousemove=function(ev){ //给document添加鼠标移动事件，使鼠标在页面任何位置滚动都可以触发此事件。

var oEvent=ev||event;

var div=document.getElementById("div");

var scrollTop=

document.documentElement.scrollTop||document.body.scrollTop;

var scrollLeft=

document.documentElement.scrollLeft||document.body.scrollLeft;

div.style.left=oEvent.clientX+scrollLeft+'px'; div.style.top=oEvent.clientY+scrollTop+'px';

// 鼠标移动事件发生时，使div在的body中位置等于鼠标在可视区的坐标，加上可视区的滚动距离。

}

</script>

</body>

注意：clientX一定要配合scrollLeft使用；clientY一定要配合scrollTop使用。

8. 一串跟随鼠标的DIV

<script>

var divs=document.getElementsByTagName('div');

document.onmousemove=function(ev){

var oEvent=ev||event;

var scrollTop=

document.documentElement.scrollTop||document.body.scrollTop;

var scrollLeft=

document.documentElement.scrollLeft||document.body.scrollLeft;

divs[0].style.left=ev.clientX+scrollLeft+'px';

divs[0].style.top=ev.clientY+scrollTop+'px';

for (n=divs.length-1;n>0;n--){

//循环倒着写：从尾到头传递，而不是从头到位传递。

divs[n].style.left=divs[n-1].style.left

divs[n].style.top=divs[n-1].style.top

}

}

</script>

1. offsetWidth等于width加padding加border减去px。（无margin）

offsetHeight等于height加padding加border减去px。（无margin）在绝对定位元素中，offsetLeft等于left加margin减去px。（无padding，无border）；offsetTop等于top加margin减去px。（无padding，无border）。

即：offsetWidth与offsetHeight为元素**视觉**尺寸，offsetLeft与offsetTop为绝对定位元素**视觉**位置。

10.键盘事件——keyCode

onclick=onmousedown+onmouseup;

onkeypress=onkeydown+onkeyup;

例子： <script>

**document**.onkeypress=function(ev){

var oEvent=ev||event;

alert(oEvent.keyCode);//获得键盘事件的键值（键码）。

}

</script>

1. 键盘控制div移动

<div></div>

<script>

odiv=document.getElementsByTagName('div')[0];

**document**.onkeydown=function(ev){

var oEvent=ev||event;

if (oEvent.keyCode==83) {

odiv.style.top=odiv.offsetTop+5+'px'

} else if (oEvent.keyCode==87) {

odiv.style.top=odiv.offsetTop-5+'px'

} else if (oEvent.keyCode==65) {

odiv.style.left=odiv.offsetLeft-5+'px'

} else if (oEvent.keyCode==68) {

odiv.style.left=odiv.offsetLeft+5+'px'

}

}

</script>

1. 鼠标点击 提交留言

<textarea id="tex1" cols="40" rows="20"></textarea>

<br />

<textarea id="tex2" cols="20" rows="1"></textarea>

<button id="btn">按钮</button>

<script>

var tex1=document.getElementById("tex1");

var tex2=document.getElementById("tex2");

var btn=document.getElementById("btn");

btn.onclick=function(){

tex1.value+=tex2.value+"\n";

//注意：在value值中加**\n**，起到换行的效果。

//文本框中的文字既不是文本框的innerHTML值，又不是文本框的innerText值，而是文本框的value值。注意加等的运用。

tex2.value=''

}

</script>

1. Ctrl+回车 提交留言

键盘事件对象的其他属性 ctrlKey、shiftKey、altKey（尽量少用此属性）三个属性是一个布尔值。当键盘按着ctrl时，ctrlKey为true，否者为false。当键盘按着shift时，shiftKey为true，否者为false；当键盘按着alt时，altKey为true，否者为false。

<textarea id="tex1" cols="40" rows="20"></textarea>

<br />

<textarea id="tex2" cols="20" rows="1"></textarea>

<button id="btn">按钮</button>

<script>

var tex1=document.getElementById("tex1");

var tex2=document.getElementById("tex2");

var btn=document.getElementById("btn");

btn.onclick=function(){

tex1.value+=tex2.value+"\n";//注意：在value值中加\n，起到换行的效果。

//文本框中的文字既不是文本框的innerHTML值，又不是文本框的innerText值，而是文本框的value值。注意加等的运用。

tex2.value='';

}

tex2.onkeydown=function(ev){ //注意：是给第二个文本框添加事件。

var oEvent=event||ev;

if (oEvent.ctrlKey&&oEvent.keyCode==13) {

//当键盘同时按着ctrl与回车时，if条件为true。

tex1.value+=tex2.value+"\n";

tex2.value=''

}

}

</script>

注意：①在页面什么位置添加键盘事件。②操作哪个键盘时，执行事件函数。

③clientX、clientY为鼠标事件的属性，keyCode为键盘事件的属性。即：**oEvent.clientX;oEvent.clientY;oEvent.keyCode。**

1. event事件对象

事件对象包含事件的一些基本信息：鼠标点击位置、用户按哪个键。

3 事件详解-3

1. 默认行为：不需要我们去编写，浏览器自带的功能。例如：在页面中点击右键出现菜单。
2. 阻止默认行为：

右键行为：oncontextmenu

<script>

document.oncontextmenu=function(e){

return false;

//阻止浏览器oncontextmenu（鼠标右键）默认事件函数。

//方法二: e.preventDefault()

}

</script>

<input type="text" id="ipt" />

<script>

var ipt=document.getElementById("ipt");

ipt.onkeydown=function(e){

return false;//阻止文本框的onkeydown默认事件函数。

//方法二: e.preventDefault()

}

</script>

只要是浏览器自带的默认事件函数，都可以被修改。

3. 自定义右键菜单

<ul id="ul" style="width: 70px;background-color: #A9A9A9;position: absolute;display: none;">

<li>登录</li>

<li>回到首页</li>

<li>加入VIP</li>

</ul>

<script>

var ul=document.getElementById("ul")

document.oncontextmenu=function(ev){

ul.style.display="block";

var oEvent=event||ev;

var scrolltop=document.body.scrollTop||document.documentElement.scrollTop;

var scrollleft=document.body.scrollLeft||document.documentElement.scrollLeft;

**//易忘点：①document.documentElement。②加‘px’**

ul.style.top=oEvent.clientY+scrolltop+'px';

ul.style.left=oEvent.clientX+scrollleft+'px';

return false;//必须放在结尾，否则ul无法显示。

//方法二: oEvent.preventDefault()

}

document.onclick=function(){

ul.style.display="none";

}

</script>

1. 只能输入数字的输入框。

方法1： <input id="ipt" type="text"/>

<span id="span"></span>

<script>

var ipt=document.getElementById("ipt");

var span=document.getElementById("span");

re=**/^\d+$/**;

ipt.onblur=function(){

if(!re.test(ipt.value)){

span.innerText="输入错误";

}else{

span.innerText="输入正确"

}

}

ipt.onfocus=function(){

span.innerText=''

}

</script>

方法2： <input type="text" id="ipt"/>

<script>

var ipt=document.getElementById("ipt");

ipt.onkeydown=function(ev){

oEvent=event||ev;

if (oEvent.keyCode!=8&&(oEvent.keyCode<48||oEvent.keyCode>57)) {

return false;

//方法二: oEvent.preventDefault()

}

}

</script>

1. div简易拖拽

<div id="div" style="width: 100px;height: 100px;background: red;position: absolute;"></div>

<script>

var div=document.getElementById("div");

var disX=0;

var disY=0;

div.onmousedown=function(ev){

var oEvent=event||ev;

var disX=oEvent.clientX-div.offsetLeft;

var disY=oEvent.clientY-div.offsetTop;

document.onmousemove=function(ev){

var oEvent=event||ev;

var l=oEvent.clientX-disX;

var t=oEvent.clientY-disY;

**问题：为什么不加scrollTop和scrollHeight:当body的高度过大，页面右侧产生滚动条，并且滚动条下滑时，disY=oEvent.clientY-div.offsetTop;可能为负值（滚动条滚动距离已涵盖在div.offsetTop中）。**

if(l<0){

l=0;

}else if(l>document.documentElement.clientWidth-div.offsetWidth){

l=document.documentElement.clientWidth-div.offsetWidth;

}

if(t<0){

t=0;

}else if(t>document.documentElement.clientHeight-div.offsetHeight){

t=document.documentElement.clientHeight-div.offsetHeight;

}

div.style.left=l+'px'

div.style.top=t+'px'

};

document.onmouseup=function(){

document.onmousemove=null;

document.onmouseup=null;

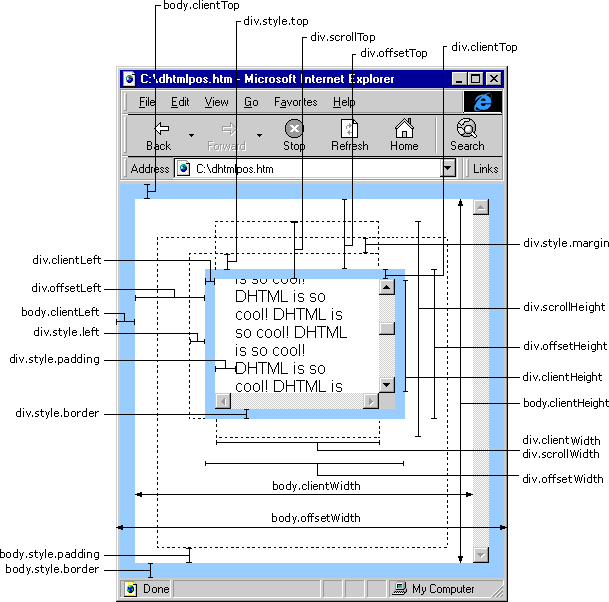
};

return false;

};

</script>

6. offset、client与scroll



|  |  |  |
| --- | --- | --- |
| client | clientWidth/Height | 元素**可视区**宽度/高度 |
| offset | offsetTop/Left | 元素**自身**距定位基准元素  上方/左方距离 |
| offsetWidth/Height | 元素**自身**宽度/高度（包括border） |
| scroll | scrollLeft/Top | 元素**卷起来**的高度/宽度 |

注意:滚动条的拉动并不会改变元素的offsetTop/Left值。

4 cookie

1. cookie——页面用来保存信息，比如：自动登录、记住用户名
2. cookie的特性——同一个网站所有的页面共享一套cookie；

数量、大小有限；过期时间

1. JS中使用cookie——document.cookie
2. cookie的使用

设置cookie——格式：名字=值； 不会覆盖； 过期时间：expires=时间

<script>

var oDate=new Date();

oDate.setDate(oDate.getDate()+30);

document.cookie='user=jake;expires='+oDate;

document.cookie='pass=123';

alert(document.cookie)

</script>

设置cookie封装函数：

function setCookie(name,value,iDay){

**var oDate=new Date();**

**oDate.setDate(oDate.getDate()+iDay);**

document.cookie=name+'='+value+';expires='+oDate;

};

setCookie('password',123456,60)

alert(document.cookie)

获取cookie封装函数：

function getCookie(name){

var str=document.cookie

var arr=str.split('; ')

//以**分号加空格**作为分隔符。

for (n=0;n<arr.length;n++) {

var arr2=arr[n].split('=');

if (arr2[0]==name) {

return arr2[1];

};

};

return ''

};

删除cookie封装函数：

function removeCookie(name){

**setCookie(name,'1',-1)**

};

5. Cookie结合拖拽/待学

6. 记住用户名

<form id="form" action="Ctrl加回车提交留言.html">

用户名：<input type="text" name="user" />

密码：<input type="password" name='password' />

<input type="submit" value="登录"/>

<a href="#">清除记录</a>

</form>

<script>

function setCookie(name,value,iDay){

var oDate=new Date();

oDate.setDate(oDate.getDate()+iDay);

document.cookie=name+'='+value+';expires='+oDate;

}

function getCookie(name){

var str=document.cookie

var arr=str.split('; ')

//以分号加空格作为分隔符。

for (n=0;n<arr.length;n++) {

var arr2=arr[n].split('=');

if (arr2[0]==name) {

return arr2[1];

};

};

return ''

};

function removeCookie(name){

setCookie(name,'1',-1)

};

var form=document.getElementById('form');

var user=document.getElementsByName('user')[0];

var aclear=document.getElementsByTagName('a')[0];

form.onsubmit=function(){//表达提交事件。

setCookie('user',user.value,30);

};

user.value=getCookie('user')

aclear.onclick=function(){

removeCookie('user');

user.value=''

}

</script>

5 AJAX-1

1. 本节课程内容：如何使用一个已经写好的Ajax库，去读取服务器数据。
2. Ajax——无刷新数据读取
3. Ajax只能从服务器读取数据，如何配置自己的服务器程序。

安装wamp软件： Wamp就是Windows Apache Mysql PHP集成安装环境，即在window下的apache、php和mysql的服务器软件。

在写Ajax应用时，一定要注意路径。

localhost/aaa.txt 注意：localhost/访问本地服务器。

aaa.txt放在wamp/www文件中。

4. Ajax应用1

//点击按钮的时候，读取abc.txt文件

<button id="btn">读取</button>

<script src="ajax.jx"></script>

<script>

var btn=document.getElementById('btn');

btn.onclick=function(){

//**ajax能且仅能从服务器读取数据。**

ajax('abc.txt?t='+new Date().getTime(),function(str){

alert(str);

})

}

</script>

5. Ajax应用2

<button>按钮1</button>

<button>按钮2</button>

<button>按钮3</button>

<div id="div" style="width: 200px;height: 200px;background-color: #A9A9A9;"></div>

<script src="ajax.jx"></script>

<script>

var div=document.getElementById('div');

var btns=document.getElementsByTagName('button');

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

btns[i].index=i;

btns[i].onclick=function(){

ajax(this.index+1+'.txt',function(str){

div.innerHTML=str;

});

};

};

</script>

注意：ajax(**ajax文件名**,**function(str){代码块}**)

6. 缓存

浏览器缓存的作用：降低服务器压力，一个服务器文件数据只读一次。

解决缓存的方法：?t='+new Date().getTime()。此时浏览器认为每次读取的都是服务器的新文件地址，不会利用上次缓存，但读取到的是同一个文件。实现服务器文件数据及时在页面更新。

7. 动态数据：请求JS或（json）文件。

eval的使用。 DOM创建元素。

Ajax从服务器读取的所有数据都是字符串类型。

eval的作用——计算字符串里的值，即删去字符串的两边引号，将字符串转化为引号内所包裹的数据。

例子1：<script>

var a='[1,2,3]';

alert(eval(a)[0]);//弹出1。

var str='function show(){alert(2)}'

eval(str);

show();//弹出2。

</script>

例子2：//读取服务器上的一个文件，文集里放了一个Json。<br />

<button id="btn">按钮</button>

<script src="ajax.js"></script>

<script>

var btn=document.getElementById('btn');

btn.onclick=function(){

ajax('date.json',function(str){

var arr=eval(str);

alert(arr[0].a);

});

}

</script>

8. Ajax原理

HTTP请求方法：

GET——用于**获取数据**（如：浏览帖子）

POST——用于**上传数据**（如：用户注册）

get把数据放在url(网址)里提交：安全性低、容量很低。便于分享。get方式会缓存。

post不把数据放在url（网址）里提交：安全性一般，容量几乎无限。不便于分享。post方式不会缓存。

例子： <form action="Ajax.html" method="get">

<input type="text" name="user" />

</form>

注意：name的作用——name属性只出现在能提交数据的表单标签上，使后台获取到提交数据的信息。后台接收到的数据为：?name值=填写内容&name值=填写内容……

5 AJAX-2

1. 如何去写一个Ajax库。
2. Ajax-Asynchronous JavaScript and XML(异步JavaScript和XML)
3. **全局变量**是全局对象window的属性，**全局函数**是全局对象window的方法。当你用一个未声明的变量：运行错误；当你用一个不存在的属性：undefined。

<script>

alert(a);//运行错误a is not defined

</script>

<script>

alert(window.a);//弹出undefined。

</script>

1. 请求状态监控

onreadystatechange事件

oAjax.readyState属性：请求状态

0——(未初始化)还没有调入open()方法

1——(载入)已调入send()方法，正在发送请求。

2——(载入完成)send()方法完成，已收到全部响应内容。

3——(解析)正在解析响应内容。

4——(完成)响应内容解析完成，可以在客户端调用了。

oAjax.status属性：请求结果。

oAjax.responseText——获取服务器文件

1. Ajax库

function Ajax(url,funSucc,fnFaild){

//打电话: ①买手机 ②拨号 ③说话 ④听

//ajax请求:①创建Ajax对象 ②链接服务器 ③发送请求 ④接受返回

//①创建Ajax对象。

var oAjax=null;

if (window.XMLHttpRequest) {

**oAjax=new XMLHttpRequest();**

} else{

oAjax=new

ActiveXObject('Microsoft.XMLHTTP')

}

//②连结服务器

//open(方法,url,是否异步)

**oAjax.open('GET',url,true);**

// ③发送请求

**oAjax.send();**

//④接收返回

oAjax.**onreadystatechange**=function(){

if (oAjax.readyState==4) {

if (oAjax.status==200) {

fnSucc(**oAjax.responseText**);

} else{

if(fnFaild){

fnFaild();

};

};

};

};

};

注意：new——→open——→send——→reponseText

1. Ajax数据

数据类型——XML结构化信息(淘汰)、Json(取代XML)

字符集——所有文件字符集相同。

6 运动课程-1

1. Div匀速运动

<button id="btn1">开始运动</button>

<button id="btn2">停止运动</button>

<div id="div" style="width: 100px;height: 100px;background-color: #DC143C;position: absolute;"></div>

<script>

var btn1 = document.getElementById('btn1');

var btn2 = document.getElementById('btn2');

var div = document.getElementById('div')

var timer=null;

//这句不可省，否则第一次点击出错。并且必须放在函数之外。

btn1.onclick = function() {

clearInterval(timer);

//不可省略，否则多次点击，速度越来越快。

var iSpeed=1;

timer = setInterval(function() {

if (div.offsetLeft >= 800) {

//不能写等于，否则在移动的时候可能跳过该数值。

clearInterval(timer)

}else{

//下行代码必须放在else中，使当div的offsetleft小于800时，才会移动。

div.style.left = div.offsetLeft + iSpeed + 'px';

};

}, 30)

};

btn2.onclick = function() {

clearInterval(timer)

};

</script>

总结：BUG1——不会停止（清除定时器的条件改为大于等于）；BUG2——到达位置后再点还会运动（**在定时器函数中**使用if else分别包裹清除计时器代码和运动代码）；BUG3——越点越快（函数运行开始清除定时器）。

注意：var timer=null;不能省，并且放在函数之外。

2. 运动框架及应用

框架即流程

在开始运动时，关闭已有的定时器。——clearInterval(timer);

把运动和停止隔开。——if/else

3. “分享到”侧边栏

<body>

<div id="div">

<span>分享到</span>

</div>

<script>

var div = document.getElementById('div');

var timer = null;

div.onmouseover = function() {

**clearInterval(timer);**

timer = setInterval(function() {

**if (div.offsetLeft == 0) {**

**clearInterval(timer);**

} else {

div.style.left = div.offsetLeft + 10 + 'px'

}

}, 30);

}

div.onmouseout=function(){

clearInterval(timer);

timer=setInterval(function(){

if (div.offsetLeft==-150) {

clearInterval(timer)

} else{

div.style.left=div.offsetLeft-10+'px'

}

},30)

}

</script>

注意：①在开始运动时，关闭已有的定时器。②把运动和停止隔开，同时放在定时器函数中。

减化版：<div id="div">

<span>分享到</span>

</div>

<script>

var div = document.getElementById('div');

var timer = null;

div.onmouseover=function(){

move(10,0)

}

div.onmouseout=function(){

move(-10,-150)

}

function move(**iSpeed,iTarget**) {

clearInterval(timer);

timer = setInterval(function() {

if (div.offsetLeft == iTarget) {

clearInterval(timer);

} else {

div.style.left = div.offsetLeft + iSpeed + 'px'

};

}, 30);

};

</script>

减化版2：<div id="div">

<span>分享到</span>

</div>

<script>

var div = document.getElementById('div');

var timer = null;

div.onmouseover=function(){

move(0);

}

div.onmouseout=function(){

move(-150);

}

function move(iTarget) {

clearInterval(timer);

timer = setInterval(function() {

var iSpeed=0;

if (div.offsetLeft<iTarget) {

iSpeed=10;

} else{

iSpeed=-10;

}

if (div.offsetLeft == iTarget) {

clearInterval(timer);

} else {

div.style.left = div.offsetLeft + iSpeed + 'px';

}

}, 30);

};

</script>

注意：函数——A：参数为目标状态；B：第一步清除计时器；C：计时器函数；D：if else判断移动方向；E：if else隔开移动与清除计时器。**一清一计两判断。**

6 运动课程-2

1. 淡入淡出的图片

<img id="img" src="img/1117566863e67fbea4m.jpg" style="opacity: 0.5;" />

<script>

var timer=null;

var alpha=0.5;

var img=document.getElementById('img');

img.onmouseover=function(){

demo(1);

};

img.onmouseout=function(){

demo(0.3);

};

var timer=null;

function demo(iTarget){

clearInterval(timer);

timer=setInterval(function(){

var iSpeed=0;

if (alpha<iTarget) { //alpha代表图片当前的透明度。

iSpeed=0.1;

} else{

iSpeed=-0.1;

}

if (Math.abs(alpha-iTarget)<0.1) {

clearInterval(timer)

alpha=iTarget

} else{

alpha=alpha+iSpeed;

img.style.opacity=alpha;

};

},10)

}

</script>

1. 缓冲运动

逐渐变慢，直到停止。

<button id="btn1">开始运动</button>

<button id="btn2">停止运动</button>

<div id="div" style="width: 100px;height: 100px;background-color: #DC143C;position: absolute;left: 800px;"></div>

<script>

var btn1 = document.getElementById('btn1');

var btn2 = document.getElementById('btn2');

var div = document.getElementById('div')

var timer=null;//这句不可省，否则第一次点击出错。并且必须放在函数之外。

btn1.onclick = function(){

demo(300);

}

function demo(iTarget) {

clearInterval(timer);//不可省略，否则多次点击，速度越来越快。

timer = setInterval(function() {

var iSpeed=(iTarget-div.offsetLeft)/8;

//解决当速度<1时，不再前进的问题。

if (iSpeed>0) {

iSpeed= Math.ceil(iSpeed)

} else{

iSpeed=Math.floor(iSpeed)

}

if (div.offsetLeft == iTarget) {

clearInterval(timer)

}else{//下行代码必须放在else中，使当div的offsetleft未达到iTarget时，才会移动。

div.style.left = div.offsetLeft + iSpeed + 'px';

}

document.title=div.offsetLeft

}, 30)

};

btn2.onclick = function() {

clearInterval(timer)

};

</script>

1. 上下滑动的侧边栏

<style>

#div{

width: 20px;height: 200px;background-color: #FF8C00;position: absolute;

}

</style>

</head>

<body style="height: 1000px;">

<div id="div"></div>

<script>

var div=document.getElementById('div');

**document.onscroll**=function(){

//document的onscroll鼠标滚动事件。

var **scrollTop=document.documentElement.scrollTop||document.body.scrollTop;**

var t=(**document.documentElement.clientHeight**-div.offsetHeight)/2+scrollTop;

//document.documentElement.clientHeight页面可视区高度。

//document.documentElement.scrollTop||document.body.scrollTop页面上卷高度。

startMove(parseInt(t));

}

var timer=null;

function startMove(iTarget){

clearInterval(timer);

timer=setInterval(function(){

var iSpeed=(iTarget-div.offsetTop)/8;

if (iSpeed>0) {

iSpeed=Math.ceil(iSpeed)

} else{

iSpeed=Math.floor(iSpeed)

};

if (div.offsetTop==iTarget) {

clearInterval(timer)

} else{

div.style.top=div.offsetTop+iSpeed+'px';

};

},50)

};

</script>

1. 匀速运动的停止条件——**距离足够近**

缓冲运动的停止条件——**两点重合**

<button id="btn1">开始运动</button>

<button id="btn2">停止运动</button>

<div id="div" style="width: 100px;height: 100px;background-color: #DC143C;position: absolute;left: 500px;"></div>

<script>

var btn1 = document.getElementById('btn1');

var btn2 = document.getElementById('btn2');

var div = document.getElementById('div')

var timer=null;//这句不可省，否则第一次点击出错。并且必须放在函数之外。

btn1.onclick =function(){

move(10)

}

function move(iTarget) {

clearInterval(timer);

//不可省略，否则多次点击，速度越来越快。

timer = setInterval(function() {

if (iTarget>div.offsetLeft) {

iSpeed=10;

} else{

iSpeed=-10;

}

if (Math.abs(div.offsetLeft-iTarget)<10) {

//不能写等于，否则在移动的时候可能跳过该数值。

clearInterval(timer)

div.style.left=iTarget+'px' }else{//下行代码必须放在else中，使当div的offsetleft小于800时，才会移动。

div.style.left = div.offsetLeft + iSpeed + 'px';

}

}, 30)

}

btn2.onclick = function() {

clearInterval(timer)

}

</script>

1. 运动框架——匀速运动、缓冲运动。

<head>

<meta charset="UTF-8">

<title></title>

<style>

#div {

width: 100px;

height: 100px;

background-color: #FF0000;

position: absolute;

}

#div2 {

width: 100px;

height: 100px;

background-color: #FF8C00;

position: absolute;

right: 0;

}

body {

height: 1000px;

}

</style>

</head>

<body>

<button id="btn1">匀速运动</button>

<button id="btn2">缓冲运动</button>

<div id="div"></div>

<div id="div2"></div>

<script>

var div = document.getElementById('div');

var btn1 = document.getElementById('btn1');

var btn2 = document.getElementById('btn2');

btn1.onclick = function() {

Move1(500);

}

btn2.onclick = function() {

Move2(500);

}

document.onscroll = function() {

var scrollTop = document.documentElement.scrollTop || document.body.scrollTop;

//为解决兼容问题，body上卷高度的两种表示方式： //**documentElement与body**

var t = scrollTop + (document.documentElement.clientHeight - div.offsetHeight) / 2;

//body可视区高度的表示方式：

//**document.documentElement.clientHeight**

t = Math.floor(t);

//解决了移动到目标位置之后，div上下0.5px范围**跳动**的问题。

Move3(t);

var m = scrollTop + (document.documentElement.clientHeight - div2.offsetHeight) / 2;

m = Math.floor(m);

Move4(m);

}

var timer = null;

//函数外首先声明timer变量，解决函数内清除计时器代码所引发的错误。

function Move1(iTarget) {

**//水平方向匀速运动框架。**

clearInterval(timer);

//函数内先清除计时器，解决**点击多次**速度越来越快的问题。

timer = setInterval(function() {

var iSpeed;

if (div.offsetLeft < iTarget) {

iSpeed = 10;

} else {

iSpeed = -10;

};

if (Math.abs(div.offsetLeft - iTarget) < 10) {

//既解决了==**无法停止**的问题，又解决了左右**两个方向**移动停止条件的矛盾。

clearInterval(timer);

div.style.left = iTarget + 'px';

//解决了停止计时器后，无法回到**iTarget位置**的问题。

} else {

div.style.left = div.offsetLeft + iSpeed + 'px';

}

}, 50)

};

function Move2(iTarget) {

**//水平方向缓冲运动框架。**

clearInterval(timer);

timer = setInterval(function() {

var iSpeed = (iTarget - div.offsetLeft) / 8

if (iSpeed > 0) {

iSpeed = Math.ceil(iSpeed);

//解决了速度衰减到1以下之后，无法**持续移动**的问题。

} else {

iSpeed = Math.floor(iSpeed);

};

if (div.offsetLeft == iTarget) {

clearInterval(timer)

} else {

div.style.left = div.offsetLeft + iSpeed + 'px'

}

document.title = div.style.left

}, 50)

};

function Move3(iTarget) {

**//垂直方向匀速运动框架。**

clearInterval(timer);

timer = setInterval(function() {

var iSpeed;

if (div.offsetTop < iTarget) {

iSpeed = 20

} else {

iSpeed = -20

};

if (Math.abs(div.offsetTop - iTarget) < 20) {

clearInterval(timer);

div.style.top = iTarget + 'px'

} else {

div.style.top = div.offsetTop + iSpeed + 'px'

}

}, 50)

}

function Move4(iTarget) {

**//垂直方向缓冲运动框架。**

clearInterval(timer);

timer = setInterval(function() {

var iSpeed = (iTarget - div2.offsetTop) / 8;

if (iSpeed < 0) {

iSpeed = Math.floor(iSpeed);

} else {

iSpeed = Math.ceil(iSpeed);

};

if (div.offsetTop == iTarget) {

clearInterval(timer);

} else {

div2.style.top = div2.offsetTop + iSpeed + 'px'

};

}, 50)

};

</script>

</body>

6 运动课程-3

1.多物体运动框架

<style>

div{

width: 100px;

height: 100px;

background-color: red;

margin-top: 30px;

}

</style>

</head>

<body>

<div></div>

<div></div>

<div></div>

<script>

var divs=document.getElementsByTagName('div')

for (n=0;n<divs.length;n++) {

**divs[n].timer=null;**

//给每一个div添加一个定时器属性，避免开启一个div定时器函数的同时清除另一个div的定时器。

divs[n].onmouseover=function(){

Move(**this,300**) //这里的this指代该方法所属于的对象div，Move函数中的shis指代Move方法所属于的对象window。

};

divs[n].onmouseout=function(){

Move(**this,100**)

}

}

function Move(**obj,iTarget**){

clearInterval(obj.timer);

obj.timer=setInterval(function(){

var iSpeed;

iSpeed=(iTarget-obj.offsetWidth)/8;

if (iSpeed>0) {

iSpeed=Math.ceil(iSpeed);

} else{

iSpeed=Math.floor(iSpeed);

};

if (iTarget==obj.offsetWidth) {

clearInterval(obj.timer)

} else{ obj.style.width=obj.offsetWidth+iSpeed+'px';

//设置div的宽度、高度、位置只能通过style设置，不能通过offset设置。

}

},50)

}

</script>

注意：与单个物体运动框架的区别——①给每一个运动物体添加一个**定时器属性**。②运动框架增加一个**参数**。

1. 多个div淡入淡出

var divs=document.getElementsByTagName('div');

for (n=0;n<divs.length;n++) {

divs[n].alpha=0.3;

//参照移动多个div的left，设置alpha为div的一个属性。

divs[n].timer=null;

divs[n].onmouseover=function(){

move(this,1)

}

divs[n].onmouseout=function(){

move(this,0.3)

}

}

function move(obj,iTaget){

clearInterval(obj.timer);

**obj.timer**=setInterval(function(){

if (**obj.alpha**<iTaget) {

iSpeed=0.1

} else{

iSpeed=-0.1

};

if (Math.abs(obj.alpha-iTaget)<0.1) {

clearInterval(obj.timer)

obj.alpha=iTaget;

} else{

obj.alpha=obj.alpha+iSpeed;

obj.style.opacity=obj.alpha;

}

},50)

}

注意：设置alpha为每个div的一个属性。

总结：多物体运动——**所有东西都不能公用：定时器、透明度……**

1. 当前样式替代offset

例子1：<div id="div"></div>

<script>

function getStyle(obj,attr){

if (obj.currentStyle) {

return obj.currentStyle[attr];

} else{

return getComputedStyle(obj,false)[attr];

//这种写法开始位置不加obj。

}

}

setInterval(function(){

var div=document.getElementById('div');

**div.style.width=parseInt(getStyle(div,'width'))-1+'px';**

},50)

</script>

注意：①offset width指content+padding+border。②在做运动时，过多使用offset容易出错，应使用当前样式替代offset。

1. 任意值变化

function getStyle(obj, attr) {

if (obj.currentStyle) {

return obj.currentStyle[attr];

} else {

return getComputedStyle(obj, false)[attr];

//这种写法开始位置不加obj。

}

};

function Move(obj, attr, iTarget) {

clearInterval(obj.timer);

obj.timer = setInterval(function() {

var icue = parseInt(getStyle(obj, attr));

var iSpeed;

iSpeed = (iTarget - icue) / 8;

if (iSpeed > 0) {

iSpeed = Math.ceil(iSpeed);

} else {

iSpeed = Math.floor(iSpeed);

};

if (icue == iTarget) {

clearInterval(obj.timer)

} else {

obj.style[attr] = icue + iSpeed + 'px';

//设置div的宽度、高度、位置只能通过style设置，不能通过offset设置。

}

}, 50)

};

6 运动课程-4

1. <script>

alert(0.07\*100); //弹出的不是7。

var a=3;

var b=3.00000000000000001;

alert(a==b);//弹出true。

</script>

注意：在写程序的时候，要避免小数。

1. 包含透明度的任意运动框架

function getStyle(obj,attr){

if (obj.currentStyle) {

return obj.currentStyle[attr];

} else{

return getComputedStyle(obj,false)[attr]

}

}

function Move(obj,attr,iTarget){

clearInterval(obj.timer);

obj.timer=setInterval(function(){

var iSpeed=0;

**//声明iSpeed，表示当前样式的变化速度。**

var icue;

**//声明icue，表示当前样式的数值。**

if (attr=='opacity') {

icue=parseInt(getStyle(obj,attr)\*100);

//**为了避免修改后续程序，乘上100，为了避免小数引发的问题，加parseInt。**

} else{

icue=parseInt(getStyle(obj,attr))

}

iSpeed=(iTarget-icue)/8;

if (iSpeed>0) {

iSpeed=Math.ceil(iSpeed);

} else{

iSpeed=Math.floor(iSpeed);

};

if (icue==iTarget) {

clearInterval(obj.timer)

} else{

if (attr=='opacity') {

obj.style[attr]=(icue+iSpeed)/100

} else{

obj.style[attr]=icue+iSpeed+'px'

}

}

},50)

}

注意：**=不等价于==**

3. 仿Flash图片展示（重点）/待学

6 运动课程-5

1. 链式运动框架

function getStyle(obj, attr) {

if (obj.currentStyle) {

return obj.currentStyle[attr]

} else {

return getComputedStyle(obj, false)[attr]

}

}

function startMove(obj, attr, iTarger, fn) {

clearInterval(obj.timer)

obj.timer = setInterval(function() {

var icue;

//声明icue，表示当前样式的数值。

var iSpeed;

//声明iSpeed，表示当前样式的变化速度。

if (attr == 'opacity') {

icue = parseInt(getStyle(obj, attr) \* 100)

} else {

icue = parseInt(getStyle(obj, attr))

};

iSpeed = (iTarger - icue) / 8

if (iSpeed > 0) {

iSpeed = Math.ceil(iSpeed)

} else {

iSpeed = Math.floor(iSpeed)

};

if (icue == iTarger) {

clearInterval(obj.timer);

if (fn) {

fn()

};

} else {

if (attr == 'opacity') {

obj.style[attr] = (icue + iSpeed) / 100;

} else {

obj.style[attr] = icue + iSpeed + 'px';

}

}

}, 50)

};

易错点：①混淆=与==。②function中的this表示当前发生事件的元素，function中function中的this表示window。③iSpeed = (iTarger - icue) / 8。

1. Jason

<script>

var obj={a:3,b:5,c:9};

var arr=[3,7,5];

alert(obj['b']);//弹出5

alert(arr[1]);//弹出7

for (i in obj) {

alert(i+":"+obj[i]);

};//for in不仅可以用来循环数组，还可以用来循环对象。

</script>

注意：对象以字符串作为下标，数组以数字作为下标。

总结：数组既可以用for循环，又可以用for in;对象只能够用for in。

3. 完美运动框架

function getStyle(obj, attr) {

if (obj.currentStyle) {

return obj.currentStyle[attr];

} else {

return getComputedStyle(obj, false)[attr];

}

}

function startMove(obj, **json**, fn) {

**clearInterval(obj.timer);** //默认这一次运动结束，所有值都到达了。

obj.timer = setInterval(function() {

**var bStop = true;**

**for (attr in json) {**

//取当前值。

var icue;

if (attr == 'opacity') {

icue = parseInt(getStyle(obj, attr) \* 100)

} else {

icue = parseInt(getStyle(obj, attr))

};

//算速度。

var iSpeed;

iSpeed = (**json[attr]** - icue) / 8

if (iSpeed > 0) {

iSpeed = Math.ceil(iSpeed)

} else {

iSpeed = Math.floor(iSpeed)

};

//检测停止。

**if (icue != json[attr]) {**

**bStop = false;**

**};**

if (attr == 'opacity') {

obj.style[attr] = (icue + iSpeed) / 100;

} else {

obj.style[attr] = icue + iSpeed + 'px';

}

};

if (bStop) {

clearInterval(obj.timer);

if (fn) {

fn();

};

};

}, 50)

}

注意：①var bStop = true;放在计时器的内部首部。②for (attr in json) {}放在计时器内部中部③if (bStop){}放在计时器的内部尾部。

易错：①iCur = parseInt(getStyle(obj, attr) \* 100);先乘100，再取数值。②if (iCur < json[attr]) {}运动不能写在此条件语句内。③在json中，属性名若无—，则可以不加引号；若有-，则必须加引号,否则ECMAScript无法识别('font-size':200)。④若通过完美运动框架改变绝对定位元素的位置，需先设置绝对定位元素的left和top。

总结： **函数：一清一计 计时器：一声明一循环一判断**

**循环：当前值、算速度、测停止、加运动**

6 运动课程-6

1. 伸缩菜单——同时淡入淡出

<script src="js/move3.js"></script>

<script>

var ul=document.getElementById('ul');

var lis=document.getElementsByTagName('li');

for (n=0;n<lis.length;n++) {

lis[n].onmouseover=function(){

var div=**this.getElementsByTagName('div')[0];**

**div.style.height='auto';**

**var iheight=div.offsetHeight;**

**div.style.height=0;**

startMove(div,{opacity:100,height:iheight})

}

}

for (n=0;n<lis.length;n++) {

lis[n].onmouseout=function(){

var div=this.getElementsByTagName('div')[0];

startMove(div,{opacity:0,height:0})

}

}

</script>

1. 运动框架演变过程：

|  |  |
| --- | --- |
| 框架名称 | 参数 |
| 单物体运动 | startMove(iTarget) |
| 多物体运动 | startMove(obj,iTarget) |
| 任意值运动 | startMove(obj,attr,iTarget) |
| 链式运动 | startMove(obj,attr,iTarget,fn) |
| 多值运动 | startMove(obj,json) |
| 完美运动 | startMove(obj,json,fn) |

1. 布局转换（元素浮动——→绝对定位）

#ul{width: 366px;margin: 0 auto;position: relative;}

#ul li{width: 100px;height: 100px;border: 1px solid black;float: left;margin: 10px;background: #A9A9A9;list-style: none;}

<script>

var ul=document.getElementById('ul');

var lis=ul.getElementsByTagName('li');

//设置每个li绝对定位的同时给每个li设置间距。

**//步骤：先设置位置，再绝对定位。两个for循环不可以合并。**

for (n=0;n<lis.length;n++) {

lis[n].style.left=lis[n].offsetLeft+'px';

lis[n].style.top=lis[n].offsetTop+'px';

};

for (n=0;n<lis.length;n++) {

lis[n].style.position='absolute';

};

</script>

注意：布局转换将**元素浮动**转化为**绝对定位**，将浮动元素容易布局，绝对定位元素容易运动的特点结合起来。

1. Div放大

<script>

var div=document.getElementById('div');

div.onmouseover=function(){ startMove(div,{**width:200,height:200,'margin-left':-50,'margin-top':-50}**)

}

div.onmouseout=function(){ startMove(div,{width:100,height:100,'margin-left':0,'margin-top':0})

}

</script>

注意：margin、top可为负值，padding、width、height不可为负值。

6 运动课程-7

5. 在第3条布局转换的基础上加事件

<script>

var ul=document.getElementById('ul');

var lis=ul.getElementsByTagName('li');

var minZinx=2;

//设置每个li绝对定位的同时给每个li设置间距。

//步骤：先**逐个设置位置，再逐个设置绝对定位。**

for (n=0;n<lis.length;n++) {

lis[n].style.left=lis[n].offsetLeft+'px';

lis[n].style.top=lis[n].offsetTop+'px';

};

for (n=0;n<lis.length;n++) {

lis[n].style.position='absolute';

};

for (n=0;n<lis.length;n++) {

lis[n].onmouseover=function(){

this.style.zIndex=minZinx++;

**//zIndex的I为大写。每次鼠标移入时，当前li的zIndex为minZindex，并且minZinx自增1。**

startMove(this,{width:200,height:200,'margin-left':-50,'margin-top':-50})

};

lis[n].onmouseout=function(){

startMove(this,{width:100,height:100,'margin-left':0,'margin-top':0})

};

};

</script>

1. 新浪微博

将一个li插入到ul的起始位置：

if (lis.length) {ul.insertBefore(li,lis[0])}

else{ul.appendChild(li)};

7.无缝滚动

<style>

\*{margin: 0;padding: 0;}

#div{position: relative;overflow: hidden;border: 1px solid black;width:1060px;height: 220px;margin: 0 auto;}

//div设置宽、高、相对定位、overflow:hidden

#div ul{position:absolute;left: 0;}

//ul设置绝对定位，left。

#div ul li{list-style: none;float: left;width: 245px;height: 200px;padding: 10px;}

//li设置宽、高、左浮动。

</style>

</head>

<body>

<div id="div">

<ul>

<li><img src="../img/1.jpg" /></li>

<li><img src="../img/2.jpg" /></li>

<li><img src="../img/3.jpg" /></li>

<li><img src="../img/4.jpg" /></li>

</ul>

</div>

<script>

var div=document.getElementById('div');

var ul=div.getElementsByTagName('ul')[0];

var lis=ul.getElementsByTagName('li');

var timer=null;

ul.innerHTML+=ul.innerHTML;

ul.style.width=lis[0].offsetWidth\*lis.length+'px'

timer=setInterval(function(){

if (ul.offsetLeft<-ul.offsetWidth/2) {

ul.style.left=0;

}

ul.style.left=ul.offsetLeft-5+'px'

},30)

</script>

</body>

8. 无缝滚动2

<style>

\*{margin: 0;padding: 0;}

#div{position: relative;overflow: hidden;border: 1px solid black;width:1060px;height: 220px;margin: 0 auto;}

/\*div设置相对定位、**宽高**、overflow\*/

#div ul{position:absolute;left: 0;}

/\*ul设置绝对定位、left\*/

#div ul li{list-style: none;float: left;width: 245px;height: 200px;padding: 10px;}

/\*li设置左浮动、**宽高**\*/

</style>

</head>

<body>

<a>←</a>

<a>→</a>

<div id="div">

<ul>

<li><img src="../img/1.jpg" /></li>

<li><img src="../img/2.jpg" /></li>

<li><img src="../img/3.jpg" /></li>

<li><img src="../img/4.jpg" /></li>

</ul>

</div>

<script>

var div=document.getElementById('div');

var ul=div.getElementsByTagName('ul')[0];

var lis=ul.getElementsByTagName('li');

var as=document.getElementsByTagName('a');

var timer=null;

var iSpeed=10

ul.innerHTML+=ul.innerHTML;

//ul中的图片复制

ul.style.width=lis[0].offsetWidth\*lis.length+'px';

//动态设置ul的宽度

div.onmouseover=function(){

clearInterval(timer);

}

div.onmouseout=function(){

**timer=setInterval(fnMove,30);**

}

**timer=setInterval(fnMove,30);**

function fnMove(){

if (**ul.offsetLeft**<-ul.offsetWidth/2) {

ul.style.left=0;

}

if (**ul.offsetLeft**>0) {

ul.style.left=-ul.offsetWidth/2+'px';

}

ul.style.left=ul.offsetLeft+iSpeed+'px';

}

as[0].onclick=function(){

iSpeed=-10;

}

as[1].onclick=function(){

iSpeed=10;

}

</script>

</body>

易错点：①div和li必须设置宽度和高度，否则无法根据li个数动态设置ul宽度。②ul.offsetLeft作为瞬移参考位置，而不是ul.style.left。③初始值的var iSpeed=10要放在计时器函数之外，否则无法改变速度方向。

6 运动课程-8

1. 弹性运动1

<script>

var div=document.getElementById('div');

var btn=document.getElementById('btn');

var iSpeed=0;

btn.onclick=function(){

startMove()

}

function startMove(){

setInterval(function(){

**iSpeed+=(300-div.offsetLeft)/50;**

div.style.left=div.offsetLeft+iSpeed+'px'

},50)

}

</script>

1. 弹性+摩擦

var iSpeed=0;

var left=0;

function startMove(obj,iTarget){

clearInterval(obj.timer);

obj.timer=setInterval(function(){

iSpeed+=(iTarget-left)/5;

iSpeed\*=0.7;

//最佳运动参数：5和0.7

if (**Math.abs(iSpeed)<1&&Math.abs(left-iTarget)<1**) {

//弹性运动终止条件：距离足够近，速度足够小。

clearInterval(obj.timer);

obj.style.left=iTarget+'px';

} else{

**left=left+iSpeed;**

//解决了无法到达指定位置——小数误差的问题。

obj.style.left=left+'px'

}

},50)

}

注意：弹性运动的使用范围不够广泛，运动过界时出错。

6 运动课程-9

1. 弹性运动不适用的地方

var iSpeed=0;

**var height=100;**

function startMove(obj,iTarget){

clearInterval(obj.timer);

obj.timer=setInterval(function(){

iSpeed=iSpeed+(iTarget-**height**)/5;

iSpeed=iSpeed\*0.7;

if (Math.abs(iSpeed)<1&&Math.abs(iTarget-height)<1) {

clearInterval(obj.timer);

obj.style.**height**=iTarget+'px';

} else{

**height=height+iSpeed;**

obj.style.**height**=height+'px';

};

},50)

}

注意：var height的**初始值**要等于**运动之前**元素的高度。

2. 弹性运动框架

var div = document.getElementById('div');

div.onmouseover = function() {

startMove(div, 200)

}

div.onmouseout = function() {

startMove(div, 20)

}

var iSpeed = 0;

var height = 20;

//①速度和初始值必须放在函数之外。②初始值为运动之前物体高度。

function startMove(obj, iTarget) {

clearInterval(obj.timer);

obj.timer = setInterval(function() {

iSpeed = iSpeed + (iTarget - height) / 5;

iSpeed = iSpeed \* 0.7;

//5和0.7为最佳运动效果。

if (Math.abs(iSpeed) < 1 && Math.abs(iTarget - height) < 1) {

//清除计时器的条件：速度足够小并且距离足够近。

clearInterval(obj.timer);

obj.style.height = iTarget + 'px';

} else {

height = height + iSpeed;

//引入变量height，精确存储运动过程中物体动态位 置，解决了无法到达指定位置——小数误差的问题。

if (height < 0) {

height = 0;

}

//解决了回弹过程中高度可能为负（运动过界）的问题。

obj.style.height = height + 'px';

};

}, 50)

};

1. 碰撞运动

function startMove(){

var div=document.getElementById('div');

setInterval(function(){

var l=div.offsetLeft+iSpeedX;

var t=div.offsetTop+iSpeedY;

if (t>**document.documentElement.clientHeight**-div.offsetHeight) {

iSpeedY\*=-1; t=document.documentElement.clientHeight-div.offsetHeight;

**//解决了div碰到边界时出现滚动条的问题。**

};

if (l>document.documentElement.clientWidth-div.offsetWidth) {

iSpeedX\*=-1; l=document.documentElement.clientWidth-div.offsetWidth

};

if (t<0) {

iSpeedY\*=-1;

t=0;

};

if (l<0) {

iSpeedX\*=-1;

l=0;

}

div.style.left=l+'px';

div.style.top=t+'px';

},50)

}

4. 拖拽  
 var div=document.getElementById('div');

div.onmousedown=function(ev){

var oEvent=ev||event;

var disX=oEvent.clientX-div.offsetLeft;

var disY=oEvent.clientY-div.offsetTop;

**document.onmousemove**=function(ev){

var oEvent=ev||event;

var l=oEvent.clientX-disX;

var t=oEvent.clientY-disY;

div.style.left=l+'px';

div.style.top=t+'px';

};

**document.onmouseup**=function(){

**document.onmousemove=null;**

**document.onmouseup=null;**

}

};

注意：onmousedown为div的事件。onmousemove与onmouseup为document的事件。

1. 拖拽+碰撞+运动

<body>

<div id="div"></div>

<script>

var div = document.getElementById('div');

var lastX = 0;

//声明变量，存储每一次拖拽过程中div的X坐标位置。

var lastY = 0;

//声明变量，存储每一个拖拽过程中div的Y坐标位置。

div.onmousedown = function(ev) {

//div的onmousedown事件。（不是document）

var oEvent = ev || event;

//解决浏览器兼容问题。

var disX = oEvent.clientX - div.offsetLeft;

//声明变量，存储鼠标在div上按下时，鼠标距离div边界的距离。

var disY = oEvent.clientY - div.offsetTop;

//声明变量，存储鼠标在div上按下时，鼠标距离div上边界的距离。

document.onmousemove = function(ev) {

//当鼠标div上点击之后，并且在页面上移动时，触发document的onmousemove事件。

var oEvent = ev || event;

var l = oEvent.clientX - disX;

//声明变量l，存储鼠标移动时div的X方向动态位置。

var t = oEvent.clientY - disY;

//声明变量l，存储鼠标移动时div的Y方向动态位置。

div.style.left = l + 'px';

//将div的X方向位置l赋值给div。

div.style.top = t + 'px';

//将div的Y方向的位置t赋值给div。

**iSpeedX = l - lastX;**

//求出每次移动时，div的X方向速度。

**iSpeedY = t - lastY;**

//求出每次移动时，div的Y方向速度。

**lastX = l;**

//每次移动之后，将div在X方向的位置赋值给变量lastX，以便于下次求解速度。

**lastY = t;**

//每次移动之后，将div在Y方向的位置赋值给变量lastY，以便于下次求解速度。

};

document.onmouseup = function() {

//当鼠标div上点击之后，并且在页面上松开鼠标时，触发document的onmouseup事件。

document.onmousemove = null;

//document的onmousemove事件结束，使div移动停止。

document.onmouseup = null;

//div停止之后，document的onmouseup事件结束。

startMove();

//将iSpeedX与iSpeedY的值传递到startMove()函数中，执行函数。

};

clearInterval(timer);

//当鼠标放在div上时，清除定时器，div停止移动。

}

var timer = null;

function startMove() {

clearInterval(timer);

timer = setInterval(function() {

iSpeedY += 3;

//传递重力作用。

var l = div.offsetLeft + iSpeedX;

//声明变量l，存储鼠标移动时div的X方向动态位置。

var t = div.offsetTop + iSpeedY;

//声明变量t，存储鼠标移动时div的Y方向动态位置。

if (t > document.documentElement.clientHeight - div.offsetHeight) {

iSpeedY \*= -0.8;

iSpeedX \*= 0.8

//碰撞下边界之后，Y方向速度取反衰减，X方向速度衰减。

t = document.documentElement.clientHeight - div.offsetHeight;

//解决了div碰到边界时出现滚动条的问题。

};

if (l > document.documentElement.clientWidth - div.offsetWidth) {

iSpeedX \*= -0.8;

iSpeedY \*= 0.8

//碰撞右边界之后，X方向速度取反衰减，Y方向速度衰减。

l = document.documentElement.clientWidth - div.offsetWidth

};

if (t < 0) {

iSpeedY \*= -0.8;

iSpeedX \*= 0.8;

//碰撞上边界之后，Y方向速度取反衰减，X方向速度衰减。

t = 0;

};

if (l < 0) {

iSpeedX \*= -0.8;

iSpeedY \*= 0.8

//碰撞左边界之后，X方向速度取反衰减，Y方向速度衰减。

l = 0;

};

if (Math.abs(iSpeedX) < 1) {

iSpeedX = 0

}; //当X方向速度小于1时，直接停止。

if (Math.abs(iSpeedY) < 1) {

iSpeedY = 0

}; //当Y方向速度小于1时，直接停止。

if (Math.abs(iSpeedX) == 0 && Math.abs(iSpeedY) == 0 && t == document.documentElement.clientHeight - div.offsetHeight) {

clearInterval(timer);

//当X、Y方向速度都为零，并且下落到地面上时，清除计时器。

} else {

div.style.left = l + 'px';

//将div的X方向位置l赋值给div。

div.style.top = t + 'px';

//将div的Y方向位置t赋值给div。

}

}, 50)

}

</script>

</body>

注意：document.body可以定位到body。不可以直接写body。

7 面向对象-1

1. 什么是对象

什么是收音机

对象是一个（封闭）整体，对外提供一些操作。

1. 什么是面向对象

使用对象的时候，只关注对象提供的功能，不关注其内部的细节。

比如：JQuery

注意：JQuery是一个面向对象的东西，但不是用面向对象的方式写成的。

3. 面向对象是一种通用思想，并非只有编程中能用，任何事情都可以用。

老总→部门主管→底层员工

中央→省→市→县

1. JavaScript中的面向对象

面向对象编程（OOP）的特点：①**抽**象，抓住核心问题。②封装，不考虑内部实现方式，只考虑功能使用。（收音机封装起来，留几个头，以供使用）③继承，从已有对象上，继承出新的对象。

传统分类：封装、继承、多态

多态：父类和子类有相同的操作，但是这些操作又不太一样。

1. 对象的组成：

**属性**——变量：状态、静态的。

**方法**——函数：过程、动态的

注意：属性与变量类似，变量是自由（window对象）的，而属性属于一个对象。方法与函数类似，函数是自由（window对象）的，方法属于一个对象。

引申：任何函数都是一个方法，任何属性都是一个变量。

1. 第一个面向对象程序

<script>

var arr=[2,4,6,9]; //创建一个数组对象

arr.show=function(){ //给数组添加一个方法

alert(this.length) //this指代数组arr

}

arr.show()

var odiv=document.getElementById("div")

//其实，给div添加事件，即为给div添加方法。

odiv.onclick=function(){

alert(this) //this指代odiv

}

function demo(){

alert(this) //弹出window，即当前函数为window对象的方法。

}

demo()

var a=10;

alert(window.a); //弹出10,即当前变量属于window对象的属性。

//全局变量是window对象的属性，全局函数是window对象的方法。

//即function demo(){}等价于window.demo=function(){}

**//demo()等价于window.demo()**

</script>

this的新定义：**当前的方法所属于的对象**。只有一种例外情况，**调用函数前加new的时候：系统自动创建一个对象Object，this指向这个对象。**

例子1：<script>

function show(){

alert(this);

}

show() //弹出window

</script>

例子2：<script>

function show(){

**//假想的系统内部工作流程：var this=new Object();**

alert(this);

}

new show() //弹出Object(新创建的对象)

</script>

1. 用工厂方式构造函数

<script>

function createPerson(name,sex){

var obj=**new** Object();

obj.name=name;

obj.sex=sex;

obj.showName=function(){

alert("我的名字叫："+this.name)

}

obj.showSex=function(){

alert("我的性别是："+this.sex)

}

return obj;

}

var person1=createPerson('明明','男');

var person2=createPerson('丽丽','女');

person1.showName();

person1.showSex();

person2.showName();

person2.showSex();

alert(person1.showName==person2.showName) //弹出false。

</script>

工厂方式的问题：①没有new ②每个对象都有一套自己的函数——浪费资源。

解决方法：①new ；②原型。

注意：函数内没有new，函数外有new；函数内有new，函数外没有new。

<script>

function createPerson(name,sex){

**//假想的系统内部工作流程： var this=new Object();**

this.name=name;

this.sex=sex;

this.showName=function(){

alert("我的名字叫："+this.name)

}

this.showSex=function(){

alert("我的性别是："+this.sex)

}

//假想的系统内部工作流程：return this;

}

var person1=new createPerson('明明','男');

person1.showName();

person1.showSex();

</script>

解决了问题1，没有new。问题2仍然存在，每个对象都有一套自己的函数——浪费资源。（见11）

1. 给数组添加方法

<script>

var arr1=[1,2,45,9,23,5]

var arr2=[2,6,3,9,45,2]

arr1.sum=function(){ //给数组1添加方法（用来求和）。

var result=0;

for (n=0;n<this.length;n++) {

//既可以写this，又可以写arr1。

result=result+this[n]

//既可以写this，又可以写arr1。

}

return result;

}

alert(arr1.sum()) //弹出数组arr1各元素和。

alert(arr2.sum()) //数组arr2没有此求和方法，不能求和。

</script>

问题：如何给一类对象统一添加方法。 Prototype（原型）

给一个数组对象添加方法与给数组类统一添加方法的区别，类似于改变唯一一个元素样式和改变同一个class的元素集合样式的区别。

Prototype（原型）的方法与class类型，给一类对象添加属性和方法。

<script>

var arr1=[1,2,45,9,23,5]

var arr2=[2,6,3,9,45,2]

Array.prototype.sum=function(){

//给Array类原型添加方法。

var result=0;

for (n=0;n<this.length;n++) {

**//只能写成this，不能写成Array。**

result=result+this[n]

**//只能写成this，不能写成Array。**

}

return result;

}

alert(arr1.sum()) //弹出数组各元素和。

alert(arr2.sum()) //弹出数组各元素和。

</script>

注意：通过类给原型链添加方法时，方法中的**this为实例对象**，而不是类。

9. 类与对象（实例）

类——模子 ；对象——蛋糕

var arr=new Array();

Array类 不具备实际的功能，只能用来构造对象。

arr对象 真正有功能的东西，被类给构造出来。

arr为从Array类中new出来的对象。

1. 原型的一个重要的功能（应用）——可以扩展系统对象的属性和方法。
2. 给原型添加的方法不会有多个副本，不会浪费资源，所有的对象只有一套。

例子1：<script>

var arr1=[1,2,45,9,23,5]

var arr2=[2,6,3,9,45,2]

Array.prototype.sum=function(){

//给数组添加方法（用来求和）。

var result=0;

for (n=0;n<this.length;n++) {

**//只能写成this，不能写成Array。**

result=result+this[n]

//只能写成this，不能写成Array。

}

return result;

}

alert(arr1.sum==Array.prototype.sum) //返回true。

alert(arr1.sum==arr2.sum) //返回true。

</script>

例子2：<script>

function createPerson(name,sex){

this.name=name; //属性

this.sex=sex;

}

createPerson.prototype.showName=function(){

alert("我的名字叫："+this.name)

}

createPerson.prototype.showSex=function(){

alert("我的性别是："+this.sex)

}

var person1=new createPerson('明明','男');

var person2=new createPerson('丽丽','女');

person1.showName();

person1.showSex();

person2.showName();

person2.showSex();

alert(person1.showName==person2.showName) //弹出true。

</script>

注意：类与构造函数是两个截然不同的概念，但是在JavaScript中，没有必要区分两者。

1. 构造函数即是类，类即是构造函数。

<script>

alert(typeof Date);//弹出function。

alert(typeof Array);//弹出function。

</script>

1. 用原型Prototype给类的实例对象添加方法的优点：简化构造函数；类中所有对象共用一套方法，节省系统资源。
2. 构造函数首字母大写（包括第一个单词）。

在构造函数中给类添加属性，因为类中**每一个对象的属性不同**；在原型链中给类添加方法，因为类中**每一个对象的方法相同**。

<script>

function Person(name,sex){ //混合法创建类（构造函数）。

this.name=name; //在构造函数中给类添加属性。

this.sex=sex;

}

Person.prototype.showName=function(){

//在原型链中给类添加方法。

alert(this.name);

}

Person.prototype.showSex=function(){

alert(this.sex);

}

var p=new Person('blue','男');

p.showName();

p.showSex();

</script>

1. 原型的优先级

无论先后顺序，给对象直接添加属性和方法的优先级高于在原型链中给类添加属性和方法。

<script>

Array.prototype.a=12;//在原型链中给类添加属性。

var arr=[1,2,3];

alert(arr.a);//弹出12。

arr.a=5; //给对象添加属性。

alert(arr.a);//弹出5。

</script>

<script>

var arr=[1,2,3];

arr.a=5;//给对象添加属性。

alert(arr.a);//弹出5。

Array.prototype.a=12;//在原型链中给类添加属性。

alert(arr.a);//弹出5。

</script>

7 面向对象-3

1. 实例：面向对象的选项卡

把面向过程的程序，改写成面向对象的形式。

**待做**

7 面向对象-4

1. Json方式的面向对象

注意：在JavaScript中出现的所有东西，都可以放入Jason。

<script>

var obj={a:2,b:[5,6,7],c:function(){

alert(this.b)

}}

obj.c() //弹出5,6,7

</script>

例子：<script>

var p1={

name:'blue',

sex:'男',

showName:function(){

alert("我的名字是"+this.name);**//this不可省略。**

},

showSex:function(){

alert("我的性别是"+this.sex); **//this不可省略。**

}

}

p1.showName()

</script>

注意：Json 能创建对象，而构造函数可以创建类。

1. 把方法放在一个Json里

有人把他叫——命名空间。

1. 拖拽和继承

**拖拽/待学**

1. 继承

对象组成：①属性；②方法

继承对象：①属性；②方法

继承：先执行父级的构造函数，然后再添加子类的属性。

<script>

function Person(name,sex){

this.name=name;

this.sex=sex;

}

Person.prototype.showNane=function(){

alert(this.name);

}

Person.prototype.showSex=function(){

alert(this.sex);

}

function Worker(name,sex,job){

//this为new出来的Worker对象ow1

**Person.call(this,name,sex);**

//构造函数伪装：调用父级的构造函数，为了继承属性。

this.job=job;

}

**Worker.prototype=Person.prototype;**

//通过原型来继承父级的方法。但这种方式也会改变父类的原型链。

Worker.prototype.showJob=function(){

alert(this.job)

}

var ow1=new Worker('blue','男','打杂的')

ow1.showJob()

alert(Person.prototype.showJob) //不是undefined。

</script>

注意：继承的优点，修改父级的bug，子级自动修改。

**方法.call(对象，实参1，实参2……)**

5.在JavaScript中，所有的对象都是引用。

<script>

var arr1=[1,2,3];

var arr2=arr1; //引用：数组arr1与arr2占用同一个内存空间。

arr2.push(4)

alert(arr1); //弹出1，2，3，4

alert(arr2);//弹出1，2，3，4

</script>

对比： <script>

var arr1=[1,2,3];

var arr2=[];

for (i in arr1) {

arr2[i]=arr1[i]

}

arr2.push(4)

alert(arr1); //弹出1，2，3

alert(arr2);//弹出1，2，3，4

</script>

1. 继承2

<script>

function Person(name,sex){

this.name=name;

this.sex=sex;

}

Person.prototype.showNane=function(){

alert(this.name);

}

Person.prototype.showSex=function(){

alert(this.sex);

}

function Worker(name,sex,job){

//this为new出来的Worker对象ow1

**Person.call(this,name,sex);**

**//构造函数伪装：调用父级的构造函数，继承父级属性。**

this.job=job; //添加自己的属性。

}

**for (i in Person.prototype) {**

**Worker.prototype[i]=Person.prototype[i]**

**} //通过原型链继承父级的方法，而避免引用。**

Worker.prototype.showJob=function(){

alert(this.job) //添加自己的方法。

}

var per1=new Person('red','女')

var ow1=new Worker('blue','男','打杂的')

per1.showNane();

per1.showSex();

ow1.showNane();

ow1.showSex();

ow1.showJob();

alert(Person.prototype.showJob) //是undefined。

</script>

步骤：继承父级属性（构造函数伪装），添加自己的属性，继承父级的方法（for in 方法避免引用），添加自己的方法。

7. instance of 判断某个对象是否是某个类的实例。

<script>

var arr=[1,23,4]

alert(arr instanceof Array) //弹出true。

alert(arr instanceof Date) //弹出false。

alert(arr instanceof Object) //弹出true。

</script>

注意：所有类都是Object的一个子类。

1. 系统对象

①本地对象（非静态对象）

什么是本地对象

常用对象：Object、Function、Array、Atring、Boolean、Number、Date、RegExp、Error

**需要实例化**，只能用new出来的。——非静态对象

②内置对象（静态对象）

什么是静态对象

Global、Math

**不需要实例化**（new）——静态对象

③宿主对象（由浏览器提供的对象），直接可以用。

DOM、BOM document和window

8 DOM高级

1. 表格应用

在做表格table时，即使不用tbody包裹tr，浏览器在解析时，仍会给所有的tr外套一个tbody。为避免混乱，在做表格时需要加上tbody。

alert(tab.getElementsByTagName(**'tbody'**)[0].getElementsByTagName(**'tr'**)[2].getElementsByTagName(**'td'**)[1].innerHTML)等价于

alert(tab.**tBodies**[0].**rows**[2].**cells**[1].innerHTML)。

注意：**table.tBodies[0]为固定写法，不可以省略tBodies[0]。**

2. 表格隔行变色+移入高亮

<script>

window.onload=function(){

var oldBgColor='';

var i=0;

var tab=document.getElementById('tab1');

for (i=0;i<tab.tBodies[0].rows.length;i++) {

**tab.tBodies[0].rows[i].style.background=i%2?'#888':''**

//若未加Tbodies，则从表头中的tr开始算。

tab.tBodies[0].rows[i].onmouseover=function(){

**oldBgColor=this.style.background;**

//在变色之前，先用一个变量存储变色之前的颜色。

this.style.background='yellow'

};

tab.tBodies[0].rows[i].onmouseout=function(){

this.style.background=oldBgColor;

};

};

};

</script>

注意：移入时先存颜色再变色，移出时变回原色。

3. 表格添加

<script>

window.onload=function(){

var tab=document.getElementById('tab1');

var tex=document.getElementById('tex1');

var btn=document.getElementById('btn');

**var iNowId=tab.tBodies[0].rows.length+1;**

btn.onclick=function(){

var tr=document.createElement('tr');

var td=null;

td=document.createElement('td');

**td.innerHTML=iNowId++;**

//解决了删除表格某一行之后，新添加行的序号发生重复的情况。

tr.appendChild(td);

td=document.createElement('td');

td.innerHTML=tex.value;

tr.appendChild(td);

td=document.createElement('td');

td.innerHTML="<a href='#'>删除</a>";

tr.appendChild(td);

td.getElementsByTagName('a')[0].onclick=function(){

tab.tBodies[0].removeChild(this.parentNode.parentNode)

}

**tab.tBodies[0].appendChild(tr);**

//tobies[0]一定不能省。

};

};

</script>

1. 表格搜索1

<script>

window.onload=function(){

var tex=document.getElementById('tex1');

var btn=document.getElementById('btn');

var tab=document.getElementById('tab1');

btn.onclick=function(){

for (i=0;i<tab.tBodies[0].rows.length;i++) {

if (tab.tBodies[0].rows[i].cells[1].innerHTML==tex.value) {

tab.tBodies[0].rows[i].style.background='yellow'

}**else{**

**tab.tBodies[0].rows[i].style.background=''**

};

};

};

};

</script>

8 DOM高级-2

5. 表格搜索2

<script>

window.onload=function(){

var tex=document.getElementById('tex1');

var btn=document.getElementById('btn');

var tab=document.getElementById('tab1');

btn.onclick=function(){

for (i=0;i<tab.tBodies[0].rows.length;i++) {

var valueInTab=tab.tBodies[0].rows[i].cells[1].innerHTML.toLowerCase();

//忽略大小写。

var valueInTex=tex.value.toLowerCase();

var arr=valueInTex.split(' ');

**var bFound=false;**

//默认没有匹配项。

for (n=0;n<arr.length;n++) {

if (valueInTab.search(arr[n])!=-1) {

**bFound=true;**

//如果表格中有搜索项，则bFound为true。

break;

};

};

if (bFound) {

tab.tBodies[0].rows[i].style.background='yellow'

}else{

tab.tBodies[0].rows[i].style.background=''

//在被选行变色的同时，让其他行变回原色。

};

};

};

};

</script>

1. 移动li-1

<script>

window.onload=function(){

var btn=document.getElementById('btn');

var ul1=document.getElementById('ul1');

var ul2=document.getElementById('ul2');

var lis=ul1.getElementsByTagName('li');

btn.onclick=function(){

ul2.appendChild(lis[0]);

};

};

</script>

1. 移动li-2

<script>

window.onload=function(){

var btn=document.getElementById('btn');

var ul1=document.getElementById('ul1');

var lis=ul1.getElementsByTagName('li');

btn.onclick=function(){

ul1.appendChild(lis[0]);

};

};

</script>

8. 排 序li

<script>

window.onload=function(){

var btn=document.getElementById('btn');

var ul1=document.getElementById('ul1');

var lis=ul1.getElementsByTagName('li');

btn.onclick=function(){

var arr=[];

**//1.将lis转化为数组。**

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

arr[i]=lis[i];

};

**//2.数组排序。**

arr.sort(**function(a,b){**

**return parseInt(a.innerHTML)-parseInt(b.innerHTML)**

**}**);

**//3. 重新插入**

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

ul1.appendChild(arr[i]);

};

};

};

</script>

注意：深刻理解排序规则函数的书写思路。

9. 表格排序

<script>

window.onload=function(){

var tab=document.getElementById('tab');

var btn=document.getElementById('btn');

btn.onclick=function(){

var i=0;

var arr=[];

//1.转成数组

for (i=0;i<tab.tBodies[0].rows.length;i++) {

arr[i]=tab.tBodies[0].rows[i];

}

//2.数组排序

arr.sort(function(a,b){

return parseInt(a.cells[0].innerHTML)-parseInt(b.cells[0].innerHTML)

})

//3.重新插入

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

tab.tBodies[0].appendChild(arr[i]);

};

};

};

</script>

注意：①在JavaScript中，所有排序问题都会用到sort。②appendChild有两个步骤，先从原父级移出，再添加到新的父级。③转成数组—数组排序—重新插入。  
10. 表格排序2

<script>

window.onload=function(){

var tab=document.getElementById('tab');

var btn=document.getElementById('btn');

**var abc=true;//是否是升序排列。**

btn.onclick=function(){

var i=0;

var arr=[];

//1.转成数组

for (i=0;i<tab.tBodies[0].rows.length;i++) {

arr[i]=tab.tBodies[0].rows[i];

}

//2.数组排序

arr.sort(function(a,b){

if (abc) {

return parseInt(a.cells[0].innerHTML)-parseInt(b.cells[0].innerHTML)

} else{

return parseInt(b.cells[0].innerHTML)-parseInt(a.cells[0].innerHTML)

};

})

//3.重新插入

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

tab.tBodies[0].appendChild(arr[i]);

};

**abc=!abc;**

};

};

</script>

1. 表单应用

<script>

window.onload = function() {

var form = document.getElementById('form1');

var user = document.getElementsByName('username')[0];

var pass = document.getElementsByName('pass')[0];

**form.onsubmit** = function() {

if (user.value == '' || pass.value == '') {

alert('您填错了');

**return false;**

};

};

**form.onreset**=function(){

**return window.confirm('是否要清空')**

};

};

</script>

注意：**onsubmit**和**onreset**是表单(form)所特有的事件，在表单提交/重置之前执行一次。

12. 表单内容校验

阻止用户输入非法字符——阻止事件（不常用）

输入时、失去焦点时验证——onkeyup、onblur

提交时检查——**onsubmit**

\*后台数据检查

1. 表格——元素获取；隔行变色、高亮显示；添加、删除；排序

表单——常用属性；事件；校验

注意：tBodies是集合，而tHead和tFoot不是集合。

8 PHP后台-1

8 PHP后台-2

8 封装自己的JS库：VQuery-1

8 封装自己的JS库：VQuery-2

8 封装自己的JS库：VQuery-3

8 封装自己的JS库：VQuery-4

8 高级桌面应用1

8 高级桌面应用2

8 事件高级应用3

1. 事件绑定框架

function myAddEvent(obj,sEvent,fn){

if (obj.attachEvent) {

obj.attachEvent(**'on'+sEvent**,fn);

}else{

obj.addEventListener(sEvent,fn);

};

}

2. 绑定事件与this

<button id="btn">按钮</button>

<script>

var btn=document.getElementById('btn');

btn.onclick=function(){

alert(this); //Button

}

btn.addEventListener('click',function(){

alert(this);**//Button**

});

btn.attachEvent('onclick',function(){

alert(this);**//Window**

})

</script>

3. 事件绑定与匿名函数

<button id="btn">按钮</button>

<script>

function aaa(){

alert('aaa')

var btn=document.getElementById('btn');

btn.addEventListener('click',aaa)

btn.removeEventListener('click',aaa)

//与添加句柄时是一个函数。点击按钮，不会弹出aaa。

</script>

<button id="btn">按钮</button>

<script>

var btn=document.getElementById('btn');

btn.addEventListener('click',function(){

alert('aaa')

})

btn.removeEventListener('click',function(){

alert('aaa')

//与添加句柄时不是一个函数，new了两次。

//点击按钮弹出aaa。

})

</script>

注意：var a=function (){alert(‘aaa’)}

是var a=new Function(“alert(‘a’)”)的简写。

绑定匿名函数，无法删除。

4. 拖拽封装函数

function drag(**id**) {

var div = document.getElementById(**id**);

div.onmousedown = function(ev) {

var oEvent = ev || event;

var disX = oEvent.clientX - div.offsetLeft;

var disY = oEvent.clientY - div.offsetTop;

document.onmousemove = function(ev) {

var oEvent = ev || event; //不可以省略。

l = oEvent.clientX - disX;

t = oEvent.clientY - disY;

div.style.left = l + 'px';

div.style.top = t + 'px';

};

document.onmouseup = function() {

document.onmousemove = null;

document.onmouseup = null;

};

};

};

注意：仅修改了**两处**，即完成函数的封装。

8 事件高级应用2

1. 拖拽-限制范围-磁性吸附

<script>

window.onload=function(){

var div=document.getElementById('div');

div.onmousedown=function(ev){

var oEvent=ev||event;

var disX=oEvent.clientX-div.offsetLeft;

var disY=oEvent.clientY-div.offsetTop;

document.onmousemove=function(ev){

var oEvent=ev||event;//不可以省略。

l=oEvent.clientX-disX;

t=oEvent.clientY-disY;

if (l>document.documentElement.clientWidth-div.offsetWidth-50) {

l=document.documentElement.clientWidth-div.offsetWidth

};

if (l<50) {

l=0;

};

if (t>document.documentElement.clientHeight-div.offsetHeight-50) {

t=document.documentElement.clientHeight-div.offsetHeight

};

if (t<50) {

t=0;

};

div.style.left=l+'px';

div.style.top=t+'px';

};

document.onmouseup=function(){

document.onmousemove=null;

document.onmouseup=null;

}

}

}

</script>

1. 拖拽-问题

<script>

window.onload=function(){

var div=document.getElementById('div');

div.onmousedown=function(ev){

var oEvent=ev||event;

var disX=oEvent.clientX-div.offsetLeft;

var disY=oEvent.clientY-div.offsetTop;

document.onmousemove=function(ev){

var oEvent=ev||event;//不可以省略。

l=oEvent.clientX-disX;

t=oEvent.clientY-disY;

div.style.left=l+'px';

div.style.top=t+'px';

};

document.onmouseup=function(){

document.onmousemove=null;

document.onmouseup=null;

};

**return false;**

//解决拖拽过程中选中文字的问题(必须写在尾部)。

};

};

</script>

3.拖拽-碰撞检测

<script>

window.onload=function(){

var div1=document.getElementById('div1');

var div2=document.getElementById('div2');

div1.onmousedown=function(ev){

var oEvent=ev||event;

var disX=oEvent.clientX-div1.offsetLeft;

var disY=oEvent.clientY-div1.offsetTop;

document.onmousemove=function(ev){

var oEvent=ev||event;//不可以省略。

l=oEvent.clientX-disX;

t=oEvent.clientY-disY;

div1.style.left=l+'px';

div1.style.top=t+'px';

var l1=div1.offsetLeft;

var r1=div1.offsetLeft+div1.offsetWidth;

var t1=div1.offsetTop;

var b1=div1.offsetTop+div1.offsetHeight;

var l2=div2.offsetLeft;

var r2=div2.offsetLeft+div2.offsetWidth;

var t2=div2.offsetTop;

var b2=div2.offsetTop+div2.offsetHeight;

if (**r1<l2||l1>r2||b1<t2||t1>b2**) {

div2.style.background='yellow';

}else{

div2.style.background='blue';

}

};

document.onmouseup=function(){

document.onmousemove=null;

document.onmouseup=null;

}

}

}

</script>

4. 带框的拖拽

<script>

window.onload=function(){

var div=document.getElementById('div');

div.onmousedown=function(ev){

var oEvent=ev||event;

var disX=oEvent.clientX-div.offsetLeft;

var disY=oEvent.clientY-div.offsetTop;

**var newDiv=document.createElement('div');**

newDiv.className='box';

**newDiv.style.width=div.offsetWidth-2+'px';**

**newDiv.style.height=div.offsetHeight-2+'px';**

newDiv.style.left=div.offsetLeft+'px';

newDiv.style.top=div.offsetTop+'px';

**document.body.appendChild(newDiv);**

document.onmousemove=function(ev){

var oEvent=ev||event;//不可以省略。

l=oEvent.clientX-disX;

t=oEvent.clientY-disY;

newDiv.style.left=l+'px';

newDiv.style.top=t+'px';

};

document.onmouseup=function(){

document.onmousemove=null;

document.onmouseup=null;

div.style.left=**newDiv.offsetLeft**+'px';

div.style.top=**newDiv.offsetTop**+'px'

**document.body.removeChild(newDiv)**

};

};

};

</script>

5. 拖拽改变大小

<script>

window.onload=function(){

var div=document.getElementById('div');

div.onmousedown=function(ev){

var oEvent=ev||event;

var disX=oEvent.clientX-div.offsetLeft;

var disY=oEvent.clientY-div.offsetTop;

document.onmousemove=function(ev){

var oEvent=ev||event;

l=oEvent.clientX-disX;

t=oEvent.clientY-disY;

**div1.style.width=l+div.offsetWidth+'px';**

**div1.style.height=t+div.offsetHeight+'px';**

};

document.onmouseup=function(){

document.onmousemove=null;

document.onmouseup=null;

}

}

}

</script>

</head>

<body>

<div id="div1">

<div id="div"></div>

</div>

</body>

8 事件高级应用3

1. 自定义滚动条

<style>

#parent{width: 400px;height: 20px;background: #ccc;position: relative;margin: 20px auto;}

#div{width: 20px;height: 20px;background: red;position: absolute;cursor: pointer;}

#div2{width: 200px;height: 300px;border: 1px dashed black;position: relative;overflow: hidden;}

#div3{position: absolute;}

</style>

<script>

window.onload=function(){

var div=document.getElementById('div');

var parent=document.getElementById('parent');

var div2=document.getElementById('div2');

var div3=document.getElementById('div3');

div.onmousedown=function(ev){

var oEvent=ev||event;

var disX=oEvent.clientX-div.offsetLeft;

document.onmousemove=function(ev){

var oEvent=ev||event;//不可以省略。

l=oEvent.clientX-disX;

if (l<0) {

l=0

};

if (l>parent.offsetWidth-div.offsetWidth) {

l=parent.offsetWidth-div.offsetWidth;

}

div.style.left=l+'px';

var scale=l/(parent.offsetWidth-div.offsetWidth); **div3.style.top=-(div3.offsetHeight-div2.offsetHeight)\*scale+'px'**

};

document.onmouseup=function(){

document.onmousemove=null;

document.onmouseup=null;

}

}

}

</script>

</head>

<body>

<div id="parent">

<div id="div"></div>

</div>

<div id="div2">

<div id="div3">

大量文字

</div>

</div>

</body>

1. 普通事件：onclick、onmousedown……

DOM事件：DOMMouseScroll (只能通过事件侦听来添加)(火狐浏览器)

1. 鼠标滚轮

<script>

function myAddEvent(obj,sEvent,fn){

if (obj.attachEvent) {

obj.attachEvent('on'+sEvent,fn);

} else{

obj.addEventListener(sEvent,fn);

}

}

window.onload=function(){

var oDiv=document.getElementById('div1');

function onMouseWheel(ev){

var oEvent=ev||event

var bDown=true;

//IE支持wheelDelta：下负上正

//FF支持detail：下正上负**bDown=oEvent.wheelDelta?oEvent.wheelDelta<0:oEvent.detail>0;**

alert(bDown);

}

myAddEvent(oDiv,'mousewheel',onMouseWheel);

myAddEvent(oDiv,'DOMMouseScroll',onMouseWheel);

/\***oDiv.onmousewheel=function(){**

**alert('a');**

**}**\*/

/\***oDiv.addEventListener('DOMMouseScroll',function(){**

**alert('a')**;//只有火狐支持。

})\*/

}

</script>

注意：一切事件的详细信息都包含在事件对象里。

1. 鼠标滚轮2

<script>

function myAddEvent(obj,sEvent,fn){

if (obj.attachEvent) {

obj.attachEvent('on'+sEvent,fn);

} else{

obj.addEventListener(sEvent,fn);

};

};

window.onload=function(){

var oDiv=document.getElementById('div1');

function onMouseWheel(ev){

var oEvent=ev||event

var bDown=true;

bDown=oEvent.wheelDelta?oEvent.wheelDelta<0:oEvent.detail>0;

if (bDown) {

oDiv.style.height=oDiv.offsetHeight+10+'px';

}else{

oDiv.style.height=oDiv.offsetHeight-10+'px';

};

**if (oEvent.preventDefault) {**

**oEvent.preventDefault()**

**};//只有火狐浏览器支持。**

**return false;**

//取消系统自带的滚轮条随滚轮滚动的效果。

};

myAddEvent(oDiv,'mousewheel',onMouseWheel);

myAddEvent(oDiv,'DOMMouseScroll',onMouseWheel);

};

</script>

1. 自定义滚动条+鼠标滚轮（难点）

<head>

<meta charset="UTF-8">

<title></title>

<style>

#parent {

width: 400px;

height: 20px;

background: #ccc;

position: relative;

margin: 20px auto;

}

#div {

width: 20px;

height: 20px;

background: red;

position: absolute;

cursor: pointer;

}

#div2 {

width: 200px;

height: 300px;

border: 1px dashed black;

position: relative;

overflow: hidden;

}

#div3 {

position: absolute;

}

</style>

<script>

function myAddEvent(obj, sEvent, fn) {

if (obj.attachEvent) {

obj.attachEvent('on' + sEvent, fn);

} else {

obj.addEventListener(sEvent, fn);

};

};

window.onload = function() {

var div = document.getElementById('div');

var parent = document.getElementById('parent');

var div2 = document.getElementById('div2');

var div3 = document.getElementById('div3');

function onMouseWheel(ev) {

var oEvent = ev || event

var bDown = true;

bDown = oEvent.wheelDelta ? oEvent.wheelDelta < 0 : oEvent.detail > 0;

if (bDown) {

setLeft(div.offsetLeft + 10);

} else {

setLeft(div.offsetLeft - 10);

};

if (oEvent.preventDefault) {

oEvent.preventDefault()

}; //只有火狐浏览器支持。

return false;

//取消系统自带的滚轮条随滚轮滚动的效果。

};

myAddEvent(parent, 'mousewheel', onMouseWheel);

myAddEvent(parent, 'DOMMouseScroll', onMouseWheel);

myAddEvent(div2, 'mousewheel', onMouseWheel);

myAddEvent(div2, 'DOMMouseScroll', onMouseWheel);

div.onmousedown = function(ev) {

var oEvent = ev || event;

var disX = oEvent.clientX - div.offsetLeft;

document.onmousemove = function(ev) {

var oEvent = ev || event; //不可以省略。

l = oEvent.clientX - disX;

setLeft(l)

};

document.onmouseup = function() {

document.onmousemove = null;

document.onmouseup = null;

};

};

function setLeft(l) {

if (l < 0) {

l = 0

};

if (l > parent.offsetWidth - div.offsetWidth) {

l = parent.offsetWidth - div.offsetWidth;

}

div.style.left = l + 'px';

var scale = l / (parent.offsetWidth - div.offsetWidth);

div3.style.top = -(div3.offsetHeight - div2.offsetHeight) \* scale + 'px'

}

}

</script>

</head>

<body>

<div id="parent">

<div id="div"></div>

</div>

<div id="div2">

<div id="div3">

大量文字

</div>

</div>

</body>

8 正则表达式-1

1. 什么是正则表达式——强大的**字符串匹配**工具

2. 字符串操作：**search**——返回要查找的字符串第一次出现的位置，如果没有则返回**-1**。

<script>

var str='12abcde345';

alert(str.search('ab')) //弹出2

</script>

3. 完全可以不用正则表达式取完成任何字符串操作。

4.正则表达式（规则表达式）能做什么：操作字符串。

5. 正则表达式的两种写法：

<script>

var str='abcdefg';

var re1=new RegExp('A');

var re2=new RegExp('B','i');

var re3=/C/;

var re4=/D/i;

alert(re1.**test**(str)); //弹出false。

alert(re2.test(str)); //弹出true。

alert(re3.test(str)); //弹出false。

alert(re4.test(str)); //弹出true。

</script>

RegExp对象：Js风格——new RegExp('a','i')

Perl风格——/a/i

Perl:第一个引入正则表达式的语言。

1. 字符串与正则配合（1）

\d在正则中表示**一个**数字——digit

<script>

var str='sadsa &\*4fdsa235'

var re=/\d/;

alert(str.search(re));//弹出8。即第一个数字的位置。

</script>

结论：在字符串中可以搜索正则。

1. 字符串match（匹配）与正则结合

字符串内容匹配的方法：**match(“匹配的内容”)**

如果字符串中有该内容，则返回该匹配内容，如果字符串中没有该内容，则返回null。

例子1：<script>

var a="hello world"

document.write(a.match("word"))

</script> //返回null。

例子2 <script>

var str='sadsa &\*4fdsa235'

var re=/\d/;

alert(str.match(re));//弹出4。即第一个数字。

</script>

例子3 <script>

var str='sadsa &\*4fdsa235'

var re=/\d/g;

alert(str.match(re));//弹出4,2,3,5。即所有的数字。

</script>

例题4 <script>

var str='sadsa &\*4fdsa235'

var re=/\d+/g;

alert(str.match(re));//弹出4,235。即所有的数字。

</script>

注意：+号表示许多的意思，个数不定，**至少一个**。

1. 字符串replace（替换）与正则结合

语法：替换内容的方法：**replace(“原内容”，“新内容”)**

replace有两个含义：①不改变原字符串；②只替换第一个。

例题： <script>

var str='aAaAaa'

var re=/a/**gi**;

alert(str.replace(re,'T'));//弹出TTTTTT。

</script>

例题2：<textarea rows="20" cols="20" id="tex1"></textarea>

<button id="btn">按钮</button>

<textarea rows="20" cols="20" id="tex2"></textarea>

<script>

var tex1=document.getElementById("tex1");

var tex2=document.getElementById("tex2");

var btn=document.getElementById("btn");

btn.onclick=function(){

var re=/我们|你们|他们/g;

tex2.value=tex1.value.replace(re,'\*\*');

}

</script>

注意：正则表达式可以用|表示，表示或的意思,|表示**字符串的或**，[abc]中的内容表示单个**字符的或**。

字符串操作：**search——寻找位置；match——匹配子串；replace——替换子串。**

1. 字符类（方括号）

**任意字符——[abc];范围——[a-z]、[0-9];排除——[^a]**

例题1：

<script>

var str='1a21c231b2adf';

var re=/1[abc]2/g;

alert(str.match(re)); 弹出1a2,1c2,1b2。

</script>

方括号的第一种用法：**或**。

例如： [a,b]——意思为**一个字符**或者a**或者逗号**或者b。

[123 456]意思为单个字符或者1或者2或者3或者空格或者……

方括号的的第二种用法：**到**。

例如： [0-9]表示任意**一个**数字。[a-z]表示任意**一个**字母。[0-9a-z]表示任意**一个**字母或者数字。

\d的等价于[0-9]

方括号的第三种用法：**除**。

例如：[^a]表示除了a意外任意**一个**字符。[^0-9a-z]表示除数字和字母之外的任意**一个**字符。

注意：**到**的优先级高于**或**，**或**的优先级高于**非**。即：[^0-9a-z]表示[^((0-9)(a-z))]。

10.读小说案例——自定义innerText

方案1： var re=/<.+>/g; ——不可行，因为会把所有内容过滤掉。

注意：**.表示任意字符，+表示任意个数**。.+表示任意个任意字符。

方案2：var re=/**<[^<>]+>**/g;——可行，删除所有不包含左右尖括号的左右尖括号。<>表示<或者>，[^<>]表示一个既不是<又不是>的字符，[^<>]+表示多个即不是<又不是>的字符。<[^<>]+>表示**一个**<字符加上**多个**即不是<又不是>的字符加上**一个**>字符。

1. **转义字符**

**.(点)——任意一个字符**； \d——一个数字（digital）； \w——[a-z0-9\_] word；

\s——空白字符（空格、制表符）space；\D——一个除了数字；\W——一个除了word；\S——一个除了空白字符。

**12. 量词——出现的次数**

{n,m}——最少n次，最多m次。 {n,}最少n次，最多不限。

{,m}最少不限，最多m次。 {n}正好n次。

**+等价于{1,} \*等价于{0,}（\*尽量少用）** ？等价于{0,1}  
13.校验邮箱

在正则中，.表示任意字符，**\.表示.**。

邮箱：一串字符、数字、下划线 @ 一串字母、数字 . 一串英文（长度是2-4）

\w+ @ [0-9a-z]+ \. [a-z]{2,4}

例题：<input type="text" id="tex" />

<button id="btn">按钮</button>

<script>

var tex=document.getElementById("tex");

var btn=document.getElementById("btn");

re=/^\w+@[0-9a-z]+\.[a-z]{2,4}$/;

//必须加^和$，否则当字符串出现正则re时，即返回true。

btn.onclick=function(){

if (re.**test**(tex.value)) {

alert("对了");

} else{

alert("输入错误")

}

}

</script>

注意：^放在[]中表示非的意思，放在//中的起始位置表示行首，$放在//的结束位置表示行尾。一般做表单校验，都要加行首和行尾。

1. 去掉首尾空格

方法1： <script>

var str=' 223 adc d '

re=/^\s+**|**\s+$/g;

alert("("+str.replace(re,'')+")")

</script>

方法2： <script>

var str=' 223 adc d '

re=/\S.+\S/g;

alert("("+str.match(re)+")")

</script>

1. 检测中文：[\u4e00-\u9fa5]

<script>

var str="文字";

var re=/[\u4e00-\u9fa5]/g;

alert(re.test(str)); //返回true。

</script>

1. 完美版getByClass

在正则中，\b表示单词边界。

例题： <script>

var str="hello word";

var re=/\bhello\b/;

alert(re.test(str));

</script>

正则总结：

A：字符串操作的方法：search、substring、charAt、split。

B：i忽略大小写，g全局匹配（找所有的）。