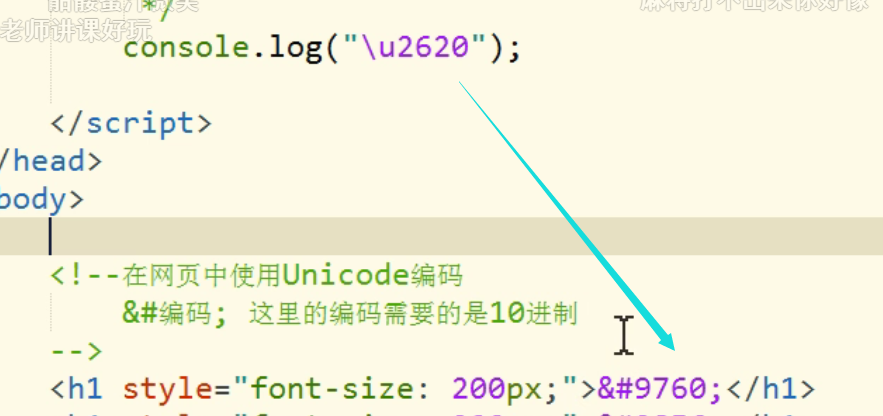
JS

Conslole.log(“\uc0001”);

\u+unicode可以输出万国码的字符

在网页中则是使用“&#”加Unicode转十进制的结果



Isnan（）；函数可以判断一个数是不是nan，nan和任何东西都不相等，包括他自己

相等运算符一般都转换为number类型再来比较，比如true==1

而

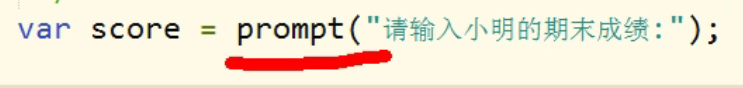
===

是全等运算符，如果类型都不同的话，直接返回false，类型相同且值相同返回true

Nan和undefined相等但是不全等

！==

不全等运算符，同理，类型不同直接返回true，类型相同且值相同返回false



Prompt在js里是输入的函数；

180和字符串比大小，先把字符串转换成数字再比较。“hello”转成数字类型是nan，nan和任何数比都是false

isNaN() 函数用于检查其参数是否是非数字值。

如果参数值为 NaN 或字符串、对象、undefined等非数字值则返回 true, 否则返回 false。

Prompt输入的即使是数字，得出来也是字符串类型型

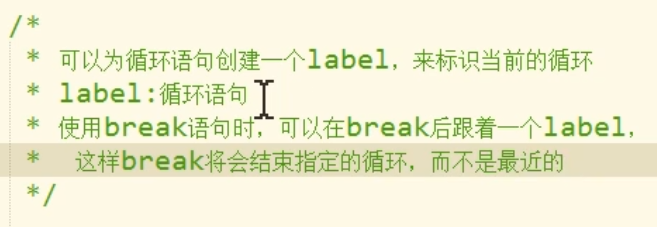
var a = prompt("输入你的数字");

        var b = parseInt(a);

JS里除法这种运算经常不准，要parseint才能舍弃小数位，保留整数位

\*&nbsp； 在JS里是一个空格的意思

<br /> 在JS里是换行的意思（22.2.28号，这不是CSS里的吗）



这样break可以结束指定的循环，可能有妙用

Continue也可以这么用

对象

增

Var obj = new Object（）；

obj.name=”TOM”;

删

Delete obj.name;

改

Obj.name=”孙悟空”;

查

Console.log(obj.name);

对象的属性值可以是任意的数据类型，甚至可以是一个对象

in运算符

语法：“属性名” in 对象 如果存在则返回true 否则false

基本数据类型和引用数据类型的区别

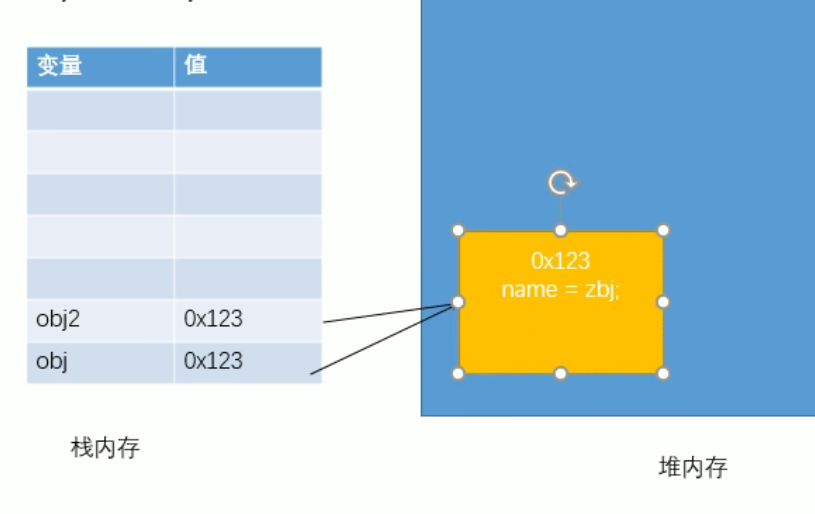
基本数据类型 a=3 b=a a++ 此时a=4而b依然=3

（基本数据类型是储存在栈内存中的）

引用数据类型 obj.name=”孙悟空” var obj2=obj;

如果将obj.name=’’猪八戒’’此时obj2.name也会=“猪八戒”；会跟着变

（引用数据类型是储存在堆内存中，把地址给了栈内存中的变量）



基本数据类型比较是比较值，而引用数据类型比较的是地址！

对象字面量

创建对象的快捷方法

Var obj={name:“孙悟空”，

Age:18，

Gender:“男”}（注意赋值是“ : ”，不是“ = ”）

这样一可以创建对象二还可以同时创建属性及给属性赋值

函数

函数其实也可以看成是一个对象，不过创建用的是new Function，不是new Object

通常的创建方式是 function fun1（形参1，形参2）

{

}

和C语言一样

枚举对象里的属性

For..in语句

语法：for(var n in 对象名称){

Console.log(“n”);

}

这样就会打印出这个对象的所有属性

而

语法：for(var n in 对象名称){

Console.log(对象名称[n]);

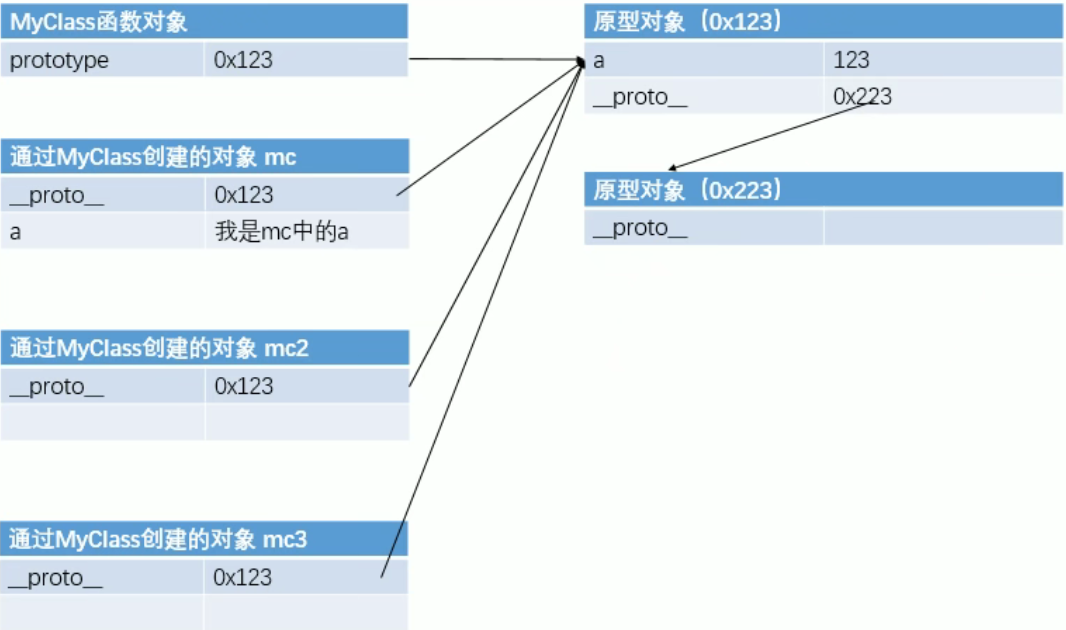
}就会打印出对象的所有属性值

函数

函数声明会在所有语句执行之前就被声明

原型对象

当构造函数被调用时，浏览器（解析器）会给构造函数加上一个prototype属性，指向这个类型的原型对象，且这个类的所有实例都有个\_\_porto\_\_属性（隐性）指向原型对象，当实例的属性或者方法找不到时，回去原型对象里去找  
同理，我们可以直接去原型对象里创建属性，该属性可以直接被该类下的所有实例访问到  
语法为：MyClass.portotype.a=123  
MyClass.portotype.SayName={....}  
Mc.a就等于123了



Tostring

Hasownproperty 方法可以判断对象是否自身有这个属性，区分出该属性是不是从原型对象来的

当打印一个一个对象时，事实上就是在调用tostring方法，一般返回【Object；Object】

如果我们复写该类的原型里的tostring方法，person.prototype.tostring= function{return”name=”+this.name+””+……}

这样console.log(person)就会出现 name=猪八戒，age=18 ..

而不是【Object；Object】，且直接打印该类的所有子对象也会这样了

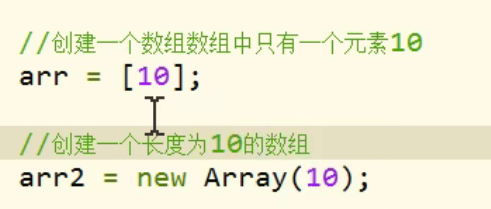
数组

数组也是对象，所以创建方式也可以是var arr =new Array();

字面量方法：var arr=[1,2,3,4,5]

Var new array=[];也可以

结果是一模一样的



数组的四个方法  
1.删除第一个数，返回被删的数，shift  
2.增加一个数在首，返回新长度，unshift  
3.删除最后一个数，返回被删的数，pop  
4.增加最后一个数，返回新长度，push

foreach数组遍历

arr.foreach(function(){ .. ..})

数组有几个元素，里面的回调函数就执行几次，元素会作为实参传入里面的函数（用不来）



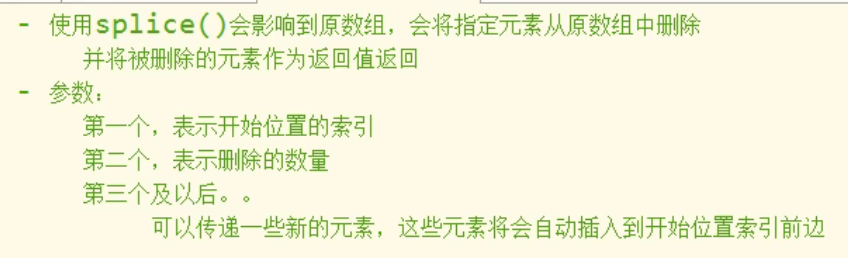
Slice(0,2)的参数有两个，第一个是开始索引(含)，第二个截止至(不含)

如0,2是第一个和第二个；1,2,就是第二个数；

只写一个就是从这个(含)开始，返回以后的所有元素

如果是负数，就是从后往前

Splice()也有两个参数例如0,1 从第一个开始，删除一个



Math  
Math是一个工具类，不用生成对象就可以直接拿来用  
math的方法有以下几种：

  
对象转成布尔值都是true

包装类

Js有三个包装类，分别是String，Number，Boolean

可以将三个基本数据类型转换成基本数据对象，因此，即使值相同，他们互相比较时，比的是地址，所以依然不同，所以基本不用

Var a = new Number（）；

A和普通的数字变量差不多，他的值还是3，但是它成为了一些对象（也就是说可以添加属性），还有一些方法

比如s=s.tostring 转换为字符串类型类型

当我们使用基本数据类型时，它依然有一些方法，如s=s.tostring ，这是浏览器自动转换为基本数据对象，再去调用方法，调用完之后又自动转换回来了

String类的方法

String类底层是字符数组的形式保存的(空格也算)，所以我们可以在变量的后面加上索引来查看第几个字符，比如str=“hello”

Str[0]=”h”;

**//length方法，求长度**

**//charat 方法，可以返回指定位置的字符**

Charat[1]=”e”

**//charcodeat方法，和charat类似，只不过返回的是code编码**

Charcodeat[1]=101;

**//concat() 方法用于连接两个或多个字符串。**

stringObject.concat(stringX,stringX,...,stringX)

使用 " + " 运算符来进行字符串的连接运算通常会更简便一些。

**//indexOf() 方法可返回某个指定的字符串值在字符串中首次出现的位置。**

**语法**

stringObject.indexOf(searchvalue,fromindex)

## 提示和注释

注释：indexOf() 方法对大小写敏感！

注释：如果要检索的字符串值没有出现，则该方法返回 -1。

**//lastIndexOf() 方法可返回一个指定的字符串值最后出现的位置，在一**个字符串中的指定位置从后向前搜索。

**语法**

stringObject.lastIndexOf(searchvalue,fromindex)

|  |  |
| --- | --- |
| **²ÎÊý** | **ÃèÊö** |
| *searchvalue* | ±ØÐè¡£¹æ¶¨Ðè¼ìË÷µÄ×Ö·û´®Öµ¡£ |
| *fromindex* | ¿ÉÑ¡µÄÕûÊý²ÎÊý¡£¹æ¶¨ÔÚ×Ö·û´®ÖÐ¿ªÊ¼¼ìË÷µÄÎ»ÖÃ¡£ËüµÄºÏ·¨È¡ÖµÊÇ 0 µ½ *stringObject*.length - 1¡£ÈçÊ¡ÂÔ¸Ã²ÎÊý£¬Ôò½«´Ó×Ö·û´®µÄ×îºóÒ»¸ö×Ö·û´¦¿ªÊ¼¼ìË÷¡£ |

从最后开始数，碰到第一个匹配的，从前往后数这个数是第几个

**//slice() 方法可提取字符串的某个部分，并以新的字符串返回被提取的部分。**

**语法**

stringObject.slice(start,end)

如果end是负数，例如(1,-1)是从第二个开始，直到倒数第一个（含）

**正则表达式**var reg = new RegExp();  
RegExp也是一个对象它的参数有  
RegExp(“正则表达式”，“匹配模式”)；  
i是忽略大小写，g则是全局匹配  
正则表达式有个方法叫test（）；  
例如  
“正则表达式”里，var reg = new RegExp(“a”);  
reg.test(“abc”)是true   reg.test(“dde”)则是false  
他用来判断里面有没有这个“正则表达式”的内容  
只要有一项没有，就是false  
还可以这么写  
var reg =/正则表达式/匹配模式  
如果想写“或”的话  比如字符串中含有a或b或c等等  |   就可以  
/a|b|c/  
或者  
[ab]  
都是a或b的意思  
[A-z]是含有任意字母的意思  
[0-9]是含有任意数字的意思  
[^ab]是不含有a或b的意思  
  
字符串有四个方法和正则表达式有关  
split   把字符串分割为字符数组返回(asd#asd#aSD#),以#分割  
search 检索与正则表达式相匹配的，返回第一个相匹配的索引值  
match 找到一个或多个与正则表达式相匹配的，封装到数组里返回  
replace替换与正则表达式相匹配的子串

innerHTML   可以通过这个方法找到元素内的HTML代码  
btn.innerhtml  
  
Getelementsbytagname  
返回的是一个类数组对象，用下标来查看  
或者用一个变量接收后，lis[i  
]innerhtml     查看内容  
  
查看元素的属性直接点  
Js.value   Js.id  
唯独class特殊，js.classname  
  
图片切换的练习  
  
**元素节点的子节点**  
getelementsbytagname  
var city= document.getelementsbyid("city");  
var lis =City.getelementsbytagname("li");  
在city下找的  
Children  
获取当前元素的所有子元素，不会包含空格，换行  
  
**元素节点的父节点**  
var pt=city.parentnode;  
Pt.innertext可以输出整个ul里的li内容，不包含标签，innerHTML包含，他们就这个区别  
节点包含空格，元素不包含  
Firstchild  
Lastchild

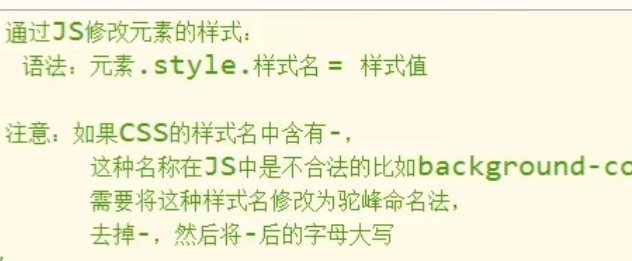
Getelementsbytagname主要原于获取表单  
items[i].checked=true  
可以设置复选框的选中状态

dom增删改查

  
这些大部分都需要使用父节点来操作  
比如insertbefore()，replace等等  
父节点.insertbefore(新节点，旧节点)  
  
为了方便，可以这样写  
子节点.parentnode.方法(需要的参数)  
修改代码  //向代码中添加广州  
City.innerhtml+="<li>广州<li>"  并不好，动静太大了，所以代码都会被重写一次，之前的事件可能会失效

简单的 HTML 表格由 table 元素以及一个或多个 tr、th 或 td 元素组成。

tr 元素定义表格行，th 元素定义表头，td 元素定义表格单元。

³¬Á´½Ó°ó¶¨µÄÏìÓ¦º¯ÊýÔÚ×îºóreturn false ¿ÉÒÔÈ¡ÏûÄ¬ÈÏÐÐÎª  
ÔÚ³¬Á´½ÓsrcÀïÐ´£¨Javascript£º£»£©Í¬ÑùÐ§¹û  
³¬Á´½ÓµÄÄ¬ÈÏÐÐÎª¾ÍÊÇÌø×ªÁ´½Ó  
ÕâÑù¾Í¿ÉÒÔµã³¬Á´½ÓÊµÏÖ¹¦ÄÜ¶ø²»Ìø×ªÒ³Ãæ  
  
confirm·½·¨Ìá¹©È·¶¨È¡Ïû°´Å¥£¬²¢´øÓÐÒ»¶ÎÌáÊ¾£¬·µ»ØÖµÊÇtrueºÍfalse  
ä¯ÀÀÆ÷Ä¬ÈÏ¸øtr±êÇ©¼ÓÒ»¸ötbody±êÇ©£¬td±êÇ©ÔÚÕâÀïÃæ  
  
domÄÚÁªÑùÊ½  
ÑùÊ½ÖµÐèÒªÊÇ“×Ö·û´®”  
¶ÁÈ¡Í¬Àí£¬  
ÔªËØ.style.ÑùÊ½Ãû  
²»¹ý¶Á³öÀ´µÄ´øpx£¬²»´æÔÚ300px+100ÕâÖÖÔËËã  
¶ÁÐ´µÄÄ¬ÈÏÊÇÔªËØµÄÄÚÁªÑùÊ½£¬ËùÒÔÔÚstyleÀïµÄÑùÊ½ÕâÑùÎÞ·¨¶ÁÈ¡µ½  
»ñÈ¡ÔªËØµ±Ç°ÏÔÊ¾µÄÑùÊ½  
  
Ö»ÓÐIEÖ§³Ö   ÔªËØ.current.style  
×¢ÒâÊÇµ±Ç°ÏÔÊ¾µÄÑùÊ½£¬²»ÊÇÄÚÁªÑùÊ½»òÕßstyleµÄ  
ÔªËØ.current.style.ÊôÐÔ  
µ±È«ÆÁÊ±£¬·µ»Ø¿í¶ÈÊ±auto  
  
ÆäËûä¯ÀÀÆ÷Ö§³ÖµÄµÄ£¬³ýÁËIE8¼°Ò»ÏÂµÄ°æ±¾  
ÐèÒªÁ½¸ö²ÎÊý£¬µÚÒ»¸öÊÇ¶ÔÏó£¬µÚ¶þ¸öÊÇÒ»¸öÎ±Àà£¬Ò»°ãÎªnull  
var obj = getcomputedstyle(box1,null)  
Õâ¸ö¶ÔÏóÀï·â×°ÁË¶ÔÏóµÄÊýÖµ£¬ÒªË­µãË­£¬Èç  
obj.width  
µ±¿í¶È100%Ê±£¬·µ»ØµÄ²»ÊÇauto  
¶øÊÇ¾ßÌåµÄÊýÖµ  
getcurrentstyleºÍgetcomputedstyle¶¼ÊÇÖ»¶Á²»¿ÉÐÞ¸ÄµÄ  
  
¼È¼æÈÝIE8¼°ÆäËûä¯ÀÀÆ÷µÄ

Window加载完毕之后，for循环会立即执行

而响应函数要在被点击之后才会执行

（响应函数：xxxxxx=function（）{ 函数体 }）

不注意这个可能会产生一些意想不到的错误



按理来说for执行完了i应该等于三，但allA[3]不存在啊，何来allA[3].onclick这个东西？

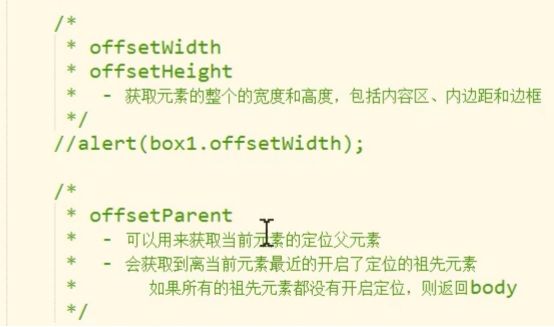
Clientheight

Clientwidth

这两个都可以获取元素的可见高度和宽度，但是不包括外边距，只包括内部的



这些都是只读，要改只能在style里改





判断滚动条有没有滚动到底

If(**scroll.heigth-scroll’top==client.height**)

鼠标移动事件

Window.onmousemove（event）

除火狐外，大部分浏览器都会传一个实参进参数列表

我们可以自己写一个

为了浏览器兼容问题，一般都会加一行

event=event||window.event

火狐不传参数，我们就自己建一个，把window.event传给他，有的话就是event=event

也无事发生

Event有很多属性，具体可查看文档，例如

Var y=event.ClientY 鼠标的Y坐标

Var x=event.ClientX 鼠标的X坐标

如何实时显示鼠标的坐标值？

1. innerhtml=”x=”+x+”y=”+y;

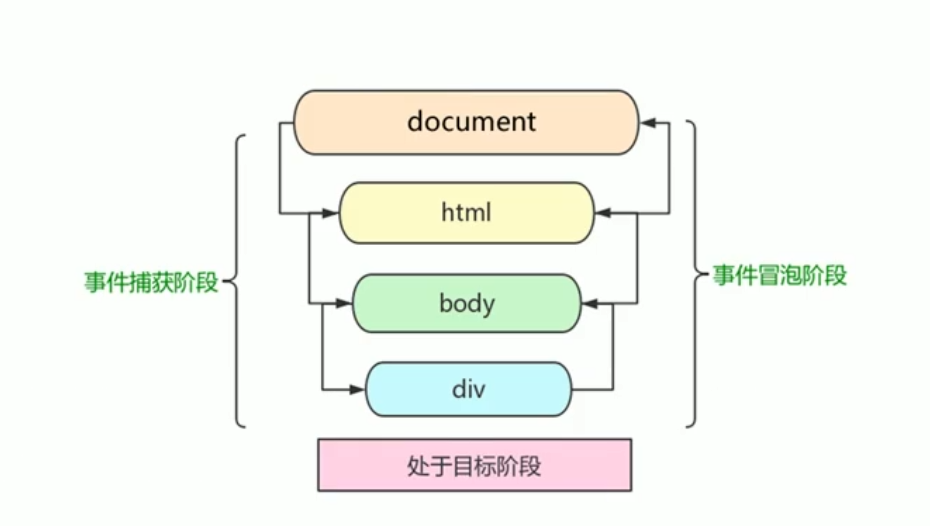
如何实时显示鼠标的坐标值？

1. innerhtml=”x=”+x+”y=”+y;

让一个div跟随鼠标移动

1. 创建这个div，设置颜色，宽高，并且，打开绝对定位，这样才可以移动
2. 在script标签里面定义一个元素，var box=document.getelementbyid(“div的id”)
3. Box.style.left=x+“px”(设置div距离左边的偏移量，这个+“px”很重要)
4. Box.style.top=y+“px”(设置div距离上边的偏移量，这个+“px”很重要)
5. 如果页面有纵横滚动条时，要加俩行
6. var sl =document.scrollLeft||document.documentElement.scrollLeft;
7. var st =document.scrollLeft||document.documentElement.scrollTop;

事件流



所有事件执行都是从捕获—目标阶段—冒泡阶段这样的

捕获阶段从父元素往下执行，如果有父元素有那种事件，则先执行父元素的事件，冒泡阶段则是从子元素开始执行，一路往上，有事件就执行，addEventListener(‘click’,function(){},这里ture就是从捕获阶段开始可以被执行到，如果是false，则是冒泡阶段被执行，捕获阶段不会被执行)

事件的冒泡

举个例子

点击div，不仅div绑定的响应函数会触发，而且body的响应函数也会触发，这种现象称之为冒泡

冒泡这个东西大多数时候都是有益的，比如div跟随鼠标，当鼠标移入另一个div2时，如果没有冒泡，这个div就不会跟随鼠标移入另一个div2，因为你触发的div2，没有触发div的document，但是有了冒泡，传导到document里，这样div也会跟随鼠标移入div2里了

如果不想要冒泡怎么办？？

S1.cancelbobble=true

这样这个对象的冒泡就被取消掉了，谁要取消就给谁写

事件的委派

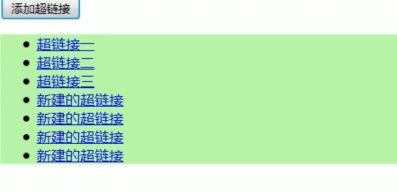
事件的委派其实就是冒泡的一个应用

要求为，只绑定一次响应函数，就可以实现所有对象都有这个效果，如论是后来的还是一开始就有的

这里以ul举例

要所有的li点击后都有效果，我们直接给ul（祖先元素）绑定效果

但是这会导致所有ul区域点击都有效果，比如这些绿色区域都会有效果



如何解决呢？我们需要知道用户点击的是哪个元素，dom事件里，target可以帮助我们

我们先给所有的li取同样的一个classname

If(event.target.classname==”link”){

Alert(“我是ul的响应函数”);

}

事件的委派

如何给一个元素绑定多个响应函数？

Addeventlistener

这个也可以给给元素绑定事件（函数）



但是IE8不支持，IE8支持的是

Attachevent



不过执行的顺序是反着的

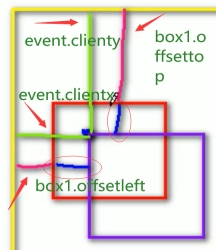
为了兼容所有的浏览器，我们自己写一个bind函数



拖拽

思路

1. 鼠标在区域内按下，获取到鼠标的坐标
2. 区域的坐标跟随鼠标移动（为了不让div产生偏移，需要减去差值）
3. 鼠标松开，取消跟随



某些情况下（ctrl+A）全选也会选中这些区域，如果区域被拖动了文字也会随着拖动，除IE浏览器外，在document..onmousedown函数里，最后一段加上 return false（解决默认行为）就可以解决

IE浏览器这样做

在down函数里面写下这一行

在uo函数里面写下这一行

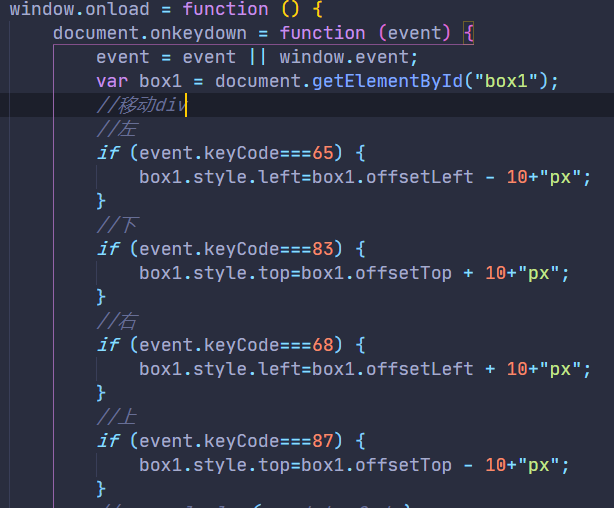
将所有的鼠标点击事件捕获到自己身上，这样就不会联系到其他元素，同时capture也就点一次执行一次，不会一直被绑定

键盘键入

Onkeydown///onkeyup

Event.onkeydown，用来检测键盘是否按下或者松开

Document.onkeydown=function（）{



}

BOM

Browser object model

BOM提供了一些对象来对浏览器来进行操作

BOM对象

Window

代表整个浏览器窗口，同时window也是网页中的全局对象

Navigato

代表当前浏览器的信息，可以通过该对象来识别不同的浏览器

Location

代表当前浏览器的地址栏信息，通过这个对象可以获取地址栏信息，控制页面跳转

History

代表当前浏览器的浏览历史记录，由于隐私原因，该对象不能获取到具体的历史记录，只能向前或者向后跳转，而且只有当次有效

Screen

代表用户屏幕的信息，可以获 取到用户的显示器屏幕信息

这些对象都是作为window对象的属性保存的，可以通过window对象来使用，也可以直接使用

Console.log（window.location）或者Console.log（location）

Navigato

用来区分用户使用的浏览器，但是最新的ie11（edge）使用了各种手段来隐藏自己是ie的事实，如果直接

Console.log（navigato.name）显示的都是（netscape）网景公司，但是navigato.userAgent显示的信息还是能区分大部分IE（8-10）和其他浏览器的

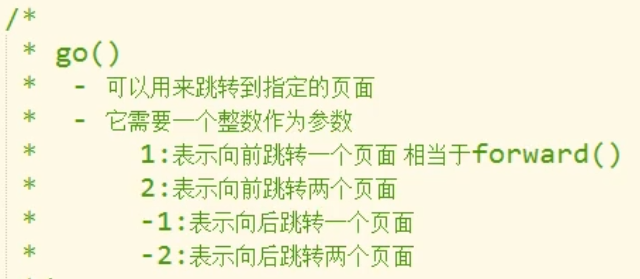
但是IE11为了隐藏，直接没有MSIE，所以我们用一些IE浏览器才有的对象来判断这个浏览器是否是IE浏览器

比如说if(ActivateXobject)，但是edge的结果居然是ture

但是if(ActivateXobject in window)这个edge是无论如何都藏不了的

History





Location

属性暂时用不到，这里讲三个方法

Assign（）；跳转至其他的页面和直接修改一个意思

reload（）；刷新页面,但是保留缓存,reload(true)才是强制刷新清除缓存

replace();替换页面,和assign的区别就是不会保存历史记录，无法返回

定时器

Interval

有setinterval和clearinterval

var timer = setInterval(function () {

            cout.innerHTML = a;

            a++;

*if* (a == 6) {

                clearInterval(timer);

            }

        }, 1000);

Setinterval(这里面写出要循环的函数体，第二个参数写间隔的时间单位为毫秒)

Var timer=setinterval

Setinterval有一个Number类型返回值，作为定时器的唯一标识

定时器切换图片

先找到几张图片，然后可以将图片路径保存近一个数组里

var indexarr=["../pic/20200628122924.png","../pic/89044244\_p0.jpg","../pic/QQ图片20210306112326.jpg"];

然后indexarr[0]就是第一张图片[1]就是第二张

改变图片的方法

Img.src=indexarr[index]；



延时调用

和定时调用很相似

Var timeout =settimeout(function(){},3000)

三秒后执行一次这个函数，只执行一次

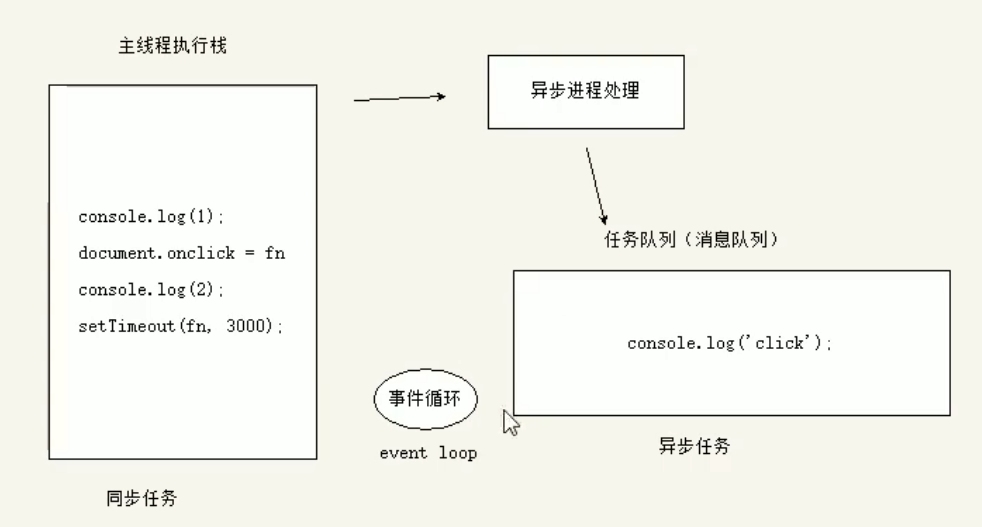
一样的还有cleartimeout（timeout）

清除延时调用

同步与异步

先执行同步队列里的任务，再执行异步任务队列，“异步进程处理”

这个模块会决定哪些异步任务进异步任务队列，比如说等待用户点击，这个的回调函数用户不点击是不会进入异步任务队列的，一直在异步进程处理那里，而倒计时三秒那个，三秒过后才会进入消息队列，主线程执行完了会不断地去异步队列里去看有没有新的任务，有则送入主线程执行



childNodes属性

在ACFUN两级导航栏中，div中有13个ul，每个ul又有不等的li，要获取并控制这些li可以使用childNods属性，首先先获取 不对不对，这些获取出来的是NodeList元素，不能用style属性

用这个，如果一个元素有很多子元素，还是上面那个例子，比如说我获取到了第二级菜单的div

这个div（class=lv2guide）中有13个ul，每个ul又有不等的li，在获取那13个ul的时候，不需要给每个ul都去加上class然后再去获取，直接var lv2guideul = lv2guide.getelementsbyclassname(‘ul’)，对不需要思维定势的去用document去.点了，要那个元素里面的东西全部同理，用那个元素去点

但是再获取ul中的li时，以上方法就不太试用了。很显然需要一个二维数组，如下图所示即可

var lv2 = new Array(); //声明一个长度为13的一维数组

*for* (let j = 0; j < 13; j++) {

        lv2[j] = new Array(); //每行有11列

        lv2[j] = lv2guideul[j].getElementsByTagName('li');//赋值

    }

lv2[1][1].style.backgroundColor = 'red'



但是数组一旦越界，后面的js代码就不会运行了，所以千万要注意不能越界

# 不知道的函数

for in函数

for (var x in obj){

用来遍历对象，x为对象的属性名，obj[x]就是那个值咯

}