Document Object Model

DOM定义了表示和修改文档所需的方法。DOM对象即为宿主对象，由厂商定义，用来操作html和css功能的一类对象集合。也有人称DOM是对HTML和XML的标准编程接口

DOM的基本操作

1. 对节点的增删改查

·查

查看元素节点

document代表整个文档

document.getElementById(‘’)//元素id在ie8下不区分大小写，而且也返回匹配name属性的元素

.getElementsByTagName(‘’) //标签名，一或多个标签名相同的以类数组方式展现，标签名写’\*’时，代表选择所有标签，是实时

.getElementByName(‘’) //注意，只有部分标签name可生效(表单，表单元素，img，iframe)

.getElementsByClassName(‘’) //类名，ie8和ie8以下版本没有，可以多个class一起，返回的也是类数组

.querySelector() //css选择器，在ie7和ie 7以下没有，例如.querySelector(‘#demo>.a>.b>span’)，它只会返回第一个有这个css的标签，不是实时的

.querySelectorAll() //css选择器，在ie7和ie 7以下没有，它会返回类数组，包含所有全文中有的被选择的标签

2.节点的类型

元素节点 —— 1

属性节点 —— 2

文本节点 —— 3 空格和回车都算文9本节点

注释节点 —— 8

Document —— 9

DocumentFragment —— 11 作用是一种隐形的筐，例如在ul中插入一万个li，如果一个一个插效率太低，再加个ul会影响结构，这就需要把这些li插入到隐藏的框中再一起插入，纯粹为了优化 var Frag = document.createDocumentFragment();

获取节点类型：nodeType

1. DOM基本操作

遍历节点树

Xx.parentNode：父节点(最顶端的父节点为parentNode)

xx.childNodes：子节点们

xx.firstChild：第一个子节点

xx.lastChild：最后一个子节点

xx.nextSibling：后一个兄弟节点

xx.previousSibling：前一个兄弟结点

基于元素节点树的遍历

parentElement：返回当前元素的父元素节点(IE不兼容)

children：返回当前元素的元素子节点们

node.childElementCount === node.children.length当前元素节点的子元素节点个数

firstElementChild-->返回的是第一个元素节点(IE不兼容)

lastElementChild-->返回的是最后一个元素节点(IE不兼容)

nextElementSibling/previousElementSibling-->返回前一个/后一个兄弟元素节点(IE不兼容)

节点的四个属性

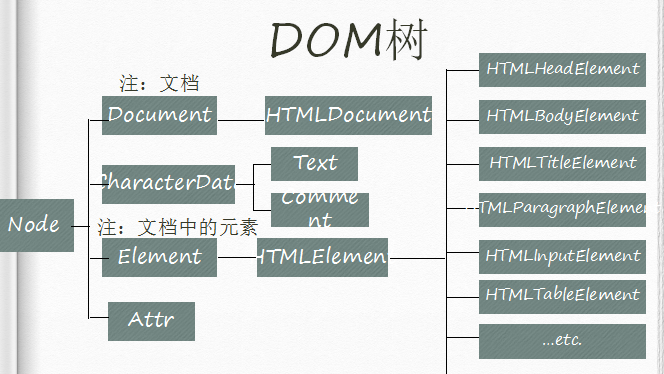
nodeName：元素的标签名，以大写形式表示，只读

nodeValue：text节点或comment节点的文本内容，可读写

nodeType：该节点类型，只读

attributes：Element节点的属性(节点)集合，每一位代表一个属性节点

节点的一个方法：Node.hasChlidNodes();判断调用的对象里有没有子节点，有返回ture，没有返回false



DOM元素的一个继承关系，node所有节点类型，Document(文档)构造函数，Element(文档中中的元素)构造函数

Document-->HTML/XMLdocument-->document(看保存的文件后缀是.html还是.xml来决定继承自哪一支)

·getElementById方法定义在Document.prototype上，即Element节点上无法使用，如div.getElementById

·getElementsByTagName方法定义在Document.prototype和Element.prototype上

·HTMLDocument.prototype上定义了一些常用属性，body.head属性代表body.head标签，如document.head==><head></head>

·document.documentElement指代的是html标签

·getElementsByClassName,querySelectorAll,querySelector在document和element上均有定义

练习题：1.遍历元素节点树，要求不能用children属性

2.封装函数，返回元素e的第n层祖先元素

3.封装函数，返回元素e的第n个兄弟节点，n为正，返回后面的兄弟节点，n为负，返回前面的，n为0，返回自己。

4.编辑函数，封装children功能，解决以前部分浏览器的兼容性问题

5.自己封装hasChildren()方法，不可用children属性

在js001文件里

DOM基本操作

增：document.createElement(‘’);在js里加不在html里加

document.createTextNode(‘’);

document.createComment(‘’);创建注释节点

document.createDcumentFragment(); DocumentFragment 节点

插：PARENTNODE.appendChild();

例：div已经creat，document.body.appendChild(div)，这时

在html中加入了div标签，直接插html中原有的标签也行，div.appendChild(p)，把p标签插入div标签的尾部，相当于剪切

PARENTNODE.insertBefore(a,b);insert a Before b

删：parent.removeChild();删除相当于剪切操作

改：parent.replaceChild(new,origin);

Element节点的一些属性

xx.innerHTML = “”:返回html中的代码，可读写，会覆盖，要写字符串形式的

xx.innerText(火狐不兼容)/textContent(老版本IE不好使)：返回html中的文本，写的话会覆盖

Element节点的一些方法

ele.setAttribute(‘’)设置标签里的属性div.setAttribute(‘class’,’demo1’)，可以设置新属性(可以是系统的也可以是自定义的(自定义的属性也可取))

ele.getAttribute(‘’)取得标签里的属性 div.getAttribute(‘class’)

dom.className = “”也可以改class

练习：请编写一段JavaScript脚本生成下面这段DOM结构。要求：使用标准的DOM方法或属性。

<div class="example">

<p class="slogan">姬成，你最帅!</p>

</div>

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

p = document.createElement('p');

text = document.createTextNode('我最帅');

document.body.appendChild(div);

div.appendChild(p);

p.appendChild(text);

div.setAttribute('class','example');

p.setAttribute('class','slogan');

日期对象date

Date()系统内部构造函数

var date = new Date(可以写日期2016,4,5,16,20);

date.toString()把毫秒数显示成当前日期，返回：Sun Apr 17 2016 14:46:38 GMT+0800 (中国标准时间)

getDate()从Date对象中返回一月的某一天(今天)，例4月17日则返回17

getDay()返回星期几周天为开始为0，例如是周天则返回0

getMonth()返回月份从0开始，例如是4月则返回3

getFullYear()今年年份，例如是2016年则返回2016

getHours/Minutes/Seconds/Milliseconds()小时，分，秒，毫秒，从0开始

getTime()返回距1970年1月1日到现在的毫秒数，记录当前你创建Date对象的时刻，计算机的时间都是用毫秒数计算的

set...设置

setInterval(function(){},隔多长时间运行一次);

setTimeout(function(){},隔多长时间开始运行);

clearInterval();/clearTimeout清除，需要传一个参数

全局对象window上的方法，内部函数this指向window

注意 ：setInterval(“func()”,1000);

查看滚动条的滚动距离，距离浏览器左，上边框的距离

window.pageXOffset/YOffset IE8及以下不兼容

document.body/documentElement.scrollLeft/scrollTop每个浏览器这两种方法只有一个有值，另一个肯定是零，解决兼容性问题把这两个属性加一下

查看视口尺寸

window/ele.innerHeight/Width(IE8及以下不兼容)不包括border

document.documentElement.clientHeight/Width(标准模式下任何浏览器都兼容)

document.body.clientWidth/Height(适用于怪异模式下的浏览器：兼容以前版本浏览器的语法，把<!DOCTYPE html>删了就是怪异模式下的了)

document.compatMode返回BackCompat是混杂模式，返回CSS1Compat不是混杂模式

查看元素的几何尺寸

domele.getBoundingClientRect();

兼容性很好，该方法返回一个对象，对象中包括left,top,right,bottom等属性，left和top代表该元素左上角的X和Y坐标，right和bottom代表该元素右下角的X和Y坐标，height和width属性在老版本IE上并未实现，返回的结果不是实时的

属性值是每一个元素顶点到浏览器边框的值

让滚动条滚动

Window上有三个方法scroll(),scrollTo(),scrollBy();三个方法都是输入X，Y坐标，使滚动条滚动到当前位置，div1.scroll(100,100)

区别：scrollBy()会在之前的数据基础上做累加,可以用scrollBy()实现快速阅读功能

查看元素尺寸

dom.offsetHeight/Width包括border，padding不包括margin

查看元素位置

dom.offsetLeft/Top对于无定位父级的元素，返回离浏览器边框的距离，有定位父级的元素则返回距离父级元素的距离

dom.offsetParent返回自己有定位的父级，如没有，返回最外侧标签body标签

脚本化css

dom.style.prop可读写行间样式，没有兼容性问题，遇到float这样的关键词前面要加css，float-->cssFloat，复合单词的属性必须拆解组合单词变成小驼峰式写法(例如background-color变成backgroundColor)，写入的值必须是字符串形式，样式必须一个一个改变(像margin和border等)，可写的方式只有style

left和right放在一起设置会出错

查询计算样式

window.getComputedStyle(ele,false);后面的可以写’before’或’after’访问伪元素，计算样式只读，返回的计算样式都是绝对值，没有相对单位，IE8及以下不兼容，返回的是对象

ele.currentStyle样式只读，返回的不是经过转换的绝对值，IE独有，返回的也是对象

这两种方法复合单词的属性必须拆解组合单词变成小驼峰式写法(例如background-color变成backgroundColor)，遇到float这样的关键词前面要加css，float-->cssFloat

封装getStyle

事件：交互体验的核心功能

事件浏览器原本就有的，就是在事件触发时添加一个回调函数(给动作加了一个响应状态)

回调函数：当满足一定状态时才执行的函数叫回调函数

事件监听有浏览器其他部分处理，执行时才交给js引擎

如何绑定事件

ele.onxxx = function (event) {} 句柄

兼容性很好但一个元素只能绑定一个处理程序，基本等同于写在HTML行间上，this指向dom元素本身

obj.addEventListener(type(不加on,’click’),fn,false);

IE9及以下不兼容，可以为一个事件绑定多个处理程序，同一个处理函数多次绑定事件是不允许的，this指向dom元素本身

obj.attachEvent(‘on’ + type,fn);

IE独有，一个事件绑定多个处理程序，同一个处理函数多次绑定事件是允许的，this指向window

解除事件处理程序

Ele.onclick = false/’’/null

ele.removeEventListener(type, fn, false)

ele.detachEvent(on + ‘type’, fn)(IE)

注：匿名函数用无法解除

封装解除绑定事件函数

事件处理模型---------事件冒泡，事件捕获

事件冒泡：结构(非视觉上)上嵌套关系的元素，会存在事件冒泡的功能，当子元素触发一个事件，这个事件会冒泡到父元素上，即同一事件，自子元素冒泡向父元素。

不管绑没绑定都会传递因为事件是自带的，与处理函数无关，最底的子元素冒泡向父元素不管中间元素是否绑定了处理函数都会冒泡向父元素，在它自身触发也会冒泡向父元素

事件捕获：结构(非视觉上)上嵌套关系的元素，会存在事件捕获的功能，即同一事件，自父元素捕获至子元素。

IE没有捕获事件，只有dom.addEventListener(type,fn,ture)有捕获，当false改为ture时不再有事件冒泡，而是事件捕获，只有chrome浏览器(用webkit内核的)提供了事件捕获模型

触发顺序：先捕获后冒泡

focus,blur,change,submit,reset,select等事件不冒泡

取消冒泡和阻止默认事件

W3C标准 event(事件对象，函数里传的参数).stopPropagation();但不支持IE9以下版本

系统在调用事件处理函数时会传入一个事件对象，可以传参代表，取消这个事件的冒泡，使他触发处理函数时只在自己身上显示，他的子级可以冒泡到他身上，但不能再冒泡向他的父级，IE独有 event.cancelBubble = ture;(新版本webkit内核的浏览器也能用)

封装取消冒泡

阻止默认事件

默认事件----表单提交，右键菜单，a便签跳转

oncontextmenu右键打开菜单事件

1. return false以对象属性的方法(style)注册的事件才生效
2. event.preventDefault();W3C标注，IE9以下不兼容
3. event.returnValue = false;兼容IE
4. 封装阻止默认事件的函数

事件对象

event||window.event(IE) IE的event不会传到处理函数的，可以用一个变量接收event用以兼容，var event = e||window.event

事件源对象(触发这个事件的源对象)：event.target火狐独有，event.srcElement IE独有

这两chrome都有，兼容性写法var target = e.target || e.srcElement

event里有clientX,Y属性鼠标当前位置离浏览器边框的距离

事件委托

利用事件冒泡和事件源对象处理

优点：1.性能 不需要循环一个一个绑定

2.灵活，有新的子元素不必重新绑定

事件分类

鼠标事件：click,mousedown,mouseup,mousemove,contextmenu,

mouseover(在上悬停),mouseout(逻辑上外面的)

click=mousedown+mouseup

mouseenter只有鼠标穿过被选元素时才触发事件，对应mouseleave

mouseover不论鼠标穿过被选元素或其子元素都会触发事件

mousedown左键右键都能触发，在event对象中有button属性，

用button来区分鼠标的按键，0/1/2，左键为0右键为2

DOM3标准规定：cilck只能监听左键，只能通过mousedown，mouseup来判断鼠标键

如何解决mousedown和click的冲突

封装拖拽，火狐稍低版本中只要拖拽的dom里没有文字就无法拖拽，dom.setCapture(),dom.releaseCapture()，调用这个方法后，屏幕上发生什么事都会捕获到他身上去(IE独有，平时用不到，知道就行)

键盘事件

Keydown，keyup，keypress按下键盘Keydown，keypress连续触发，抬起keyup才触发，keypress只针对字符键(ASC||有的)，而keydown是所有，键盘事件的event里有个which属性，里面的值是这个按键的唯一标识，且keydown和keypress的值不一样，keypress的event里的charCode属性，charCode是ASC码值，

keydown的charCode值为0

String.fromCharCode()可接收一个指定的Unicode值，返回成一个字符串

Function Person() {} var obj = new Person() obj.say()算是动态

Function Person(} {} Person.say = function () {} Person.say();算是静态

文本操作事件

Input，focus，blur，change

<input type=”text/radio(单选框)” oninput=””>输入一次就触发一次事件，focus当鼠标聚焦时，blur当失去焦点时触发事件，change：input中文本有没有变化，如果有则触发事件

HTML标签<select></select>下拉菜单，要取消它的样式-webkit-appearance:none

事件分类

窗体操作类(window上的事件)

scroll load

Window.load = function(){}等整个文档都加载并执行完毕才会执行load事件中的代码，最好不用，会严重降低效率，在load里写document.write会清除前面全部的文档流

scroll当滚动条滚动触发事件

Confirm()方法用于显示一个带有指定消息和ok及取消按钮的对话框，点击确定返回ture，点击取消返回false

正则表达式：匹配字符串的

补充：转义字符”\” \n换行 想定义多行字符串，在字符串最后加一个\

作用：匹配特殊字符，或匹配有特殊搭配原则的字符的最佳选择

两种创建方法：直接量 new RegExp()推荐使用直接量创建

直接量：var regExp = /abc/gim; gim表示三种修饰功能

var regExp = new RegExp(“abc”,”g”)或

var regExp = new RegExp(“regExp”)不填第二个参数

var regExp = RegExp(regExp);也可以创建，相当于赋值，完全一样

在字面量中，\w,,\s,\d有特殊的作用，而在内置构造函数创建中会把\w理解为转义字符再加w，得写成\\w才行

g:global全局查找 i:ignore执行对大小写不敏感 m:执行多行匹配

，并且以每一行为一个单独的字符串去匹配

var reg = /abc/; var str = “abcdab”

str.match(reg)返回匹配的字符串，以数组的方式返回

表达式：

[a-zA-ZA-z0-9]类似于一个区间，查找方括号之间的字符，每次只匹配一位字符

[^]这个^是除了的意思，[^a]就是除了a都符合要求

()内优先级提高，单写没区别，有子表达式的意思，会再匹配一遍子表达式中的信息 |正则中的或

元素符

. 查找单个字符，除了换行和行结束符\r \n

\w查找单词字符：范围：[A-z0-9\_] \W查找非单词字符

\d查找数字字符 \D查找非数字字符

\s查找空白字符：空格符，制表符\t，回车符\r，换行符\n，垂直制表符\v，换页符\f，只要是空白的就是空白字符

\S查找非空白字符 var reg = /\s\S/可匹配所有字符

\b单词边界 var reg = /\bacc/g; var str = “”accb bdaccd efg”，匹配出的是第一个acc，分左右，/\ba/和/a\b/是不一样的

\B

量词

n+ 匹配任何连续包含至少一个n的字符串，至少有一个至多匹配 1 ~ N个 等同于n{1, }

n\* 匹配任何连续包含零个到多个n的字符串 \*可以匹配出空，

0 ~ N个 等同于n{0, }

n？ 匹配任何连续包含零个或一个n的字符串 0 ~ 1 等同于n{0,1}

N{2}匹配任何包含两个n的字符串，n{2，4} 匹配任何包含两个到四个n的字符串，符合贪婪匹配原则，有多则匹配多，n{1，}至少一个到正无穷个n的字符串

后面加问号符合最小匹配，不再符合贪婪匹配

^abc是以开头的a为开头的字符串 $bca是以结尾的a为结尾的字符串，^abc$可以固定字符串，必须得是abc

例题：写一个正则表达式，检验字符串首尾是否含有数字？

var reg = /^\d | \d$/ig

正向预查，正向断言

var reg = /a(? = abc)/g; 匹配后面是abc的那个a

var str = “aabc”; 匹配的就是a

非正向预查

var reg = /a(?! abc)/g; 匹配后面不是abc的那个a

RegExp对象的方法

test:检索字符串中指定的值，返回true或false，reg.test(str); 如果有返回true否则返回false

exec:检索字符串中指定的值，返回找到的值，并确定其位置，会返回index值，为匹配出的字符串开头位置，可多次匹配

lastindex在全局匹配的情况下从哪块开始，可读写，游标的作用，

从零开始，匹配的第二次从第一次匹配结束的位置开始匹配

支持正则表达式的string对象上的方法

search 检索与正则表达式相匹配的值的打头的位置，就找一次，即使全局查找也只返回一个位置，找不到返回-1

match 与exec方法类似，但没有有游标值，加上全局匹配就只显示匹配出的字符了，是个数组

split把字符串分割为字符串数组

replace特别灵活 字符串改成字符串只会改第一位，想用其他方式改可以写正则表达式，颠倒ab，var reg = /(\w)(\w)/g; str = “ab”; str.repalce(reg,”$2$1”) $1代表第一个子表达式，第二个参数还可以写函数，关注函数的返回，如果想用$这个字符就写$$

反向引用 var reg = /(\w)(\w)\2\1/可匹配出”abba””zxxz”等字符串，\1\2复制了两个子表达式匹配出的值，不是复制子表达式，

var reg = /(?:)/ 这么写括号就不再有子表达式的意思，仅仅是括号内优先级提升的作用

toUpperCase()将当前字符串转化为大写的

例题：请将如下变量名转换为小驼峰式

my-first-name ----> myFirstName

解： var str = my-first-name; var reg = /-(\w)/

str.replace(reg, function ($,$1) {

return $1.toUpperCase();

})

将10000000改为10.000.000的形式

解： var reg = /(?=\B((\d){3})+$))/g,

str='10000000000';

var str1 = str.replace(reg,".");

console.log(str1);

JSON是一种传输数据的格式（以对象为样板，本质上就是对象，但用途有区别，对象就是本地用的，json是用来传输的）

对象的属性名上加””来区分本地对象和json “name”:”hgf”;

JSON.parse(); string — > json

JSON.stringify(); json — > string

传递数据时需要数据的转换

异步加载JS的需求

js加载的缺点：同步加载，加载工具方法没必要阻塞文档，js加载会影响页面效率，一旦网速不好，那么整个网站将等待js加载而不进行后续渲染等工作

放到body里所有标签的后面写是因为dom文档全部解析完后就可以下载并执行js了

有些工具方法需要按需加载，用到再加载，不用不加载

三大树

DOMTree 遍历符合深度优先 一个一个标签读，但并不加载标签内的资源，能认出这个标签就解析完毕了

CSSTree script会阻塞DOM和css成树

RenderTree 把HTML和css相结合的树 reflow重构

javascript 异步加载 的 三种方案

1.defer 异步加载，但要等到dom文档全部解析完才会被执行。只有IE能用。<script defer=”defer”></script>，能把js写在script 标签里

2.async 异步加载，加载完就执行，不用等dom解析完毕，async只能加载外部脚本，不能把js写在script 标签里。IE9以下不能·用，移动端一般用async，<script async=“async”><script>

1.2 执行时也不阻塞页面，异步加载不能写document.write()

1. 创建script，插入到DOM中，加载完毕后callBack，一种兼容方式

var script = document.createElement(‘script’);

script.type = “text/javascript”;

//什么时候下载完这个库

script.onload = function () {} //IE没有

(script.readyState = “loading” //没加载完 = “complete” || “loaded”//加载完毕 IE独有的属性)

Script.onreadystatechange = function () {

If(script.readyState == “complete” || script.readyState == “loaded”){ //..... }

} //IE独有的事件，一但readyState的状态改变就会触发事件处理函数

必须写这两个事件处理函数等js下载完再执行你需要的函数，假如js文件特别大，不等js文件下载完就调用文件里的函数会报错，这两个事件处理函数要放在下载js前，如果网速很快在给事件绑定函数前js就下载完了，状态已经变为loaded，事件绑定函数什么用也没有

script.src = “js001.js”;

这么创建script，一但执行到这就开始下载，下载是异步的

Document.body.appendchild(script);

调用这个封装函数时第二个参数要借助函数引用，直接传库里的函数会报错，因为未定义

JS加载时间线 整个程序指向的顺序 优化提高效率，很重要

1、创建Document对象，开始解析web页面。解析HTML元素和他们的文本内容后添加Element对象和Text节点到文档中。这个阶段document.readyState = 'loading'。

2、遇到link外部css，创建线程加载，并继续解析文档。异步加载css文件

3、遇到script外部js，并且没有设置async、defer，浏览器加载，并阻塞，等待js加载完成并执行该脚本，然后继续解析文档。

4、遇到script外部js，并且设置有async、defer，浏览器创建线程加载，并继续解析文档。

对于async属性的脚本，脚本加载完成后立即执行。（异步禁止使用document.write()）

5、遇到img/iframe等，先正常解析dom结构，然后浏览器异步加载src，并继续解析文档。

6、当文档解析完成，document.readyState = 'interactive'。

7、文档解析完成后，所有设置有defer的脚本会按照顺序执行。（注意与async的不同,但同样禁止使用document.write();

8、document对象触发DOMContentLoaded事件，相当于dom解析完毕，但确实在解析后一步，解析完成后同步脚本也异步执行，也就是第七步不会阻塞第八步，这也标志着程序执行从同步脚本执行阶段，转化为事件驱动阶段。

9、当所有async的脚本加载完成并执行后、img等加载完成后，document.readyState = 'complete'，window对象触发load事件。

10、从此，以异步响应方式处理用户输入、网络事件等。

document.on(readystatechange)事件，能监听document.readystate属性值的变化

DOMContentLoaded时间不能用document.on+’xxx’来绑定

补充

三种标准模式的写法

标准模式：不能兼容以前浏览器版本

1.<!DOCTYPE html> html5的写法

2.<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"> 四

3.<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> 四到五之间的html

怪异，混杂模式：能兼容以前版本浏览器，用以前的语法，不是两种语法都能用

只能同时存在一种模式

<label> for 属性 — > js中表示htmlFor

<label for=”only”>username</lable>

<input id=”only” type=”text” style=”border:1px solid black;”>用label标签把username和input标签绑定在一起

属性映射 HTML属性 映射到Element属性

系统自定义的特性(行间上的叫特性)和自带的有特殊定义的属性是有映射关系的，我们自定义的属性则没有，div.setAttribute

(‘abc’,’123’)div标签上会有体现，但div.abc是undefined，用getAttribute()得到abc的值，且div.abc不能写，没有映射关系，div1.className=’demo’可以读写在标签上，div.setAttribute(‘class’,’demo’)也可以，特定的class有映射关系

img图片预加载：当图片太大，把图片写在定义变量中，先下载，当js执行到那需要图片直接就可以用

byClassName封装函数，因为IE8及以下不兼容

Math.random() 和彩票程序 0-36的随机数

页面呈现流程

1. 浏览器把获取到的htnl代码解析成一个DOM Tree，html的每个标签都是树的一个结点，根节点就是document对象<html/>，
2. 浏览器把所有样式(css和浏览器样式设置)解析成样式结构体，在解析过程中去掉浏览器不能识别的样式
3. DOM Tree和样式结构体结合后构建呈现render Tree
4. Render Tree创建成功后，浏览器就可以根据rende Tree绘制了

创建渲染树需要以下步骤：

1. 从DOM Tree的根节点开始遍历所有可视结点
2. 不可见标签会被忽略(设置了display-none，脚本标签等)
3. 对于可视节点找到对应的css样式

回流

1. 计算在设备中精确的位置和尺寸，这正是布局阶段的工作，也成为回流
2. 从根节点开始遍历，计算出每个对象的几何样式

绘制

1. 可是节点的样式，几何外观，将这些信息渲染为屏幕上真是的像素点

重绘：render Tree的一些元素需要更新属性，而这些属性只影响外观，风格，而不会影响布局，

回流：当render Tree中的一部分因为元素的尺寸，布局改变而需要重新构建，这就是回流