## Window

它是一个全局对象（CEO、大boss）

特点：全局变量或全局函数都默认属于window对象 相当于是它的属性或方法

## 定时器7

### setTimeout定时器

setTimeout(callback,time)

过time毫秒后执行callback 只执行一次

Callback:函数 在这里是回调函数

Time:毫秒

返回值：数字 代表当前定时器的序号（1，2，3.....）

清除setTimeout所产生的定时器：

clearTimeout(定时器序号) ; //可以用来拆炸弹

### setInterval 定时器

setInterval (callback,time) 执行多次

清除定时器：

clearInterval(定时器序号) ;

## Location对象属性

用来操作地址栏

获取地址：

console.log(location. href );

//https://www.jd.com/index.html?userName=admin&pwd=123456#beijing

location.reload(); 刷新 (了解)

## 需要了解的对象

### History 了解

History.forword() 加载下一个历史记录

History.back() 加载上一个历史记

history.go(num); num=-1 上一个页面 num=1 下一个页面

### Navigator 了解

用来获取当前浏览器的信息（所在的系统、浏览器的版本）

navigator.userAgent 判断用户浏览器的类型

navigator.platform 判断浏览器所在的系统平台类型

## DOM

浏览器在加载页面时，把网页上标签、标签属性、文本都转换成对象，按照层次关系以树状的方式显示模型

## 获取节点对象

写在button后面

不用写在最后的方法：在script中写onload

document.getElementById('标签的id值');

根据标签的id值获取一个节点对象。 返回单个节点对象

document.getElementsByTagName('标签名');

根据标签名获取一组节点对象。返回一个伪数组，找不到返回空伪数组

document.querySelector('选择器'); // ul li .active #id

根据选择器获取单个节点对象。返回一个节点对象 找不到返回null

document.querySelectorAll('选择器'); // ul li .active

根据标签的值获取一组节点对象。返回一个伪数组 找不到返回空伪数组

想要取单个值时加下标var Node = document.querySelectorAll('.time')[0];

## 事件

### 事件三要素

事件目标（事件源）：触发事件的元素

事件类型：交互方式（鼠标点击、鼠标移入、鼠标离开等）

事件处理程序：事件触发后要执行的程序，用函数表示

### 绑定事件

事件目标 . 事件类型 = 事件处理程序

***divNode***.onmouseenter = **function**(){

***imgNode***.**src** = **'img/02.jpg'**;   
};

### 事件类型

① onclick

鼠标点击某个节点对象时

② onmouseenter

鼠标移入到某个节点对象时

③ onmouseleave

鼠标离开某个节点对象时

④ onmousemove

鼠标在某个节点对象上移动时

⑤ onchange

点击表单节点对象时

### 操作事件步骤：

1. 获取所要操作的和要改变的节点对象 div节点对象 图片节点对象

**var *divNode*** = ***document***.querySelector(**'div'**);

② 给要操作的对象绑定事件（鼠标进入、鼠标离开）

***divNode***.onmouseenter = **function**(){

③ 在鼠标进入事件中，改变图片的路径

④ 在鼠标离开事件中，改变图片的路径

## 节点对象的基本属性

### 非表单的基本属性

节点对象.id

节点对象.title

节点对象.className 类名

节点对象.href //针对a标签用的多

节点对象.src // 针对图片用的多

节点对象.innerHTML //获取节点对象中的内容（包含文本和被嵌套的标签）

节点对象.innerText

//获取节点对象中的内容（仅仅是文本内容） 设置节点对象中内容时，可添加标签和文本，但标签不会被解析,会当做普通的文本 注意 innerText 在火狐低版本有兼容性问题。火狐低版本中使用 textContent 代替

### 表单节点的基本属性

节点对象.value;【所有表单标签都可以用】获取表单里的内容

节点对象.checked;【针对多选框和单选框用的多】 true or false

节点对象.readOnly;【针对文本框或文本域】 true只读or false可读可写

节点对象.disabled【针对按钮用的多】 true禁用 or false不禁用

## 自定义属性

### 自定义标签属性

用户根据需求，自己给标签添加的自己定义的标签属性

如：

<img src='wc.jpg' bigImg='bigWc.jpg' />

bigImg='bigWc.jpg' 就是用户自定义的标签属性

### 操作方式

* 获取

节点对象.getAttribute('属性名');   //会返回标签的属性的值

* 设置

节点对象.setAttribute('属性名','值');   //会修改或添加标签属性

* 删除

节点对象.removeAttribute('属性名');   //会删除标签的属性

## 获取节点对象的style属性

节点对象.style.样式属性名; backgroundColor -去掉 后面的字母大写

仅仅只能够获取行内样式属性，内部和外联样式属性无法获取。

getComputedStyle(节点对象,null).样式属性名;

getComputedStyle(节点对象,null)['样式属性名'];

获取节点对象的style属性对象 在IE8及IE8以下无法使用getComputedStyle

节点对象.currentStyle["样式属性名"];

节点对象.currentStyle.样式属性名;

缺点：仅仅支持IE,其他浏览器不兼容

### 兼容处理

获取指定节点对象的指的样式的属性值：

参数：

nodeObj 节点对象

propertyName 样式属性名

返回值：样式属性对应的值

propertyName可以不用写，直接在调用函数的时候点出来、、

**function** *getStyleValue*(nodeObj,propertyName){  
 **if**(**window**.getComputedStyle){ *//如果支持谷歌 其他浏览器 返回true* **return** getComputedStyle(nodeObj,**null**)[propertyName];  
 }**else**{  
 **var** a=nodeObj.**currentStyle**[propertyName]*// IE浏览器* **return** a;  
 }  
}

或

**function** *getStyleValue*(node,attr){  
 **var** styleNode;  
 **if**(node.**currentStyle**){  
 styleNode = node.**currentStyle**;  
 }**else** {  
 styleNode = getComputedStyle(node,**null**);  
 }  
 **return** styleNode[attr];  
}

在谷歌等浏览器使用IE node.currentStyle==undefined

返回false

## 设置节点对象的style属性

节点对象.style.样式属性名 = 值;

节点对象.style['样式属性名'] = 值;

## 案例

**var *lisNode*** = **document**.querySelectorAll(**'.tab ul li'**);

**var *divItemsNode***=**document**.querySelectorAll(**'.tab-content-item'**);  *//循环遍历每一个li，并给每一个li绑定点击事件 onclick***for**(**var *index*** = 0;***index***<***lisNode***.**length**;***index***++){  
 ***lisNode***[***index***].**xiaBiao** = ***index***;

*//当点击它要对另一个节点做改变时，对这个要点击的节点 申请一个新属性——下标。* ***lisNode***[***index***].onclick = **function**(){ *//this代表当前触发的那个节点* **for**(**var** j = 0;j<***lisNode***.**length**;j++){  
 ***lisNode***[j].**className** = **''**;  
 ***divItemsNode***[j].**className** = **'tab-content-item'**;  
 }  
 **this**.**className** = **'active'**; //当前点击的节点  
 ***divItemsNode***[**this**.**xiaBiao**].**className** = **'tab-content-item active'**; // 没有点击，但产生变化的节点  
 };  
}

## 根据关系查找节点对象

\* 节点对象.parentNode;

获取父节点对象

son.parentNode.parentNode 可以找到爷爷 以此类推

节点对象.childNodes;

获取所有的子节点对象，包含空白文本节点和标签节点对象

\* 节点对象.children;

获取所有的子节点对象，仅仅包含标签节点对象

## 节点对象的类型

节点对象.nodeType 返回当前节点对象的类型，以数字的方式

| 节点类型 | 含义 | 名称 |
| --- | --- | --- |
| Document 9 | 整个文档（document） | 文档节点 |
| Element 1 | HTML元素（比如<body>、<a>等） | 元素节点 |
| Attribute 2 | HTML元素的属性（比如class=”right”） | 属性节点 |
| Text 3 | HTML文档中出现的文本 | 文本节点 |

## 创建节点对象

方式一：

document.write('内容');

缺点：会覆盖整个网页的内容不实用

方式二：

节点对象.innerHTML = ‘内容’;

优点：对于添加多嵌套的内容操作方便。

缺点：浪费内存，影响页面性能。会覆盖部分网页元素以及事件。

用处：创建多种标签

方式三：

document.createElement('标签名');

优点：不会覆盖原有的元素的事件

缺点：对于添加嵌套多的内容操作麻烦。

用处：创建节点对象。不会自己插入网页中，而是在内存中。

一般先用createElement创建标签，再通过.innerHTML把内容添加到标签中

## 添加节点对象

方式一：

语法：父节点对象.appendChild(新的子节点对象);

作用：把一个新的子节点对象追加到父节点对象中的最后。

方式二：

语法：父节点对象.insertBefore(新的子节点对象,旧的子节点对象);

作用：把新的子节点对象 追加到父节点对象中的旧的子节点对象之前。

## 删除节点对象

方式一：

语法：父节点对象.removeChild(子节点对象);

作用：删除父节点对象中的某一个子节点对象

方式二： （了解）

语法：节点对象.remove();

作用：删除某一个节点对象 自杀

缺点：有兼容性问题，ie低版本浏览器不支持

## 拓展

### 复制节点对象

语法：节点对象.cloneNode(boolean);

注意：无法复制事件

Boolean值：

false：默认值，浅拷贝，只复制外层的标签，不复制里面的内容；

True：深拷贝,复制该节点对象的所有内容（包含外层的标签和里面的内容）；

### 替换节点对象

语法：父节点对象.replaceChild(newNode,oldNode);

作用：替换父节点对象中的某个子节点对象

## 事件类型

### 鼠标事件

onclick

作用：鼠标点击事件，鼠标点击某个节点对象的行为。

onmouseover

作用：鼠标移入事件，鼠标移入某个节点对象的行为，冒泡。

onmouseout

作用：鼠标移出事件，鼠标移出某个节点对象的行为，冒泡。

onmouseenter

作用：和onmouseover相似，鼠标移入事件，鼠标移入某个节点对象的行为。但这个事件不会产生冒泡。

onmouseleave

作用：和onmouseout相似，鼠标移出事件，鼠标移出某个节点对象的行为。但这个事件不会产生冒泡。

onmousemove

作用：鼠标移动事件，鼠标在某个节点对象上移动时产生的行为。

onmousedown

作用：鼠标按下事件，鼠标在某个节点对象上按下时产生的行为。

onmouseup

作用：鼠标弹起事件，鼠标在某个节点对象上弹起时产生的行为。

oncontextmenu【拓展】

作用：鼠标右键菜单事件，鼠标在页面上右键点击时产生的行为。

### 键盘事件

onkeydown

作用：键盘键按下事件。

onkeyup

作用：键盘键弹起事件。

### UI相关事件

onload

作用：页面加载完后（图片、视频、音频、各种文件等）要执行的程序

### 表单事件

onfocus

作用：元素获取焦点（光标）事件。针对文本框、密码框、文本域

onblur

作用：元素失去焦点事件。针对文本框、密码框、文本域

onchange

作用：元素内容改变事件。针对多选框

oninput

作用：输入事件。针对文本框、密码框、文本域

兼容性：IE8及以下版本不支持，可以用onkeyup事件代替

## 获取事件对象

### 方式一：标准方式

语法：

事件目标.事件名 = function(形参){

//这个 形参 就是当前事件相关的 事件对象

}

document.onmousemove = function(e){

// e 就是事件对象

}

### 方式二：IE低版本支持的方式

document.onmousemove = function(e){

//IE低版本支持：window.event

var \_event = window.event; //获取事件对象

}

### 兼容写法

事件目标.事件名 = function(e){

var \_e = window.event||e; //兼容写法

}

document.onmousemove = function(e){

var \_e = window.event||e; //兼容写法

}

补充：短路问题

num= 0||1 //1

Num= 1||2 //1

从左往右判断 输出第一个true值 返回的是数字本身

## 事件对象的公共属性和方法

### 事件对象的公共属性

事件对象.target

作用：获取当前的事件目标节点对象。

兼容性处理方式：var target = 事件对象.target||事件对象.srcElement;

事件对象.type

获取当前事件类型

### 事件对象的公共方法

事件对象.preventDefault();

作用：阻止和默认行为（如：浏览器默认鼠标右键显示菜单）

❀ 兼容性处理方式及代码：

document.oncontextmenu = function(e){

//获取事件对象

var \_e = e||window.event;

alert('鼠标右键被点击');

if(\_e.preventDefault){ //检测浏览器是否支持此方法

\_e.preventDefault(); //标准方式阻止浏览器的默认行为

}else{ //不支持

\_e.returnValue = false; //IE低版本方式阻止浏览器的默认行为

}

}

或

document.oncontextmenu = function(e){

//获取事件对象

var \_e = e||window.event;

alert('鼠标右键被点击');

Return false; // 直接写这个。。。

}

事件对象.stopPropagation( );

作用：阻止事件冒泡

❀ 兼容性处理方式及代码：

//阻止事件冒泡

if(\_e.stopPropagation){

//浏览器是否支持该方法 若不支持返回undefined

\_e.stopPropagation(); //标准处理方式

}else{

\_e.cancelBubble = true;//IE低版本处理方式

}

## 键盘事件对象相关的属性和方法

事件对象.altKey ctrlKey shiftKey

作用：检测是否按下键盘上的 Alt 键。 按下返回 true

事件对象.keyCode

作用：返回被敲击的键生成的 Unicode 字符码(ascii码)

## 鼠标事件对象相关的属性和方法

事件对象.clientX / 事件对象.clientY

作用：鼠标在浏览器可视区域中的坐标

事件对象.pageX / 事件对象.pageY

作用：获取鼠标在整个文档区域中的坐标

事件对象.offsetX / 事件对象.offsetY

作用：获取鼠标在指定的元素的区域中的坐标

补充：

取消window文本选中拖拽的默认行为

window.getSelection().removeAllRange();

取消图片的默认行为：

Return false；

## 总结 :

①

**document**.onmousemove=**function**(e1) {  
};

onmousemove鼠标移动范围是整个屏幕所以是 document

这里也可以写imgNode

***imgNode***.onmousedown= **function** (e) {

};

Onmousedown 鼠标按下范围是图片 所以是 imgNode

②什么时候需要写e

**document**.onmousemove=**function**(e1) {

};

鼠标移动是一个持续时间 所以要声明e

***imgNode***.onmouseup=**function**() {  
}

鼠标抬起是一个瞬间动作 所以不需要e

## 事件委托 improtant

原理：

事件代理的原理用到的就是事件冒泡和目标元素，把事件处理器添加到父元素，等待子元素事件冒泡，并且父元素能够通过target（IE为srcElement）判断是哪个子元素，从而做相应处理。

优点:

① 可以提高程序的性能。

② 可以为未来新添加的元素绑定事件。（将原有绑定事件代码替换）

代码：

***ul***.onclick = **function**(e){**var** \_e = ***window***.**event**||e; *//\_target是所触发的那个节点目标，（最先触发的那个目标→节点对象）* **var** \_target = \_e.target||\_e.srcElement; *//节点对象.tagName 获取当前节点对象的标签名，注意：以大写的方式返回* **if**(\_target.tagName.toLowerCase()==**'li'**){  
 *//\_e.target → li节点对象  
 alert*(\_target.**innerHTML**)  
 }  
  
}

## 事件监听绑定事件

用处：用一个按钮控制 一个元素是否有事件 先把这个事件写出来，用一个变量表示， 再用事件监听绑定一个按钮来控制这个元素是否具有这个事件。。。。好烧脑。。

功能：事件监听绑定事件  
 参数：  
 node 事件目标，节点对象  
 type 事件类型 string 不加on  
 fn 事件处理程序 函数的定义  
 返回值：没有，目的仅仅是实现绑定事件的功能

封装：  
**function** *addEvent*(node,type,fn){  
 **if**(node.addEventListener==**undefined**){  
 *//IE* node.attachEvent(**'on'** + type,fn);  
 }**else** {  
 node.addEventListener(type,fn);  
 }  
}  
调用：  
*addEvent*(***btn***,**'click'**,**function**(){  
 *alert*(1);  
});

事件监听

imageBox.addEventListener('transitionend', function () {

}

## 事件监听解绑事件

必须先用变量获取需要绑定的函数，然后放在监听函数里

功能：解绑事件

参数：

node 事件目标 节点对象

type 事件类型 string

handlerName 事件处理程序名称 函数

返回值：无

function removeEvent(node,type,handlerName){

if(node.removeEventListener){ //检测浏览器是否支持标准方式

//支持

node.removeEventListener(type,handlerName); //标准方式

}else{

//不支持

node.detachEvent('on' + type,handlerName); //IE方式

}

}

## 三大系列

### offset系列

节点对象.offsetWidth

作用：获取当前节点对象的宽度，返回数字，不包含单位。

宽度：width + padding(左右) + border（左右）;

节点对象.offsetHeight

作用：获取当前节点对象的高度，返回数字，不包含单位。

高度：height + padding(上下) + border(上下)

节点对象.offsetParent

获取节点对象的最近的定位的上级节点对象

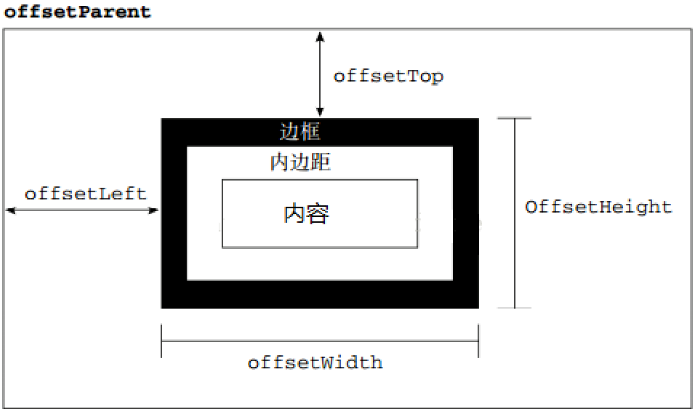
一直往上找，直到找到相对定位的父辈元素，若没有则为body

节点对象.offsetLeft

作用：获取当前节点对象的x坐标，相对于其最近的定位的上级元素的坐标。否则，相对于body。

节点对象.offsetTop

作用：获取当前节点对象的y坐标，相对于其最近的定位的上级元素的坐标。否则，相对于body。



### client系列

节点对象.clientWidth

作用：获取当前节点对象的宽度，返回数字，不包含单位。

宽度：width + padding(左右) ; （不包含border）

节点对象.clientHeight

作用：获取当前节点对象的高度，返回数字，不包含单位。

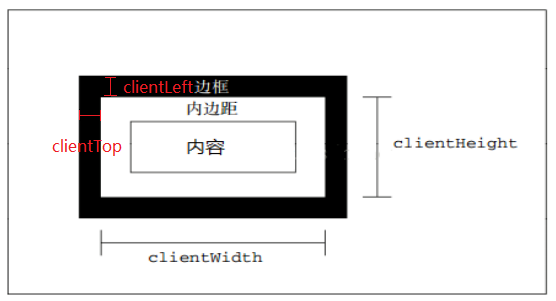
高度：height + padding(上下); （不包含border）

节点对象.clientLeft;

作用：获取当前节点对象的padding-left的外边界，距离border-left外边界的距离。实际上就是左边框的厚度。

节点对象.clientTop;

作用：获取当前节点对象的padding-top的外边界，距离border-top外边界的距离。实际上就是上边框的厚度。



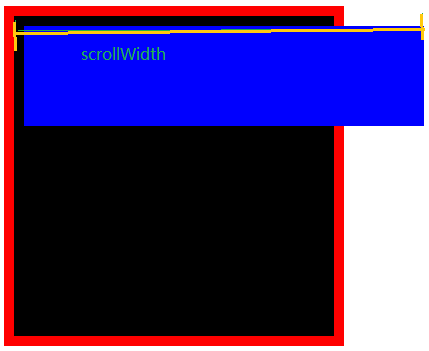
### scroll系列

节点对象.scrollWidth

作用:获取当前节点对象的宽度，返回数字，不包含单位。

宽度：width+padding（左右）+ 溢出部分 重叠部分不重复计算

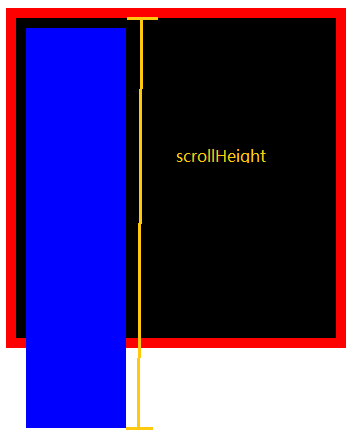
没有溢出 则为width + padding(左右) ; 和clientwidth一样



节点对象.scrollHeight

作用：获取当前节点对象的高度，返回数字，不包含单位。

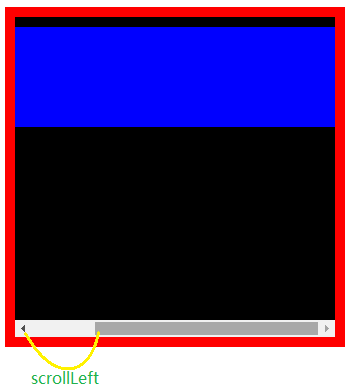
高度：height + padding(上下) + 溢出部分; 重叠部分不重复计算



节点对象.scrollLeft

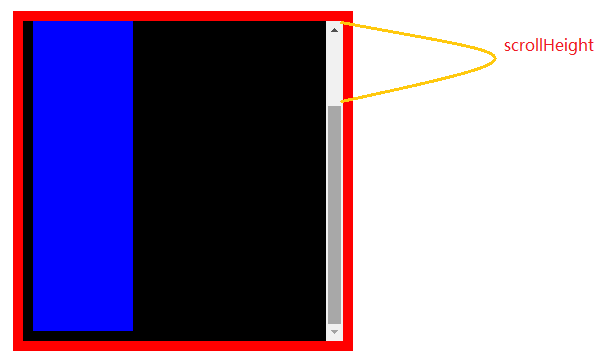
作用：获取被卷去的横向宽度

节点对象是黑色盒子 即切割滚动条内内容的模块



节点对象.scrollHeight

作用：获取被卷去的纵向高度



补充：

onscroll 滚动条事件 只要滚动条滚动就触发

但是在IE低版本浏览器给document绑定onscroll事件是没有效果的。 在IE低版本可以给window绑定onscroll,其他浏览器也支持给window绑定onscroll

***所以写成 window***.onscroll = **function**(){ }

document.body document.documentElement

scrollTop

获取滚动条被卷去的元素的间距。

（针对，最外层的滚动条）；；

**var** b = ***document***.body.scrollTop||***document***.documentElement.scrollTop;  
**var** v2 = b.**scrollTop**;

## 动画

### 常见问题

❀ 解决定时器叠加问题

在点击按钮后先清除定时器再添加定时器

***btnNode***.onclick= **function** () {  
 clearInterval(***flag***);  
 ***flag***=setInterval(**function** () {

},10);}

❀ 调用函数获取属性

封装函数：

**function** *getStyleValue*(nodeObj,propertyName){

获取属性：

nodeObj.**style**[propertyName]

❀ 停止动画

**if**(a>end) {  
 clearInterval(flag);  
 **return**;  
}

❀ 处理多个物体运动

node.flag = 定时器 目的是让每一个运动的节点对象（物体）都有属于自己的定时器的标识 和 定时器

即 把接收定时器的flag 改成 node.flag

❀ 处理连续动画

回调函数

形参中加入 fn

函数体中：

**if**(fn!=**undefined**){  
 fn();*//调用*}

调用：

*animate*(***box1***,**'left'**,600,20,**function**(){

......

}