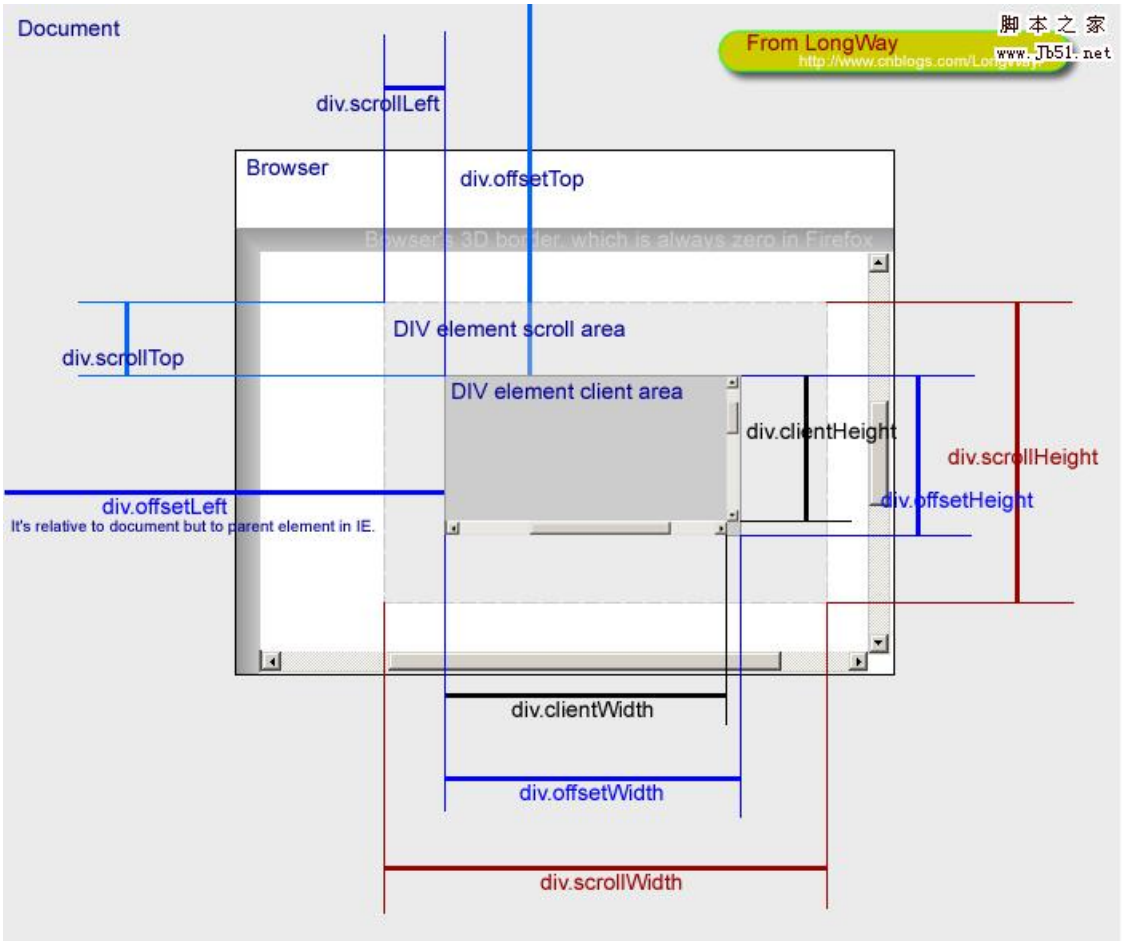
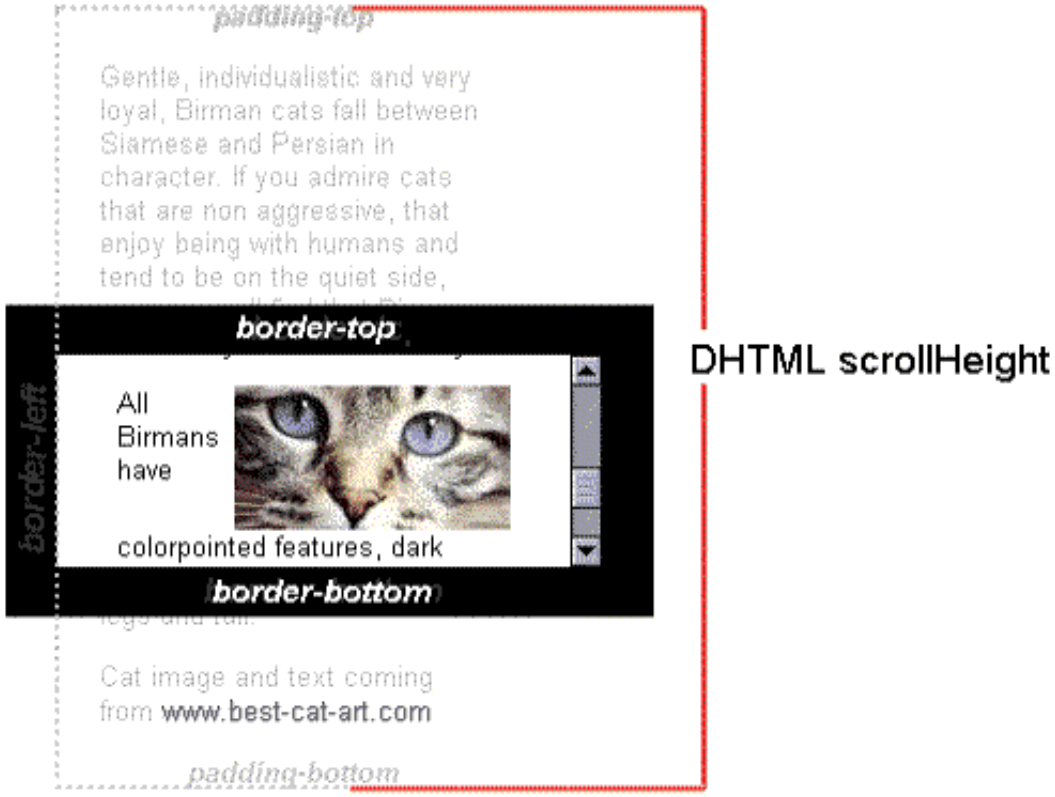
原生js获取元素位置相关信息



1、offsetTop, offsetLeft：只读属性。要确定的这两个属性的值，首先得确定元素的offsetParent。offsetParent指的是距该元素最近的position不为static的祖先元素，如果没有则指向body元素。确定了offsetParent，offsetLeft指的是元素左侧偏移offsetParent的距离，同理offsetTop指的是上侧偏移的距离。

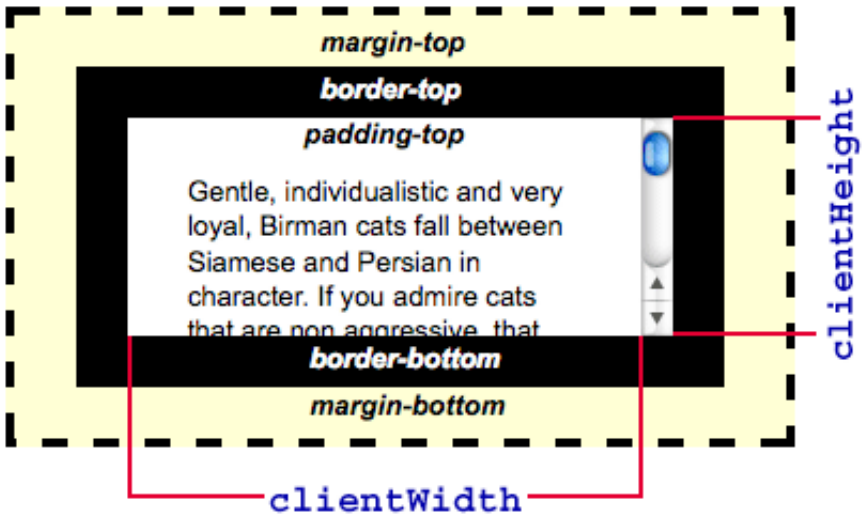
2、offsetHeight, offsetWidth：只读属性。这两个属性返回的是元素的高度或宽度，包括元素的边框、内边距和滚动条。返回值是一个经过四舍五入的整数。

3、scrollHeight, scrollWidth：只读属性。返回元素内容的整体尺寸，包括元素看不见的部分（需要滚动才能看见的）。返回值包括padding，但不包括margin和border。



4、scrollTop, scrollLeft：图中已经表示的很明白了。如果元素不能被滚动，则为0。

5、window.innerWidth, window.innerHeight：只读。视口（viewport）的尺寸，包含滚动条 clientHeight, clientWidth：包括padding，但不包括border, margin和滚动条。如下图：



6、Element.getBoundingClientRect()：只读，返回浮点值。这个方法非常有用，常用于确定元素相对于视口的位置。该方法会返回一个DOMRect对象，包含left, top, width, height, bottom, right六个属性：

left, right, top, bottom：都是元素（不包括margin）相对于视口的原点（视口的上边界和左边界）的距离。

height, width：元素的整体尺寸，包括被滚动隐藏的部分；padding和border参与计算。另外，heigth=bottom-top, width=right-left。

一、网页的大小和浏览器窗口的大小

一张网页的全部面积，就是它的大小。通常情况下，网页的大小由内容和CSS样式表决定。

浏览器窗口的大小，则是指在浏览器窗口中看到的那部分网页面积，又叫做viewport（视口）。

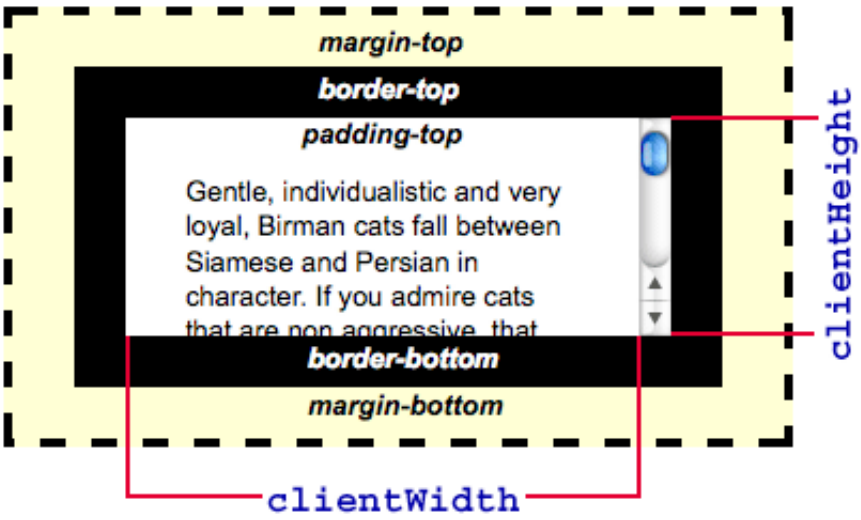
如果网页的内容能够在浏览器窗口中全部显示（也就是不出现滚动条），那么网页的大小和浏览器窗口的大小是相等的。如果不能全部显示，则滚动浏览器窗口，可以显示出网页的各个部门。

获取网页的大小

网页上每个元素都有clientHeight和clientWidth属性。这两个属性指元素的内容部分再加上padding所占据的视觉面积，他们都是只读属性。

clientHeight 属性表示元素的内部高度，以像素计。该属性包括内边距，但不包括水平滚动条（如果有）、边框和外边距。

clientWidth 属性表示元素的内部宽度，以像素计。该属性包括内边距，但不包括垂直滚动条（如果有）、边框和外边距。



因此，document元素的clientHeight和clientWidth属性，就代表了网页的大小。

```
function getViewport(){
    if (document.compatMode == "BackCompat"){
        return {
            width: document.body.clientWidth,
            height: document.body.clientHeight
        }
    } else {
        return {
            width: document.documentElement.clientWidth,
            height: document.documentElement.clientHeight
        }
    }
}
```

上面的getViewport函数就可以返回浏览器窗口的高和宽。使用的时候，有三个地方需要注意：

1. 这个函数必须在页面加载完成后才能运行，否则document对象还没生成，浏览器会报错。

2. 大多数情况下，都是document.documentElement.clientWidth返回正确值。但是，在IE6的quirks模式中，document.body.clientWidth返回正确的值，因此函数中加入了对文档模式的判断。

3. clientWidth和clientHeight都是只读属性，不能对它们赋值。

获取网页大小的另一种方法

网页上的每个元素还有scrollHeight和scrollWidth属性，指包含滚动条在内的该元素的视觉面积。

那么，document对象的scrollHeight和scrollWidth属性就是网页的大小，意思就是滚动条滚过的所有长度和宽度。

```
function getPagearea(){
       if (document.compatMode == "BackCompat"){
           return {
               width: document.body.scrollWidth,
               height: document.body.scrollHeight
           }
       } else {
           return {
               width: document.documentElement.scrollWidth,
               height: document.documentElement.scrollHeight
           }
       }
   }
```

但是，这个函数有一个问题。如果网页内容能够在浏览器窗口中全部显示，不出现滚动条，那么网页的clientWidth和scrollWidth应该相等。但是实际上，不同浏览器有不同的处理，这两个值未必相等。所以，我们需要取它们之中较大的那个值，因此要对getPagearea()函数进行改写。

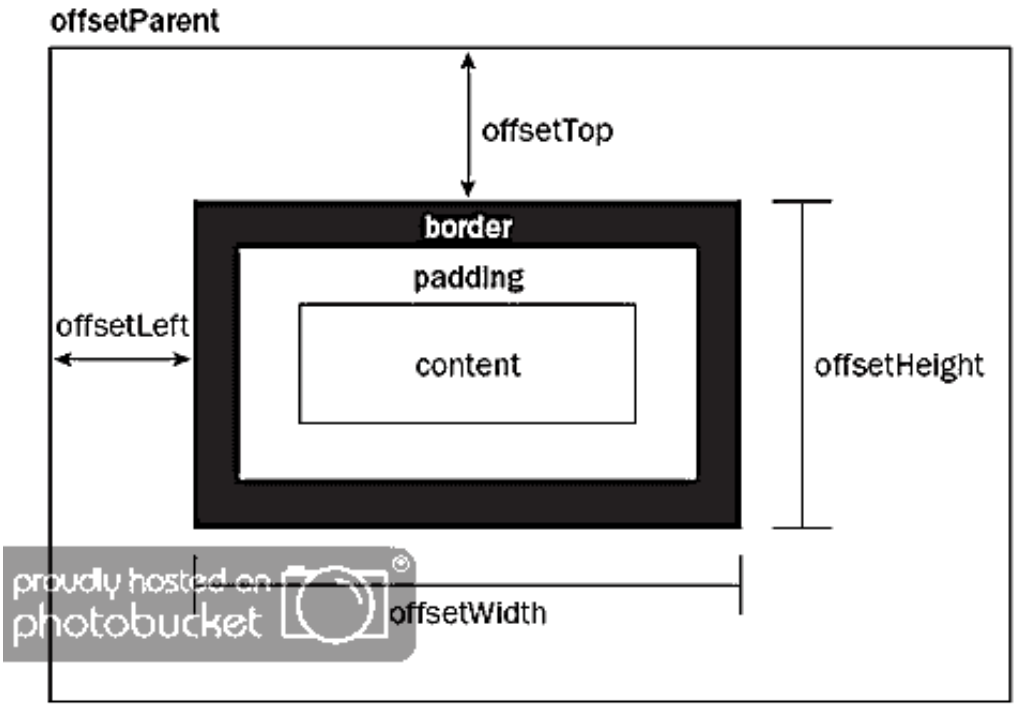
```
function getPagearea(){
       if (document.compatMode == "BackCompat"){
           return {
               width: Math.max(document.body.scrollWidth,
                               document.body.clientWidth),
               height: Math.max(document.body.scrollHeight,
                               document.body.clientHeight)
           }
       } else {
           return {
               width: Math.max(document.documentElement.scrollWidth,
                               document.documentElement.clientWidth),
               height:
Math.max(document.documentElement.scrollHeight,
                               document.documentElement.clientHeight)
```

```
        }
    }
}
```

获取网页元素的绝对位置

网页元素的绝对位置，指该元素的左上角相对于整张网页左上角的坐标。这个绝对位置要通过计算才能得到。

首先，每个元素都有offsetTop和offsetLeft属性，表示该元素的左上角与父容器（offsetParent对象）左上角的距离。所以，只需要将这两个值进行累加，就可以得到该元素的绝对坐标。




```
function getElementLeft(element){
    var actualLeft = element.offsetLeft;
    var current = element.offsetParent;

    while (current !== null){
        actualLeft += current.offsetLeft;
        current = current.offsetParent;
    }
```

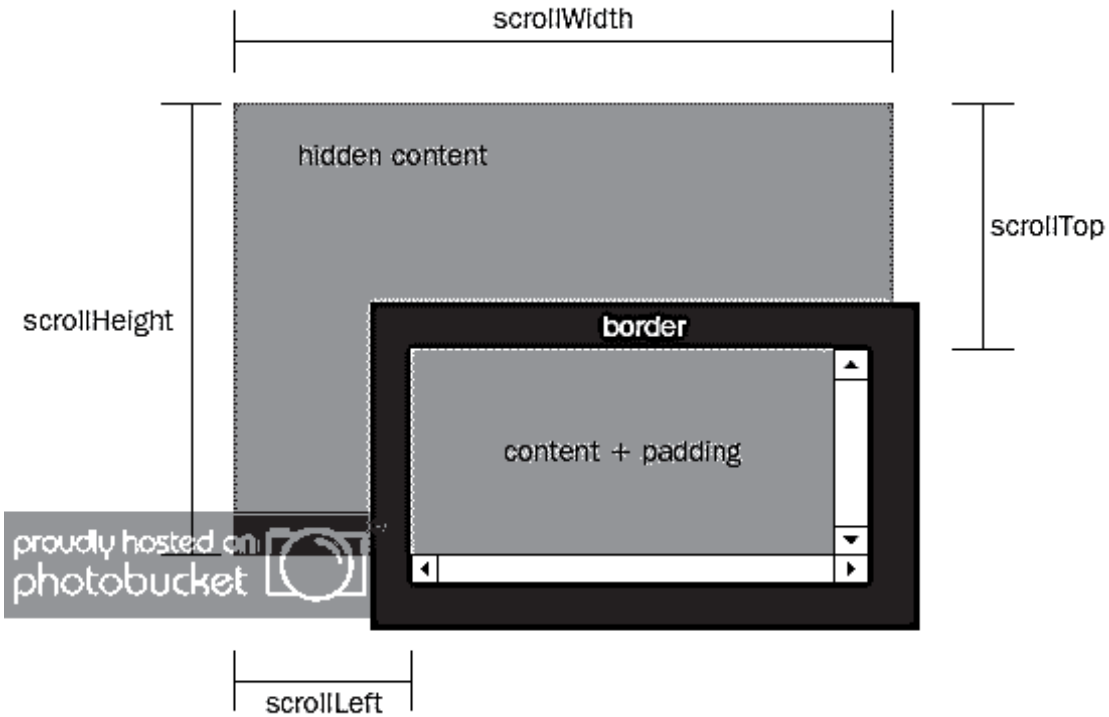
```
    return actualLeft;
}

function getElementTop(element){
    var actualTop = element.offsetTop;
    var current = element.offsetParent;

    while (current !== null){
        actualTop += current.offsetTop;
        current = current.offsetParent;
    }

    return actualTop;
}
```

由于在表格和iframe中，offsetParent对象未必等于父容器，所以上面的函数对于表格和iframe中的元素不适用。

获取网页元素的相对位置

网页元素的相对位置，指该元素左上角相对于浏览器窗口左上角的坐标。

有了绝对位置以后，获得相对位置就很容易了，只要将绝对坐标减去页面的滚动条滚动的距离就可以了。滚动条滚动的垂直距离，是document对象的scrollTop属性；滚动条滚动的水平距离是document对象的scrollLeft属性。




```
    function getElementViewLeft(element){
        var actualLeft = element.offsetLeft;
        var current = element.offsetParent;

        while (current !== null){
            actualLeft += current.offsetLeft;
            current = current.offsetParent;
        }

        if (document.compatMode == "BackCompat"){
            var elementScrollLeft=document.body.scrollLeft;
        } else {
            var
elementScrollLeft=document.documentElement.scrollLeft;
        }

        return actualLeft-elementScrollLeft;
    }

    function getElementViewTop(element){
        var actualTop = element.offsetTop;
        var current = element.offsetParent;

        while (current !== null){
```

```
            actualTop += current. offsetTop;
            current = current.offsetParent;
        }

         if (document.compatMode == "BackCompat"){
            var elementScrollTop=document.body.scrollTop;
        } else {
            var elementScrollTop=document.documentElement.scrollTop;
        }

        return actualTop-elementScrollTop;
    }
```

scrollTop和scrollLeft属性是可以赋值的，并且会立即自动滚动网页到相应位置，因此可以利用它们改变网页元素的相对位置。另外，element.scrollIntoView()方法也有类似作用，可以使网页元素出现在浏览器窗口的左上角。

获取元素位置的快速方法

除了上面的函数以外，还有一种快速方法，可以立刻获得网页元素的位置。

那就是使用getBoundingClientRect()方法。它返回一个对象，其中包含了left、right、top、bottom四个属性，分别对应了该元素的左上角和右下角相对于浏览器窗口（viewport）左上角的距离。

所以，网页元素的相对位置就是

```
var X= this.getBoundingClientRect().left;
var Y =this.getBoundingClientRect().top;
```

再加上滚动距离，就可以得到绝对位置

```
var
X=this.getBoundingClientRect().left+document.documentElement.scroll
Left;
var
Y=this.getBoundingClientRect().top+document.documentElement.scrollT
op;
```

目前，IE、Firefox 3.0+、Opera 9.5+都支持该方法，而Firefox 2.x、Safari、

Chrome、Konqueror不支持。