**DOM操作CSS**

**修改元素的样式：**

**语法：**

元素.style.样式名 = 样式值

注意：如果CSS的样式名中含有-，

这种名称在JS中是不合法的比如background-color

需要将这种样式名修改为驼峰命名法，

去掉-，然后将-后的字母大写

我们通过style属性设置的样式都是内联样式，

而内联样式有较高的优先级，所以通过JS修改的样式往往会立即显示

但是如果在样式中写了!important，则此时样式会有最高的优先级，

即使通过JS也不能覆盖该样式，此时将会导致JS修改样式失效

所以尽量不要为样式添加!important

**读取元素的内联样式**

**语法：**

元素.style.样式名

通过style属性设置和读取的都是内联样式

无法读取样式表中的样式

**获取元素的当前显示的样式**

**1.元素.currentStyle.样式名**

eg.

box1.currentStyle.width = "200px";

它可以用来读取当前元素正在显示的样式

如果当前元素没有设置该样式，则获取它的默认值

currentStyle只有IE浏览器支持，其他的浏览器都不支持

**2.getComputedStyle()**

需要两个参数

第一个：要获取样式的元素

第二个：可以传递一个伪元素，一般都传null

该方法会返回一个对象，对象中封装了当前元素对应的样式

可以通过对象.样式名来读取样式

eg.

getComputedStyle(box1,null).width

如果获取的样式没有设置，则会获取到真实的值，而不是默认值

比如：没有设置width，它不会获取到auto，而是一个长度

但是该方法不支持IE8及以下的浏览器

通过currentStyle和getComputedStyle()读取到的样式都是只读的，

不能修改，如果要修改必须通过style属性去覆盖原有的样式

**3.定义一个函数，用来获取指定元素的当前的样式**

参数：

obj 要获取样式的元素

name 要获取的样式名

function getStyle(obj , name){

if(window.getComputedStyle){//不加window的时候是变量 没找着 会报错

//加了window. 整个就是一个属性 没找着就是undefined

//正常浏览器的方式，具有getComputedStyle()方法

return getComputedStyle(obj , null)[name];

}else{

//IE8的方式，没有getComputedStyle()方法

return obj.currentStyle[name];

//之所以用[name]而不是.name是因为读取对象的属性时.name必须是一个常量

//而[name]中的name可以是一个变量

}

//上下两种写法本质上没有区别 但是ie11中两种都有 为了优先使用getComputedStyle 所以推荐上面这种

if(obj.currentStyle){

return obj.currentStyle[name];

}else{

return getComputedStyle(obj , null)[name];

}

//三目运算符

//return window.getComputedStyle?getComputedStyle(obj , null)[name]:obj.currentStyle[name];

}

**其他DOM操作的CSS属性**

以下的值只能获取不能修改

**clientWidth**

**clientHeight**

- 这两个属性可以获取元素的可见宽度和高度

- 这些属性都是不带px的，返回都是一个数字，可以直接进行计算

- 会获取元素宽度和高度，包括内容区和内边距

- 这些属性都是只读的，不能修改

**语法：**

box1.clientHeight;

box1.clientWidth;

**offsetWidth**

**offsetHeight**

- 获取元素的整个的宽度和高度，包括内容区、内边距和边框

——不能获取隐藏元素的高度和宽度

**语法：**

box1.offsetWidth

**offsetParent**

- 可以用来获取当前元素的定位父元素

- 会获取到离当前元素最近的开启了定位的祖先元素（只要position不是static）

如果所有的祖先元素都没有开启定位，则返回body

**语法：**

var op = box1.offset Parent;

**offsetLeft**

- 当前元素相对于其定位父元素的水平偏移量

**offsetTop**

- 当前元素相对于其定位父元素的垂直偏移量

**语法：**

box1.offsetLeft

**scrollWidth**

**scrollHeight**

- 可以获取元素整个滚动区域的宽度和高度

**语法：**

box4.scrollWidth

**scrollLeft**

- 可以获取水平滚动条滚动的距离

**scrollTop**

- 可以获取垂直滚动条滚动的距离

**语法：**

box4.scrollTop

注意：chrome认为浏览器的滚动条是body的

火狐等浏览器认为浏览器的滚动条是html根标签的

var st = document.documentElement.scrollTop

||document.body.scrollTop;

当满足scrollHeight - scrollTop == clientHeight

//说明垂直滚动条滚动到底了

当满足scrollWidth - scrollLeft == clientWidth

//说明水平滚动条滚动到底

**DOM修改和读取属性**

**读取**

**getAttribute()**

参数：属性名的字符串

**语法：**

oPic.getAttribute("src");//获取img的路径 返回路径名

**对象.属性名**

**语法：**

oPic.src //这种写法会返回整个带有协议的路径

**修改**

**setAttribute()**

第一个参数：属性名的字符串

第二个参数：属性的值

**语法：**

oPic.getAttribute("src","./img/1.jpg");//修改img的路径

**删除**

removeAttribute()

参数：属性名的字符串

语法：

oPic.removeAttribute("scr")//移除src这个属性

总结:checked selected..等属性不可以使用

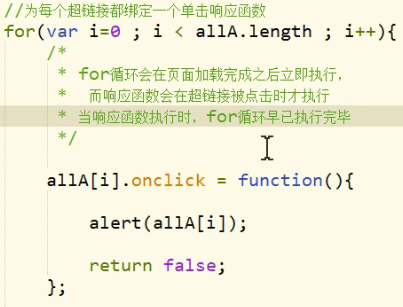
setAttribute removeAttribute getAttribute

他们有自己的一套操作方式

直接使用对象.属性名来获取或者修改

oCbox.checked = true;

**绑定单击响应函数时出现的索引问题**





**类的操作**

通过style属性来修改元素的样式，每修改一个样式，浏览器就需要重新渲染一次页面。
这样的执行的性能是比较差的，而且这种形式当我们要修改多个样式时，也不太方便。

我们可以通过修改元素的class属性来间接的修改样式

这样一来我们只需要修改一次，即可同时修改多个样式

此时浏览器只需要重新渲染页面一次，性能比较好，并且这种方式可以使表现(css)和行为(js)进一步的分离

**定义一个函数，用来向一个元素中添加指定的class属性值**

**参数:**

**obj 要添加class属性的元素**

**cn 要添加的class值**

function addClass(obj, cn) {

//检查obj中是否含有cn

if (!hasClass(obj, cn)) {

obj.className += " " + cn;

}

}

**判断一个元素中是否含有指定的class属性值**

如果有该class，则返回true，没有则返回false

function hasClass(obj, cn) {

//判断obj中有没有cn class

//创建一个正则表达式

//var reg = /\bb2\b/;//字面量不能传变量 得用构造函数

var reg = new RegExp("\\b" + cn + "\\b");

return reg.test(obj.className);

}

**删除一个元素中的指定的class属性**

function removeClass(obj, cn) {

//创建一个正则表达式

var reg = new RegExp("\\b" + cn + "\\b");

//删除class

obj.className = obj.className.replace(reg, "");

}

**切换一个类**

toggleClass可以用来切换一个类

如果元素中具有该类，则删除

如果元素中没有该类，则添加

function toggleClass(obj, cn) {

//判断obj中是否含有cn

if (hasClass(obj, cn)) {

//有，则删除

removeClass(obj, cn);

} else {

//没有，则添加

addClass(obj, cn);

}

}

在谷歌浏览器中，想要获取不可见区域的高度或宽度（滚动条移动的距离）

如果有申明头 DTD （<!DOCTYPE html>）,就用 document.documentElement.scrollTop

如果没有申明头DTD （<!DOCTYPE html>） 就用document.body.scrollTop

注意：没有DTD会认为滚动条是body的

有DTD会认为浏览器的滚动条是html根标签的

ie9以上 可以用window.pageYOffset

兼容写法

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

罗赞的封装方法

function getScroll() {

//如果有window.pageYOffset 就用这个来获取 垂直和水平的页面偏移量（滚动条滚动的距离）

        if (window.pageYOffset != undefined) {

            return {

                left: window.pageXOffset,

                top: window.pageYOffset,

            }

//如果没有这个方法 就看有没有声明头DTD 有的话document.compatMode返回"CSS1ComPat"

        } else if (document.compatMode === "CSS1Compat") {

            return {

                left: document.documentElement.scrollLeft,

                top: document.documentElement.scrollTop,

            }

        }

//否则document.compatMode返回"BackComPat"

        return {

            left: document.body.scrollLeft,

            top: document.body.scrollTop,

        }

    }

个人认为最简单的写法就是综合一下

function getScroll() {

//如果有window.pageYOffset 就用这个来获取 垂直和水平的页面偏移量（滚动条滚动的距离）

        if (window.pageYOffset != undefined) {

            return {

                left: window.pageXOffset,

                top: window.pageYOffset,

            }

//如果没有这个方法 就看有没有声明头DTD 有的话document.compatMode返回"CSS1ComPat"

        } else {

return{

left:document.documentElement.scrollLeft||document.body.scrollLeft,

top:document.documentElement.scrollTop||document.body.scrollTop

}

}

    }