[原生JS获取CSS样式并修改](http://www.cnblogs.com/webarn/p/6390334.html)

**大家都知道在JavaScript原生操作中获取元素的样式，在实际操作是使用时比较频繁的一件事，这里像大家介绍下获取css样式的方法，希望可以帮助一些需要的人，如果有幸被大牛看到，有更好的办法，欢迎提出！！！**

**一、 行内元素样式获取：**

[复制代码](javascript:void(0);)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>js获取外部样式</title>

</head>

<body>

<div id="div" style="width:200px;height:200px;border:3px solid cyan;"></div>

</body>

<script>

//获取行间样式

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

var width = div.style.width;

alert(width);//200px

</script>

</html>

[复制代码](javascript:void(0);)

 在JS代码中，alert会弹出div元素的width为200px；达到了要获取元素宽度的目的，但是这种简单的方法只适用于获取元素的行内元素样式，并不能获取内部样式和外部样式，在项目中，写入行内元素这种方案并不被大家认同，所以这种方法，只是让大家了解下。

**二、 非行间样式元素获取：**

如果元素样式并非行间样式，利用上面这种方法并不能获取到元素的样式，需要使用另一种方法：

[复制代码](javascript:void(0);)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>js获取内部部样式</title>

<style>

#div{

width: 200px;

height: 200px;

border:3px solid cyan;

}

</style>

</head>

<body>

<div id="div""></div>

</body>

<script>

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

//获取div的width

var width = window.getComputedStyle(div,null).width;

alert(width);//200px

//修改div的width

div.style.cssText='width:300px;'

</script>

</html>

[复制代码](javascript:void(0);)

在上面代码中，使用了window.getComputedStyle('元素',null).'样式',此方法兼容性:火狐 谷歌 IE9 获取的为计算后的样式,但是这里需要注意的是用此方法获取的值为只读模式,并不能修改,所以用了style.cssText方式修改了其属性,这里需要注意下书写方式.

**三 、兼容性:**

前段潜规则,凡是好的东西都不能通用,是的,你猜对了,任性的IE并不能使用以上方法,但是IE有自己的方法为currentStyle,用法一样,就不再赘述了,下面直接写出已经处理兼容的代码如下:

[复制代码](javascript:void(0);)

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

if (div.currentStyle) {

console.log(div.currentStyle.width);

}else{

　　console.log(getComputedStyle(div,null).width);

}

[复制代码](javascript:void(0);)

方法封装:

[复制代码](javascript:void(0);)

function getStyle(obj,name){

if (obj.currentStyle) {

return obj.currentStyle[name];

}else{

return getComputedStyle(obj,false)[name];

}

}

为了日后方便查询，本人翻阅了一些资料总结了以下方法，仅限原生JS，如有不对的地方欢迎指出！只求大家看完觉得有学到点什么就OK了！

**一、可以通过DOM节点对象的style对象(即CSSStyleDeclaration对象)来读写文档元素的CSS样式**

        　    如：var elm = document.getElementById('test');

                        elm.style.color = 'black';

**二、通过Element对象的getAttribute()、setAttribute()、removeAttribute()直接读写style属性**

        　    如：elm.setAttribute('style','color:red;line-height:30px');

**三、通过CSSStyleDeclaration对象的cssText属性和setProperty()、removeProperty等方法**

　　　　如：elm.style.cssText ='color:red;line-height:30px';

              　　  elm.style.removeProperty('color');

                 　   elm.style.setProperty('color', 'green', 'important');

                        elm.style.cssText = ''; //快速清空该规则的所有声明

        每一条CSS规则的样式声明部分（大括号内部的部分），都是一个CSSStyleDeclaration对象，它的属性和方法：

            属性：

                1.cssText：当前规则的所有样式声明文本。该属性可读写，即可用来设置当前规则。

                2.length：当前规则包含多少条声明。

                3.parentRule：包含当前规则的那条规则，同CSSRule接口的parentRule属性。

            方法：

                1.getPropertyPriority()方法返回指定声明的优先级，如果有的话，就是“important”，否则就是空字符串；

                2.getPropertyValue方法返回指定声明的值；

                3.item(index)方法返回指定位置的属性名，一般用[index]语法更直接；

                4.removeProperty方法用于删除一条CSS属性，返回被删除的值；

                5.setProperty方法用于设置指定的CSS属性，没有返回值;

**四、利用document.styleSheets属性，返回当前页面的所有StyleSheet对象（即所有样式表），它是一个只读的类数组对象，它的元素是CSSStyleSheet对象(继承自StyleSheet对象)，该对象的属性方法如下：**

        属性：

                1.cssRules类数组对象，元素是样式表中CSS规则CSSStyleRule对象；IE9以下为rules；

                2.disabled属性用于打开或关闭一张样式表，值为true或disabled；

                3.ownerNode属性返回StyleSheet对象所在的DOM节点，通常是<link>或<style>。对于那些由其他样式表引用的样式表，该属性为null；

                4.因为CSS的@import命令允许在样式表中加载其他样式表，就有了parentStyleSheet属性，它返回包括了当前样式表的那张样式表。如果当前样式表是顶层样式表，则该属性返回null；

                5.type属性返回StyleSheet对象的type值，通常是text/css；

                6.title属性返回StyleSheet对象的title值；

                7.href属性是只读属性，返回StyleSheet对象连接的样式表地址。对于内嵌的style节点，该属性等于null；

                8.media属性表示这个样式表是用于屏幕（screen），还是用于打印（print），或两者都适用（all），该属性只读，默认值是screen；

        方法：deleteRule()从样式表中删除一条规则，insertRule()向样式表中插入一条新规则，IE9以下为addRule()、removeRule()；

           如： document.styleSheets[0].insertRule('#test:hover{color: white;}',0);

　　　　　document.styleSheets[0].deleteRule(0); //删除样式表中的第一条规则

　　　　　document.styleSheets[0].cssRules[1].selectorText; //返回选择器字符串

　　　　　document.styleSheets[0].cssRules[1].cssText; //返回规则字符串，含选择器

　　　　　document.styleSheets[0].cssRules[1].style.border;

　　　　　document.styleSheets[0].cssRules[1].style.cssText; //返回当前规则的所有样式声明字符串

**五、用window对象的getComputedStyle方法，第一个参数是Element对象，第二个参数可以是null、空字符串、伪元素字符串，该方法返回一个只读的表示计算样式的CSSStyleDeclaration对象，它代表了实际应用在指定元素上的最终样式信息，即各种CSS规则叠加后的结果；**

        如：var color = window.getComputedStyle(elm, ':before').color;

                var color = window.getComputedStyle(elm, ':before').getPropertyValue('color');

        或：var color = window.getComputedStyle(elm, null).color;

        表示计算样式的CSSStyleDeclaration对象与表示内联样式的CSSStyleDeclaration对象的区别：

            1.计算样式的属性是只读的；

            2.计算样式的值是绝对值，类似百分比和点之类相对的单位将全部转换为以'px'为后缀的字符串绝对值，其值是颜色的属性将以“rgb（#，#，#）”或“rgba（#，#，#，#）”的格式返回;

            3.不计算复合属性，只基于最基础的属性，如不要查询margin，而单独查询marginTop等;

            4.计算样式对象未定义cssText属性；

            5.计算样式同时具有欺骗性，使用时需注意，在查询某些属性时的返回值不一定精准，如查询font-family；

            6.IE9以下不支持getComputedStyle方法，IE的Element对象有currentStyle属性；

**六、直接添加样式表**

        1.创建标签<style>添加一张内置样式表

            var style1 = document.createElement('style');

            style1.innerHTML = 'body{color:red}#top:hover{background-color: red;color: white;}';

            document.head.appendChild(style1);

        2.另一种是添加外部样式表，即在文档中添加一个link节点，然后将href属性指向外部样式表的URL

            var link1 = document.createElement('link');

            link1.setAttribute('rel', 'stylesheet');

            link1.setAttribute('type', 'text/css');

            link1.setAttribute('href', 'reset-min.css');

            document.head.appendChild(link1);

**七、window.matchMedia方法用来检查CSS的mediaQuery语句。各种浏览器的最新版本（包括IE 10+）都支持该方法，对于不支持该方法的老式浏览器，可以使用第三方函数库**[**matchMedia.js**](https://github.com/paulirish/matchMedia.js/)**；**

            下面是mediaQuery语句的一个例子：

                @media all and (max-device-width: 700px) {

                      body {background: #FF0;}

                }

        window.matchMedia方法接受一个mediaQuery语句的字符串作为参数，返回一个MediaQueryList对象。该对象有以下两个属性：

        media：返回所查询的mediaQuery语句字符串。

        matches：返回一个布尔值，表示当前环境是否匹配查询语句。

                var result = window.matchMedia('(max-width: 700px)');  
                if (result.matches) {  
                      console.log('页面宽度小于等于700px');  
                } else {  
                      console.log('页面宽度大于700px');

                }

        window.matchMedia方法返回的MediaQueryList对象有两个方法，用来监听事件：addListener方法和removeListener方法。如果mediaQuery查询结果发生变化，就调用指定的回调函数；

                var mql =  window.matchMedia("(max-width: 700px)");  
                mql.addListener(mqCallback);// 指定回调函数  
                mql.removeListener(mqCallback);// 撤销回调函数

                function mqCallback(mql) {

                      if (mql.matches) {// 宽度小于等于700像素}

                      else { // 宽度大于700像素}

                }

本文参考：

MDN： <https://developer.mozilla.org/zh-CN/docs/Web/API>

阮一峰javascript参考：<http://javascript.ruanyifeng.com/dom/css.html>

javascript权威指南第六版