style对象的属性和方法

cssText

通过这个style对象的属性能访问到style特性（写在外联表里的不算）中的CSS代码。

在写入模式下，赋给cssText的值会重写整个style特性的值（写在外联表里的不算），也就是说，以前通过style特性指定的样式信息都将丢失。

格式：

myDiv.style.cssText = " color: red; *background*: **black**; border: 20px dashed blue";

HTML

<**p** style=**"***border*: **dashed blue** 20**px"**>**an example**</**p**>

CSS

p{  
 *color*: **red**;  
 *background*: **black**;  
}

JS

var *P* = document.getElementsByTagName("p")[0];  
alert(*P*.style.cssText);// border: 20px dashed blue;

//这时的全部样式为，{color: red; *background*: **black**; border: 20px dashed blue;}  
*P*.style.cssText = "color:green";  
alert(*P*.style.cssText);// color: green;

//这时的全部样式为，{color: green; *background*: **black**;}  
}

getPropertyValue()方法取得的始终是CSS属性值的字符串表示；依然是只能访问到style特性

var *value* = *P*.style.getPropertyValue(*P*.style[0]);//P.style.item(0)也可以  
alert(*value*);//green

item（index）方法，返回给定位置的css属性的名称

还有一个getPropertyCSSValue，但是不常用，在chrome中测试

Uncaught TypeError: P.style.getPropertyCSSValue is not a function

据说要被w3c废弃，最新的草案里没有了。有些浏览器也不支持。

removeProperty(propertyName);

从元素的样式中移除某个CSS属性，使用这个方法移除一个属性，意味着将会为该属性应用默认的样式（从其他的样式表经层叠而来）

var *P* = document.getElementsByTagName("p")[0];  
*P*.style.cssText = "color:green;background-color : red";  
var *value* = *P*.style.getPropertyValue(*P*.style[1]);  
*P*.style.removeProperty("background");  
alert(*value*);//red

alert(*P*.style.cssText);// color: green;

最终背景颜色为外联样式表里的black，但是alert(value)返回的还是red

但是alert(P.style.cssText)还是正常的返回了移除样式以后的结果

可以接收行内元素（style特性）样式和其他样式表层叠而来并影响到当前元素的样式信息。

getComputedStyle()方法（除了IE都支持）

接受两个参数：要取得计算样式的元素和一个伪元素字符串（例如：after）,如果不需要伪元素信息，第二个参数可以是null

一个小例子

var *computedStyle* = document.defaultView.getComputedStyle(*P*,null);//或者window.defaultView.getComputedStyle(P,null)  
alert(*computedStyle*);//[object CSSStyleDeclaration]  
alert(*computedStyle*.backgroundColor);//rgb(0, 0, 0)

alert(*computedStyle*.color);rgb(255, 192, 203);//就是在行内定义的pink

document.defaultView属性返回当前 document 对象所关联的 window 对象，如果没有，会返回 null。IE9以下不支持。基本上用window.getComputedStyle()就可以了，只有一种情况必须用defaultView，就是在firefox3.6上访问子框架内的样式 (iframe)

注意：computedStyle.border不会在所有浏览器中都返回值（因为不同浏览器解释综合属性（如border）的方式不同，因为设置这种属性实际上会设计很多其他属性），但是computedStyle.borderLeftWidth会返回值。

currentStyle属性（IE支持，类似getComputedStyle()方法）

var *computedStyle* = *P*.currentStyle;  
alert(*computedStyle*);//[object]  
alert(*computedStyle*.backgroundColor);//black  
alert(*computedStyle*.color);rgb(255, 192, 203);//pink

注意：无论在哪个浏览器中，所有计算的样式都是只读的，不能修改计算后样式对象中的CSS属性，此外，计算后的样式也包含属于浏览器内部样式表的样式信息，因此任何具有默认值的CSS属性都会表现在计算后的样式中。

PS：不要指望某个CSS属性的默认值在不同浏览器中是相同的，如果你需要元素具有某个特定的默认值，应该手工在样式表中指定该值。

操作样式表：

CSSStyleSheet类型表示的是样式表，包括通过<link>元素包含的样式表和在<style>元素中定义的样式表。

除了属性disabled：表示样式表是否禁用的布尔值。是可读可写的。

其他的属性、方法一律只可读。

应用于文档的所有样式表是通过document.styleSheets集合来表示的。通过这个集合的length属性可以获知文档中样式表的数量，而通过方括号语法或item()方法可以访问每一个样式表。

var *sheet* = document.styleSheets[0];  
alert(*sheet*.href);//file:///G:/17%20vacation/example.css

DOM规定了一个包含 CSSStyleSheet对象的属性，名叫sheet（不兼容IE）;IE支持styleSheet属性，

例子：在不同浏览器中都能取得样式表对象的一个函数

function getStyleSheet(*element*) {  
 return element.sheet || element.styleSheet;  
}  
var *link* = document.getElementsByTagName("link")[0];  
var *sheet* = getStyleSheet(*link*);  
alert(*sheet*.href);//file:///G:/17%20vacation/example.css

CSS规则

var *sheet* = document.styleSheets[0];  
var *rules* = *sheet*.cssRules || *sheet*.rules;  
var *rule* = *rules*[0];  
alert(*rule*.selectorText);//p  
alert(*rule*.style.color);//red  
alert(*rule*.cssText);//p { color: red; background: black none repeat scroll 0% 0%; }  
alert(*rule*.style.cssText);//color: red; background: black none repeat scroll 0% 0%;

在本地运行时，chrome报错

example1.js:201 Uncaught TypeError: Cannot read property '0' of nul

ff和ie表现正常。

style.cssText属性和cssText属性类似，但不相同，前者只包含样式信息，后者还包含选择符文本和围绕样式信息的花括号。

rule.style.backgroundColor = "grey";

注意：以这种方式修改规则会影响页面中适用于该规则的所有元素。

就是说，有多少个p元素，就会有多少个p元素应用这个样式。

创建规则

insertRule()方法，接受两个参数：规则文本和表示在哪里插入规则的索引

为了兼容IE8及以下，再加一个addRule()

该方法接受三个参数，

两个必选参数：选择符文本和CSS样式信息；

一个可选参数：插入规则的位置。

sheet.addRule("body", "background-color: grey", 0);

有关这个方法的规定中说，最多可以使用addRule()添加4095条样式规则，超过这个上限的调用会导致错误。

跨浏览器的函数：接受四个参数

要向其中添加规则的样式表以及addRule（）方法的三个参数

function insertRule(*sheet*, *selectorText*, *cssText*, *position*) {  
 if(*sheet*.insertRule){  
 *sheet*.insertRule(*selectorText* + "{" + *cssText* + "}", *position*);  
 }else if(*sheet*.addRule){  
 *sheet*.addRule(*selectorText*,*cssText*,*position*)  
 }  
}  
insertRule(document.styleSheets[0],"body","background-color: red",0);

虽然可以像这样添加规则，但是随着要添加规则的增多，这种方法就会变得非常繁琐，因此，如果要添加的规则非常多，建议用动态加载样式表的技术。

删除规则

deleteRule()，接受一个参数：

要删除的规则的位置。

由于IE又有一个自己的方法removeRule(),所以再写一个兼容函数

function deleteRule(*sheet*, *index*) {  
 if (*sheet*.deleteRule){  
 *sheet*.deleteRule(*index*);  
 } else if (*sheet*.removeRule){  
 *sheet*.removeRule(*index*)  
 }  
}  
deleteRule(document.styleSheets[0],0);

删除规则可能会影响CSS层叠的效果，谨慎使用！！