第 21 章 DOM 操作表格及样式

学习要点：

1.操作表格

2.操作样式

主讲教师：李炎恢
合作网站：http://www.ibeifeng.com
讲师博客：http://hi.baidu.com/李炎恢

DOM 在操作生成 HTML 上，还是比较简明的。不过，由于浏览器总是存在兼容和陷阱，导致最终的操作就不是那么简单方便了。本章主要了解一下 DOM 操作表格和样式的一些知识。

一．操作表格

<table>标签是 HTML 中结构最为复杂的一个，我们可以通过 DOM 来创建生成它，或者 HTML DOM 来操作它。(PS：HTML DOM 提供了更加方便快捷的方式来操作 HTML，有手册)。

```
//需要操作的 table
<table border="1" width="300">
    <caption>人员表</caption>
    <thead>
        <tr>
            <th>姓名</th>
            <th>性别</th>
            <th>年龄</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>张三</td>
            <td>男</td>
            <td>20</td>
        </tr>
        <tr>
            <td>李四</td>
            <td>女</td>
            <td>22</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td colspan="3">合计：N</td>
```

```
            </tr>
        </tfoot>
</table>

//使用 DOM 来创建这个表格
var table = document.createElement('table');
table.border = 1;
table.width = 300;

var caption = document.createElement('caption');
table.appendChild(caption);
caption.appendChild(document.createTextNode('人员表'));

var thead = document.createElement('thead');
table.appendChild(thead);

var tr = document.createElement('tr');
thead.appendChild(tr);

var th1 = document.createElement('th');
var th2 = document.createElement('th');
var th3 = document.createElement('th');

tr.appendChild(th1);
th1.appendChild(document.createTextNode('姓名'));
tr.appendChild(th2);
th2.appendChild(document.createTextNode('年龄'));

document.body.appendChild(table);
```

PS：使用 DOM 来创建表格其实已经没有什么难度，就是有点儿小烦而已。下面我们再使用 HTML DOM 来获取和创建这个相同的表格。

HTML DOM 中，给这些元素标签提供了一些属性和方法

属性或方法	说明
caption	保存着<caption>元素的引用
tBodies	保存着<tbody>元素的 HTMLCollection 集合
tFoot	保存着对<tfoot>元素的引用
tHead	保存着对<thead>元素的引用
rows	保存着对<tr> 元素的 HTMLCollection 集合
createTHead()	创建<thead>元素，并返回引用
createTFoot()	创建<tfoot>元素，并返回引用

createCaption()	创建<caption>元素，并返回引用
deleteTHead()	删除<thead>元素
deleteTFoot()	删除<tfoot>元素
deleteCaption()	删除<caption>元素
deleteRow(pos)	删除指定的行
insertRow(pos)	向 rows 集合中的指定位置插入一行

<tbody>元素添加的属性和方法

属性或方法	说明
rows	保存着<tbody>元素中行的 HTMLCollection
deleteRow(pos)	删除指定位置的行
insertRow(pos)	向 rows 集合中的指定位置插入一行，并返回引用

<tr>元素添加的属性和方法

属性或方法	说明
cells	保存着<tr>元素中单元格的 HTMLCollection
deleteCell(pos)	删除指定位置的单元格
insertCell(pos)	向 cells 集合的指定位置插入一个单元格，并返回引用

PS：因为表格较为繁杂，层次也多，在使用之前所学习的 DOM 只是来获取某个元素会非常难受，所以使用 HTML DOM 会清晰很多。

```
//使用 HTML DOM 来获取表格元素
var table = document.getElementsByTagName('table')[0];  //获取 table 引用

//按照之前的 DOM 节点方法获取<caption>
alert(table.children[0].innerHTML);                    //获取 caption 的内容
```

PS：这里使用了 children[0]本身就忽略了空白，如果使用 firstChild 或者 childNodes[0]需要更多的代码。

```
//按 HTML DOM 来获取表格的<caption>
alert(table.caption.innerHTML);                        //获取 caption 的内容

//按 HTML DOM 来获取表头表尾<thead>、<tfoot>
alert(table.tHead);                                    //获取表头
alert(table.tFoot);                                    //获取表尾

//按 HTML DOM 来获取表体<tbody>
alert(table.tBodies);                                  //获取表体的集合
```

PS：在一个表格中<thead>和<tfoot>是唯一的，只能有一个。而<tbody>不是唯一的可以有多个，这样导致最后返回的<thead>和<tfoot>是元素引用，而<tbody>返回的是元素集合。

```
//按 HTML DOM 来获取表格的行数
alert(table.rows.length);                    //获取行数的集合，数量

//按 HTML DOM 来获取表格主体里的行数
alert(table.tBodies[0].rows.length);         //获取主体的行数的集合，数量

//按 HTML DOM 来获取表格主体内第一行的单元格数量(tr)
alert(table.tBodies[0].rows[0].cells.length);    //获取第一行单元格的数量

//按 HTML DOM 来获取表格主体内第一行第一个单元格的内容(td)
alert(table.tBodies[0].rows[0].cells[0].innerHTML);  //获取第一行第一个单元格的内容

//按 HTML DOM 来删除标题、表头、表尾、行、单元格
table.deleteCaption();                       //删除标题
table.deleteTHead();                         //删除<thead>
table.tBodies[0].deleteRow(0);               //删除<tr>一行
table.tBodies[0].rows[0].deleteCell(0);      //删除<td>一个单元格

//按 HTML DOM 创建一个表格
var table = document.createElement('table');
table.border = 1;
table.width = 300;

table.createCaption().innerHTML = '人员表';

//table.createTHead();
//table.tHead.insertRow(0);
var thead = table.createTHead();
var tr = thead.insertRow(0);

var td = tr.insertCell(0);
td.appendChild(document.createTextNode('数据'));

var td2 = tr.insertCell(1);
td2.appendChild(document.createTextNode('数据 2'));

document.body.appendChild(table);
```

PS：在创建表格的时候<table>、<tbody>、<th>没有特定的方法，需要使用 document 来创建。也可以模拟已有的方法编写特定的函数即可，例如：insertTH()之类的。

二．操作样式

CSS 作为(X)HTML 的辅助，可以增强页面的显示效果。但不是每个浏览器都能支持最新的 CSS 能力。CSS 的能力和 DOM 级别密切相关，所以我们有必要检测当前浏览器支持 CSS 能力的级别。

DOM1 级实现了最基本的文档处理，DOM2 和 DOM3 在这个基础上增加了更多的交互能力，这里我们主要探讨 CSS，DOM2 增加了 CSS 编程访问方式和改变 CSS 样式信息。

DOM 一致性检测

功能	版本号	说明
Core	1.0、2.0、3.0	基本的 DOM,用于表现文档节点树
XML	1.0、2.0、3.0	Core 的 XML 扩展，添加了对 CDATA 等支持
HTML	1.0、2.0	XML 的 HTML 扩展，添加了对 HTML 特有元素支持
Views	2.0	基于某些样式完成文档的格式化
StyleSheets	2.0	将样式表关联到文档
CSS	2.0	对层叠样式表 1 级的支持
CSS2	2.0	对层叠样式表 2 级的支持
Events	2.0	常规的 DOM 事件
UIEvents	2.0	用户界面事件
MouseEvents	2.0	由鼠标引发的事件(如：click)
MutationEvents	2.0	DOM 树变化时引发的事件
HTMLEvents	2.0	HTML4.01 事件
Range	2.0	用于操作 DOM 树中某个范围的对象和方法
Traversal	2.0	遍历 DOM 树的方法
LS	3.0	文件与 DOM 树之间的同步加载和保存
LS-Async	3.0	文件与 DOM 树之间的异步加载和保存
Valuidation	3.0	在确保有效的前提下修改 DOM 树的方法

```
//检测浏览器是否支持 DOM1 级 CSS 能力或 DOM2 级 CSS 能力
alert('DOM1 级 CSS 能力：' + document.implementation.hasFeature('CSS', '2.0'));
alert('DOM2 级 CSS 能力：' + document.implementation.hasFeature('CSS2', '2.0'));
```

PS：这种检测方案在 IE 浏览器上不精确，IE6 中，hasFeature()方法只为 HTML 和版本 1.0 返回 true，其他所有功能均返回 false。但 IE 浏览器还是支持最常用的 CSS2 模块。

1.访问元素的样式

任何 HTML 元素标签都会有一个通用的属性：style。它会返回 CSSStypeDeclaration 对象。下面我们看几个最常见的行内 style 样式的访问方式。

CSS 属性及 JavaScript 调用

CSS 属性	JavaScript 调用
color	style.color
font-size	style.fontSize
float	非 IE：style.cssFloat
float	IE：style.styleFloat

```
var box = document.getElementById('box');          //获取 box
box.style.cssFloat.style;                          //CSSStyleDeclaration
box.style.cssFloat.style.color;                    //red
box.style.cssFloat.style.fontSize;                 //20px
box.style.cssFloat || box.style.styleFloat;        //left，非 IE 用 cssFloat，IE 用 styleFloat
```

PS：以上取值方式也可以赋值，最后一种赋值可以如下：

```
typeof box.style.cssFloat != 'undefined' ?
                box.style.cssFloat = 'right' : box.style.styleFloat = 'right';
```

DOM2 级样式规范为 style 定义了一些属性和方法

属性或方法	说明
cssText	访问或设置 style 中的 CSS 代码
length	CSS 属性的数量
parentRule	CSS 信息的 CSSRule 对象
getPropertyCSSValue(name)	返回包含给定属性值的 CSSValue 对象
getPropertyPriority(name)	如果设置了!important，则返回，否则返回空字符串
item(index)	返回指定位置 CSS 属性名称
removeProperty(name)	从样式中删除指定属性
setProperty(name,v,p)	给属性设置为相应的值，并加上优先权

```
box.style.cssText;                                 //获取 CSS 代码
//box.style.length;                                //3，IE 不支持
//box.style.removeProperty('color');               //移除某个 CSS 属性，IE 不支持
//box.style.setProperty('color','blue');           //设置某个 CSS 属性，IE 不支持
```

PS：Firefox、Safari、Opera9+、Chrome 支持这些属性和方法。IE 只支持 cssText，而 getPropertyCSSValue()方法只有 Safari3+和 Chrome 支持。

PS：style 属性仅仅只能获取行内的 CSS 样式，对于另外两种形式内联<style>和链接<link>方式则无法获取到。

虽然可以通过 style 来获取单一值的 CSS 样式，但对于复合值的样式信息，就需要通过计算样式来获取。DOM2 级样式，window 对象下提供了 getComputedStyle()方法。接受两个

参数，需要计算的样式元素，第二个伪类(:hover)，如果没有没有伪类，就填 null。

PS：IE 不支持这个 DOM2 级的方法，但有个类似的属性可以使用 currentStyle 属性。

```
var box = document.getElementById('box');
var style = window.getComputedStyle ?
                    window.getComputedStyle(box, null) : null || box.currentStyle;
alert(style .color);                          //颜色在不同的浏览器会有 rgb()格式
alert(style .border);                         //不同浏览器不同的结果
alert(style .fontFamily);                     //计算显示复合的样式值
alert(box.style.fontFamily);                  //空
```

PS：border 属性是一个综合属性，所以他在 Chrome 显示了，Firefox 为空，IE 为 undefined。所谓综合性属性，就是 XHTML 课程里所的简写形式，所以，DOM 在获取 CSS 的时候，最好采用完整写法兼容性最好，比如：border-top-color 之类的。

2.操作样式表

使用 style 属性可以设置行内的 CSS 样式，而通过 id 和 class 调用是最常用的方法。

```
box.id = 'pox';                               //把 ID 改变会带来灾难性的问题
box.className = 'red';                        //通过 className 关键字来设置样式
```

在添加 className 的时候，我们想给一个元素添加多个 class 是没有办法的，后面一个必将覆盖掉前面一个，所以必须来写个函数：

```
//判断是否存在这个 class
function hasClass(element, className) {
    return element.className.match(new RegExp('(\\s|^)'+className+'(\\s|$)'));
}

//添加一个 class，如果不存在的话
function addClass(element, className) {
    if (!hasClass(element, className))      {
        element.className += " "+className;
    }
}

//删除一个 class，如果存在的话
function removeClass(element, className) {
    if (hasClass(element, className)) {
        element.className = element.className.replace(
                            new RegExp('(\\s|^)'+className+'(\\s|$)'),' ');
    }
}
```

之前我们使用 style 属性，仅仅只能获取和设置行内的样式，如果是通过内联<style>或链接<link>提供的样式规则就无可奈何了，然后我们又学习了 getComputedStyle 和 currentStyle，这只能获取却无法设置。

CSSStyleSheet 类型表示通过<link>元素和<style>元素包含的样式表。

```
document.implementation.hasFeature('StyleSheets', '2.0')    //是否支持 DOM2 级样式表
document.getElementsByTagName('link')[0];      //HTMLLinkElement
document.getElementsByTagName('style')[0];     //HTMLStyleElement
```

这两个元素本身返回的是 HTMLLinkElement 和 HTMLStyleElement 类型，但 CSSStyleSheet 类型更加通用一些。得到这个类型非 IE 使用 sheet 属性，IE 使用 styleSheet；

```
var link = document.getElementsByTagName('link')[0];
var sheet = link.sheet || link.styleSheet;              //得到 CSSStyleSheet
```

属性或方法	说明
disabled	获取和设置样式表是否被禁用
href	如果是通过<link>包含的，则样式表为 URL，否则为 null
media	样式表支持的所有媒体类型的集合
ownerNode	指向拥有当前样式表节点的指针
parentStyleSheet	@import 导入的情况下，得到父 CSS 对象
title	ownerNode 中 title 属性的值
type	样式表类型字符串
cssRules	样式表包含样式规则的集合，IE 不支持
ownerRule	@import 导入的情况下，指向表示导入的规则，IE 不支持
deleteRule(index)	删除 cssRules 集合中指定位置的规则，IE 不支持
insertRule(rule, index)	向 cssRules 集合中指定位置插入 rule 字符串，IE 不支持

```
sheet.disabled;                                 //false，可设置为 true
sheet.href;                                     //css 的 URL
sheet.media;                                    //MediaList，集合
sheet.media[0];                                 //第一个 media 的值
sheet.title;                                    //得到 title 属性的值
sheet.cssRules                                  //CSSRuleList，样式表规则集合
sheet.deleteRule(0);                            //删除第一个样式规则
sheet.insertRule("body {background-color:red}", 0);  //在第一个位置添加一个样式规则
```

PS：除了几个不用和 IE 不支持的我们忽略了，还有三个有 IE 对应的另一种方式：

```
sheet.rules;                                    //代替 cssRules 的 IE 版本
sheet.removeRule(0);                            //代替 deleteRule 的 IE 版本
sheet.addRule("body", "background-color:red", 0);//代替 insertRule 的 IE 版本
```

除了刚才的方法可以得到 CSSStyleSheet 类型，还有一种方法是通过 document 的 styleSheets 属性来获取。

```
document.styleSheets;                    //StyleSheetList，集合
var sheet = document.styleSheets[0];     //CSSStyleSheet，第一个样式表对象
```

为了添加 CSS 规则，并且兼容所有浏览器，我们必须写一个函数：

```
var sheet = document.styleSheets[0];
insertRule(sheet, "body", "background-color:red;", 0);

function insertRule(sheet, selectorText, cssText, position) {
    //如果是非 IE
    if (sheet.insertRule) {
        sheet.insertRule(selectorText + "{" + cssText + "}", position);
    //如果是 IE
    } else if (sheet.addRule) {
        sheet.addRule(selectorText, cssText, position);
    }
}
```

为了删除 CSS 规则，并且兼容所有浏览器，我们必须写一个函数：

```
var sheet = document.styleSheets[0];
deleteRule(sheet, 0);

function deleteRule(sheet, index) {
    //如果是非 IE
    if (sheet.deleteRule) {
        sheet.deleteRule(index);
    //如果是 IE
    } else if (sheet.removeRule) {
        sheet.removeRule(index);
    }
}
```

通过 CSSRules 属性(非 IE)和 rules 属性(IE)，我们可以获得样式表的规则集合列表。这样我们就可以对每个样式进行具体的操作了。

```
var sheet = document.styleSheets[0];          //CSSStyleSheet
var rules = sheet.cssRules || sheet.rules;    //CSSRuleList，样式表的规则集合列表
var rule = rules[0];                          //CSSStyleRule，样式表第一个规则
```

CSSStyleRule 可以使用的属性

属性	说明
cssText	获取当前整条规则对应的文本，IE 不支持
parentRule	@import 导入的，返回规则或 null，IE 不支持

parentStyleSheet	当前规则的样式表，IE 不支持
selectorText	获取当前规则的选择符文本
style	返回 CSSStyleDeclaration 对象，可以获取和设置样式
type	表示规则的常量值，对于样式规则，值为 1，IE 不支持

```
rule.cssText;                    //当前规则的样式文本
rule.selectorText;               //#box，样式的选择符
rule.style.color;                //red，得到具体样式值
```

PS：Chrome 浏览器在本地运行时会出现问题，rules 会变成 null，只要把它放到服务器上允许即可正常。

总结：三种操作 CSS 的方法，第一种 style 行内，可读可写；第二种行内、内联和链接，使用 getComputedStyle 或 currentStyle，可读不可写；第三种 cssRules 或 rules，内联和链接可读可写。

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(yc60.com)联合提供：

本次主讲老师：李炎恢

我的博客：hi.baidu.com/李炎恢/

我的邮件：yc60.com@gmail.com