第 20 章 DOM 进阶

学习要点：

1.DOM 类型

2.DOM 扩展

3.DOM 操作内容

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 自身存在很多类型，在 DOM 基础课程中大部分都有所接触，比如 Element 类型：表示的是元素节点，再比如 Text 类型：表示的是文本节点。DOM 也提供了一些扩展功能。

一. DOM 类型

DOM 基础课程中，我们了解了 DOM 的节点并且了解怎样查询和操作节点，而本身这些不同的节点，又有着不同的类型。

DOM 类型

类型名	说明
Node	表示所有类型值的统一接口，IE 不支持
Document	表示文档类型
Element	表示元素节点类型
Text	表示文本节点类型
Comment	表示文档中的注释类型
CDATASection	表示 CDATA 区域类型
DocumentType	表示文档声明类型
DocumentFragment	表示文档片段类型
Attr	表示属性节点类型

1.Node 类型

Node 接口是 DOM1 级就定义了，Node 接口定义了 12 个数值常量以表示每个节点的类型值。除了 IE 之外，所有浏览器都可以访问这个类型。

Node 的常量

常量名	说明	nodeType 值
ELEMENT_NODE	元素	1
ATTRIBUTE_NODE	属性	2
TEXT_NODE	文本	3

CDATA_SECTION_NODE	CDATA	4
ENTITY_REFERENCE_NODE	实体参考	5
ENTITY_NODE	实体	6
PROCESSING_INSTRUCETION_NODE	处理指令	7
COMMENT_NODE	注释	8
DOCUMENT_NODE	文档根	9
DOCUMENT_TYPE_NODE	doctype	10
DOCUMENT_FRAGMENT_NODE	文档片段	11
NOTATION_NODE	符号	12

虽然这里介绍了 12 种节点对象的属性，用的多的其实也就几个而已。

```
alert(Node.ELEMENT_NODE);                    //1，元素节点类型值
alert(Node.TEXT_NODE);                       //2，文本节点类型值
```

我们建议使用 Node 类型的属性来代替 1,2 这些阿拉伯数字，有可能大家会觉得这样岂不是很繁琐吗？并且还有一个问题就是 IE 不支持 Node 类型。

如果只有两个属性的话，用 1，2 来代替会特别方便，但如果属性特别多的情况下，1、2、3、4、5、6、7、8、9、10、11、12，你根本就分不清哪个数字代表的是哪个节点。当然，如果你只用 1，2 两个节点，那就另当别论了。

IE 不支持，我们可以模拟一个类，让 IE 也支持。

```
if (typeof Node == 'undefined') {                    //IE 返回
    window.Node = {
        ELEMENT_NODE : 1,
        TEXT_NODE : 3
    };
}
```

2.Document 类型

Document 类型表示文档，或文档的根节点，而这个节点是隐藏的，没有具体的元素标签。

```
document;                                    //document
document.nodeType;                           //9，类型值
document.childNodes[0];                      //DocumentType，第一个子节点对象
document.childNodes[0].nodeType;             //非 IE 为 10，IE 为 8
document.childNodes[1];                      //HTMLHtmlElement
document.childNodes[1].nodeName;             //HTML
```

如果想直接得到<html>标签的元素节点对象 HTMLHtmlElement，不必使用 childNodes 属性这么麻烦，可以使用 documentElement 即可。

```
document.documentElement;                    //HTMLHtmlElement
```

在很多情况下，我们并不需要得到<html>标签的元素节点，而需要得到更常用的<body>标签，之前我们采用的是：document.getElementsByTagName('body')[0]，那么这里提供一个

更加简便的方法：document.body。

```
document.body;                    //HTMLBodyElement
```

在<html>之前还有一个文档声明：<!DOCTYPE>会作为某些浏览器的第一个节点来处理，这里提供了一个简便方法来处理：document.doctype。

```
document.doctype;                 //DocumentType
```

PS：IE8 中，如果使用子节点访问，IE8 之前会解释为注释类型 Comment 节点，而 document.doctype 则会返回 null。

```
document.childNodes[0].nodeName   //IE 会是#Comment
```

在 Document 中有一些遗留的属性和对象合集，可以快速的帮助我们精确的处理一些任务。

```
//属性
document.title;                   //获取和设置<title>标签的值
document.URL;                     //获取 URL 路径
document.domain;                  //获取域名，服务器端
document.referrer;                //获取上一个 URL，服务器端

//对象集合
document.anchors;                 //获取文档中带 name 属性的<a>元素集合
document.links;                   //获取文档中带 href 属性的<a>元素集合
document.applets;                 //获取文档中<applet>元素集合，已不用
document.forms;                   //获取文档中<form>元素集合
document.images;                  //获取文档中<img>元素集合
```

3.Element 类型

Element 类型用于表现 HTML 中的元素节点。在 DOM 基础那章，我们已经可以对元素节点进行查找、创建等操作，元素节点的 nodeType 为 1，nodeName 为元素的标签名。

元素节点对象在非 IE 浏览器可以返回它具体元素节点的对象类型。

元素对应类型表

元素名	类型
HTML	HTMLHtmlElement
DIV	HTMLDivElement
BODY	HTMLBodyElement
P	HTMLParamElement

PS：以上给出了部分对应，更多的元素对应类型，直接访问调用即可。

4.Text 类型

Text 类型用于表现文本节点类型，文本不包含 HTML，或包含转义后的 HTML。文本节点的 nodeType 为 3。

在同时创建两个同一级别的文本节点的时候，会产生分离的两个节点。

```
var box = document.createElement('div');
var text = document.createTextNode('Mr.');
var text2 = document.createTextNode(Lee!);
box.appendChild(text);
box.appendChild(text2);
document.body.appendChild(box);
alert(box.childNodes.length);                    //2，两个文本节点
```

PS：把两个同邻的文本节点合并在一起使用 normalize()即可。

```
box.normalize();                                 //合并成一个节点
```

PS：有合并就有分离，通过 splitText(num)即可实现节点分离。

```
box.firstChild.splitText(3);                     //分离一个节点
```

除了上面的两种方法外，Text 还提供了一些别的 DOM 操作的方法如下：

```
var box = document.getElementById('box');
box.firstChild.deleteData(0,2);                  //删除从 0 位置的 2 个字符
box.firstChild.insertData(0,'Hello.');           //从 0 位置添加指定字符
box.firstChild.replaceData(0,2,'Miss');          //从 0 位置替换掉 2 个指定字符
box.firstChild.substringData(0,2);               //从 0 位置获取 2 个字符，直接输出
alert(box.firstChild.nodeValue);                 //输出结果
```

5.Comment 类型

Comment 类型表示文档中的注释。nodeType 是 8，nodeName 是#comment，nodeValue 是注释的内容。

```
var box = document.getElementById('box');
alert(box.firstChild);                           //Comment
```

PS：在 IE 中，注释节点可以使用！当作元素来访问。

```
var comment = document.getElementsByTagName('!');
alert(comment.length);
```

6.Attr 类型

Attr 类型表示文档元素中的属性。nodeType 为 11，nodeName 为属性名，nodeValue 为属性值。DOM 基础篇已经详细介绍过，略。

二．DOM 扩展

1.呈现模式

从 IE6 开始开始区分标准模式和混杂模式(怪异模式)，主要是看文档的声明。IE 为 document 对象添加了一个名为 compatMode 属性，这个属性可以识别 IE 浏览器的文档处于什么模式如果是标准模式，则返回 CSS1Compat，如果是混杂模式则返回 BackCompat。

```
if (document.compatMode == 'CSS1Compat') {
    alert(document.documentElement.clientWidth);
```

```
} else {
    alert(document.body.clientWidth);
}
```

PS：后来 Firefox、Opera 和 Chrome 都实现了这个属性。从 IE8 后，又引入 documentMode 新属性，因为 IE8 有 3 种呈现模式分别为标准模式 8，仿真模式 7，混杂模式 5。所以如果想测试 IE8 的标准模式，就判断 document.documentMode > 7 即可。

2.滚动

DOM 提供了一些滚动页面的方法，如下：

```
document.getElementById('box').scrollIntoView();    //设置指定可见
```

3.children 属性

由于子节点空白问题，IE 和其他浏览器解释不一致。虽然可以过滤掉，但如果只是想得到有效子节点，可以使用 children 属性，支持的浏览器为：IE5+、Firefox3.5+、Safari2+、Opera8+和 Chrome，这个属性是非标准的。

```
var box = document.getElementById('box');
alert(box.children.length);                    //得到有效子节点数目
```

4.contains()方法

判断一个节点是不是另一个节点的后代，我们可以使用 contains()方法。这个方法是 IE 率先使用的，开发人员无须遍历即可获取此信息。

```
var box = document.getElementById('box');
alert(box.contains(box.firstChild));           //true
```

PS：早期的 Firefox 不支持这个方法，新版的支持了，其他浏览器也都支持，Safari2.x 浏览器支持的有问题，无法使用。所以，必须做兼容。

在 Firefox 的 DOM3 级实现中提供了一个替代的方法 compareDocumentPosition()方法。这个方法确定两个节点之间的关系。

```
var box = document.getElementById('box');
alert(box.compareDocumentPosition(box.firstChild));    //20
```

关系掩码表

掩码	节点关系
1	无关(节点不存在)
2	居前(节点在参考点之前)
4	居后(节点在参考点之后)
8	包含(节点是参考点的祖先)
16	被包含(节点是参考点的后代)

PS：为什么会出现 20，那是因为满足了 4 和 16 两项，最后相加了。为了能让所有浏览器都可以兼容，我们必须写一个兼容性的函数。

```
//传递参考节点(父节点)，和其他节点(子节点)
function contains(refNode, otherNode) {
    //判断支持 contains，并且非 Safari 浏览器
    if (typeof refNode.contains != 'undefined' &&
                !(BrowserDetect.browser == 'Safari' && BrowserDetect.version < 3)) {
        return refNode.contains(otherNode);
    //判断支持 compareDocumentPosition 的浏览器，大于 16 就是包含
    } else if (typeof refNode.compareDocumentPosition == 'function') {
        return !!(refNode.compareDocumentPosition(otherNode) > 16);
    } else {
        //更低的浏览器兼容，通过递归一个个获取他的父节点是否存在
        var node = otherNode.parentNode;
        do {
            if (node === refNode) {
                return true;
            } else {
                node = node.parentNode;
            }
            } while (node != null);
        }
        return false;
}
```

三．DOM 操作内容

虽然在之前我们已经学习了各种 DOM 操作的方法，这里所介绍是 innerText、innerHTML、outerText 和 outerHTML 等属性。除了之前用过的 innerHTML 之外，其他三个还么有涉及到。

1.innerText 属性

```
document.getElementById('box').innerText;        //获取文本内容(如有 html 直接过滤掉)
document.getElementById('box').innerText = 'Mr.Lee';       //设置文本(如有 html 转义)
```

PS：除了 Firefox 之外，其他浏览器均支持这个方法。但 Firefox 的 DOM3 级提供了另外一个类似的属性：textContent，做上兼容即可通用。

```
document.getElementById('box').textContent;     //Firefox 支持

//兼容方案
function getInnerText(element) {
    return (typeof element.textContent == 'string') ?
                        element.textContent : element.innerText;
}

function setInnerText(element, text) {
    if (typeof element.textContent == 'string') {
```

```
            element.textContent = text;
        } else {
            element.innerText = text;
        }
    }
```

2.innerHTML 属性

这个属性之前就已经研究过，不拒绝 HTML。

```
document.getElementById('box').innerHTML;    //获取文本(不过滤 HTML)
document.getElementById('box').innerHTML = '<b>123</b>';  //可解析 HTML
```

虽然 innerHTML 可以插入 HTML，但本身还是有一定的限制，也就是所谓的作用域元素，离开这个作用域就无效了。

```
box.innerHTML = "<script>alert('Lee');</script>";     //<script>元素不能被执行
box.innerHTML = "<style>background:red;</style>"; //<style>元素不能被执行
```

3.outerText

outerText 在取值的时候和 innerText 一样，同时火狐不支持，而赋值方法相当危险，他不单替换了文本内容，还将元素直接抹去。

```
var box = document.getElementById('box');
box.outerText = '<b>123</b>';
alert(document.getElementById('box'));            //null，建议不去使用
```

4.outerHTML

outerHTML 属性在取值和 innerHTML 一致，但和 outerText 也一样，很危险，赋值的之后会将元素抹去。

```
var box = document.getElementById('box');
box.outerHTML = '123';
alert(document.getElementById('box'));            //null，建议不去使用，火狐旧版未抹去
```

PS：关于最常用的 innerHTML 属性和节点操作方法的比较，在插入大量 HTML 标记时使用 innerHTML 的效率明显要高很多。因为在设置 innerHTML 时，会创建一个 HTML 解析器。这个解析器是浏览器级别的(C++编写)，因此执行 JavaScript 会快的多。但，创建和销毁 HTML 解析器也会带来性能损失。最好控制在最合理的范围内，如下：

```
for (var i = 0; i < 10; i ++) {
    ul.innerHTML = '<li>item</li>';            //避免频繁
}
//改
for (var i = 0; i < 10; i ++) {
    a = '<li>item</li>';                       //临时保存
}
ul.innerHTML = a;
```

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