什么是 DOM？

DOM 是基于树的API

DOM 是 W3C（万维网联盟）的标准。

DOM 定义了访问 HTML 和 XML 文档的标准：

“W3C 文档对象模型 （DOM） 是中立于平台和语言的接口，它允许程序和脚本动态地访问和更新文档的内容、结构和样式。”

W3C DOM 标准被分为 3 个不同的部分：

• 核心 DOM - 针对任何结构化文档的标准模型

• XML DOM - 针对 XML 文档的标准模型

• HTML DOM - 针对 HTML 文档的标准模型

编者注：DOM 是 Document Object Model（文档对象模型）的缩写。

什么是 XML DOM？

XML DOM 定义了所有 XML 元素的对象和属性，以及访问它们的方法。

如果您需要学习 XML DOM，请访问我们的 XML DOM 教程。

什么是 HTML DOM？

HTML DOM 是：

• HTML 的标准对象模型

• HTML 的标准编程接口

• W3C 标准

HTML DOM 定义了所有 HTML 元素的对象和属性，以及访问它们的方法。

换言之，HTML DOM 是关于如何获取、修改、添加或删除 HTML 元素的标准。

DOM 节点

根据 W3C 的 HTML DOM 标准，HTML 文档中的所有内容都是节点：

• 整个文档是一个文档节点

• 每个 HTML 元素是元素节点

• HTML 元素内的文本是文本节点

• 每个 HTML 属性是属性节点

• 注释是注释节点

HTML DOM 节点树

HTML DOM 将 HTML 文档视作树结构。这种结构被称为节点树：

HTML DOM Tree 实例

通过 HTML DOM，树中的所有节点均可通过 JavaScript 进行访问。所有 HTML 元素（节点）均可被修改，也可以创建或删除节点。

节点父、子和同胞

节点树中的节点彼此拥有层级关系。

父（parent）、子（child）和同胞（sibling）等术语用于描述这些关系。父节点拥有子节点。同级的子节点被称为同胞（兄弟或姐妹）。

• 在节点树中，顶端节点被称为根（root）

• 每个节点都有父节点、除了根（它没有父节点）

• 一个节点可拥有任意数量的子

• 同胞是拥有相同父节点的节点

下面的图片展示了节点树的一部分，以及节点之间的关系：

警告！

DOM 处理中的常见错误是希望元素节点包含文本。

在本例中：

<title>DOM 教程</title>，元素节点 <title>，包含值为 "DOM 教程" 的文本节点。

可通过节点的 innerHTML 属性来访问文本节点的值。

您将在稍后的章节中学习更多有关 innerHTML 属性的知识。

HTML DOM 方法

方法是我们可以在节点（HTML 元素）上执行的动作。

编程接口

可通过 JavaScript （以及其他编程语言）对 HTML DOM 进行访问。

所有 HTML 元素被定义为对象，而编程接口则是对象方法和对象属性。

方法是您能够执行的动作（比如添加或修改元素）。

属性是您能够获取或设置的值（比如节点的名称或内容）。

getElementById() 方法

getElementById() 方法返回带有指定 ID 的元素：

HTML DOM 对象 - 方法和属性

一些常用的 HTML DOM 方法：

• getElementById(id) - 获取带有指定 id 的节点（元素）

• appendChild(node) - 插入新的子节点（元素）

• removeChild(node) - 删除子节点（元素）

一些常用的 HTML DOM 属性：

• innerHTML - 节点（元素）的文本值

• parentNode - 节点（元素）的父节点

• childNodes - 节点（元素）的子节点

• attributes - 节点（元素）的属性节点

您将在本教程的下一章中学到更多有关属性的知识。

一些 DOM 对象方法

这里提供一些您将在本教程中学到的常用方法：

方法 描述

getElementById() 返回带有指定 ID 的元素。

getElementsByTagName() 返回包含带有指定标签名称的所有元素的节点列表（集合/节点数组）。

getElementsByClassName() 返回包含带有指定类名的所有元素的节点列表。

appendChild() 把新的子节点添加到指定节点。

removeChild() 删除子节点。

replaceChild() 替换子节点。

insertBefore() 在指定的子节点前面插入新的子节点。

createAttribute() 创建属性节点。

createElement() 创建元素节点。

createTextNode() 创建文本节点。

getAttribute() 返回指定的属性值。

setAttribute() 把指定属性设置或修改为指定的值。

HTML DOM 属性

编程接口

可通过 JavaScript （以及其他编程语言）对 HTML DOM 进行访问。

所有 HTML 元素被定义为对象，而编程接口则是对象方法和对象属性。

方法是您能够执行的动作（比如添加或修改元素）。

属性是您能够获取或设置的值（比如节点的名称或内容）。

innerHTML 属性

获取元素内容的最简单方法是使用 innerHTML 属性。

innerHTML 属性对于获取或替换 HTML 元素的内容很有用。

实例

下面的代码获取 id="intro" 的 <p> 元素的 innerHTML：

实例

<html>

<body>

<p id="intro">Hello World!</p>

<script>

var txt=document.getElementById("intro").innerHTML;

document.write(txt);

</script>

</body>

</html>

nodeValue 属性

nodeValue 属性规定节点的值。

• 元素节点的 nodeValue 是 undefined 或 null

• 文本节点的 nodeValue 是文本本身

• 属性节点的 nodeValue 是属性值

获取元素的值

下面的例子会取回 <p id="intro"> 标签的文本节点值：

实例

<html>

<body>

<p id="intro">Hello World!</p>

<script type="text/javascript">

x=document.getElementById("intro");

document.write(x.firstChild.nodeValue);

</script>

</body>

</html>

nodeType 属性

nodeType 属性返回节点的类型。nodeType 是只读的。

比较重要的节点类型有：

元素类型 NodeType

元素 1

属性 2

文本 3

注释 8

文档 9

HTML DOM 访问

访问 HTML 元素（节点）

访问 HTML 元素等同于访问节点

您能够以不同的方式来访问 HTML 元素：

• 通过使用 getElementById() 方法

• getElementsByTagName() 返回带有指定标签名的所有元素。

getElementsByClassName() 方法

如果您希望查找带有相同类名的所有 HTML 元素，请使用这个方法：

document.getElementsByClassName("intro");

上面的例子返回包含 class="intro" 的所有元素的一个列表：

注释：getElementsByClassName() 在 Internet Explorer 5,6,7,8 中无效。

HTML DOM - 修改

修改 HTML 元素

修改 HTML DOM 意味着许多不同的方面：

• 改变 HTML 内容

• 改变 CSS 样式

• 改变 HTML 属性

• 创建新的 HTML 元素

• 删除已有的 HTML 元素

• 改变事件（处理程序）

创建 HTML 内容

改变元素内容的最简答的方法是使用 innerHTML 属性。

下面的例子改变一个 <p> 元素的 HTML 内容：

实例

<html>

<body>

<p id="p1">Hello World!</p>

<script>

document.getElementById("p1").innerHTML="New text!";

</script>

</body>

</html>

改变 HTML 样式

通过 HTML DOM，您能够访问 HTML 元素的样式对象。

下面的例子改变一个段落的 HTML 样式：

实例

<html>

<body>

<p id="p2">Hello world!</p>

<script>

document.getElementById("p2").style.color="blue";

</script>

</body>

</html>

创建新的 HTML 元素

如需向 HTML DOM 添加新元素，您首先必须创建该元素（元素节点），然后把它追加到已有的元素上。

实例

<div id="d1">

<p id="p1">This is a paragraph.</p>

<p id="p2">This is another paragraph.</p>

</div>

<script>

var para=document.createElement("p");

var node=document.createTextNode("This is new.");

para.appendChild(node);

var element=document.getElementById("d1");

element.appendChild(para);

</script>

HTML DOM - 修改 HTML 内容

改变 HTML 内容

改变元素内容的最简单的方法是使用 innerHTML 属性。

下面的例子更改 <p> 元素的 HTML 内容：

实例

<html>

<body>

<p id="p1">Hello World!</p>

<script>

document.getElementById("p1").innerHTML="New text!";

</script>

</body>

</html>

改变 HTML 样式

通过 HTML DOM，您能够访问 HTML 对象的样式对象。

下面的例子更改段落的 HTML 样式：

实例

<html>

<body>

<p id="p2">Hello world!</p>

<script>

document.getElementById("p2").style.color="blue";

</script>

</body>

</html>

使用事件

HTML DOM 允许您在事件发生时执行代码。

当 HTML 元素”有事情发生“时，浏览器就会生成事件：

• 在元素上点击

• 加载页面

• 改变输入字段

你可以在下一章学习更多有关事件的内容。

下面两个例子在按钮被点击时改变 <body> 元素的背景色：

实例

<html>

<body>

<input type="button" onclick="document.body.style.backgroundColor='lavender';"

value="Change background color" />

</body>

</html>

添加、删除和替换 HTML 元素。

创建新的 HTML 元素 - appendChild()

如需向 HTML DOM 添加新元素，您首先必须创建该元素，然后把它追加到已有的元素上。

实例

<div id="div1">

<p id="p1">This is a paragraph.</p>

<p id="p2">This is another paragraph.</p>

</div>

<script>

var para=document.createElement("p");

var node=document.createTextNode("This is new.");

para.appendChild(node);

var element=document.getElementById("div1");

element.appendChild(para);

</script>

创建新的 HTML 元素 - insertBefore()

上一个例子中的 appendChild() 方法，将新元素作为父元素的最后一个子元素进行添加。

如果不希望如此，您可以使用 insertBefore() 方法：

实例

<div id="div1">

<p id="p1">This is a paragraph.</p>

<p id="p2">This is another paragraph.</p>

</div>

<script>

var para=document.createElement("p");

var node=document.createTextNode("This is new.");

para.appendChild(node);

var element=document.getElementById("div1");

var child=document.getElementById("p1");

element.insertBefore(para,child);

</script>

替换 HTML 元素

如需替换 HTML DOM 中的元素，请使用 replaceChild() 方法：

实例

<div id="div1">

<p id="p1">This is a paragraph.</p>

<p id="p2">This is another paragraph.</p>

</div>

<script>

var para=document.createElement("p");

var node=document.createTextNode("This is new.");

para.appendChild(node);

var parent=document.getElementById("div1");

var child=document.getElementById("p1");

parent.replaceChild(para,child);

</script>

HTML DOM - 事件

onafterprint文档打印之后运行的脚本。

onbeforeprint文档打印之前运行的脚本。

onload页面结束加载之后触发。

onresize当浏览器窗口被调整大小时触发。

onunload一旦页面已下载时触发（或者浏览器窗口已被关闭）。

Form 事件

由 HTML 表单内的动作触发的事件（应用到几乎所有 HTML 元素，但最常用在 form 元素中）：

onblur元素失去焦点时运行的脚本。

onchange在元素值被改变时运行的脚本。

onfocus当元素获得焦点时运行的脚本。

onselect在元素中文本被选中后触发。

onsubmit在提交表单时触发。

Keyboard 事件(键盘事件)

onkeydown在用户按下按键时触发。

onkeypress在用户敲击按钮时触发。

onkeyup当用户释放按键时触发。

Mouse 事件 由鼠标或类似用户动作触发的事件：

Onclick 元素上发生鼠标点击时触发。

ondblclick元素上发生鼠标双击时触发。

onmousedown当元素上按下鼠标按钮时触发。

onmousemove当鼠标指针移动到元素上时触发。

onmouseout当鼠标指针移出元素时触发。

onmouseover当鼠标指针移动到元素上时触发。

onmouseup当在元素上释放鼠标按钮时触发。

onscroll 当元素滚动条被滚动时运行的脚本。

Media 事件

由媒介（比如视频、图像和音频）触发的事件（适用于所有 HTML 元素，但常见于媒介元素中，比如 <audio>、<embed>、<img>、<object> 以及 <video>）:

更多media事件 连接http://www.w3school.com.cn/tags/html\_ref\_eventattributes.asp