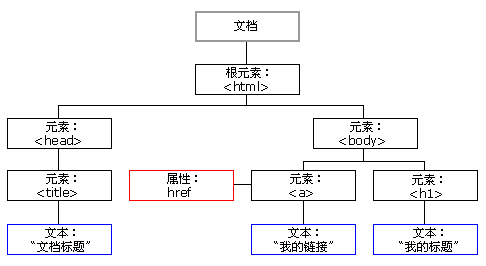
HTML DOM学习

# 简介

HTML DOM 定义了访问和操作 HTML 文档的标准方法。

DOM 将 HTML 文档表达为树结构。

## HTML DOM 树



## 什么是 DOM？

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

**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**。

## 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。**

# HTML DOM 节点

**在 HTML DOM 中，所有事物都是节点。DOM 是被视为节点树的 HTML。**

## DOM 节点

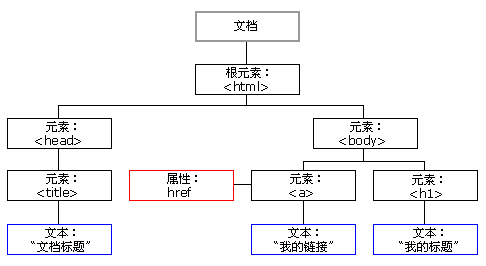
根据 W3C 的 HTML DOM 标准，HTML 文档中的所有内容都是节点：

* 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。

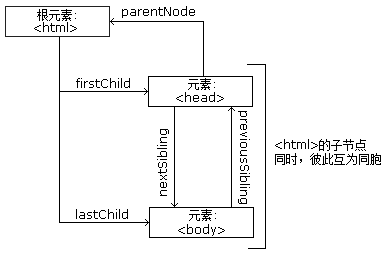
## 节点父、子和同胞

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：



## 请看下面的 HTML 片段：

<html>

<head>

<title>DOM 教程</title>

</head>

<body>

<h1>DOM 第一课</h1>

<p>Hello world!</p>

</body>

</html>

从上面的 HTML 中：

* <html> 节点没有父节点；它是根节点
* <head> 和 <body> 的父节点是 <html> 节点
* 文本节点 "Hello world!" 的父节点是 <p> 节点

并且：

* <html> 节点拥有两个子节点：<head> 和 <body>
* <head> 节点拥有一个子节点：<title> 节点
* <title> 节点也拥有一个子节点：文本节点 "DOM 教程"
* <h1> 和 <p> 节点是同胞节点，同时也是 <body> 的子节点

并且：

* <head> 元素是 <html> 元素的首个子节点
* <body> 元素是 <html> 元素的最后一个子节点
* <h1> 元素是 <body> 元素的首个子节点
* <p> 元素是 <body> 元素的最后一个子节点

## 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。

# 3.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：

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

## 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

## 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。 |

# 4.HTML DOM 属性

属性是节点（HTML 元素）的值，您能够获取或设置。

## 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。

## nodeName 属性

nodeName 属性规定节点的名称。

* nodeName 是只读的
* 元素节点的 nodeName 与标签名相同
* 属性节点的 nodeName 与属性名相同
* 文本节点的 nodeName 始终是 #text
* 文档节点的 nodeName 始终是 #document

**注释：**nodeName 始终包含 HTML 元素的大写字母标签名。

## 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

## 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 |

# 5.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
* 通过使用 getElementsByTagName() 方法
* 通过使用 getElementsByClassName() 方法

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

# 6.HTML DOM - 修改

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>

## 改变 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>

## 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>

在本例中，由函数执行相同的代码：

### 实例

<html>

<body>

<script>

function ChangeBackground()

{

document.body.style.backgroundColor="lavender";

}

</script>

<input type="button" onclick="ChangeBackground()"

value="Change background color" />

</body>

</html>

下面的例子在按钮被点击时改变 <p> 元素的文本：

### 实例

<html>

<body>

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

<script>

function ChangeText()

{

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

}

</script>

<input type="button" onclick="ChangeText()" value="Change text">

</body>

</html>

## 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>

## 删除已有的 HTML 元素

如需删除 HTML 元素，您必须清楚该元素的父元素：

### 实例

<div id="div1">

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

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

</div>

<script>

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

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

parent.removeChild(child);

</script>

**提示：**能否在不引用父元素的情况下删除某个元素？

很抱歉。DOM 需要了解您需要删除的元素，以及它的父元素。

这里提供一个常用的解决方法：找到您需要删除的子元素，然后使用 parentNode 属性来查找其父元素：

## 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>

# 7.HTML DOM - 事件

HTML DOM 允许 JavaScript 对 HTML 事件作出反应。

## 对事件作出反应

当事件发生时，可以执行 JavaScript，比如当用户点击一个 HTML 元素时。

如需在用户点击某个元素时执行代码，请把 JavaScript 代码添加到 HTML 事件属性中：

onclick=*JavaScript*

HTML 事件的例子：

* 当用户点击鼠标时
* 当网页已加载时
* 当图片已加载时
* 当鼠标移动到元素上时
* 当输入字段被改变时
* 当 HTML 表单被提交时
* 当用户触发按键时

# 8.HTML DOM - 导航

通过 HTML DOM，您能够使用节点关系在节点树中导航。

## HTML DOM 节点列表

getElementsByTagName() 方法返回**节点列表**。节点列表是一个节点数组。

下面的代码选取文档中的所有 <p> 节点：

### 实例

var x=document.getElementsByTagName("p");

可以通过下标号访问这些节点。如需访问第二个 <p>，您可以这么写：

y=x[1];

**注释：**下标号从 0 开始。

## HTML DOM 节点列表长度

length 属性定义节点列表中节点的数量。

您可以使用 length 属性来循环节点列表：

### 实例

x=document.getElementsByTagName("p");

for (i=0;i<x.length;i++)

{

document.write(x[i].innerHTML);

document.write("<br />");

}

## 导航节点关系

您能够使用三个节点属性：parentNode、firstChild 以及 lastChild ，在文档结构中进行导航。

## DOM 根节点

这里有两个特殊的属性，可以访问全部文档：

* document.documentElement - 全部文档
* document.body - 文档的主体

### 实例

<html>

<body>

<p>Hello World!</p>

<div>

<p>DOM 很有用!</p>

<p>本例演示 <b>document.body</b> 属性。</p>

</div>

<script>

alert(document.body.innerHTML);

</script>

</body>

</html>

## childNodes 和 nodeValue

除了 innerHTML 属性，您也可以使用 childNodes 和 nodeValue 属性来获取元素的内容。

下面的代码获取 id="intro" 的 <p> 元素的值：

### 实例

<html>

<body>

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

<script>

var txt=document.getElementById("intro").childNodes[0].nodeValue;

document.write(txt);

</script>

</body>

</html>

在上面的例子中，getElementById 是一个方法，而 childNodes 和 nodeValue 是属性。

# 9.HTML DOM 实例

## Event 对象

### [哪个鼠标按钮被点击？](http://www.w3school.com.cn/tiy/t.asp?f=hdom_event_button)

<html>

<head>

<script type="text/javascript">

function whichButton(event)

{

var btnNum = event.button;

if (btnNum==2)

{

alert("您点击了鼠标右键！")

}

else if(btnNum==0)

{

alert("您点击了鼠标左键！")

}

else if(btnNum==1)

{

alert("您点击了鼠标中键！");

}

else

{

alert("您点击了" + btnNum+ "号键，我不能确定它的名称。");

}

}

</script>

</head>

<body onmousedown="whichButton(event)">

<p>请在文档中点击鼠标。一个消息框会提示出您点击了哪个鼠标按键。</p>

</body>

</html>

### [光标的坐标是？](http://www.w3school.com.cn/tiy/t.asp?f=hdom_event_clientx)

<html>

<head>

<script type="text/javascript">

function show\_coords(event)

{

x=event.clientX

y=event.clientY

alert("X 坐标: " + x + ", Y 坐标: " + y)

}

</script>

</head>

<body onmousedown="show\_coords(event)">

<p>请在文档中点击。一个消息框会提示出鼠标指针的 x 和 y 坐标。</p>

</body>

</html>

### [被按的按键的 unicode 是？](http://www.w3school.com.cn/tiy/t.asp?f=hdom_event_keycode)

<html>

<head>

<script type="text/javascript">

function whichButton(event)

{

alert(event.keyCode)

}

</script>

</head>

<body onkeyup="whichButton(event)">

<p><b>注释：</b>在测试这个例子时，要确保右侧的框架获得了焦点。</p>

<p>在键盘上按一个键。消息框会提示出该按键的 unicode。</p>

</body>

</html>

### [相对于屏幕，光标的坐标是？](http://www.w3school.com.cn/tiy/t.asp?f=hdom_event_screenxy)

<html>

<head>

<script type="text/javascript">

function coordinates(event)

{

x=event.screenX

y=event.screenY

alert("X=" + x + " Y=" + y)

}

</script>

</head>

<body onmousedown="coordinates(event)">

<p>

在文档中点击某个位置。消息框会提示出指针相对于屏幕的 x 和 y 坐标。

</p>

</body>

</html>

### [shift 键被按了吗？](http://www.w3school.com.cn/tiy/t.asp?f=hdom_event_shiftkey)

<html>

<head>

<script type="text/javascript">

function isKeyPressed(event)

{

if (event.shiftKey==1)

{

alert("The shift key was pressed!")

}

else

{

alert("The shift key was NOT pressed!")

}

}

</script>

</head>

<body onmousedown="isKeyPressed(event)">

<p>在文档中点击某个位置。消息框会告诉你是否按下了 shift 键。</p>

</body>

</html>

### [哪个元素被点击了？](http://www.w3school.com.cn/tiy/t.asp?f=hdom_event_srcelement)

<html>

<head>

<script type="text/javascript">

function whichElement(e)

{

var targ

if (!e) var e = window.event

if (e.target) targ = e.target

else if (e.srcElement) targ = e.srcElement

if (targ.nodeType == 3) // defeat Safari bug

targ = targ.parentNode

var tname

tname=targ.tagName

alert("You clicked on a " + tname + " element.")

}

</script>

</head>

<body onmousedown="whichElement(event)">

<p>在文档中点击某个位置。消息框会提示出您所点击的标签的名称。</p>

<h3>这是标题</h3>

<p>这是段落。</p>

<img src="/i/eg\_mouse2.jpg" />

</body>

</html>

### [哪个事件类型发生了？](http://www.w3school.com.cn/tiy/t.asp?f=hdom_event_type)

<html>

<head>

<script type="text/javascript">

function getEventType(event)

{

alert(event.type);

}

</script>

</head>

<body onmousedown="getEventType(event)">

<p>在文档中点击某个位置。消息框会提示出被触发的事件的类型。</p>

</body>

</html>

其他实例请访问：http://www.w3school.com.cn/example/hdom\_examples.asp