## DOM：Document Object Model （文本对象模型）

D:文档---------html文档或xml文档

O:对象---------document对象的属性和方法

M:模型

DOM是针对xml(html)的基于树的API

DOM树：节点（node）的层次

DOM把一个文档表示为一棵家谱树（父，子，兄弟）

DOM定义了Node的接口以及许多种节点类型来表示XML节点的多个方面

### 节点及其类型

1. 元素节点
2. 属性节点：元素的属性，可以直接通过属性的方式来操作。
3. 文本节点：是元素节点的子节点，其内容为文本。

### 在HTML文档的什么位置编写js代码？

1. 直接在html页面中书写代码。
   1. 缺点：
      1. js和html强耦合，不利于代码的维护。
      2. 若click相应函数比较复杂的，则需要先定义一个函数，然后再在onclick属性中完成对函数的引用，比较麻烦。
2. 一般地，不能在body节点之前来直接获取body内的节点，因为此时html文档树还没有加载完成，获取不到指定的节点。
3. 可以在整个html文档的最后编写类似代码，但这不符合习惯。
4. 一般地，在body节点之前编写js代码，但需要利用wiodow.onload事件，该事件在当前文档完全加载之后被触发，所以其中的代码可以获取到当前文档的任何节点。

### 如何来获取元素节点：

1. document.getElementById: 根据id属性获取对应的单个节点。
2. document.getElementsByTagName: 根据标签名获取指定节点名字的数组，数组对象 length属性可以获取数组的长度。
3. document.getElementsByName：根据节点的name属性获取符合条件的节点素组。

### 获取属性节点：

1. 可以直接通过cityNode.id这样的方式来获取和设置属性节点的值。
2. 通过元素节点的getAttributeNode方法来获取属性节点，然后在通过nodeValue来读写属性值。

### 获取元素节点的子节点（只有元素节点才有子节点）：

1. childNodes属性获取全部的子节点,但该方法不实用，因为如果要获取指定的节点的指定子节点的集合，可以直接调用元素节点的getElementsByTagName()方法来获取。
2. firstChild属性获取第一个子节点。
3. lastChild属性获取最后一个子节点。

### 获取文本节点:

1. 步骤：元素节点 🡪获取元素节点的子节点。
2. 若元素节点只有文本节点一个子节点，例如<li id="bj" name="BeiJing">北京</li>，

可以先获取到指定的原树节点eleNode，然后利用eleNode.firstChild.nodeValue的方法来读写其文本节点的值。

### 节点的属性（按元素节点，属性节点，文本节点来说明）:

1)nodeType: 1 , 2 , 3;只读属性

2)nodeName: 返回对应的节点的名字，只读属性。

3)nodeValue: null , 属性值 ，文本值 ，可读写的属性。

### 创建一个元素节点：

1. createElement():按照给定的标签名创建一个新的元素节点。方法只有一个参数。被创建的元素节点的名字，是一个字符串。

方法返回值：是一个指向新建节点的引用指针，返回值是一个元素节点，所以它的nodeType属性值等于1.

\*\* 新元素节点不会自动添加到文档里，它只是一个存在于JavaScript上下文的对象。

### 创建一个文本节点：

1. createTextNode():创建一个包含着给定文本的新文本节点，这个方法的返回值是一个指向新文本节点。

方法只有一个参数：新建文本节点所包含的文本字符串，新元素节点不会自动添加到文档里。

### 为元素节点添加子节点：

1. appendChild():

var reference = element.appendChild(newChild):

给定子节点newChild将成为给定元素节点element的最后一个子节点。

方法的返回值是一个指向新增子节点的引用指针。

### 节点的替换：

1)replaceChild():把一个给定父元素里的一个子节点替换为另外一个子节点。

var reference = element.replaceChild(newChild,oldChild);0

2)该节点除了替换功能以外还有移动的功能

3)该方法只能完成单向替换，若需要使用双向替换，需要自定义函数。

/\*

**function** replaceEle(aNode,bNode){

//获取aNode,bNode的父节点，使用parentNode属性

**var** aParent = aNode.parentNode;

**var** bParent = bNode.parentNode;

**if**(aParent && bParent){

//2.克隆aNode或bNode

**var** aNode1 = aNode.cloneNode(**true**);

//3.分别调用aNode的父节点和bNode的父节点的eplaceChild()

//方法实现节点的互换.

bParent.replaceChild(aNode1,bNode);

aParent.replaceChild(bNode,aNode);

}

}

\*/

### 12.插入节点

1). insertBefore(): 把一个给定节点插入到一个给定元素节点的给定子节点的前面

var reference = element.insertBefore(newNode,targetNode);

节点 newNode 将被插入到元素节点 element 中并出现在节点 targetNode 的前面. 节点 targetNode 必须是 element 元素的一个子节点。

### 13.删除节点：

1). removeChild(): 从一个给定元素里删除一个子节点

var reference = element.removeChild(node)

返回值是一个指向已被删除的子节点的引用指针。某个节点被removeChild()方法删除是，这个节点所包含的所有子节点将同时被删除，如果想删除某个节点，但不知道它的父节点是哪一个,parentNode属性可以帮忙。

### 14. innerHTML属性:

1). 浏览器几乎都支持该属性, 但不是 DOM 标准的组成部分. innerHTML 属性可以用来读, 写某给定元素里的 HTML 内容

## 原版

1. 节点及其类型:

1). 元素节点

2). 属性节点: 元素的属性, 可以直接通过属性的方式来操作.

3). 文本节点: 是元素节点的子节点, 其内容为文本.

2. 在 html 文档的什么位置编写 js 代码?

0). 直接在 html 页面中书写代码.

<button id="button" onclick="alert('hello world');">Click Me!</button>

缺点:

①. js 和 html 强耦合, 不利用代码的维护

②. 若 click 相应函数是比较复杂的, 则需要先定义一个函数, 然后再在 onclick 属性中完成对函数的引用, 比较麻烦

1). 一般地, 不能在 body 节点之前来直接获取 body 内的节点, 因为此时 html 文档树还没有加载完成,

获取不到指定的节点:

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Untitled Document</title>

<script type="text/javascript">

var cityNode = document.getElementById("city");

//打印结果为 null.

alert(cityNode);

</script>

</head>

<body>

......

2). 可以在整个 html 文档的最后编写类似代码, 但这不符合习惯

3). 一般地, 在 body 节点之前编写 js 代码, 但需要利用 window.onload 事件,

该事件在当前文档完全加载之后被触发, 所以其中的代码可以获取到当前文档的任何节点.

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Untitled Document</title>

<script type="text/javascript">

window.onload = function(){

var cityNode = document.getElementById("city");

alert(cityNode);

};

</script>

</head>

<body>

......

3. 如何来获取元素节点:

1). \*\*document.getElementById: 根据 id 属性获取对应的单个节点

2). \*\*document.getElementsByTagName:

根据标签名获取指定节点名字的数组, 数组对象 length 属性可以获取数组的长度

3). document.getElementsByName:

根据节点的 name 属性获取符合条件的节点数组,

但 ie 的实现方式和 W3C 标准有差别:

在 html 文档中若某节点(li)没有 name 属性,

则 ie 使用 getElementsByName 不能获取到节点数组, 但火狐可以.

4). 其它的两个方法,　ie 根本就不支持, 所以不建议使用

4. 获取属性节点:

1). \*\*可以直接通过 cityNode.id 这样的方式来获取和设置属性节点的值

2). 通过元素节点的 getAttributeNode 方法来获取属性节点,

然后在通过 nodeValue 来读写属性值

5. 获取元素节点的子节点(\*\*只有元素节点才有子节点!!):

1). childNodes 属性获取全部的子节点, 但该方法不实用. 因为如果要获取指定的节点

的指定子节点的集合, 可以直接调用元素节点的 getElementsByTagName() 方法来获取.

2). firstChild 属性获取第一个子节点

3). lastChild 属性获取最后一个子节点

6. 获取文本节点:

1). 步骤: 元素节点 --> 获取元素节点的子节点

2). 若元素节点只有文本节点一个子节点,

例如 <li id="bj" name="BeiJing">北京</li>, <p>你喜欢哪个城市?</p>,

可以先获取到指定的元素节点 eleNode,

然后利用 eleNode.firstChild.nodeValue 的方法来读写其文本节点的值

7. 节点的属性:

1). nodeName: 代表当前节点的名字. 只读属性.

\*\*如果给定节点是一个文本节点, nodeName 属性将返回内容为 #text 的字符串

2). nodeType：返回一个整数, 这个数值代表着给定节点的类型.

只读属性. 1 -- 元素节点, 2 -- 属性节点, 3 -- 文本节点

\*\*3). nodeValue：返回给定节点的当前值(字符串). 可读写的属性

①. 元素节点, 返回值是 null.

②. 属性节点: 返回值是这个属性的值

③. 文本节点: 返回值是这个文本节点的内容

8. 创建一个元素节点:

1). createElement(): 按照给定的标签名创建一个新的元素节点. 方法只有一个参数：被创建的元素节点的名字, 是一个字符串.

方法的返回值：是一个指向新建节点的引用指针. 返回值是一个元素节点, 所以它的 nodeType 属性值等于 1.

\*\*新元素节点不会自动添加到文档里, 它只是一个存在于 JavaScript 上下文的对象.

9. 创建一个文本节点:

1). createTextNode(): 创建一个包含着给定文本的新文本节点. 这个方法的返回值是一个指向新建文本节点引用指针. 它是一个文本节点, 所以它的 nodeType 属性等于 3.

方法只有一个参数：新建文本节点所包含的文本字符串. 新元素节点不会自动添加到文档里

10. 为元素节点添加子节点:

1). appendChild(): var reference = element.appendChild(newChild): 给定子节点 newChild 将成为给定元素节点 element 的最后一个子节点.

方法的返回值是一个指向新增子节点的引用指针.

11. 节点的替换:

1). replaceChild(): 把一个给定父元素里的一个子节点替换为另外一个子节点

var reference = element.replaceChild(newChild,oldChild);

返回值是一个指向已被替换的那个子节点的引用指针

2). 该节点除了替换功能以外还有移动的功能.

3). 该方法只能完成单向替换, 若需要使用双向替换, 需要自定义函数:

/\*\*

\* 互换 aNode 和 bNode

\* @param {Object} aNode

\* @param {Object} bNode

\*/

function replaceEach(aNode, bNode){

if(aNode == bNode){

return;

}

var aParentNode = aNode.parentNode;

//若 aNode 有父节点

if(aParentNode){

var bParentNode = bNode.parentNode;

//若 bNode 有父节点

if(bParentNode){

var tempNode = aNode.cloneNode(true);

bParentNode.replaceChild(tempNode, bNode);

aParentNode.replaceChild(bNode, aNode);

}

}

}

12. 插入节点:

1). insertBefore(): 把一个给定节点插入到一个给定元素节点的给定子节点的前面

var reference = element.insertBefore(newNode,targetNode);

节点 newNode 将被插入到元素节点 element 中并出现在节点 targetNode 的前面. 节点 targetNode 必须是 element 元素的一个子节点。

2). 自定义 insertAfter() 方法

/\*\*

\* 将 newChild 插入到 refChild 的后边

\* @param {Object} newChild

\* @param {Object} refChild

\*/

function insertAfter(newChild, refChild){

var refParentNode = refChild.parentNode;

//判断 refChild 是否存在父节点

if(refParentNode){

//判断 refChild 节点是否为其父节点的最后一个子节点

if(refChild == refParentNode.lastChild){

refParentNode.appendChild(newChild);

}else{

refParentNode.insertBefore(newChild, refChild.nextSibling);

}

}

}

13. 删除节点:

1). removeChild(): 从一个给定元素里删除一个子节点

var reference = element.removeChild(node);

返回值是一个指向已被删除的子节点的引用指针. 某个节点被 removeChild() 方法删除时, 这个节点所包含的所有子节点将同时被删除.

如果想删除某个节点, 但不知道它的父节点是哪一个, parentNode 属性可以帮忙。

14. innerHTML属性:

1). 浏览器几乎都支持该属性, 但不是 DOM 标准的组成部分. innerHTML 属性可以用来读, 写某给定元素里的 HTML 内容

15. 其它属性, 参看 API: nsextSibling, previousSibling 等