javascript DOM操作

## 节点及其类型

元素节点

属性节点: 元素的属性, 可以直接通过属性的方式来操作.

文本节点: 是元素节点的子节点, 其内容为文本.

|  |
| --- |
| 在文档对象模型 (DOM) 中，每个节点都是一个对象。DOM 节点有三个重要的属性 ：nodeName : 节点的名称；nodeValue ：节点的值；nodeType ：节点的类型  **#1. nodeName 属性: 节点的名称，是只读的。**  1. 元素节点的 nodeName 与标签名相同  2. 属性节点的 nodeName 是属性的名称  3. 文本节点的 nodeName 永远是 #text  4. 文档节点的 nodeName 永远是 #document  **#2. nodeValue 属性：节点的值**  1. 元素节点的 nodeValue 是 undefined 或 null  2. 文本节点的 nodeValue 是文本自身  3. 属性节点的 nodeValue 是属性的值  **#3. nodeType 属性: 节点的类型，是只读的。以下常用的几种结点类型:**  元素类型 节点类型  元素 1  属性 2  文本 3  注释 8  文档 9 |

## 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.  <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  #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:  <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>  ......  #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 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>  ...... |

## 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 |

## 获取和设置属性节点

|  |
| --- |
| #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  #获取属性节点的值。  elementNode.getAttribute(name)  name：要想查询的元素节点的属性名字  #设置属性节点的值  elementNode.setAttribute(name,value)  name: 要设置的属性名。  value: 要设置的属性值。 |

## 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>, <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 |

## 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  #2nodeType：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 |

## 创建一个元素节点

|  |
| --- |
| document.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. |

## 创建一个文本节点

|  |
| --- |
| 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.  注意：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:

|  |
| --- |
| 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);  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);  }  }  } |

## 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);  }  }  } |

## 删除节点

|  |
| --- |
| #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。 |

## 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

其它属性, 参看 API: nsextSibling, previousSibling 等