Javascript DOM 编程

讲师：佟刚

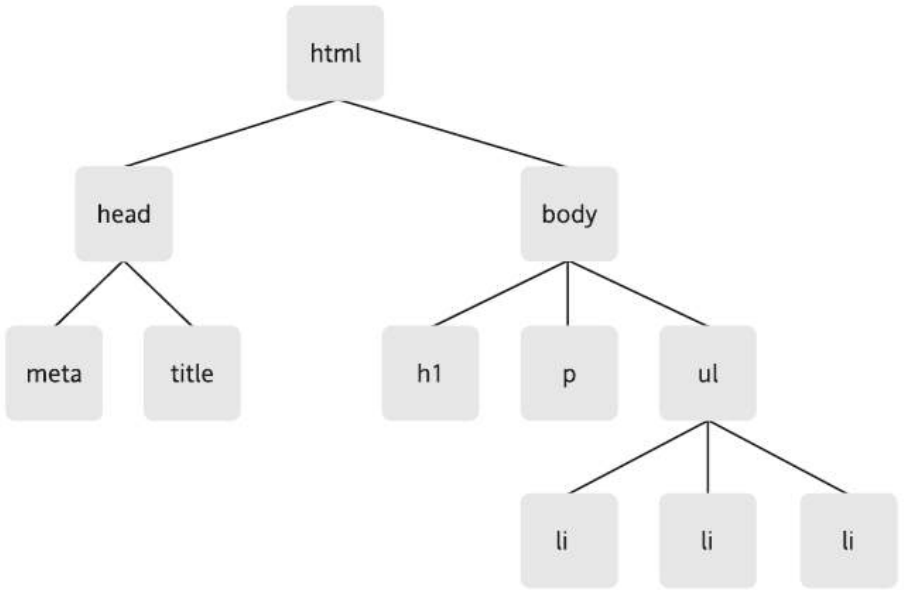
新浪微博：尚硅谷-佟刚

DOM

- 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

DOM树

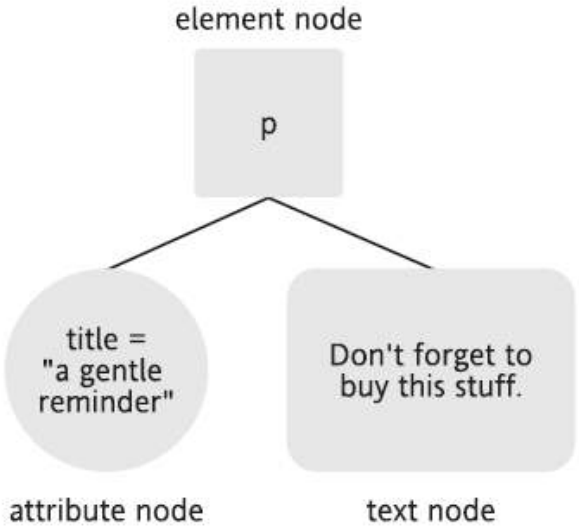
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <meta http-equiv="content-type" content="text/html;
➥ charset=utf-8" />
    <title>Shopping list</title>
  </head>
  <body>
    <h1>What to buy</h1>
    <p title="a gentle reminder">Don't forget to buy this stuff.</p>
    <ul id="purchases">
      <li>A tin of beans</li>
      <li>Cheese</li>
      <li>Milk</li>
    </ul>
  </body>
</html>
```



节点及其类型

- 节点（node）：**来源于网**络理论，代表网络中的**一个**连接点。网络是由节点构成的集合

```
<p title="a gentle reminder">Don’t forget to buy this stuff.</p>
```



Node接口的特性和方法

特性/方法	类型/返回类型	说　明
nodeName	String	节点的名字：根据节点的类型而定义
nodeValue	String	节点的值：根据节点的类型而定义
nodeType	Number	节点的类型常量值之一
ownerDocument	Document	指向这个节点所属的文档
firstChild	Node	指向在childNodes列表中的第一个节点
lastChild	Node	指向在childNodes列表中的最后一个节点
childNodes	NodeList	所有子节点的列表
previousSibling	Node	指向前一个兄弟节点；如果这个节点就是第一个兄弟节点，那么该值为null
nextSibling	Node	指向后一个兄弟节点；如果这个节点就是最后一个兄弟节点，那么该值为null
hasChildNodes()	Boolean	当childNodes包含一个或多个节点时，返回真
attributes	NamedNodeMap	包含了代表一个元素的特性的Attr对象；仅用于Element节点
appendChild(*node*)	Node	将*node*添加到childNodes的末尾
removeChild(*node*)	Node	从childNodes中删除*node*
replaceChild(*newnode*, *oldnode*)	Node	将childNodes中的*oldnode*替换成*newnode*
insertBefore(*newnode*, *refnode*)	Node	在childNodes中的*refnode*之前插入*newnode*

查找元素节点

- **getElementById()**
  - 寻找一个有着给定 id 属性值的元素，返回值是一个有着给定 id 属性值的元素节点。如果不存在这样的元素，它返回 null。

    var element = document.getElementById(ID);
  - **该方法只能用于 document 对象**

查找元素节点

- getElement**s**By**TagName**()
  - 寻找有着给定**标签名**的**所有元素**，这个方法将返回**一个节点集合**，这个集合可以当作一个数组来处理。这个集合的 **length** 属性等于当前文档里有着给定标签名的所有元素的总个数。

```
var elements = element.getElementsByTagName(tagName);
```

  - **该方法不必非得用在整个文档上。它也可以用来在某个特定元素的子节点当中寻找有着给定标签名的元素。**

```
var container =   document.getElementById(“content”);
var paras = container.getElementsByTagName(“p”);
alert(paras.length);
```

查看是否存在子节点

- hasChildNodes()
  - 该方法用来检查一个元素是否有子节点，返回值是 true 或 false.

    var booleanValue = element.hasChildNodes();
  - **文本节点和属性节点不可能再包含任何子节点**，所以对这两类节点使用 hasChildNodes 方法的返回值永远是 false.
  - 如果 hasChildNodes 方法的返回值是 false，则 childNodes,firstChild,lastChild 将是空数组和空字符串。

DOM 属性 -- nodeName

- 文档里的每个节点都有以下属性。
- nodeName:一个字符串，其内容是给定节点的名字。

  var name = node.nodeName;

  - 如果给定节点是一个元素节点或属性节点，nodeName 属性将返回这个元素的名字。
  - **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。**
  - **nodeName 是一个只读属性。**

DOM 属性 -- nodeType

- nodeType：返回一个整数，这个数值代表着给定节点的类型。
  - nodeType 属性返回的整数值对应着 12 种节点类型：
    - Node.ELEMENT_NODE (1)　　-- 元素节点
    - Node.ATTRIBUTE_NODE (2)　-- 属性节点
    - Node.TEXT_NODE (3)　　-- 文本节点
    - …
  - **nodeType 是个只读属性**

DOM 属性 -- nodeValue

- nodeValue：返回给定节点的当前值（字符串）
  - 如果给定节点是一个**属性节点**，返回值是这个**属性的值**。
  - **如果给定节点是一个文本节点，返回值是这个文本节点的内容。**
  - 如果给定节点是一个**元素节点**，返回值是 **null**
  - nodeValue 是一个 **读/写** 属性，但不能对元素节点的 nodeValue 属性设置值，但**可以为文本节点的 nodeValue 属性设置一个值**。

```
var message = document.getElementByld(“fineprint”);
if(message.firstChild.nodeType == 3)
   message.firstChild.nodeValue = “this might work”;
```

替换节点

- 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**。
  - **如果被插入的子**节点还有子节点，则那些子节点也被插入到目标节点中

```
var container = document.getElementById(“content”);
var message = document.getElementById(“fineprint”);
var announcement = document.getElementById(“headline”);
var oldmessage = container.replaceChild(announcement,message);
container.appendChild(oldmessage);
```

查找属性节点

- getAttribute()
  - 返回一个给定元素的一个**给定属性节点的值**

    var attributeValue = element.getAttribute(attributeName);
  - 给定属性的名字必须以字符串的形式传递给该方法。
  - 给定属性的值将以字符串的形式返回，如果给定属性不存在，getAttribute() 将返回一个空字符串.

设置属性节点

- setAttribute()
  - 将给定元素节点**添加一个新的属性值或改变它的现有属性的值**。

    element.setAttribute(attributeName,attributeValue);
  - 属性的名字和值必须以字符串的形式传递给此方法
  - **如果这个属性已经存在，它的值将被刷新；如果不存在，setAttribute()方法将先创建它再为其赋值。**

```
var para = document.createElement(“p”);
para.setAttribute(“id”,”fineprint”);
var container = document.getElementById(“content”);
container.appendChild(para);
```

创建新元素节点

- **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：将被创建的元素的名字，是一个字符串.

    var reference = document.createElement(element);
  - 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**，新节点没有 nodeParent 属性，它只是一个存在于 JavaScript 上下文的对象.

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

创建新文本节点

- **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。

    var reference = document.createTextNode(text);
  - **方法只有一个参数：新建文本**节点所包含的文本字符串
  - **方法的返回**值：是一个指向新建节点的引用指针。它是**一个文本节点，所以它的** nodeType **属性等于** 3.
  - **新元素**节点不会自动添加到文档里，新节点没有 nodeParent 属性

    var oText = document.createTextNode(“Hello World”);

插入节点(1)

- appendChild()
  - 为给定元素增加一个子节点： **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。
  - 该方法**通常与 createElement() createTextNode() 配合使用**

```
var para = document.createElement(“p”);
var message = document.createTextNode(“Hello World”);
para.appendChild(message);
document.body.appendChild(para);
```

  - **新**节点可以被追加给文档中的任何一个元素

插入节点(2)

- 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**。
  - 该方法通常与 createElement() 和 createTextNode() 配合使用

```
var container = document.getElementByld(“content”);
var message = document.getElementByld(“fineprint”);
var para = document.createElement(“p”);
container.insertBefore(para,message);
```

插入节点(3)

- DOM 没有提供 insertAfter() 方法

```
/**
 * @param {Object} newElement: 将被插入的新元素
 * @param {Object} targetElement: 新元素将被插入到它前面的目标元素
 */
function insertAfter(newElement, targetElement){
    //获取目标元素的父节点
    var parent = targetElement.parentNode;
    //如果目标元素是最后一个元素, 则新元素紧跟着插入到目标元素的后面
    if(parent.lastChild == targetElement){
        parent.appendChild(newElement);
    }
    //如果目标元素不是最后一个元素, 则新元素插入到目标元素的下一个兄弟元素
    //的前边, 即目标元素的后边
    else{
        parent.insertBefoe(newElement, targetElement.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。**

    ```
    var container = document.getElementById("content");
    var message = document.getElementById("fineprint");
    container.removeChild(message);
    ```
  - **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。**

    ```
    var message = document.getElementById("fineprint");
    var container = message.parentNode;
    container.removeChild(message);
    ```

遍历节点树

- ChildNodes：返回一个数组，这个数组由给定元素节点的子节点构成：

  var nodeList = node.childNodes;

  - **文本节点和属性节点都不可能再包含任何子节点**，所以它们的 ChildNodes 属性永远会返回一个空数组。
  - 如果想知道某个元素有没有子节点，可以用 hasChildNodes 方法。
  - 如果想知道某个元素有多少个子节点，可以用 childNodes 数组的 length 属性。
  - **childNodes 属性是一个只读属性**。

获取第一个子节点

- firstChild：该属性返回一个给定元素节点的第一个子节点，返回这个节点对象的指针。

  var reference = node.firstChild;

  - 文本节点和属性节点都不可能包含任何子节点，所以它们的 firstChild 属性永远会返回 null。
  - 某个元素的 firstChild 属性等价于这个元素的 childNodes 节点集合中的第一个节点，即：

    var reference = node.ChildNodes[0];

  - firstChild 属性是一个**只读属性**。

获取最后一个子节点

- lastChild：对应 firstChild 的一个属性。
- nextSibling: 返回一个给定节点的下一个子节点。
- parentNode：返回一个给定节点的父节点。
  - parentNode 属性返回的节点永远是一个**元素**节点，因为只有元素节点才有可能包含子节点。
  - **document 节点的没有父节点**。
- previousSibling：返回一个给定节点的上一个子节点

innerHTML属性

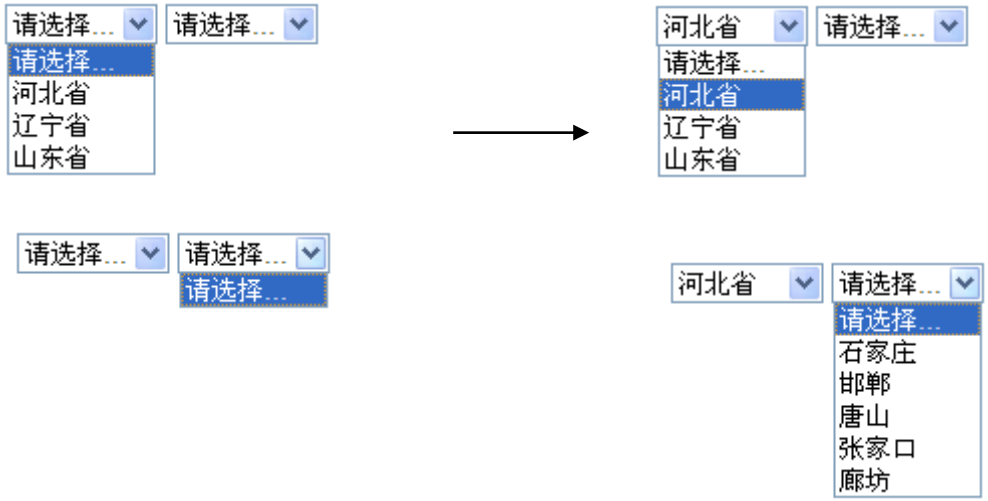
- 浏览器几乎都支持该属性，但不是 DOM 标准的组成部分。
- innerHTML 属性可以用来读，写某给定元素里的 HTML 内容。
- 代码1:

```
var testdiv = document.getElementById("testdiv");
alert(testdiv.innerHTML);
```

- 代码2：

```
var testdiv = document.getElementById("testdiv");
testdiv.innerHTML = "<p>I inserted <em>this</em> content.</p>";
```

练习1



问题分析


```
for(var i = 1; i < citiesOptionElementsLength; i++){
    citiesElement.removeChild(citiesOptionElements[i]);
}
```

i = 1；数组为[“请”,“沈”,“大”,“鞍”,“抚”,“铁”] 删除 “沈阳” 节点

i = 2; 数组为[“请”,“大”,“鞍”,“抚”,“铁”] 删除 “鞍” 节点

i = 3 数组为[“请”,“大”,“抚”,“铁”] 删除 “铁” 节点

i = 4 数组为[“请”,“大”,“抚”] 出错. 数组下标越界

练习2：

添加新员工

name: email: salary:

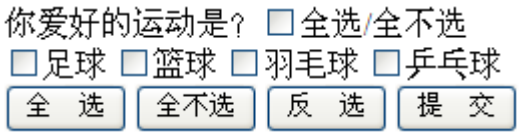
Submit

Name	Email	Salary	
Tom	tom@tom.com	5000	Delete
Jerry	jerry@sohu.com	8000	Delete
Bob	bob@tom.com	10000	Delete

练习3：

employeeName(firstNameLastName):	
employee_id:	
email:	
phone:	
hiredate:	
job_id:	
salary:	
department_id:	

练习4：



当 checkbox 节点有 checked 属性时, 在 HTML 页面上即表现为 "被选中"

利用 setAttribute("checked", "checked"); 实现选中,

removeAttribute("checked"); 实现不被选中

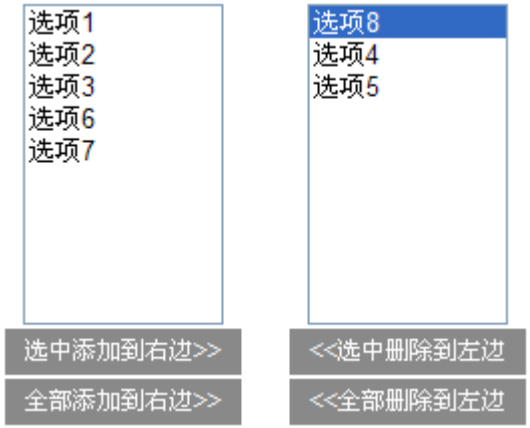
也可以利用

document.getElementById("checkedAll_2").checked = "checked";

实现选中,

利用 document.getElementById("checkedAll_2").checked = null; 实现不被选中

练习5




尚硅谷
www.atguigu.com