AJAX(二)

主要内容

- DOM模型概述
- DOM模型结构
- 处理DOM模型中的结点
- 改变文档的层次结构
- 表格的操作

DOM模型概述

- DOM模型:Document Object Model即文档对象模型,它定义了操作文档对象的接口.DOM模型是一个对象模型.
- DOM模型由3部分组成:分别是核心、Html和Xml。其中核心部分包括了最底层的文档操作接口，适合于html和xml;html部分包括了针对html的操作接口；xml部分定义了针对xml的接口。
- DOM在Ajax中发挥着重要的作用：它是最核心的结构，是所有Ajax开发的基础架构。如果没有DOM模型，就没有办法在客户端改变页面内容，所有的局部刷新、异步请求也就无从实现。熟练掌握DOM模型的相关技术，才算真正掌握了Ajax的开发精髓。

DOM模型结构

- DOM中的文档层次结构：在DOM模型中，整个文档是一棵树的结构，树的根结点是document对象，表示整个文档结点对象，并且仅包含一个子结点<html>.
- 结点的概念：在DOM模型中，整个文档就是由层次不同的多个结点组成的，可以说结点代表了全部的内容，每个结点都可以看成一棵树的根结点，整个结点是一个递归的结构。因此掌握了结点的用法，也就掌握了对DOM模型的用法。
- 结点的类型：在xml（html）中，不仅每个闭合的标记是一个结点，而且闭合标记中的文本，标记内的属性也都是结点，分别为元素结点，文本结点和属性结点。
- 例如：<label for="checkbox1">label1</label>

处理DOM模型中的结点

- 直接引用结点：
  - 使用document.getElementById()引用指定id的结点。
  - 在html中，每个标记都可以有一个id属性。标准中规定：这个id必须是整个文档中唯一的。
  - 例如：<body>
    -

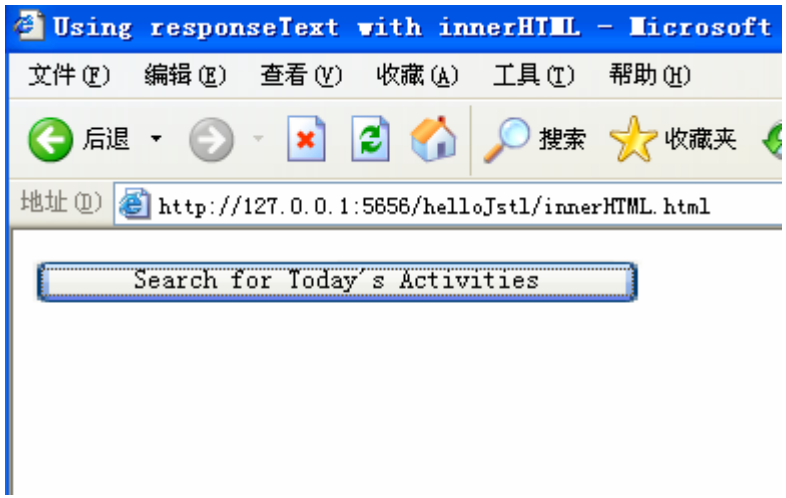
    </body>
  - var div1 = document.getElementById(“div1”);
  - 可以使用innerHTML属性改变结点的内容。
  - div1.innerHTML=“innerHtml changed.”
  - 注意：这种用法不被所有的浏览器所支持，考虑到浏览器的兼容性，应该避免使用这个方法，而应该使用标准的DOM模型方法。

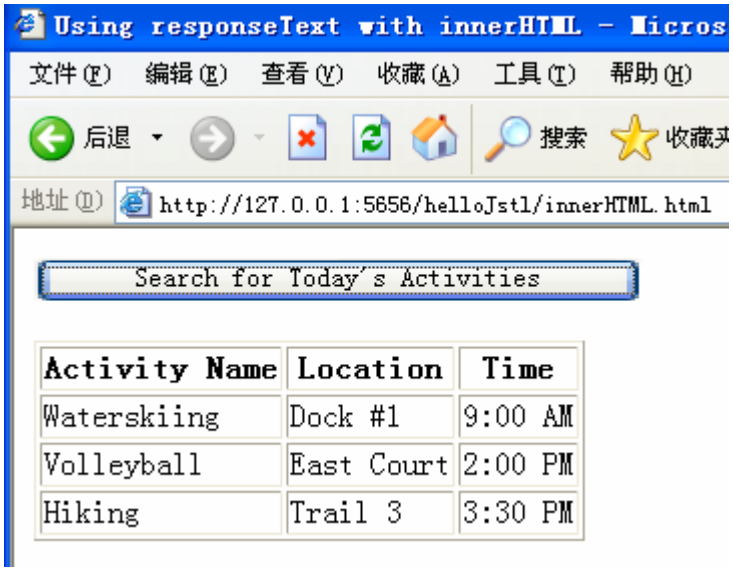
处理DOM模型中的结点

- 直接引用结点：
  - 使用document.getElementsByTagName()引用指定标记名称的结点。
  - 这个方法可以用来获取指定标记的元素结点集合，返回一个数组，包含对这些结点的应用。
  - var div1 = document.getElementsByTagName(“div”);
  - 可以使用innerHTML属性改变结点的内容。
  - div1[0].innerHTML=“innerHtml changed.”

练习：

- 已知服务器端的资源：一个html1.xml文档（table）。
- 在点击按钮后，在div的位置显示一个表格。





间接引用结点

- 引用子结点：每个结点都有一个childNodes集合属性，类型是数组对象，表示该结点的所有子结点的集合。这些子结点按照其在文档中出现的顺序排列，因此可以通过索引依次访问各个字结点。
  - 例如：

    获取html结点用document.childNodes[0];
    获取head和body结点分别用以下两条语句引用：
    document.childNodes[0].childNodes[0]
    document.childNodes[0].childNodes[1]
  - 除了使用childNodes属性，还有两个属性可以用于引用子结点，分别是：

    element.firstChild;//element.childNodes[0]
    element.lastChild;//element.childNodes[element.childNodes.length-1]

间接引用结点

- 引用父结点：Html的结点层次是一个树状的结构，除了根结点外，每个结点都仅有一个父结点，可以由parentNode属性来引用。
- 语法如下：
  - element.parentNode

间接引用结点

- 引用兄弟结点：属于同一个文档层次的结点称为兄弟结点。
- 语法如下：
  - element.nextSibling

    ```
    //element.parentNode.childNodes[index+1]
    ```

  - element.previousSibling

    ```
    // element.parentNode.childNodes[index-1]
    ```

获取结点信息

- 使用nodeName属性获取结点名称
- 语法如下：
  - Node.nodeName
  - 对于不同的结点类型，nodeName的返回值有所差异：

    元素结点：返回标记的名称，例如： 则返回div

    属性结点：返回属性的名称，例如：id=“div1”则返回id

    文本结点：返回文本的内容。

获取结点信息

- 使用nodeType以数字的形式返回他们的类型
- 语法如下:
  - Node.nodeType
  - 对于不同的结点类型，nodeType的返回值有所差异:

    元素结点：返回1

    属性结点：返回2

    文本结点：返回3。

获取结点信息

- 使用nodeValue获取结点的值
- 语法如下：
  - Node.nodeValue
  - 对于不同的结点类型，nodeValue的返回值有所差异：

    元素结点：返回null

    属性结点：返回undefined

    文本结点：返回文本内容

练习:

- 改写demo.html例子
- 使用nodeValue获取结点的内容.

处理属性结点

- 获取和设置属性结点的值
  -

```
var img = document.getElementById( “imgDemo” );
alert(img.src);
img.src= “hello1.jpg”
```

处理属性结点

- 使用setAttribute() 添加一个属性
  - elementNode.setAttribute(attributeNme,attributeValue);

- 使用getAttribute() 获取一个属性
  - elementNode.getAttribute(attributeNme);

处理文本结点

- 获取文本结点的值
  - 使用innerHTML或者nodeValue
  - 例如:

```
documents.getElementById(“demo”).innerHTML;
//documents.getElementById(“demo”).nodeValue;
```

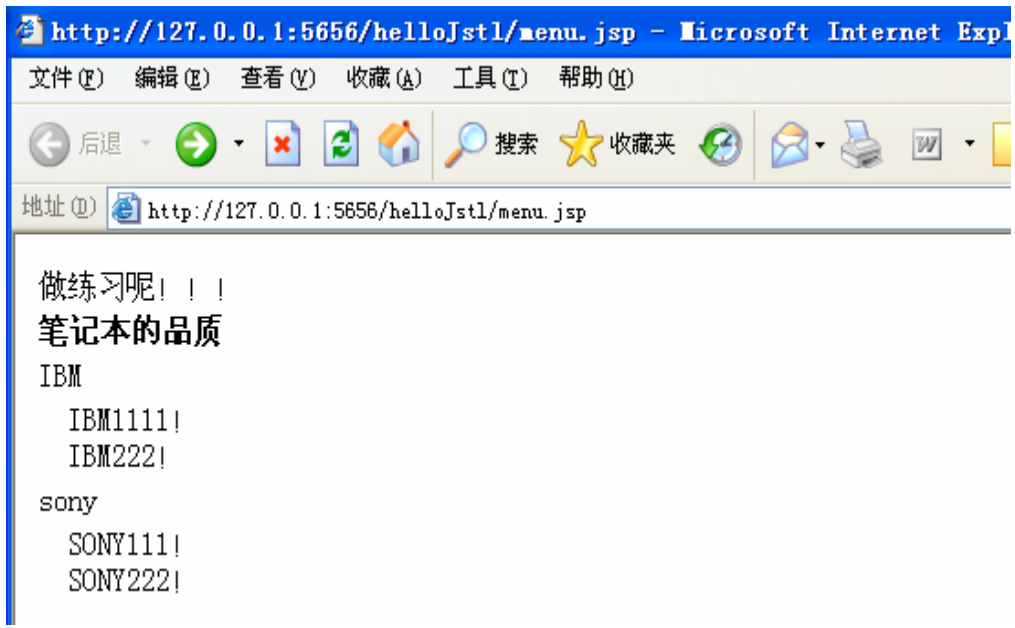
使用innerHTML改变结点的内容

- 可以使用innerHTML设置文本内容:

  document.getElementByld(“demo”).innerHTML=“hello”;

练习：


http://127.0.0.1:5656/helloJstl/menu.jsp - Mic
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://127.0.0.1:5656/helloJstl/menu.jsp
做练习呢！！！
笔记本的品质
IBM
sony


http://127.0.0.1:5656/helloJstl/menu.jsp - Microsoft Internet Expl
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://127.0.0.1:5656/helloJstl/menu.jsp
做练习呢！！！
笔记本的品质
IBM
IBM1111！
IBM222！
sony
SONY111！
SONY222！

改变文档的层次结构

- 使用document.createElement()方法创建元素结点
  - 例如：要创建一个 结点，可以这样实现

    Var divname=document.createElement(“div”);

改变文档的层次结构

- 使用document.createTextNode()方法创建文本结点
  - 例如：要创建一个hello文本结点，可以这样实现

    Var divname=document.createTextNode(“hello”);

改变文档的层次结构

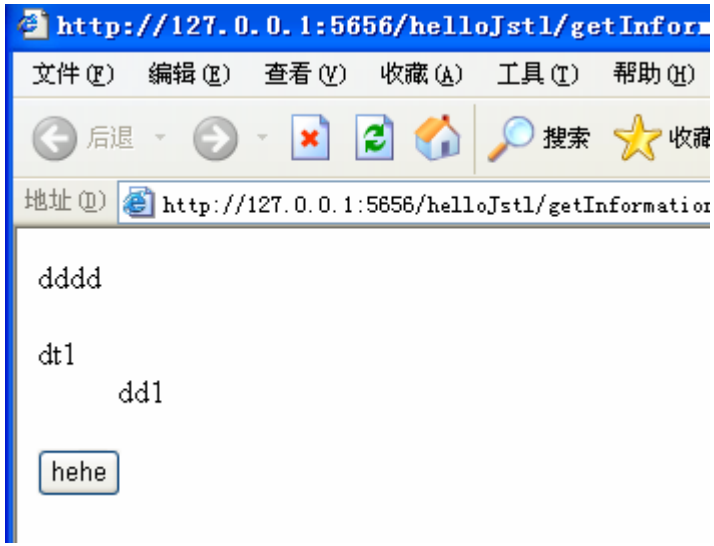
- 使用appendChild()方法添加结点
  - 例如：要添加一个新结点，可以这样实现
  - <dl id="dll">
    <dt>dt1</dt>
    <dd>dd1</dd>
  - </dl>
  - Var dll = document.getElementById("dll");
  - Var dd = document.createElement("dd");
  - Var tn = document.createTextNode("dd2");
  - dd.appendChild(tn);
  - dll.appendChild(dd);

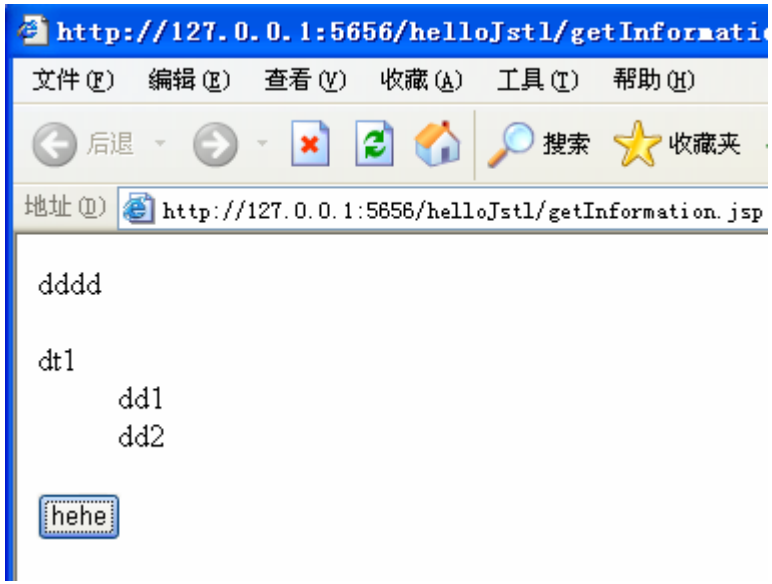
改变文档的层次结构

- 使用removeChild()方法删除结点
  - 例如：要删除一个结点，可以这样实现
  - <dl id="dll">
    <dt>dt1</dt>
    <dd>dd1</dd>
  - </dl>
  - Var dll = document.getElementById("dll");
  - Var dd = document.createElement("dd");
  - Var tn = document.createTextNode("dd2");
  - dll.removeChild(dll.childNodes[0]);

改变文档的层次结构

- 使用appendChild()方法添加结点





练习：

- 动态加载列表：


http://127.0.0.1:5656/helloJstl/dyList.jsp
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://127.0.0.1:5656/helloJstl/dyList.jsp
商品搜索
品牌选择：
请选择
商品信息


http://127.0.0.1:5656/helloJstl/dyList.jsp
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://127.0.0.1:5656/helloJstl/dyList.jsp
商品搜索
品牌选择：
IBM
商品信息
IBM1
IBM1
IBM2

练习：

- 动态刷新列表：


http://127.0.0.1:5656/helloJstl/aotuRefresh.
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退
搜索
收藏夹
地址(D) http://127.0.0.1:5656/helloJstl/aotuRefresh.jsp
正在热卖的电脑
型号 产品1
销售数量 100

改变文档的层次结构

- 使用insertBefore方法插入子结点
  - appendChild方法只能将结点添加到所有子结点之后，使用insertBdfore可以将结点插入到新结点之前。
  - <dl id="dll">
    <dt>dt1</dt>
    <dd>dd1</dd>
  - </dl>
  - Var dll = document.getElementById("dll");
  - Var dd = document.createElement("dd");
  - Var tn = document.createTextNode("dd2");
  - dd.appendChild(tn);
  - dll.insertBefore(dd,dll.childNodes[0]);//dll.appendChild(dd);

表格操作

- 尽管表格仍然对应一定结构的xml(html)标记,但使用标准DOM方法并不能在浏览器中正常工作,必须使用DHTMl中定义的接口对其进行操作.

表格操作

- 操作表格的属性和方法
  - 表格对象
    - rows：属性，返回所有行对象的集合数组
    - insertRow(下标)：在指定下标位置增加新行，下标从0开始，返回新增的行对象
    - deleteRow(下标)：删除指定下标位置的行，下标从0开始
  - 行对象
    - cells：属性，返回此行的所有单元格的集合数组
    - insertCell(下标)：在指定下标位置增加新单元格，下标从0开始，返回新增的单元格对象

表格操作

- 使用标准DHTML方法创建表格

```
<head>
<script>
function createTable(){
        var con = document.getElementById("tableTest");
        var varTable=document.createElement("table");
        varTable.setAttribute("border","1");
        varTable.setAttribute("boderColor","black");
        for(var i=0;i<5;i++){
                var varTr=varTable.insertRow(i);
                for(var j=0;j<4;j++){
                        var varTd = varTr.insertCell(j);
                        var varTn = document.createTextNode(i.toString()+j.toString());
                        varTd.appendChild(varTn);
                }
        }
        con.appendChild(varTable);
}
</script>
</head>
<body>
<div id ="tableTest"></div>
<input type="button" value ="createTable" onclick="createTable()">
</body>
```

End Thanks