Лабораторная работа № 7 XML

«Программная обработка XML документов с помощью XML DOM»

Цель работы:

Ознакомление с основными принципами XML DOM и методами программной обработки XML документов путем манипулирования узлами дерева документа.

Краткие теоретические сведения:

XML DOM определяет объекты и свойства всех XML элементов и методы (интерфейс) для доступа к ним. Иначе говоря, XML DOM описывает каким образом необходимо получать, изменять, добавлять и удалять XML элементы.

В соответствии с моделью DOM все, что содержится внутри XML документа - является узлом. То есть XML документ представляется в виде дерева узлов, которыми являются элементы, атрибуты и текст.

Поскольку структуры HTML и XML документов очень похожи, а HTML DOM и XML DOM являются частями более общего стандарта DOM, то и многие аспекты HTML DOM легко переносимы в XML DOM. Поэтому основное внимание будет уделено именно специфическим особенностям именно XML DOM. Рекомендуется предварительное ознакомление с лабораторной работой 3.

Структурный анализ (парсинг) XML

Все современные браузеры имеют встроенные XML анализаторы (парсеры) для чтения и обработки XML. Анализатор считывает XML документ, размещает его в памяти и преобразует в XML DOM объект, доступный для языков программирования. Все примеры здесь приведены на JavaScript.

Имеются некоторые отличия между анализаторами в Microsoft и в других браузерах. Первый поддерживает как загрузку XML файлов, так и текстовых строк, содержащих XML код, в то время как в других браузерах используются раздельные анализаторы. При этом все анализаторы имеют функции для перемещения по дереву XML документа, доступа, вставки и удаления узлов в дереве.

Рассмотрим пример загрузки XML объектов (файлов и строк) с помощью XML анализатора Microsoft.

```
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.load("timetable.xml");
```

В первой строке программы создается пустой объект XML документа Microsoft. Далее для предотвращения работы сценария до полной загрузки документа флаг асинхронности устанавливается в "false". В третьей строке содержится инструкция загрузить XML файл "timetable.xml".

В следующем пример происходит загрузка строки с XML кодом для последующего анализа.

```
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.loadXML(txt);
```

Следует обратить на разницу между методами load() и loadXML() по их назначению.

Замечание. *Современные браузеры не допускают междоменные обращения к файлам из соображений безопасности, т.е. сама веб-страница (с программным кодом) и XML файл должны*

физически находиться на одном сервере. В противном случае браузер выдаст сообщение об ошибке доступа.

Ниже приведены также кроссплатформенные реализации загрузки XML файла и XML строки соответственно.

```
<html>
	<body>
		<script type="text/javascript">
			try //Internet Explorer
			  {
			  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
			  }
			catch(e)
			  {
			  try //Firefox, Mozilla, Opera, etc.
			    {
			    xmlDoc=document.implementation.createDocument("","",null);
			    }
			  catch(e) {alert(e.message)}
			  }
			try
			  {
			  xmlDoc.async=false;
			  xmlDoc.load("timetable.xml");
			  document.write("xmlDoc is loaded, ready for use");
			  }
			catch(e) {alert(e.message)}
		</script>
	</body>
</html>
```

```
<html>
	<body>
		<script type="text/javascript">
			text="<timetable>";
			text=text+"<lesson>";
			text=text+"<timeFrom>08.00</timeFrom>";
			text=text+"<subject>Deutsch</subject>";
			text=text+"<teacher>Borisova</teacher>";
			text=text+"</lesson>";
			text=text+"/<timetable>";

			try //Internet Explorer
			  {
			  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
			  xmlDoc.async="false";
			  xmlDoc.loadXML(text);
			  }
			catch(e)
			  {
			  try //Firefox, Mozilla, Opera, etc.
			    {
			    parser=new DOMParser();
			    xmlDoc=parser.parseFromString(text,"text/xml");
			    }
			  catch(e) {alert(e.message)}
			  }
			document.write("xmlDoc is loaded, ready for use");
		</script>
	</body>
</html>
```

Программный интерфейс XML DOM

В рамках DOM модели XML можно рассматривать как множество узловых объектов. Доступ к ним осуществляется с помощью JavaScript или других языков программирования. Программный интерфейс DOM включает в себя набор стандартных свойств и методов. Свойства представляют некоторые сущности (например, <day>), а методы - действия над ними (например, добавить <lesson>). В XML DOM используются практически те же свойства и методы, что и в HTML DOM.

Например, результатом выполнения следующего ниже JavaScript кода будет текстовой содержимое элемента <subject> в файле timetable.xml. .

```
txt = xmlDoc.getElementsByTagName("subject")[0].childNodes[0].nodeValue;
```

Результат: "Deutsch".

В рамках DOM XML возможны 3 способа доступа к узлам:

1. С помощью метода getElementsByTagName(name). При этом возвращаются все узлы с указанным именем тэга (в виде индексированного списка). Первый элемент в списке имеет нулевой индекс.
2. Путем обхода узлов дерева с использованием циклических конструкций.
3. Путем перемещения по дереву с использованием отношений между узлами.

Для определения длины списка узлов используется свойство length.

Перемещение между узлами дерева

В XML DOM отношения между узлами определены в виде следующих свойств узлов:

- parentNode
- childNodes
- firstChild
- lastChild
- nextSibling
- previousSibling

Характер отношений между узлами представлен на следующем рисунке:

Игнорирование пустых текстовых узлов

Firefox и некоторые другие браузеры воспринимают неотображаемые символы как текстовые узлы (в отличие от Internet Explorer). Такая ситуация приводит к проблемам при использовании свойств firstChild, lastChild, nextSibling, previousSibling. Для того, чтобы игнорировать такие пустые текстовые узлы можно использовать следующий прием:

```
function get_nextSibling(n)
        {
        y = n.nextSibling;
        while (y.nodeType!=1)
          {
          y = y.nextSibling;
          }
        return y;
        }
```

Поскольку узлы элементов имеют тип 1, то в том случае, когда узел-потомок не является узлом элемента, будет происходить перемещение к следующему узлу до тех пор, пока не будет найден узел элемента.

Изменение значения атрибута

Узлы атрибутов могут принимать текстовые значения. Изменение этого значения реализуется либо через метод setAttribute(), либо через свойство узла атрибута nodeValue Метод setAttribute() изменяет значение существующего атрибута или создает новый атрибут.

Например:

```
xmlDoc = loadXMLDoc("timetable.xml");
x = xmlDoc.getElementsByTagName('lesson');
x[0].setAttribute("type","lab");
Свойство nodeValue можно использовать для изменения значения атрибута узла:
xmlDoc = loadXMLDoc("timetable.xml");
x = xmlDoc.getElementsByTagName("lesson")[0];
y = x.getAttributeNode("type");
y.nodeValue = "lab";
```

Удаление узла из дерева реализуется с помощью метода removeChild():

```
xmlDoc=loadXMLDoc("timetable.xml ");
y=xmlDoc.getElementsByTagName("lesson")[0];
xmlDoc.documentElement.removeChild(y);
```

Порядок выполнения лабораторной работы

При выполнении данной лабораторной работы потребуется XML документ:

```
<?xml version="1.0"?>
<timetable>
<day dayOfWeek="Monday">
 <lesson type="practical">
     < timeFrom>08.00</timeFrom#62;
      <timeTo>09.30</timeTo>
      <subject>Deutsch</subject>
      <teacher>Borisova</teacher>
      <room>216</room>
    </lesson>
    <lesson type="lecture">
      <timeFrom>09.40</timeFrom>
      <timeTo>11.10</timeTo>
      <subject>SAP Administration</subject>
      <teacher>Egorov</teacher>
      <room>384</room>
    </lesson>
    <lesson  type="practical">
     < timeFrom>11.20<</timeFrom>
      <timeTo>12.50<</timeTo>
      <subject>SAP Administration</subject>
      <teacher>Petrov</teacher>
      <room>384</room>
   < /lesson>
</day>
</timetable>
```

1. Создание JavaScript сценария загрузки XML документа.

   Создайте текстовый файл loadxmldoc.js, содержащий описание функции загрузки XML документа:

```
function loadXMLDoc(dname)
```

```
{
try //Internet Explorer
  {
  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  }
catch(e)
  {
  try //Firefox, Mozilla, Opera, etc.
    {
    xmlDoc=document.implementation.createDocument("","",null);
    }
  catch(e) {alert(e.message)}
  }
try
  {
  xmlDoc.async=false;
  xmlDoc.load(dname);
  return(xmlDoc);
  }
catch(e) {alert(e.message)}
return(null);
}
```

и сохраните его в той же папке, где находится файл timetable.xml.

Код вызова этой функции может выглядеть следующим образом:

```
<html>
        <head >
                <script type="text/javascript" src="loadxmldoc.js" >
                </script>
        </head>

        <body>
                <script type="text/javascript">
                   xmlDoc=loadXMLDoc("timetable.xml");
                   document.write("xmlDoc is loaded, ready for use");
                </script>
        </body>
</html>
```

2. Перемещение по дереву узлов.

Подготовьте следующую HTML страницу:

```
<html>
        <head>
                <script type="text/javascript" src="loadxmldoc.js">
                </script>
        </head>
        <body>
                <script type="text/javascript">
                   xmlDoc = loadXMLDoc("timetable.xml");
                   x = xmlDoc.getElementsByTagName("subject");
                   for (i=0; i<x.length; i++)
                    {
                       document.write(x[i].childNodes[0].nodeValue);
                       document.write("<br />");
                    }
                </script>
        </body>
</html>
```

После загрузки страницы в браузере можно будет увидеть следующий результат:

```
DeutschSAP AdministrationSAP Administration
```

3. Изменение значения элемента.

   Следующий пример демонстрирует изменение значения элемента <subject>:

```
xmlDoc=loadXMLDoc("timetable.xml");
x=xmlDoc.getElementsByTagName("subject")[0].childNodes[0];
x.nodeValue="Java programming";

x = xmlDoc.getElementsByTagName("subject");
for (i=0; i<x.length; i++)
{
  document.write(x[i].childNodes[0].nodeValue);
  document.write("<br />");
}}
```

   Внесите соответствующие изменения в предыдущую страницу и загрузите ее в браузере.

4. Перемещение по узлам дерева с использованием отношений между ними.

   Следующий код показывает, как используя отношения firstChild и nextSibling можно получить для текущего узла список его дочерних узлов:

```
x = xmlDoc.getElementsByTagName("lesson")[0].childNodes;
y = xmlDoc.getElementsByTagName("lesson")[0].firstChild;

for (i = 0; i < x.length; i++)
{
if (y.nodeType == 1)
  {
  document.write(y.nodeName + "<br />");
  }
y=y.nextSibling;
```

   Внесите необходимые изменения в html страницу и загрузите ее в браузере.

Контрольные задания:

В приведенном ниже XML документе описана экзаменационная ведомость:

```
<gradeReport id="120851">
<date>10-06-2008</date>
<subject>Computer Science Fundamentals</subject>
<examiner>prof.Litvinov</examiner>
   <gradeList>
        <gradeRecord id="1">
           <student>Ivanov</student>
           <grade>4</grade>
        </gradeRecord>
        <gradeRecord id="2">
           <student>Petrov</student>
           <grade>3</grade>
        </gradeRecord>
        <gradeRecord id="3">
           <student>Sidorov</student>
           <grade>5</grade>
        </gradeRecord>
   </gradeList>
</gradeReport>
```

1. Используя методы DOM XML, сформируйте HTML страницу, содержащую таблицу из трех столбцов: номер, студент, оценка.
2. Используя методы DOM XML, замените цифровые значения оценок их словесными эквивалентами, например "4" на "good".