

DOM

Ненко Табаков

Пламен Танов

Технологическо училище “Електронни системи”

Технически университет – София

15 октомври 2008



ЛИТЕРАТУРА НЕОБХОДИМИ ПРОГРАМИ

- W3C спецификация - <http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/>
- Java API документация - <http://java.sun.com/javase/6/docs/api/>
- Eclipse - www.eclipse.org
 - WTP – Web Tools Project

ВЪВЕДЕНИЕ

- *DOM (Document Object Model)* е програмен интерфейс за приложения (*API – Application Programming Interface*) за обработка на валиден *HTML* и добре оформен *XML*
- Определя логическата структура на документа и как той може да бъде достигнат (манипулиран)
- С *DOM* могат да се създават документи, да се предвижва в тях (да се обхожда структурата им) и да се променят, добавят или изтриват елементи и съдържание

КАКВО Е *DOM*

- Документите имат логическа структура, която е дърво
- Всеки документ съдържа един корен (документен елемент), един или повече елемента, нула или повече коментара и/или инструкции за обработка
- *DOM* определя интерфейсите и обектите, които представят и манипулират документ
- *DOM* определя семантиката на тези интерфейси и обекти – поведение и атрибути
- *DOM* определя връзката между тези интерфейси и обекти

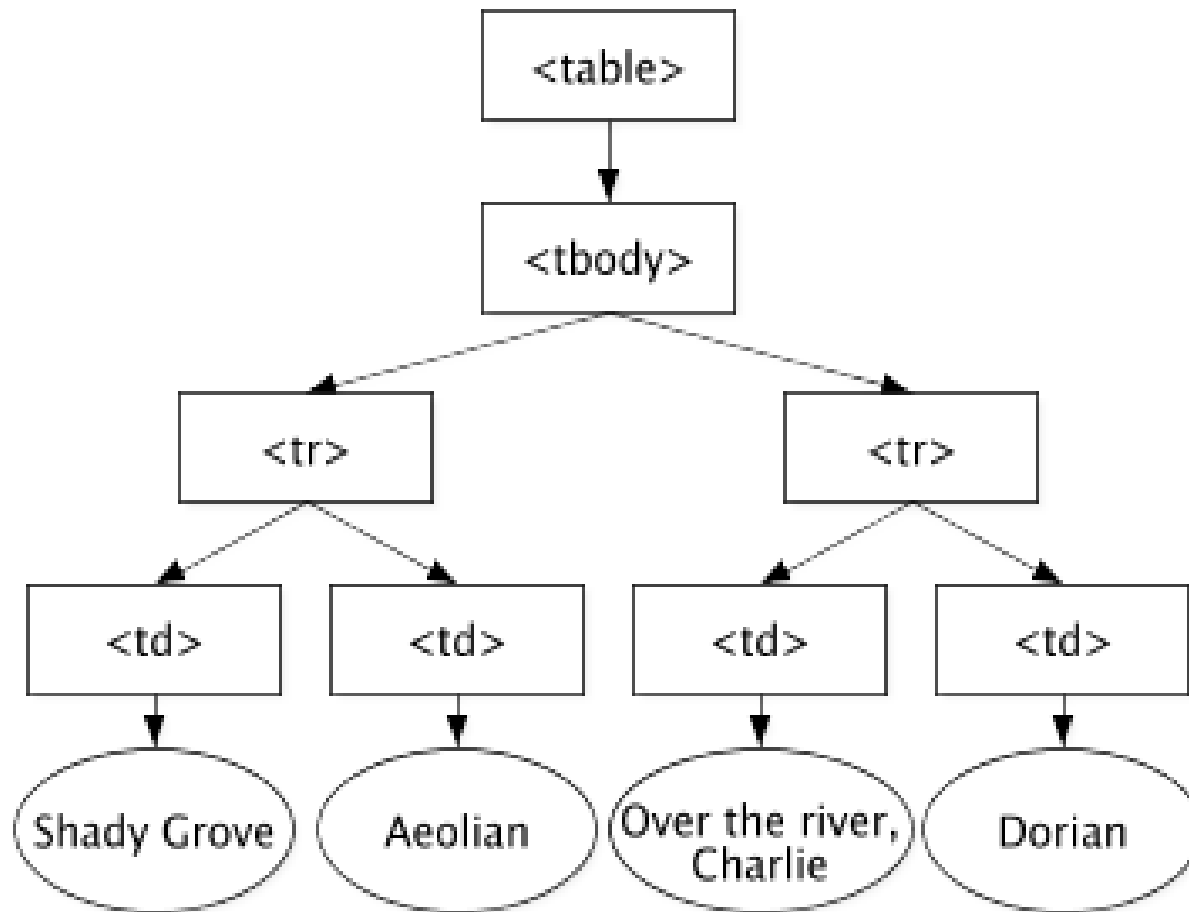
ПРИМЕР

какво е *DOM*

```
<table>
  <tbody>
    <tr>
      <td>Shady Grove</td>
      <td>Aeolian</td>
    </tr>
    <tr>
      <td>Over the River, Charlie</td>
      <td>Dorian</td>
    </tr>
  </tbody>
</table>
```

ПРИМЕР

какво е *DOM*



КАКВО НЕ Е *DOM*

- *DOM* не е начин за записване на *XML* и *HTML* документи. *DOM* определя как *XML* и *HTML* документи се представят като обекти, така че да може да се използват в обектно ориентирани програми
- *DOM* не е множество от структури от данни. Той е обектен модел, който определя интерфейси
- *DOM* не определя каква може да бъде информацията в един документ или как да бъде представена

ИСТОРИЯ

- *DOM* се е появил като спецификация за да може *JavaScript* и *Java* програми да бъдат преносими за различни уеб браузъри
- *Dynamic HTML* е непосредствения предшественик на *DOM*
- *DOM* се разработва от **DOM Working Group** в **W3C**

СТРУКТУРНИЯТ МОДЕЛ НА *DOM*

- *DOM* представя документите като йерархия от възли (*Node* обекти)
- Възлите могат да имат деца
- Децата могат да бъдат възли със свои деца или плоски данни без йерархия под тях – листа (*leaf*)
- За *XML* и *HTML* типовете възли и кои от тях могат да имат деца са както следва:
- *Document* -- *Element* (1), *ProcessingInstruction* (0..*),
Comment (0..*), *DocumentType* (0..1)
- *DocumentFragment* – като *Document*, само без *DocumentType*

СТРУКТУРНИЯТ МОДЕЛ НА *DOM*

- *DocumentType* – няма деца
- *Element* -- *Element*, *Text*, *Comment*, *ProcessingInstruction*, *CDATASection*
- *Attr* -- *Text*
- *ProcessingInstruction* – няма деца
- *Comment* – няма деца
- *Text* – няма деца
- *CDATASection* – няма деца
- *Entity* -- *Element*, *ProcessingInstruction*, *Comment*, *Text*, *CDATASection*

ИНТЕРФЕЙСЪТ *NodeList*

- *DOM* специфицира интерфейсът *NodeList*
- Той има един метод: `Node item(unsigned long index);`
- Този интерфейс представлява подреден списък и целта му е да представя колекции от данни от рода на:
 - Списък с децата на един възел
 - Списък с елементи върнати от метода *Element.getElementsByTagName(namespaceURI, localName)*

СТРУКТУРНИЯ МОДЕЛ НА *DOM*

- Интерфейсът *NodeList* е динамичен
- Това означава, че всяка промяна в структурата на документа се отразява и във всички обекти, които са инстанции на този интерфейс
- Ако съществува обект от тип *NodeList*, който съдържа децата на един *Element* и към този елемент се добавят още деца (или се изтрият, промени се съдържанието им и т.н.) то тези промени ще се отразят в *NodeList* обекта автоматично

СТРУКТУРНИЯ МОДЕЛ НА *DOM*

- За всеки тип данни в *DOM* са създадени интерфейси
- Интерфейсите *Text*, *Comment* и *CDATASection* са наследници на интерфейса *CharacterData*

ЙЕРАРХИЧЕН И ПЛОСЪК МОДЕЛ НА *DOM*

- Съществуват два подхода за работа с възли в *DOM*
- Първият подход е като се използва йерархия от интерфейси, които представят различните части на един документ – елемент, коментар, атрибут и т.н. Като тези класове са наследници на интерфейса *Node*
- Вторият подход е като се използва само интерфейсът *Node* и работата с документа става само чрез него и неговите методи и атрибути

ЙЕРАРХИЧЕН И ПЛОСЪК МОДЕЛ НА *DOM*

- Когато се използва йерархичният модел един възел (инстанция на *Node*) се представя (*casting*) като съответния тип – елемент (*Element* обект), коментар (*Comment* обект) и други. Всеки от тези интерфейси има удобни методи и полета, които за значими за типа си
- При плоския модел се използват само обекти от тип *Node*

```
node.getNodeName();  
  
Element element = (Element) node;  
element.getTagName();
```

КОНВЕНЦИЯ ЗА ИМЕНАТА

- Име на метод започващо с „*remove*“ се използва, когато методът премахва възел от структурния модел
 - Ако има смисъл той (възелът) се връща като резултат
- Име на метод започващо с „*delete*“ се използва, когато методът изтрива данни от възли
 - В този случай методът не връща резултат

ОСНОВНИ ТИПОВЕ В *DOM*

- *DOMString* – поредица от символи
- *DOMTimeStamp* – абсолютно или относително време – число в милисекунди
- *DOMUserData* – данни дефинирани от потребителя
- *DOMObject* - обект

DOMException

- *DOM* операциите генерират изключение в извънредни ситуации – когато операцията е невъзможно да се изпълни
- При възникване на грешка методите връщат специални кодови константи (достъп до елемент, който е извън обсега и т.н.)
- *DOMException* съдържа едно поле – код на грешката

ИНТЕФРЕЙСЪТ *NODE*

- *Node* интерфейсът е основният тип в целия *DOM*
- Той представя един възел в дървовидната структура на документа
- Методите в него позволяват придвижване в дървовидната структура на документа
- В наследниците на *Node* тези методи съществуват, но не винаги имат логически смисъл – например, интерфейсът *Text* (наследник на *Node*) не може да има деца и опитът да се добавят генерира изключение *DOMException*

ИНТЕФРЕЙСЪТ *DOCUMENT*

- Наследник на интерфейс *Node*
- *Document* интерфейсът представя цял *XML* документ
- Този интерфейс е коренът на документа
- Този интерфейс дефинира методи за създаване на:
 - Елементи
 - Атрибути
 - Коментари
 - Текстови възли
 - Функции за обработка

ИНТЕФРЕЙСЪТ *ELEMENT*

- Наследник на интерфейс *Node*
- *Element* интерфейсът представя един елемент в *XML* документ
- Елементите могат да имат атрибути асоциирани с тях

ИНТЕФРЕЙСЪТ *ATTR*

- Наследник на интерфейс *Node*
- *Attr* интерфейсът представя атрибут на даден елемент
- Макар и наследник на *Node* интерфейса, *DOM* спецификацията не счита атрибутите за част от дървото на документа. В този интерфейс методите за предвижване из дървото нямат никакъв СМИСЪЛ

DOM В JAVA

- <http://java.sun.com/javase/6/docs/api/org/w3c/dom/Node.html>
- <http://java.sun.com/javase/6/docs/api/org/w3c/dom/NodeList.html>
- **Използват се класовете и интерфейсите:**
 - `javax.xml.parsers.DocumentBuilder;`
 - `javax.xml.parsers.DocumentBuilderFactory;`
 - `javax.xml.parsers.ParserConfigurationException`

 - `org.w3c.dom.Document;`
 - `org.w3c.dom.Element;`
 - `org.w3c.dom.Node;`
 - `org.w3c.dom.NodeList;`
 - `org.xml.sax.SAXException;`
 - **и други**

DOM В JAVA

```
DocumentBuilderFactory factory = DocumentBuilderFactory
                                .newInstance();

DocumentBuilder builder = factory.newDocumentBuilder();

Document document = builder.parse(new
                                File("library.xml"));

Node firstChild = document.getFirstChild();

System.out.println("First child is " +
                   firstChild.getNodeName());
```


DOM В JAVA

```
public class NodeRecognizer {  
  
    private static String nodeTypes[] = { "none", "Element",  
    "Attr", "Text", "CDATA", "EntityRef", "Entity", "ProcInst",  
    "Comment", "Document", "DocType", "DocFragment",  
    "Notation" };  
  
    public static void main(String args[]) {  
        DocumentBuilderFactory factory =  
            DocumentBuilderFactory.newInstance();  
  
        try {  
            DocumentBuilder builder =  
                factory.newDocumentBuilder();  
  
            Document document = builder.parse(new File(  
                "library.xml"));  
        }  
    }  
}
```

DOM В JAVA

```
Node firstChild = document.getFirstChild();
NodeList children = firstChild.getChildNodes();
for (int i = 0; i < children.getLength(); i++) {
    Node node = children.item(i);

    System.out.println("Node " + node.getNodeName() +
        " is a/an " + nodeTypes[node.getNodeType()]);
}
} catch (ParserConfigurationException e) {
    e.printStackTrace();
} catch (SAXException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
}
```