ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Εισαγωγή στην Επιστήμη και Τεχνολογία των Υπηρεσιών

Ενότητα 12: Document Object Model (DOM) - 2

Χρήστος Νικολάου
Τμήμα Επιστήμης Υπολογιστών

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται στην άδεια χρήσης **Creative Commons** και ειδικότερα

Αναφορά – Μη εμπορική Χρήση – Όχι Παράγωγο Έργο v. 3.0

(Attribution – Non Commercial – Non-derivatives)

- Εξαιρείται από την ως άνω άδεια υλικό που περιλαμβάνεται στις διαφάνειες του μαθήματος, και υπόκειται σε άλλου τύπου άδεια χρήσης. Η άδεια χρήσης στην οποία υπόκειται το υλικό αυτό αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

XML
Document Object Model (DOM)
Part 2
605.444 / 635.444

David Silberberg

Lecture 19

Creating a Document Through DOM

- Let‟s create a new document *school.xml*:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="school.css"?>
<school id="0994">
      <name>Johns Hopkins University</name>
      <phone>800-548-3647</phone>
      <phone>410-516-8728</phone>
      <phone>443-778-6231</phone>
</school>
```

Imports

```
import java.io.*;

// DOM imports
import org.w3c.dom.Attr;
import org.w3c.dom.Comment;
import org.w3c.dom.Document;
import org.w3c.dom.DOMImplementation;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.ProcessingInstruction;
import org.w3c.dom.Text;

// Parser import
import org.apache.xerces.dom.DOMImplementationImpl;
import org.apache.xerces.parsers.DOMParser;
```

Create File

```
public class MakeFile {
    private static final String FILE_DIR = "";
    private FileWriter file;
    private BufferedWriter fOut;
    private boolean fCanonical;

    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Usage: java MakeFile [filename]");
            System.exit(0);
        }
        System.out.println("Making new file: " + args[0] + "\n");
        new MakeFile(args[0]);
    }
```

Create Document and PI

```
public MakeFile(String filename) {
      try {
         // Create new DOM tree
         // This is Apache XERCES specific
         DOMImplementation domImpl = new DOMImplementationImpl();
         // Create root element
         Document doc = domImpl.createDocument(null /*namespace*/,
                                  "school" /*root*/, null /*doctype*/);


         // Create PI
         ProcessingInstruction pi =
            doc.createProcessingInstruction("xml-stylesheet",
                                  "type=\"text/css\" href=\"school.css\"");
         doc.appendChild(pi);  // appends child to top of document
```

Create the Document Root

```
// Get the root - in this case, the root is "school"
Element root = doc.getDocumentElement();

// Set the root attribute
root.setAttribute("id", "0994");

// Create child elements
Element schoolName = doc.createElement("name");
Text schoolText =
    doc.createTextNode("Johns Hopkins University");
root.appendChild(schoolName);
schoolName.appendChild(schoolText);
```

Create Children

```
// Create child elements
Element phoneName = doc.createElement("phone");
Text phoneText =
    doc.createTextNode("800-548-3647");
root.appendChild(phoneName);
phoneName.appendChild(phoneText);

phoneName = doc.createElement("phone");
phoneText =
    doc.createTextNode("410-516-8728");
root.appendChild(phoneName);
phoneName.appendChild(phoneText);
```

Write Document

```
        phoneName = doc.createElement("phone");
         phoneText = doc.createTextNode("443-778-6231");
         root.appendChild(phoneName);
         phoneName.appendChild(phoneText);

         // Write out the file.
         // Assume a routine that writes out the file (as in the last lecture)
         file = new FileWriter(FILE_DIR +  filename + ".xml");
         fOut = new BufferedWriter(file);
         write(doc);
      }
      catch (IOException ioe) {
         System.out.println(ioe.getMessage());
      }
   }
```

Document Writer

```
/*
   * The Apache Software License, Version 1.1
   *
   *
   * Copyright (c) 1999, 2000 The Apache Software Foundation.  All rights
   * reserved.
   *
   * Redistribution and use in source and binary forms, with or without
   * modification, are permitted provided that the following conditions
   * are met:
   *
   * yada yada yada */
```

Write Document Node

```
/** Writes the specified node, recursively. */
public void write(Node node) throws IOException {
    // is there anything to do?
    if (node == null)
        return;
    short type = node.getNodeType();
    switch (type) {
    case Node.DOCUMENT_NODE: {
        if (!fCanonical) {
            fOut.write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
            fOut.flush();
        }
        Document document = (Document)node;
        write(document.getDocumentElement());
        break;
    }
```

Write Element Node

```
case Node.ELEMENT_NODE: {
   fOut.write('<');
   fOut.write(node.getNodeName());
   Attr attrs[] = sortAttributes(node.getAttributes());
   for (int i = 0; i < attrs.length; i++) {
      Attr attr = attrs[i];
      fOut.write(' ');
      fOut.write(attr.getNodeName());
      fOut.write("=\"");
      normalizeAndPrint(attr.getNodeValue());
      fOut.write("");
   }
   fOut.write('>');
   fOut.flush();
   Node child = node.getFirstChild();
   while (child != null) {
      write(child);
      child = child.getNextSibling();
   }
   break;
}
```

Write Entity Reference Node

```
case Node.ENTITY_REFERENCE_NODE: {
   if (fCanonical) {
      Node child = node.getFirstChild();
      while (child != null) {
         write(child);
         child = child.getNextSibling();
      }
   }
   else {
      fOut.write('&');
      fOut.write(node.getNodeName());
      fOut.write(';');
      fOut.flush();
   }
   break;
}
```

Write CDATA and Text Nodes

```
case Node.CDATA_SECTION_NODE: {
    if (fCanonical) {
        normalizeAndPrint(node.getNodeValue());
    }
    else {
        fOut.write("<![CDATA[");
        fOut.write(node.getNodeValue());
        fOut.write("]]>");
    }
    fOut.flush();
    break;
}

case Node.TEXT_NODE: {
    normalizeAndPrint(node.getNodeValue());
    fOut.flush();
    break;
}
```

Write Processing Instruction Node

```
    case Node.PROCESSING_INSTRUCTION_NODE: {
        fOut.write("<?");
        fOut.write(node.getNodeName());
        String data = node.getNodeValue();
        if (data != null && data.length() > 0) {
            fOut.write(' ');
            fOut.write(data);
        }
        fOut.write("?>");
        fOut.flush();
        break;
    }
}
```

Write Element Node

```
 if (type == Node.ELEMENT_NODE) {
      fOut.write("</");
      fOut.write(node.getNodeName());
      fOut.write('>');
      fOut.flush();
   }

} // write(Node)
```

Sort Attributes

```
/** Returns a sorted list of attributes. */
protected Attr[] sortAttributes(NamedNodeMap attrs) {

    int len = (attrs != null) ? attrs.getLength() : 0;
    Attr array[] = new Attr[len];
    for (int i = 0; i < len; i++) {
        array[i] = (Attr)attrs.item(i);
    }
    for (int i = 0; i < len - 1; i++) {
        String name = array[i].getNodeName();
        int index = i;
        for (int j = i + 1; j < len; j++) {
            String curName = array[j].getNodeName();
            if (curName.compareTo(name) < 0) {
                name = curName;
                index = j;
            }
        }
```

Sort Attributes

```
        if (index != i) {
            Attr temp = array[i];
            array[i] = array[index];
            array[index] = temp;
        }
    }

    return array;

} // sortAttributes(NamedNodeMap):Attr[]
```

Normalize and Print Text

```
/** Normalizes and prints the given string. */
   protected void normalizeAndPrint(String s)
       throws IOException {

       int len = (s != null) ? s.length() : 0;
       for (int i = 0; i < len; i++) {
           char c = s.charAt(i);
           normalizeAndPrint(c);
       }

   } // normalizeAndPrint(String)
```

Normalize and Print Text

```
/** Normalizes and print the given character. */
protected void normalizeAndPrint(char c)
    throws IOException {

    switch (c) {
    case '<': {
        fOut.write("&lt;");
        break;
    }
    case '>': {
        fOut.write("&gt;");
        break;
    }
    case '&': {
        fOut.write("&amp;");
        break;
    }
```

Normalize and Print Text

```
        case "": {
            fOut.write("&quot;");
            break;
        }
        case '\r':
        case '\n': {
            if (fCanonical) {
                fOut.write("&#");
                fOut.write(Integer.toString(c));
                fOut.write(';');
                break;
            }
            // else, default print char
        }
        default: {
            fOut.write(c);
        }
        }
    } // normalizeAndPrint(char)
}
```

General Document Operations

- **DOMImplementation**
  - Attempts to provide a standard interface onto creating DOM documents
  - **import org.w3c.dom.DOMImplementation;**
  - Creates Documents
  - Creates DOCTYPEs

```
public DocumentType createDocumentType(
    String qualifiedName, String publicId, String systemId);

public Document createDocument(
    String NamespaceURI, String qualifiedName, DocumentType docType);
```

Example

- This DOM code:

```
DOMImplementation domImpl = new DOMImplementationImpl();
DocumentType docType = domImpl.createDocumentType(
      "student", null /* public id */, "file:///DTDs/student.dtd" /* sys id */);
Document doc = domImpl.createDocument(null /*uri*/, "student"
    /*qname*/, docType);
```

- Creates an internal representation of the following XML document:

```
<?xml version="1.0"?>
<!DOCTYPE student SYSTEM "file:///DTDs/student.dtd">
<student>
</student>
```

Document Methods

- General **Document** methods
- **import org.w3c.dom.Document;**

- public DocumentType getDoctype()
  - Gets the document type
- public DOMImplementation getImplementation()
  - Gets the document implementation
  - Not much that you can do with it except create a DocumentType or Document
- public Element getDocumentElement()
  - Gets the root element of the document
  - This is the starting point for navigating through the document
  - This is also the starting point for creating/modifying the document

Document Creation Methods

- Creates parts of the document
- Does not place it into the document until told to do so
  - Handled by the Node operations
  - Node.insertBefore(…)
  - Node.replaceChild(…)
  - Node.appendChild(…)
  - Will be covered in the Node operations section
- public Element createElement(String tagName)
  - Creates element with „tagName“
  - Throws DOMException

Document Creation Methods

- public Text createTextNode(String textData)
  - Creates text node to be attached to an element
- public Comment createComment(String commentData)
  - Creates a comment node with the comment data
- public ProcessingInstruction createProcessingInstruction (String target, String data)
  - Creates a PI with target and data
  - Data must be a String in the format „attr="val" attr="val" …“
- public Attribute createAttribute(String attributeName)
  - Creates attribute name
  - Append text to it later

Document Nodes - Retrieval

- public NodeList getElementsByTagName(String tagname)
  - Very useful
  - Gets a list of nodes that are named „tagname“
  - Each node contains the entire subtree beneath the node
- public NodeList getElementsByTagNameNS

  (String namespaceURI, String tagname)
  - Similar to getElementsByTagName
  - Gets a list of nodes that are named „tagname“ in the namespace „namespaceURI“

Example

```
Document doc = domImpl.createDocument(null, "student",
    docType);
...
NodeList nodeList =
    (NodeList)doc.getElementsByTagName("phone");
System.out.println("The JHU telephone numbers are:");
for (int j=0; j<nodeList.getLength(); j++) {
      Node phoneNode = nodeList.item(j);
      // process Node
      String phoneNumber = phoneNode.getNodeValue();
      System.out.println("      " + phoneNumber + "\n");
}
```

Node Operations

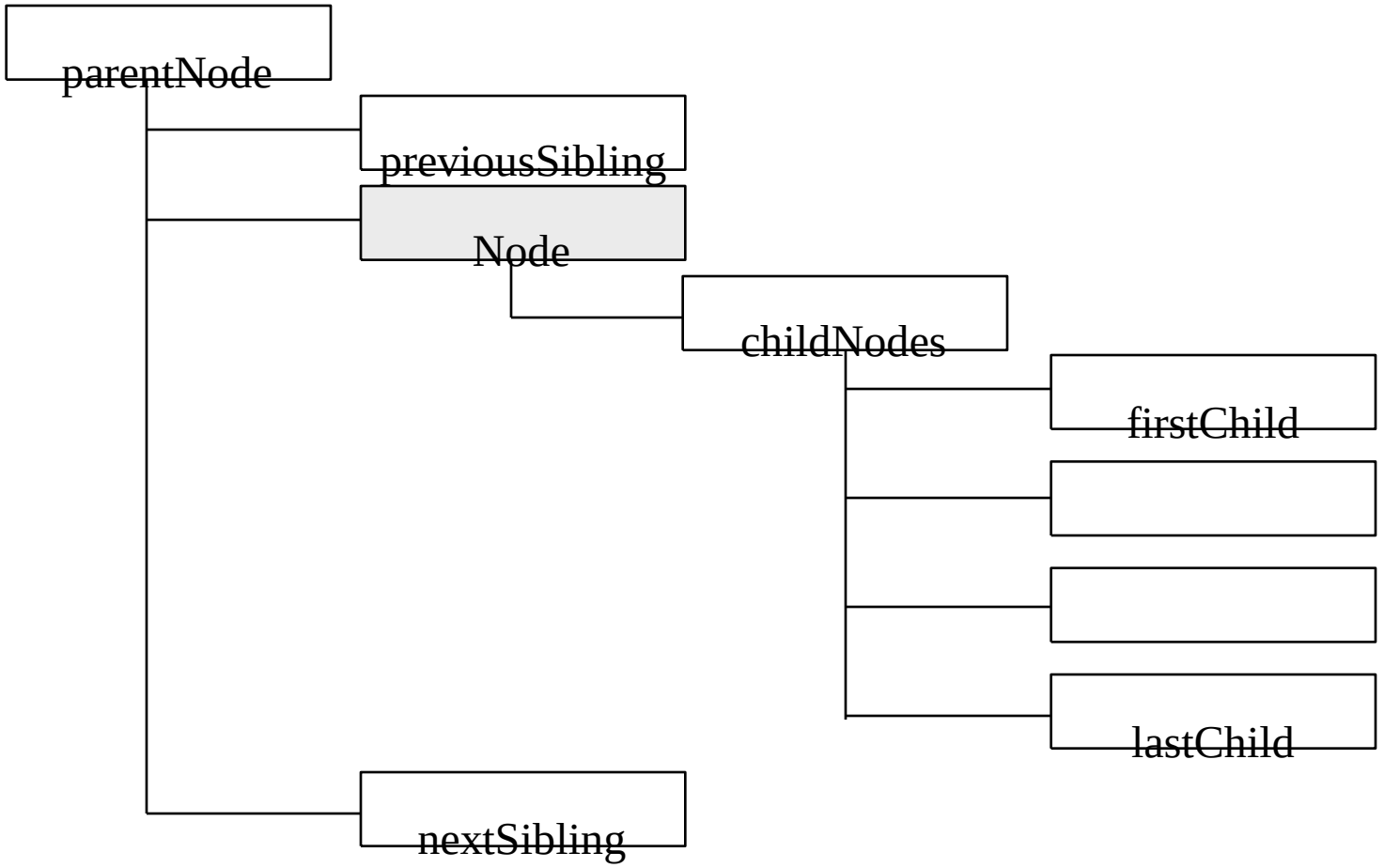
- Provides general operations on **Nodes**
- **import org.w3c.dom.Node;**

- public String getNodeName();
  - Retrieves node name
- public String getNodeValue() throws DOMException;
  - Retrieves the value of the node
  - Usually text, comment, etc.
- public void setNodeValue(String nodeValue) throws DOMException;
  - Sets the value of the node
  - Replaces value if it already exists

Node Operations (2)

- public short getNodeType()
  - Determines type of node for specific processing
  - ELEMENT_NODE
  - ATTRIBUTE_NODE
  - TEXT_NODE
  - CDATA_SECTION_NODE
  - ENTITY_REFERENCE_NODE
  - ENTITY_NODE
  - PROCESSING_INSTRUCTION_NODE
  - COMMENT_NODE
  - DOCUMENT_NODE
  - DOCUMENT_TYPE_NODE

Navigating a DOM Tree



Navigating a DOM Tree (2)

- These are very difficult to do in SAX
- public Node getParentNode();
  - Retrieves the parent node
- public NodeList getChildNodes();
  - Retrieves the child node
- public Node getFirstChild();
  - Retrieves the first child node
- public Node getLastChild();
  - Retrieves the last child node
- public Node getPreviousSibling();
  - Retrieves the previous node at the same level

Node Operations (3)

- public Node getNextSibling();
  - Retrieves the next node at the same level
- public NamedNodeMap getAttributes();
  - Retrieves a map of attributes

```
NamedNodeMap attrList = (NamedNodeMap)node.getAttributes();
for (int j=0; j<attrList.getLength(); j++) {
        Node attribute = attrList.item(j);
        // process attribute
}
```

Node Operations (4)

- public Document getOwnerDocument();
  - Gets the owner document for processing
  - (It is unlikely that you do not already have this.)
- public Node insertBefore (Node newChild, Node refChild) throws DOMException;
  - Places a new node before the referenced node
  - New node becomes the „previousSibling“ of the referenced node
- public Node replaceChild(Node newChild, Node oldChild) throws DOMException;
  - Removes the old child
  - Puts in its place the new Child

Node Operations (5)

- public Node removeChild(Node oldChild) throws DOMException;
  - Removed the child from the node
  - parentNode.removeChild(someChildNode);

- public Node appendChild (Node newChild) throws DOMException;
  - Appends the new child to the end of the child node list of the parent node
  - parentNode.appendChild(someChildNode);

Node Operations (6)

- public Node cloneNode(boolean deep)
  - Clones the current node
  - Either makes a deep or shallow copy
- public String getNamespaceURI();
  - Retrieves the URI of the namespace of the current node
- public String getPrefix();
  - Retrieves the prefix of the node name if it is a Namespace prefix
  - Otherwise, a zero-length string is returned
- public String setPrefix() throws DOMException
  - Sets the prefix of a node name

Example Node Operations

```
// Assume that „doc“ is the current Document
Document doc = domImpl.createDocument(null, "student", docType);
// Create document
…
// Process document
Element root = doc.getDocumentElement();
Node node = root.getFirstChild();
System.out.println("The JHU telephone numbers are:");
while (node != null) {
        if (node.getNodeName().equalsIgnoreCase("phone") {
                System.out.println("      " + node.getNodeValue() + "\n");
        }
        node = node.getNextSibling();
}
```

Element Operations

- Represent XML **Elements**
- **import org.w3c.dom.Element;**
- Elements are types of nodes
  - Element-specific methods mostly deal with the element name, attributes, and namespaces
  - Methods that deal with children are mostly found in Node methods because these are common to all nodes
- public String getTagName();
  - Gets the name of the element
- public String getAttribute(String name)
  - Gets the attribute value of a specific attribute

Element Operations (2)

- public void setAttribute(String name, String value) throws DOMException
  - Sets an attribute‟s value
- public void removeAttribute (String name) throws DOMException
  - Removes an attribute
- public Attr getAttributeNode(String name)
  - Gets the attribute as a node
- public void setAttributeNode (Attr newAttr) throws DOMException
  - Defines a new attribute for the element
- public void removeAttribute (Attr oldAttr)
  - Removes old attribute

Attribute Operations

- **Attr**ibutes are children of Elements
- **import org.w3c.dom.Attr;**

- public String getName()
  - Gets the name of the Attribute
- public String getValue()
  - Gets the value of the Attribute
- public void setValue(String value) throws DOMException;
  - Sets the attribute value
- public Element getOwnerElement()
  - Gets the owner element.

Setting Attributes

```
// Give each phone number a unique id
int id = 0;
Element root = doc.getDocumentElement();
Node node = root.getFirstChild();
while (node != null) {
        if (node.getNodeType() == Node.ELEMENT_NODE ) {
                Element  elt = (Element)node;
                if (elt.getTagName().equalsIgnoreCase("phone") {
                        Attr idAttr = doc.createAttribute("id");
                        idAttr.setValue(new String(++id));
                        elt.setAttributeNode(idAttr);
                }
        }
        node = node.getNextSibling();
}
```

Result

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="school.css"?>
<school id="0994">
        <name>Johns Hopkins University</name>
        <phone id=“1”>800-548-3647</phone>
        <phone id=“2”>410-516-8728</phone>
        <phone id=“3”>443-778-6231</phone>
</school>
```

Character Data Methods

- **CharacterData** is a „super“ interface for
  - **import org.w3c.dom.CharacterData;**
  - **import org.w3c.dom.CDATASection;**
  - **import org.w3c.dom.Comment;**
  - **import org.w3c.dom.Text;**
- Methods for accessing and setting data within a character data node

- public String getData() throws DOMException;
  - Gets the actual data

Character Data Methods (2)

- public String setData(String data) throws DOMException;
  - Sets the actual data
- public int getLength();
  - Gets the length of the data string
- public String substringData(int offset, int count) throws DOMException;
  - Gets a subset of the data
- public String appendData(String data) throws DOMException;
  - Appends data onto the pre-existing text

Character Data Methods (3)

- public String insertData(int offset, String data) throws DOMException;
  - Inserts the data into the character string
- public String deleteData(int offset, String data) throws DOMException;
  - Deletes the data from the character string
- public String replaceData(int offset, int count, String data) throws DOMException;
  - Replaces the data in the character string

Example Character Data

```
// Remove the area code from the telephone number and place it as an attribute
// in the element tag
Element root = doc.getDocumentElement();
Node node = root.getFirstChild();
while (node != null) {
      if (node.getNodeType() == Node.ELEMENT_NODE ) {
            Element  elt = (Element)node;
            if (elt.getTagName().equalsIgnoreCase("phone") {
                  moveAreaCode(elt);
            }
      }
      node = node.getNextSibling();
}
```

Example Character Data (2)

```
public void moveAreaCode(Element elt) {
    Document doc = elt.getOwnerDocument();
    Text  phoneNumber = (Text)elt.getFirstChild();
    if (phoneNumber.getLength() >= 3) {
        String areaCode = phoneNumber.substringData(0, 3);
        Attr idAttr = doc.createAttribute("areacode");
        idAttr.setValue(areacode);
        elt.setAttributeNode(idAttr);
        phoneNumber.deleteData(0,4);
    }
}
```

Result

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="school.css"?>
<school id="0994">
        <name>Johns Hopkins University</name>
        <phone areacode=“800”>548-3647</phone>
        <phone areacode=“410”>516-8728</phone>
        <phone areacode=“443”>778-6231</phone>
</school>
```

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης