Homework: XML Exercise

1. Objectives

- Become familiar with the DOM paradigm;

- Use an existing XML parser;

- Transform the content of an XML document into an HTML page.

2. Description

You are required to write a HTML/JavaScript program, which takes the URL of an XML document containing the top online Newspaper information, parses the XML file, and extracts the list of top online newspapers, displaying them in a table. The JavaScript program will be embedded in an HTML file so that it can be executed within a browser.

- Your program should display a text box to enter the XML file name as shown below on Figure 1. On clicking the "Submit Query" button, your program should display the table as shown below, Figure 2. If the text box is left empty and Submit Query is clicked, an appropriate error message must be shown.

Enter URL for Newspaper List XML File

w4-reference/newslist.xml

Submit Query

Figure 1: Initial Page

- Once the XML file downloads, a JavaScript function within the HTML file parses the XML document that was passed as an input to the popped up window.

- After parsing the XML document, a table should be displayed

consisting of the data for all top online Newspapers that are contained in the XML file. For example, given the following XML document:

http://cs-server.usc.edu:45678/hw/hw4/newslist.xml

the table below should be displayed:

Top Slot	Newspaper	City / State	HomePage	Logo
1	The New York Times	• New York • NY	http://www.nytimes.com	
2	Daily News	• New York • NY	http://www.nydailynews.com	
3	Washington Post	• Washington • DC	https://www.washingtonpost.com/	
4	USA Today	• Arlington • VA	http://www.usatoday.com	
5	Los Angeles Times	• Los Angeles • CA	http://www.latimes.com	
6	The Washington Times	• Washington • DC	http://www.washingtontimes.com	
7	The Wall Street Journal	• New York • NY	http://www.wsj.com	
8	Greenwich News	• Chicago • IL	http://www.chicagotribune.com	

Figure 2: Table containing newspapers from newsslist.xml

Here is a version of the *newslist.xml* file containing the data that is displayed above:

```xml
<?xml version="1.0"?>
<Workbook>
 <Table>
  <Header>
   <Data>Slot</Data>
   <Data>Newspaper</Data>
   <Data>Location</Data>
   <Data>HomePage</Data>
   <Data>Logo</Data>
  </Header>

  <Row>
   <Slot>1</Slot>
   <Newspaper>The New York Times</Newspaper>
   <Location>
      <City>New York</City>
      <State>NY</State>
   </Location>
   <HomePage>http://www.nytimes.com</HomePage>
   <Logo>http://cs-server.usc.edu:45678/hw/hw4/NYT_logo.png</Logo>
  </Row>

  <Row>
   <Slot>2</Slot>
   <Newspaper>Daily News</Newspaper>
   <Location>
      <City>New York</City>
      <State>NY</State>
   </Location>
   <HomePage>http://www.nydailynews.com</HomePage>
   <Logo>http://cs-server.usc.edu:45678/hw/hw4/DN_logo.png</Logo>
  </Row>

  <Row>
   <Slot>3</Slot>
   <Newspaper>Washington Post</Newspaper>
   <Location>
      <City>Washington</City>
      <State>DC</State>
   </Location>
   <HomePage>https://www.washingtonpost.com/</HomePage>
   <Logo>http://cs-server.usc.edu:45678/hw/hw4/WP_logo.png</Logo>
  </Row>

  <Row>...</Row>
  <Row>...</Row>
  <Row>...</Row>
  <Row>...</Row>
 </Table>
</Workbook>
```

3. Error Handling

In case of a parsing error, your program should show an alert box indicating an error was detected. For example if you try to load the incorrectly formatted XML file:

http://cs-server.usc.edu:45678/hw/hw4/invalid.xml

then you should show an alert box if the XML file is not valid, as in the following figure:
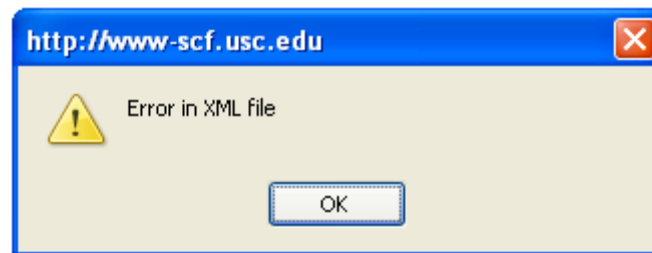

Figure 3: Error generated from invalid.xml in Firefox

Another error condition that should be checked for is an XML file containing NO radio stations. An example of an XML files which does not contain company entries:

```xml
<?xml version="1.0"?>
<Workbook>
  <Table>
    <Header>
        <Data>Slot</Data>
        <Data>Newspaper</Data>
        <Data>Location</Data>
        <Data>HomePage</Data>
        <Data>Logo</Data>
    </Header>
  </Table>
</Workbook>
```

In addition, you program should handle the case when the XML file does not exist. A proper message should be displayed.

The schema of the input XML files won't change. We won't test the case where the order of tags is changed or one of the tags is missed. In other words, every *Row* always contains the sub-tags: *Slot*, *Newspaper*, *Location*, *HomePage*, and *Logo* in the "same" given order. Note that inside the

Location tag, there may be ZERO, ONE or TWO tags (ZERO tags will mean an Internet-only newspaper).

If the value of a tag is empty, you should display a blank cell or N.A.

You are required to handle the case where the Header data values are different. Please note that the Data tag values might differ but the XML structure remains the same. For example, the Header can be,

```
<Header>
    <Data>Index</Data>
    <Data>Paper</Data>
    <Data>City / State</Data>
    <Data>HomePage</Data>
    <Data>Logo</Data>
</Header>
```

instead of,

```
<Header>
    <Data>Top Slot</Data>
    <Data>Newspaper</Data>
    <Data>Location</Data>
    <Data>HomePage</Data>
    <Data>Logo</Data>
</Header>
```

No other error conditions need be checked. In all cases if an error is found your program should show an alert box indicating the error was detected.

4. Hints

- *Step 1: Writing Your HTML/JavaScript program - Using the DOM Parser*

  Here's how you could use the Microsoft DOM API and the Mozilla DOM API (used in Firefox) to load and parse an XML document into a DOM tree, and then use the DOM API to extract information from that document.

  ```
  <script type="text/javascript">
  var xmlDoc;
  function loadXML(url) {
      if (window.XMLHttpRequest)
    {
  ```

```
       // code for IE7+, Firefox, Chrome, Opera, Safari
       xmlhttp=new XMLHttpRequest();
    }
  else
   {
    // code for IE6, IE5
      xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
  xmlhttp.open("GET",url,false);  //open, send, responseXML are
  xmlhttp.send();              //properties of XMLHTTPRequest
  xmlDoc=xmlhttp.responseXML;
  return xmlDoc;
 }
// ....... processing the document goes here
</script>
```

Now you can generate the HTML table from the DOM tree. You can assume that every xml file will have identical tag names. However, the company entries may occur in any order.

Your task is to write a program that transforms this type of XML file into the table as shown above.

For example you can access the child nodes of the documents as follows:

        footb = xmlDoc.documentElement.childNodes;

Note that unlike the Java-based DOM, which provides methods such as getChildNodes( ) and getNodeType( ) that return respectively a node list of children of a current node and the type of a node, with the DOM you have to access the element properties directly, as in:

        footbNodeList = footb.item(i).childNodes;
        if (footbNodeList.item(j).nodeType == ELEMENT_NODE)

Below is the link to the web page that demonstrates how to handle white spaces in Mozilla:

http://developer.mozilla.org/en/docs/Whitespace_in_the_DOM

- *Step 2: Display the Resulting HTML Document*

  You should use the DOM document.write method to output the required HTML.

- *Step 3: Use JavaScript control syntax*

  The only program control statements that you will need to use for this exercise are the "if" and the "for" statements. The syntax of both statements is practically identical to the syntax of the corresponding statement in the C, C++ and Java languages, as in:

  ```
  if (footbNodeList.item(j).nodeName=="Logo") {
      // do stuff
  }
  for (j=0;j<footbNodeList.length;j++) {
      // do more stuff
  }
  ```

Please make a note of the following issue:

Cross-Site Scripting (XSS):

JavaScript cannot call the resources from another domain. This is called cross side scripting which is not allowed in browsers. Therefore, you must put your XML files and your script in the same domain. Please make sure, when you are testing your implementation, to place both the HTML file and the XML file on your local machine IN THE SAME FOLDER. The files can be copied from here:

http://cs-server.usc.edu:45678/hw/hw4/invalid.xml

http://cs-server.usc.edu:45678/hw/hw4/newslist.xml

Window.open() method must be used to pop up a new window which would display the final widget.

Image files can be either local or remote, as these files do not exhibit the cross-site scripting issue.

USC Disclaimer:
You should avoid displaying the USC disclaimer in your pages. Otherwise, this will result in a 0.5-point penalty.

Scrollable Window:
The popup window should be scrollable so the user can read all records listed in the window.

6. Material You Need to Submit

On your course homework page, your link for this homework should go to a page that looks like the one displayed in the Figure on page 1. **This page should include your entire JavaScript/HTML/CSS program in a single file**. Also you should submit your source code electronically to the csci571 account so that it can be graded and compared to all other students' code via the MOSS code comparison tool. If your submission is composed of multiple files, 3 points will be deducted.