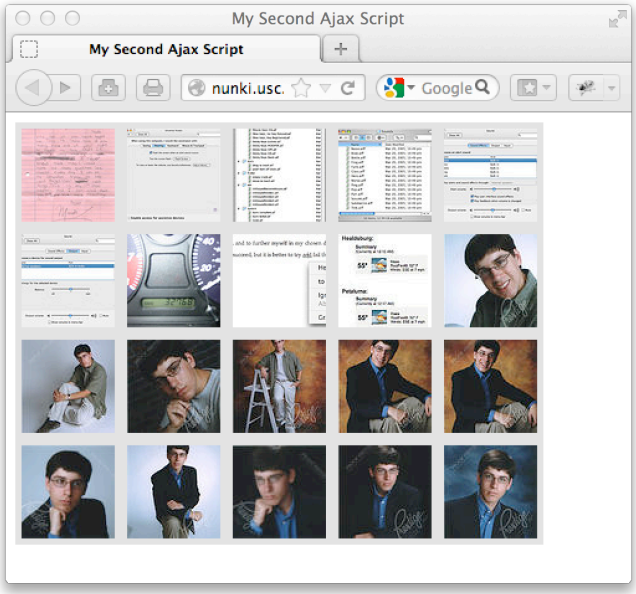
Computer Science 571 2nd Exam
Prof. Papa
Tuesday, December 6, 2012, 7:00pm – 8:20pm

Name: **Student ID Number:**

1. **This is a closed book exam.**
2. **Please answer all questions on the test**

JavaScript and Ajax Questions [20 pts]



Below is the HTML source code that produces the web page above.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html>
  <head>
    <title>My Second Ajax Script</title>
    <script src="script02.js" type="text/javascript" language="Javascript">
    </script>
  </head>
  <body>
    <div id="pictureBar">
    </div>
  </body>
</html>
```

Below is the JavaScript source code, script02.js, that was imported into the HTML above, but some of the lines are missing, replaced by XXXXXXXs. Fill in the missing code. Notice that the XMLHttp request invokes a file named "flickrfeed.xml" containing the picture data.

```
window.onload = initAll;
var xhr = false;

function initAll() {
	if (window.XMLHttpRequest) {
		xhr = new XMLHttpRequest();
	}
	else {
		if (window.ActiveXObject) {
			try {
				xhr = new ActiveXObject("Microsoft.XMLHTTP");
			}
			catch (e) { }
		}
	}

	if (xhr) {
		xhr.onreadystatechange = showPictures;
		xhr.open("GET", "flickrfeed.xml", true);
		xhr.send(null);
	}
	else {
		alert("Sorry, but I couldn't create an XMLHttpRequest");
	}
}

function showPictures() {
	var tempDiv = document.createElement("div");
	var pageDiv = document.getElementById("pictureBar");

	if (xhr.readyState == 4) {
		if (xhr.status == 200) {
			tempDiv.innerHTML = xhr.responseText;
			var allLinks = tempDiv.getElementsByTagName("a");

			for (var i=1; i<allLinks.length; i+=2) {
				pageDiv.appendChild(allLinks[i].cloneNode(true));
			}
		}
		else {
			alert("There was a problem with the request " +
xhr.status);
		}
	}
}
```

REST Questions [10 pts]

Each question is worth 2 points.

Q1: What are the 3 fundamental aspects of REST design patters?

A1: client, server and resources

Q2: What are the 4 simple operations of REST?

A2: PUT, GET, POST, DELETE

Q3: What is the typical representation of resources in REST?

A3: documents

Q4: What is the major reason of the gain in popularity of REST versus other approaches?

A4: simplicity

Q5: REST:

[X] is Platform independent
[X] is Language Independent
[X] is based on the HTTP standard
[X] is able to run behind firewalls
[ ] is a W3C recommendation
[ ] is based on cookies
[X] does not offer built-in security
[ ] ALL OF THE ABOVE

XML Schema Questions [20 pts]

Consider the following XML empty complex element that has no content, and only one attribute:

```
<product prodid="1345" />
```

[10 pts] Write an XML Schema that defines a "product" element using, xs:element. Xs:complexType and positive integer xs:attribute:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

<xs:element name="product">
  <xs:complexType>
    <xs:attribute name="prodid"
type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

```
</xs:schema>
```

Consider the following simple XML instance document, note.xml:

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

[10 pts] Write an XML Schema that conforms to such document:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Java Servlet Questions [10 pts]

Below is the Java source which implements a portion of the proxy back-end of Homework #8, but some of the code is missing, replaced by XXXXXXXs. Fill in the missing code.

```
import java.io.*;
import java.net.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;

public class SimpleServlet extends HttpServlet {

      public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            PrintWriter out = response.getWriter();
        try {

            String title = request.getParameter("title");
            String type = request.getParameter("title_type");


            if(type.equalsIgnoreCase("all types"))
                  type = "all_types";
            else if(type.equalsIgnoreCase("feature film"))
                  type = "feature";
            else if(type.equalsIgnoreCase("tv series"))
                  type = "tv_series";
            else if(type.equalsIgnoreCase("video games"))
                  type = "game";

            response.setContentType("text/html");
          String data = getPerlData(title,type);

          String json = parseXMLData(data);
          out.println(json);

        } catch(Exception e) {
            System.out.println(e.getMessage());
        }
      }

      public String getPerlData(String title, String type) throws Exception
{

            String url_title = "title="+title;
            String url_type = "type="+type;
        URL url = new URL("http://cs-server.usc.edu:18493/cgi-
bin/imdb_return_xml.pl?"+url_title+"&"+url_type);
        URLConnection connection = url.openConnection();

        BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
        String xmlString;
        String totalString="";
        while((xmlString = in.readLine())!=null)
          {
            totalString += xmlString;
          }
        in.close();

        return totalString;
      }
```

```
      public String parseXMLData(String data) throws JDOMException,
IOException {

            SAXBuilder builder = new SAXBuilder();
            Document doc = builder.build(new StringReader(data));
            Element rsp = doc.getRootElement();
            Element results = rsp.getChild("results");

            int count = results.getAttribute("total").getIntValue();
            List resultList = results.getChildren("result");

            String[] cover = new String[50];
            String[] title = new String[50];
            String[] year = new String[50];
            String[] director = new String[50];
            String[] rating = new String[50];
            String[] details = new String[50];

            if(count > 5)
                  count = 5;

            for(int i = 0; i < count ; i++) {
                  Element individualResult = (Element)resultList.get(i);
                  cover[i] =
individualResult.getAttribute("cover").getValue();
                  title[i] =
individualResult.getAttribute("title").getValue();
                  year[i] =
individualResult.getAttribute("year").getValue();
                  director[i] =
individualResult.getAttribute("director").getValue();
                  rating[i] =
individualResult.getAttribute("rating").getValue();
                  details[i] =
individualResult.getAttribute("details").getValue();
            }
            String parsedString = "{\n"+"\"results\":{\n";
            parsedString += "\"result\":[\n";
            int i;
            for(i=0;i<count-1;i=i+1)
            {
                  parsedString += "{\"cover\":"+"\""+cover[i]+"\",\n";
                  parsedString += "\"title\":"+"\""+title[i]+"\",\n";
                  parsedString += "\"year\":"+"\""+year[i]+"\",\n";
                  parsedString += "\"director\":"+"\""+director[i]+"\",\n";
                  parsedString += "\"rating\":"+"\""+rating[i]+"\",\n";
                  parsedString += "\"details\":"+"\""+details[i]+"\"\n";
                  parsedString += "},\n";
            }
            parsedString += "{\"cover\":"+"\""+cover[i]+"\",\n";
            parsedString += "\"title\":"+"\""+title[i]+"\",\n";
            parsedString += "\"year\":"+"\""+year[i]+"\",\n";
            parsedString += "\"director\":"+"\""+director[i]+"\",\n";
            parsedString += "\"rating\":"+"\""+rating[i]+"\",\n";
            parsedString += "\"details\":"+"\""+details[i]+"\"\n";
            parsedString += "}]}\n}";
            return parsedString;
```

```
        }
}
```

Web Performance Questions [10 pts]

Each question is worth 2 points.

Q1: Where should you put stylesheets and why to optimize performance?

A1: At the top, because IE blocks rendering of the page until all style sheets have been examined.

Q2: Where should you put scripts and why to optimize performance?

A2: At the bottom, because scripts block everything from rendering below them in the page

Q3: Should you "inline" JavaScript code or use external files and why to optimize performance?

A3: Use external files because they are cached

Q4: Should you scale images or not, and why?

A3: No, because 1) rescaling in the browser is time consuming and 2) sending an image larger than needed transfers more bytes than necessary

Q5: What are the two rules that minimize the transfer of information between server and browser?

A5: compression and JavaScript minification

HTML5 Questions [10 pts]

Each question is worth 2 points.

Q1: In HTML5 is it possible to perform "document editing" just using HTML elements and attribues?

A1: No, JavaScript code using the HTML5 Document Editing API is required.

Q2: Which of the following capabilities have included in HTML5?

A2:

[X] canvas
[X] video and audio
[X] local SQL database
[X] geolocation
[X] CSS 2D/3D transformations
[ ] flash plugin
[ ] ALL OF THE ABOVE

Q3: Why have FRAMES been removed from HTML5?

A3: Frames have been removed from HTML5 because their usage affected usability and accessibility for the end user in a negative way in HTML4

Q4: What happened to the "align" attribute used on caption, iframe, img, input, and many other elements in HTML4?

A4: they have been removed from HTML5 and are handled by CSS exclusively

Q5: Consider the following HTML5:

```
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #000000;">
</canvas>
```

What is the purpose of the "id" attribute?

A5: It is used to obtain a "handle" to the canvas using getElementById so that a "drawing context" object can be create to draw on the canvas.

JSON /AJAX Questions [20 pts]

Each is worth 2 points.

Q1: Of the URLs below, which have the same origin?

a. http://www.ajaxbook.com
b. http://www.ajaxbook.com:8443
c. https://www.ajaxbook.com
d. http://ajaxbook.com
e. http://carsearch.ajaxbook.com

A1: All of the above URLs have different origins

Q2: Creating an instance of the XMLHttpRequest object is the same in all browsers

a. True
b. False

A2: False

Q3: Which XMLHttpRequest method actually makes the request?

a. getAllResponseHeaders()
b. makeRequest()
c. open()
d. send()

A3: send()

Q4: What MIME type must be set when using XMLHttpRequest to send a POST request of a form?

a. application/octet-stream
b. Application/xhtml+xml
c. application/x-www-form-urlencoded

A4: c

Q5: Which of the following MIME types will cause the web browser to automatically parse an XML response?

a. text/xml
b. text/javascript
c. text/plain
d. text/html

A5: a

Q6: Which JavaScript method is used primarily with JSON?

a. eval()
b. test()
c. run()
d. exec()

A6: a

Q7: Which field of XMLHttpRequest will be populated with a JSON response?

a. responseXML
b. responseText
c. response
d. statusText

A7: b

Q8: Which of the following is not valid JSON?

```
a.  { "name": "Bob" }
b.  { "user": { " name": "Bob" } }
c.  { "name": "Bob", "getName": function() { return this.name } }
```

A8: c

Q9: The XMLHttpRequest object can be used to upload a file.

```
a.  True
b.  False
```

A9: b

Q10: The web browser displays a visual progress indicator while an XMLHttpRequest is being processed.

```
a.  True
b.  False
```

A10: b