FH Bielefeld
Fachbereich Technik
Studiengang Allgemeine Informatik

13. Juni 2013

Probeklausur Webbasierte Anwendungen

Sommersemester 2013

Kommentar:

- Auf dem Tisch befinden sich außer den ausgeteilten Klausurausdrucken nur ein Stift und der Studentenausweis. Es sind keine weiteren Hilfsmittel erlaubt (wie z.B. Notizenpapier, Taschenrechner, Handy, Lebensmittel, Formelsammlungen, Skripte, Lehrbücher). Alle nicht erlaubten Gegenstände sind in einer geschlossenen Tasche aufzubewahren. Taschen, die nicht abschließbar sind, werden beim Dozenten zur Aufbewahrung abgegeben.
- Unterhaltungen sind während der Klausur nicht erlaubt.
- Alle Zuwiderhandlungen werden als Betrugsversuch angesehen und die Klausur als nicht bestanden gewertet.
- Notizenseiten befinden sich am Ende der Klausur. Diese dürfen von Ihnen verwendet werden und sollen angeheftet bleiben. Die Rückseiten der Aufgabenblätter sollten Sie ebenfalls für Ihre Lösungen verwenden.
- Es sollen in den Antworten die Notationen und Definitionen verwendet werden, welche in der Lehrveranstaltung eingeführt worden sind. Eigene Begriffe und Notationen müssen vor dem Gebrauch eingeführt und definiert werden.

Vom Prüfling auszufüllen:

Name:

Vorname:

Matrikelnummer:

Unterschrift:

Vom Prüfer auszufüllen:

Identitätskontrolle: **Punkte Theorie: …. /13 Praxis: …./35**

Unterschrift: **Punkte gesamt:..../48 Note:**

.

Aufgabe 1 (1P)

Erläutern Sie den Begriff „Usability von Webanwendungen."

Lösung: Usability == Gebrauchstauglichkeit

- ☐ Wie ...
- ☐ einfach
- ☐ angenehm
- ☐ zielgerichtet

… kann ein Produkt eingesetzt werden?

- ☐ Kontext

- ☐ Mensch-Maschine-Interaktion (Human Computer Interaction

Aufgabe 2 (1P)

Was beschreibt der Begriff „WEB3.0"?

Lösung:

Semantic Web
Erkennungsalgorithmen für Sematische Bedeutungen innerhalb der Webseiten. -> Lernen möglich.
-> Computer beginnen aktiv zu helfen
Semantic web hilft den Computern die Bedeutung von Webpages zu verstehen.

Aufgabe 3 (1P)

Wie erfolgt die Übergabe der Daten vom Client zum Server mit dem HTTP-GET Befehl?

Lösung:
Im Querystring

Aufgabe 4 (1P)

Nennen Sie eine clientseitige Skriptsprache, die speziell für Webanwendungen entwickelt wurde und direkt in XHTML-Seiten eingebunden werden kann!

Lösung:
JavaScript

Aufgabe 5 (1P)

Welche Ressourcen von Webanwendungen sollten in Proxy-Caches oder Browser-Caches zwischengespeichert werden?

Lösung:
Ressourcen, die nur selten oder nur in bestimmten Abständen geändert werden, sollten gecached werden.

Aufgabe 6 (4P)

Beschreiben Sie je 2 Vorteile und 2 Nachteile der Nutzung von Cookies in Webanwendungen!

Lösung:

Vorteile der Nutzung von Cookies:

- *nicht an HTML gebunden*
- *Unabhängig von Dialoghistorie*
- *Bleiben je nach Konfiguration sehr lange erhalten*

➔ *Session kann nach längerer Zeit wieder aufgenommen werden*

Nachteile der Nutzung von Cookies:

- *Können vom User unterbunden werden*
- *Beschränkung in Größe und Anzahl pro Server*

Aufgabe 7 (1P)

Vergleichen Sie Javascript und Java anhand des Sprachkonzepts der Typisierung!

Lösung:

Javasvript hat schwache Typisierung

Java ist streng typiziert

Aufgabe 8 (2 P)

Nennen Sie die jeweiligen Methodennamen im DOM, für welche folgende Aussagen richtig sind:

a) fügt einen Knoten an das Ende des Dokumentenbaumes ein
b) kopiert einen Knoten

Lösung:

Methode	**Argument**	**Beschreibung**
appendChild	newNode	fügt Knoten an das Ende ein.
cloneNode		kopiert einen Knoten

Aufgabe 9 (1P)

Vergleichen Sie JavaScript und Java Applet anhand des Programmiersprachentyps.

Lösung:

JavaScript ist eine clientseitige Skriptsprache.
JavaApplet ist eine höhere objektorientierte Programmiersprache (Java)

Ende Theorieteil
Gesamt 13 Punkte

Aufgabe 10 (2P)

Zeichnen Sie in das oberhalb abgebildete Browserfenster die Ausgabe ein, die bei Aufruf des unten angegebenen Programms erfolgt!

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
   <title>JavaScript und DOM</title>
<script type="text/javascript">
//<![CDATA[
```

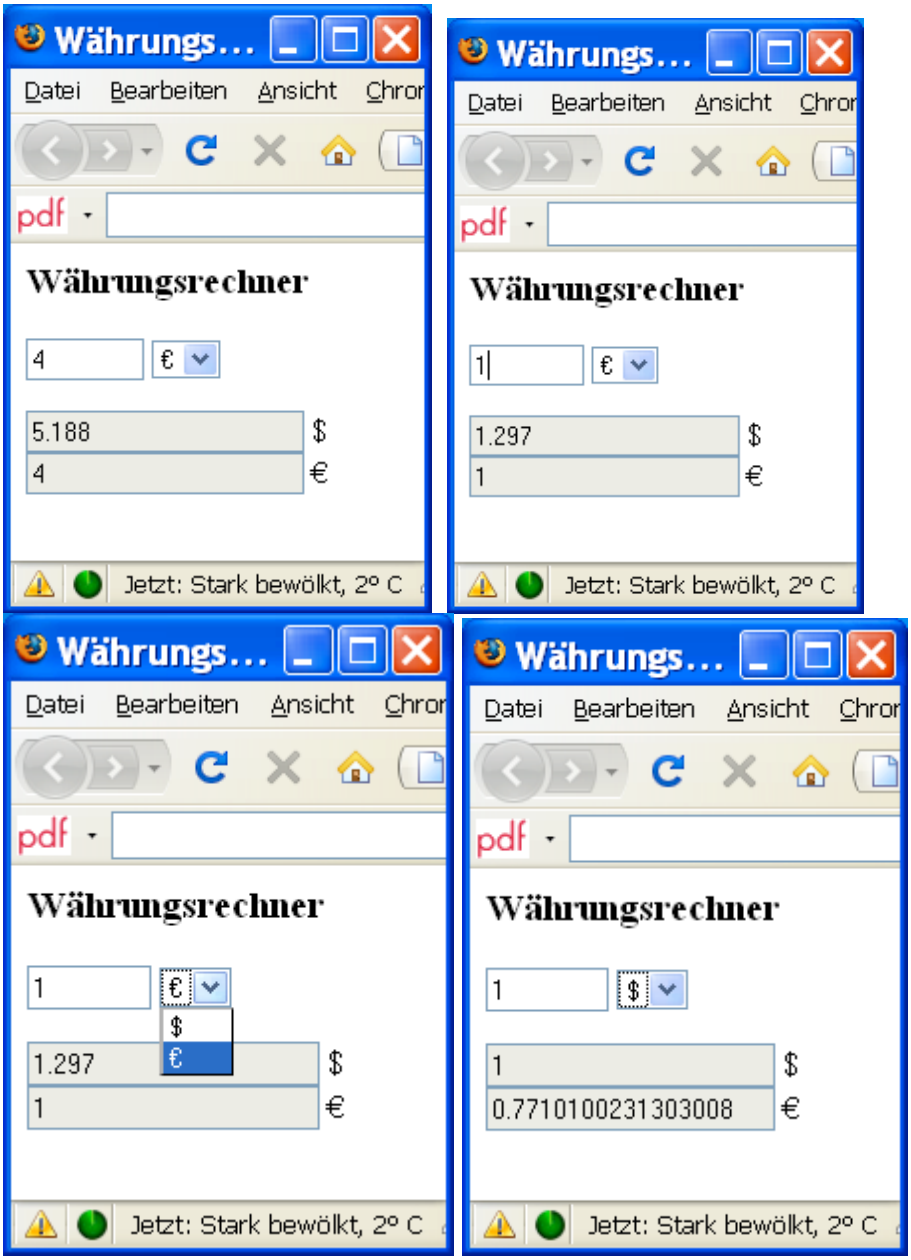
```
window.onload = function()
{
        var e = document.getElementById("e1");
        e.firstChild.nodeValue="Kein Umzug!";

}
//]]>
</script>
</head>
<body>
<h1 id="e1">Umzug der Informatik in den Neubau zum WS 2013/14</h1>
</body>
</html>
```

Lösung:

Aufgabe 11 (14P)

Schreiben Sie den Programmcode in **einem** HTML- Dokument unter zusätzlicher Verwendung von JavaScript im selben Dokument, um die nachfolgend dargestellten Web-Seiten mit einem einfachen Währungsrechner zur Umrechnung zwischen den beiden Währungen $ und € zu realisieren! Es sollen genau der im Bild gezeigte Inhalt sowie genau das darin gezeigte Design erzielt werden. Die Währungseinheiten $ und € sind über eine Drop-Down-Liste oben rechts auswählbar. Im linken oberen Feld wird der Wert eingegeben, für den die Umrechnung und Ergebnisausgabe immer genau dann erfolgen soll, wenn:
1. ein neuer Zahlenwert im oberen linken Eingabefeld eingegeben wird
2. im oberen rechten Feld eine neue Auswahl getroffen wurde.
Die unteren beiden Felder dienen nur der Ergebnisausgabe, dort können keine Werte eingegeben werden.



Lösung: (klausurGrA_Aufg18.html)

```
<html>                                          (1P, html-tag begin,ende)
<head>
 <title>W&auml;hrungsrechner (Stand 10. Februar 2009)</title>
 <script type="text/javascript">       (1P, script type Def. Javascript function im header)
  <!--
   function rechne()                  (1p funktionsdeklaration)
   {
    var menge = document.Waehrung.eingabewert.value;
    switch (document.Waehrung.einheit.options.selectedIndex)
    {
     case 0: faktor = 1.297;
             document.Waehrung.Dollar.value = menge;
             document.Waehrung.Euro.value = menge / faktor;
             break;
     case 1: faktor = 1.297;
```

```
            document.Waehrung.Dollar.value = menge * faktor;
            document.Waehrung.Euro.value = menge;
            break;
            }                                   (2P Realisierung der Umrechnung zwischen
                                    beiden Einheiten, je nach Auswahl in der Drop-Downliste)
    }
   //-->
  </script>                                     (1P tags ende script, head, )
</head>
<body>                                          (1P body begin, ende)
  <h3>W&auml;hrungsrechner</h3>                 (1P Überschrift oder fetter Druck, ggf mit css)
  <form name="Waehrung">                        (1P Form begin, ende)
    <input type="text" name="eingabewert" size="6" maxlength="5" onKeyUp="rechne()">
(1P input type text für wert, 1P onkeyup oder ähnlicher event mit funktionsaufruf der
javascriptfunktion im header)
    <select name="einheit" onChange="rechne()">          (1P select , 1P onChange mit
aufruf)
      <option value="Dollar">$</option>                  (1P Option Dollar und Euro )
      <option value="Euro">€</option>
    </select>
    <p><input type="text" name="Dollar" readonly> $   (1P – readonly für beide )
    <br><input type="text" name="Euro" readonly> €
  </form>
</body>
</html>
```

Aufgabe 12 (9P)

Schreiben Sie ein Servlet, welches die Cookies aus dem Client ausliest, löscht und auf der Webpage

1) alle gelöschten Cookies durchnummeriert sowie
2) den Namen und
3) den Wert dieser Cookies anzeigt!
   (siehe auch die untere Abbildung)

Füllen Sie die mit „……." bezeichneten Programmstellen aus und benutzen Sie den nachfolgend angegebenen Rahmen für Ihre Implementierungen!

```
import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class CookieLesen extends …… {
protected void ……...(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      response…….. ("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
```

```
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Servlet Cookies Loeschen</h1>");
        Cookie[] c = ……..;
           if(c != null)
              for (int i = 0; i < c.length; i++){
                 Cookie cookie = new Cookie(c[i].getName(), c[i].getValue());









              }
        out.println("</body>");
        out.println("</html>");
        out.close();
    }
}
```

Lösung:

```
import java.io.*;
import java.net.*;
```

```
import javax.servlet.*;
import javax.servlet.http.*;

public class CookieLesen extends HttpServlet {                                    (1P)
protected void doGet(HttpServletRequest request, HttpServletResponse response)    (1P)
   throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");        (1P)
      PrintWriter out = response.getWriter();
      out.println("<html>");
      out.println("<body>");
      out.println("<h1>Servlet Cookies Loeschen</h1>");
      Cookie[] c = request.getCookies();                         (1P)
         if(c != null)
            for (int i = 0; i < c.length; i++){
               Cookie cookie = new Cookie(c[i].getName(), c[i].getValue());
               cookie.setMaxAge(0);                              (1P)
               response.addCookie(cookie);                       (1P)
               out.println("<b>Cookie Nr. " + (i+1) +"</b></br>");  (1P)
               out.println("Name: " + c[i].getName()+"</br>");      (1P)
               out.println("Value: "+ c[i].getValue()+"</br>");     (1P)
            }
      out.println("</body>");
      out.println("</html>");
      out.close();
   }
}
```

Aufgabe 13 (4P)

In einer Webapplikation steht Ihnen eine Klasse *Customer* zur Verfügung. Der komplette Sourcecode der Klasse *Customer* lautet folgendermaßen:

```
public class Customer {
   private String user     = new String();
   private String password = new String();

   public Customer(){}
   public Customer( String user, String password){
      this.user = user;
      this.password = password;
   }

   public String getUser(){return user;}
   public void setUser( String user){
      this.user = user;
   }
   public String getPassword() {return password;}
   public void setPassword( String password){
      this.password = password;
   }
}
```

Füllen Sie nun die mit „**……..**“ gekennzeichneten Stellen in einem Ausschnitt aus einer jsp-Seite richtig aus, um die jeweiligen aktuellen Werte für costumer und Passwort auf der Webseite auszugeben!

…

```
<h1>Ihre Anmeldedaten</h1>
   <jsp:useBean id="customer" class="shop.book.data.Customer"/>
    <!– alle Properties setzen→
   <jsp:…… name="……..." property="*" />
   <br>
   <table>
   <tr><td>Benutzername:</td><td><……..></td></tr>
   <tr><td>Passwort:</td><td><……..></td></tr>
   </table>
…
```

Lösung:

```
<h1>Ihre Anmeldedaten</h1>
   <jsp:useBean id="customer" class="shop.book.data.Customer"/>
    <!– alle Properties setzen→
   <jsp:setProperty name="customer" property="*" />                    (1P)  (1P)
   <br>
   <table>
   <tr><td>Benutzername:</td><td><%=customer.getUser()%></td></tr>   (1P)
   <tr><td>Passwort:</td><td><%=customer.getPassword()%></td></tr>    (1P)
   </table>
```

Aufgabe 14 (6P)

Ein Applet überschreibt seine init() – Methode wie im nachfolgenden Codeausschnitt angedeutet:

```
public class Ueb_Add extends Applet implements ActionListener
{

int parameter2;
int parameter3;
int result = 0;
Button addButton, mulButton;

    public void init()
    {
        Parameter1 = Integer.parseInt(getParameter("param1"));
        Parameter2 = Integer.parseInt(getParameter("param2"));
….
```

Schreiben Sie eine HTML-Seite, die dieses Applet einbindet und die geforderten Eingangwerte übergibt!

Lösung:

```
<html>                                                  (1P html begin ende)
<body>
<APPLET CODE="Ueb_Add" codebase="./build/classes"       (1P Applet oder object tag)
WIDTH=300 HEIGHT=240>
 <param name="param1" value="13">                       (2P)
 <param name="param2" value="2">                        (2P)
 </APPLET>
  </body>
</html>
```

Punktevergabe:

1	**46**
1,3	**43**
1,7	**41**
2	**39**
2,3	**36**
2,7	**34**
3	**31**
3,3	**29**
3,7	**26**
4	**24**
5	

Notizenseite: