

Lernziele

- Erste(?) Begegnung mit Computersprachen
- Kennenlernen von Grundkonzepten wie Syntax und Verschachtelung
- Erfahrung mit der "Pingeligkeit" des Rechners bzgl. syntaktischer Korrektheit

Aufgabe 1) Einführung anhand der Anwendung von einfachen Tags

Verwenden Sie einen Editor Ihrer Wahl, z.B. Notepad(!), um den Text in der Datei beispieltext.txt wie unten angezeigt zu formatieren.

Reading Programs

Some years ago, when COBOL was the great white programming hope, one heard much talk about the possibility of executives being able to read programs. With the perspective of time, we can see that this claim was merely intended to attract the funds of executives who hoped to free themselves from bondage to their programmers. Nobody can seriously have believed that executives could read programs. Why should they? Even programmers do not read programs.

But isn't it quite proper that only the machine should read programs? Weren't the programs written for the machine? Yes and no. Even if we were not concerned with program modification and with the interfaces between programs, reading programs might not be such a bad idea from the point of view of learning about programming.

Programming is, among other things, a kind of writing. One way to learn writing is to write, but in all other forms of writing, one also reads. We read examples - both good and bad - to facilitate learning. But how many programmers learn to write programs by reading programs? A few, but not many [...] In the old days - which in computing is not so long ago - we had less easy access to machines and couldn't afford to wait for learning from actual machine runs. Turnaround was often so bad that the programmers would while away the time by reading each others' programs. Some even went so far as to read programs from the program library - which in those days was still a library in the old sense of the term.

But, alas, times change [...] Late at night, when the old timer is curled up in bed with a *sexy subroutine* or a *mystifying macro*, the young blade is busily engaged in a dialogue with his terminal.

Error and Ego

Programming - perhaps more than any other profession - is an **individual activity** [...] What difference can it make how many other programmers you run into during the day. If asked, most programmers would probably say they preferred to work alone in a place where they wouldn't be disturbed by other people.

Well, what is wrong with "owning" programs? Artists "own" paintings; authors "own" books; architects "own" buildings. Don't these attributions lead to admiration and emulation of good workers by lesser ones? Isn't it useful to have an author's name on a book so we have a better idea of what to expect when we read it? And wouldn't the same apply to programs? Perhaps it would - if people read programs, but we know that they do not. Thus, the admiration of individual programmers cannot lead to an emulation of their work, but only to an affectation of their mannerisms. This is the same phenomenon we see in "art colonies" where everyone knows how to look like an artist, but few, if any, know how to paint like one.

Egoless Programming

What is to be done about the problem of the ego in programming? A typical text on management would say that the manager should exhort all his programmers to redouble their efforts to find their errors. Perhaps he would go around asking them to show him their errors each day. This method, however, would fail by going precisely in the opposite direction to what our knowledge of psychology would dictate, for the average person is going to view such an investigation as a personal trial. Besides, not all programmers have managers - or managers who would know an error even if they saw one outlined in red.

from Gerald M. Weinberg, "The Psychology of Computer Programming"

Teilaufgabe a) Rahmenwerk

Öffnen Sie die Datei "beispieltext.txt" im Webbrowser. Erzeugen Sie nun eine neue Datei mit dem Namen "beispieltext.html" und fügen Sie in diese Datei den folgenden HTML-Rahmen ein.

```
<html>
<head>
<title>Beispieltext</title>
</head>

<body>
</body>
</html>
```

Kopieren Sie anschließend den Beispieltext(=Seiteninhalt) zwischen die Tags <body> und </body>. Speichern Sie die Datei und öffnen Sie sie im Webbrowser.

Welche Veränderung stellen Sie an der Darstellung des Textes, z.B. am Umbruch fest? Wie wird der Text dargestellt, wenn Sie die Datei wieder als ".txt" speichern?

Teilaufgabe b) Formatierung der Absatzstruktur

Verwenden Sie die Tags <h1> </h1> um die Überschriften zu formatieren sowie <p> um die Absätze zu definieren. Das schließende Tag </p> ist optional, gehört aber zum guten Stil. Fügen Sie zwischen den Absätzen ggf. ausreichend Leerraum (Whitespace) ein, um die Lesbarkeit des HTML-Quellcodes zu erhöhen. Öffnen Sie anschließend das Ergebnis

im Webbrowser. Variieren Sie die Größe (Breite) des Browserfenster und beachten Sie den dynamischen Umbruch der Absätze. Wie reagiert der Browser auf Fehler wie fehlerhafte Schreibweise von Tag-Namen (z.B. <g1> statt <h1>) oder das Vergessen von schließenden Tags (z.B. <h1> ohne nachfolgendes </h1>)?

Teilaufgabe c) Formatierung innerhalb der Absätze

Verwenden Sie die Tags , <i> </i> und <u> </u> zur Formatierung der fetten, kursiven und unterstrichenen Textteile. Verwenden Sie außerdem das Tag um die Anfangsbuchstaben in jedem Absatz etwas zu vergrößern.

Für die Quellenangabe können Sie ebenfalls die Tags <i> und verwenden. Um den Text rechtsbündig zu formatieren fügen Sie in das <p>-Tag das Attribut align="right" ein.

Schauen Sie sich das Resultat anschließend im Webbrowser an.

Teilaufgabe d) Experimente mit weiteren Tags

Experimentieren Sie mit weiteren Tags wie <h2>, <h3>,
 oder , , <blink> sowie deren Attributen wie <p align="left|center|right"> oder . Versuchen Sie außerdem, bereits bekannte Tags durch Verschachtelung zu kombinieren (z.B. Fett und Kursiv). Was passiert, wenn Sie das "äußere" Tag vor dem "inneren" schließen. Halten Sie diese Vorgehensweise für eine gute Idee?

Aufgabe 2) Umgang mit Fehlern

Teilaufgabe a) Fehlende Tags

Entfernen Sie aus Ihrer Datei einige öffende und schließende Tags. Wie verändert sich dadurch das Aussehen der Seite?

Teilaufgabe b) Unbekannte Tags

Fügen Sie in die Datei "beispieltext.html" einige Fantasie-Tags ein oder verwenden Sie die Zeichen "<" und ">" um Teile des Texts "auszuknipsen". Was passiert, wenn Sie nur "<" verwenden (also nicht mehr schließen).

Hinweis: Ausknipsen von Teilen eines Dokuments (oder Programms) kann manchmal auch gewollt sein, z.B. bei der Fehlersuche. Jedoch sollten dafür Kommentare verwendet werden ("auskommentieren"). In HTML

```
<!-- das ist ein Kommentar -->
```

Daneben sollten Kommentare auch zur Kommentierung verwendet werden ;-)

Triviale Folgerungen

- Der Rechner hat wenig bis kein Verständnis für "geringfügig falsche Syntax".
- Wenn man etwas ändert und nicht speichert, sieht man die Änderung nicht.

Aufgabe 3) Darstellung von Sonderzeichen in HTML

Die Tags in HTML haben einen einheitlichen Aufbau. Dieser wird vom Browser verwendet, um Tags von den restlichen Inhalten der Seite unterscheiden zu können. Konkret bedeutet dies: Alles was in "< >" steht bzw. mit "<" anfängt, wird vom Browser als Tag interpretiert, z.B. "<hallo>" aber auch "Aus a < 4 folgt...". Um die verschiedenen Versionen von HTML verarbeiten zu können, die im Web eingesetzt werden, ignoriert (d.h. verschluckt) der Browser unbekannte Tags.

Um spezielle Zeichen (insbesondere "<" und ">" aber auch Sonderzeichen wie die deutschen Umlaute) trotzdem in einem HTML Dokument verwenden zu können, gibt es für diese eine spezielle Schreibweise. Diese folgt dem Aufbau "&xx;" wobei xx das fragliche Zeichen identifiziert. Solche kryptischen Schreibweisen werden als "Fluchtsequenzen" (Escape-Sequences) bezeichnet – man kurzzeitig "flüchtet" aus der normalen Verarbeitung. Dieses Prinzip findet auch in anderen Bereichen der Informatik Anwendung. Grund ist nicht immer die besondere Bedeutung von Zeichen sondern auch die Nichtdruckbarkeit bestimmter Zeichen im jeweiligen Zeichensatz.

Beispiele

<	<	"less than"
>	>	"greater than"
ä	ä	"a-UMLaut"
Ä	Ä	"A-UMLaut"
ß	ß	"SZ-Ligatur?"

Teilaufgabe a) Anwendung

Fügen Sie nun einige der oben gezeigten Sonderzeichen ein (andere Umlaute analog zu ä/Ä). Achtung: In Europa funktionieren Umlaute ggf. auch, wenn man Sie direkt im HTML-Text verwendet. Diese Lösung ist jedoch nicht portabel – die Fluchtsequenzen schon. Wie reagiert der Browser auf unbekannte Fluchtsequenzen?

Aufgabe 4) Bilder in HTML

Zur Integration von Bildern in ein HTML Dokument werden diese nicht (wie z.B. in Word) direkt eingebettet sondern über das -Tag referenziert. Dabei besteht neben der Skalierung des Bilds auf eine vorgegebene Höhe und Breite auch die Möglichkeit, einen beschreibenden Text anzugeben. Dieser wird angezeigt, wenn der Anwender die Anzeige von Bildern deaktiviert hat (z.B. um Bandbreite zu sparen) oder gar keine Bilder sehen kann (z.B. blinde Anwender oder Anwender mit textorientierten Browsern). Im Interesse der Zugänglichkeit (Accessibility) sollte diese Möglichkeit stets genutzt werden.

Hinweis: Für behördliche Seiten existiert eine ganze Reihe von Vorschriften zur Verbesserung der Zugänglichkeit von Webseiten.

Teilaufgabe a) Bild einfügen und zentrieren

Fügen Sie mit Hilfe des ``-Tags das Bild "it-job-icon.gif" unten in die Datei ein und zentrieren Sie es durch einen entsprechend ausgerichteten Absatz (`<p>`-Tag) oder Bereich (`<div>`-Tag).

Aufgabe 5) Listen in HTML

Bei dieser Aufgabe sollen einfache und verschachtelte Listen in HTML erzeugt werden.

Teilaufgabe a) Einfache Liste

Erzeugen Sie eine neue Datei mit dem Namen "listen.html" und fügen Sie in diese den HTML Rahmen ein. Definieren Sie anschließend mit Hilfe der Tags `` `` und `` folgende Liste :

- Audi
- Mercedes-Benz
- Volkswagen

Fügen Sie nach eigenem Ermessen eine Überschrift und weitere Elemente oder Formatierungen ein und betrachten Sie anschließend das Ergebnis im Webbrowser.

Teilaufgabe b) Numerierte Liste

Ändern Sie die Tags `` `` in `` `` und betrachten Sie anschließend das Ergebnis im Webbrowser

Teilaufgabe c) Verschachtelte Listen

Nun sollen die Automarken um eine exemplarische Aufzählung von Modellen ergänzt werden. Fügen Sie dazu jeweils nach dem ``-Tag mit der Markenbezeichnung eine ungeordnete Liste mit den Modellnamen an. Die fertige Liste soll dann wie folgt aussehen:

- 1.Audi
 - A8
 - A6
 - A4
 - A3
- 2.Mercedes-Benz
 - S-Klasse
 - E-Klasse
 - C-Klasse
 - A-Klasse
- 3.Volkswagen
 - Passat
 - Golf
 - Polo

Sorgen Sie für eine ausreichende Lesbarkeit des HTML-Quellcodes, indem Sie die inneren Listen auf einer neuen Zeile beginnen und analog der späteren Anzeige etwas einrücken:

Beispiel:

```
<ol>
<li>Obst
  <ul>
    <li>Apfel
    <li>Birne
  </ul>
<li>Gemüse
  <ul>
    <li>Tomate
    <li>Paprika
  </ul>
</ol>
```

Überzeugen Sie sich im Webbrowser von der Korrektheit ihrer Seite.

Teilaufgabe d) Weitere Experimente

Fügen Sie weitere Elemente und/oder Ebenen ein oder formatieren Sie die Elemente zusätzlich mit Hilfe anderer Tags. Anregung: Modellbezeichnung fett und dannach ein Zeilenumbruch mit `
` gefolgt von beschreibendem Text ("Ein tolles Auto").

Triviale Folgerungen:

- Verschachtelung funktioniert immer wie "russische Puppen", nie über Kreuz
- Leerraum und Einrückung vereinfachen die Lesbarkeit (und die Fehlersuche)

Aufgabe 6) Tabellen in HTML

Bei dieser Aufgabe sollen einfache und verschachtelte Tabellen erstellt werden. Dabei werden Sie die Schwierigkeit kennenlernen, eine an sich 2-dimensionale Struktur in eine 1-dimensionale Darstellung zu überführen (zu "serialisieren"). Dieses Problem trat bei Listen noch nicht auf.

Hinweis: Der Rechner legt auch die 2- oder mehrdimensionalen Datenstrukturen, die wir später kennenlernen werden, 1-dimensional im linear organisierten Speicher ab (die Speicherzellen sind genau wie die Zeilen im Quelltext einfach durchnummeriert).

Teilaufgabe a) Eine einfache Tabelle

Erzeugen Sie eine neue Datei mit dem Namen "tabellen.html", fügen Sie den HTML-Rahmen ein und erzeugen Sie anschließend folgende einfache Tabelle mit Hilfe der Tags `<table>` `</table>` (Tabellenrahmen), `<tr>` `</tr>` (Tabellenzeile) und `<th>` `</th>` (Tabellenüberschrift) bzw. `<td>` `</td>` (Tabellenzelle).

Die Breite der Tabelle und die Anzeige (oder Unterdrückung) der Umrandung kann mit den Attributen `width="100%"` und `border="0"` des `<table>`-Tags gesteuert werden.

Sorgen Sie ggf. durch entsprechende Einrückung von `<table>`, `<tr>` und `<th>/<td>` für eine klare optische Gliederung des Quellcodes.

<i>Name</i>	<i>Genre</i>	<i>Hersteller</i>	<i>Plattform</i>
Pac Man	Maze	Midway	Arcade
Jumpman	Jump and Run	Epyx	C64
Doom	3d Shooter	id Software	PC
Warcraft3	Realtime Strategy	Blizzard	PC

Teilaufgabe b) Verschachtelte Tabellen

Tabellen (ohne Rand) können auch für Layout Zwecke eingesetzt werden. Dazu ist es manchmal nützlich, Zellen innerhalb einer Zeile mit Hilfe des Attributs `colspan="n"` (die nächsten n-Zellen zusammenfassen) zu verbinden. Eine Verbindung von Tabellen innerhalb einer Spalte ist nicht ohne weiteres möglich. Um dies zu erreichen, müssen mehrere Tabellen verschachtelt werden, d.h. eine oder mehrere Tabellenzellen der "äußeren" Tabelle enthalten weitere "innere" Tabellen. Versuchen Sie folgende Struktur durch verschachtelte Tabellen und Einsatz des `colspan`-Attributs nachzubilden.

Gehen Sie dafür zunächst von einer 1x2 Tabelle aus (Zeilen x Spalten) und fügen Sie in die rechte Spalte eine 3x2 Tabelle ein bei der Sie die Zellen der mittleren Zeile zusammenfassen.

Manche Strukturen lassen sich jedoch auf diese Weise nicht nachbilden (oder doch?)

Beispiel:

Einige der Layout-Manager die wir später bei der GUI Programmierung kennenlernen werden (`GridLayout` und `GridBagLayout`) basieren auf dem gleichen Prinzip. `GridBagLayout` bietet auch ohne Verschachtelung die Möglichkeit zur Zusammenfassung von Zellen einer Zeile oder Spalte und bietet damit mehr Flexibilität (um den Preis von mehr Komplexität).

Aufgabe 7) Frames

Viele Webseiten enthalten wiederkehrende Elemente wie z.B. eine Kopfzeile oder eine Navigationsleiste. Um diese nicht auf jeder Webseite erneut implementieren (und bei Änderungen zig-mal anpassen zu müssen) besteht die Möglichkeit, mit Hilfe von sog. Frames mehrere Webseiten zu einer Gesamtansicht im Browserfenster zu verbinden.

Hinweis: Das Prinzip, Redundanz zu vermeiden und Bestandteile, die man mehrfach benötigt, möglichst nur einmal zu realisieren und wiederzuverwenden, zieht sich durch alle Bereiche der Informatik. Die Anwendung dieses Prinzips erspart nicht nur Realisierungs- und Testaufwand (durch Wiederverwendung bewährter Komponenten) sondern vermeidet außerdem Fehler bei Änderungen (man muss nur eine Stelle ändern, kann also keine vergessen). In Anbetracht der Komplexität und Langlebigkeit heutiger Systeme ein wichtiger Aspekt. Allerdings muss der einmalige Mehraufwand für die Erreichung der Wiederverwendbarkeit in jedem Einzelfall gegen das Wiederverwendungspotential und die Lebensdauer des Ergebnisses abgewogen werden.

Frames funktionieren dabei im Grunde ähnlich wie Tabellen, da Sie das Browserfenster in Zellen einteilen und ebenso wie Tabellen verschachtelt werden können. Allerdings kann jedes sog. Frameset das Browserfenster nur entweder in Spalten oder Zeilen unterteilen. Damit ist die Verschachtelung bei Frames eher die Regel als die Ausnahme.

Teilaufgabe a) Ein einfaches Frameset

Frames sind gewissermaßen ein eigener "Dokumententyp". Dies äußert sich darin, dass eine HTML-Dokument nur Frames ODER Informationen enthalten kann (wenn man die Möglichkeit von inneren Frames über <iframe> einmal kurz außen vorlässt. Diese erlauben die Einbettung eines Frames irgendwo in einer Webseite – ähnlich wie eine Grafik mit). Erzeugen Sie eine neue Datei mit dem Namen "frames.html" und folgendem Inhalt:

```
<frameset cols="200,*">  
  <frame src="listen.html">  
  <frame src="beispieltext.html">  
</frameset>
```

Dieser Code definiert ein Spalten-Frameset (engl. column=Spalte, Zeilen über rows="") bei dem die erste Spalte 200 Pixel breit ist und die zweite Spalte den Rest (*) des Platzes einnimmt. In der ersten Spalte wird die vorher erstellte Listenstruktur angezeigt, die zweite Spalte enthält den Beispielttext.

Teilaufgabe b) Ein Frameset für Bookmarks

In dieser Teilaufgabe soll die linke Spalte zu einer Navigationsleiste ausgebaut werden. Dazu erzeugen wir zunächst zwei neue HTML-Dateien mit den Namen "lesezeichen.html" und "startseite.html". Die beiden Dateien sollen zunächst nur einen HTML-Rahmen mit dem Inhalt "Navigation" und "Startseite" enthalten. Um die beiden Dateien in das Frameset einzubinden gehen Sie wie folgt vor:

Tragen Sie den Dateinamen "lesezeichen.html" in das src-Attribut des ersten Frames ein.
Tragen Sie den Dateinamen "startseite.html" in das src-Attribut des zweiten Frames ein.
Fügen Sie beim zweiten Frame das Attribut name="inhalt" hinzu.

Öffnen Sie anschließend zur Erfolgskontrolle das Frameset im Browser.

Teilaufgabe c) Einbau von Links zur Navigation

Nun sollen in die Seite "lesezeichen.html" Links zur Navigation eingebaut werden. Dabei ist zu beachten, dass die verlinkten Seiten nicht im Navigations-Frame sondern im Inhalts-Frame geöffnet werden sollen. Dies kann mit dem target-Attribut des <a>-Tags gesteuert werden. Tragen Sie dazu Links zu verschiedenen Webseiten (Fachhochschule, Google, ...) in folgendem Format in die Navigationsseite ein:

```
<a href="http://www.meineliebblingsseite.de" target="inhalt">MeineLieblingsseite</a>
```

Der Wert des target-Attributs muss genau dem Wert des name-Attributs des zweiten Frames entsprechen. Zur Formatierung der Links können Sie nach eigenem Ermessen Zwischenüberschriften (<h1> - <h3>, z.B. "Suchmaschinen", "Nachrichten", ...) und Umbrüche (<p>,
 oder aber /) einfügen. Ein letzter Test und Voila!...

Aufgabe 8) Stylesheets

Diese Aufgabe wird als interaktives Tutorial durchgeführt