Input - Output in Javascript

Jobst-Hartmut Lüddecke

3. April 2013

Zusammenfassung

Diese Lektion des 2. Semesters bringt einen schnellen Einstieg in die Programmierung mit *Javascript*. Die einzelnen Teile sind auf die wesentlichsten Bestandteile bewusst beschränkt um gleich ein *kleines Erfolgserlebnis* zu ermöglichen.

Es gibt also ein kleines *hello world* Skript, mit einem *Kochrezept*. Dann einen kleinen Exkurs in Sachen *Variablen* und *Operatoren* und schließlich ein paar *Alert-Boxen* als einfache Möglichkeit der Ein- und Ausgabe.

Diese Lektion ist in Hinblick auf *HTML 5* überarbeitet worden, da das *Script* sich jetzt in einer eigenen Datei befinden soll.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Prinzip eines Javascripts

```
<!DOCTYPE html>
<html lang="de" xml:lang="de">
<head>
<title>Minimal-Beispiel Hallo mit Javascript</title>
<script language="javascript" src="hallo.js">
</script>
</head>
<body onload=hallo()>
</body>
</html>
```

Abbildung 1: Beispiel einer HTML5-Seite in der das Javascript *hallo.js* ausgeführt wird

```
function hallo(){
    document.write("Hallo!");
    document.write("<br>");
}
```

Abbildung 2: Beispiel des Javascripts *hallo.js*

1.1 Kochrezept

- Die ersten beiden Zeilen der HTML-Seite legen fest, dass es sich um eine *HTML5*-Seite handelt.
- In den **Kopf der HTML-Seite** wird ein Link auf ein *Script* eingefügt. Also als *Optionen* des *Tags* **script** gibt es hier einmal *language* mit der die Scriptsprache (hier Javascript) festgelegt wird und als zweite Option *src* in der der Link auf die Datei mit dem Script steht.
- In der Datei *hallo.js* erfolgt die Definition eines Programmteils. Dies sind *Funktionen*, die einen *Rückgabewert* und *Übergabewerte* haben. Die *Übergabewerte* die **an die Funktion** übergeben werden stehen in *runden Klammern* hinter der *Funktion*. Wenn keine Werte an die Funktion übergeben werden, müssen trotzdem die runden Klammern gesetzt werden, der Inhalt zwischen den Klammern bleibt dann leer.

- Nach der *Funktion* mit ihrem *Namen* und den *Übergabeparametern* in *runden Klammern* steht das eigentliche Programm dieser Funktion in *geschweiften Klammern*. Achten Sie unbedingt immer darauf **den jeweils richtigen** *Klammer-Typ* zu verwenden. So ein Fehler wird leicht übersehen und Sie verbringen viel Zeit mit der Fehlersuche!
- Achten Sie darauf, dass zwischen *Funktionsnamen* und *runder Klammer auf* **keine Leerzeichen und keine Zeilenumbrüche stehen**. Analog können Leerzeichen und Zeilenumbrüche zwischen *runder Klammer zu* und *geschweifter Klammer auf* Probleme bereiten, denn Leerzeichen und Zeilenumbrüche werden als **Trennsymbol** interpretiert. Damit wird der Inhalt der Klammern dann nicht mehr als zur Funktion zugehörig interpretiert. Dies kann sein, muss aber nicht. Unterschiedliche Implementationen von Javascript sind unterschiedlich fehlertolerant. Auf der sicheren Seite liegen Sie, wenn Sie **keine Leerzeichen** dazwischen machen.
- Funktionen können andere Funktionen aufrufen, die **vorher definiert** worden sind, also weiter oben im Skript programmiert worden sind.
- Dann wird irgendwo die *Funktion* aufgerufen. Dafür gibt es mehrere Möglichkeiten auf die ich später eingehen werde. **In diesem Fall** wird die Funktion beim *Laden der HTML-Seite* aufgerufen, was im `<body>`-Tag mit der *Option* `onload` geschieht. Welche Funktion geladen und ausgeführt werden soll steht nach dem Gleichheitszeichen. Also `<body onload=hallo()>` bedeutet, dass beim Laden der der HTML-Seite die *Funktion* mit dem Namen `hallo()` ohne Übergabeparameter (leere runde Klammern) aufgerufen und damit beim Start der HTML-Seite ausgeführt wird.
- Funktionen können *Rückgabewerte* haben. Diese kann man direkt einer Variablen zuweisen. Dazu später mehr.

2 document.write

Diese einfachste Art der Ausgabe haben wir bereits genutzt. Hiermit wird ganz einfach **dynamisch HTML-Text** in das Browser-Fenster geschrieben. Dabei ist zu achten, dass für die Formatierungen HTML-Tags als Text, also in Anführungsstrichen, auszugeben sind.

Beispiel: Ein Zeilenumbruch bewirkt man mit der Ausgabe eines HTML-Tags `"<br>"` (in diesem Fall in Anführungszeichen) und nicht mit einem `\n`, wie es in Javascript-Texten üblich ist(s.u.).

3 Kurzer Einstieg zum Thema *Variablen*

Salopp gesagt, die *Variable* ist eine *Schublade* in die man etwas hineintun kann. Mit dem *Namen* spricht man die *Variable* an und bezeichnet damit die Schublade. Der *Inhalt der Variablen* ist der *Wert* und in unserem Beispiel der *Inhalt der Schublade*.

Zunächst muss die *Variable* dem Programm bekannt gemacht werden und ein entsprechender *Speicherplatz* reserviert werden. Dies nennt man die *Deklaration* der Variablen. Man macht es in *Javascript* mit dem *Befehl*

```
var variablenname;
```

Technisch gesehen wird damit ein Speicherplatz *allokiert*[1] und die *Adresse* dieses Speicherplatzes auf den *Variablennamen* abgebildet. Der *Inhalt* der *Variablen* ist dann der *Wert*, der in diesem Speicherplatz unter dieser Speicher-Adresse abgelegt ist.

[1] Ein festgelegter Platz im Arbeitsspeicher (RAM) reserviert und mit dem Variablen-Namen als Adresse angesprochen.

Wurde die *Variable* nicht *initialisiert*, also der *Variablen* kein *Anfangswert* zugewiesen, dann ist der *Wert* der *Variablen* solange nicht definiert, bis wir der Variablen einen Wert zugewiesen haben. Solange dies nicht geschehen ist, ist der Wert der Variablen (Inhalt der Schublade) also ungewiss, praktisch eine *Wundertüte*.

Der *Wert* der *Variablen* kann verschiedenen *Typs* sein. In anderen Programmiersprachen muss man diesen Typ gleich bei der Deklaration mit angeben. In Javascript wird dies sehr locker gesehen und erst bei der *Zuweisung* eines *Wertes* eines bestimmten *Typs* wird der *Typ der Variablen* festgelegt.

Doch **Vorsicht:** Hier ist der Programmierer gefordert **Ordnung zu halten**. Es findet von *Javascript* **keine Überprüfung des Typs** – wie z.B. in Programmiersprachen wie *Pascal* – statt. Damit muss der Programmierer selbst dafür sorgen, dass er keine *Äpfel mit Birnen multipliziert*.

Für uns sind zunächst folgende 3 Typen wichtig:

3.1 Numerische Variable

Eine *Zahl* repräsentiert eine interne Darstellung, mit der gerechnet werden kann. Bitte erinnern Sie sich an die Veranstaltung mit *Integer-Zahlen*, *Real-Zahlen Worten* usw. In *Javascript* ganz einfach mit:

```
integerzahl = 1234;
realzahl = 3.141592654;
```

und schon hat die vorher deklarierte Variable *integerzahl* den ganzzahligen *Wert* **1234** *zugewiesen* bekommen und ist somit vom *Typ* eine *Integer-Zahl* mit der man auch rechnen kann. Entsprechend hat jetzt die Variable *realzahl* den Wert π in einer internen Darstellung als *Gleitpunktzahl* mit Exponenten (real).

3.2 Text-Variable

Text ist eine Kette von einzelnen Zeichen (Character) und wird daher auch als *Zeichenkette*, oder als *String* bezeichnet.

Damit der Computer auch erkennt, wo diese Zeichenkette anfängt und wo sie aufhört, gibt es *Begrenzungszeichen* oder auch *Delimiter*. In *Javascript* ist dies – wie auch in vielen anderen Programmiersprachen – das amerikanische *Anführungszeichen*. Schreibe ich nun folgenden Befehl:

```
text = "dies ist eine Zeichenkette";
```

Dann habe ich die Variable *text* zu einer *Textvariablen* gemacht und ihr den *Wert* **dies ist eine Zeichenkette** zugewiesen. Dies ist also eine Folge von 26 einzelner Zeichen, bei der selbstverständlich die Leerzeichen mitzählen, denn sie sind gültige Zeichen in dieser Kette, die ihren eigenen Platz einnehmen.

Nicht mitgezählt und nicht in der Variablen zugewiesen sind die Anführungsstriche (Begrenzungszeichen).

3.3 Boolesche Variable

Die *boolesche* Variable kennt nur die zwei Werte *true* und *false*. Dies bedeutet *true*, also *wahr* wenn eine Bedingung erfüllt ist und *false*, also *falsch*, wenn diese Bedingung nicht erfüllt ist. Brauchen werden wir solche Variablen noch in der Lektion über *if* und *switch* um *Fallunterscheidungen* machen zu können.

```
bool = true;
```

3.4 Deklaration und Zuweisung eines Startwertes

Um etwas Schreibarbeit sparen zu können, gibt es auch die Möglichkeit die *Deklaration* der Variablen und die *Initialisierung*, also die *Zuweisung* eines Wertes in einem Befehl zu machen. Dabei handelt es sich um *Anfangswerte*.

Bei späteren neuen *Zuweisungen* ist das *var* der *Deklaration* wegzulassen. Schreibt man an späterer Stelle noch einmal ein *var* mit dem gleichen Namen, wird eine **weitere** *Variable* mit *gleichem Namen*, aber anderer Speicheradresse erzeugt. *Javascript* merkt es nicht unbedingt, aber sie haben Probleme in Ihrem Programm.

```
var zahl = 1234;
var bool = false;
```

4 Operatoren

Hier kommen *elementare Operatoren*. Auf *spezielle Notationen* (für schreibfaule Mitbürger) und *Bit-Operatoren* gehe ich an dieser Stelle nicht ein.

4.1 Arithmetische Operatoren

Dies sind Operatoren für Rechenoperationen mit Zahlen und numerischen Variablen.

+ Addition.

– Subtraktion.

***** Multiplikation.

/ Division.

% Modulo, Rest der ganzzahligen Division.

4.2 Operatoren für Zeichenketten

Operatoren zur Behandlung von *Strings*. Hier wird das **+** zur Zusammensetzung von mehren Texten gebraucht.

+ Zusammensetzung von *Strings* mit anderen Zeichenketten und/oder Inhalten von Textvariablen.

4.3 Logische Operatoren

&& AND, logische UND-Verknüpfung. Das Ergebnis kann nur *true* sein, wenn **beide Bedingungen erfüllt sind**.

|| OR, logische ODER-Verknüpfung. Das Ergebnis kann *true* sein, wenn die eine *oder* die andere Bedingung erfüllt ist, bzw. beide Bedingungen erfüllt sind.

! NOT, also nicht, bzw das Gegenteil. Wenn eine Bedingung *nicht wahr* ist, ist sie also *falsch*. Entsprechend ist *not false* ein *true*.

4.4 Vergleichsoperatoren

Diese Operatoren werden wir noch ausführlich bei den *Schleifen* und *Abfragen* benötigen und behandeln um *Bedingungen* zu formulieren.

== gleich. Bitte achten Sie darauf, dass **ein Gleichheitszeichen** eine **Zuweisung** ist und **zwei Gleichheitszeichen** die **Prüfung** auf Äquivalenz sind.

!= ungleich

< kleiner

<= kleiner oder gleich

> größer

>= größer oder gleich

5 Dialog-Boxen

Es gibt in *Javascript* einfache, *vordefinierte Funktionen* um einfache Dialog-Boxen *auf-poppen* zu lassen. Mit *alert* geht eine Hinweis-Box auf, die wieder verschwindet, wenn man den *OK-Button* drückt. Bei *confirm* hat man die Wahl zwischen *OK* und *Cancel* und bei *prompt* kann man nach einem gegebenen Hinweis-Text eine Text-Abfrage als Eingabe machen.

Kompaktere Programmierung ist möglich, nur habe ich an dieser Stelle darauf verzichtet um die Beispiele leichter verständlich zu machen.

Achtung: Die u.a. Beispiele sind auf das Wesentliche beschränkte *Code-Fragmente*, die um den Rest der HTML-Seite ergänzt werden müssen. Hier muss man mitdenken und nicht blind abtippen!

5.1 Alert

Ausgabe einer Meldung in einer Box. Die Box verschwindet, wenn der *OK-Button* gedrückt wurde. Der auszugebende Text kann entweder ein Text in Anführungsstrichen , eine gefüllte Textvariable, oder eine Kombination sein, die mit + verknüpft worden ist. Zeilenumbrüche sind in diesem Fall mit `\n` (für *new line*) und nicht mit `"<br>"` anzugeben, da die Ausgabe über eine *Javascript-Funktion* und nicht als dynamischer HTML-Text erfolgt!

```
var text;
text = "Bla bla bla \n und bla";
alert(text);
```

Abbildung 3: Prinzip der Alert-Box in Javascript

```
var meldung_ok;
var text;
text = "Bla bla bla \n und noch mehr bla";
meldung_ok = confirm(text);
if (meldung_ok){<Anweisung für ok>;}
else {<Anweisung für Cancel>;}
```

Abbildung 4: Prinzip der Confirm-Box in Javascript

5.2 Confirm

Im Prinzip ein *alert* mit *Rückgabewert*. D.h. die Funktion liefert einen *booleschen Wert* zurück der einer entsprechenden *Variablen* zugewiesen werden kann. Der Wert ist *true*, wenn der *OK-Button* gedrückt wurde und *false* bei *Cancel*. Dieser Wert kann später z.B. mit einer *if-Abfrage* ausgewertet werden. **Die *Aktionen* bei *if* und *else* stehen hier nur exemplarisch** und sind kein echter *Javascript-Code*.

5.3 Prompt

Mit *prompt* wird nun ein konkreter Eingabe-Text über ein Fenster mit Hinweis-Text abgefordert und über die Funktion zurück geliefert. Dieser *Rückgabewert* wird dann sinnvollerweise einer *Text-Variablen* zugewiesen und später weiter verarbeitet. Noch einmal: Der *Wert* der Variablen *text* wird der Funktion *prompt* als *Eingabeparameter* **zur Ausgabe** übergeben. Die Funktion *prompt* liefert als *Rückgabewert* den **eingegebenen Text**, den man für die weitere Verarbeitung der Variablen *eingabe* zuweist, denn sonst ist die Eingabe weg.

Danach findet im Beispiel innerhalb des *Eingabeparameters* von *alert* eine Zusammensetzung von Text und dem *Wert* der Variablen *eingabe* mit Hilfe des Operatoren **+** statt, um mit *alert* diesen kompletten Text auszugeben.

Ich hoffe Sie konnten so weit folgen.

```
var eingabe;
var text;
text = "Geben Sie Bitte Ihren Geburtsort ein:\n";
eingabe = prompt(text);
alert("Sie kamen in " + eingabe + " zur Welt.");
```

Abbildung 5: Prinzip der Prompt-Box in Javascript

6 Aufgaben

- Schreiben Sie ein Skript mit einer *Alert*-Box, die einen sinnvollen Text ausgeben soll.

- Schreiben Sie ein Skript mit einer *Confirm*-Box, bei der unterschiedliche *Alert*-Boxen anzeigen, welche Wahl getroffen worden ist.
- Programmieren Sie eine Funktion, bei der 3 verschiedene Texteingaben über ein *Prompt* erfolgen sollen und kombinieren Sie diese 3 Eingaben zu einem sinnvollen Text, der über ein *Alert* zur Kontrolle ausgegeben werden soll.
- Dokumentieren Sie ihre Programme in Prosa (Absichtserklärung, Realität, Abweichungen, Probleme, Lösungen) mit Kommentaren im Listing.