Einführung in Javascript

Die folgende Einführung soll Dir zentrale Grundlagen vermitteln und dabei auch das grnudlegende Verständnis von Javascript vermitteln. Aber fangen wir gleich an.

Letztlich ist alles was Du mit HTML und CSS erreichen kannst vor allem eines schön und statisch, aber eben nicht dynamisch. Kurzum: egal was jemand auf Deiner Seite macht, es wird sich außer das Öffnen eines neuen Fensters nicht wirklich etwas ändern. Es gibt von dieser Regel nur sehr wenige Ausnahmen wie etwa Slide-Shows, aber auch diese Änderungen laufen immer nach einem festgelegten Schema ab und können nicht beeinflusst werden. Damit also etwas mehr Dynamik in Deine Seite kommt, benötigst Du ein weiteres Werkzeug, das Dir genau das ermöglicht. Eines davon ist Javascript.

Doch was genau ist eigentlich Javascript?

JavaScript ist eine Programmiersprache, die als Zusatztechnik in Webseiten eingebunden wird. Die JavaScript-Programme, auch Scripte genannt, werden vom Web-Browser clientseitig[1] interpretiert. Das heißt, sie werden in Prozessoranweisungen übersetzt und ausgeführt → sie werden implementiert (nicht kompiliert). → JavaScripte haben Zugriff auf das Browserfenster und das darin angezeigte HTML-Dokument.

Was man vorher unbedingt wissen sollte:

- Javascript(=JS) ist Case sensitive.
  z.B.:

- JS verwendet Unicode Zeichensatz.
  z.B.:

- in JS werden die Anweisungen „**statements**“ genannt. Jedes statement wird mit einem Semikolon abgeschlossen:
  z.B.:

- Kommentare können auf zwei Arten eingefügt werden:

Möchte man beispielsweise einen einfachen Taschenrechner programmieren, benötigt man Variablen. In diesen Variablen werden dann bestimmte Werte eingetragen. Variablen werden durch folgende Syntax deklariert: var, let und const. Dabei gelten in JS folgende Regeln:

- var:

- let:

- const:

1 Nicht aber Serverseitig wie etwa PHP oder Perl. Dort wird auf dem Server interpretiert und das Ergebnis dann als HTML-Code wieder an den Client versendet. Vorteil hierbei ist eine Entlastung des Clients, die Nicht-Sichtbarkeit des Codes, Passwörter sind verschlüsselt auf dem Server gespeichert. Nachteil ist vor allem der hohe Traffic, da ständig Daten zwischen Server und Client hin und her gesendet werden müssen. Außerdem ist der Funktionsumfang etwas kleiner.

Wie bei CSS kann der Javascript-Code sowohl im Body- als auch im Head-Bereich stehen und natürlich kann dieser auch in eine externe Datei ausgelagert werden. Zu Übungszwecken setzen wir zunächst alle Befehle in den Body eines HTML-Dokuments.
Damit der Browser weiß, wo HTML-Code anfängt bzw. aufhört und an welcher Stelle er statt HTML-Code JS-Code interpretieren muss, werden alle JS-Anweisungen wie folgt gekennzeichnet:

```
1  <script type="text/javascript">
2  'use strict'
3      ...
4  </script>
```

Seit HTML5 kann man auch die Tags abkürzen und statt dessen nur noch `<script> … </script>` schreiben. Damit wir JS besser lernen, verwenden wir immer (!) nach dem öffnenden Tag die Anweisung wie in Zeile 2 `'use strict'`. Würden wir das weg lassen, würde der eine Browser uns mögliche Fehler verzeihen, ein anderer Browser hingegen nicht. Damit wir also unabhängig vom verwendeten Browser sehen, ob wir alles richtig gemacht haben, benutzen wir ausschließlich die `strict`-Variante.

Folgende weitere Regeln gelten dabei:

1. Javascript beginnt und endet mit dem Script Tag, wie man es oben sieht.
2. JS kann überall auf der Seite eingefügt werden. Wenn JS sofort -wenn die Seite geladen wird- aufgerufen werden soll, kann man es im Head- Bereich hinter den Titel (`<title>`) setzen. Möchte man jedoch HTML Elemente in der Seite manipulieren, sollte man die Script Tags ans Ende setzen, damit diese schon geladen sind. Eine weitere Möglichkeit besteht darin Funktionen über Events aufzurufen, was wir aber erst später lernen werden.
3. In Anweisungen (siehe unten: `alert("Hallo Welt!")`) dürfen keine Zeilenumbrüche vorkommen.

Fangen wir an:

In einem separaten Fenster sollen die Worte: „Hallo Welt!“ ausgegeben werden. Der dazu notwendige JS-Befehl ist dabei: `window.alert(“ ... “);`[2]

Übrigens können hier auch statt der doppelten einfache Anführungszeichen verwendet werden. Allerdings darf man nicht bei einer Ausgabe die beiden Anführungszeichen kombinieren → also entweder beide doppelt oder beide einfach.

Um diese Ausgabe im Dokument zu erzeugen, muss man statt des Befehls `alert` den Befehl `document.write(...);`[3] benutzen.

Statt eines konkreten Textes, der vielleicht noch angepasst werden soll (z.B. die Anrede mit Herr oder Frau), ist es nützlich, den anzuzeigenden Text in einer Variablen vorher zu speichern.

Dazu muss ein Variablenname vergeben werden und der Inhalt darin abgespeichert werden, also:

```
Variablentyp Variablenname='Inhalt';
```

2 Man darf hier auch die Kurzform `alert` statt `window.alert` verwenden.

3 Was genau `document.write` bedeutet ergibt sich von selbst. Entscheidend ist hier die Struktur. Alles was wir auf der Seite einfügen wollen beginnt mit „`document.`“. Dahinter wird dann der Befehl geschrieben, was genau auf der Seite zu tun ist.

Aufgaben:

1. Nicht jeder Name kann als Variablennamen benutzt werden. Ein paar Namen sind in JS verboten. Welche Namen dürfen nicht verwendet werden? Benenne die dazu gehörigen Regeln.
2. Variablen speichern Werte. Diese Werte können bestimmten Datentypen zugeordnet werden. JS unterscheidet zwischen 6 primitiven Datentypen. Benenne diese 6 Datentypen und erläutere was einen primitiven von einem komplexen Datentyp unterscheidet.
3. Finde heraus, was bei Variablen Deklaration, Zuweisung und Initialisierung bedeutet.
4. Mit welcher Tastenkombination öffnet man die Konsole in Chromium[4].

Beispiele für die Arbeit mit Variablen

Zunächst müssen wir eine Variable deklarieren, d.h. wir geben ihr einen Namen. Nennen wir unsere erste Variable „Zahl“. Dann sieht das im Code wie folgt aus:

```
var Zahl;
```

Nun weisen wir dieser Variablen einen Wert zu. Der Einfachheit halber nehmen wir die Zahl 42, was ja die ganze Wahrheit sein soll. Als Code steht dann:

```
Zahl=42;
```

Aufgaben:

Überprüfe alle Aufgaben auch mit Hilfe der Konsole, ob Deine Programme fehlerfrei laufen

5. Deklariere nun eine weitere Variable, die Du „halbeWahrheit“ nennst. Nun soll dieser Variablen auch der Wert von 42:2 zugeordnet werden. (Programmiere hier bitte auch die Rechnung, nicht einfach nur das Ergebnis.) Gib das Ergebnis im Dokument und ein weiteres Mal in einem neuen Fenster aus.
6. Untersuche, was passiert, wenn man zu einer Zahl x den Wert y hinzu addiert. (beide Werte sollen in einer separaten Variablen stehen, also Zahl1 = … und Zahl2 = … sowie Zahl3=Zahl1+Zahl2.
7. Untersuche, was passiert, wen man statt Zahlen zwei Zeichen bzw. Worte addiert.
8. Du findest in Deinem Importordner eine Datei mit dem Namen „Variablen1.html“ Dokumentiere, was dort genau geschieht und erläutere dabei insbesondere den Unterschied zwischen der Fehlermeldung „undefined“ und „ReferenceError“.

Vergiss nie, den strict-Modus zu benutzen sowie in Deinem Code auch die passenden Kommentare einzufügen!

4 Im zweifel kann man in jedem Browser mit einem Rechtsklick in dem sich dann öffnenden Auswahlmenü „untersuchen“ anklicken. Das führt dann auch auf die Konsole.

Ein kleines Spiel: 17+4 (vereinfachte Version)

Wir wollen nun ein erstes kleines Spiel programmieren. Vielleicht kennst Du das Spiel 17 und 4. Dabei werden Karten nacheinander von einem Stapel aufgedeckt. Der Spieler sagt „Weiter“, wenn er eine weitere Karte aufgedeckt haben will. Der Zahlenwert der aufgedeckten Karten wird dabei einfach addiert. Ziel ist es, dass die Summe aller Kartenwerte genau 21 ergibt. Liegt man drüber hat man verloren, liegt man drunter entscheidet, wer von den Spielern am nächsten an der Zahl 21 liegt. Wir vereinfachen das Spiel, indem wir es mit einem Würfel spielen. Wir würfeln einfach immer weiter, bis wir „Stop“ sagen. Dann werden die Punkte notiert. Dazu ein Beispiel:

Spieler 1 würfelt folgende Zahlen:  Weiter →  Weiter →  Weiter →  Weiter →  Stop
→ Also hat Spieler 1 nun insgesamt 19 Punkte

Spieler 2 würfelt dagegen folgende Zahlen:  Weiter →  Weiter →  Weiter →  Weiter →  Weiter →
→ Damit hat Spieler 2 verloren, weil er mehr als 21 Punkte hat.

Nun wollen wir dieses Spiel am Browser simulieren. Dazu benötigen wir Zufallszahlen; einen Button, der ein sogenanntes Event[5] auslöst und eine Funktion, die die Werte zusammen rechnet.

Daher ist es an der Zeit nun zunächst folgende Aufgaben zu bearbeiten:

9. Finde heraus, wie in JS Zufallszahlen erzeugt werden können und programmiere dies so, dass die Zufallszahlen auch angezeigt werden. Programmiere dies so, dass Du ein rechtsoffenes Intervall angibst. (Recherchiere, was ein rechtsoffenes (im Gegensatz zum geschlossenen)Intervall ist.)
10. Sicherlich hast Du schon gemerkt, dass Deine Zufallszahlen alle zwischen 0 und 1 liegen, was für unser Spiel nur wenig nützlich ist. Also müssen die Zahlen mit 6 multipliziert werden. Programmiere auch dies und schau Dir die Ergebnisse genau an.
11. Nun hast Du Zufallszahlen zwischen 0 und 5,9999… erzeugt. Daher musst Du nun alle Ergebnisse konsequent abrunden. Recherchiere auch hier, wie man in JS Zahlen so abrundet, dass nur ganze Zahlen vorhanden sind. Wenn Du das erledigt hast, teste erneut Deine Ergebnisse und ergänze den Code so, dass nur noch Zahlen zwischen 1 und 6 angezeigt werden.
12. Erkundige Dich wie man eine Funktion in JS programmiert und was eine Funktion macht. Notiere Dir Deine Ergebnisse und erstelle eine Übersicht dazu.

5 In unserem Fall ist das Event die Addition der gewürfelten Zahlen.