API

OPEN ACCESS

Publiziert im Januar 2020
DOI: 10.15460/apimagazin.2020.1.25

Zeitschriftenartikel

Begutachtet

Begutachtet:

Prof. Christine Gläser
HAW Hamburg
Deutschland

Erhalten: *07. Janauar 2020*

Akzeptiert: *16. Januar 2020*

Publiziert: *29. Januar 2020*

Empfohlene Zitierung:

BUESCHENFELDT, Dr.-Ing. Maika, 2020: *Erste Schritte mit JavaScript - Ein kurzes Tutorial für den Einstieg.* In: *API* **1**(1) [Online] Verfügbar unter: DOI: 10.15460/apimagazin.2020.1.25

Erste Schritte mit JavaScript - Ein kurzes Tutorial für den Einstieg

Dr.-Ing. Maika Büschenfeldt[1]

[1] Hochschule für Angewandte Wissenschaften, Hamburg, Deutschland
Wissenschaftliche Mitarbeiterin

Korrespondenz: redaktion-api@haw-hamburg.de

Zusammenfassung

Im Tutorial konzentrieren wir uns auf die Verwendung von JavaScript zur Manipulation von HTML-Elementen. In dieser kurzen Einführung wird der Webbrowser Chrome (Vers. 79.0.3945.88) und dessen Entwicklungswerkzeuge verwendet. Grundlegende HTML- und CSS-Kenntnisse werden vorausgesetzt.

Schlagwörter: JavaScript, Dokument Object Model, Client-Side, Tutorial

Abstract

In this tutorial we will focus on the use of JavaScript to manipulate HTML elements. In this short introduction we will use the web browser Chrome (vs. 79.0.3945.88) and its developer tools. Basic knowledge of HTML and CSS is assumed.

DOI: 10.15460/apimagazin.2020.1.25

1 Vorbemerkung

JavaScript wurde ursprünglich als clientseitige Skriptsprache des Webbrowsers konzipiert. Durch die Einbettung einer JavaScript-Engine in den Webbrowser *(Client)* wird es möglich, HTML und CSS um die interaktiven Möglichkeiten einer Programmiersprache zu erweitern. Man spricht in diesem Zusammenhang auch von dynamischem HTML. Inzwischen wird JavaScript auch außerhalb von Browsern auf Webservern oder in Microcontrollern verwendet.

Die Integration einer JavaScript-Engine in den Webbrowser macht die Entwicklung von clientseitigen JavaScript-Anwendungen sehr einfach: Wir benötigen nur einen einfachen Texteditor, um den Programmcode zu schreiben. Moderne Browser wie Chrome oder Firefox stellen zusätzlich Entwicklungswerkzeuge für den Einsatz von JavaScript bereit.

Im folgenden Tutorial konzentrieren wir uns auf die Verwendung von JavaScript zur Manipulation von HTML-Elementen. In dieser kurzen Einführung wird der Webbrowser Chrome (Vers. 79.0.3945.88) und dessen Entwicklungswerkzeuge verwendet. Grundlegende HTML- und CSS-Kenntnisse werden vorausgesetzt.

2 JavaScript einbinden

Da clientseitiges JavaScript im Webbrowser läuft, kann eine JavaScript-Anwendung ohne Probleme in eine HTML-Seite eingebunden werden. Wir benötigen dazu nur eine einfache Textdatei, die wir mit der Dateiendung *js* versehen. Die Dateiendung *js* steht für JavaScript. Auf diese Weise wird aus einer reinen Textdatei eine JavaScript-Datei. Eine JavaScript-Datei binden wir über das *<script>*-Tag ein.

```
<script src="myjs.js"></script>
```

Da wir mittels JavaScript gezielt auf HTML-Elemente über die Tag-Namen sowie die Attribute *id* und *name* eines HTML-Elements zugreifen werden, ist es wichtig, dass diese Elemente bei Aufruf eines JavaScripts schon bekannt sind. Wir binden das Script deshalb möglichst weit unten im HTML-Dokument ein, am besten direkt oberhalb des schließenden *</body>*-Tags.

Die Positionierung des *<script>*-Tags und die Manipulation von HTML-Elementen wird verständlicher, wenn wir uns das Zusammenwirken von JavaScript und HTML im Document Object Model (DOM) vergegenwärtigen.

3 Hintergrundwissen: JavaScript und das DOM-Modell

Den Quellcode einer HTML-Seite haben wir bisher nur als einen mit HTML-Tags formatieren Text kennengelernt. Wir vertiefen unser Verständnis, wenn wir uns vorstellen, was der Browser bei der Verarbeitung einer HTML-Seite macht:

DOI: 10.15460/apimagazin.2020.1.25

Beim Laden einer HTML-Seite verarbeitet der Browser den HTML-Quellcode von oben nach unten und überführt diesen in eine hierarchische Objektstruktur, das sogenannte Document Object Model (DOM). Dieses Modell wird im Arbeitsspeicher vorgehalten und bildet die Grundlage für die gezielte Manipulation von HTML-Elementen durch JavaScript.

Das DOM-Modell besteht aus verschachtelten Knoten, die als Baumstruktur angeordnet sind. In dieser Baumstruktur lassen sich verschiedene Knotentypen unterscheiden:

- Elementknoten bestehen aus den HTML-Elementen.
- Attributknoten sind assoziierte Objekte der HTML-Elemente.
- Textknoten bezeichnen den normalen Text, der innerhalb der Textauszeichnungselemente (Tags) steht.

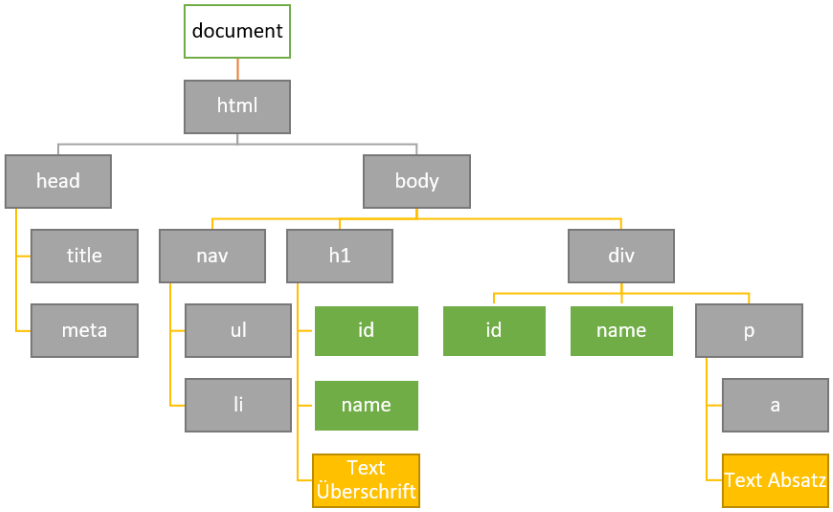


Abb. 1 - DOM-Modell

Nicht nur der Browser nutzt diese Struktur für seine Operationen, sondern auch CSS und JavaScript beziehen sich auf das DOM-Modell im Speicher.

4 Erste Schritte

Wir beginnen mit einer kleinen JavaScript-Funktion, die wir in unsere (noch leere) Script-Datei *(myjs.js)* schreiben.

```
1 function hello()
2 {
3       var message ="";
4       message = document.getElementById(„myInput“).value;
5       console.log(message);
6       message = ‚<h2>‘+message+‘</h2>‘;
7       document.getElementById(‚gruss‘).innerHTML=message;
8 }
```

DOI: 10.15460/apimagazin.2020.1.25

In die korrespondierende HTML-Datei schreiben wir den Code für eine Überschrift, ein Texteingabefeld, einen Button und einen leeren * *-Container. Das JavaScript wird in Zeile 10 über das *<script>*-Tag eingebunden.

```
1 <html>
2 <head>
3 <title>JavaScript Demo</title>
4 </head>
5 <body>
6 <h1>JavaScript Demo</h1>
7 <input type="text" id="myInput" class="form-control" placeholder="Gib was ein!">
8 <button class="btn btn-primary" onclick="javascript:hello()">Click me </button>
9 <div id="gruss"></div>
10 <script src="myjs.js"></script>
11 </body>
12 </html>
```

Mit diesen wenigen Code-Zeilen lernen wir bereits wichtige Dinge über JavaScript kennen:

Zeile 1: Durch das Schlüsselwort *function* leiten wir eine Funktion ein. Funktionen gehören zu den Grundkonzepten von JavaScript. Sie fassen eine Reihe von Anweisungen in einem Block zusammen. Der Aufruf erfolgt über den Namen der Funktion. Im HTML-Code wird die Funktion über den Eventhandler *onclick* (HTML-Datei: Zeile 8) aufgerufen.

Zeile 2 und Zeile 8: Die Anweisungen der Funktion stehen innerhalb der geschweiften Blockklammern. Die Blockstruktur eines Programms ist ein grundsätzliches Charakteristikum moderner Programmiersprachen. Ein Programm besteht aus Anweisungsblöcken und wird durch diese strukturiert. Jede Anweisung steht für einen Befehl und wird mit einem Semikolon abgeschlossen.

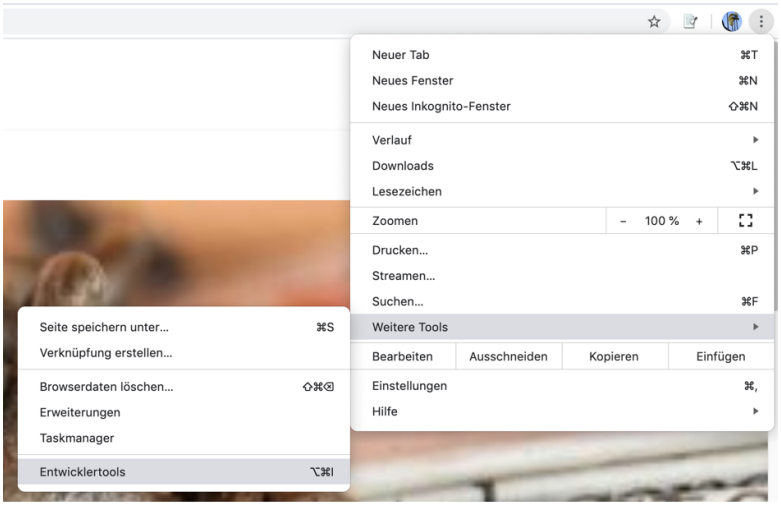
Zeile 3: Durch das Schlüsselwort *var* wird eine Variable deklariert (d. h. bekannt gemacht). Variablen-Deklarationen erfolgen immer, bevor der Programmcode ausgeführt wird. Variablen sind Speicherbereiche. Sie lassen sich als „Behälter" begreifen, in denen „Werte" gespeichert werden können. Sie haben einen Namen (Bezeichner). Über diesen Namen kann auf den Variablenwert zugegriffen werden. Weil die Variable *message* innerhalb einer Funktion deklariert wird, kann auf diese nur innerhalb der Funktion zugegriffen werden.

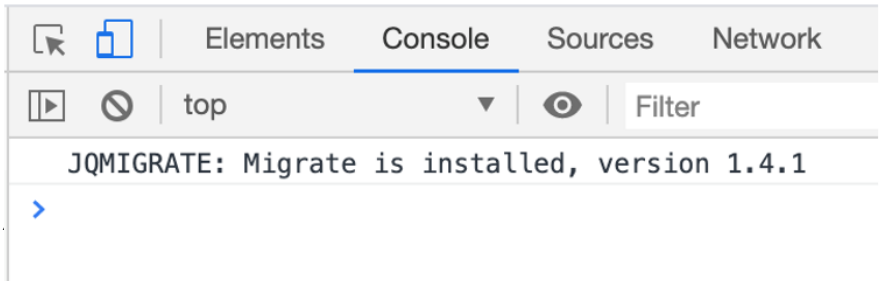
Zeile 4: Der Variablen mit dem Namen *message* wird ein Wert zugewiesen. Der zugewiesene Wert stammt aus dem Texteingabefeld im HTML-Code. Der Zugriff auf den Inhalt des Texteingabefeldes erfolgt über die DOM-Methode *getElementById* und die Eigenschaft

DOI: 10.15460/apimagazin.2020.1.25

value. Als Parameter wird (innerhalb der runden Klammern) der Wert des *id*-Attributs angegeben. Hierdurch erfolgt der Zugriff auf den Attributknoten *id* des HTML-Elements *<input>*. Den Wert des Attributknotens finden wir in Zeile 7 des HTML-Codes.

Zeile 5: Die Methode *console.log* schreibt eine Nachricht in die Konsole. Die Entwicklungstools des Chrome-Browsers stellen uns die JavaScript-Konsole zur Verfügung. Wir finden die Entwicklertools im Chrome-Menü unter „Weitere Tools“. In den Entwicklertools finden wir die Konsole unter dem Karteikartenreiter „Console“.





Zeile 6: Um den Inhalt der Variablen legen wir noch das HTML-Tag *<h2>*. Es entsteht eine Zeichenkette, die nun nicht mehr nur aus dem Inhalt des Texteingabefeldes besteht, sondern auch einen öffnenden und schließenden *<h2>*-Tag umfasst. Bei dem Zeichen + handelt es sich um den JavaScript-Verknüpfungsoperator. Mit einem Verknüpfungsoperator lassen sich Zeichenketten zusammenfügen. Um zu sehen, wie sich der Inhalt der Variablen *message* verändert, kann die Methode *console.log* verwendet werden.

Zeile 7: Unter Verwendung der DOM-Methode *getElementById* in Kombination mit der Eigenschaft *innerHTML* wird der Inhalt der Variablen *message* in den leeren * *-Container (HTML-Datei: Zeile 4) geschrieben.

DOI: 10.15460/apimagazin.2020.1.25

5 Das Ergebnis

Wenn alles geklappt hat, können wir in unsere HTML-Seite einen Text eingeben, der dann nach dem Drücken des „Click me“-Buttons im Browserfenster angezeigt wird.

6 Weitermachen

Wer seine Kenntnisse nach diesem kleinen Einstieg ausbauen möchte, findet im Wiki *Selfhtml* alles, was nötig ist, um tiefer in die JavaScript-Welt einzutauchen.

Einsteiger-Tutorial:
SELFHTML, 2019: JavaScript/Tutorials/Einstieg. [Online] Stand: 2019-07-17 [Zugriff am: 2020-01-16] Verfügbar unter: https://wiki.selfhtml.org/wiki/JavaScript/Tutorials/Einstieg

JavaScript-Referenz mit allen Sprachelementen, Kontrollstrukturen, Objektmethoden und Standardobjekten: SELFHTML, 2019: JavaScript. [Online] Stand: 2019-07-31 [Zugriff am: 2020-01-16] Verfügbar unter: https://wiki.selfhtml.org/wiki/JavaScript