

Ulrike Häßler

19,99 €
inkl. E-Book

JAVASCRIPT Schnelleinstieg

Dynamische Webseiten
programmieren in 14 Tagen

Zahlreiche
Praxisbeispiele
und Übungen



Inhalt

Einleitung

Programmieren lernen in 14 Tagen	11
Der Sandkasten – Programmbeispiele zum Download	11
Fragen und Feedback	12

Vorwort

Sofort ins kalte Wasser – ein fremdes Skript lesen und verstehen	13
Das aufgeschobene Vorwort	14
JavaScript-Versionen und Browser-Versionen	15



Vorbereitungen und Werkzeuge

1.1 HTML und CSS	17
1.2 JavaScript einbinden	19
1.3 JavaScript laden	21
1.4 Mein Freund, der Editor	23
1.5 Die Browser-Konsole – die Sprechstundenhilfe	25
1.6 Praktische Konsolen-Befehle	29



Variablen und Syntax

2.1 Variablen	33
2.2 Syntax – die »Grammatik« von JavaScript	39



Grundlegende Datentypen

3.1	Primitive (einfache) Datentypen	41
3.2	Zahlen – Number	42
3.3	Boolean – Wahrheitswerte	45
3.4	Strings – Zeichenketten	46
3.5	undefined, null und empty	50
3.6	Datentyp-Konvertierungen	51



Zusammengesetzte Datentypen

4.1	Objekte	57
4.2	Das Objekt Math	58
4.3	Übungen: Math	62
4.4	Mathematische Operatoren	62
4.5	Übung: Modulo	67
4.6	String-Eigenschaften und -Methoden	68
4.7	Übung: Strings ersetzen	72
4.8	Weitere String-Methoden	72
4.9	Übung: Zeichenkette extrahieren	78
4.10	Reguläre Ausdrücke	79



Programme steuern – Ablaufkontrolle

5.1	Abfragen und Schleifen	83
5.2	if-then-else	85
5.3	switch	89
5.4	Übung: Stundenplan mit switch/case	91
5.5	for-Schleifen	92
5.6	while-Schleifen	95
5.7	Übung: Zahlen von 1 bis 100 ausgeben	98

5.8	Logische Operatoren	98
5.9	Ternary-Operator	102
5.10	Übung: if-Anweisungen kürzen	105
5.11	Implizite Typ-Konvertierung in Abfragen	105



Funktionen

6.1	Funktionen erzeugen und aufrufen	107
6.2	Globaler Gültigkeitsbereich (Scope)	111
6.3	Übung: Globaler Scope, Block-Scope und Funktions-Scope	114
6.4	Funktionsausdrücke (Function Expressions)	114
6.5	Arrow-Funktionen	116
6.6	Debugging	118
6.7	Übungen: Funktionen	121



Objekte

7.1	Grundlagen	123
7.2	for-in-Schleife für Objekte	128
7.3	Das Objekt »document«	129
7.4	Konstruktor-Funktionen	130
7.5	Klassen	133
7.6	Datum und Zeit	134
7.7	Übung: Date und Time	140



Arrays

8.1	Arrays – Sammlungen unter einem Variablennamen	141
8.2	Array-Methoden	143
8.3	Arrays sortieren	152
8.4	Arrays verschachteln und zusammenfügen	154

8.5	Arrays durchlaufen	157
8.6	Array-Methoden höherer Ordnung: Callbacks	165
8.7	Übungen: array.filter()	170
8.8	Sparse Arrays – Arrays mit Lücken	171



JSON – JavaScript Object Notation

9.1	JSON-Objekte und JSON-Arrays	173
9.2	JSON-Objekte in Strings umwandeln und umgekehrt	177
9.3	Übung: Objekte/JSON	180



Document Object Model

10.1	DOM – das Document Object Model	181
10.2	Zugriff auf die DOM-Elemente	185
10.3	DOM-Methoden und -Eigenschaften	186
10.4	Zugriff mit CSS-Selektoren – querySelector() und querySelectorAll()	190
10.5	DOM-Navigation	195
10.6	Zugriff auf Inhalte mit innerHTML, innerText und TextContent	198
10.7	Elemente ins DOM einfügen	201
10.8	Übung: Elemente im DOM einbinden	207
10.9	Praxisbeispiel – komplexe Strukturen einfügen	207
10.10	Übung: Bild in das Fragment setzen	209
10.11	DOM-Elemente erzeugen und platzieren	210
10.12	Elemente ersetzen und entfernen	212
10.13	Übung: Elemente entfernen	217
10.14	CSS-Stile und -Klassen ändern	217



Ereignisse – Events

11.1	Event Handling – Ereignisse erkennen	223
11.2	Eigenschaften und Methoden der Maus-Events	232
11.3	Übung: Mauszeiger verfolgen	237
11.4	Tastatur-Events	237
11.5	Übung: Tastatur-Ereignisse	240
11.6	Vorbestimmte Verhalten verhindern – event.preventDefault()	240
11.7	Übung: Formulare prüfen	244
11.8	Event Delegation – ein Event Handler für viele Elemente	244
11.9	Ereignisse des Window-Objekts	245



Formulare prüfen und Daten versenden

12.1	input – Eingaben in Formularfelder	247
12.2	Formulardaten versenden	253
12.3	Datum und Dauer – Formulareingaben	256



Asynchrones JavaScript

13.1	Zeitgesteuerte Anwendungen	259
13.2	AJAX – XMLHttpRequest – Kommunikation mit dem Server	268
13.3	Das Fetch-API – GET – Daten abholen	275
13.4	Übung: showProduct(option)	281
13.5	Fetch Async/Await – warten auf die Antwort	281



Window, Cookies und die Web-APIs

14.1	Grundsätzliches zum Window-Objekt	285
14.2	Das Window-Objekt und das Navigator-Objekt	288
14.3	IntersectionObserver – Überwachung des sichtbaren Bereichs der Webseite	293
14.4	Web Animations API	297
14.5	Cookies – Gedächtnis für Webseiten	302
14.6	Übung: Cookies für Leaflet-Kartenposition	306
14.7	Web Storage – Speicher im Browser	307
14.8	Übung: Local Storage	310

Anhang

Glossar	311
Error – Fehlermeldungen	312
Keywords (Schlüsselwörter)	313
Quellen	314
Bildnachweis	314

Stichwortverzeichnis

Einleitung

Programmieren lernen in 14 Tagen

Mit diesem Buch haben Sie sich für einen einfachen, praktischen und fundierten Einstieg in JavaScript entschieden. Sie lernen ohne unnötigen Ballast alles, was SIE wissen müssen, um JavaScript effektiv für Projekte in Ihrem Berufs- und Interessensgebiet einzusetzen. Alles, was Sie dafür benötigen, sind grundlegende Kenntnisse in HTML und CSS.

Wenn Sie Zeit genug haben, können Sie jeden Tag ein neues Kapitel durcharbeiten und so innerhalb von zwei Wochen Programmieren lernen. Am besten lesen Sie das Buch neben der Computer-Tastatur und probieren die Programmbeispiele und Übungen gleich aus. Sie werden schnell erste Erfolge erzielen und Freude an der Programmierung mit JavaScript finden.

Der Sandkasten – Programmbeispiele zum Download

Sie müssen die Beispiele in diesem Buch nicht abtippen. Der Code der Beispiele und zu den Übungen steht Ihnen auf der Webseite des Verlags unter www.mitp.de/0778 zur Verfügung.

Fast immer liegen die Beispiel-Lösungen in einer externen JavaScript-Datei, sodass Sie die Wahl haben, sich eine eigene Lösung zu erarbeiten oder den Lösungsweg zu studieren.

Fragen und Feedback

Unsere Verlagsprodukte werden mit großer Sorgfalt erstellt. Sollten Sie trotzdem einen Fehler bemerken oder eine andere Anmerkung zum Buch haben, freuen wir uns über eine direkte Rückmeldung an lektorat@mitp.de.

Falls es zu diesem Buch bereits eine Errata-Liste gibt, finden Sie diese unter www.mitp.de/0778 im Reiter DOWNLOADS.

Wir wünschen Ihnen viel Erfolg und Spaß bei der Programmierung mit JavaScript!

Vorwort

Sofort ins kalte Wasser – ein fremdes Skript lesen und verstehen

Die ersten Gehversuche mit JavaScript starten die meisten Webdesigner mit einem fertigen Skript, das sie z.B. auf GitHub finden und vielleicht in einzelnen Anweisungen nach ihrer Vorstellung ändern.

Für den ersten Schritt in die Programmierung mit JavaScript reicht eine normale Webseite. Das `script`-Element am Ende des Dokuments (vor dem schließenden `body`-Tag) braucht keinerlei Attribute. Innerhalb der öffnenden und schließenden `script`-Tags wirken JavaScript-Anweisungen nur auf dieser Seite.

HTML:

```
<!DOCTYPE html>
<html>
<head>
  <title>Sofort ins kalte wasser!</title>
</head>
<body>
  <h1>Sofort ins kalte wasser!</h1>
  <blockquote>
    <p class="hallo">Hallo welt!</p>
    <p>Die klassische Einführung in die Programmierung.</p>
    <cite>Obi web kenobi</cite>
  </blockquote>
<script>
document.querySelector ("body").style = "color: green";
</script>
</body>
</html>
```

Der Aufruf `document.querySelector` ist das JavaScript-Äquivalent zum CSS-Selektor. In den runden Klammern des Aufrufs kann ein beliebiger CSS-Selektor stehen. In diesem Beispiel ist es ein HTML-Tag-Selektor. Nach den

schließenden runden Klammer erzeugt `.style` ein `style`-Attribut im öffnenden `body`-Tag und `"color: green"` ist eine ganz normale CSS-Regel.

`document.querySelector` ist sozusagen das JavaScript-Äquivalent eines CSS-Selektors und erreicht die Elemente über einen CSS-Selektor. Nach demselben Muster würde ein `CSS-class`-Selektor in den runden Klammern wirken. Schreiben Sie `".hallo"` in die runden Klammern und wählen Sie eine andere CSS-Regel, z.B. die Schriftgröße und die Hintergrundfarbe, um gezielt den Text im `p`-Element mit der Klasse `hallo` zu stylen.

```
document.querySelector(".hallo").style =  
    "font-size: 24px; background: yellow";
```

Der CSS-Selektor wird mit demselben Punkt vor dem Klassennamen angesprochen wie in der CSS-Datei, das Semikolon trennt die beiden Regeln voneinander.

Das aufgeschobene Vorwort

Einsatz von JavaScript

In den frühen Zeiten des Internets gab es nur statische Seiten auf der Basis von HTML. Erst CSS, JavaScript und PHP haben das Internet populär gemacht. In den Anfängen brachte JavaScript Effekte und kleine Animationen für die Webseite und übernahm dann mehr die Rolle des Vermittlers zwischen der Anwendung auf dem Server und dem Benutzer. Heute gestaltet JavaScript nicht nur Webseiten, sondern sitzt auch hinter PDF-Dateien und in Photoshop, in den Anwendungen für mobile Geräte und ist der Kern von Node.js, einem weiteren Webserver neben Apache.

Dieses Buch konzentriert sich auf die Grundlagen von JavaScript für Webseiten, auf praktische Anwendungen von einfachen Effekten bis hin zur Prüfung und Übermittlung von Formulardaten.

- Der erste Teil widmet sich den einfachen Vokabeln und der Grammatik von JavaScript: Datentypen wie Zahlen und Zeichenketten, Objekten und Arrays, Kontrollstrukturen und Funktionen.
- Der zweite Teil dreht sich um die Anwendung dieser Grundlagen in Webseiten: Wie ändert man Inhalte, erkennt Ereignisse wie einen Mausklick, erzeugt HTML-Elemente *on the fly* und weist ihnen CSS zu?

- Die Reaktion auf Aktionen des Benutzers eröffnet schließlich den Weg in die Kommunikation mit dem Server: Wie funktioniert die Übergabe der Benutzereingaben aus dem Formular an die Anwendung auf dem Server?

Programmiersprachen vs. natürliche Sprachen

Programmiersprachen haben viel mit natürlichen Sprachen gemeinsam: Sie haben Vokabeln und eine Grammatik – ihre Syntax. So wie man eine natürliche Sprache nicht allein aus dem Lehrbuch lernt, brauchen die Vokabeln der Programmiersprache und ihre Anwendung praktische Übung. Es dauert, bis Sie eine Diskussion in italienischer Sprache führen können, aber für den Anfang ist es super, wenn wir den Fremdenführer gut verstehen. Das Verständnis für ein fremdes Skript ist ein großer Schritt, um eigene Programme zu entwickeln.

Bis Sie als Einsteiger den ersten Algorithmus für eine Aufgabe formulieren können, passiert dasselbe wie bei den ersten Sätzen einer frisch erlernten Sprache: Da stottert man schon mal, aber mit jedem Gespräch wird es flüssiger.

Zwei Schritte vor, ein Schritt zurück

Bei den Grundlagen von JavaScript stellen sich Einsteiger die Frage: »Was mache ich mit dieser ganzen Theorie überhaupt?«. Dabei geht es weder um Vollständigkeit noch um technische Akkuratess, sondern um das Verständnis für die Konzepte. Erst wenn es nach der Vorstellung von einfachen Datentypen, Arrays, Objekten und Funktionen an das Ändern des HTML-Dokuments (das Document Object Model) geht, offenbaren sich viele Konzepte von JavaScript. Das vorliegende Buch greift darum an vielen Stellen zwei Schritte vor und verweist andererseits aus den Kapiteln zur Anwendung von JavaScript in Webseiten zurück auf die Grundlagen. Keine Sorge: Der praktische Umgang wirft ein neues Licht auf die Konzepte und bringt das tiefe Verständnis.

JavaScript-Versionen und Browser-Versionen

ECMAScript und JavaScript

JavaScript wurde 1995 von Netscape entwickelt und von Microsoft als JScript adaptiert. Ein Standard wurde aber erst durch die ECMA (European Computer Manufacturers Association) erreicht.

Die Grundlagen von JavaScript werden von der ECMA standardisiert, das W3C steuert alles rund um das Document Object Model zu, also den Teil von JavaScript, der sich speziell auf Webseiten bezieht. Auf jeden Fall aber werden JavaScript-Versionen als ECMAScript veröffentlicht. ECMAScript 6 war ein großer Sprung im Jahr 2015 und ist eine stabile Version für alle modernen Browser. Heute nutzt die ECMA Jahreszahlen anstelle von Versionsnummern und beschert uns jedes Jahr mit einer neuen Versionsnummer.

Das läuft nicht anders als bei neuen Elementen in HTML oder neuen Eigenschaften für CSS: Bis neue Versionen für die Programmierung von Webseiten genutzt werden können, muss sicher sein, dass die Browser die Neuheiten auf einer breiten Basis unterstützen. Darum gilt die Version 6 von JavaScript bis heute als Eckpfeiler für Online-Anwendungen.

Browser-Unterstützung

Das Erscheinen einer neuen Version bedeutet natürlich nicht, dass alle Browser direkt auf den Zug aufspringen. Als Entwickler fiebert man dem einen oder anderen lang ersehnten Feature entgegen, aber wir können es in der Regel erst nutzen, wenn die Mainstream-Browser die neuen Funktionen in ihre Versionen eingebaut haben und der größte Teil der potenziellen Benutzer wiederum auf die aktuellen Versionen der Browser umgestiegen ist.



Vorbereitungen und Werkzeuge

1.1 HTML und CSS

1.1.1 Korrektes HTML

Es ist nicht wichtig, ob Sie bereits PHP, TypeScript oder Java programmiert haben: Ohne solide Kenntnis und Erfahrung mit HTML und CSS ist der Einstieg in JavaScript steil und steinig. Schließlich ist ein Hauptgrund für den Einsatz von JavaScript der Eingriff in die Darstellung der Webseite. Valides – also korrektes – HTML ist die Basis für JavaScript. Obwohl die Browser viele HTML-Fehler ignorieren, schlagen falsch verschachteltes HTML und fehlende Hochkommas im HTML-Tag zurück. Darum gehört der Validator zum Alltag des JavaScript-Programmierers.

Abb. 1.1: <https://validator.w3.org> – Prüfen, ob die HTML-Struktur stimmt

1.1.2 Wie viel CSS-Kenntnis ist wichtig?

Wie gut Sie sich mit CSS auskennen sollten, hängt davon ab, auf welche Einsatzgebiete von JavaScript Sie sich in erster Linie vorbereiten:

- Animationen und Effekte
- Design und Layout von Webseiten
- Kommunikation mit dem Server

Wenn Sie JavaScript einsetzen wollen, um die Grenzen von CSS-Animationen zu überschreiten – etwa Animationen starten, wenn sie im Browserfenster sichtbar werden –, sind gute Kenntnisse in CSS-Animationen gefragt.

Bei Änderungen am Design von Content-Management-Systemen wie WordPress, Joomla oder Drupal kann das CSS anspruchsvoll sein. Dort sitzen auf einem einzelnen Element gut ein Dutzend CSS-Eigenschaften aus unterschiedlichen Quellen.

Konzepte wie Cascading (das Einfließen von Stilen aus vorangegangenen CSS-Regeln) und CSS-Selektoren brauchen ein tiefes Verständnis. Ein `section`-Element einer Seite beherbergt oft ein gutes Dutzend Stile aus verschiedenen Quellen:



Abb. 1.2: Der Erste-Hilfe-Koffer für den Einblick ins CSS ist die Browser-Konsole.

Die Kommunikation mit Anwendungen auf dem Server ist ein weiteres Einsatzgebiet von JavaScript. Das Einlesen und Weiterreichen von Benutzereingaben kommt mit einfachen Eingriffen in die Darstellung im Browser aus. Hier reichen solide Grundlagen von CSS.

1.1.3 HTML- und CSS-Kenntnisse – alles halb so wild

Zur Programmierung mit Sprachen wie Java oder C gehört die Formatierung der Ausgabe. Die Benutzerschnittstelle einer Desktop-Anwendung ist ein Monster von Menüs und Programmfenstern sowie die Reaktion auf Benutzereingaben und auf Nachrichten vom Betriebssystem. In JavaScript überneh-

men HTML und CSS die Formatierung der Benutzerschnittstelle vollständig. Selbst wenn die Kenntnis von HTML und CSS nur oberflächlich ist, haben Sie damit schon einen großen Teil des Weges zur Programmierung von Webseiten »im Kasten«.

1.2 JavaScript einbinden

1.2.1 Ein Sandkasten für JavaScript – Sandbox

JavaScript braucht keinen Webserver für die Ausführung, sondern läuft in einem beliebigen Ordner auf dem eigenen Rechner. Ein Texteditor und ein Browser, das ist schon alles, was für die ersten Schritte mit JavaScript erforderlich ist.

Dem Editor (besser »Programmeditor«) ist ein eigener Abschnitt gewidmet, aber für die ersten Schritte reichen etwa TextEdit auf dem Mac, Notepad unter Windows oder nano unter Unix. Wichtig ist, dass der Editor keine Formatierung einbringt – keine Buttons für fette oder kursive Schrift, keine Schriftgrößen oder -farben.

Die Vorbereitungen für die ersten Schritte mit JavaScript beschränken sich also auf das Anlegen eines Ordners – der Sandbox. Der Name des Ordners ist nicht relevant – `meine-webseite`, `sandbox` oder `klara`. JavaScript wird im Browser ausgeführt, für die Formatierung der Ausgabe sorgen HTML und CSS. Das macht den Einstieg in JavaScript einfacher als das Erlernen anderer Programmiersprachen.

```
▼ sandbox
  ► index.html
  ► script.js
```

Im ersten Teil dieses Buchs ist die HTML-Datei nur ein Grundgerüst aus `head` und `body`.

HTML:

```
<head>
  <title></title>
</head>
<body>
  <h1>Hallo Browser!<h1>
</body>
```

Die Datei `index.html` öffnet sich durch einen Doppelklick im Browser, aber Änderungen nach dem Öffnen der Seite zeigt der Browser erst an, wenn die Seite neu geladen wird. Darum empfiehlt Abschnitt 1.4 Programmeditoren mit einem Live-Server. Doch zunächst erst einmal zum Einbinden von JavaScript in Webseiten.

1.2.2 Das HTML-script-Element

JavaScript-Anweisungen können mit dem HTML-script-Tag direkt in das HTML-Dokument gesetzt oder als Datei mit der Namensweiterung `.js` in eine gesonderte Datei geschrieben werden. Das script-Tag bindet die externe JavaScript-Datei mit einem `src`-Attribut ein.

Das script-Element kann an beliebiger Stelle eines HTML-Dokuments stehen: im Seitenkopf oder innerhalb der öffnenden und schließenden `body`-Tags. Die empfohlene Arbeitsweise ist eine Skript-Datei mit der Dateierweiterung `.js` und dem `defer`-Attribut nach dem Laden der CSS-Dateien im Kopf der HTML-Datei.

HTML:

```
<head>
  <script src="script.js" defer></script>
</head>
<body> ... </body>
</html>
```

Viele Content-Management-Systeme binden das JavaScript noch vor dem schließenden `body`-Tag ein. Ein `defer`-Attribut wird dann nicht gebraucht.

```
<head> ... <head>
<body>
  ...
  <script src="script.js"></script>
</body>
```

Am Ende können Sie kleine Skripte auch direkt vor dem schließenden `body`-Tag einsetzen.