JavaScript Programmieren für Einsteiger

Paul Fuchs

2. Auflage: Oktober 2019

© dieser Ausgabe 2019 by BMU Media GmbH

ISBN 978-3-96645-016-4

Herausgegeben durch:
BMU Media GmbH
Hornissenweg 4
84034 Landshut

JavaScript Programmieren für Einsteiger

Inhaltsverzeichnis

1. Einleitung 9

1.1 JavaScript: eine Programmiersprache für dynamische Internetseiten 9
1.2 Die Entstehung von JavaScript 11
1.3 Die Sicherheit bei JavaScript-Anwendungen 13
1.4 JavaScript und HTML 14
1.5 Serverseitige und clientseitige Anwendungen 15
1.6 Für wen bietet es sich an, JavaScript zu erlernen? 18

2. Die Vorbereitungsmaßnahmen 20

2.1 Der Webbrowser: unverzichtbar für die Ausführung von JavaScript-Programmen 20
2.2 Der Texteditor für die Erstellung des Programmcodes 22

3. Die ersten Schritte mit Javascript 26

3.1 Anwendungsbeispiele: Was kann JavaScript? 26
3.2 Ein Hallo-Welt-Programm mit JavaScript erstellen 31
3.3 Kommentare für ein einfacheres Verständnis des Codes 34
3.4 JavaScript-Programme in eigene Dateien schreiben 37
3.5 "use strict": modernen JavaScript-Code erstellen 39
3.6 Eine Eingabe des Anwenders aufnehmen 41
3.7 Übungsaufgabe: einfache JavaScript-Programme selbst erstellen 44

4. Variablen in JavaScript 49

4.1 Welche Funktion haben Variablen in der Informatik? 49
4.2 Variablentypen 50
4.3 Variablen in JavaScript verwenden 52
4.4 let und var: Unterschiedliche Möglichkeiten für die Deklarierung von Variablen 56
4.5 Konstanten verwenden 57
4.6 Datentypen ermitteln und verändern 58
4.7 Operationen mit Variablen durchführen 64
4.8 Übungsaufgabe: eigene Programme mit Variablen schreiben 66

5. Die if-Abfrage: unverzichtbar für die Ablaufsteuerung 71

5.1 Der Aufbau der if-Abfrage ... 72
5.2 Vergleichsoperatoren für das Aufstellen einer Bedingung ... 74
5.3 Mehrere Bedingungen mit logischen Operatoren verbinden ... 78
5.4 Weitere Optionen mit else und else if einfügen ... 80
5.5 Das switch-Statement: Alternative zur if-Abfrage ... 82
5.6 Übungsaufgabe: Abfragen in den Programmen verwenden ... 86

6. Zusammengesetzte Datentypen in JavaScript 91

6.1 Arrays ... 91
6.2 Map und WeakMap ... 97
6.3 Set und WeakSet ... 99
6.4 Übungsaufgabe: mit zusammengesetzten Datentypen arbeiten ... 102

7. Schleifen für die Wiederholung einzelner Programmteile 108

7.1 Die while-Schleife ... 108
7.2 Die do-while-Schleife ... 112
7.3 Die for-Schleife ... 113
7.4 Sonderformen der for-Schleife ... 116
7.5 Schleifen mit break und continue steuern ... 119
7.6 Übungsaufgabe: Schleifen selbst erstellen ... 122

8. Funktionen in Javascript 128

8.1 Eine Funktion erstellen ... 128
8.2 Eine Funktion aufrufen ... 130
8.3 Der Gültigkeitsbereich der Variablen ... 132
8.4 Funktionen mit Übergabewerten ... 135
8.5 Funktionen mit Rückgabewert ... 137
8.6 Übungsaufgabe: mit Funktionen arbeiten ... 138

9. Objektorientierte Programmierung mit JavaScript 143

9.1 Was bedeutet objektorientierte Programmierung? ... 143
9.2 Javascript: Objektorientierung – ursprünglich ohne Klassen ... 147
9.3 Funktionen als Konstruktor verwenden ... 148
9.4 Vererbung durch Prototypen ... 150
9.5 Klassen in JavaScript ... 155
9.6 Methoden erstellen und anwenden ... 157
9.7 Datenkapselung in Javascript ... 161
9.8 Vorgefertigte Objekte und Methoden verwenden ... 165
9.9 Übungsaufgaben: Objekte verwenden ... 168

10. Fehlerbehandlung in JavaScript 173

10.1 Verschiedene Arten von Fehlern ... 173
10.2 Syntaxfehler beheben ... 174
10.3 Ausnahmen für Laufzeitfehler erstellen ... 177
10.4 Logische Fehler durch Debugging erkennen ... 181

11. JavaScript und Webbrowser 188

11.1 Die besonderen Anwendungsmöglichkeiten von JavaScript ... 188
11.2 Events in JavaScript ... 189
11.3 Verschiedene Objektmodelle ... 189

12. Browser Events 193

12.1 Was sind Events und wie lässt sich damit ein Programm steuern? ... 193
12.2 Auf Events reagieren: verschiedene Vorgehensweisen ... 195
12.3 Die Struktur der Seite: Wo werden Events ausgelöst? ... 200
12.4 Events delegieren ... 205
12.5 Mouse- und Keyboard-Events ... 207
12.6 Übungsaufgabe: Mit Events arbeiten ... 210

13. Das window-Objekt 216

13.1 Dialogfenster für Hinweise und Bestätigungen ... 216
13.2 Fenster schließen und neue Fenster öffnen ... 217
13.3 Den zeitlichen Ablauf steuern ... 219
13.4 Übungsaufgabe: mit dem window-Objekt arbeiten ... 223

14. Das document-Objekt 230

14.1 Auf die Inhalte des DOM-Baums zugreifen ... 230
14.2 Auf einzelne Elemente der Seite gezielt zugreifen ... 235
14.3 Weitere Gestaltungsmöglichkeiten ... 238
14.4 Übungsaufgabe: Dynamische Seiten mit dem document-Objekt erzeugen ... 241

15. Formulare mit JavaScript bearbeiten 247

15.1 Formulare: die einzelnen Bestandteile ... 247
15.2 Events für Formulare ... 250
15.3 Spezielle Methoden für Formularelemente ... 253
15.4 Die Eingaben der Formularfelder überprüfen ... 255
15.5 Übungsaufgabe: Formulare mit JavaScript erstellen ... 260

16. Weitere vordefinierte Objekte in JavaScript 264

16.1 Location ... 264
16.2 Images ... 267
16.3 History ... 269
16.4 Style ... 270
16.6 Übungsaufgabe: Vordefinierte Objekte verwenden ... 272

17. Datenspeicherung in JavaScript: Cookies und localStorage 278

17.1 Javascript: stark eingeschränkte Möglichkeiten für die Datenspeicherung ... 278
17.2 Cookies verwenden ... 279
17.3 Daten mit localStorage speichern ... 286
17.4 Übungsaufgabe: Informationen mit JavaScript speichern ... 290

18. Anwendungsbeispiel: Ein Mathematik-Quiz erstellen 295

18.1 Den grundlegenden Aufbau der Seite und des Spielfelds festlegen ... 296
18.2 Das Spiel beginnen: Spielstand erstellen und Position hervorheben ... 299
18.3 Funktionen für die Buttons erstellen ... 300
18.4 Die Fragen stellen ... 305

19. jQuery: effizient mit JavaScript arbeiten 314

19.1 Was ist jQuery? ... 314
19.2 Die Vorbereitungsmaßnahmen für die Verwendung von jQuery ... 316
19.3 Selektoren: HTML-Elemente über jQuery ansteuern ... 318
19.4 Die Inhalte der Seite mit jQuery verändern und auswerten ... 324
19.5 Events mit jQuery bearbeiten ... 329
19.6 Spezielle Effekte mit jQuery einfügen ... 330
19.7 Übungsaufgabe: mit jQuery arbeiten ... 334

20. AJAX: Eine Verbindung aus serverseitiger und clientseitiger Programmierung 340

20.1 Was ist AJAX und welche Vorteile bietet diese Technik? ... 340
20.2 Die Vorbereitungsmaßnahmen: einen lokalen Webserver installieren ... 342
20.3 Zusätzliche Informationen mit AJAX anfordern ... 345
20.4 Eine allgemeine Funktion für die Anforderung der Daten ... 350
20.5 AJAX und PHP: Beispiel für die Verbindung mit einer serverseitigen Scriptsprache ... 353
20.6 Mit AJAX Daten an den Server übermitteln ... 356
20.7 Übungsaufgabe: Internetseiten mit AJAX gestalten ... 364

Alle Programmcodes aus diesem Buch sind als PDF zum Download verfügbar. Dadurch müssen Sie sie nicht abtippen:
https://bmu-verlag.de/books/javascript/

Außerdem erhalten Sie die eBook Ausgabe zum Buch im PDF Format kostenlos auf unserer Website:

https://bmu-verlag.de/books/javascript/
Downloadcode: siehe Kapitel 20

Kapitel 1

Einleitung

Eine Programmiersprache zu erlernen, stellt eine spannende Aufgabe dar, die allerdings auch viel Zeit und ein erhebliches Durchhaltevermögen erfordert. Insbesondere wenn man gerade erst mit dem Programmieren anfängt und noch keine anderen Sprachen beherrscht, ist der Einstieg mit viel Mühe verbunden. JavaScript – die Programmiersprache, die hier behandelt wird – stellt dabei keine Ausnahme dar. Dieses Buch richtet sich jedoch speziell an Programmier-Anfänger. Selbst einfache Schritte werden dabei detailliert erklärt, sodass keine Verständnis-Schwierigkeiten auftreten sollten. Der erklärende Text wird durch zahlreiche Code-Beispiele ergänzt, die es erlauben, die Funktionen gleich auszuprobieren. Hinzu kommen Übungsaufgaben, die es ermöglichen, mit den erworbenen Fähigkeiten einfache Programme selbst zu entwerfen.

Bevor wir mit der praktischen Arbeit beginnen, sollen jedoch die wesentlichen Eigenschaften von JavaScript vorgestellt werden. Das ist wichtig, um die Anwendungsmöglichkeiten kennenzulernen und um zu wissen, was mit JacaScript alles möglich ist. Der Leser kann anhand dieser Informationen überprüfen, ob sich diese Programmiersprache für die eigenen Ziele eignet.

1.1 JavaScript: eine Programmiersprache für dynamische Internetseiten

Bei den meisten Programmiersprachen handelt es sich um sogenannte General Purpose Languages (GPL) – um Sprachen mit einem allgemeinen Anwendungsbereich. Damit ist es möglich, Desktop-Anwendungen und viele weitere Programme zu gestalten. JavaScript wird offiziell ebenfalls als GPL eingestuft. Allerdings ist der allgemeine Anwendungsbereich hierbei nur theoretischer Natur. Der wesentliche Ein-

satzzweck dieser Sprache besteht darin, dynamische Internetseiten zu erstellen.

Internetseiten wurden ursprünglich in HTML verfasst. Dabei handelt es sich jedoch um keine Programmiersprache, die einen Ablauf vorgeben und auf Eingaben des Anwenders reagieren kann. Vielmehr ist HTML eine reine Auszeichnungssprache, die verschiedene Elemente des Texts markiert. Diese Markierungen ermöglichen daraufhin eine passende Anordnung und ein ansprechendes Layout. Die Seiten sind jedoch statisch: Wenn der Server sie einmal ausgeliefert hat, sind keine Änderungen mehr möglich.

Als sich das Internet immer weiter ausbreitete, bestand bei vielen Webdesignern der Wunsch, dynamische Seiten zu erzeugen. Das bedeutet, dass diese nicht immer die gleichen Inhalte bereitstellen. Anstatt dessen sollen sie diese dynamisch erzeugen und an die aktuellen Anforderungen anpassen. Für die Erzeugung dynamischer Internetseiten entstanden im Laufe der Zeit viele verschiedene Techniken. Eine von ihnen ist die Programmiersprache JavaScript. Die Funktionen, die sie anbietet, sind speziell auf diesen Anwendungsbereich ausgelegt. Sie stellt vielfältige Möglichkeiten bereit, um den Aufbau, das Layout und den Inhalt zu verändern. Aufgrund der hohen Spezialisierung konnte sich JavaScript in diesem Bereich durchsetzen und zählt mittlerweile zu den etablierten Web-Technologien.

Die Anwendungsmöglichkeiten für andere Bereiche waren jedoch lange Zeit nur theoretischer Natur. JavaScript enthält zwar viele Befehle, die man auch in Desktop-Anwendungen und in anderen Bereichen verwenden könnte. Allerdings stand keine passende Umgebung für die Umsetzung zur Verfügung. JavaScript wurde ausschließlich im Web-Browser ausgeführt und kam daher nur für Internetanwendungen zum Einsatz.

Aufgrund der großen Beliebtheit dieser Sprache hat sich der Anwendungsbereich jedoch inzwischen ausgeweitet. 2009 erschien beispielsweise Node.js. Dabei handelt es sich um eine Plattform, über die sich mit JavaScript Serveranwendungen programmieren lassen. Später ka-

men auch Entwicklungsumgebungen wie Electron oder NW.js hinzu, mit denen es möglich ist, Desktop-Anwendungen mit dieser Programmiersprache zu erstellen.

Dennoch bleiben Browser-Anwendungen das mit großem Abstand häufigste Einsatzgebiet von Javascript. In den übrigen Bereichen kommt die Sprache nach wie vor selten zum Einsatz. Daher empfiehlt sich dieses Buch in erster Linie für Leser, die lernen möchten, dynamische Webseiten zu gestalten. Wenn man hingegen Desktop- oder Server-Anwendungen erstellen will, ist es sinnvoller, eine andere Programmiersprache auszuwählen. Auch die Anwendungsbeispiele in diesem Buch beziehen sich ausschließlich auf Browser-Anwendungen.

1.2 Die Entstehung von JavaScript

Die Entstehung von JavaScript ist eng mit der Entwicklung der Webbrowser verbunden. In den Anfangszeiten des Internets waren die Browser nur dazu in der Lage, Texte darzustellen. Das änderte sich 1993 mit der Entwicklung von NCSA Mosaic. Dieser Browser ermöglichte erstmals eine grafische Darstellung und trug erheblich zur Ausbreitung des Internets bei. Im Folgejahr entstand das Unternehmen Mosaic Communications, das viele der ursprünglichen Entwickler von NCSA Mosaic beschäftigte. Dieses entwarfen den neuen Webbrowser Mosaic Netscape, der Ende 1994 erschien. Trotz des ähnlich lautenden Namens handelte es sich dabei um eine vollständige Neuentwicklung. Um Rechtsstreitigkeiten mit dem Unternehmen NCSA, das den ersten Mosaic-Webbrowser entwickelt hatte, zu vermeiden, wurde dieser kurze Zeit später in Netscape Navigator umbenannt.

Der Unternehmensgründer Marc Andreessen war jedoch überzeugt davon, dass die Entwicklung des Internets alleine mit statischen HTML-Seiten nur langsam voranschreiten würde. Daher entstand die Idee, eine weitere Sprache zu entwickeln, die es ermöglichen sollte, dynamische Inhalte im Internet darzustellen.

Die neue Sprache sollte die Seiten nicht eigenständig erzeugen. Die Idee bestand darin, sie in den HTML-Code zu integrieren, um diesen mit dynamischen Inhalten zu ergänzen. Die Anforderung an diese Sprache bestand außerdem darin, dass sie einfach zu erlernen sein muss. Das war notwendig, um die Anwendung Web-Designern, bei denen es sich meistens nicht um professionelle Programmierer handelt, schmackhaft zu machen. Außerdem entstehen viele Web-Projekte als private Initiativen. Dafür sind häufig Personen ohne große Programmier-Erfahrung verantwortlich. Damit diese die neue Technik ebenfalls nutzen können, ist ein einfacher Aufbau sehr wichtig.

1995 stellte Netscape Communications den Programmierer Brendan Eich ein. Dieser wurde mit der Entwicklung der neuen Programmiersprache beauftragt. Doch kam es bereits kurze Zeit später zu einer Kooperation mit Sun Microsystems. Dieses Unternehmen hatte die Programmiersprache Java entwickelt und begann damit, sie für Webanwendungen einzusetzen. Trotzdem entschied Netscape Communications, die Entwicklung der neuen Programmiersprache voranzutreiben. Sie sollte jedoch keine Konkurrenz zu Java sein, sondern eine Ergänzung dazu. Diese Zusammenarbeit führte allerdings dazu, dass sich die neue Sprache hinsichtlich ihrer Syntax an Java orientierte. Die erste Version erschien 1995 unter der Bezeichnung LiveScript und wurde im Netscape Navigator 2.0 umgesetzt.

Abgesehen von einigen syntaktischen Gemeinsamkeiten war LiveScript jedoch ganz anders aufgebaut als Java. Dennoch entschied sich das Unternehmen nur wenige Monate nach der Ersterscheinung, die Programmiersprache in JavaScript umzubenennen. Dafür gab es jedoch keine technischen Gründe. Die Umbenennung diente lediglich dem Marketing, um von der großen Beliebtheit von Java zu profitieren. Die ähnliche Namensgebung führt jedoch noch heute manchmal zu Verwechslungen. Es ist daher stets wichtig, zu berücksichtigen, dass Java und JavaScript zwei vollkommen unterschiedliche Programmiersprachen darstellen.

1.3 Die Sicherheit bei JavaScript-Anwendungen

Wenn man sich mit JavaScript befasst, stößt man früher oder später auf den Aspekt der Sicherheit. Dieser spielt bei dieser Programmiersprache eine entscheidende Rolle. Das liegt daran, dass die Anwender hierbei ein Programm auf ihrem Computer ausführen, das meistens von einer für sie unbekannten Quelle stammt. Wenn man im Internet surft, dann trifft man dabei auf unzählige Seiten, deren Seriosität man nur schwer einschätzen kann. Daher besteht dabei immer die Möglichkeit, dass es sich um einen zwielichtigen Anbieter handelt, der versucht, sensible Daten zu entwenden oder den Computer lahmzulegen.

Wenn man ein Programm im Fachhandel kauft oder aus dem Internet herunterlädt, muss man die Installation immer zunächst bestätigen. Das Betriebssystem gibt dabei den Urheber des Programms an und fragt den Anwender, ob er dieses ausführen will. Dabei sollte man nur zustimmen, wenn die Software von einer vertrauenswürdigen Quelle stammt. Bei einem JavaScript-Programm ist der Ablauf jedoch ganz anders. Wenn man die Ausführung dieser Programmiersprache im Browser erlaubt hat, werden die Programme, die in die Webseiten eingefügt sind, automatisch ausgeführt. Dabei muss der Anwender keine gesonderte Zustimmung abgeben.

Das könnte für den Anwender sehr riskant sein. Auf diese Weise wäre es möglich, dass er einen Virus installiert oder zulässt, dass das Programm Daten entwendet. Aus diesem Grund gibt es ein spezielles Sicherheitssystem, das für einen zuverlässigen Schutz sorgt. Dieses wird als Sandbox bezeichnet. Das bedeutet, dass der Browser hierfür eine spezielle Umgebung schafft und diese von den übrigen Funktionen des Rechners abschirmt. JavaScript-Programme können daher nur Elemente verwenden, die der Browser ihnen zur Verfügung stellt. Es ist jedoch nicht möglich, auf das Dateisystem zuzugreifen. Das verhindert es beispielsweise, dass ein JavaScript-Programm Dateien ausliest. Das ist sehr wichtig, da es diese sonst ohne Zustimmung des Anwenders an einen beliebigen Server übermitteln könnte. Außerdem kann es keine Programme auf dem Computer installieren. Das beugt der Ausbreitung von Schadsoftware vor.

Die Sandbox sorgt zwar für eine hohe Sicherheit. Allerdings schränkt sie die Möglichkeiten des Programmierers deutlich ein. In anderen Sprachen ist es beispielsweise üblich, neue Dateien zu erstellen, um Daten dauerhaft zu speichern. Das Programm liest diese dann beim Start ein, damit der Anwender auf die bereits bei vorherigen Sitzungen erstellten Werte zugreifen kann. Das ist aufgrund des Sandbox-Prinzips mit JavaScript jedoch nicht möglich – zumindest solange es sich um eine Browser-Anwendung handelt. Das ist bei der Konzeption derartiger Anwendungen stets zu berücksichtigen.

Obwohl das Sandbox-Prinzip bei der Ausführung von JavaScript-Programmen für eine gute Sicherheit sorgt, gibt es viele Anwender, die dennoch Bedenken haben. Aus Furcht vor Viren und anderen Schadprogrammen deaktivieren sie die Ausführung von JavaScript-Programmen daher vollständig. Diesen Aspekt muss der Programmierer bei der Gestaltung seiner Programme ebenfalls berücksichtigen. Wenn man eine Internetseite mit JavaScript erstellt, sollte man stets darauf achten, dass diese auch dann ausführbar sein muss, wenn der Anwender diese Programmiersprache deaktiviert hat. Falls man diesen Aspekt nicht berücksichtigt, schließt man zahlreiche Anwender von der Nutzung der Seite aus.

1.4 JavaScript und HTML

Ursprünglich bestanden Internetseiten aus Text, dessen einzelne Bestandteile mit HTML ausgezeichnet wurden. Diese Abkürzung steht für Hypertext Markup Language. Markup Language bedeutet übersetzt Auszeichnungssprache. Das sagt aus, dass es sich hierbei nicht um eine Programmiersprache handelt. Die HTML-Tags dienen lediglich dazu, die Funktion einzelner Textelemente festzulegen und der Seite auf diese Weise eine Struktur zu verleihen.

HTML-Seiten sind stets statisch. Das bedeutet, dass sie immer den gleichen Inhalt haben, unabhängig davon, wer die Seite aufruft und zu welchem Zeitpunkt dies geschieht. Eine individuelle Anpassung der Inhalte ist damit nicht möglich. In den Anfangszeiten des Internets, als

dieses vorwiegend dazu diente, wissenschaftliche Informationen auszutauschen, war diese Funktionsweise ausreichend. Als es in den 90er Jahren eine immer größere Bedeutung für den Handel und für die Freizeitgestaltung erlangte, wurden jedoch schon bald die Grenzen dieser Technik deutlich.

Wie bereits in Kapitel 1.2. beschrieben, führten diese Einschränkungen von HTML zur Entwicklung von JavaScript und von weiteren Techniken, die eine dynamische Erzeugung von Webinhalten ermöglichen. Dabei ist es jedoch wichtig, zu beachten, dass JavaScript HTML nicht vollständig ersetzt. Vielmehr handelt es sich hierbei um eine Ergänzung zu dieser Auszeichnungssprache.

Das führt dazu, dass die JavaScript-Elemente stets in den HTML-Code eingefügt werden. Eine eigenständige Verwendung von JavaScript – ganz ohne HTML – ist nicht sinnvoll. Das bedeutet, dass ein JavaScript-Entwickler stets über solide HTML-Kenntnisse verfügen muss.

Dieses Lehrbuch setzt voraus, dass der Leser bereits die wichtigsten Grundlagen von HTML beherrscht. Wenn dies nicht der Fall ist, sollte man zunächst ein Lehrbuch zu diesem Thema bearbeiten. Die Grundlagen von HTML sind jedoch innerhalb kurzer Zeit zu erlernen. Da es sich hierbei nicht um eine Programmiersprache, sondern um eine einfache Auszeichnungssprache handelt, sollte sich der Aufwand hierfür in Grenzen halten. Außerdem verwenden die Beispiele in diesem Buch einen möglichst einfachen HTML-Code. Daher sind nur die wichtigsten Grundkenntnisse erforderlich, um die Bedeutung zu verstehen.

1.5 Serverseitige und clientseitige Anwendungen

Um dynamische Internetseiten zu erstellen, ist es notwendig, ein Computerprogramm auszuführen und die Inhalte an die individuellen Anforderungen anzupassen. Dieses Programm kann an zwei unterschiedlichen Orten ausgeführt werden: entweder auf dem Server oder auf dem Rechner des Anwenders. Im ersten Fall spricht man von serverseitigen Programmierung, im zweiten Fall von clientseitigen Programmen.

Beide Alternativen weisen in der Praxis recht große Unterschiede auf. Bei der serverseitigen Programmierung erzeugt das Programm bei jedem Aufruf eine neue Seite. Daher kann es die Inhalte dynamisch anpassen. Dabei erstellt es jedoch reinen HTML-Code. Das bedeutet, dass die Seite statisch wird, sobald sie einmal ausgeliefert wurde. Daher sind danach keine Änderungen mehr möglich. Das dynamische Element der Seite besteht darin, dass die Inhalte bei der Erstellung individuell zusammengestellt werden. Dabei kann das Programm Werte aus Datenbanken verwenden und damit bei jedem Aufruf eine andere Seite erzeugen. Eingabewerte des Anwenders kann das Programm über Formularfelder aufnehmen. Allerdings ist es nicht möglich, die entsprechenden Werte direkt auf der Seite selbst zu verarbeiten. Wenn der Anwender das Formular abschickt, übermittelt er dem Server die eingegebenen Inhalte. Dieser wertet diese aus und gestaltet damit eine neue Seite. Das Ergebnis sendet er daraufhin wieder an den Anwender.

Bei clientseitigen Anwendungen übergibt der Server dem Besucher ein Programm, das dieser dann auf seinem eigenen Rechner ausführt. Das ermöglicht es, Seiten zu erstellen, die auch für den Endanwender dynamisch sind. Man kann hierbei die Eingaben des Nutzers direkt aufnehmen und weiterverarbeiten. Um die Ergebnisse anzuzeigen, ist es nicht notwendig, die Seite neu zu laden.

Die Verwendung serverseitig gestalteter Webseiten bietet viele Vorteile. Dazu zählt insbesondere der Zugriff auf Datenbanken und Dateien, die auf dem Server abgelegt sind. Das macht es möglich, Daten dauerhaft zu speichern. Bei einem Webshop ist es beispielsweise sehr wichtig, dass dieser die Bestellungen eines Kunden dauerhaft registriert. Eine clientseitige Programmierung wäre hierfür nicht geeignet. Ein weiterer Vorteil besteht in der erhöhten Sicherheit. Der Ersteller des Programms führt dieses auf seinem eigenen Server aus. Daher ist nicht davon auszugehen, dass er hier Schadsoftware verbreiten will. Wenn das Programm hingegen auf dem Browser des Anwenders ausgeführt wird, kann das Möglichkeiten für Angriffe bieten. Auch in umgekehrter Richtung besteht ein Sicherheitsrisiko. Der Anwender könnte das

Programm manipulieren, um fehlerhafte Ergebnisse an den Server zu übermitteln.

Die clientseitige Programmierung bietet hingegen den Vorteil, dass diese die spezifischen Einstellungen des Systems des Anwenders berücksichtigen kann. Wenn beispielsweise die Uhrzeit angezeigt werden soll, dann zeigt ein clientseitiges Programm stets den Wert der Zeitzone an, in der sich der Anwender befindet. Das ist bei der serverseitigen Programmierung nicht der Fall. Außerdem ist es möglich, die Werte immer wieder zu aktualisieren, ohne dass es erforderlich ist, die Seite neu zu laden. Darüber hinaus nimmt bei der clientseitigen Programmierung der Datenverkehr ab, da ein Austausch zwischen Server und Client deutlich seltener notwendig ist. Das ist insbesondere bei langsamen Netzwerken oder bei sehr umfangreichen Programmen von Bedeutung.

JavaScript kommt fast immer für die clientseitige Programmierung zum Einsatz. Zwar gibt es mittlerweile auch Umgebungen, die serverseitige JavaScript-Anwendungen erlauben, doch ist das bislang sehr selten. Aufgrund der genannten Vor- und Nachteile kommt JavaScript vorwiegend für Bereiche zum Einsatz, in denen die Sicherheit nur eine untergeordnete Rolle spielt. Auch wenn es notwendig ist, die Werte dauerhaft abzuspeichern, stellt JavaScript in der Regel nicht das richtige Mittel dar. Diese Programmiersprache kommt in der Praxis in erster Linie für gestalterische Elemente zum Einsatz. Sie bietet sich auch für in die Webseite integrierte Funktionen an – beispielsweise für einen Währungsrechner. Außerdem kommt JavaScript häufig zur Validierung der Daten eines Formularfelds zum Einsatz – solange diese nicht sicherheitsrelevant ist.

In der Praxis ist es üblich, sowohl die server- als auch die clientseitige Programmierung gemeinsam zu verwenden. Für serverseitige Scripte kommt meistens die Programmiersprache PHP zum Einsatz. Mit deren Hilfe werden HTML-Dokumente erstellt. Dabei ist es jedoch möglich, auch JavaScript-Elemente einzufügen, sodass eine Mischung aus beiden Alternativen besteht.

Sehr beliebt ist außerdem AJAX. Diese Abkürzung steht für Asychronous JavaScript and XML. Mit dieser Technik ist es möglich, Daten zwischen dem Server und dem Client auszutauschen, ohne dass dafür die Seite komplett neu erstellt werden muss. Auch hierbei handelt es sich um eine Verbindung zwischen serverseitiger und clientseitiger Programmierung. JavaScript stellt hierfür eine wichtige Grundlage dar.

1.6 Für wen bietet es sich an, JavaScript zu erlernen?

Die vorhergehenden Abschnitte haben bereits deutlich umrissen, worin das wesentliche Anwendungsgebiet von JavaScript besteht. Diese Programmiersprache dient in erster Linie dazu, clientseitig programmierte dynamische Internetseiten zu erstellen. Ihr Einsatzgebiet ist sehr spezifisch – für andere Aufgaben kommt sie nur äußerst selten zum Einsatz. Daher eignet sich diese Programmiersprache in erster Linie für Personen, die genau in diesem Bereich aktiv werden möchten. JavaScript zu lernen, bietet sich daher vorwiegend für Webdesiger an. Die Sprache gibt die Möglichkeiten, dem Layout der Seite dynamische Elemente hinzuzufügen und weitere Funktionen zu integrieren.

Personen, die JavaScript lernen, verfügen normalerweise bereits über einige Erfahrung im Bereich der Erstellung statischer Internetseiten. Der Grund dafür liegt zum einen darin, dass es sich hierbei um eine sinnvolle Weiterbildungsmaßnahme handelt, die die bereits bestehenden Fähigkeiten optimal ergänzt. Zum anderen sind solide HTML-Kenntnisse eine wichtige Voraussetzung, um mit JavaScript zu arbeiten. Personen, die schon über Erfahrungen bei der Erstellung statischer HTML-Seiten verfügen, bringen daher alle notwendigen Grundlagen mit.

In den bisherigen Abschnitten wurde erwähnt, dass es mittlerweile auch Entwicklungsumgebungen gibt, die es zulassen, JavaScript-Programme für Server- oder Desktop-Anwendungen zu erstellen. Das widerspricht in gewisser Weise der Aussage, dass sich das Erlernen von

JavaScript ausschließlich für Personen eignet, die im Bereich des Webdesigns arbeiten möchten. Allerdings ist es dabei wichtig, zu beachten, dass die Verwendungsmöglichkeiten außerhalb der Browser-Anwendungen bislang stark eingeschränkt sind. Wer das Ziel hat, Server- oder Desktop-Anwendungen zu erstellen, sollte daher besser von Anfang an eine andere Programmiersprache auswählen. Die genannten Möglichkeiten eignen sich in erster Linie für Personen, die bereits über gute JavaScript-Kenntnisse verfügen und sich neue Anwendungsfelder eröffnen möchten, ohne auf eine neue Programmiersprache umzusteigen.

Ein weiterer Punkt, der JavaScript von vielen anderen Programmiersprachen abhebt, ist die Komplexität der damit erstellten Programme. JavaScript zeichnet sich im Bereich der Browser-Anwendungen dadurch aus, dass die Programme innerhalb einer Sandbox ablaufen. Das bedeutet, dass sie keinen Zugriff auf Funktionen des Betriebssystems haben. Auch der Zugang zur Hardware oder zu Netzwerken ist nicht möglich. Der wesentliche Verwendungszweck von JavaScript besteht in der Erstellung von dynamischen Design-Elementen und kleinen in die Webseite integrierten Anwendungen. Die Komplexität solcher Programme ist meistens gering. Daher handelt es sich bei JavaScript-Entwicklern häufig um Fachkräfte die keine studierten Informatiker sind. Vielmehr spielen gestalterische Aspekte eine wichtige Rolle. Daher handelt es sich hierbei häufig um Designer, die sich zusätzlich zu ihren eigentlichen Aufgaben die entsprechenden Programmierkenntnisse aneignen.

Kapitel 2

Die Vorbereitungsmaßnahmen

Bevor man damit beginnen kann, Programme in JavaScript zu erstellen, ist es notwendig, den Computer auf diese Aufgabe vorzubereiten. Für die Entwicklung eines Programms sind grundsätzlich zwei Elemente erforderlich. Zum einen ist ein Text-Editor notwendig, um den Quellcode zu erstellen. Wenn das Programm abgeschlossen ist, ist zum anderen eine spezielle Software für die Ausführung erforderlich.

Beim Quellcode handelt es sich um Informationen in Textform. Damit daraus ein Programm wird, ist es jedoch notwendig, den Text in die Maschinensprache zu übersetzen. Diese ist für den Prozessor lesbar, sodass dieser die entsprechenden Befehle ausführen kann. Für die Übersetzung gibt es zwei Möglichkeiten. Entweder kommt dafür ein Compiler zum Einsatz, der aus dem Code ein ausführbares Programm in Maschinensprache erstellt. Dieses kann man dann ohne weitere Hilfsmittel ausführen. Eine andere Möglichkeit besteht darin, das Programm während der Laufzeit zu interpretieren. Das bedeutet, dass die Übersetzung bei jeder Ausführung aufs Neue stattfindet. Das Programm ist nur dann ausführbar, wenn ein entsprechender Interpreter auf dem Computer installiert ist. Bei JavaScript handelt es sich um eine interpretierte Sprache. Der Interpreter, der dafür zum Einsatz kommt, sollte jedoch auf den meisten Computern bereits installiert sein. Hierbei handelt es sich um einen gewöhnlichen Webbrowser.

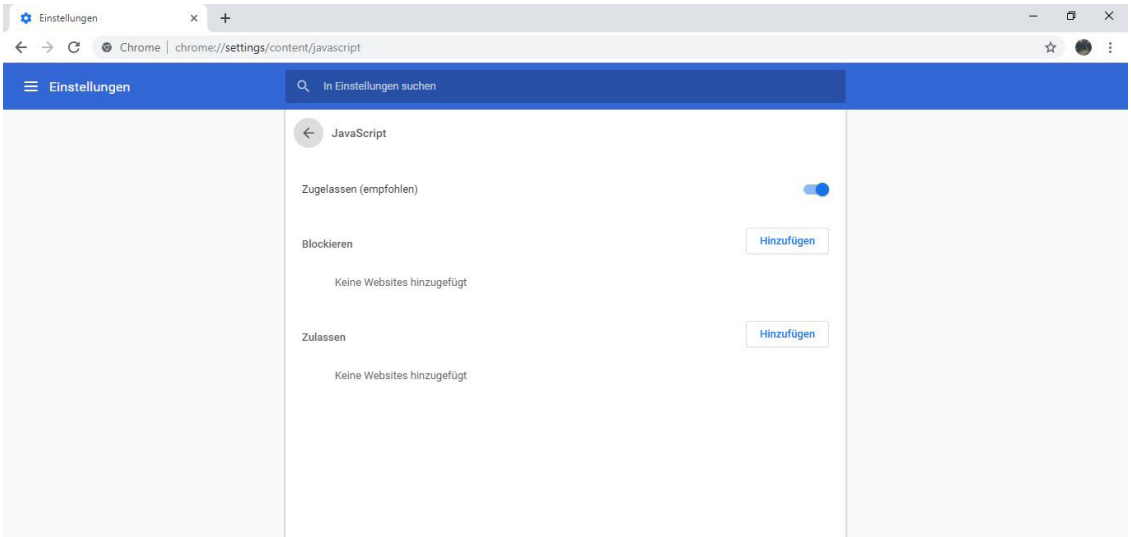
2.1 Der Webbrowser: unverzichtbar für die Ausführung von JavaScript-Programmen

Um die Programme in diesem Buch auszuführen, ist ein Webbrowser notwendig. Dabei ist lediglich darauf zu achten, dass dieser JavaScript unterstützen muss. Das ist heutzutage jedoch bei allen gängigen Mo-

dellen der Fall – von Google Chrome über Mozilla Firefox bis hin zu Microsoft Edge und vielen weiteren.

Jeder dieser Webbrowser eignet sich für die Ausführung von JavaScript Programmen. Die Auswahl bleibt dabei dem Leser überlassen. Bei der Erstellung dieses Buchs kam der Browser Chrome von Google zum Einsatz. Das bedeutet, dass die Screenshots zu den Programmbeispielen mit diesem Browser erstellt sind. Grundsätzlich kann der Leser jedoch auch gerne ein anderes Modell verwenden. Dabei muss lediglich beachtet werden, dass es dabei zu minimalen Unterschieden in der Darstellung der grafischen Elemente der Programme kommen kann. Das sollte allerdings in der Regel kein Problem darstellen. Wer Wert auf eine genaue Übereinstimmung legt, sollte jedoch den gleichen Browser verwenden. Wenn dieser noch nicht auf dem Computer installiert ist, kann man ihn ganz einfach gratis im Internet unter https://www.google.com/chrome/ herunterladen.

Aus Sicherheitsgründen bieten alle Browser die Möglichkeit, Javascript zu deaktivieren. In diesem Fall lassen sich die Programme selbstverständlich nicht ausführen. Deshalb ist es wichtig, diesen Punkt zu überprüfen und gegebenenfalls JavaScript zu aktivieren. In Google Chrome findet man diese Möglichkeit unter den erweiterten Einstellungen:



Screenshot 1 Die Ausführung von JavaScript zulassen

2.2 Der Texteditor für die Erstellung des Programmcodes

Um den Quellcode zu verfassen, ist ein Texteditor notwendig. Dabei handelt es sich um eine Software, die die Inhalte als reinen Text abspeichert. Das stellt einen wesentlichen Unterschied zu Textverarbeitungsprogrammen wie Microsoft Word dar. Diese speichern neben dem eigentlichen Text noch vielfältige weitere Informationen zum Layout ab. Daher sind sie zum Programmieren vollkommen ungeeignet. Wenn man einen mit Word erstellten Code mit einem Browser ausführen will, kommt es zu einem Fehler. Daher ist es unbedingt notwendig, einen passenden Texteditor zu verwenden.

Die meisten Betriebssysteme verfügen bereits über einen integrierten Texteditor. Bei einem Windows-Rechner ist dies beispielsweise der Microsoft Editor – auch unter dem Namen NotePad bekannt. Bei Linux ist je nach Distribution häufig Gedit oder Kate vorinstalliert. Mit diesen Programmen wäre es grundsätzlich möglich, JavaScript-Programme zu erstellen. Allerdings sind die Funktionen dabei stark eingeschränkt. Insbesondere der Microsoft Editor bietet nur einen minimalen Funktionsumfang. Das macht das Programmieren damit sehr schwierig. Aus diesem Grund ist es insbesondere für Windows-Nutzer sehr zu empfehlen, einen zusätzlichen Texteditor für das Programmieren mit JavaScript zu installieren.

Dieser bietet zahlreiche hilfreiche Funktionen. Bereits wenn man das erste Programm erstellt, fällt auf, dass dabei die einzelnen Bestandteile des Quellcodes unterschiedliche Farben erhalten. Das dient dazu, die verschiedenen Schlüsselbegriffe hervorzuheben. Das ermöglicht einen übersichtlicheren Code. Diese Funktion wird als Syntax-Highlighting bezeichnet. Darüber hinaus erzeugen hochwertige Texteditoren automatische Einrückungen. Wenn später Schleifen, Funktionen oder if-Abfragen in ein Programm eingefügt werden, stellen diese eine abgeschlossene Einheit dar. Um dies deutlich zu machen, rückt der Texteditor sie automatisch ein. Auch das trägt zur Übersichtlichkeit des Quellcodes bei. Wenn man diese Einheiten abgeschlos-

sen hat, kann man sie auch einklappen. Das bedeutet, dass ihr Inhalt nicht mehr angezeigt wird. Das sorgt für eine kürzere Darstellung, die es erleichtert, die wesentlichen Bestandteile des Programms zu erkennen. Bei Bedarf kann man diese jedoch auch jederzeit wieder ausklappen.


```
Datei  Bearbeiten  Format  Ansicht  ?
<html>
<head>
<title>
Meine erste JavaScript-Seite
</title>
</head>
<body>
<script>
alert ("Willkommen zum JavaScript-Kurs");
</script>
</body>
</html>
```

Screenshot 2 Programmcode mit dem Microsoft Editor

```
File  Edit  Selection  View  Go  Debug  Terminal  Help        beispiel.html - Visual Studio Code
Welcome    beispiel.html ×
1  <html>
2      <head>
3          <title>
4              Meine erste Javascript-Seite
5          </title>
6      </head>
7      <body>
8          <script>
9          alert ("Willkommen zum JavaScript-Kurs");
10         </script>
11     </body>
12 </html>
```

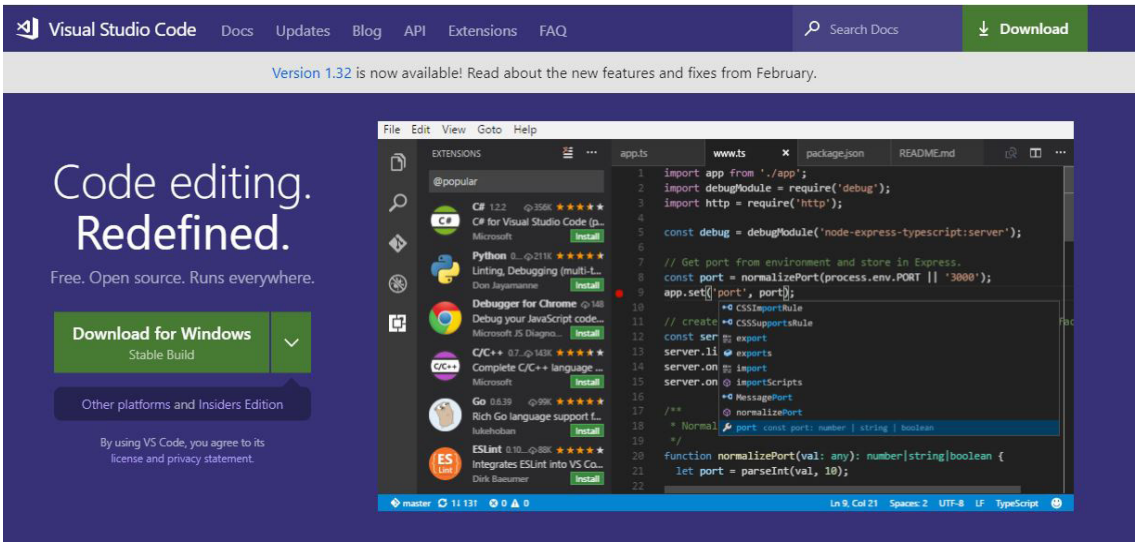
Screenshot 3 Der gleiche Code mit Syntax-Highlighting und automatischen Einrückungen

Die Funktionen des Texteditors beschränken sich jedoch nicht nur auf die Erzeugung eines übersichtlichen Codes. Darüber hinaus ist in der Regel eine Autofill-Funktion integriert. Das bedeutet, dass der Programmierer seine Eingaben automatisch vervollständigen kann. Sobald man die ersten Buchstaben eines Befehls eintippt, erscheint eine Auswahlliste mit den möglichen Optionen. Das beschleunigt nicht nur die Erstellung des Codes. Darüber hinaus lassen sich auf diese Weise viele Fehler vermeiden, die entstehen können, wenn man den Code von Hand eingibt.

Besonders praktisch ist es, eine integrierte Entwicklungsumgebung zu verwenden. Nach der englischen Bezeichnung Integrated Development Environment wird diese häufig als IDE abgekürzt. Diese bietet noch viele weitere praktische Funktionen. Beispielsweise kann man damit den Code mit einem einzigen Mausklick ausprobieren. Insbesondere während der Entwicklungsphase sorgt das für eine erhebliche Arbeitserleichterung. Darüber hinaus sind dabei in der Regel Instrumente für das Debugging enthalten. Diese dienen dazu, Fehler im Programm zu entdecken.

Aufgrund der genannten Vorteile soll für die Entwicklung der Programme in diesem Buch eine IDE zum Einsatz kommen. Hierfür gibt es viele verschiedene Angebote. Die Wahl fällt dabei auf Visual Studio Code (Nicht zu verwechseln mit der ebenfalls von Microsoft entwickelten IDE Visual Studio). Diese IDE unterstützt JavaScript und ist außerdem gratis erhältlich. Darüber hinaus nimmt die Software nur wenig Speicherplatz in Anspruch und läuft auch auf Rechnern mit einer geringen Hardwareausstattung sehr gut. Visual Studio Code steht unter folgendem Link zu Download bereit:

https://code.visualstudio.com/



Screenshot 4 Die Download-Seite von Visual Studio Code

Nach dem Download öffnet sich der Installations-Assistent für diese Software. Dabei ist es ausreichend, alle Standard-Einstellungen zu übernehmen und das Programm auf diese Weise zu installieren. Bereits nach wenigen Sekunden sollte dann ein hochwertiger Texteditor für die Entwicklung von JavaScript-Programmen zur Verfügung stehen.

Kapitel 3

Die ersten Schritte mit Javascript

Wenn ein Webbrowser und ein Texteditor auf dem Computer installiert sind, kann es mit dem Programmieren losgehen. Im ersten Schritt werden dabei einige Programme erzeugt, die typische JavaScript-Funktionen vorstellen. Dabei werden nicht alle Details des Codes erklärt. Das wäre für den Anfänger noch etwas zu kompliziert. Vielmehr geht es dabei darum, die Anwendungsmöglichkeiten dieser Programmiersprache vorzustellen. So kann sich der Leser ein genaueres Bild davon machen. Diese Beispielprogramme sollen einen praktischen Überblick darüber verschaffen, was mit JavaScript alles möglich ist. Erst danach wird die Erstellung eines JavaScript-Programms von Grund auf erklärt. Dabei werden alle Einzelheiten des Quellcodes erörtert, sodass der Leser bei jedem Befehl genau versteht, welche Funktion dieser ausführt. Damit der Einstieg möglichst leicht fällt, kommen dabei sehr einfache Beispielprogramme zum Einsatz.

3.1 Anwendungsbeispiele: Was kann JavaScript?

Dieser Abschnitt stellt einige Anwendungsbeispiele von JavaScript vor. Auf diese Weise kann der Leser sich zu Beginn ein genaueres Bild davon machen, was mit JavaScript alles möglich ist. Die einzelnen Befehle der Beispielprogramme werden an dieser Stelle jedoch nicht erläutert. Es geht hierbei lediglich darum, die Funktionen kennenzulernen. Detaillierte Beschreibungen zu den einzelnen Kommandos folgen im weiteren Verlauf dieses Buches.

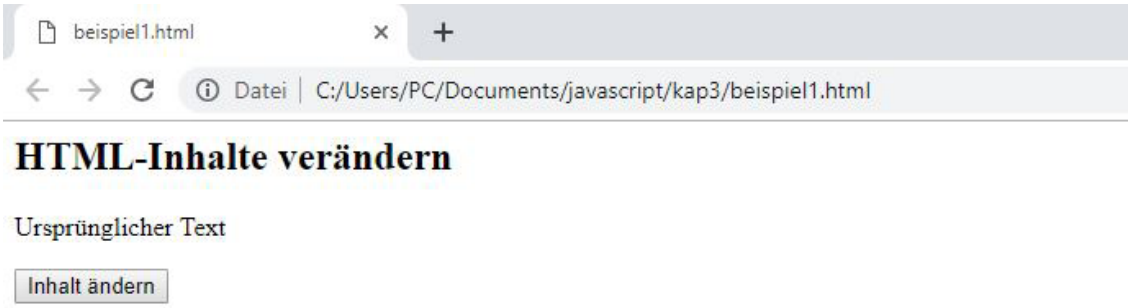
Dennoch stellt es eine sinnvolle Übung dar, den Code abzutippen und ihn selbst auszuprobieren. Dazu ist es notwendig, die IDE Visual Studio Code zu öffnen. Danach kann man den Beispielcode einge-

ben. Zum Abschluss muss man das Programm abspeichern. Es muss genau wie gewöhnliche HTML-Dateien die Endung .html oder .htm erhalten. Wenn man die Datei im entsprechenden Ordner anklickt, erscheint beim ersten Versuch eine Auswahlliste mit den möglichen Optionen zum Öffnen. Hierbei muss man einen Webbrowser auswählen und diese Auswahl abspeichern. Daraufhin wird bei jedem weiteren Öffnen der Datei automatisch der Webbrowser für die Ausführung gewählt.

Das erste Programm soll zeigen, dass es mit JavaScript möglich ist, Textinhalte auf einer Seite zu verändern:

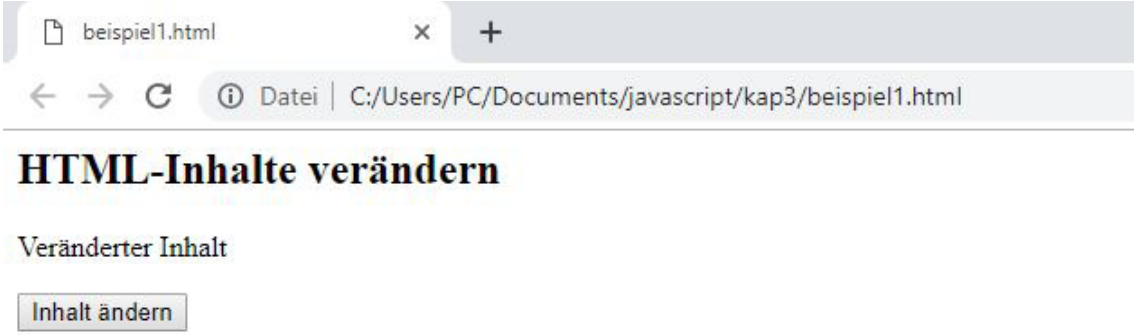
```
<html>
    <body>
        <h2>HTML-Inhalte verändern</h2>
        <p id ="absatz">Ursprünglicher Text</p>
        <button type = "button" onclick = "document.
        getElementById('absatz').innerHTML =
        'Veränderter Inhalt'">Inhalt ändern</button>
    </body>
</html>
```

Anmerkung: Da die einzelnen Zeichen in der Druckversion nur schwer zu erkennen sind, wird darauf hingewiesen, dass nach dem Wort “Inhalt” zunächst ein einfaches und danach ein doppeltes Anführungszeichen steht.



Screenshot 5 Die ursprüngliche Seite

Wenn man dieses Programm ausführt, erscheint zunächst der ursprüngliche Text auf der Seite. Wenn man jedoch den Button darunter anklickt, ändert das JavaScript-Programm den Inhalt dieses HTML-Elements, sodass ein neuer Text angezeigt wird:

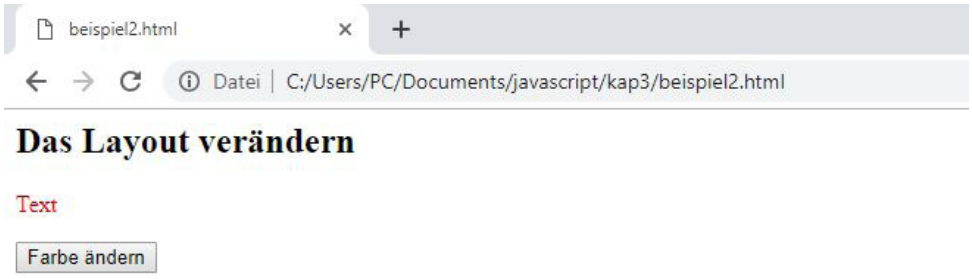


Screenshot 6 Die Seite nach dem Klick auf den Button

Das zweite Beispiel zeigt, dass man mit Javascript auch das Layout einer Seite beeinflussen kann:

```
<html>
    <body>
        <h2>Das Layout verändern</h2>
        <p id ="absatz">Text</p>
        <button type = "button" onclick = "document.
        getElementById('absatz').style.color =
        'red'">Farbe ändern</button>
    </body>
</html>
```

Wenn man das Programm aufruft, erscheint der Text zunächst in schwarzer Farbe. Wenn man auf den Button klickt, wird er jedoch in Rot angezeigt. Dieses Beispiel zeigt, dass man mit JavaScript nicht nur HTML-Elemente verändern kann. Darüber hinaus ist es möglich, auch Einfluss auf die Layout-Vorgaben zu nehmen, die mit CSS definiert werden.



Screenshot 7 Der Text erscheint nach dem Klick auf den Button in roter Farbe

JavaScript macht es auch möglich, Grafiken zu beeinflussen. Das zeigt das folgende Programm:

```
<html>
    <body>
        <h2>Grafiken auswechseln</h2>
        <button onclick="document.getElementById('meinBild').
        src='bild1.gif'">Bild 1</button>
        <img id="meinBild" src="bild1.gif">
        <button onclick="document.getElementById('meinBild').
        src='bild2.gif'">Bild 2</button>
    </body>
</html>
```

Screenshot 8 Die Darstellung des roten Rechtecks (bild1.gif)

Damit dieses Programm funktioniert, ist es notwendig, dass im gleichen Ordner wie diese Datei zwei Grafiken abgespeichert sind – eine unter dem Namen bild1.gif und die andere unter der Bezeichnung bild2.gif. Für dieses Beispiel wurden ein rotes und ein grünes Rechteck erstellt. Um die Funktionsweise selbst zu überprüfen, ist es jedoch möglich, beliebige Fotos oder Grafiken zu verwenden – solange man diese unter der entsprechenden Bezeichnung abspeichert.

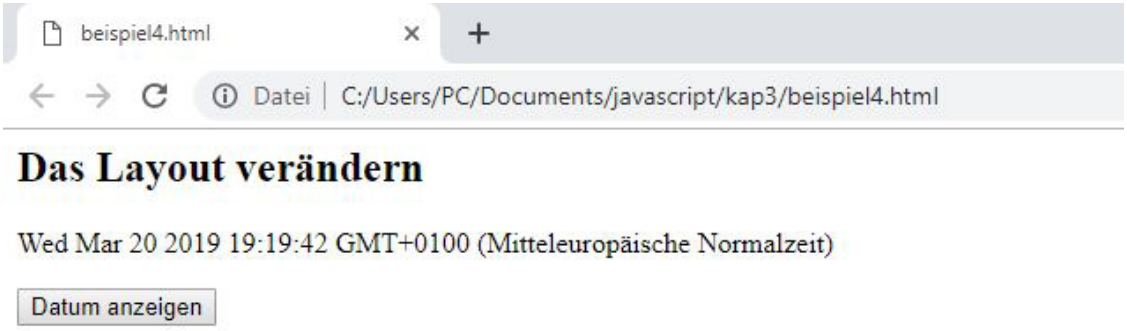
Wenn man nun auf den Button mit der Aufschrift "Bild 2" drückt, tauscht JavaScript die Grafik aus und das grüne Rechteck erscheint.

Screenshot 9 Die Darstellung des grünen Rechtecks (bild2.gif)

Das letzte Beispiel zeigt, dass JavaSript auch manche Betriebssystemfunktionen nutzen kann. Dazu wird die Funktion `Date()` verwendet. Wenn man den Button betätigt, zeigt diese die aktuelle Zeit in der Zeitzone des Anwenders an:

```
<html>
    <body>
        <h2>Auf Betriebssystem-Funktionen zugreifen</h2>
        <p id ="absatz">Hier erscheint gleich das Datum</p>
```

```
        <button type = "button" onclick = "document.
        getElementById('absatz').innerHTML =
       Date()">Datum anzeigen</button>
    </body>
</html>
```



Screenshot 10 Die Ausgabe des Datums

3.2 Ein Hallo-Welt-Programm mit JavaScript erstellen

Die meisten Lehrbücher, die Anfängern das Programmieren beibringen, beginnen mit einem Hallo-Welt-Programm – ganz unabhängig davon, um welche Programmiersprache es sich dabei handelt. Dieses stellt in der Regel die einfachste Möglichkeit dar, um ein Programm in der entsprechenden Sprache zu erzeugen. Seine Funktion besteht lediglich darin, eine kurze Nachricht auszugeben. Diese lautet meistens "Hallo Welt!" – beziehungsweise auf Englisch "Hello World!".

Der Sinn dieses Hallo-Welt-Programms besteht darin, die Grundlagen der entsprechenden Programmiersprache kennenzulernen. Dabei muss der Anwender sich nur mit einem einzigen Befehl befassen, sodass der Einstieg leicht fällt. Dabei geht es in erster Linie darum, die Strukturen kennenzulernen, die bei jedem Programm zu beachten sind.

Wenn man ein Hallo-Welt-Programm mit JavaScript erstellen will, dann hat man dabei mehrere Alternativen für die Ausgabe. Eine Möglichkeit besteht darin, den Ausgabetext in den Quellcode der HTML-Seite einzubetten. In diesem Fall erscheint er direkt auf der entsprechenden Seite.

Es gibt jedoch noch einen anderen Befehl, der genauso einfach zu verwenden ist. Dieser fügt den Text nicht in die bestehende Seite ein. Anstatt dessen öffnet er ein kleines Fenster und gibt ihn hier aus. Diese Funktion zeigt auf den ersten Blick, dass JavaScript viele zusätzliche Möglichkeiten bietet, die mit reinem HTML nicht verfügbar sind. Daher soll unser Hallo-Welt-Programm in JavaScript gleich diese erweiterte Funktion nutzen. Es soll nicht nur die entsprechende Nachricht ausgeben, sondern diese in einem eigenen kleinen Fenster anzeigen. Hierfür dient der alert-Befehl.

Um mit der Arbeit zu beginnen, ist es jedoch zunächst notwendig, eine einfache HTML-Seite zu erstellen. Wie in der Einleitung bereits geschrieben, werden für dieses Buch grundlegende HTML-Kenntnisse vorausgesetzt. Daher sollte es keine Probleme bereiten, den Aufbau der folgenden Seite zu verstehen:

```
<html>
    <head>
        <title>
            Meine erste Javascript-Seite
        </title>
    </head>
    <body>

    </body>
</html>
```

Grundsätzlich sei erwähnt, dass in diesem Lehrbuch stets ein sehr einfacher HTML-Code verwendet wird. Der Fokus liegt dabei auf den Javascript-Elementen. Um eine komplette Seite zu erzeugen, sind in diesem Beispiel auch `head`- und `title`-Tags enthalten. Darauf wird in den folgenden Beispielen jedoch häufig ebenfalls verzichtet, um diese so einfach wie möglich zu gestalten. Lediglich wenn man die ent-

sprechenden Inhalte mit JavaScript beeinflussen will, werden diese aufgeführt.

Die Beispielseite enthält zwar einen Titel, der `body`-Bereich ist jedoch noch leer. An dieser Stelle soll nun der JavaScript-Code eingefügt werden. Dieser muss stets mit dem script-Tag eingeführt werden. Daran erkennt der Browser, dass es sich hierbei um ein JavaScript-Programm handelt, dessen Befehle er ausführen muss.

Früher musste innerhalb des `script`-Tags das Attribut `type="text/javascript"` stehen. In vielen älteren Lehrbüchern und Codebeispielen ist dieses noch zu finden. Mittlerweile ist das jedoch nicht mehr notwendig. Daher ist ein einfaches script-Tag ohne weitere Attribute ausreichend, um den JavaScript-Code kenntlich zu machen.

In die `script`-Tags muss man dann den `alert`-Befehl einfügen. Dieser erzeugt eine kleine Info-Box. Danach steht eine Klammer, in die man – innerhalb von Anführungszeichen – einen beliebigen Text einfügen kann. Dieser wird dann im entsprechenden Fenster angezeigt.

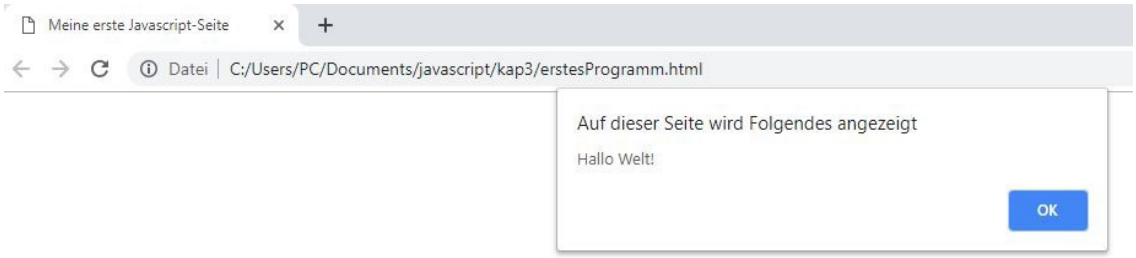
```
<script>
        alert ("Hallo Welt!");
</script>
```

In diesem Beispiel steht nach dem Befehl ein Semikolon – wie in vielen anderen Programmiersprachen üblich. Das ist in JavaScript jedoch optional, wenn danach ein Zeilenumbruch folgt. Daher könnte man dieses Zeichen hier auch weglassen. Nur wenn man zwei Befehle in die gleiche Zeile schreiben will, ist es notwendig, diese durch ein Semikolon voneinander zu trennen. Wenn man den JavaScript-Code nun in das HTML-Dokument einfügt, ergibt sich folgender Inhalt:

```
<html>
    <head>
        <title>
            Meine erste Javascript-Seite
        </title>
```

```
    </head>
    <body>
        <script>
        alert ("Hallo Welt!");
        </script>
    </body>
</html>
```

Damit ist das Hallo-Welt-Programm bereits abgeschlossen. Nun muss man es noch abspeichern – wie schon erwähnt immer mit der Endung .html oder .htm. Wenn man dieses anschließend im Browser aufruft, erscheint die entsprechende Nachricht in einer kleinen Info-Box:



Screenshot 11 Die Ausgabe der Seite

3.3 Kommentare für ein einfacheres Verständnis des Codes

Wenn man den Quellcode eines Programms betrachtet, ist es häufig auf den ersten Blick schwierig, die genaue Funktionsweise zu verstehen. Wenn der Programmierer den Code erstellt, weiß er selbstverständlich, welchen Zweck dieser erfüllen soll. Allerdings kommt es immer wieder vor, dass er später daran noch Veränderungen vornehmen muss. Dazwischen können mehrere Jahre vergehen, sodass er sich nicht mehr genau daran erinnert, welche Aufgabe ein bestimmter Programmteil erfüllt. Darüber hinaus kommt es immer wieder vor, dass auch andere Programmierer Veränderungen vornehmen oder Erweiterungen hinzufügen.

In diesen Fällen wäre es notwendig, den Code Zeile für Zeile zu lesen und dabei jedes Mal genau zu überlegen, welcher Zweck damit erfüllt

werden soll. Das ist jedoch sehr aufwendig und außerdem können dabei immer wieder Fehler auftreten.

Um diese Probleme zu vermeiden, ist es in fast allen Programmiersprachen möglich, Kommentare einzufügen. Dabei handelt es sich um Text, der zur Erklärung des Programmcodes dient. Er hat allerdings keinerlei Auswirkungen auf dessen Ablauf. Auf diese Weise kann der Programmierer ganz einfach festhalten, welche Aufgaben ein bestimmter Abschnitt des Programms erfüllt. Kommentare sind sehr hilfreich, um das Verständnis des Codes zu erleichtern.

Damit der eingefügte Text keinen Einfluss auf den Ablauf des Programms nimmt, ist es notwendig, ihn entsprechend zu kennzeichnen. In JavaScript gibt es hierfür zwei verschiedene Möglichkeiten. Diese sind genau die gleichen wie in Java, C++ und in vielen weiteren Programmiersprachen.

Zum einen ist es möglich, einen einzeiligen Kommentar einzufügen. Um diesen zu kennzeichnen, ist ein doppelter Schrägstrich notwendig (`//`). Es ist möglich, vor diesen Zeichen JavaScript-Code zu schreiben. Dieser wird wie gewohnt ausgeführt. Alles, was danach steht, wird bei der Ausführung jedoch nicht berücksichtigt. Es ist nicht möglich, den Kommentar abzuschließen, um danach weiteren Code anzubringen. Dieser kann erst in einer neuen Zeile stehen.

Darüber hinaus ist es möglich, mehrzeilige Kommentare zu verwenden. Diese verfügen jeweils über eine eigene Zeichenfolge, um den Anfang und das Ende zu markieren. Um den Beginn des Kommentars zu kennzeichnen, kommt der Schrägstrich gefolgt vom Stern-Symbol (`/*`) zum Einsatz. Der Schluss des Kommentars wird durch die gleichen Symbole markiert – allerdings in umgekehrter Reihenfolge (`*/`). Obwohl diese Form der Kennzeichnung normalerweise für mehrzeilige Kommentare zum Einsatz kommt, wäre es damit auch möglich, eine einzelne Zeile als Kommentar zu kennzeichnen. Man könnte sogar nur einen Teil der Zeile als Kommentar deklarieren und danach mit dem Code fortfahren. Diese Vorgehensweise ist jedoch unüblich und daher nicht zu empfehlen.

Bei der Verwendung von JavaScript-Kommentaren ist es wichtig, darauf zu achten, dass diese nur innerhalb des `script`-Tags gültig sind. Setzt man sie außerhalb, stellen sie Teil des HTML-Codes dar. Hier werden sie jedoch nicht als Kommentare erkannt, sodass der Inhalt auf der Seite erscheint. Um Kommentare innerhalb des HTML-Codes anzubringen, ist eine andere Kennzeichnung notwendig: `<!--` für den Startpunkt und `-->` für den Endpunkt. Das folgende Codebeispiel zeigt nochmals das gleiche Programm aus dem vorherigen Abschnitt – dieses Mal allerdings mit verschiedenen Kommentaren.

```
<html>
    <head>
        <title>
            Meine erste Javascript-Seite
        </title>
    </head>
    <body>
        <script>
        /*
        Dieses Programm dient als einfaches Beispiel, um eine
        Nachricht in einer Info-Box
        Auszugeben. Zu diesem Zweck stellt es den alert-Befehl vor.
        */
        alert ("Hallo Welt!");
        //Nach der Erzeugung des Fensters folgen keine weiteren
        //Inhalte.
        </script>
        <!--Hier geht der HTML-Code weiter. Daher ist ein HTML-
        Kommentar notwendig.-->
    </body>
</html>
```

Im weiteren Verlauf dieses Buchs werden immer wieder Kommentare zum Einsatz kommen. In einem einfachen Programm wie in diesem Beispiel wäre dies sicherlich notwendig. Man würde den Code auch ohne diese Hilfe problemlos verstehen. Wenn jedoch später etwas längere Programme entstehen, sind diese Anmerkungen sehr nützlich.

Kommentare spielen nicht nur eine wichtige Rolle für das Verständnis des Codes. Darüber hinaus sind sie auch während der Entwicklung von großer Bedeutung. Dabei treten immer wieder Situationen auf, in denen ein bestimmter Teil des Programms noch nicht funktionsfähig ist,

weil dafür zunächst ein weiterer Bereich programmiert werden muss. Das ist beispielsweise häufig bei der Verwendung von Funktionen der Fall. Dennoch ist es sinnvoll, die entsprechenden Programmteile bereits einzufügen, um eine klare Struktur vorzugeben. Das würde jedoch beim Ausführen zu einem Fehler führen. Um das zu vermeiden, ist es üblich, diese Bereiche als Kommentare zu kennzeichnen. So werden sie nicht berücksichtigt, sodass das Programm lauffähig bleibt. Wenn die notwendigen Funktionen dann programmiert sind, kann man einfach die Kommentar-Symbole entfernen. Diese Vorgehensweise wird als Auskommentieren bezeichnet.

3.4 JavaScript-Programme in eigene Dateien schreiben

Weiter vorne in diesem Kapitel wurde erklärt, dass JavaScript-Code stets in eine HTML-Seite eingebettet sein muss. Das trifft zwar nach wie vor zu. Allerdings bedeutet das nicht, dass er auch in der gleichen Datei stehen muss. Es ist auch möglich, für das JavaScript-Programm eine eigene Datei zu erstellen. Diese enthält dann nur die entsprechenden Befehle in dieser Programmiersprache – ganz ohne HTML-Code.

Um diese Vorgehensweise zu verdeutlichen, soll der Code aus unserem ersten Beispielprogramm nun in eine eigene Datei ausgelagert werden. Der einzige JavaScript-Befehl, der darin auftauchte, war der `alert`-Befehl. Daher ist es lediglich notwendig, diesen aus dem HTML-Dokument zu entfernen. Es ist sinnvoll, ihn auszuschneiden und daraufhin in Visual Studio Code ein neues leere Dokument zu öffnen. Hier kann man das entsprechende Kommando dann wieder einfügen. Dieses stellt den einzigen Inhalt der neu zu erstellenden Datei dar.

Wenn man eine Datei mit reinen JavaScript-Befehlen erstellt, darf man diese nicht mehr mit der Endung .html abspeichern. Anstatt dessen erhält sie eine neue Endung, die speziell für JavaScript-Programme vorgesehen ist: .js. Die Datei soll daher unter dem Namen meinScript.js abgespeichert werden. Sie enthält lediglich folgenden Befehl:

```
alert ("Hallo Welt!");
```

Nun muss man noch einige Änderungen im HTML-Dokument vornehmen. Dieses soll nach wie vor den Rahmen der Seite vorgeben. Das `script`-Tag bleibt nun jedoch leer. Dafür wird es mit einem weiteren Attribut versehen: `src="meinScript.js"`. Dieses sorgt dafür, dass die eben erstellte Datei meinScript.js in das HTML-Dokument eingebunden wird. Voraussetzung hierfür ist, dass sie sich im gleichen Verzeichnis wie die HTML-Datei befindet. Ist das nicht der Fall, muss man noch den entsprechenden Pfadnamen angeben.

Dieses Attribut bewirkt, dass die hier genannte Datei eingelesen wird. Alle Befehle, die darin enthalten sind, werden sofort ausgeführt. Wenn man die neu gestaltete HTML-Datei nun aufruft, erscheint genau wie im bisherigen Programm die Info-Box mit der entsprechenden Ausgabe, obwohl die Datei selbst keinen JavaScript-Befehl enthält. Der vollständige Code sieht dann so aus:

```
<html>
    <head>
        <title>
            Meine erste Javascript-Seite
        </title>
    </head>
    <body>
        <script src="meinScript.js">
        </script>
    </body>
</html>
```

Wenn es sich um kleine Programme wie in diesem Beispiel handelt, ist diese Technik nicht allzu sinnvoll. Hier wäre es einfacher, den JavaScript-Code einfach in die HTML-Datei zu schreiben. Bei längeren Dateien, die mehrere JavaScript-Bestandteile enthalten, ist das jedoch sehr zu empfehlen. Auf diese Weise kann man die einzelnen Bereiche klar trennen und dadurch die Übersichtlichkeit erhöhen.

Die Auslagerung des JavaScript-Codes in separate Dateien bietet noch einen weiteren Vorteil. Auf diese Weise lässt er sich ganz einfach wiederverwenden. Wenn an einer anderen Stelle die gleiche Aufgabe er-

neut ausgeführt werden soll, muss der Programmierer die Datei lediglich nochmals aufrufen.

3.5 "use strict": modernen JavaScript-Code erstellen

Wie bei fast allen Programmiersprachen kam es auch bei JavaScript im Laufe der Jahre immer wieder zu einigen Neuerungen. Auf diese Weise wird die Programmiersprache an neue Entwicklungen angepasst. In vielen Fällen dienen die neuen Versionen auch dazu, Fehler aus den bisherigen Funktionen zu beseitigen und einige falsche Entscheidungen, die in der Vergangenheit getroffen wurden, wieder rückgängig zu machen.

Bei Javascript war das lange Zeit jedoch nicht der Fall. Zwar kam es auch hier mit der Zeit zu einigen Veränderungen. Dabei handelte es sich jedoch ausschließlich um Erweiterungen. Fehler und schlecht implementierte Funktionen wurden hingegen nicht verbessert. Der Grund dafür bestand darin, dass die Entwickler viel Wert auf eine abwärtskompatible Programmiersprache legten. Das bedeutet, dass auch Programme, die mit allen vorherigen Versionen erstellt wurden, lauffähig bleiben sollten. Wenn man hingegen Fehler verbessert, kann das dazu führen, dass Programme, die die entsprechenden Funktionen verwenden, plötzlich nicht mehr funktionsfähig sind.

Die Abwärtskompatibilität stellt zwar einen großen Vorteil dar. Auf diese Weise müssen die Webentwickler ältere Programme nicht abändern, wenn es zu einer Aktualisierung kommt. Allerdings hat sie ihren Preis: Auf diese Weise bleiben fehlerhafte Stellen dauerhaft in der Programmiersprache bestehen.

Aus diesem Grund beschlossen die Entwickler im Jahre 2009, eine neue JavaScript-Version zu erstellen, die alle bekannten Schwachstellen aus der Programmiersprache entfernen sollte. Auf diese Weise kam es zur Veröffentlichung von ECMA Script 5 (ES5). Dabei handelt es sich um eine stark verbesserte JavaScript-Version, die viele Fehler beseitigte.

Allerdings war sie nicht mehr abwärtskompatibel. Das bedeutet, dass viele ältere Programme damit nicht mehr ausgeführt werden können.

Das widersprach den bisherigen Grundsätzen von JavaScript. Aus diesem Grund wird diese neue Version bei der Ausführung eines JavaScript-Programms nicht automatisch berücksichtigt. Der Browser verwendet dabei zunächst immer die herkömmliche Version, die auch die Ausführung älterer Programme ermöglicht.

Wenn man ein neues Programm schreibt, spielt die Abwärtskompatibilität jedoch keine Rolle. Daher ist es sinnvoll, stets die neuere Ausführung zu wählen, die moderne Strukturen unterstützt und in der viele fehlerhafte Stellen beseitigt wurden. Allerdings ist es dafür notwendig, dem Browser zunächst mitzuteilen, dass er diese Version verwenden soll. Das geschieht, indem man den Befehl `"use strict";` in den Code einfügt. Anstatt in doppelten Anführungszeichen kann dieser auch in einfachen Anführungszeichen stehen.

Dabei muss man beachten, dass dieser Befehl stets ganz am Anfang des JavaScript-Codes eingefügt werden muss. Vor ihm dürfen keine anderen Kommandos stehen. Lediglich Kommentare sind hier erlaubt. Außerdem ist es nicht möglich, diesen Befehl rückgängig zu machen. Wenn er zu Beginn eines Dokuments steht, ist er für die gesamte Datei gültig.

Dabei gibt es jedoch eine Ausnahme. Im weiteren Verlauf dieses Buchs werden noch Funktionen vorgestellt. Wenn man den `use-strict-Befehl` zu Beginn einer Funktion einfügt, ist er nur in diesem Bereich gültig.

Es ist empfehlenswert, neue Programme in dieser modernen Version zu verfassen. Auch dieses Lehrbuch wird noch zahlreiche Befehle vorstellen, die nur unter Verwendung des `use-strict`-Befehls verfügbar sind. In einfachen Programmen wie in unserem ersten Beispiel, macht es jedoch keinen Unterschied, ob diese Ergänzung vorhanden ist. Dennoch ist es sinnvoll, sich deren Verwendung von Anfang an anzugewöhnen. Daher soll er fortan zu Beginn aller JavaScript-Programme stehen.

Wenn man das erste Beispielprogramm entsprechend abändert, sieht es wie folgt aus:

```
<html>
    <head>
        <title>
            Meine erste Javascript-Seite
        </title>
    </head>
    <body>
        <script>
        "use strict";
        alert ("Hallo Welt!");
        </script>
    </body>
</html>
```

Diese Änderung hat in diesem Fall jedoch keinerlei Auswirkungen auf den Ablauf des Programms.

3.6 Eine Eingabe des Anwenders aufnehmen

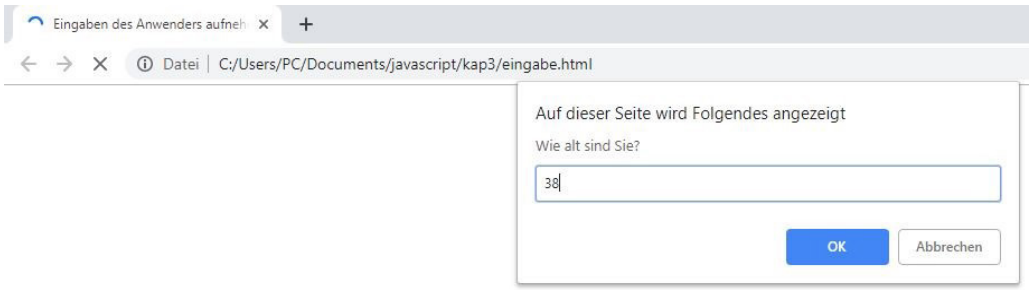
Eine wichtige Eigenschaft von Computerprogrammen besteht darin, dass sie eine Interaktion mit dem Anwender ermöglichen. Auf diese Weise können sie benutzerdefinierte Werte aufnehmen und diese in den nachfolgenden Befehlen weiterhin verwenden. So entsteht ein ganz individueller Ablauf. Die Verwendung und Verarbeitung von Eingaben des Anwenders stellt auch einen der wesentlichen Unterschiede zwischen JavaScript-Seiten und statischen HTML-Seiten dar.

Aus diesem Grund sind diese Interaktionsmöglichkeiten ein zentrales Element jeder Programmiersprache. JavaScript bietet dafür sehr viele Möglichkeiten. Viele davon verwenden jedoch fortgeschrittene Funktionen, die erst im weiteren Verlauf dieses Buchs vorgestellt werden. Es gibt aber auch eine ganz einfache Möglichkeit: den `prompt`-Befehl. Dieser erzeugt ein neues Fenster, das ähnlich aufgebaut ist wie beim `alert`-Befehl. Allerdings enthält dieses ein Eingabefeld. Hier kann der Anwender beliebige Werte eintragen. Um das Eingabefenster zu erzeugen, muss man den Befehl `prompt` in das Programm schreiben. Da-

nach folgt eine Klammer mit einem Text in Anführungszeichen. Hier kann man eine Frage oder eine Handlungsanweisung eintragen. Wenn man beispielsweise das Alter des Anwenders abfragen will, könnte die entsprechende Code-Zeile so aussehen:

```
prompt ("Wie alt sind Sie?");
```

Wenn man diesen nun in den bisherigen Programmcode anstelle des `alert`-Befehls einfügt, erscheint bereits ein entsprechendes Eingabefenster, in das man einen Wert eintragen kann:



Screenshot 12 Das Eingabefenster

Nun kann man zwar bereits einen Wert eingeben, doch hat dies noch keinerlei Auswirkungen auf das Programm. Daher ist es notwendig, ihn auch an einer anderen Stelle wieder auszugeben. Häufig wird der Wert, der im `prompt`-Fenster eingelesen wird, in einer Variablen abgespeichert. Die Verwendung von Variablen wird jedoch erst im nächsten Kapitel vorgestellt. Daher soll hier eine andere Alternative zum Einsatz kommen. In diesem Programm wird der `prompt`-Befehl direkt in einen `alert`-Befehl eingefügt. Auf diese Weise gibt das Programm den entsprechenden Wert sofort wieder aus:

```
alert (prompt ("Wie alt sind Sie?"));
```

Wenn man das Programm nun erneut ausführt, erscheint zunächst wieder das Eingabefenster. Wenn man dieses ausfüllt und auf "OK" drückt, öffnet sich sofort ein weiteres Fenster, das durch den `alert`-Be-

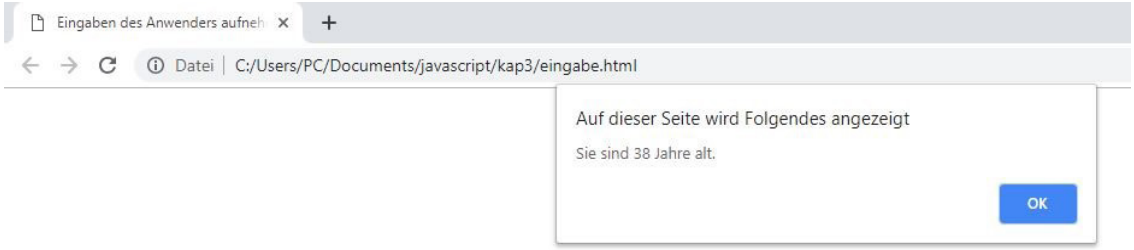
fehl erzeugt wird. Dieses enthält die Zahl, die der Nutzer eingegeben hat.

Wenn man nur die Zahl ausgibt, ist die Ausgabe jedoch schwer verständlich. Daher ist es sinnvoll, sie mit etwas Text zu garnieren. Diesen muss man in Anführungszeichen setzen und kann ihn dann ebenfalls in den `alert`-Befehl einfügen. Um ihn mit dem Wert aus dem `prompt`-Befehl zusammenzusetzen, kommt das Pluszeichen zum Einsatz:


```
alert ("Sie sind " + prompt ("Wie alt sind Sie?") + " Jahre alt.");
```

Wenn man das Programm nun erneut ausführt, erscheint eine ansprechende Ausgabe mit dem eingegebenen Alter. Der vollständige Programmcode hierfür sieht so aus:

```
<html>
    <head>
        <title>
            Eingaben des Anwenders aufnehmen
        </title>
    </head>
    <body>
        <script>
        "use strict";
        alert ("Sie sind " + prompt ("Wie alt sind Sie?") + " Jahre
        alt.");
        </script>
    </body>
</html>
```



Screenshot 13 Das Ausgabefenster

3.7 Übungsaufgabe: einfache JavaScript-Programme selbst erstellen

Um das Programmieren zu erlernen, ist es sehr wichtig, selbst Programme zu erstellen. Auf diese Weise lernt man die wesentlichen Funktionen kennen und eignet sich eine große Routine bei der Bearbeitung der entsprechenden Aufgaben an. Wenn Sie die Programmbeispiele aus den bisherigen Abschnitten abgetippt und auf Ihrem Rechner ausgeführt haben, haben Sie dabei bereits den ersten Schritt unternommen. Der zweite Schritt besteht darin, die Programme von Grund auf selbst zu erstellen. Wenn man sich selbst überlegt, wie man ein bestimmtes Problem lösen könnte und daraufhin ein passendes Programm erstellt, erweitert man die eigenen Fähigkeiten deutlich.

Aus diesen Gründen sind am Ende der meisten Kapitel einige Übungsaufgaben aufgeführt. Diese greifen die Inhalte der vorherigen Abschnitte auf, um sie zu vertiefen. Allerdings werden auch die Kenntnisse aus den vorherigen Kapiteln als bekannt vorausgesetzt. Oftmals sind sie für die Bearbeitung der entsprechenden Aufgaben ebenfalls notwendig. Sollten neue Befehle erforderlich sein, werden diese kurz erklärt.

Die Aufgaben sind stets mit einer Musterlösung versehen. Allerdings ist es ratsam, diese erst zu betrachten, wenn man ein eigenes Programm für die entsprechende Aufgabe erstellt hat. Nur wenn Sie alleine überhaupt nicht weiterkommen, sollten Sie hier nachschauen. Darüber hinaus sollten Sie beachten, dass es in der Informatik fast immer mehrere Lösungswege gibt. Die Musterlösung stellt daher nur eine von vielen Möglichkeiten dar. Um das Problem richtig zu lösen, ist es nicht notwendig, dass der Code genau übereinstimmt. Wenn Ihr Programm alle Anforderungen erfüllt, ist die Aufgabe korrekt bearbeitet – selbst wenn es dabei zu Abweichungen von der Musterlösung kommen sollte.

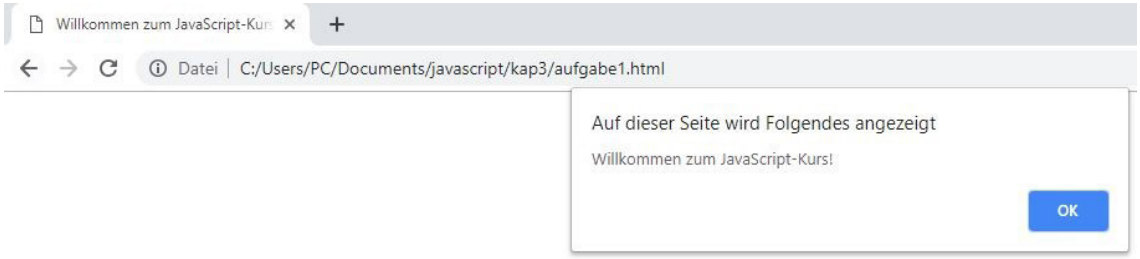
1. Schreiben Sie ein Programm, das den Besucher zum JavaScript-Kurs begrüßt.

2. Fragen Sie den Besucher nach seinem Namen und erstellen Sie eine personalisierte Begrüßung
3. Erstellen Sie ein Programm, das die gleichen Aufgaben erfüllt wie in Aufgabe 2. Allerdings soll der JavaScript-Code nun in eine externe Datei ausgelagert werden. Erstellen Sie sowohl die HTML- als auch die JavaScript-Datei.

Lösungen:

1.

```
<html>
    <head>
        <title>
            Willkommen zum JavaScript-Kurs!
        </title>
    </head>
    <body>
        <script>
        "use strict";
        alert ("Willkommen zum JavaScript-Kurs!");
        </script>
    </body>
</html>
```

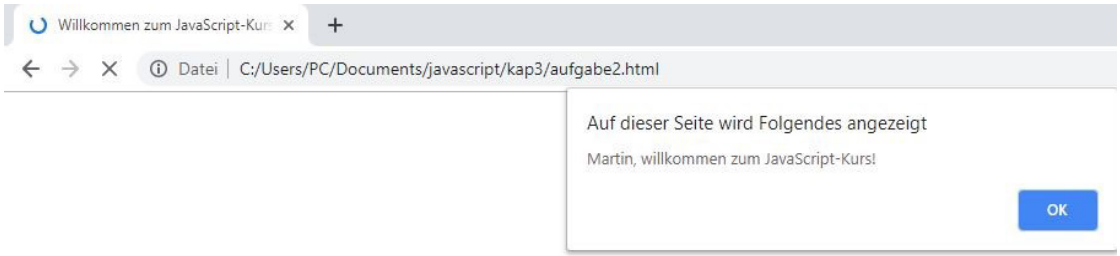


Screenshot 14 Die Ausgabe der Begrüßung

2.

```
<html>
    <head>
        <title>
            Willkommen zum JavaScript-Kurs!
        </title>
    </head>
    <body>
        <script>
        "use strict";
        alert (prompt ("Ihr Name:") + ", willkommen zum
        JavaScript-Kurs!");
```

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



Screenshot 15 Die individualisierte Begrüßung

3.

HTML-Datei:

```
<html>
    <head>
        <title>
            Willkommen zum JavaScript-Kurs!
        </title>
    </head>
    <body>
        <script src = "begruessung.js">
        </script>
    </body>
</html>
```

JavaScript-Datei:

```
"use strict";
alert (prompt ("Ihr Name:") + ", willkommen zum JavaScript-Kurs!");
```

Alle Programmcodes aus diesem Buch sind als PDF zum Download verfügbar. Dadurch müssen Sie sie nicht abtippen:
https://bmu-verlag.de/books/javascript/

Außerdem erhalten Sie die eBook Ausgabe zum Buch im PDF Format kostenlos auf unserer Website:

https://bmu-verlag.de/books/javascript/
Downloadcode: siehe Kapitel 20