

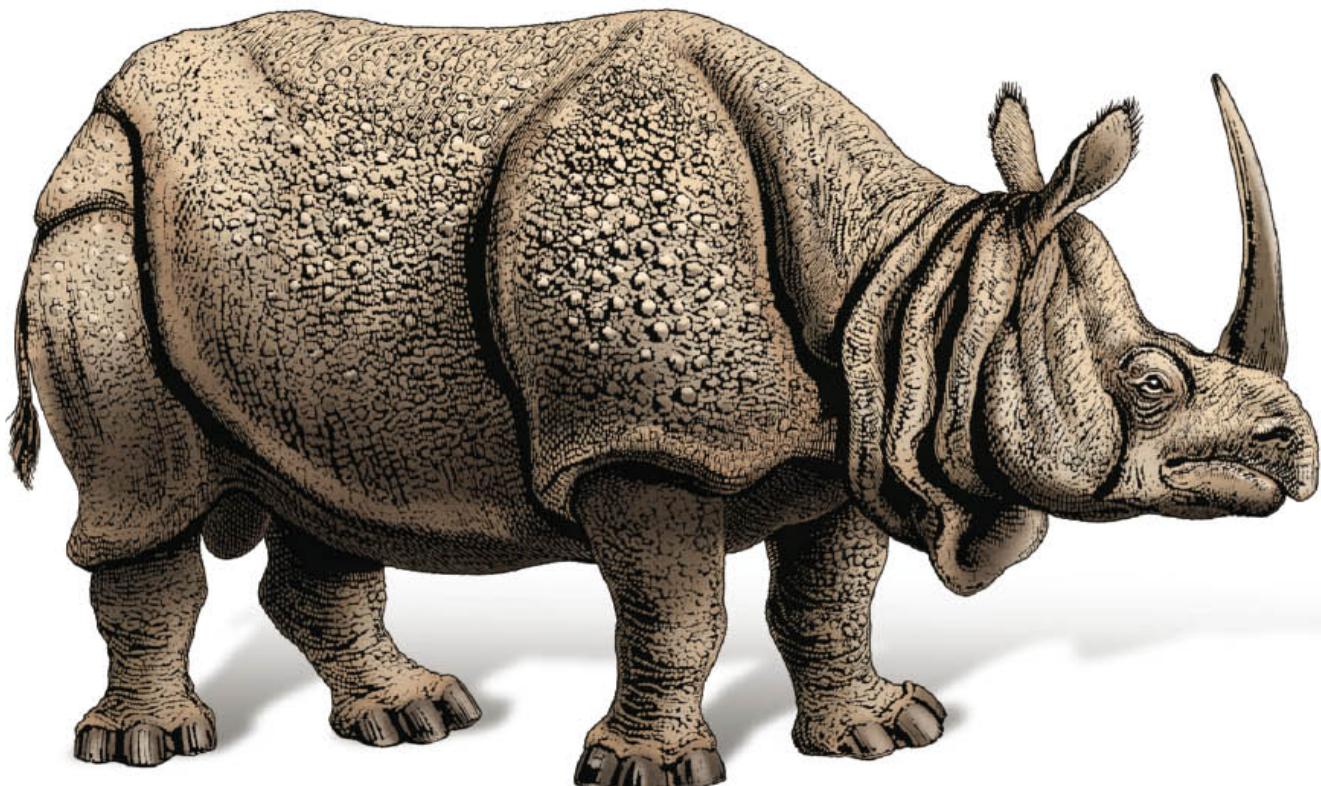
O'REILLY®

Der
Klassiker
in der
1. Auflage

JavaScript

Das Handbuch für die Praxis

Meistern Sie die beliebte Sprache
für Web und Node.js



David Flanagan
Übersetzung von Jørgen W. Lang
und Jens Olaf Koch

Lob für *JavaScript: Das Handbuch für die Praxis*, siebte Auflage

»In diesem Buch erfahren Sie Dinge über JavaScript, von denen Sie bisher nicht einmal geahnt hatten, wie dringend Sie sie wissen wollten. Die Qualität Ihres Codes und Ihre Produktivität werden extrem davon profitieren. Davids Wissen über JavaScript, dessen Feinheiten und typische Fehlerquellen ist verblüffend, und seine Qualitäten sind in diesem wirklich außergewöhnlichen Handbuch überall spürbar.«

- *Schalk Neethling, Senior Frontend Engineer bei MDN Web Docs*

»David Flanagan nimmt die Leser mit auf eine Führung durch die JavaScript-Welt und vermittelt Ihnen ein komplettes Bild aller Sprachfeatures und des gesamten Ökosystems.«

- *Sarah Wachs, Frontend Developer und Women Who Code Berlin Lead*

»Für alle Entwicklerinnen und Entwickler, die produktiv an Codebasen aus unterschiedlichsten Entwicklungsstadien

von JavaScript arbeiten wollen (einschließlich der neuesten Funktionen), wird es sich auszahlen, sich intensiv auf dieses umfassende und maßgebliche Buch einzulassen.«

- Brian Sletten, Präsident von Bosatsu Consulting



Zu diesem Buch – sowie zu vielen weiteren O'Reilly-Büchern – können Sie auch das entsprechende E-Book im PDF-Format herunterladen. Werden Sie dazu einfach Mitglied bei oreilly.plus⁺:

www.oreilly.plus

7. AUFLAGE

JavaScript - Das Handbuch für die Praxis

*Meistern Sie die beliebte Sprache
für Web und Node.js*

David Flanagan

*Deutsche Übersetzung von
Jørgen W. Lang und Jens Olaf Koch*

O'REILLY®

David Flanagan

Lektorat: Ariane Hesse

Übersetzung: Jørgen W. Lang und Jens Olaf Koch

Korrektorat: Sibylle Feldmann, www.richtiger-text.de

Satz: III-satz, www.drei-satz.de

Herstellung: Stefanie Weidner

Umschlaggestaltung: Karen Montgomery, Michael Oreal, www.oreal.de

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im
Internet über <http://dnb.d-nb.de> abrufbar.

ISBN:

Print 978-3-96009-157-8

PDF 978-3-96010-491-9

ePub 978-3-96010-492-6

mobi 978-3-96010-493-3

7. Auflage 2021

Translation Copyright für die deutschsprachige Ausgabe © 2021 dpunkt.verlag
GmbH

Wieblinger Weg 17
69123 Heidelberg

© 2021 dpunkt.verlag GmbH

Authorized German translation of the English edition of »JavaScript: The
Definitive Guide«, 7E, ISBN 9781491952023 © 2020 David Flanagan.

This translation is published and sold by permission of O'Reilly Media, Inc.,
which owns or controls all rights to publish and sell the same.

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem
Imprint »O'REILLY«.

O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly
Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.



Hinweis:

Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

Schreiben Sie uns:

Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: kommentar@oreilly.de.

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag noch Übersetzer können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

Meinen Eltern, Donna und Matt, voller Liebe und Dankbarkeit.

Inhalt

Vorwort

1 Einführung in JavaScript

- 1.1 JavaScript erkunden
- 1.2 Hello World
- 1.3 Ein Rundgang durch JavaScript
- 1.4 Beispiel: Häufigkeitshistogramme
- 1.5 Zusammenfassung

2 Die lexikalische Struktur

- 2.1 Der Text eines JavaScript-Programms
- 2.2 Kommentare
- 2.3 Literale
- 2.4 Identifier und reservierte Wörter
- 2.5 Unicode
- 2.6 Optionale Semikola
- 2.7 Zusammenfassung

3 Typen, Werte und Variablen

- 3.1 Übersicht und Definitionen
- 3.2 Zahlen
- 3.3 Text
- 3.4 Boolesche Werte
- 3.5 null und undefined
- 3.6 Symbole
- 3.7 Das globale Objekt

- 3.8 Unveränderbare primitive Werte und veränderbare Objektreferenzen
- 3.9 Typumwandlungen
- 3.10 Variablen-deklaration und -zuweisung
- 3.11 Zusammenfassung

4 Ausdrücke und Operatoren

- 4.1 Elementare Ausdrücke
- 4.2 Initialisierungsausdrücke von Objekten und Arrays
- 4.3 Ausdrücke zur Funktionsdefinition
- 4.4 Ausdrücke für den Eigenschaftszugriff
- 4.5 Aufrufausdrücke
- 4.6 Ausdrücke zur Objekterstellung
- 4.7 Operatoren im Überblick
- 4.8 Arithmetische Ausdrücke
- 4.9 Relationale Ausdrücke
- 4.10 Logische Ausdrücke
- 4.11 Zuweisungsausdrücke
- 4.12 Auswertungsausdrücke
- 4.13 Weitere Operatoren
- 4.14 Zusammenfassung

5 Anweisungen

- 5.1 Anwendungsausdrücke
- 5.2 Zusammengesetzte und leere Anweisungen
- 5.3 Bedingungen
- 5.4 Schleifen
- 5.5 Sprünge
- 5.6 Verschiedene Anweisungen
- 5.7 Deklarationen
- 5.8 Zusammenfassung

6 Objekte

- 6.1 Einführung in Objekte
- 6.2 Objekte erstellen

- 6.3 Eigenschaften abfragen und zuweisen
- 6.4 Eigenschaften löschen
- 6.5 Eigenschaften prüfen
- 6.6 Eigenschaften aufzählen
- 6.7 Objekte erweitern
- 6.8 Objekte serialisieren
- 6.9 Objektmethoden
- 6.10 Erweiterte Syntax für Objektliterale
- 6.11 Zusammenfassung

7 Arrays

- 7.1 Arrays erstellen
- 7.2 Array-Elemente lesen und schreiben
- 7.3 Sparse-Arrays
- 7.4 Array-Länge
- 7.5 Array-Elemente hinzufügen und löschen
- 7.6 Über Arrays iterieren
- 7.7 Mehrdimensionale Arrays
- 7.8 Array-Methoden
- 7.9 Arrayartige Objekte
- 7.10 Strings als Arrays
- 7.11 Zusammenfassung

8 Funktionen

- 8.1 Funktionen definieren
- 8.2 Funktionen aufrufen
- 8.3 Funktionsargumente und -parameter
- 8.4 Funktionen als Werte
- 8.5 Funktionen als Namensräume
- 8.6 Closures
- 8.7 Funktionseigenschaften, -methoden und -konstruktoren
- 8.8 Funktionale Programmierung
- 8.9 Zusammenfassung

9 Klassen

- 9.1 Klassen und Prototypen
- 9.2 Klassen und Konstruktoren
- 9.3 Klassen erstellen mit dem Schlüsselwort `class`
- 9.4 Existierende Klassen um Methoden erweitern
- 9.5 Subklassen
- 9.6 Zusammenfassung

10 Module

- 10.1 Module mit Klassen, Objekten und Closures
- 10.2 Module in Node
- 10.3 Module in ES6
- 10.4 Zusammenfassung

11 Die JavaScript-Standardbibliothek

- 11.1 Sets und Maps
- 11.2 Typisierte Arrays und binäre Daten
- 11.3 Mustererkennung mit regulären Ausdrücken
- 11.4 Datum und Uhrzeit
- 11.5 Fehlerklassen
- 11.6 Serialisierung und Parsing mit JSON
- 11.7 Die Internationalisierungs-API
- 11.8 Die Console-API
- 11.9 URL-APIs
- 11.10 Timer
- 11.11 Zusammenfassung

12 Iteratoren und Generatoren

- 12.1 Wie Iteratoren funktionieren
- 12.2 Iterierbare Objekte implementieren
- 12.3 Generatoren
- 12.4 Erweiterte Generatorfunktionen
- 12.5 Zusammenfassung

13 Asynchrones JavaScript

- 13.1 Asynchrone Programmierung mit Callbacks
- 13.2 Promises

- 13.3 `async` und `await`
- 13.4 Asynchrone Iteration
- 13.5 Zusammenfassung

14 Metaprogrammierung

- 14.1 Eigenschaftsattribute
- 14.2 Objekte erweitern
- 14.3 Das prototype-Attribut
- 14.4 Wohlbekannte Symbole
- 14.5 Template-Tags
- 14.6 Die Reflect-API
- 14.7 Proxy-Objekte
- 14.8 Zusammenfassung

15 JavaScript im Webbrowser

- 15.1 Grundlagen der Webprogrammierung
- 15.2 Events
- 15.3 Dokumente skripten
- 15.4 CSS skripten
- 15.5 Dokumentgeometrie und Scrolling
- 15.6 Webkomponenten
- 15.7 SVG: Scalable Vector Graphics
- 15.8 Grafiken auf einem `<canvas>`
- 15.9 Audio-APIs
- 15.10 Location-Objekt, Navigation und Browserverlauf
- 15.11 Netzwerkoperationen
- 15.12 Clientseitige Speicherung
- 15.13 Worker-Threads und Messaging
- 15.14 Beispiel: Die Mandelbrot-Menge
- 15.15 Zusammenfassung und Vorschläge für die weitere Lektüre

16 Serverseitiges JavaScript mit Node

- 16.1 Grundlagen der Node-Programmierung
- 16.2 Node ist standardmäßig asynchron
- 16.3 Buffer

- 16.4 Events und EventEmitter
- 16.5 Datenströme
- 16.6 Details zu Prozessen, CPU und Betriebssystem
- 16.7 Mit Dateien arbeiten
- 16.8 HTTP-Clients und -Server
- 16.9 Nicht-HTTP-basierte Netzwerkserver und -clients
- 16.10 Mit Kindprozessen arbeiten
- 16.11 Worker-Threads
- 16.12 Zusammenfassung

17 JavaScript-Werkzeuge und -Erweiterungen

- 17.1 Linting mit ESLint
- 17.2 JavaScript-Formatierung mit Prettier
- 17.3 Unit-Tests mit Jest
- 17.4 Paketverwaltung mit npm
- 17.5 Code-Bundling
- 17.6 Transpilierung mit Babel
- 17.7 JSX: Markup-Ausdrücke in JavaScript
- 17.8 Typüberprüfung mit Flow
- 17.9 Zusammenfassung

Index

Vorwort

Dieses Buch behandelt die Programmiersprache JavaScript und die von Webbrowsern und von Node implementierten JavaScript-APIs. Ich habe es für Leser geschrieben, die bereits etwas Programmiererfahrung haben und JavaScript lernen wollen. Es ist aber auch für Programmierer gedacht, die JavaScript bereits verwenden, ihr Wissen darüber aber erweitern und die Sprache wirklich beherrschen wollen. In diesem Buch möchte ich JavaScript umfassend und eindeutig dokumentieren und detailliert in die wichtigsten client- und serverseitigen APIs einführen, die für JavaScript-Programme zur Verfügung stehen. Deshalb ist es ein langes und ausführliches Buch. Ich hoffe aber, dass sein gründliches Studium belohnt wird und sich die Zeit, die Sie mit der Lektüre verbringen, in Form einer höheren Produktivität beim Programmieren schnell wieder bezahlt macht.

Frühere Ausgaben dieses Buchs enthielten einen umfassenden Referenzteil. Ich halte es aber nicht mehr für sinnvoll, dieses Material in gedruckter Form bereitzustellen, weil man heutzutage aktuelles Referenzmaterial jederzeit online findet. Wenn Sie etwas nachschlagen müssen, das mit dem Sprachkern oder clientseitigem JavaScript zu tun hat, empfehle ich Ihnen, die MDN-Website (<https://developer.mozilla.org>) zu

besuchen. Für serverseitige Node-APIs sollten Sie direkt zur Quelle gehen und die Node.js-Referenzdokumentation (<https://nodejs.org/api>) konsultieren.

Konventionen, die in diesem Buch verwendet werden

In diesem Buch verwende ich die folgenden typografischen Konventionen:

Kursiv

Wird zur Hervorhebung und zur Angabe der ersten Verwendung eines Begriffs, aber auch für E-Mail-Adressen, URLs und Dateinamen verwendet.

Nicht proportional

Wird in JavaScript-Code sowie in CSS- und HTML-Listings verwendet – und im Allgemeinen für alles, was Sie beim Programmieren in genau der angegebenen Form per Tastatur eingeben würden.

Nicht proportional kursiv

Wird gelegentlich bei der Erläuterung der JavaScript-Syntax verwendet.

Nicht proportional fett

Zeigt Befehle oder anderen Text an, der vom Benutzer wortgetreu eingegeben werden muss.



Dieses Element kennzeichnet einen allgemeinen Hinweis.

Dieses Element weist auf eine Warnung hin.



Beispielcode

Ergänzendes Material (Codebeispiele, Übungen usw.) für dieses Buch stehen zum Download bereit unter:

- https://oreil.ly/javascript_defgd7

Dieses Buch soll Sie bei der Erledigung Ihrer Aufgaben unterstützen. Wenn in diesem Buch Beispielcode angeboten wird, können Sie diesen in Ihren Programmen und Ihrer Dokumentation verwenden. Sie brauchen uns nicht um Erlaubnis zu bitten, es sei denn, Sie reproduzieren einen wesentlichen Teil des Codes. Beispielsweise erfordert das Schreiben eines Programms, das mehrere Codeblöcke aus diesem Buch verwendet, keine Genehmigung. Der Verkauf oder Vertrieb von Beispielen aus O'Reilly-Büchern benötigt dagegen eine Genehmigung. Für die Beantwortung einer Frage, indem aus diesem Buch oder dem Beispielcode zitiert wird, ist keine Genehmigung erforderlich. Das Einbinden einer beträchtlichen Menge an Beispielcode aus diesem Buch in die Dokumentation Ihres Produkts erfordert dagegen eine Genehmigung.

Wir freuen uns über eine Quellenangabe, verlangen sie aber im Allgemeinen nicht. Eine Quellenangabe umfasst in der Regel den Titel, den Autor, den Verlag und die ISBN, zum Beispiel: »*JavaScript: The Definitive Guide*, siebte Auflage, von David Flanagan (O'Reilly). Copyright 2020 David Flanagan, ISBN 9781491952023«.

Wenn Sie glauben, dass Ihre Verwendung der Codebeispiele über einen fairen Gebrauch oder die oben erteilten Genehmigungen hinausgeht, können Sie uns unter permissions@oreilly.com gern kontaktieren.

O'Reilly Online-Lernen



Seit mehr als 40 Jahren bietet *O'Reilly Media* (<https://oreilly.com>) Technologie- und Businesstraining sowie Wissen und Einsichten, um Unternehmen zum Erfolg zu verhelfen.

Unser einzigartiges Netzwerk aus Experten und Innovatoren teilt sein Wissen und seine Expertise in Büchern, Artikeln und auf unserer Onlinelernplattform. Die Onlinelernplattform von O'Reilly bietet Ihnen On-Demand-Zugriff auf Live-Schulungen, detaillierte Lernpfade, interaktive Codierumgebungen sowie eine umfangreiche Sammlung an Texten und Videos von O'Reilly und über 200 anderen Verlagen. Für weitere Informationen besuchen Sie bitte <https://oreilly.com>.

Danksagungen

An der Entstehung dieses Buchs waren viele Personen beteiligt. Ich möchte meiner Lektorin Angela Rufino dafür danken, dass sie mich auf Kurs gehalten hat, und für ihre Geduld angesichts meiner versäumten Termine. Danke auch an meine technischen Gutachter Brian Sletten, Elisabeth Robson, Ethan Flanagan, Maximiliano Firtman, Sarah Wachs und Schalk Neethling. Ihre Kommentare und Vorschläge haben deutlich zur Qualität dieses Buchs beigetragen.

Das Produktionsteam von O'Reilly hat seine gewohnt gute Arbeit geleistet: Kristen Brown leitete den

Produktionsprozess, Deborah Baker fungierte als Produktionsredakteurin, Rebecca Demarest war für die Abbildungen zuständig und Judy McConville für den Index.

Zu den Lektorinnen, Gutachtern und Mitwirkenden früherer Ausgaben dieses Buchs gehören: Andrew Schulman, Angelo Sirigos, Aristoteles Pagaltzis, Brendan Eich, Christian Heilmann, Dan Shafer, Dave C. Mitchell, Deb Cameron, Douglas Crockford, Dr. Tankred Hirschmann, Dylan Schiemann, Frank Willison, Geoff Stearns, Herman Venter, Jay Hodges, Jeff Yates, Joseph Kesselman, Ken Cooper, Larry Sullivan, Lynn Rollins, Neil Berkman, Mike Loukides, Nick Thompson, Norris Boyd, Paula Ferguson, Peter-Paul Koch, Philippe Le Hegaret, Raffaele Cecco, Richard Yaker, Sanders Kleinfeld, Scott Furman, Scott Isaacs, Shon Katzenberger, Terry Allen, Todd Ditchendorf, Vidur Apparao, Waldemar Horwat und Zachary Kessin.

Das Schreiben dieser siebten Ausgabe hielt mich viele lange Nächte von meiner Familie fern, bei der ich mich in Liebe dafür bedanken möchte, dass sie meine Abwesenheiten ausgehalten hat.

- *David Flanagan, im März 2020*

KAPITEL 1

Einführung in JavaScript

JavaScript ist *die* Programmiersprache des Webs. Der weitaus größte Teil moderner Websites nutzt JavaScript, und alle modernen Webbrowsers – auf Desktops, Tablets und Smartphones – besitzen JavaScript-Interpreter, was JavaScript zur am häufigsten eingesetzten Programmiersprache überhaupt macht. In den letzten zehn Jahren hat Node.js die JavaScript-Programmierung auch außerhalb von Webbrowsern ermöglicht, und der dramatische Erfolg von Node hat dazu geführt, dass JavaScript nun auch die unter Softwareentwicklern am häufigsten verwendete Programmiersprache ist. Egal ob Sie bei null anfangen oder JavaScript bereits professionell einsetzen, dieses Buch wird Ihnen dabei helfen, diese Sprache zu beherrschen.

Wenn Sie schon mit anderen Programmiersprachen vertraut sind, ist es vielleicht hilfreich, zu wissen, dass es sich bei JavaScript um eine dynamische, interpretierte Hochsprache handelt, die für objektorientierte wie auch funktionale Programmierstile sehr gut geeignet ist. Die Variablen von JavaScript sind untypisiert. Ihre Syntax basiert lose auf Java, aber die Sprachen sind ansonsten nicht miteinander verwandt. JavaScript leitet seine

erstklassigen Funktionen von Scheme und seine prototypenbasierte Vererbung von der wenig bekannten Sprache Self ab. Sie müssen aber diese Sprachen weder kennen noch auch nur mit den Begriffen vertraut sein, um dieses Buch verwenden und JavaScript erlernen zu können.

Der Name »JavaScript« ist leicht irreführend. Abgesehen von einer oberflächlichen Ähnlichkeit der Syntax unterscheidet sich JavaScript komplett von Java. Darüber hinaus ist JavaScript längst über seine Wurzeln als Skriptsprache hinausgewachsen und zu einer robusten und effizienten Allzwecksprache geworden, die sich für ernsthaftes Softwareengineering und für Projekte mit riesigen Codebasen eignet.

Damit man eine Sprache nutzen kann, muss sie eine Plattform- bzw. Standardbibliothek besitzen, um zum Beispiel die Ein- und Ausgabe zu ermöglichen. Der Sprachkern von JavaScript legt eine minimale API für den Umgang mit Zahlen, Text, Arrays, Sets, Maps usw. fest, besitzt aber keine Ein- und Ausgabefunktionalität. Diese (und fortgeschrittenere Features der Bereiche Netzwerk, Speicher und Grafik) liegen ganz in der Verantwortung der *Hostumgebung*, in die JavaScript eingebettet ist.

JavaScript: Namen, Versionen und Modi

JavaScript wurde in den frühen Tagen des Webs bei Netscape entwickelt, und die Bezeichnung *JavaScript* selbst ist ein von Sun Microsystems (jetzt Oracle) lizenziertes Warenzeichen, das zur Beschreibung der Implementierung der Sprache durch Netscape (jetzt Mozilla) verwendet wird. Netscape hatte die Sprache zur Standardisierung bei der ECMA eingereicht – der *European Computer Manufacturers Association* –, und aufgrund von Markenrechtsproblemen erhielt die standardisierte Version der Sprache den sperrigen Namen »ECMAScript«. In der Praxis spricht aber eigentlich jeder von JavaScript. In diesem Buch werden der Name *ECMAScript* und die Abkürzung *ES* benutzt, um speziell auf den Sprachstandard und auf bestimmte Versionen dieses Standards hinzuweisen.

In den 2010er-Jahren wurde von allen Webbrowsern überwiegend Version 5 des ECMAScript-Standards unterstützt. Deshalb wird in diesem Buch *ES5* als die Kompatibilitäts-Baseline betrachtet, sodass frühere Versionen der Sprache nicht behandelt werden. ES6 wurde 2015 veröffentlicht und fügte wichtige neue Funktionen – darunter die Klassen- und Modulsyntax – hinzu, die JavaScript von einer Skriptsprache in eine ernst zu nehmende, allgemein einsetzbare Sprache verwandelten, die sich auch für die Softwareentwicklung in großem Maßstab eignet. Seit ES6 wird die ECMAScript-Spezifikation jährlich überarbeitet, und die Versionen der Sprache – ES2016, ES2017, ES2018, ES2019 und ES2020 usw. – werden jetzt nach dem Jahr der Veröffentlichung benannt.

Während der Entwicklung von JavaScript versuchten die Sprachdesigner, Fehler in den frühen Versionen (vor ES5) zu korrigieren. Um die Abwärtskompatibilität aufrechtzuerhalten, ist es nicht möglich, ältere Sprachmerkmale (sogenannte *Legacy-Features*) zu entfernen, egal wie mangelhaft sie sind. Aber in ES5 und späteren Versionen können Programme in den *Strict Mode* – den *strikten* oder *strict-Modus* – von JavaScript wechseln, in dem eine Reihe von frühen Sprachfehlern korrigiert wurde. Um diesen Modus zu aktivieren, benutzt man die »use strict«-Anweisung, die in [5.6.3](#) besprochen wird. Dieser Abschnitt fasst auch die Unterschiede zwischen Legacy-JavaScript und JavaScript im *strict-Modus* zusammen. In ES6 und später aktiviert die Verwendung neuer Sprachfunktionen oft implizit den *strict-Modus*. Wenn Sie beispielsweise das ES6-Schlüsselwort `class` verwenden oder ein ES6-Modul erstellen, ist der gesamte Code innerhalb der Klasse oder des Moduls automatisch »strict«, und die alten, mangelhaften Funktionen sind in diesen Kontexten nicht verfügbar. In diesem Buch behandle ich auch die Legacy-Features von JavaScript, weise aber darauf hin, dass diese im *strict-Modus* nicht verfügbar sind.

Die ursprüngliche Hostumgebung für JavaScript war ein Webbrowser, und das ist immer noch die am häufigsten verwendete Ausführungsumgebung für JavaScript-Code. In einer Webbrowserumgebung kann JavaScript-Code Input durch Maus und Tastatur des Benutzers sowie durch HTTP-Anfragen erhalten, dem Nutzer aber auch per HTML und CSS Ausgaben anzeigen.

Seit 2010 ist eine weitere Hostumgebung für JavaScript-Code verfügbar. Anstatt JavaScript auf die Arbeit mit den APIs zu beschränken, die von einem Webbrowser bereitgestellt werden, gibt Node JavaScript Zugriff auf das

gesamte Betriebssystem, sodass JavaScript-Programme Dateien lesen und schreiben, Daten über das Netzwerk senden und empfangen sowie HTTP-Anfragen stellen und bedienen können. Node ist eine beliebte Wahl zur Implementierung von Webservern und außerdem ein bequemes Werkzeug zum Schreiben einfacher Hilfsskripte als Alternative zu Shell-Skripten.

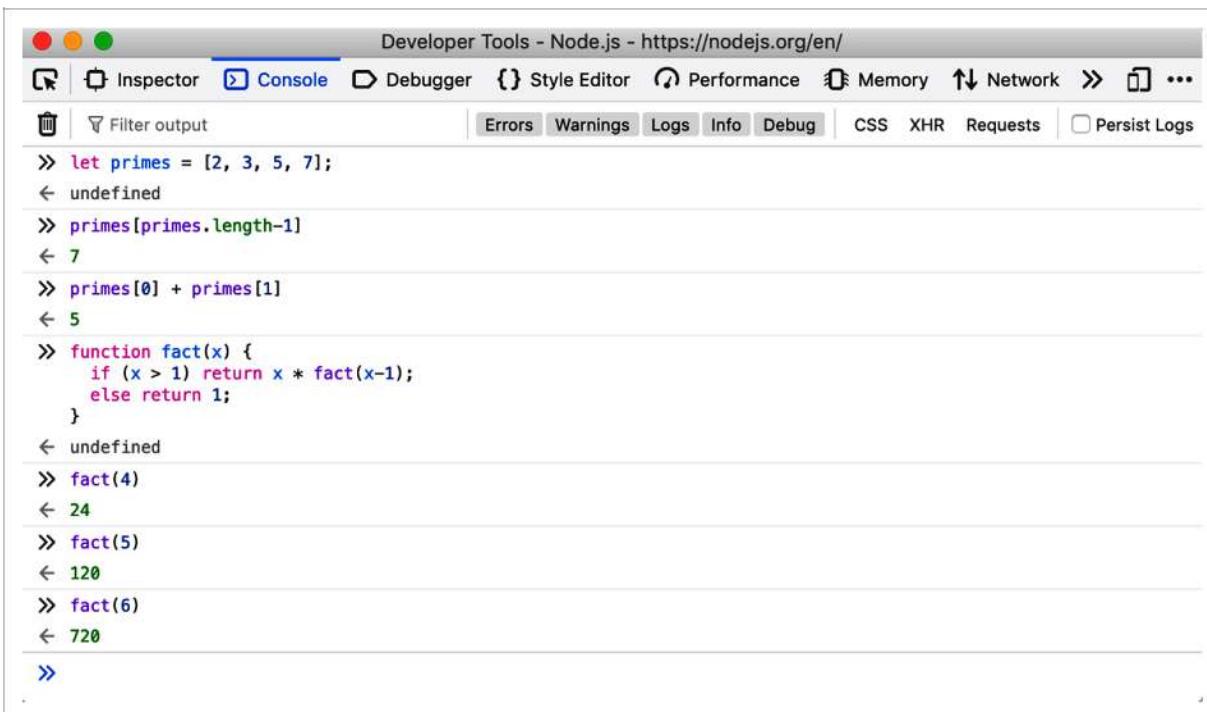
Der größte Teil dieses Buchs konzentriert sich auf die Programmiersprache JavaScript selbst, [Kapitel 11](#) dokumentiert die JavaScript-Standardbibliothek, [Kapitel 15](#) und [16](#) stellen Webbrower sowie Node als Hostumgebungen vor.

In diesem Buch werden zunächst die Grundlagen beschrieben, um dann darauf aufbauend zu fortgeschritteneren Abstraktionen auf höherer Ebene überzugehen. Die Kapitel sind dazu gedacht, mehr oder weniger der Reihe nach gelesen zu werden. Aber das Erlernen einer neuen Programmiersprache ist kein linearer Prozess und ebenso wenig das Beschreiben einer Sprache: Jedes Sprachfeature hängt mit anderen Merkmalen zusammen, und dieses Buch ist mit Querverweisen gespickt - manchmal auf zurückliegende Kapitel, manchmal auf Material, das Sie noch nicht gelesen haben. Dieses Einführungskapitel gibt Ihnen einen ersten Überblick über die Sprache und führt in die wichtigsten Merkmale ein, die das Verständnis einer detaillierteren Behandlung in den folgenden Kapiteln erleichtern werden. Falls Sie bereits JavaScript programmieren, können Sie dieses Kapitel wahrscheinlich überspringen. (Obwohl Sie sicher gern [Beispiel 1-1](#) am Ende des Kapitels lesen möchten, bevor Sie zu einem anderen Kapitel übergehen.)

1.1 JavaScript erkunden

Wenn man eine neue Programmiersprache lernt, ist es wichtig, die Beispiele auszuprobieren, sie dann selbst zu modifizieren und erneut auszuprobieren, um das erworbene Verständnis der Sprache zu testen. Dazu benötigen Sie einen JavaScript-Interpreter.

Am einfachsten lassen sich ein paar Zeilen JavaScript ausprobieren, indem man - mit F12 bzw. Strg+Umschalt+I in Windows- bzw. Befehlstaste+Optionstaste+I in MacUmgebungen - die Entwicklertools in einem Webbrower öffnet und auf die Registerkarte *Konsole* wechselt. Sie können dann in der Eingabeaufforderung Code eingeben und während der Eingabe die Ergebnisse sehen. Die Entwicklerwerkzeuge erscheinen oft als Fenster am unteren oder rechten Rand des Browserfensters, aber Sie können sie normalerweise als separate Fenster abkoppeln (wie in [Abbildung 1-1](#) gezeigt), was oft recht bequem ist.



The screenshot shows the Firefox Developer Tools interface with the 'Console' tab selected. The title bar reads 'Developer Tools - Node.js - https://nodejs.org/en/'. The console window displays the following JavaScript session:

```
» let primes = [2, 3, 5, 7];
← undefined
» primes[primes.length-1]
← 7
» primes[0] + primes[1]
← 5
» function fact(x) {
    if (x > 1) return x * fact(x-1);
    else return 1;
}
← undefined
» fact(4)
← 24
» fact(5)
← 120
» fact(6)
← 720
»
```

Abbildung 1-1: Die JavaScript-Konsole in den Entwicklertools von Firefox

Um JavaScript-Code auszuprobieren, können Sie auch Node von <https://nodejs.org> herunterladen und installieren. Sobald Node auf Ihrem System eingerichtet ist, öffnen Sie einfach ein Terminalfenster und geben **node** ein, um eine interaktive JavaScript-Sitzung wie diese zu starten:

```
$node
Welcome to Node.js v14.15.0.

Type ".help" for more information.

> .help
.break      Sometimes you get stuck, this gets you out
.clear      Alias for .break
.editor     Enter editor mode
.exit       Exit the repl
.help       Print this help message
.load       Load JS from a file into the REPL session
.save       Save all evaluated commands in this REPL session to a
file

Press ^C to abort current expression, ^D to exit the repl

> let x = 2, y = 3;
undefined
> x + y
5
> (x === 2) && (y === 3)
true
```

```
> (x > 3) || (y < 3)
```

```
false
```

1.2 Hello World

Wenn Sie bereit sind, mit längeren Codeblöcken zu experimentieren, eignen sich diese zeilenbasierten interaktiven Umgebungen normalerweise nicht mehr besonders, und Sie werden es wahrscheinlich vorziehen, Ihren Code in einem Texteditor zu schreiben. Von dort aus können Sie den Code in die JavaScript-Konsole oder in eine Node-Sitzung kopieren. Oder Sie speichern Ihren Code in einer Datei (die herkömmliche Dateinamenserweiterung für JavaScript-Code lautet `.js`) und führen dann diese Datei mit Node aus:

```
$ node snippet.js
```

Wenn Sie Node in einer solchen nicht interaktiven Weise verwenden, wird der Wert des ausgeführten Codes nicht automatisch ausgegeben, sodass Sie das selbst vornehmen müssen. Verwenden Sie die Funktion `console.log()`, um Text und andere JavaScript-Werte in Ihrem Terminalfenster oder in der Konsole der Entwicklerwerkzeuge eines Browsers anzuzeigen – zum Beispiel indem Sie eine Datei `hello.js` erstellen, die folgende Codezeile enthält:

```
console.log("Hello World!");
```

Wenn Sie diese Datei mit `node hello.js` ausführen, wird die Meldung »Hello World!« ausgegeben.

Möchten Sie die gleiche Nachricht in der JavaScript-Konsole eines Webbrowsers ausgeben, erstellen Sie eine neue Datei mit dem Namen *hello.html* und fügen diesen Text darin ein:

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

Dann laden Sie *hello.html* in Ihren Webbrowser mit einer *file://*-URL wie dieser:

```
file:///Users/username/javascript/hello.html
```

Öffnen Sie nun das Fenster der Entwicklertools, um die Begrüßung in der Konsole zu sehen.

1.3 Ein Rundgang durch JavaScript

Dieser Abschnitt enthält eine kurze Einführung in die JavaScript-Sprache anhand von Codebeispielen. Nach diesem einführenden Kapitel steigen wir richtig tief in JavaScript ein und beginnen damit ganz weit unten - [Kapitel 2](#) erklärt Dinge wie JavaScript-Kommentare, Semikola und den Unicode-Zeichensatz. In [Kapitel 3](#) wird es schon ein bisschen spannender: Es werden JavaScript-Variablen erklärt sowie die Werte, die man ihnen zuweisen kann.

Hier ist etwas Beispielcode, der die Höhepunkte dieser beiden Kapitel veranschaulicht:

```
// Alles nach einem doppelten Schrägstrich ist ein Kommentar in  
normaler Sprache.
```

```
// Lesen Sie Kommentare sorgfältig: Sie erläutern den  
JavaScript-Code.
```

```

// Eine Variable ist ein symbolischer Name für einen Wert.

// Variablen werden mit dem Schlüsselwort let deklariert:

let x;                                // Eine Variable namens x
deklarieren.

// Mit dem =-Zeichen können Variablen Werte zugewiesen werden.

x = 0;                                  // Jetzt hat die Variable x den Wert
0.

x                                // => 0: Eine Variable wird zu ihrem
Wert ausgewertet.

// JavaScript unterstützt eine ganze Reihe von Wertetypen.

x = 1;                                // Zahlen.

x = 0.01;                             // Zahlen können ganze oder reelle
Zahlen sein.

x = "hello world";                  // Textsequenzen in
Anführungszeichen.

x = 'JavaScript';                   // Auch einfache Anführungszeichen
sind möglich.

x = true;                            // Ein boolescher Wert.

x = false;                           // Der andere boolesche Wert.

x = null;                            // null ist ein spezieller Wert, der
für "kein Wert"

                                // steht.

x = undefined;                     // undefined ist ein weiterer
spezieller Wert wie null.

```

Zwei weitere sehr wichtige Datentypen, mit denen man in JavaScript-Programmen arbeiten kann, sind Objekte und

Arrays. Diese Typen behandeln wir in den [Kapiteln 6 und 7](#), aber sie sind so wichtig, dass sie Ihnen in diesem Buch bereits viele Male zuvor begegnen werden:

```
// Der wichtigste Datentyp in JavaScript ist das Objekt.  
  
// Ein Objekt ist eine Sammlung von Name/Wert-Paaren  
  
// (bzw. eine Map, in der Zeichenketten auf Werte abgebildet  
werden).  
  
let book = { // Objekte sind von geschweiften  
    Klammern umschlossen.  
    topic: "JavaScript", // Die Eigenschaft "topic" hat den  
    Wert "JavaScript".  
  
    edition: 7 // Die Eigenschaft "edition" hat den  
    Wert 7.  
}; // Die geschweifte Klammer markiert  
das Ende  
// des Objekts.  
  
// Zugriff auf die Eigenschaften eines Objekts mit . oder []:  
  
book.topic // => "JavaScript"  
  
book["edition"] // => 7: Eine weitere Möglichkeit,  
auf  
// Eigenschaftswerte zuzugreifen.  
  
book.author = "Flanagan"; // Neue Eigenschaften durch  
Zuweisung erstellen.  
  
book.contents = {} // {} ist ein leeres Objekt ohne  
Eigenschaften.  
  
// Bedingter Zugriff auf Eigenschaften mit ?. (ES2020):
```

```
book.contents?.ch01?.sect1 // => undefined: book.contents hat  
keine  
                                         // Eigenschaft ch01.  
  
// JavaScript unterstützt auch Arrays (numerisch indexierte  
Listen) mit Werten:  
  
let primes = [2, 3, 5, 7]; // Ein Array mit 4 Werten, begrenzt  
durch [ und ].  
  
primes[0]                  // => 2: Das erste Element (Index 0)  
des Arrays.  
  
primes.length               // => 4: Anzahl der Elemente im  
Array.  
  
primes[primes.length-1]     // => 7: Das letzte Element des  
Arrays.  
  
primes[4] = 9;              // Ein neues Element durch Zuweisen  
hinzufügen.  
  
primes[4] = 11;              // Oder ein vorhandenes Element  
durch Zuweisen  
                                         // verändern.  
  
let empty = [];              // [] ist ein leeres Array ohne  
Elemente.  
  
empty.length                // => 0  
  
// Arrays und Objekte können andere Arrays und Objekte  
enthalten:  
  
let points = [                // Ein Array mit 2 Elementen.  
    {x: 0, y: 0},             // Jedes Element ist ein Objekt.  
    {x: 1, y: 1}  
];
```