David Herman

# JavaScript effektiv

## 68 Dinge, die ein guter JavaScript-Entwickler wissen sollte

→ Mit einem Geleitwort von JavaScript-Erfinder Brendan Eich

dpunkt.verlag

Stimmen zur englischen Originalausgabe:

»*Effective JavaScript* ist unverzichtbare Lektüre für jeden, der begriffen hat, dass JavaScript nicht nur ein Spielzeug ist, und der die Möglichkeiten dieser Sprache voll ausschöpfen möchte. Dave Herman vermittelt den Lesern ein tiefes, fundiertes und praxisnahes Verständnis der Sprache und führt sie von einem Beispiel zum anderen, sodass Sie zu denselben Schlussfolgerungen kommen können wie er. Dies ist kein Buch für Leute, die einfache Kochrezepte haben wollen, sondern eine Sammlung der unter Mühen gewonnenen Erfahrungen in Form einer Führung. Es ist eines der wenigen Bücher über JavaScript, die ich ohne Vorbehalte empfehlen kann.«
– Alex Russell, Mitglied von TC39, Softwareingenieur, Google

»Bevor ich *Effective JavaScript* gelesen habe, dachte ich, es sei nur ein x-beliebiges Buch darüber, wie man besseren JavaScript-Code schreibt. Tatsächlich aber hat dieses Buch viel mehr zu bieten – es vermittelt Ihnen ein tiefes Verständnis der Sprache. Und das ist von entscheidender Bedeutung. Ohne dieses Verständnis kennen Sie die Sprache an sich gar nicht, sondern wissen nur, wie andere Programmierer Code schreiben.

Wenn Sie ein wirklich guter JavaScript-Entwickler werden wollen, dann lesen Sie dieses Buch! Ich für meinen Teil wünschte, ich hätte es schon zur Hand gehabt, als ich mit der JavaScript-Programmierung begann.«
– Anton Kovalyov, Entwickler von JSHint

»Es kommt nicht oft vor, dass ein Programmiersprachen-Experte auf so angenehme und verständliche Weise schreibt wie David. Seine Tour durch die Syntax und Semantik von JavaScript ist sowohl gut lesbar als auch äußerst erhellend. Warnungen vor JavaScripts Tücken ergänzen realistische Anwendungsfälle, und das alles wird in einem angenehmen Tempo vorgestellt. Nach Abschluss der Lektüre werden Sie das Gefühl haben, die Sprache gut und umfassend zu beherrschen.«
– Paul Irish, Entwicklerbeirat, Google Chrome

»Wenn Sie nach einem Buch suchen, dass Ihnen formale, aber trotzdem gut lesbare Einsichten in die Sprache JavaScript vermittelt, dann haben Sie es jetzt gefunden. Für JavaScript-Entwickler mit Vorkenntnissen ist es eine Fundgrube an Wissen, und selbst JavaScript-Veteranen können hier sicherlich noch das eine oder zehn andere Dinge lernen. Für erfahrene Anwender anderer Sprachen, die den Sprung ins kalte JavaScript-Wasser wagen, ist dieses Buch Pflichtlektüre, um sich schnell zurechtzufinden. Welchen Hintergrund Sie auch immer mitbringen, Dave Herman erledigt seine Aufgabe, JavaScript zu erklären – inklusive der schönen Seiten, der Warzen und allem dazwischen –, mit Bravour.«
– Rebecca Murphey, leitende JavaScript-Entwicklerin, Bocoup

»*Effective JavaScript* von Dave Herman erfüllt die Erwartungen an ein Programmierbuch aus der Buchreihe über effektive Softwareentwicklung und ist Pflichtlektüre für jeden, der sich ernsthaft mit JavaScript-Programmierung beschäftigt. Das Buch bietet ausführliche Erklärungen der inneren Mechanismen von JavaScript, durch die die Leser die Möglichkeiten der Sprache besser ausnutzen können.«
– Erik Arvidsson, leitender Softwareingenieur

»Nur selten hat man die Gelegenheit, von einem Meister seines Handwerks zu lernen. Dieses Buch macht genau das möglich! Es ist das JavaScript-Gegenstück zu einer Zeitmaschine, mit der Philosophen ins fünfte Jahrhundert vor Christus reisen können, um von Plato zu lernen.«
– Rick Waldron, JavaScript-Botschafter, Bocoup

**David Herman** ist Senior Researcher bei Mozilla Research und tätig im Ecma TC39, dem technischen Ausschuss, der sich um die Standardisierung von JavaScript kümmert. Er hat einen Bachelor in Informatik vom Grinnell College sowie einen Master- und einen Doktortitel in Informatik der Northeastern University.

David Herman

# JavaScript effektiv

## 68 Dinge, die ein guter JavaScript-Entwickler wissen sollte

Mit einem Geleitwort von Brendan Eich, dem Erfinder von JavaScript

dpunkt.verlag

Lektorat: René Schönfeldt
Übersetzung: G&U Language & Publishing Services GmbH, *www.gundu.com*
Satz: G&U Language & Publishing Services GmbH, *www.gundu.com*
Fachgutachter: Marcus Ross, Buchholz
Herstellung: Frank Heidt
Umschlaggestaltung: Helmut Kraus, *www.exclam.de*
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

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
Buch 978-3-86490-127-0
PDF 978-3-86491-424-9
ePub 978-3-86491-425-6

1. Auflage 2014

Ringstraße 19 B
69115 Heidelberg

Authorized translation from the English language edition, entitled Effective JavaScript: 68 specific ways to harness the power of JavaScript, 1st Edition, 0321812182 by Herman, David, published by Pearson Education, Inc, publishing as Addison-Wesley Professional, Copyright © 2013 dpunkt.verlag GmbH.



5 4 3 2 1 0

# Inhaltsverzeichnis

*Für Lisa, meine Liebe*

# Geleitwort

Wie inzwischen allgemein bekannt sein dürfte, habe ich JavaScript im Mai 1995 innerhalb von zehn Tagen entworfen – unter Stress und unter widersprüchlichen Anweisungen meiner Vorgesetzten: »Sorgen Sie dafür, dass es so aussieht wie Java!«, »Achten Sie darauf, dass es für Anfänger leicht erlernbar ist!«, »Stellen Sie sicher, dass es fast alles steuern kann, was es im Netscape-Browser gibt!«

Bei zwei wichtigen Aspekten habe ich mich von Anfang an um Richtigkeit bemüht (Funktionen erster Klasse und Objektprototypen), aber ansonsten bestand meine Reaktion auf die wechselnden Anforderungen und den unzumutbar engen Termin darin, JavaScript von Anfang an möglichst flexibel zu gestalten. Es war mir klar, dass Entwickler bei den ersten Versionen Patches anbringen mussten, um Fehler zu korrigieren, und bessere Vorgehensweisen auszuprobieren hatten als diejenigen, die ich in Form der mitgelieferten Bibliotheken zusammengeschustert hatte. Bei vielen Sprachen gibt es nur sehr eingeschränkte Möglichkeiten zur Veränderung. So ist es beispielsweise nicht möglich, integrierte Objekte zur Laufzeit zu bearbeiten oder zu erweitern oder Bindungen von Standardbibliotheksnamen durch Zuweisungen zu überschreiben. In JavaScript dagegen ist es möglich, fast jedes Objekt komplett zu ändern.

Ich glaube, dass ich angesichts der Umstände eine wohl ausgewogene Entscheidung für das Design getroffen habe. Gewiss führt dies in manchen Anwendungsbereichen zu Herausforderungen (beispielsweise bei der gefahrlosen Vermischung von vertrauenswürdigem und nicht vertrauenswürdigem Code innerhalb der Sicherheitsgrenzen eines Browsers). Allerdings war es unverzichtbar, ein sogenanntes Monkey-Patching zuzulassen, damit Entwickler Standardobjekte bearbeiten können, um sich um Fehler herumzulavieren oder um in älteren Browsern Emulationen von moderneren Möglichkeiten bereitzustellen (wie es bei der sogenannten Polyfill-Bibliothek der Fall ist, die man in Deutschland wohl eher »Moltofill-Bibliothek« genannt hätte).

Neben diesen teilweise profanen Verwendungszwecken hat die Formbarkeit von JavaScript auch verschiedene kreative Weiterentwicklungen durch die Benutzergemeinde ermöglicht. Führende Benutzer haben Toolkit- und Framework-Bibliotheken nach dem Muster anderer Sprachen erstellt – Prototype nach dem Vorbild von Ruby, MochiKit nach Python, Dojo nach Java und TIBET nach Smalltalk. Schließlich kam die Bibliothek jQuery (»New Wave JavaScript«), die für mich bei meiner ersten Begegnung im Jahr 2007 wie ein Nachzügler wirkte, und eroberte die JavaScript-Welt im Sturm, indem sie es vermied, sich an älteren JavaScript-Bibliotheken und damit an anderen Sprachen zu orientieren. Stattdessen bohrte sie das »Abfragen-und-umsetzen-Modell« des Browsers auf und vereinfachte es radikal.

Führende Benutzer und ihre Innovationsnetze haben damit ein »Eigenbau-JavaScript« geschaffen, das nach wie vor in anderen Bibliotheken nachgebildet und vereinfacht wird und auch den Bemühungen zur Standardisierung im Web unterworfen wird.

Im Rahmen dieser Entwicklung ist JavaScript abwärtskompatibel geblieben (wobei manche statt von »backward« von »bugward« sprechen, also »mit den alten Fehlern kompatibel«) und natürlich immer noch von Haus aus veränderbar. Das gilt auch noch nach der Ergänzung um bestimmte Methoden in der letzten Version des ECMAScript-Standards, die Objekte gegen Erweiterung und Objekteigenschaften gegen das Überschreiben schützen. Die Entwicklung von JavaScript ist aber noch lange nicht abgeschlossen. Wie bei jeder lebenden Sprache und jedem biologischen System ist Veränderung langfristig die einzige Konstante. Ich kann mir nicht vorstellen, dass es jemals eine einzige »Standardbibliothek« oder einen Programmierstil geben wird, der alle anderen ungültig macht.

Keine Sprache ist frei von Eigenheiten oder so streng, dass man universell gültige empfehlenswerte Vorgehensweisen dafür aufstellen könnte, und JavaScript ist eher das Gegenteil. Mehr als bei den meisten anderen Programmiersprachen müssen JavaScript-Entwickler daher einen guten Stil, eine korrekte Anwendung und empfehlenswerte Vorgehensweisen studieren und anwenden, um wirkungsvollen Code zu schreiben. Allerdings glaube ich, dass es ganz wichtig ist, nicht überzureagieren und strenge oder gar dogmatische Richtlinien aufzustellen.

Dieses Buch verfolgt einen ausgeglichenen Ansatz der auf belegbaren Problemen und konkreten Erfahrungen beruht, ohne strenge und übergenaue Vorschriften zu machen. Für viele Menschen, die versuchen, effektiven JavaScript-Code zu schreiben, ohne Ausdrucksstärke und die Offenheit für neue Ideen zu opfern, wird dieses Buch eine entscheidende Hilfestellung und ein treuer Begleiter sein. Es ist sehr zielgerichtet geschrieben, leicht zu lesen und bietet hervorragende Beispiele.

Ich habe das große Vorrecht, David Herman seit 2006 persönlich zu kennen, als ich im Auftrag von Mozilla zum ersten Mal Kontakt mit ihm aufnahm, um ihn als gern gesehenen Experten in das Standardisierungskomitee der Ecma aufzunehmen. Davids tiefe und allürenfreie Fachkompetenz und seine Begeisterung für JavaScript machen sich auf jeder Seite bemerkbar. Bravo!

*Brendan Eich*

# Vorwort

Um eine Programmiersprache zu lernen, müssen Sie sich mit der *Syntax* vertraut machen, d.h. mit den Formen und Strukturen, aus denen sich gültige Programme zusammensetzen, und mit der *Semantik*, also der Bedeutung oder dem Verhalten dieser Formen. Wollen Sie eine Sprache aber richtig beherrschen, müssen Sie auch ihre *Pragmatik* verstehen, also die Art und Weise, in der Sie die Merkmale der Sprache einsetzen, um wirkungsvolle Programme zu schreiben. Letzteres kann besonders knifflig sein, vor allem in einer so flexiblen und ausdrucksstarken Sprache wie JavaScript.

In diesem Buch geht es um die Pragmatik von JavaScript. Es bietet keine Einführung in die Sprache, und ich setze voraus, dass Sie bereits eine gewisse Vertrautheit mit der Programmierung im Allgemeinen und mit JavaScript im Besonderen mitbringen. Es gibt viele hervorragende Einführungen in JavaScript, beispielsweise *Das Beste an JavaScript* von Douglas Crockford und *Die Kunst der JavaScript-Programmierung* von Marijn Haverbeke. Mit diesem Buch dagegen möchte ich Ihr Verständnis von JavaScript vertiefen, sodass Sie die Sprache wirkungsvoll einsetzen können, um vorhersagbarere, zuverlässigere und wartungsfreundlichere JavaScript-Anwendungen und -Bibliotheken zu schreiben.

## JavaScript und ECMAScript

Bevor wir uns dem eigentlichen Stoff widmen, ist es sinnvoll, die Terminologie zu klären. In diesem Buch geht es um eine Sprache, die fast überall als JavaScript bekannt ist. In dem offiziellen Standard, der die Spezifikation festlegt, geht es jedoch um eine Sprache, die EMCAScript genannt wird. Die Geschichte ist ziemlich verzwickt, aber im Grunde genommen geht es um eine markenrechtliche Frage. Aus Rechtsgründen durfte die Standardisierungsorganisation – Ecma International – den Namen JavaScript für den Standard nicht verwenden. (Um die

Sache noch verwirrender zu machen, hat diese Organisation ihren ursprünglichen Namen ECMA – was für European Computer Manufacturers Association stand – in Ecma International [ohne Großschreibung!] geändert. Zu diesem Zeitpunkt war die Versalschreibweise ECMA aber schon wie in Stein gemeißelt.)

Wenn formal von ECMAScript gesprochen wird, ist damit gewöhnlich die »ideale«, durch den Ecma-Standard festgelegte Sprache gemeint. JavaScript kann heutzutage alles Mögliche bedeuten – die Sprache, wie sie heute üblicherweise verwendet wird, aber auch die konkrete JavaScript-Engine eines bestimmten Herstellers. Gewöhnlich wird der Begriff austauschbar für beides verwendet. Der Klarheit und Einheitlichkeit halber verwende ich in diesem Buch den Begriff *ECMAScript* nur, wenn es um den offiziellen Standard geht, und sonst nutze ich die Bezeichnung *JavaScript*. Außerdem verwende ich die gängige Abkürzung *ES5* für die fünfte Edition des ECMAScript-Standards.

## JavaScript im Web

Wer über JavaScript spricht, muss fast zwangsläufig auch das Web erwähnen. Zurzeit ist JavaScript die einzige Programmiersprache mit integrierter Unterstützung für clientseitige Skripte in allen wichtigen Webbrowsern. Darüber hinaus ist JavaScript mit dem Aufkommen der Plattform *Node.js* in den letzten Jahren zu einer weit verbreiteten Sprache für serverseitige Anwendungen geworden.

Trotzdem ist dies ein Buch über JavaScript und nicht über Webprogrammierung. Manchmal ist es hilfreich, Beispiele und Anwendungszwecke aus dem Bereich des Webs anzugeben, aber der Schwerpunkt dieses Buches liegt auf der Sprache an sich – auf ihrer Syntax, Semantik und Pragmatik – und nicht auf den APIs und Technologien der Webplattform.

## Ein Hinweis zur Nebenläufigkeit

Ein eigenartiger Aspekt von JavaScript ist die Tatsache, dass es keine Spezifikation für das Verhalten bei Nebenläufigkeit gibt. Bis einschließlich zur fünften Edition sagt der ECMAScript-Standard nichts über das Verhalten von JavaScript-Programmen in interaktiven und nebenläufigen Umgebungen. In Kapitel 7 geht es um die Nebenläufigkeit, wobei also im Grunde genommen inoffizielle Merkmale von JavaScript beschrieben werden. In der Praxis nutzen jedoch alle wichtigen JavaScript-Engines das gleiche Modell für die Nebenläufigkeit.

Die Arbeit mit nebenläufigen und interaktiven Programmen ist außerdem ein zentrales und gemeinsames Prinzip der JavaScript-Programmierung, obwohl sie im Standard nicht erwähnt wird. Möglicherweise werden diese allgemein üblichen Aspekte des Nebenläufigkeitsmodells von JavaScript in zukünftigen Ausgaben des ECMAScript-Standards formalisiert.

# Danksagung

Für dieses Buch bin ich Brendan Eich, dem Erfinder von JavaScript, sehr verpflichtet. Ich bin ihm sehr dankbar für seine Einladung, an der Standardisierung von JavaScript mitzuarbeiten, und für seine Betreuung und Unterstützung meiner Karriere bei Mozilla.

Viel von dem Stoff in diesem Buch wurde durch hervorragende Blogeinträge und Onlineartikel angeregt und beeinflusst. Viel gelernt habe ich von den Postings von Ben »Cowboy« Alman, Erik Arvidsson, Mathias Bynens, Tim »Creationix« Caswell, Michaeljohn »Inimino« Clement, Angus Croll, Andrew Dupont, Ariya Hidayat, Steven Levithan, Pan Thomakos, Jeff Walden und Juriy »Kangax« Zaytsev. Die wichtigste Quelle für dieses Buch war natürlich die ECMAScript-Spezifikation, die seit Edition 5 unermüdlich von Allen Wirfs-Brock bearbeitet und aktualisiert wird. Auch das Mozilla Developer Network ist nach wie vor eine der eindrucksvollsten und qualitativ hochwertigsten Onlinequellen für JavaScript-APIs und -Merkmale.

Während ich dieses Buch plante und schrieb, standen mir viele Berater zur Seite. John Resig gab mir nützliche Ratschläge zum Schreiben, bevor ich begann. Blake Kaplan und Patrick Walton halfen mir, meine Gedanken zu sammeln und den Aufbau des Buches in den frühren Stadien zu planen. Während der Abfassung bekam ich großartigen Rat von Brian Anderson, Norbert Lindenberg, Sam Tobin-Hochstadt, Rick Waldron und Patrick Walton.

Es war eine große Freude, mit der Belegschaft von Pearson zusammenzuarbeiten. Olivia Basegio, Audrey Doyle, Trina MacDonald, Scott Meyers und Chris Zahn waren meinen Fragen gegenüber immer sehr aufgeschlossen, reagierten geduldig auf meine Verzögerungen und passten sich meinen Erfordernissen an. Ich könnte mir keine besseren Umstände für eine Premiere als Autor vorstellen. Außerdem fühle ich mich äußerst geehrt, zu dieser hervorragenden Buchreihe beitragen zu

fügen. Ich war schon lange ein Fan von *Effective* C++, bevor ich auch nur zu träumen wagte, dass ich selbst eines Tages das große Privileg genießen dürfte, ein *Effective*-Buch zu schreiben.

Als ich mein traumhaftes Team an Fachgutachtern kennenlernte, konnte ich mein Glück kaum fassen. Ich fühlte mich geehrt, dass sich Erik Arvidsson, Rebecca Murphey, Rick Waldron und Richard Worth bereit erklärt hatten, dieses Buch durchzusehen. Von ihnen habe ich unschätzbare Kritik und Vorschläge erhalten. In mehr als nur einem Fall haben sie mich vor wahrhaft peinlichen Fehlern bewahrt.

Ein Buch zu schreiben war eine erschreckendere Erfahrung, als ich es mir vorgestellt hatte. Ohne die Unterstützung meiner Freunde und Kollegen hätte ich wahrscheinlich die Nerven verloren. Ich bin mir nicht sicher, ob es ihnen zu jenem Zeitpunkt schon klar war, aber Andy Denmark, Rick Waldron und Travis Winfrey gaben mir die Ermutigung, die ich brauchte, wenn mir Zweifel kamen.

Den Großteil dieses Buches habe ich in dem fabelhaften Java Beach Café im wunderschönen Parkside-Viertel von San Francisco geschrieben. Die Mitarbeiter dort kennen mich alle mit Namen und wissen schon, was ich bestellen werde, bevor ich es tue. Ich bin ihnen sehr dankbar dafür, dass sie mir ein so gemütliches Plätzchen zum Arbeiten bereitgestellt und mich immer mit genügend Nahrung und Koffein versorgt haben.

Mein kleiner pelziger Katzenfreund Schmoopy hat sich auch nach Kräften bemüht, zu diesem Buch beizutragen. Zumindest hopste er ständig auf meinen Schoß und saß vor dem Bildschirm. (Es *könnte* auch etwas mit der Wärme des Laptops zu tun haben.) Schmoopy ist schon seit 2006 mein treuer Kumpel, und ich kann mir ein Leben ohne das kleine Pelzknäuel nicht mehr vorstellen.

Meine ganze Familie war von Anfang bis Ende begeistert von dem Projekt und hat mich dabei unterstützt. Leider sind meine Großeltern Frank und Miriam Slamar von uns gegangen, bevor ich ihnen das Endprodukt vorstellen konnte. Aber sie freuten sich und waren stolz auf mich, und in dieses Buch sind auch einige Kindheitserfahrungen aus der Zeit eingeflossen, in der ich mit Frank BASIC-Programme schrieb.

Schließlich verdanke ich Lisa Silveria, der Liebe meines Lebens, mehr, als ich jemals in irgendeiner Einleitung zu einem Buch zum Ausdruck bringen könnte.

# 1 Darf ich vorstellen: JavaScript

Bei der Gestaltung von JavaScript wurde auf ein vertrautes Erscheinungsbild Wert gelegt. Die Syntax erinnert an Java, die verwendeten Konstrukte (wie Funktionen, Arrays, Dictionaries und reguläre Ausdrücke) kommen in vielen Skriptsprachen vor – und so scheint JavaScript für jeden, der über etwas Programmiererfahrung verfügt, leicht erlernbar. Wegen der geringen Anzahl an Grundprinzipien, die der Sprache zugrunde liegen, können zudem selbst Anfänger schon mit wenig Lernaufwand erste Programme schreiben.

JavaScript ist in der Tat leicht zugänglich. Doch wer die Sprache richtig beherrschen will, muss sich etwas Zeit nehmen, um die Semantik genau zu verstehen und die Eigenheiten und die wirkungsvollsten Idiome der Sprache gut kennenzulernen. Jedes Kapitel dieses Buches deckt daher einen Themenbereich der effektiven Verwendung von JavaScript ab. Im ersten Kapitel beginnen wir mit den Grundlagen.

## Welche Version von JavaScript verwenden Sie? — Thema 1

*ECMAScript*

Wie viele erfolgreiche Technologien hat sich auch JavaScript mit der Zeit weiterentwickelt. Ursprünglich wurde die Sprache als Ergänzung zu Java für die Programmierung interaktiver Webseiten herausgebracht, doch schließlich löste sie Java als vorherrschende Programmiersprache für das Web sogar ab. Die Beliebtheit von JavaScript führte dazu, dass die Sprache 1997 unter dem Namen ECMAScript international standardisiert wurde. Heute gibt es viele konkurrierende Implementierungen von JavaScript, die jeweils mit unterschiedlichen Versionen des ECMAScript-Standards konform sind.

*ES3*

Die dritte Edition des ECMAScript-Standards (gewöhnlich ES3 genannt) wurde 1999 abgeschlossen und ist nach wie vor die am häufigsten eingesetzte Version von JavaScript.

*ES5*

Die nächste größere Verbesserung bildete die Edition ES5, die 2009 veröffentlicht wurde. In ihr wurden einige neue Eigenschaften eingeführt und es wurden diverse zuvor noch nicht spezifizierte, aber bereits durch viele Browser unterstützte Merkmale standardisiert. ES5 wird aber noch nicht überall unterstützt. Deshalb werde ich in diesem Buch eigens darauf hinweisen, wenn sich ein Thema oder ein bestimmter Ratschlag auf diese Version bezieht.

*Nicht standardisierte Merkmale*

Neben den verschiedenen Editionen des Standards gibt es noch eine Reihe von nicht standardisierten Merkmalen, die zwar von einigen JavaScript-Implementierungen unterstützt werden, aber nicht von allen. Beispielsweise lassen viele JavaScript-Engines das Schlüsselwort `const` zur Definition von Variablen zu, obwohl der ECMAScript-Standard die Syntax und das Verhalten dieses Schlüsselworts nicht definiert. Außerdem weist `const` in den einzelnen Implementierungen jeweils ein unterschiedliches Verhalten auf. In einigen Fällen ist es z.B. nicht möglich, `const`-Variablen zu aktualisieren:

```
const PI = 3.141592653589793;
PI = "modified!";
PI; // 3.141592653589793
```

Andere Implementierungen dagegen behandeln `const` einfach als Synonym von `var`:

```
const PI = 3.141592653589793;
PI = "modified!";
PI; // "modified!"
```

Aufgrund der langen Geschichte von JavaScript und der Unterschiede zwischen den Implementierungen ist es nicht ganz einfach, den Überblick darüber zu behalten, welche Merkmale auf welcher Plattform zur Verfügung stehen. Verschärft wird dieses Problem noch dadurch, dass die wichtigste Umgebung, in der JavaScript eingesetzt wird, – nämlich der Webbrowser – den Programmierern keine Kontrolle darüber gibt, welche Version von JavaScript ihren Code ausführt. Da die Endbenutzer unterschiedliche Versionen der verschiedensten Browser einsetzen, müssen Webprogramme mit Bedacht geschrieben werden, damit sie in allen Browsern einheitlich funktionieren.

Auf der anderen Seite ist JavaScript nicht ausschließlich für die clientseitige Webprogrammierung da. Andere Verwendungszwecke sind beispielsweise serverseitige Programme, Browsererweiterungen und Skripte für Mobil- und Desktopanwendungen. In diesen Fällen wissen Sie genauer, welche Version von JavaScript zur Verfügung steht, und es ist daher sinnvoll, die zusätzlichen Merkmale zu nutzen, die in der Implementierung für die betreffende Plattform zur Verfügung stehen.