Cay Horstmann

JavaScript für Ungeduldige

Der schnelle Einstieg in modernes JavaScript

dpunkt.verlag

Cay Horstmann ist Hauptautor von *Core Java™*, Band I und II, 11. Auflage (Pearson, 2018), *Scala for the Impatient*, 2. Auflage (Addison-Wesley, 2016) und *Core Java SE 9 for the Impatient* (Addison-Wesley, 2017). Er ist emeritierter Professor für Informatik an der San José State University (Kalifornien, USA), Java-Champion und häufiger Redner auf Konferenzen der Computerbranche.

Cay Horstmann

JavaScript für Ungeduldige

Der schnelle Einstieg in modernes JavaScript

Cay Horstmann

Lektorat: Melanie Andrisek
Übersetzung: Volkmar Gronau
Copy-Editing: Alexander Reischert, *www.aluan.de*
Satz: G&U Language & Publishing Services GmbH, Flensburg, *www.GundU.com*
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, *www.exclam.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 978-3-86490-801-9
PDF 978-3-96910-093-6
ePub 978-3-96910-094-3
mobi 978-3-96910-095-0

1. Auflage 2021

Wieblinger Weg 17
69123 Heidelberg

Authorized translation from the English language edition, entitled MODERN JAVASCRIPT FOR THE IMPATIENT, 1st Edition by CAY HORSTMANN, published by Pearson Education, Inc, publishing as Addison-Wesley Professional, Copyright © 2020 Pearson Education, Inc

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: hallo@dpunkt.de.

5 4 3 2 1 0

Inhalt

Stichwortverzeichnis

Für Chi, die geduldigste Person in meinem Leben

Vorwort

Erfahrene Programmierer, die mit Sprachen wie Java, C# oder C++ vertraut sind, finden sich oft in Situationen wieder, in denen sie mit JavaScript arbeiten müssen. Das liegt daran, dass es immer mehr webgestützte Benutzerschnittstellen gibt und JavaScript nun einmal die *Lingua franca* der Browser ist. Das Electron-Framework hat die Anwendung dieser Sprache auf Rich-Client-Anwendungen ausgedehnt, und es gibt verschiedene Möglichkeiten, um JavaScript-Apps für Mobilgeräte zu erstellen. Auch serverseitig wird JavaScript immer häufiger eingesetzt.

Vor vielen Jahren galt JavaScript als eine Sprache zur Programmierung im Kleinen. Ihre Features konnten für umfangreiche Programme ziemlich verwirrend und fehleranfällig sein. Mit den heutigen Standardisierungsbemühungen und dem Angebot an Werkzeugen hat die Sprache sich jedoch weit über diese bescheidenen Anfänge hinaus entwickelt.

Leider ist es schwierig, modernes JavaScript zu lernen, ohne mit veraltetem JavaScript überschüttet zu werden. In den meisten Büchern, Kursen und Blogposts geht es um den Übergang von älteren JavaScript-Versionen, was für Personen nicht hilfreich ist, die von anderen Sprachen kommen.

Das ist die Lücke, die ich mit diesem Buch füllen möchte. Ich gehe davon aus, dass Sie bereits ein

kompetenter Programmierer sind und sich mit Verzweigungen und Schleifen, Funktionen, Datenstrukturen und den Grundlagen der objektorientierten Programmierung auskennen. Ich erkläre Ihnen, wie Sie in modernem JavaScript produktiv arbeiten können, und erwähne veraltete Merkmale nur am Rande. Sie lernen hier, wie Sie modernes JavaScript nutzen, ohne über die Fallstricke der Vergangenheit zu stolpern.

JavaScript ist nicht perfekt, aber es hat sich für die Programmierung von Benutzerschnittstellen und für viele serverseitige Aufgaben als gut geeignet erwiesen. Wie Jeff Atwood es einmal vorausschauend formulierte: »Jede Anwendung, die in JavaScript geschrieben werden *kann, wird* irgendwann auch in JavaScript geschrieben.«

Arbeiten Sie dieses Buch durch, um zu lernen, wie Sie die nächste Version Ihrer Anwendung in modernem JavaScript schreiben können.

Fünf goldene Regeln

Wenn Sie auf einige wenige klassische Features von JavaScript verzichten, können Sie das Maß an geistiger Anstrengung erheblich verringern, um die Sprache zu erlernen und anzuwenden. Die folgenden Regeln werden Ihnen jetzt zwar noch nicht viel sagen, aber ich führe Sie hier trotzdem zum späteren Nachschlagen auf – auch um Ihnen zu zeigen, wie beruhigend wenige es sind:

1. Deklarieren Sie Variablen mit `let` oder `const`, nicht mit `var`.
2. Verwenden Sie den strikten Modus.
3. Seien Sie sich immer über die Typen im Klaren und vermeiden Sie die automatische Typkonvertierung.
4. Machen Sie sich damit vertraut, wie Prototypen funktionieren, aber verwenden Sie für Klassen,

Konstruktoren und Methoden die moderne Syntax.

5. Verwenden Sie `this` nicht außerhalb von Konstruktoren und Methoden.

Darüber hinaus gibt es noch eine Metaregel: *Schauen Sie sich keinen Wat-Code an*, also diese verwirrenden Ausschnitte aus JavaScript-Code, die mit einem sarkastischen (und orthografisch nicht korrekten) »Wat?!« kommentiert sind. Manche Leute haben Spaß daran zu zeigen, wie furchtbar JavaScript angeblich ist, indem sie obskuren Code analysieren. Aus solchen Übungen habe ich jedoch nie etwas Sinnvolles mitgenommen. Welchen Vorteil bietet es Ihnen etwa zu wissen, dass `2 * ['21']` den Wert `42` ergibt, `2 + ['40']` aber nicht, wenn Ihnen die goldene Regel Nr. 3 klipp und klar sagt, dass Sie nicht mit Typkonvertierungen herumdoktern sollten? Wenn ich in eine solche verwirrende Situation gerate, frage ich mich normalerweise, wie ich sie vermeiden kann, anstatt sie in allen unnützen Einzelheiten zu erklären.

Lernstoff von unterschiedlichem Niveau

Den Lernstoff in diesem Buch habe ich so angeordnet, dass Sie die benötigten Informationen leicht wiederfinden können, wenn Sie sie brauchen. Das ist allerdings nicht unbedingt die richtige Anordnung, wenn Sie das Buch zum ersten Mal lesen. Um Ihnen das Lernen zu erleichtern, habe ich jedes Kapitel mit einem Symbol für das Niveau des behandelten Stoffs versehen. Einzelne Abschnitte, die auf einem höheren Niveau angesiedelt sind, bekommen dabei ihre eigenen Symbole. Lassen Sie diese Abschnitte bei der ersten Lektüre aus und lesen Sie sie erst dann, wenn Sie dazu bereit sind.

Zur Kennzeichnung habe ich die folgenden Symbole verwendet:

Der ungeduldige Hase steht für ein **grundlegendes Thema**, das selbst die ungeduldigsten Leser nicht überspringen sollten.

Alice kennzeichnet ein Thema von **mittlerem Niveau**, mit dem sich die meisten Programmierer vertraut machen sollten, allerdings nicht unbedingt beim ersten Lesen.

Die Grinsekatze weist auf ein **fortgeschrittenes Thema** hin, das Framework-Entwicklern ein Lächeln entlocken mag. Die meisten Anwendungsentwickler können diese Abschnitte jedoch getrost ignorieren.

Der verrückte Hutmacher schließlich kennzeichnet **komplizierte Themen**, die einen in den Wahnsinn treiben können und sich nur für Leser mit krankhafter Neugier eignen.

Der Aufbau dieses Buches

In **Kapitel 1** legen wir mit den Grundlagen von JavaScript los: mit Werten und ihren Typen, mit Variablen und vor allem mit Objektliteralen. **Kapitel 2** behandelt den Steuerungsfluss. Wenn Sie mit Java, C# oder C++ vertraut sind, können Sie dieses Kapitel wahrscheinlich einfach überfliegen. In **Kapitel 3** lernen Sie Funktionen und die funktionale Programmierung kennen, die in JavaScript eine große Rolle spielt. JavaScript hat zwar ein Objektmodell, allerdings unterscheidet es sich stark von dem klassengestützter Programmiersprachen. **Kapitel 4** beschreibt dieses Objektmodell ausführlich, wobei der Schwerpunkt auf der modernen Syntax liegt. Die **Kapitel 5 und 6** behandeln die Bibliotheksklassen, die Sie am häufigsten zur Arbeit mit Zahlen, Datumsangaben, Strings

und regulären Ausdrücken verwenden werden. Diese ersten sechs Kapitel sind auf grundlegendem Niveau, wobei einige etwas anspruchsvollere Abschnitte eingestreut wurden.

Die folgenden vier Kapitel behandeln Themen von mittlerem Niveau. In **Kapitel 7** erfahren Sie, wie Sie mit Arrays und anderen Sammlungstypen aus der JavaScript-Standardbibliothek arbeiten. Wenn Ihr Programm von Benutzern in aller Welt verwendet wird, sollten Sie der Internationalisierung besondere Aufmerksamkeit schenken, um die es in **Kapitel 8** geht. **Kapitel 9** über asynchrone Programmierung ist für alle Programmierer äußerst wichtig. Die asynchrone Programmierung war in JavaScript ziemlich kompliziert, ist mit der Einführung von Promises und der Schlüsselwörter `async` und `await` aber viel einfacher geworden. Außerdem hat JavaScript jetzt ein standardmäßiges Modulsystem, das Thema von **Kapitel 10** ist. Darin erfahren Sie, wie Sie Module von anderen Programmierern nutzen und ihre eigenen schreiben können.

In **Kapitel 11** geht es um Metaprogrammierung auf fortgeschrittenem Niveau. Sie sollten es lesen, wenn Sie Werkzeuge erstellen, um beliebige JavaScript-Objekte zu analysieren und umzuwandeln. **Kapitel 12** schließt die Erörterung von JavaScript mit einem weiteren fortgeschrittenen Thema ab, nämlich Iteratoren und Generatoren: zwei äußerst nützliche Mechanismen, um beliebige Folgen von Werten zu durchlaufen und zu produzieren.

Schließlich gibt es noch ein Bonuskapitel, nämlich **Kapitel 13** über TypeScript. Dabei handelt es sich um eine Erweiterung von JavaScript, die eine Typüberprüfung zur Kompilierzeit bietet. Es gehört nicht zum JavaScript-Standard, ist aber sehr populär. Lesen Sie dieses Kapitel,

um selbst zu entscheiden, ob Sie beim einfachen JavaScript bleiben oder die Typisierung zur Kompilierzeit nutzen wollen.

Dieses Buch soll Ihnen solide Grundkenntnisse der *Sprache* JavaScript verleihen, sodass Sie sie souverän nutzen können. Informationen über die Werkzeuge und Frameworks, die einem ständigen Wandel unterliegen, müssen Sie dagegen an einem anderen Ort suchen.

Warum ich dieses Buch geschrieben habe

JavaScript ist eine der am häufigsten verwendeten Programmiersprachen der Welt. Wie viele Programmierer kannte ich zunächst ein bisschen Pidgin-JavaScript. Eines Tages aber musste ich ziemlich überstürzt echtes JavaScript lernen. Doch wie?

Es gab zwar eine Menge Bücher, mit denen Programmierer, die sich nur gelegentlich mit Webentwicklung befassen, ein bisschen JavaScript lernen konnten, aber so viel verstand ich von der Sprache ohnehin. Das *Nashornbuch* von Flanagan[1] war 1996 zwar großartig, setzt die Leser von heute aber zu vielen Missgriffen der Vergangenheit aus. *Das Beste an JavaScript* von Douglas Crockford[2] hat die JavaScript-Welt 2008 wachgerüttelt, aber ein Großteil seiner Botschaft ist bereits in nachfolgende Änderungen der Sprache eingeflossen. Außerdem gibt es viele Bücher, die JavaScript-Programmierern der alten Schule die Welt der modernen Standards nahebringen, aber sie beschäftigen sich mit zu vielen klassischen JavaScript-Elementen, die mir nicht behagen.

Das Web ist voll von Blogs zum Thema JavaScript, allerdings von sehr unterschiedlicher Qualität. Einige sind korrekt, aber viele zeigen nur ein ziemlich schwaches

Verständnis. Für mich war es nicht sehr sinnvoll, das Web nach Blogs zu durchsuchen und jeweils zu prüfen, wie wahrheitsgetreu sie sind.

Merkwürdigerweise konnte ich kein Buch für die Millionen Programmierer finden, die Java oder eine ähnliche Sprache kennen und JavaScript in seiner heutigen Form ohne den historischen Ballast lernen wollen.

Also musste ich es selbst schreiben.

Danksagung

Ein weiteres Mal möchte ich meinem Herausgeber Greg Doench für die Unterstützung dieses Projekts danken sowie Dmitry Kirsanov und Alina Kirsanova für das Korrekturlesen und den Satz des Buches. Mein besonderer Dank geht an meine Lektoren Gail Anderson, Tom Austin, Scott Davis, Scott Good, Kito Mann, Bob Nicholson, Ron Mak und Henri Tremblay, die sorgfältig Fehler aufgespürt und wohlüberlegte Vorschläge für Verbesserungen gemacht haben.

Cay Horstmann
Berlin
März 2020

1
Werte und Variable

In diesem Kapitel lernen Sie die Datentypen kennen, mit denen Sie in JavaScript-Anwendungen arbeiten können: Zahlen, Strings und andere primitive Typen sowie Objekte und Arrays. Sie erfahren hier, wie Sie solche Werte in Variablen speichern, wie Sie Werte von einem Typ in einen anderen umwandeln und wie Sie sie mithilfe von Operatoren kombinieren.

Selbst begeisterte JavaScript-Programmierer geben zu, dass einige Konstrukte von JavaScript – die eigentlich dabei helfen sollen, Programme möglichst kurz und knapp zu schreiben – zu widersinnigen Ergebnissen führen können und daher am besten vermieden werden sollten. In diesem und den folgenden Kapiteln werde ich solche Probleme aufzeigen und einige einfache Regeln für sicheres Programmieren vorstellen.

1.1 JavaScript ausführen

Es gibt verschiedene Möglichkeiten, um während der Lektüre dieses Buches JavaScript-Programme auszuführen. Da JavaScript ursprünglich zur Ausführung in einem Browser gedacht war, können Sie JavaScript-Code in eine HTML-Datei einbetten. Um Werte anzuzeigen, rufen Sie darin die Methode `window.alert` auf. Eine solche Datei sieht wie folgt aus:

```
<html>

   <head>

      <title>My First JavaScript Program</title>

      <script type="text/javascript">

         let a = 6

         let b = 7

         window.alert(a * b)

      </script>

   </head>

   <body>

   </body>

</html>
```

Wenn Sie die Datei in einem Browser öffnen, wird das Ergebnis wie in Abbildung 1–1 in einem Dialogfeld angezeigt:

Abb. 1–1 *Ausführung von JavaScript-Code in einem Browser*

Sie können auch kurze Folgen von Anweisungen in die Konsole eingeben, die zu den Entwicklerwerkzeugen des Browsers gehört. Suchen Sie den Menübefehl oder die Tastenkombination zur Anzeige dieser Werkzeuge. (In vielen Browsern ist das die Taste F12 oder die Kombination Strg + Alt + I bzw. cmd + alt + I auf dem Mac.) Bringen Sie dann die Registerkarte *Konsole* in den Vordergrund und geben Sie darin Ihren JavaScript-Code ein (siehe Abb. 1–2).

Abb. 1–2 *Ausführung von JavaScript-Code in der Entwicklerkonsole*

Eine dritte Möglichkeit besteht darin, Node.js von `http://node.js.org` zu installieren, ein Terminalfenster zu öffnen und darin das Programm `node` auszuführen. Dadurch wird eine JavaScript-»REPL« gestartet (Read-Eval-Print Loop, also etwa »Lese-, Ausführungs- und Ausgabeschleife«). Darin können Sie Befehle eingeben und sich die Ergebnisse anzeigen lassen (siehe Abb. 1–3).

Abb. 1–3 *Ausführung von JavaScript-Code in der Node.js-REPL*

Wenn Sie längere Codefolgen ausführen wollen, schreiben Sie die Anweisungen in eine Datei. Für Ausgaben verwenden Sie dabei die Methode `console.log`. Beispielsweise können Sie die folgenden Anweisungen in eine Datei aufnehmen, die Sie `first.js` nennen:

```
let a = 6

let b = 7

console.log(a * b)
```

Führen Sie anschließend den folgenden Befehl aus:

```
node first.js
```

Im Terminal wird nun die Ausgabe des Befehls `console.log` angezeigt.

Sie können auch eine Entwicklungsumgebung wie Visual Studio Code, Eclipse, Komodo IDE oder WebStorm verwenden. Darin können Sie, wie in Abbildung 1–4 gezeigt, JavaScript-Code bearbeiten und ausführen:

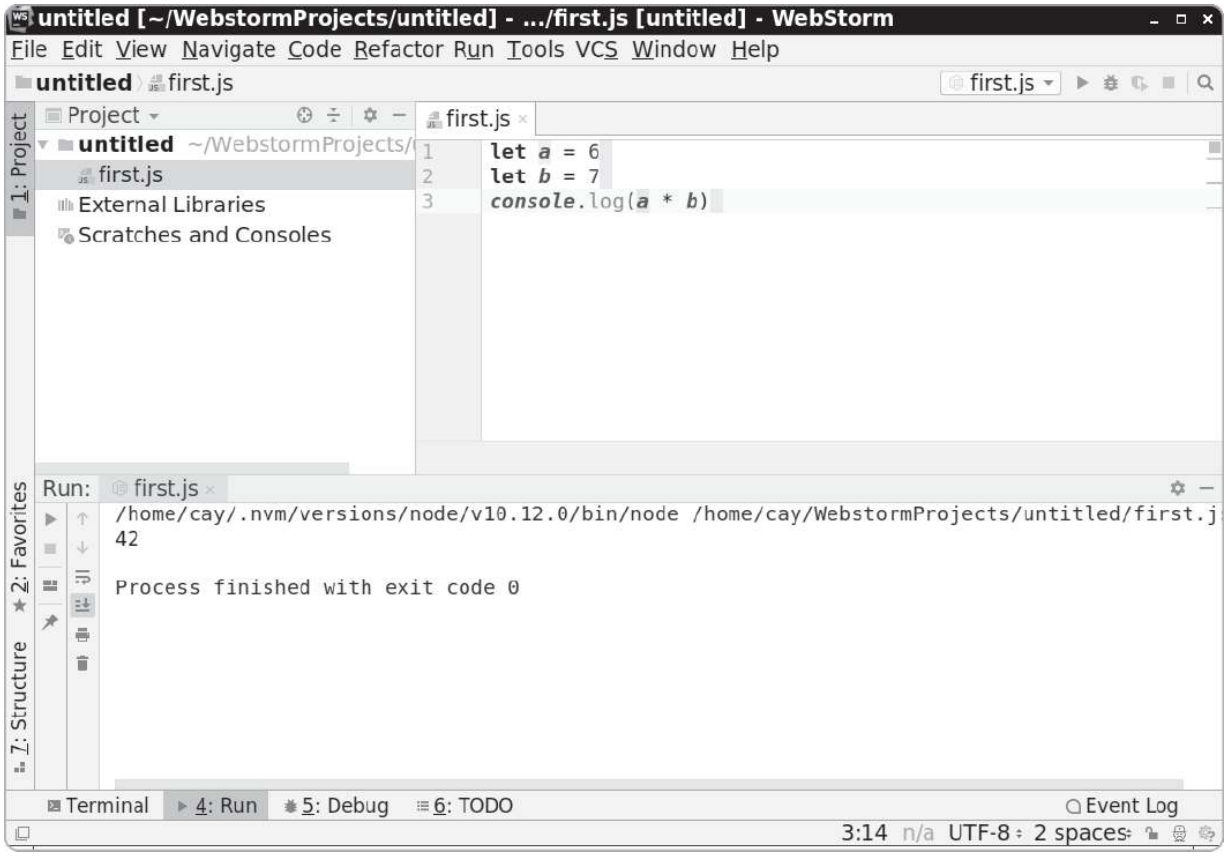

Abb. 1–4 *Ausführung von JavaScript-Code in einer Entwicklungsumgebung*

1.2 Typen und der Operator typeof

Werte in JavaScript sind jeweils von einem der folgenden Typen:

- eine Zahl
- einer der booleschen Werte `false` oder `true`
- einer der besonderen Werte `null` oder `undefined`
- ein String
- ein Symbol

- ein Objekt

Die Typen, die keine Objekte sind, werden zusammengenommen als *primitive Typen* bezeichnet.

Mehr über diese Typen erfahren Sie in den folgenden Abschnitten. Die einzige Ausnahme bilden die Symbole, die erst in Kapitel 11 behandelt werden.

Den Typ eines gegebenen Werts können Sie mit dem Operator `typeof` herausfinden, der einen der Strings `'number'`, `'boolean'`, `'undefined'`, `'object'`, `'string'` oder `'symbol'` oder einen von wenigen weiteren möglichen Strings zurückgibt. Beispielsweise ergibt `typeof 42` den String `'number'`.

Hinweis

Obwohl der Typ `null` nicht identisch mit dem Typ `object` ist, ergibt `typeof null` den String `'object'`. Das ist leider historisch so gewachsen.

Vorsicht

Ähnlich wie in Java können Sie in JavaScript Objekte als Wrapper für Zahlen, boolesche Werte und Strings konstruieren. Beispielsweise werden sowohl `typeof new Number(42)` als auch `typeof new String('Hello')` zu `'object'` ausgewertet. Allerdings gibt es in JavaScript keinen sinnvollen Grund dafür, solche Wrapper-Instanzen zu erstellen. Da sie eher für Verwirrung sorgen, ist ihre Verwendung in vielen Programmierstandards untersagt.

1.3 Kommentare

In JavaScript können Sie zwei verschiedene Arten von Kommentaren einfügen. Einzeilige Kommentare beginnen mit `//` und laufen bis zum Ende der Zeile:

```
// Einzeiliger Kommentar
```

Dagegen können mit `/*` und `*/` abgegrenzte Kommentare mehrere Zeilen umspannen:

```
/*

   mehrzeiliger

   Kommentar

*/
```

In diesem Buch verwende ich eine Serifenschrift, um die Kommentare in Listings deutlicher hervorzuheben. In Ihrem Texteditor werden die Kommentare wahrscheinlich farbig gekennzeichnet sein.

Hinweis

Anders als Java bietet JavaScript keine besondere Formatierung für Kommentare zur Dokumentation. Für diesen Zweck können Sie jedoch Drittanbieterwerkzeuge wie JSDoc (http://usejsdoc.org) nutzen.

1.4 Variablendeklarationen

Mit der Anweisung `let` können Sie einen Wert in einer Variablen speichern:

```
let counter = 0
```