Ralph Steyer

Programmieren mit JavaScript

Die vielseitige Sprache für Webentwicklung & mehr – Grundlagen und fortgeschrittene Techniken

Springer Vieweg

Programmieren mit JavaScript

Ralph Steyer

Programmieren mit JavaScript

Die vielseitige Sprache für Webentwicklung & mehr – Grundlagen und fortgeschrittene Techniken

Springer Vieweg

Ralph Steyer
Bodenheim, Rheinland-Pfalz, Deutschland

ISBN 978-3-658-45576-7 ISBN 978-3-658-45577-4 (eBook)
https://doi.org/10.1007/978-3-658-45577-4

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über https://portal.dnb.de abrufbar.

Planung/Lektorat: David Imgrund
Springer Vieweg ist ein Imprint der eingetragenen Gesellschaft Springer Fachmedien Wiesbaden GmbH und ist ein Teil von Springer Nature.
Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Vorwort

JavaScript! Eine Skript- bzw. Programmiersprache, die sich längst etabliert hat. Lange Zeit wurde sie eher belächelt und nicht wirklich ernst genommen. In den letzten Jahren hat sich das massiv geändert. Aus dem hässlichen Entlein ist ein stolzer Schwan geworden. JavaScript schafft es jetzt schon viele Jahre in fast allen Statistiken zu den meistgenutzten bzw. wichtigsten Programmiersprachen weltweit auf das Podium.

Dazu trägt sicher bei, dass man mit JavaScript (oder kurz JS) nicht nur Leben, Dynamik und Aktivität in Webpräsentationen bringen kann, wie es am Anfang der Geschichte von JavaScript und auch die ersten Jahre danach der Fall war. JavaScript hat längst das Korsett eines Browsers, wo es die ideale Ergänzung zu HTML (Hyper Text Markup Language) und Style Sheets darstellt, verlassen. Dennoch ist ein Webbrowser immer noch das „natürliche Habitat" von JavaScript. Das ursprüngliche Ziel von Netscape (Erfinder von JavaScript), Webseiten zu dynamisieren, den Browser zu steuern und Aktivität vom Webserver auf den Clientrechner zu verlagern, wurde nie aufgegeben oder eingeschränkt und so gut wie keine Webseiten im Internet kommt ohne JavaScript aus. Im Gegenteil – im Web übernimmt JavaScript als einzig relevante Möglichkeit zum Programmieren im Browser immer mehr neue Aufgaben. Sogenannte Rich Internet Applications (RIAs) mit „reichhaltigen" Möglichkeiten, die Webapplikationen auf eine Stufe wie Desktopapplikationen stellen, basieren im Client wesentlich auf JavaScript. Dies umfasst sowohl viele optische als auch funktionale Features wie animierte Inhaltsaufbereitung, das dynamische Nachladen von Daten oder komfortable Benutzereingabemöglichkeiten. Auch wenn dies oft HTML5 zugeordnet wird. HTML5 bietet vielfach nur in Verbindung mit JavaScript echten Nutzen oder ist sogar bei einigen Features in Wirklichkeit nur JavaScript selbst.

Die Einsatzgebiete von JavaScript haben sich aber parallel ebenfalls jenseits des Browsers massiv erweitert. So gibt es – um erst einmal im Web zu bleiben – Webserver, die auf JavaScript als zentrale Technik zur serverseitigen Programmierung setzen. Das schafft einmal die Situation, dass sowohl im Client (Browser) als auch Server (Webserver) die gleiche Programmiertechnologie verwendet wird. Das erleichtert viele Dinge. Zudem hat sich gezeigt, dass man auf Serverseite durch JavaScript sowohl in der Entwicklungszeit als auch der Performance im laufenden Betrieb riesige Vorteile erreichen

kann, die vor wenigen Jahren noch undenkbar gewesen waren. Auch in mächtige Sprachen wie Java bzw. JavaFX/FXML wird JavaScript mehr und mehr als Ergänzung ausdrücklich integriert und ebenso bei der Steuerung von Micro-Controllern setzt man verstärkt auf JavaScript.

JavaScript bietet uns also so viele interessante Möglichkeiten, dass eine Beschäftigung damit äußerst nützlich und vor allem spannend und unterhaltsam ist. Sie können dabei JavaScripts erstellen, ohne ein einziges Programm kaufen zu müssen. Zudem ist JavaScript einfach zu lernen und vor allem unabhängig von einer speziellen Plattform. Ob Sie Linux, Windows, macOS oder ein anderes Betriebssystem verwenden – JavaScript wird auf nahezu allen Plattformen zu finden sein. Sie können unter der einen Plattform entwickeln und Anwender können eine beliebige andere Plattform verwenden.

Um mit einer persönlichen Bemerkung zum Abschluss des Vorworts zu kommen. Ich nutze professionell gut ein Dutzend Programmiersprachen und habe vor allen Dingen sehr viel mit Java und C# zu tun. JavaScript, was ich seit dessen erstem Auftauchen schon verwende, ist aber klar meine Lieblingssprache, der im Moment höchstens Python halbwegs Konkurrenz machen kann. Und noch etwas – Programmieren mit JavaScript macht einfach Spaß.

Kontakt: http://www.rjs.de und http://blog.rjs.de

Frühjahr
Sommer 2024

Ihr Autor
Ralph Steyer

Inhaltsverzeichnis

Einleitung

1

Übersicht
Bevor es mit JavaScript richtig losgeht, sollen in diesem einleitenden Kapitel einige Dinge geklärt werden, die Ihnen die folgende Arbeit mit diesem Buch und der Sprache im Allgemeinen erleichtern werden. Insbesondere sorgen wir an der Stelle dafür, dass Ihnen die Voraussetzungen zum Programmieren und Ausführen von JavaScript überhaupt zur Verfügung steht.

1.1 Das Ziel des Buchs

JavaScript wurde ursprünglich im Jahr 1995 von Netscape für dynamisches HTML in Webbrowsern vorgestellt. Ursprünglich LiveScript genannt, wurde der Name in Kooperation mit Sun geändert, um eine gemeinsame Marketingstrategie mit dessen damals fast gleichzeitig veröffentlichter Sprache Java fahren zu können.

Netscape Communications war ein amerikanisches Softwareunternehmen, das in den 90er-Jahren äußert prägend für das WWW war. Netscape stellte 1994 den Browser Netscape Navigator als Nachfolger des NCSA Mosaic vor und vollzog eine wechselvolle Geschäftsentwicklung, bis es 2003 aufgelöst wurde. **Sun Microsystems, Inc.** war ein amerikanischer Hersteller von Hard- und Software, der im Jahr 2010 durch die Oracle Corporation übernommen wurde.

Die Bedeutung von JavaScript, die schon aktuell größer als je in der Geschichte des Webs ist, wächst aufgrund der immer stärkeren Verbreitung von Rich Internet Applications, mobilen Apps auf Basis von Webtechnologien, serverseitigem Scripting und diversen weiteren Zielgebieten. Nicht zuletzt hat HTML5 die Bedeutung von JavaScript noch einmal gesteigert und die Anforderungen an JavaScripts wachsen. Waren es früher hauptsächlich eher einfache, kleine Skripte, die eine Webseite ein bisschen ergänzt

R. Steyer, *Programmieren mit JavaScript,* https://doi.org/10.1007/978-3-658-45577-4_1

haben, sind aktuelle JavaScripts in der Regel tragende Bestandteile der RIAs oder serverseitiger Applikationen. Damit muss man in solchen Applikationen richtig professionell mit JavaScript programmieren. Moderne JavaScript-Applikationen werden auf dem gleichen Niveau konzipiert und erstellt wie Desktop-Applikationen mit .NET oder verteilte Client-Server-Applikationen – teilweise noch viel professioneller durch das heterogene Umfeld, in denen diese RIAs funktionieren müssen, sowie die mangelnden Fangzäune in der Sprache selbst, die Programmierer in Technologien wie .NET oder Java absichern. Bei JavaScript wird deshalb mehr die Qualität des Programmierers selbst gefordert. JavaScript schafft den Spagat zwischen einer einfachen Einsteigerlösung und – mittlerweile – einem hochprofessionellen, wenngleich für Programmierer aus strengen Sprachen etwas eigenwilligen, Programmierumfeld.

Gelegentlich nutzt man statt dem Begriff „Webseite“ auch „Website“. Es stellt sich die Frage, ob das identisch ist? Die Antwort ist aber nicht ganz einfach und nicht streng festgelegt. Der Unterschied kann einmal rein auf die Sprachpräferenz und regionale Konventionen reduziert werden. Es sind dann synonyme Begriffe. Jedoch wird teils „Website“ auch verwendet, um auf eine ganze Sammlung von Webseiten zu verweisen, die unter einer bestimmten Domain oder Subdomain gruppiert sind. Der Begriff steht also für die gesamte Präsenz im Internet, einschließlich aller Seiten, die unter einer bestimmten Domain oder Subdomain gehostet werden. „Webseite“ wird dann jedoch manchmal spezifischer für eine einzelne Seite innerhalb einer Website verwendet. Also etwa eine HTML-Datei.

Dieses Buch ist ein Praxisbuch für den ambitionierten Einstieg in JavaScript und eng verwandte Technologien sowie einige ergänzende Techniken. Es soll Ihnen sowohl beim Selbststudium helfen als auch Basis dafür sein, in entsprechenden Kursen und Seminaren JavaScript zu lernen.

- Wir erstellen im Laufe des Buchs immer wieder praktische Beispiele. Die Quellcodes des Buchs finden Sie nach Kapiteln und teils darin erstellten Projekten sortiert auf den Webseiten des Verlags bzw. in einem Repository auf GitHub. Die Namen der jeweilig aktuellen Dateien beziehungsweise Projekte werden als Hinweise oder direkt im Text vor den jeweiligen Beispielen angegeben und bei Bedarf wiederholt. Ich empfehle allerdings, dass Sie die Beispiele allesamt von Hand selbst erstellen. Das ist für Verständnis und Lernen eindeutig besser als ein reines Kopieren oder nur Anschauen.

1.2 An wen sich das Buch wendet

Das Buch wendet sich im Wesentlichen an Leser, die von Grund auf den sicheren Umgang im großen Umfeld der Programmierung mit JavaScript lernen wollen. Dies umfasst neben JavaScript Aspekte der allgemeinen Webprogrammierung sowie Grundlagen der Webserver-Programmierung. Dabei sind als Zielgruppe dieses Werks auch Webseitenersteller gedacht, die JavaScript als Ergänzung zur „normalen“ Arbeit betrachten und ein vollständiges Webprojekt (nicht nur reine HTML-Seiten) realisieren wollen. Dabei soll

der Umgang mit JavaScript und ergänzenden Technologien von Anfang an erklärt werden. Allerdings werden wir nicht auf einem zu niedrigen Level aufsetzen und auch bis zu fortgeschrittenen JavaScript-Techniken kommen. Von daher haben wir insbesondere in den letzten Kapiteln schon den Anspruch, richtig anspruchsvolle Techniken kennenzulernen und beherrschen. Wir beginnen jedoch sehr langsam und grundlegend.

Diese Beschreibung der Zielgruppe soll bedeuten, Sie benötigen nicht unbedingt Vorwissen im Bereich der Webprogrammierung und auch der Programmierung im Allgemeinen. Allerdings erleichtert es den Umgang mit dem Buch, wenn Sie schon einmal ein Programm oder Skript geschrieben haben (egal in welcher Programmiersprache) oder zumindest wissen, was Programmierung ist. Gerade in den späteren Kapiteln ist das von Vorteil. Was ich bei einem Leser bzw. einer Leserin voraussetzen möchte, ist Folgendes:

- Sie haben einen Rechner zur Verfügung und können ihn einigermaßen sicher bedienen. Das bedeutet, Sie kennen sich in Grundzügen mit Ihrem Betriebssystem aus.
- Sie sollten Zugang zum Internet haben und mit dem Internet zurechtkommen.
- Es ist sinnvoll, wenn Sie bereits Webseiten erstellt haben. Nicht notwendigerweise mit CSS oder Skripten, aber zumindest sollte das Thema in Grundzügen bekannt sein. Sie brauchen keinesfalls ein HTML-Experte zu sein, da dieses Thema für JavaScript nicht in jedem Detail bekannt sein muss. Aber grundsätzlich sollten Ihnen die Idee und das Konzept von HTML vertraut sein.
- Es ist wie gesagt hilfreich, wenn Sie bereits Programme oder Skripte erstellt haben, aber es ist nicht zwingend notwendig. Allerdings ist ein professioneller Programmierhintergrund sicher kein Nachteil. Sollten Sie noch nie programmiert haben, können Sie das meines Erachtens anhand von JavaScript hervorragend lernen. Sie müssen sich nur etwas anstrengen und aufpassen, dass Sie sich ob der wenig strengen Regeln in JavaScript nicht gleich von Anfang an einen unsauberen Programmierstil angewöhnen. Ich gebe mir Mühe, Sie an einen sauberen und professionellen Programmierstil heranzuführen und Sie davon abzuhalten, die konzeptionellen Schwächen von JavaScript für eine schlampige Programmierung auszunutzen.

1.3 Voraussetzungen

Es gibt ein paar Dinge, die Sie unbedingt brauchen, wenn Sie erfolgreich Programmierung mit JavaScript lernen und in der Praxis anwenden wollen. Aber keine Angst – viel ist das nicht und kosten tut es auch wenig bis gar nichts.

1.3.1 Hardware

Natürlich ist ein Computer mit Internetzugang notwendig. Meist wird dabei ein PC mit Intel-kompatibler Architektur die bevorzugte Wahl sein und darauf werden wir uns bei

Ausführungen beschränken. Die Fraktion der Apple-Anwender soll natürlich nicht vergessen werden, denn im Web ist sie (wie allgemein im medialen Umfeld) ziemlich stark vertreten. Aber grundsätzlich laufen JavaScripts auf allen wichtigen Computerplattformen, weshalb wir hier keine echten Besonderheiten beachten müssen. Ihr Computer braucht auch nicht sonderlich leistungsfähig zu sein.

1.3.2 Die Software

Die Erstellung von JavaScripts erfordert im Grunde außer einem beliebigen Editor und einem Browser zum Testen keine weitere Software. Gleichwohl ist eine Reihe von Programmen sinnvoll bzw. nützlich. Schauen wir das etwas genauer an.

1.3.2.1 Das Betriebssystem

Die im Rahmen des Buchs zum Einsatz kommenden Technologien sind – wie fast alle wichtigen Internettechnologien – plattformneutral. Sie lassen sich unter jedem modernen Betriebssystem nutzen und auch die Entwicklung von Webseiten sowie JavaScript-Applikationen kann man auf verschiedensten Plattformen durchführen. Für die meisten Leser werden sich eines der folgenden drei Betriebssysteme anbieten. Dies sind Linux, Windows oder macOS. Nutzen Sie das Betriebssystem, das Ihnen am besten passt und nicht zu alt ist. Eine Hürde könnte es darstellen, wenn Sie keine Rechte zur Installation von Software haben. Das ist bei Firmenrechnern oft der Fall. Um bestimmte Programme wie einen Webserver oder Node.js verwenden zu können, müssen Sie aber vermutlich Installationsrechte haben.

1.3.2.2 Der Editor bzw. die IDE

Wie erwähnt genügt zur Erstellung von JavaScripts oder auch Webseiten im Allgemeinen (und auch serverseitigen Sprachen wie PHP oder Beschreibungssprachen wie HTML oder XML) als Minimalausstattung ein reiner Klartexteditor, wie er bei jedem Betriebssystem mitgeliefert wird. In der Praxis verwenden fast alle professionellen Programmierer jedoch Programmierwerkzeuge, die sie bei der Erstellung und Analyse des Quelltextes unterstützen. Solche Programme kennen einige Bestandteile einer Programmier- oder Beschreibungssprache und unterstützen einfache und teilweise auch komplexere Standardvorgänge, beispielsweise das Maskieren (die kodierte Darstellung) von Sonderzeichen, das Einfügen von Quellcodeschablonen oder eine bessere Übersicht durch farbliche Kennzeichnung von bekannten Befehlen. Einige Editoren bieten auch die Befehle einer verwendeten Sprache direkt an – etwa durch Menüs oder Symbolleisten, wo man diese auswählen kann (auch mit der Maus). Notepad++ (https://notepad-plus.sourceforge.net/de/site.htm) unter Windows bzw. dessen „Kopie“ Notepadqq (https://notepadqq.com) unter Linux hat zum Beispiel so eine Unterstützung und ist im Grunde noch als echter Editor zu sehen. Bei vielen mächtigen Programmiersprachen wie Java, C/C++ oder C# kommen sogenannte **IDEs** (Integrated Development Environment) zum Einsatz.

Diese gestatten eine Programmierung und Ausführung von Applikation aus einer integrierten, gemeinsamen Oberfläche heraus. Microsoft stellt etwa mit Visual Studio eine sehr mächtige IDE zur Verfügung, die in der Community-Version kostenlos ist (https://www.microsoft.com/germany) und auch für die Webprogrammierung verwendet werden kann. Andere IDEs für Webentwicklung basieren auf der mächtigen Entwicklungsumgebung Eclipse (https://www.eclipse.org). Es gibt zudem noch eine ganze Reihe weiterer IDEs, die Sie gerne verwenden können. Allerdings sollten Sie beachten, dass der Umgang mit diesen umfassenden IDEs oft nicht ganz trivial ist und sie meist sehr „überladen" daherkommen. Zudem fordern sie oft nicht unerhebliche Ressourcen von der Hardware. Visual Studio Code oder kurz VS Code (https://code.visualstudio.com/), was eine leichtere, abgespeckte Version von Visual Studio darstellt und eigentlich näher an einem Editor denn einer IDE ist, ist meines Erachtens eine perfekte Wahl zur Programmierung mit JavaScript und anderen Webtechnologien. VS Code steht für verschiedene Betriebssysteme bereit und bietet Codevervollständigung, Syntax Highlighting, einen Navigationsbaum etc. VS Code wird im Folgenden das Referenzsystem darstellen.

1.3.2.3 Verschiedene Browser und JavaScript-Engines

Gerade wenn Sie JavaScript für den Einsatz in Webseiten erstellen, sollten Sie mehrere Browser zur Verfügung haben. Zwar sind die früher leider üblichen hohen Abweichungen in der Darstellung und Funktionalität von Webseiten in verschiedenen Browsern mittlerweile nicht mehr so gravierend. Dennoch sollte man unbedingt in verschiedenen relevanten Browsern Webseiten bzw. JavaScripts testen. Denn in der Regel wissen Sie ja nicht, mit welchen Browsern Ihre Besucher auf Ihren Webseiten vorbeikommen. Moderne Browser sind aber nicht nur primitive Anzeigeprogramme für Webseiten, sondern mit vielfältigen Features aufgebohrt worden, die auch die Webentwicklung betreffen. Es gibt in allen Browsern **Entwicklertools,** die Sie meist mit der Taste F12, einem entsprechenden Menübefehl oder dem Kontextmenü auf einer Webseite aufrufen können. Dazu gibt es meist Erweiterungen (Add-ons), die für die Webentwicklung sinnvoll sein können. Ebenso haben moderne Browser allesamt eine integrierte JavaScript-Engine zur Verfügung.

Hintergrundinformation

Eine **JavaScript-Engine** ist vereinfacht ausgedrückt ein Programm, das JavaScript-Code ausführen kann. Sie besteht in der heutigen Zeit in der Regel aus einem Compiler, der den JavaScript-Code bei der Ausführung in Maschinencode übersetzt, und einem Laufzeitsystem, das den übersetzten Code ausführt und die erforderlichen Umgebungen bereitstellt, um den Code auszuführen.

Einige bekannte JavaScript-Engines sind das:

- **V8** wurde von Google entwickelt und ist eine der am häufigsten verwendeten JavaScript-Engines. Das Programm wird in Google Chrome und dem Node.js-Laufzeitumfeld verwendet.
- **SpiderMonkey** ist die JavaScript-Engine, die von Mozilla entwickelt wurde und in Firefox verwendet wird. SpiderMonkey stellte einen der ersten JavaScript-Compiler bereit und hat seitdem viele Optimierungen erfahren.
- JavaScriptCore wird von Apple entwickelt und ist in Safari und einigen anderen Apple-Produkten integriert. Es ist auch die Engine, die im WebKit-Projekt verwendet wird.

- **Chakra** wurde ursprünglich von Microsoft entwickelt und in älteren Versionen des Internet Explorer-Browsers verwendet. Später wurde es durch **ChakraCore** ersetzt, eine Open-Source-Version, die in Microsoft Edge und Node.js integriert ist.
- **Rhino** ist eine JavaScript-Engine, die in Java geschrieben wurde und die von Mozilla entwickelt wurde. Rhino unterstützt die Ausführung von JavaScript in Java-Anwendungen.

Diese Engines sind entscheidend für die Ausführung von JavaScript-Code auf verschiedenen Plattformen und in verschiedenen Anwendungsfällen und haben im Laufe der Zeit erhebliche Fortschritte bei der Leistung und den Funktionen gemacht. Mittlerweile bemerkt man auch so gut wie keine relevanten Unterschiede hinsichtlich des Verhaltens.

Nun ist der Begriff des **Compilers** gefallen und man muss den Kontext ausführlicher betrachten. Denn JavaScript ist von Anfang an eine reine Interpreter-Sprache gewesen. Interpreter als auch Compiler dienen der Übersetzung von Quellcode und die Konzepte sollen nicht einfach vorausgesetzt werden. Insbesondere nicht das JIT-Konzept, das moderne JavaScript-Engines verfolgen. Wenn Sie Skripte oder Programme erstellen, schreiben Sie den sogenannten **Quellcode** oder **Quelltext**, der der englischen Sprache angelehnt ist. Aber zur Ausführung muss dieser übersetzt werden, und zwar in ein Format, dass ein Computer versteht und das ist **Maschinencode**. Dazu gibt es verschiedene Möglichkeiten.

- Einmal gibt es den **Interpreter**. Dieser sorgt für eine Übersetzung in Maschinencode während der Programmausführung. Interpreter übersetzen den Quellcode eines Programms zeilenweise.
- Die Alternative ist der **Compiler**, der den kompletten Quelltext vor der Laufzeit übersetzt.
- In den vergangenen Jahren hat sich auch eine **zweistufige Übersetzung** etabliert – die Übersetzung mit Zwischencode. Das wird in Java oder dem .NET-Framework gemacht. Denn ein Nachteil bei kompilierten Programmen ist, dass diese Programme maschinenabhängig sind und somit nicht auf jeder Computerplattform, zum Beispiel Linux und Windows, ausgeführt werden können. In Java oder bei der .NET-Plattform wird Quellcode nicht zu einem ausführbaren Programm, sondern in einen Zwischencode, kompiliert (1. Schritt). Dieser Code ist für alle Plattformen gleich und kann mithilfe des entsprechenden plattformspezifischen Interpreters (2. Schritt) auf der jeweiligen Plattform ausgeführt werden.

Bei JavaScript als interpretierte Sprache, wird Quellcode Zeile für Zeile ausgeführt wird und alte Webbrowser führen den JavaScript-Code auf diese Weis aus, was zu relativ langsamer Ausführung führte. Just-In-Time-Compilation (JIT) ist ein Ansatz, um die Leistung von interpretierten Sprachen zu verbessern. Bei der JIT-Kompilierung wird der JavaScript-Code nicht nur interpretiert, sondern auch in maschinenlesbaren Code übersetzt, der direkt von der CPU ausgeführt werden kann. Dieser Prozess geschieht zur Laufzeit, was bedeutet, dass der JIT-Compiler während der Ausführung des Programms aktiv ist. So funktioniert die moderne JIT-Kompilierung in JavaScript-Engines:

1. JavaScript-Code wird zunächst interpretiert, um die Ausführung zu starten. Dies ermöglicht eine schnelle Bereitstellung des Codes, da kein komplexer Kompilierungsprozess durchlaufen werden muss.
2. Dann findet sogenanntes Profiling statt: Während der Interpretation sammelt der JIT-Compiler Informationen über den tatsächlichen Code, der ausgeführt wird, wie häufig bestimmte Funktionen aufgerufen werden und welche Typen von Daten verwendet werden.
3. Daraufhin beginnt eine Optimierung, die auf den gesammelten Profiling-Daten basiert. Insbesondere identifiziert der JIT-Compiler Abschnitte des Codes, die häufig ausgeführt werden und die für eine Leistungssteigerung optimiert werden könnten. Es kann verschiedene

Optimierungstechniken anwenden, wie z. B. das Entfernen von unnötigen Operationen oder das Inline-Aufrufen von Funktionen.
4. Die nun folgende Kompilierung übersetzt die optimierten Teile des Codes in maschinenlesbaren Code, der direkt von der CPU ausgeführt werden kann. Dieser Prozess kann mehr Zeit in Anspruch nehmen, aber da er nur für die am häufigsten verwendeten Codeabschnitte durchgeführt wird, ist der Overhead akzeptabel.
5. Durch Iteration wird dieser Prozess immer wieder dynamisch wiederholt, während das Programm läuft. Wenn sich das Verhalten des Codes ändert oder neue Teile des Codes ausgeführt werden, passt sich der JIT-Compiler an und optimiert den Code entsprechend neu.

Durch die Verwendung von JIT-Kompilierung können JavaScript-Engines die Ausführungsgeschwindigkeit erheblich verbessern, indem sie den Code dynamisch optimieren, während er ausgeführt wird. Dies trägt dazu bei, dass JavaScript-Anwendungen performanter und reaktionsschneller als früher sind.

1.3.2.4 Node.js

Wenn wir JavaScript ohne Browser als Umgebung nutzen als auch uns mit serverseitiger JavaScript-Programmierung sowie fortgeschrittenen JavaScript-Techniken beschäftigen wollen, werden wir auf ein Projekt mit Namen **Node.js** beziehungsweise NodeJS setzen. Bei Node.js (https://nodejs.org) handelt es sich erst einmal im weiteren Sinn sowohl um ein Framework als auch um ein Interpreter-System unter der MIT-Lizenz. Node.js ist also eine Open-Source-JavaScript-Laufzeitumgebung, die auf der V8-JavaScript-Engine von Google basiert. Genaugenommen versteht man Node.js als eine Plattform zum Betrieb von Netzwerkanwendungen und das können auch Webserver sein. Kompakt kann man sich merken – Node.js ermöglicht die Ausführung von JavaScript-Code außerhalb eines Webbrowsers und wurde entwickelt, um hochskalierbare Anwendungen zu erstellen und zu betreiben. Node.js ist modular aufgebaut und es gibt einige Standardmodule, die direkt in das Binärpaket kompiliert werden, das Sie bei einer Standardinstallation erhalten. Aber es können auch zusätzliche Module eingebunden werden, was wir an mehreren Stellen machen werden.

▶ In JavaScript bezieht sich ein **Modul** auf eine Möglichkeit, Code in separaten Dateien zu organisieren und zu strukturieren. Diese Dateien können dann über Import- und Exportanweisungen verbunden werden, um Funktionalität zwischen ihnen zu teilen.

Zur Verwaltung der Module gibt es den Paketmanager **npm**, der mittlerweile auch von vielen verteilten OpenSource-Projekten zur Softwareverwaltung eingesetzt wird. Node.js steht für Linux, macOS, Windows und weitere Betriebssystem zur Verfügung und kann über einen einfachen Assistenten auf einem Computer installiert werden. Das sollten Sie tun, wenn Ihnen Node.js noch nicht zur Verfügung steht, denn wir werden in der Folge recht oft auf Node.js zum Ausführen von JavaScripts setzen. Unter https://nodejs.org/en/download/ (Abb. 1.1) erhalten Sie die aktuellsten Setup-Dateien für Ihr Betriebssystem zum Download.

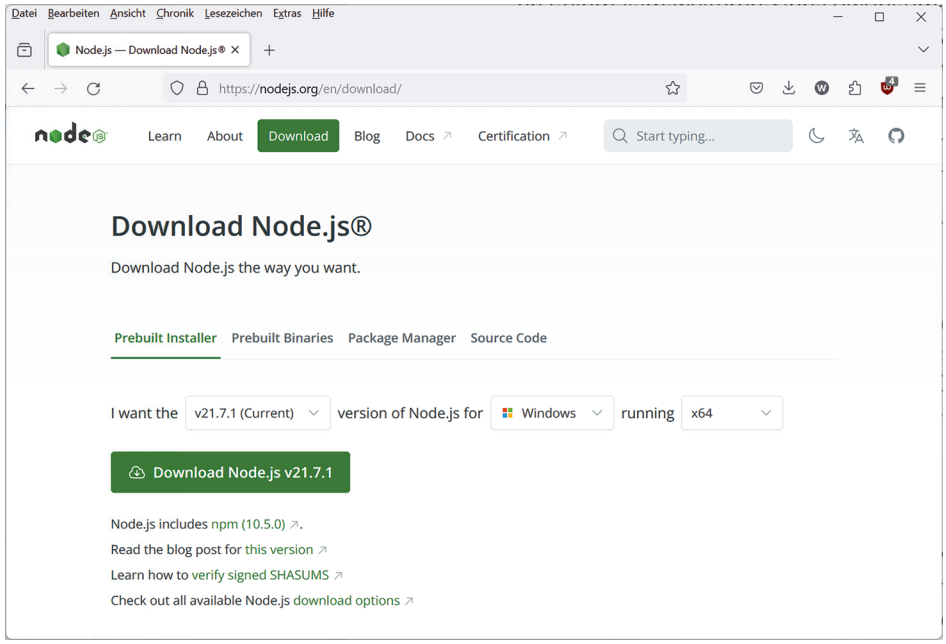

Abb. 1.1 Hier gibt es Node.js

1.3.2.5 Webserver

Auch wenn Sie eine HTML-Datei direkt vom Dateisystem in den Browser laden und damit ein enthaltenes JavaScript ausführen können, ist es besser, wenn Sie die Dateien stattdessen von einem Webserver anfordern. In einigen Situationen (etwa zum realistischen Durchführen einer Ajax-Datenanfrage) ist es meist sogar unabdingbar, dass Sie Ihre Webapplikation nicht direkt vom Dateisystem, sondern über das Protokoll *http* oder *https* vom Server anfordern. Dementsprechend benötigen Sie Zugang zu einem Webserver. Einige IDEs beinhalten einen internen Webserver. Aber es gibt auch andere Möglichkeiten. Entweder stellt Ihnen Ihr Provider einen solchen Zugang bereit oder aber Sie installieren auf Ihrem Rechner oder in Ihrem Netzwerk lokal einen solchen Server zum Testen. Unter Windows bietet sich der IIS (früher stand das für Internet Information Server und mittlerweile für Internet Information Services) von Microsoft an, besonders dann, wenn Sie sowieso mit Microsoft-Technologien arbeiten wollen. Für viele Fälle eignet sich eine Rundum-sorglos-Lösung namens **XAMPP**. XAMPP können Sie für verschiedene Betriebssysteme aus dem Internet laden können (unter https://www.apachefriends.org/de/). Hinter XAMPP steckt ein Non-profit-Projekt namens Apache Friends, das sich die Förderung des Apache Webservers und verbundener Technologien wie MySQL, PHP und Perl auf die Fahne geschrieben hat. Für XAMPP gibt es nach der Installation ein Control-Panel, worüber sich die einzelnen Dienste starten lassen (Abb. 1.2).

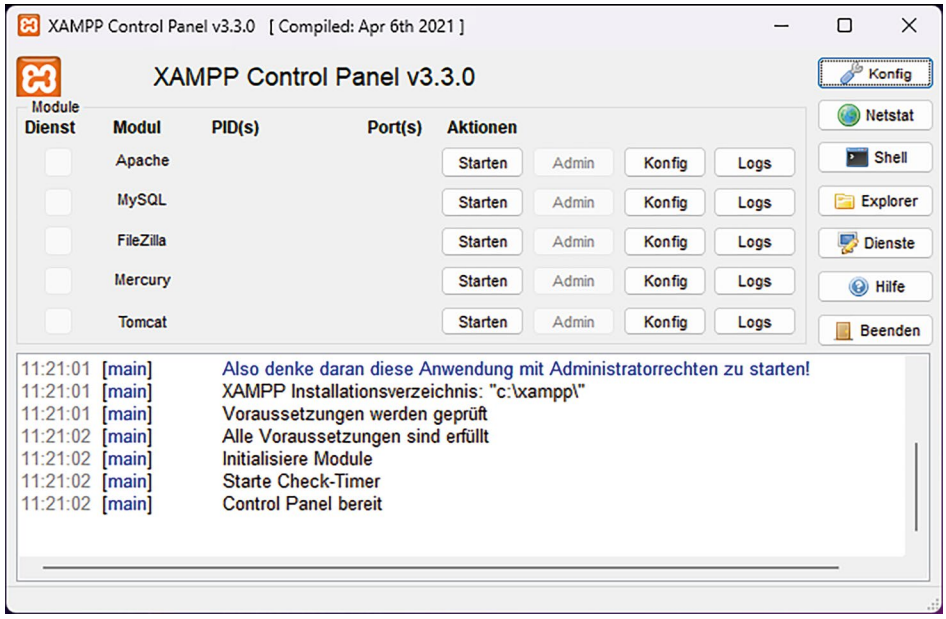

Abb. 1.2 Das Control-Panel von XAMPP

Der Webserver steht Ihnen nach dem Start über die Angabe von *localhost* in der Adresszeile des Browsers zur Verfügung, wenn Sie ihn auf dem gleichen Rechner laufen lassen wie Ihren Browser. Das *http*-Protokoll davor wird in der Regel automatisch vom Browser ergänzt. Andernfalls geben Sie in der Adresszeile einfach die IP-Nummer oder den Namen des Rechners ein, auf dem der Webserver läuft.

Die Angabe *localhost* ist ein Alias für den aktuellen Rechner. Mithilfe dieser **sogenannten Loopback-Adresse**, die der IP-Nummer 127.0.0.1 entspricht, können Sie auf Ihren eigenen Rechner so zugreifen, als würden Sie dies über ein Netzwerk tun. Sie benötigen dazu nicht einmal eine Netzwerkkarte.

Das Bereitstellen von Daten über den Webserver erfolgt bei Apache in der Regel über das Unterverzeichnis *htdocs*. Dieses ist das Wurzelverzeichnis aller Zugriffe über den Webserver. Oder mit anderen Worten: Wenn ein Anwender die Adresse des Webservers ohne weitere Verzeichnis- oder Dateiangaben angibt, bekommt er den Inhalt dieses Verzeichnisses beziehungsweise die Vorgabedatei darin angezeigt. Wurzelverzeichnis eines Webservers bedeutet aber auch, dass sämtliche Verzeichnisangaben beim Zugriff aus einem Browser auf einem Host ab diesem Verzeichnis aus gesehen werden. Wo konkret sich *htdocs* im Verzeichnisbaum des Rechners befindet, kann je nach Konfiguration von Apache individuell angegeben werden. Bei XAMPP unter Windows kann es sich zum Beispiel auf dem Installationslaufwerk unter *apachefriends* und dann *xampp* befinden.

Das sieht aber ein Aufrufer der Webseite nicht und darf dieses aus Sicherheitsgründen auch nicht sehen.

- Wenn Sie die folgenden Beispiele des Buchs zum Testen über den Apache-Webserver anfordern wollen oder müssen, können Sie die Dateien direkt in *htdocs* bereitstellen, aber es ist zu empfehlen, in *htdocs* ein eigenes Verzeichnis für unsere Experimente zu erstellen und diese sogar für einzelne Projekte bzw. Kapitel noch zu unterteilen.

- Beachten Sie, dass bei Pfadangaben unter Apache in der Regel Groß- und Kleinschreibung relevant ist. Gerade Windows-Anwender sind das meist nicht gewohnt, aber dies ist ja auch im Internet so gut wie immer der Fall. Am besten benennen Sie alle Verzeichnisse und Dateien konsequent klein. Vermeiden Sie auch Sonderzeichen oder deutsche Umlaute bei Verzeichnis- und Dateinamen.

Erste Beispiele – Der Sprung ins kalte Wasser

2

Übersicht

In dem Kapitel kommen wir zum Erstellen erster JavaScript-Programme beziehungsweise -Skripte und Sie lernen, wie Sie diese ausführen können. Dabei werden wir verschiedene Wege kennenlernen bzw. im ersten Einsatz sehen. Damit sind Tools und Strategien gemeint, die Sie im Umgang mit JavaScript immer wieder benötigen. Dazu möchte ich Ihnen direkt ein paar einfache JavaScript-Anwendungen vorstellen, deren Syntax erst an späterer Stelle tiefer beleuchtet wird. Geben Sie jeweils nur den Quelltext exakt wie vorgegeben ein (mit Beachtung der Groß- und Kleinschreibung, Klammern, Semikolon etc.). Bevor wir uns in Details einarbeiten, schaffen wir uns also praktische Grundlagen, damit nachfolgende Schritte leichter verständlich werden. Wichtig ist hier, dass Sie den Umgang mit einem Editor und der grundsätzlichen Erstellung von Quelltext sowie einem Browser (in Bezug zu JavaScript) und Node.js üben und dabei erste Effekte sehen, die auf JavaScript beruhen.

2.1 Der Interaktivmodus – die Kommandozeile bei JavaScript

Nachfolgend erstellen wir ein paar ganz einfache Programme[1] mit minimaler Syntax, die wir in der sogenannten **Kommandozeile** ausführen wollen. Damit werden die eingegebenen JavaScript-Anweisungen unmittelbar ausgeführt. Das nennt man den **Interaktivmodus**. Wie schon erwähnt, ist die Verwendung von JavaScript traditionell mit einer Integration in eine Webseite verbunden, aber diese Beschränkung ist schon lange

[1] „Programme“ ist eigentlich auch schon zu hoch gegriffen. Aber es ist wirklich JavaScript, was Sie hier sehen werden.

R. Steyer, *Programmieren mit JavaScript,* https://doi.org/10.1007/978-3-658-45577-4_2

gefallen und war auch früher – etwas verborgen –nicht zwingend. JavaScript kann entweder in der Konsole eines Webbrowsers direkt ausgeführt werden oder man verwendet eine JavaScript-Laufzeitumgebung wie Node.js. In beiden Fällen wird keine Webseite als Gerüst bzw. Verankerung des Skripts benötigt. Allerdings können Sie dann auch nicht auf die Bestandteile der Webseite wie Formulare oder Grafiken zugreifen. Wenn Sie in einem Browser die Konsole öffnen (beispielsweise mit einem Rechtsklick in eine beliebige Webseite und Öffnen der Entwicklertools oder dem zugehörigen Menübefehl oder meist auch der Taste F12), können Sie dort direkt JavaScript-Anweisungen notieren. Zum Beispiel für die Berechnung des Ergebnisses von 1 + 1. Sie brauchen bloß 1 + 1 (oder auch irgendeine andere einfache mathematische Formel) dort einzugeben, um das Ergebnis in der Konsole zu sehen (Abb. 2.1).

Die Berechnung wird mit JavaScript durchgeführt und sie sehen das Ergebnis 2, denn in der Konsole eines modernen Browsers haben Sie unmittelbar Zugriff auf dessen integrierter JavaScript-Engine. Teils müssen Sie mit der Enter-Taste die Eingabe bestätigen, bevor Sie das Ergebnis sehen. Teils wird das Ergebnis direkt schon angezeigt, wenn Sie noch bei der Eingabe sind. Zumindest soweit die Eingabe schon auszuwerten ist. Das geht auch mit Node.js. Wenn Sie Node.js mit Standardeinstellungen installiert haben, öffnen Sie eine Eingabeaufforderung/Shell/Konsole Ihres Betriebssystems und geben dort *node* ein. Sie gelangen danach im Eingabemodus von Node.js (**REPL**) und können da vollkommen analog wie in der Browserkonsole JavaScript-Befehle eingeben (Abb. 2.2).

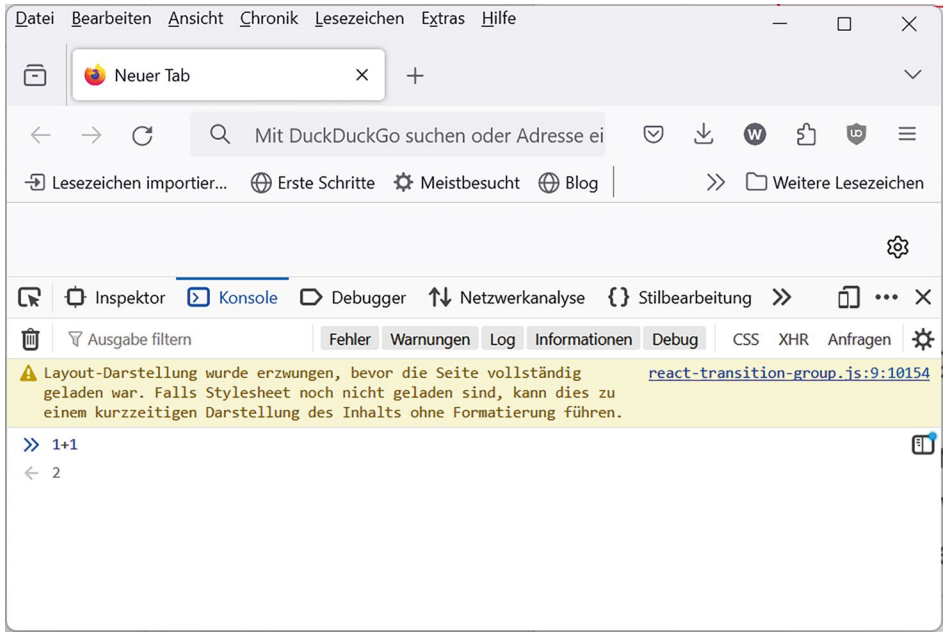

Abb. 2.1 Interaktivmodus in der Konsole von Firefox

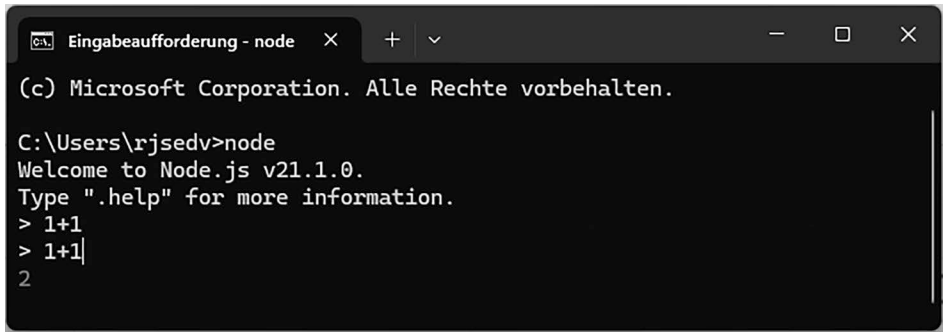

Abb. 2.2 REPL bei Node.js

REPL steht für Read Evaluate Print Loop und ist eine Programmiersprachenumgebung (im Grunde ein Konsolenfenster), die einen einzelnen Ausdruck als Benutzereingabe akzeptiert und das Ergebnis nach der Ausführung an die Konsole zurückgibt. Verwandte Umgebungen kennt man bei MySQL oder Python.

In der REPL können Sie analog wie in der Browserkonsole JavaScript-Befehle eingeben und Ausgaben durch JavaScript vornehmen. Sie sehen auch da nach der Bestätigung mit der Enter-Taste unmittelbar das Ergebnis. Die REPL-Sitzung bietet damit allgemein eine bequeme Möglichkeit, einfachen JavaScript-Code schnell zu testen, aber auch JavaScript-Code aus Dateien zu laden oder in Dateien zu speichern. Dabei gibt es in der REPL ein paar grundlegende Befehle, die Sie kennen sollten. Um diese von JavaScript-Anweisungen deutlich zu trennen, beginnen diese alle mit einem Punkt. Mit der Anweisung *.exit* (beachten Sie den vorangehenden Punkt) beenden Sie die Node.js-Konsole. Dazu gibt es weitere REPL-Anweisungen. Mit *.help* können Sie sich diesen Befehlssatz der REPL anzeigen lassen. Wichtig sind die Anweisungen *.load* zum Laden einer externen JavaScript-Datei in die REPL-Session und *.save* zum Speichern aller ausgeführten Befehle in der aktuellen REPL-Session in eine Datei. Eine externe JavaScript-Datei können Sie aber auch als Parameter an Node.js beim Aufruf übergeben und damit direkt ausführen, ohne in den Eingabemodus von Node.js gehen zu müssen. Das werden wir gleich im auch tun.

2.2 Anweisungen in (echten) Quelltext auslagern

Wenn Sie in der Kommandozeile direkt Anweisungen notieren, werden diese nach Eingabe der Enter-Taste unmittelbar interpretiert, also einzeln und nacheinander im Rahmen der speziellen Umgebung (Interaktivmodus). In der Regel wird man aber Anweisungen in eine extra Textdatei auslagern – den Quelltext (auch Sourcecode oder Quellcode), der dann ohne Bestätigen der einzelnen Anweisungen abgearbeitet wird. Das hat auch den

Vorteil, dass die Anweisungen wiederverwendet werden können. Diesen Klartext kann man mit einem beliebigen Editor erstellen.

- Die übliche **Dateierweiterung** für JavaScript--Quellcode ist *.js*. Das entspricht auch der Tatsache, dass man gelegentlich JavaScript durch JS abkürzt.

Die Anweisung zur Berechnung zweier Zahlen, die direkt in die Konsole eines Browsers oder der REPL eingegeben und dann dort ausgegeben wurden, kann man in eine externe Datei mit JavaScript-Quellcode auslagern. Diese kann in eine Webseite eingebunden oder von Node.js direkt geladen und ausgeführt werden. Der reine Quellcode für Node.js und einen Browser wird identisch sein, sofern nicht auf Elemente einer Webseite zugegriffen werden soll. Da wir uns in der Situation jedoch nicht innerhalb der JavaScript-Laufzeitumgebung befinden, muss man angeben, wo die Ausgabe von einer eventuellen Berechnung erfolgen soll. Man muss erst einmal festhalten, dass JavaScript selbst keinerlei Möglichkeiten hat, eine Eingabe oder Ausgabe bereitzustellen. Diese sind jedoch ganz entscheidend, denn Computerprogramme arbeiten so gut wie immer nach dem EVA-Prinzip.

▶ Das **EVA-Prinzip** steht für **E**ingabe, **V**erarbeitung und **A**usgabe und beschreibt ein grundlegendes Konzept in der Informationstechnologie, ohne dessen Einhaltung Software meist nicht sinnvoll genutzt werden kann.

Wie kann man das Dilemma lösen? JavaScript läuft im Rahmen einer Laufzeitumgebung und diese stellt eine Schnittstelle mit passenden Objekten bereit. Da JavaScript objektbasierend (in neueren Versionen sogar objektorientiert) ist, gibt es also als Ergänzung der eigentlichen Kernsprache zur Ausgabe passende Objekte.

▶ **Objekte** sind fundamentale Datenstrukturen, die verwendet werden, um Daten zu organisieren und zu speichern. Objekte können **Eigenschaften/Attribute** haben sowie **Methoden**, die aufgerufen werden können. Zusammengefasst nennt man Eigenschaften/Attribute und Methoden **Member**. Für den Zugriff auf die Member verwendet man die **Punktnotation**. Sie wird durch einen Punkt (.) zwischen dem Objektnamen und dem Eigenschaftsnamen oder dem Methodennamen dargestellt. Wenn Sie die Punktnotation verwenden, um auf eine Eigenschaft zuzugreifen, erhalten Sie den Wert dieser Eigenschaft. Wenn Sie sie verwenden, um auf eine Methode zuzugreifen, können Sie die Methode aufrufen und deren Rückgabewert verwenden.

Das Objekt zur Ausgabe in der Konsole nennt sich passender Weise *console* und wird sowohl von einem Browser als auch Node.js bereitgestellt. Mit dem Objekt *console* und dessen Methode *log* erhalten Sie Zugang zu einer Ausgabe in der Konsole.

- Mit der *log*-Methode können Sie einen Parameter oder auch mehrere – durch Kommata getrennte –Parameter in einer Zeile der Konsole ausgeben.

Daher notieren Sie in die JavaScript-Datei *javascriptdatei1.js* folgenden Code:

```
console.log (1 + 1);
```

Beachten Sie Groß- und Kleinschreibung, die Klammern und das Semikolon am Ende. Sie führen diese JavaScript-Datei *javascriptdatei1.js* so mit Nodes.js ohne explizites Öffnen der RPEL aus, wenn Sie sich in dem Verzeichnis mit der JavaScript-Datei befinden:

Beispiel

node javascriptdatei1.js. ◀

Nach dem oben genannten Aufruf sehen Sie das Ergebnis in der Konsole, aus der Sie *node* aufgerufen haben(Abb. 2.3).

Der externe JavaScript-Code kann auch in eine Webseite eingebunden werden. Diese könnte in einer ganz einfachen Form so aussehen *(kap 2_1.html):*

```
<html><body><script src="lib/js/javascriptdatei1.js"></script>
</body></html>
```

▶ Wir werden aus Platzgründen insbesondere HTML-Code im Buch kompakt formiert ausgeben und wiederholende oder nicht relevante Passagen immer wieder weglassen. Die vollständigen Quellcodes finden Sie zum Download. In der Praxis werden Sie den Quellcode auch meist übersichtlicher schreiben.

Diesen Quellcode sollten Sie in ein eigenes Verzeichnis speichern. Etwa *kap2*. Beachten Sie, dass die JavaScript-Datei in einem Unterverzeichnis *lib* und dessen Unterverzeichnis *js* gespeichert ist. So Strukturierung ist in der Praxis fast immer zu finden. Wenn Sie die HTML-Datei direkt per Doppelklick oder über einen Webserver laden, sehen Sie die

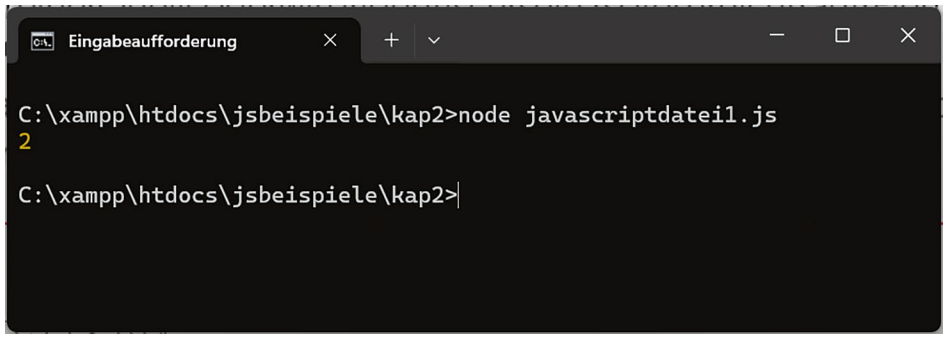

Abb. 2.3 Node.js hat die externe JavaScript-Datei geladen und ausgeführt

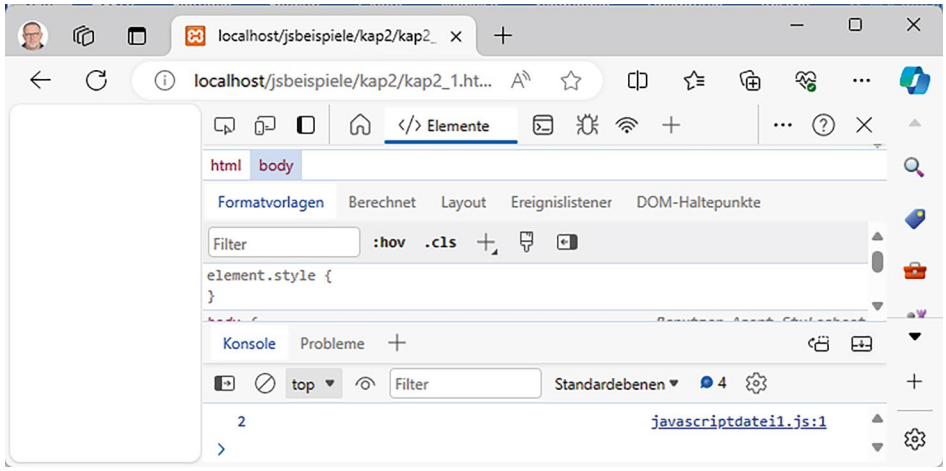

Abb. 2.4 Auch im Browser kann man mit der externen JavaScript-Datei in die Konsole schreiben – hier Edge

Ausgabe in der Konsole des Browsers (Abb. 2.4). Falls der Webserver auf dem gleichen Rechner läuft wie Ihr Webbrowser, würde der Aufruf wie folgt lauten, wenn Sie sich an meine Empfehlung bezüglich der Wahl eines Verzeichnisses gehalten haben und dieses dann im Installationsverzeichnis von Xampp unter *htdocs* bereitstellen:

Beispiel

http://localhost/kap2/kap2_1.html ◀

Sollte das Beispiel nicht funktionieren, überprüfen Sie zuerst Ihren Quelltext. Achten Sie auf Klammern, Hochkommata, Semikolon sowie Groß- und Kleinschreibung. Stimmt der Quellcode exakt mit der Vorgabe überein und das Beispiel funktioniert dennoch nicht, wird eventuell JavaScript in Ihrem Browser deaktiviert oder blockiert sein. Dann müssen Sie JavaScript in den Einstellungen Ihres Browsers aktivieren bzw. eine eventuell blockierende Erweiterung wie NoScript oder eine Firewall ausschalten, womit die Ausführung von Skripten unterbunden wird. Der Zugriff über http://localhost wird in der Regel (bei den meisten Browsereinstellungen) auch als sicher angesehen, der Zugriff über das *file*-Protokoll (einfaches Öffnen der Webseite mit einem Doppelklick) hingegen oft nicht.

▶ Das direkte Laden der JavaScript-Datei selbst in den Browser zeigt nur den Quelltext an. Er wird nicht interpretiert. JavaScript im Browser wird also immer (!) über eine Webseite an die JavaScript-Engine übergeben. Einzige Ausnahme ist der oben genannte Weg, dass Sie JavaScript-Befehle direkt in das Konsolenfenster

des Browsers schreiben, was dem Interaktivmodus in Node.js entspricht. In älteren Browsern konnte man mit einem vorangestelltem *javascript:* (das ist als Protokoll zu verstehen und der Doppelpunkt zwingend) in der Adressleiste des Browsers JavaScript-Anweisungen ausführen lassen. Das ist aber in vielen neueren Browser deaktiviert und zudem wenig sinnvoll.

2.3 Ein einfaches Mitteilungsfenster

Die Ausgabe bei einem Browser in die Konsole vorzunehmen ist unüblich und meist auch wenig sinnvoll. Ausgaben durch JavaScript sollten in der Webseite selbst oder einem visuellen Dialog des Browsers erfolgen. Kommen wir nun dazu, wie eine Ausgabe in einem Browser über ein einfaches Mitteilungsfenster vornehmen kann. Dazu muss das JavaScript wieder über eine Webseite (ein HTML-Dokument) in den Browser geladen werden. Dieses Mal soll aber ohne externe JavaScript-Datei gearbeitet werden, sondern JavaScript direkt in den HTML-Code geschrieben werden. Um – wie versprochen – die Theorie in diesem Kapitel knapp zu halten, wollen wir nur festhalten, dass JavaScript so oder so in einem Browser als Bestandteil einer Webseite zu sehen ist und damit in einer HTML-Seite referenziert wird. Damit haben Sie Zugang zum DOM, was eine Schnittstelle von Objekten des Browsers darstellt. Diese wiederum stellen zahlreiche nützliche Eigenschaften und Methoden bereit. Beispielsweise die Methode *alert*, um ein Dialogfenster anzuzeigen.

▶ **DOM** steht für Document Object Model, und ist eine Spezifikation einer Programmierschnittstelle mit Objekten, welches vor allen Dingen HTML- oder XML-Dokumente (XML – Extensible Markup Language) als eine Baumstruktur darstellt. Jedes Element als auch Attribut wird als Knoten verstanden. Knoten sind Objekte, die einen Teil des Dokumentes repräsentieren. Die Schnittstelle ist plattform- und programmiersprachenunabhängig und erlaubt standardisierte Zugriffe auf die Struktur und das Layout eines Dokumentes. Die DOM-Schnittstelle ist in die gesamte Objektschnittstelle eines Browsers nahtlos integriert.

Erstellen wir also so eine HTML-Datei. Geben Sie den nachfolgenden Quelltext in einem Editor ein und nennen Sie die Datei *kap 2_2.html*.

```
<html><body><script>alert(1 + 1);</script></body></html>
```

In der Webseite finden Sie einen vereinfachten internen Skript-Container. Die eigentliche JavaScript-Anweisung steht darin. Wenn Sie die Datei mit einem Doppelklick oder den lokalen Webserver aufrufen, sollte der Browser ein kleines Dialogfenster mit dem Text anzeigen, der das Ergebnis der Berechnung darstellt (Abb. 2.5).