Vorwort

Nahezu jede große Website – egal, ob das Backend in Java, PHP, Ruby oder Python programmiert wurde – verwendet im Browser JavaScript als Sprache. Selbst die Konkurrenz zu JavaScript im Browser – Adobe Flash – wird in ActionScript, einem Dialekt von JavaScript, programmiert. Softwareartefakte, die in JavaScript erstellt wurden, nehmen einen immer größeren Raum im Entwickleralltag ein.

An wen richtet sich dieses Buch?

Der Leser dieses Buchs lernt JavaScript-Entwicklung als professionelle Softwareentwicklung kennen. Teils muss er als Backend-Entwickler JavaScript-Code überarbeiten, der z. B. von einer Webagentur erstellt wurde, teils muss er auch selbst JavaScript-Artefakte erzeugen oder ein ganzes Framework oder eine Library in JavaScript schreiben oder pflegen. Nach der Lektüre des Buchs beherrscht er die Grundlagen der Sprache im Webbrowser. Zudem ermöglicht das Wissen dieses Buchs, auch die Anwendungsmöglichkeiten von JavaScript auf dem Server einzuschätzen und zu bewerten.

Dieses Buch richtet sich daher in erster Linie an Enterprise-Enwickler (meist mit einem Java-EE-Hintergrund), die sich für JavaScript interessieren, sich schnell darin einarbeiten wollen, aber nicht mit den gängigen Büchern für Webentwickler zufrieden sind. Ziel des Buchs ist es, aus einem Backend-Entwickler einen Full-Stack-Entwickler zu machen, der JavaScript seinem Werkzeugkasten hinzugefügt hat.

Wer neu in der Programmierung ist und JavaScript als erste Programmiersprache erlernen möchte, dem empfehle ich Marijn Haverbekes »Eloquent JavaScript« (»Die Kunst der JavaScript-Programmierung«). Wer JavaScript bereits kennt, aber einen fundierten Überblick über den Kern der Sprache wünscht, der lese lieber Douglas Crockfords »JavaScript: The Good Parts: Working with the Shallow Grain of JavaScript«. Wer eine Referenz zu JavaScript sucht, der ist mit Stefan Kochs »JavaScript: Einführung, Programmierung und Referenz« oder mit David Flanagans »JavaScript. The Definitive Guide« besser beraten.

Wie ist das Buch aufgebaut?

Unbestritten hat JavaScript viele Schwächen. Trotzdem ist JavaScript auch die am meisten missverstandene Programmiersprache. Aus diesem Grund sollte man sich mit den Kernkonzepten von JavaScript – also mit dem, was als Core JavaScript bezeichnet wird – eingehend beschäftigen und sie gut verstehen – unabhängig davon, ob man mit JavaScript im Browser oder auf dem Server entwickeln möchte.

Die Sprachkonzepte von Core JavaScript, dem Sprachkern von JavaScript, werden in den ersten Kapiteln des Buchs knapp, aber möglichst vollständig behandelt. Neuerungen in ES5 (ECMAScript[1] 5th Edition) werden natürlich ebenfalls angesprochen. Dabei wird JavaScript sowohl als schwach typisierte, dynamische und funktionale als auch als objektorientierte Programmiersprache betrachtet. Da JavaScript im Kern keine klassenbasiert-objektorientierte Sprache wie Java, sondern eine prototypische Sprache wie Self ist, werden besonders die Unterschiede herausgestellt. Ein prototypisches Sprachkonzept ist so flexibel, dass es leicht möglich ist, objektorientierte Konzepte durch ein Framework selbst zu implementieren.

Nachdem die Grundlagen der Sprache behandelt wurden und dem Entwickler die unterschiedlichen Paradigmen der Sprache bekannt sind, wird die Verwendung von JavaScript dann sowohl auf dem Server als auch im Browser beschrieben. Dabei werden gängige Projekte wie Node.js auf dem Server und die Bibliothek jQuery im Browser verwendet.

JavaScript ist eine nur schwach typisierte Sprache. Es gibt keinen Compiler, der einen selbst vor technischem Unfug warnt. Gerade in größeren Projekten und bei der Erstellung von Frameworks ist daher eine testgetriebene Entwicklung wichtig und eine möglichst große Testabdeckung wünschenswert. Darum wird testgetriebene Entwicklung in JavaScript beschrieben. Abgerundet wird das Buch durch einen Ausflug in die Build-Automatisierung und einen umfangeichen Anhang.

Codebeispiele sind in der Regel möglichst kurz gehalten. Der Leser benötigt also keine Entwicklungsumgebung, um sie nachzuvollziehen. Daher eignet sich dieses Buch auch als U-Bahn-Lektüre.

Wie liest man dieses Buch?

Dieses Buch kann von vorne bis hinten, aber auch selektiv gelesen werden. Wer einen Workshop erwartet, der sollte das Buch komplett lesen und versuchen, die Beispiele nachzuvollziehen. Wer allerdings nur einige Bereiche kennenlernen oder sein Wissen in Teilgebieten vertiefen möchte, dem werden nun die einzelnen Kapitel vorgestellt:

---

1. JavaScript wird unter dem Namen ECMAScript standardisiert, denn der Name »JavaScript« ist ein eingetragenes Warenzeichen von Sun Microsystems bzw. Oracle.

Kapitel 1 – Die Geschichte von JavaScript

Zum Verständnis der Sprache und der unterschiedlichen Laufzeitumgebungen ist es hilfreich, zumindest einen kurzen Überblick über die Entwicklung der Sprache zu kennen – von den Anfängen bei Netscape und den Browserkriegen über das Web 2.0 bis hin zu modernen Anwendungen im und außerhalb des Browsers.

Kapitel 2 – JavaScript-Laufzeitumgebungen

JavaScript benötigt eine Laufzeitumgebung, in der es ausgeführt wird. Die am häufigsten verwendete Laufzeitumgebung ist sicherlich der Webbrowser. Aber JavaScript lässt sich auch »headless«, d.h. außerhalb des Browsers, ausführen. Um die Beispiele der folgenden Kapitel möglichst effektiv (d.h. ohne einen Browser oder eine IDE) nachvollziehen zu können, werden verschiedene Laufzeitumgebungen aufgesetzt, um in diesen Umgebungen mit der Sprache zu experimentieren.

Kapitel 3 – Core-JavaScript

JavaScript ähnelt im Kern gängigen Sprachen wie C oder Java. Allerdings gibt es auch Unterschiede zu diesen Sprachen. Daher werden in diesem Kapitel die wichtigsten Kernfeatures der Sprache vorgestellt: von Anweisungen und Operatoren über Kontrollstrukturen bis hin zu Werten und Literalen. Jeder, der sich für die Sprache interessiert, sollte dieses Kapitel lesen, denn die Sprache birgt doch einige Überraschungen in sich.

Kapitel 4 – JavaScript als funktionale Programmiersprache

JavaScript ist eine Multi-Paradigmen-Sprache. Unterschiedliche Konzepte der funktionalen Programmierung, der prototypischen Programmierung und der objektorientierten Programmierung lassen sich in JavaScript mischen. In diesem Kapitel wird auf die funktionalen Aspekte von JavaScript eingegangen. Neben den Funktionen an sich werden auch Entwurfsmuster beschrieben, die in einer funktionalen JavaScript-Entwicklung Anwendung finden. Dieses Kapitel ist vor allem für Entwickler interessant, die bisher nur wenig in JavaScript programmiert haben. Zudem ist es Grundlage für die folgenden Kapitel.

Kapitel 5 – JavaScript als prototypische Programmiersprache

JavaScript ist durchaus eine objektorientierte Programmiersprache. Prototypisch bezeichnet die Art von Objektorientierung, die JavaScript verwendet. Allerdings kennt JavaScript keine Klassen, sondern lediglich Objekte und deren Prototypen. In diesem Kapitel wird beschrieben, wie sich Objekte erzeugen und verändern lassen. Dieses Kapitel ist Pflichtlektüre für alle Entwickler, die in einer klassisch-objektorientierten Sprache zu Hause sind, denn die Unterschiede von JavaScript zu verbreiteten Sprachen wie Java sind doch zu groß, um sie ignorieren zu können.

Kapitel 6 – JavaScript als objektorientierte Programmiersprache

Obwohl das prototypische Paradigma von JavaScript ein sehr mächtiges Paradigma ist, fühlen sich Entwickler, die aus einer klassisch-objektorientierten Sprache kommen, mit einer prototypischen Sprache oft unwohl. Über verschiedene Entwurfsmuster zur Objekterzeugung wird eine eigene klassenbasierte Vererbung realisert. Diese eigene Implementierung wird gängigen Implementierungen gegenübergestellt. Sowohl die eigene als auch gängige Implementierungen fühlen sich in JavaScript jedoch wie ein Fremdkörper an. Daher werden die prototypische Vererbung und daraus resultierende Entwurfsmuster vorgestellt. ECMAScript 5th Edition ist die aktuelle Version von JavaScript. Sie wird in vielen Browsern zwar noch nicht verwendet, bietet aber viele neue Features, um mit Objekten zu arbeiten. Diese neuen Features werden kurz vorgestellt, auch wenn sich diese nur in Ausnahmefällen (wie auf dem Server) nutzen lassen. Dieses Kapitel ist für Entwickler interessant, die sich nicht nur für die praktische Anwendung, sondern auch für Sprachkonzepte interessieren.

Kapitel 7 – Eingebaute Objekte

Viel wurde bisher über Objekte und deren Erzeugung geschrieben. JavaScript hat selbst auch ein paar eigene Objekte – die API der Sprache: Dies sind Boolean, Number, String, Array, Object, Function, RegExp, Date und einige Objekte zur Fehlerbehandlung. Dieses Kapitel sollten Sie zumindest überfliegen, um sich einen Eindruck von den wenigen Objekten, die JavaScript mitbringt, zu verschaffen. Es gibt nur wenige Sprachen, deren API ein Entwickler komplett beherrschen kann. Daher lohnt es sich, dieses Kapitel nicht nur zu überfliegen, sondern auch zusammen mit dem Anhang komplett zu lesen, um sich ein Wissen über JavaScript möglichst vollständig anzulegen.

Kapitel 8 – Entwurfsmuster

Gängige Entwurfsmuster kommen aus klassenbasierten[2] Sprachen wie C++ oder Java. Dieses Kapitel zeigt, wie sich diese Entwurfsmuster auch in JavaScript sinnvoll anwenden lassen. Neben klassischen Entwurfsmustern der sogenannten Gang of Four (GoF[3]) werden auch Entwurfsmuster vorgestellt, die in JavaScript häufiger als klassische GoF-Patterns anzutreffen sind. Entwurfsmuster sind bei Enterprise-Entwicklern sehr beliebt, darum darf dieses Kapitel in keinem Buch für Enterprise-Entwickler fehlen. Wer sein bisheriges Wissen über Entwurfsmuster auf JavaScript übertragen möchte, dem sei dieses Kapitel ans Herz gelegt.

---

2. In diesem Buch wird unterschieden zwischen objektorientierten und klassenbasierten objektorientierten Sprachen.
3. Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides.

Kapitel 9 – JavaScript im Webbrowser – Teil 1

Zwar ist dies kein klassisches Buch über Webentwicklung, doch darf ein Kapitel über den Webbrowser in keinem JavaScript-Buch fehlen. Der Browser ist die Laufzeitumgebung, für die JavaScript entwickelt wurde. Der meiste Code, der in JavaScript geschrieben wurde und wohl auch in Zukunft geschrieben wird, ist Code, der im Webbrowser läuft. In diesem Kapitel werden daher die wichtigsten Grundlagen vermittelt, die ein Entwickler über JavaScript im Browser wissen muss. Für Entwickler, die bereits die Grundlagen der Webentwicklung kennen, ist dieses Kapitel ein Repetitorium. Für Entwickler, die bisher im Backend unterwegs waren, ist dieses Kapitel aber sicher ein Pflichtkapitel.

Kapitel 10 – JavaScript auf dem Server mit Node

JavaScript wird in der Regel im Browser verwendet. Allerdings erlebt es zurzeit auch auf dem Server eine Renaissance. Als serverseitige Umgebung wird Node und das Framework »Express« vorgestellt. Dabei wird auch auf Modularisierung und Packaging eingegangen, denn Node motiviert das Verwenden von CommonJS-Modulen. Zwar werden die meisten Enterprise-Entwicklern in ihrem beruflichen Umfeld nur selten mit Node in Berührung kommen, doch ist dieses Kapitel trotzdem interessant, denn die Besonderheiten der Entwicklung mit JavaScript werden hier auf den Server übertragen. Für einen Full-Stack-Entwickler, der sowohl im Frontend als auch im Backend unterwegs ist, klingt es verführerisch, nur noch in einer einzigen Sprache zu entwickeln: JavaScript. Dies klingt umso verlockender, wenn der Trend von Thin Clients und intelligenten Servern hin zu Smart Clients und Thin Servern sich fortsetzen sollte. Eine Investition in Wissen um JavaScript auf dem Server ist also eine Investition in die Zukunft, eine Wette auf zukünftige Paradigmen der modernen Webentwicklung.

Kapitel 11 – JavaScript im Webbrowser – Teil 2

Nachdem eine erste serverseitige Anwendung erstellt wurde, steigt dieses Buch nun tiefer in die Webentwicklung im Webbrowser ein. Es werden die Grundlagen der Formularverarbeitung angesprochen und jQuery bzw. jQuery-UI werden kurz vorgestellt. Selbst wer das Kapitel über Node übersprungen hat, sollte an dieser Stelle wieder einsteigen. Die besprochenden Libraries werden zwar nicht tief gehend behandelt, allerdings sollte dieses Kapitel einen Überblick vermitteln.

Kapitel 12 – Build-Automatisierung

JavaScript ist keine kompilierende Sprache. Trotzdem macht Build-Automatisierung auch für JavaScript Sinn, denn Build-Automatisierung umfasst mehr als das Kompilieren von Quellcode in Binärcode, nämlich das Skripten von Tätigkeiten, die ein Entwickler während des Softwareentwicklungsprozesses täglich durchführt. Zu diesen Tätigkeiten, die sich leicht automatisieren lassen, gehört das Kompilieren des Quellcodes, das Paketieren (Packaging) des Codes bzw. des Binärcodes, das Durchführen von Unittests und Integrationstests, das Überprüfen von Coding-Conventions und anderen Qualitätsanforderungen, das Deployment auf Test- und Livesysteme und die Generierung von Dokumentationen und Release-Notes. Diese Tätigkeiten lassen sich auch über ein Shell-Skript automatisieren. Strukturierter lässt sich dies aber über ein Build-System erreichen. In diesem Kapitel werden zunächst die einzelnen Tätigkeiten beschrieben. Anschließend wird gezeigt, wie sich diese Tätigkeiten durch gängige Build-Systeme automatisieren lassen. Build-Automatisierung wird hier aus der Perspektive eines Enterprise-Java-Entwicklers beschrieben. Leser, die aus einem anderen Umfeld kommen, können sich durch dieses Kapitel inspirieren lassen, ähnliche Lösungen mit anderen Technologien zu realiseren.

Kapitel 13 – Testen

Da JavaScript-Anwendungen immer komplexer werden, kommt auch dem Testen eine immer größere Bedeutung zu. Gerade in größeren Projekten und bei der Erstellung von Frameworks ist eine testgetriebene Entwicklung wichtig und eine möglichst große Testabdeckung wünschenswert. Der Trend automatisierter Tests entwickelte sich dabei von der klassischen testgetriebenen Entwicklung (Test Driven Development, TDD) hin zum Behaviour Driven Development (BDD). Verteilte Tests in unterschiedlichen Webbrowsern lassen sich mit modernen Tools wie JsTestDriver automatisieren. Der Leser bekommt einen Überblick über das Testing in JavaScript. Da automatisierte Tests in keinem Werkzeugkasten fehlen sollten, ist dieses Kapitel eine Pflichtlektüre für professionelle Entwickler.

Referenz, Nachwort, Literaturliste und Index

Abgeschlossen wird das Buch von einer Referenz und einem Index zum Nachschlagen. Eine Literaturliste soll zum Weiterlesen anregen.

Danksagung

An dieser Stelle möchte ich zuerst meiner Frau Katharina und meinen Kindern Emma und Ada danken, dass sie mir die Zeit gaben, dieses Buch zu schreiben.

Außerdem möchte ich meinen Lektoren René Schönfeldt und Gabriel Neumann für ihre konstruktiven Vorschläge und allen Beteiligten im dpunkt.verlag danken. Ohne sie wäre dieses Buch nie entstanden. Ursula Zimpfer danke ich für ihre Korrekturen.

Ein großes Dankeschön geht auch an meine Korrekturleser Michael Gustav Simon, Enno Thieleke, Dr. Markus Liebelt und Sören Lünsdorf. Das Gleiche gilt für Benjamin Schmid, Golo Roden und meine anonymen Gutachter. Ihr teils sehr frühes Feedback war unbezahlbar!

Dieses Buch wäre auch ohne meine literarischen Vorbilder Stefan Koch, David Flanagan, Douglas Crockford, John Resig, Dustin Diaz, Stoyan Stefanov, Christian Johansen, Marjin Haverbeke und die zahllosen Mitwirkenden am Mozilla Developer Network nie entstanden.

Sollte ich jemanden vergessen haben oder es versäumt haben, einen Artikel, aus dem ich geschöpft habe, zu erwähnen, dann lassen Sie es mich wissen, sodass ich dies nachtragen kann.