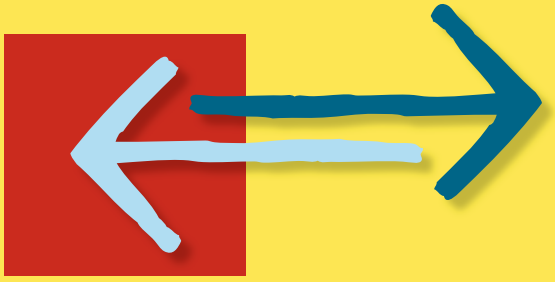
peter leo **GORSKI**
luigi **LO IACONO**
hoai viet **NGUYEN**



WebSockets

Moderne **HTML5-Echtzeit-anwendungen** entwickeln

HANSER

Gorski/Lo Iacono/Nguyen

WebSockets

Peter Leo Gorski
Luigi Lo Iacono
Hoai Viet Nguyen

WebSockets

Moderne HTML5-Echtzeitanwendungen entwickeln

HANSER

Die Autoren:

Peter Leo Gorski, Köln
Prof. Dr.-Ing. Luigi Lo Iacono, Bonn
Hoai Viet Nguyen, Köln

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.

Lektorat: Sieglinde Schärl
Herstellung: Irene Weilhart
Copy editing: Sandra Gottmann, Münster
Layout: Peter Leo Gorski mit LaTeX
Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München
Umschlagrealisation: Stephan Rönigk
Druck und Bindung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN: 978-3-446-44371-6
E-Book-ISBN: 978-3-446-44438-6

**»Der Weltuntergang steht bevor,
aber nicht so, wie Sie denken.
Dieser Krieg jagt nicht alles in die Luft,
sondern schaltet alles ab.«**

Tom DeMarco
Als auf der Welt das Licht ausging

ca. 560 Seiten. Hardcover
ca. € 19,99 [D] / € 20,60 [A] / sFr 28,90
ISBN 978-3-446-43960-3
Erscheint im November 2014

Hier klicken zur Leseprobe

**Sie möchten mehr über Tom DeMarco und seine Bücher erfahren.
Einfach reinklicken unter www.hanser-fachbuch.de/special/demarco**

Inhalt

Vorwort

Netzwerke umgeben uns quasi überall. Viele Dinge des täglichen Geschäfts- und Privatlebens sind ohne Kommunikation über ein digitales Datennetz nicht mehr vorstellbar. Zur Lingua franca der Protokolle der Anwendungsschicht hat sich in den letzten Jahren das HTTP gemausert. Angeschoben durch den immensen Erfolg des Web bedienen sich heute eine Vielzahl anderer Anwendungen dem HTTP. Darunter z.B. die Lokalisierung und Ansteuerung von Geräten im Hausnetz via UPnP, die Speicherung von Daten in der Cloud alla Dropbox und vermehrt auch das TV. Die Gründe hierfür sind vielfältig. Der simple Aufbau von HTTP und die zahlreich verfügbaren Implementierungen spielen dabei sicherlich eine Hauptrolle. In den Nebenrollen finden sich Darsteller wie die durchgängige Firewall-Freundlichkeit und neuerlich auch die ubiquitäre Verbreitung von HTTP in Geräten jeden Couleurs. Ist man mit seinen Anwendungen und Diensten an Reichweite interessiert, führt derzeit kein Weg an HTTP vorbei.

Neben all dieser Euphorie ist es ratsam, alle Eigenschaften des neuen Gefährten zu kennen, da sich daraus auch seine Grenzen ableiten lassen. An der Zustandslosigkeit von HTTP lässt sich dies schön verdeutlichen. Erlaubt es auf der einen Seite ein vereinfachtes Caching und globale Skalierbarkeit durch Content Distribution Networks (CDN), erschwert es die Implementierung von Warenkörben in E-Commerce Anwendungen. Letzteres hat man im Laufe der Zeit gut in den Griff bekommen. Für andere Nachteile haben sich dagegen noch keine Patentrezepte etabliert. Hierzu zählt z.B. die unidirektional ausgelegte Kommunikation. Gemäß der Protokollspezifikation meldet sich immer der Client mit einer Anfrage an den Server, der auf diese mit einer entsprechenden Antwort reagiert. Dass sich der Server selbstinitiativ bei einem Client meldet, ist nicht vorgesehen. Dies liegt auch darin begründet, dass die Verbindung zwischen Client und Server nicht dauerhaft, sondern nur für die Zeit des Austausches von Anfragen und ihrer Antworten besteht. In modernen Anwendungen ist das Pushen von Nachrichten oder Daten vom Server zum Client eine häufige Anforderung, die es technisch umzusetzen gilt. Genau an dieser Stelle setzt die WebSocket-Technologie an.

Dem Wildwuchs an Ansätzen, die zum Aufbrechen der Kommunikationseinbahnstraße von HTTP vorgeschlagen wurden und uneinheitlich zur Verwendung gekommen sind, wird mit WebSockets eine standardisierte Lösung entgegengestellt. Der Hunger nach einem derartigen Standard war groß. Dies lässt sich daran ablesen, dass die Unterstützung von WebSockets durch alle einschlägigen Industriegrößen bekannt gegeben wurde, als sich die Standardisierung noch in den Kinderschuhen befand. Bereits heute lässt sich kein Webbrowser vermissen, der nicht WebSockets mit an Bord hat. Obwohl die Standardisierung

zum aktuellen Zeitpunkt noch nicht abgeschlossen ist, kann davon ausgegangen werden, dass es sich hierbei in nicht allzu langer Zeit um einen etablierten Standard handeln wird.

Das vorliegende Buch stellt diesen neuen Grundpfeiler für moderne Anwendungen auf Grundlage von HTTP vor. Dabei geht es auf alle Aspekte der WebSocket-Technologie in angemessener Tiefe ein. Meine vorweggenommene Motivation wird zu Beginn des Buches, angereichert mit dem notwendigen Background, aufgegriffen und somit auf das eigentliche Thema hingeführt. Dann stehen die WebSockets im Rampenlicht. Mit dem unter den Fittichen der IETF stehendem WebSocket-Protokoll geht es los. Hier wird Bit genau beleuchtet, wie sich die verschiedenen Frames zwischen Client und Server bewegen. Das zeigt unter anderem, wie viel effektiver die Übertragung mittels WebSockets im Vergleich zu HTTP ist, wenn der Overhead der HTTP-Header wegfällt. Im Anschluss lernt man die von der W3C verantwortete JavaScript API kennen, mit der sich die Clientseite einer verteilten WebSocket-Anwendung in einem Browser implementieren lässt. Das ist allerdings noch nicht alles, was für ein ganzheitliches Projekt benötigt wird. Die Serverseite fehlt und hier sieht das Bild nicht ganz so einheitlich aus. Fehlende Standards machen es erforderlich, sich die vom ausgewählten Framework entsprechend bereitgestellten APIs anzueignen. Diesem Umstand Rechnung tragend, werden in dem Buch verschiedene serverseitige Frameworks behandelt, die für Java und JavaScript bereitstehen. Abgerundet wird das Ganze durch zahlreiche Beispiele, die auch größere Zusammenhänge nachvollziehbar und verständlich machen.

Mir hat die Lektüre das notwendige Know-how und das Verständnis in die neuen Möglichkeiten vermittelt, mit dem ich nun spannenden neuen Projekten unter Verwendung von WebSockets entgegenblicke. Ich wünsche Ihnen ebenso viel Erkenntnisgewinn beim Lesen, Ihr

Alexander Leschinsky Köln, im November 2014
Geschäftsführer der G&L Systemhaus GmbH

Danksagung

Die Seiten eines Buches füllen sich maßgeblich durch die Federn der Autoren. Einfluss auf die Formulierungen und Aufbereitungen der produzierten Inhalte haben allerdings eine Vielzahl von weiteren Personen. Auch an diesem Werk haben viele gute Geister konstruktiv mitgewirkt, denen wir hiermit ein herzliches Dankeschön aussprechen möchten (Nennungen in alphabetischer Reichenfolge): Aline Jaritz, Carsten Möhrke, Daniel Torkian, Stephan Mattescheck und Sven Wagner

Ein Dankeschön geht außerdem an die Fachhochschule Köln, an den Carl Hanser Verlag, an unsere Lektorin Sieglinde Schärl, für das reibungslose Management, an Irene Weilhart, für die Hilfe beim Buchsatz und selbstverständlich an Sandra Gottmann, die eine Fülle von Fehlern entdeckt hat, die uns beim Schreiben durch die Finger gerutscht sind.

Möchte man eine Webanwendungen oder gar einen Webservice entwickeln, kommt man unweigerlich mit einer Vielzahl von Web-Technologien, Frameworks und anderen Artefakten in Kontakt. Um dem Leser die WebSocket-Technologie näherzubringen, nehmen auch wir daher Bezug auf viele Zeilen Code und Software, die wir nicht immer selbst entwickelt haben. Deshalb möchten wir den Organisationen, Unternehmen und Entwicklern, einen besonderen Dank aussprechen, insbesondere denen, die Ihre Entwicklungen jedem zur freien Verfügung stellen, um etwas daraus lernen und um gemeinsam Technologien voranbringen zu können (erneut in alphabetischer Reihenfolge): Agendaless Consulting and Contributors (supervisor), Alexis Deveria (caniuse.com), Amazon.com Inc., Apple Inc., Automattic (Socket.io), Brian McKelvey (WebSocket-Node), Canonical Foundation, Eclipse Foundation, Eric Lawrence (Fiddler), Firebug Working Group, GNU/Linux, Google Inc., IETF, ISO, Joyent Inc. (Node.js), Kaazing Corporation (WebSocket.org), LearnBoost (WebSocket.io), Microsoft Corporation, Mozilla Corporation, Opera Software, Oracle Corporation, Patrick Wied (heatmap.js), Play Framework, strongloop (Express.js), Tim Fox & VMWare & Red Hat (vert.x), Twitter Inc. (Bootstrap), W3C, Wireshark-Community und alle, die noch im Buch erwähnt werden und die wir versehentlich an dieser Stelle vergessen haben.

Ein Teil der Grafiken, die wir für dieses Buch angefertigt haben, sind mithilfe von lizenzfreien Cliparts der Internetseite *www.clker.com/* angefertigt worden. Auch dafür sind wir sehr dankbar.

Die Fertigstellung eines Buchs benötigt vor allem viel Zeit, Geduld, Fleiß und Ausdauer. Auch das wäre für uns ohne die Unterstützung der Menschen, die uns am nächsten stehen, nicht möglich gewesen. Vielen Dank an Anh Trung, Antonio, Ba Tho, Barbara, Em Yen, Fabio, Giuliana, Inge, Marco, Me Chau, Peter und Tati.

An letzter Stelle möchten wir natürlich auch Ihnen für den Kauf dieses Buches und Ihr Interesse an den WebSockets danken!

1 Einleitung

Einer der technischen Hauptakteure im Web ist HTTP, das *Hypertext Transfer Protocol* [FGM+99], und das ist so, weil HTTP den genauen Kommunikationsablauf zwischen Webbrowser und Webserver festlegt. Genau wie ein Protokoll für eine königliche Gala zu Hofe alle Regeln in Bezug auf Kleidung, Etikette und Umgangsformen zusammenfasst, so legt ein Kommunikationsprotokoll alle Regeln zum Nachrichtenaustausch fest. Anders als bei Hofe sind das bei einem Kommunikationsprotokoll aber vielmehr Regeln in Bezug auf den Aufbau, das Format und die Codierung der ausgetauschten Nachrichten. Außerdem muss in einem Protokoll festgelegt sein, wer die Kommunikation beginnt und wie diese dann Nachricht für Nachricht bis zum Kommunikationsende abläuft. HTTP verwendet hier einen einfachen Request-Response-Nachrichtenaustausch. Dieser wird immer vom Webclient durch eine Anfrage (engl. *Request*) initiiert und entsprechend vom Webserver mit einer Antwort (engl. *Response*) beantwortet. Sobald Sie also z. B. eine Webadresse – die URL – in die Adresszeile eines Webbrowsers eingeben oder einen Bookmark bzw. Link anklicken, setzen Sie damit eine Anfrage an einen Server ab, der diese dann bearbeitet und die Antwort daraufhin zurücksendet (siehe Bild 1.1).

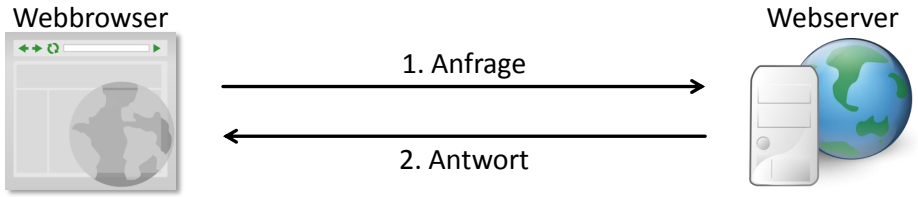


Bild 1.1 Grundlegender HTTP-Nachrichtenaustausch

In diesem einfachen Request-Response-Nachrichtenaustausch liegen viele Gründe für den großen Erfolg von HTTP und nicht zuletzt des Webs. Dazu zählen u. a. die guten Skalierungseigenschaften, denen wir die Größe des Webs verdanken. Wie Sie sich aber sicher jetzt schon denken, bringt das Nachrichtenpaar nicht nur Vorteile mit sich. Und tatsächlich ist die Protokollregel, dass der Request immer vom Client ausgehen muss, eine Zwangsjacke, die bestimmte Bewegungsfreiheiten in Form von Kommunikationsmustern nicht zulässt. Der Teil einer Webanwendung, der auf dem Server ausgeführt wird, kann sich nämlich nicht von sich aus an einen Client wenden. Diese Eigenschaft des Protokolls schränkt folglich Anwendungsfälle ein, in denen die Serverseite Statusänderungen an die betroffenen Clients weiterreichen können muss. Typische Beispiele hierfür sind Chatanwendun-