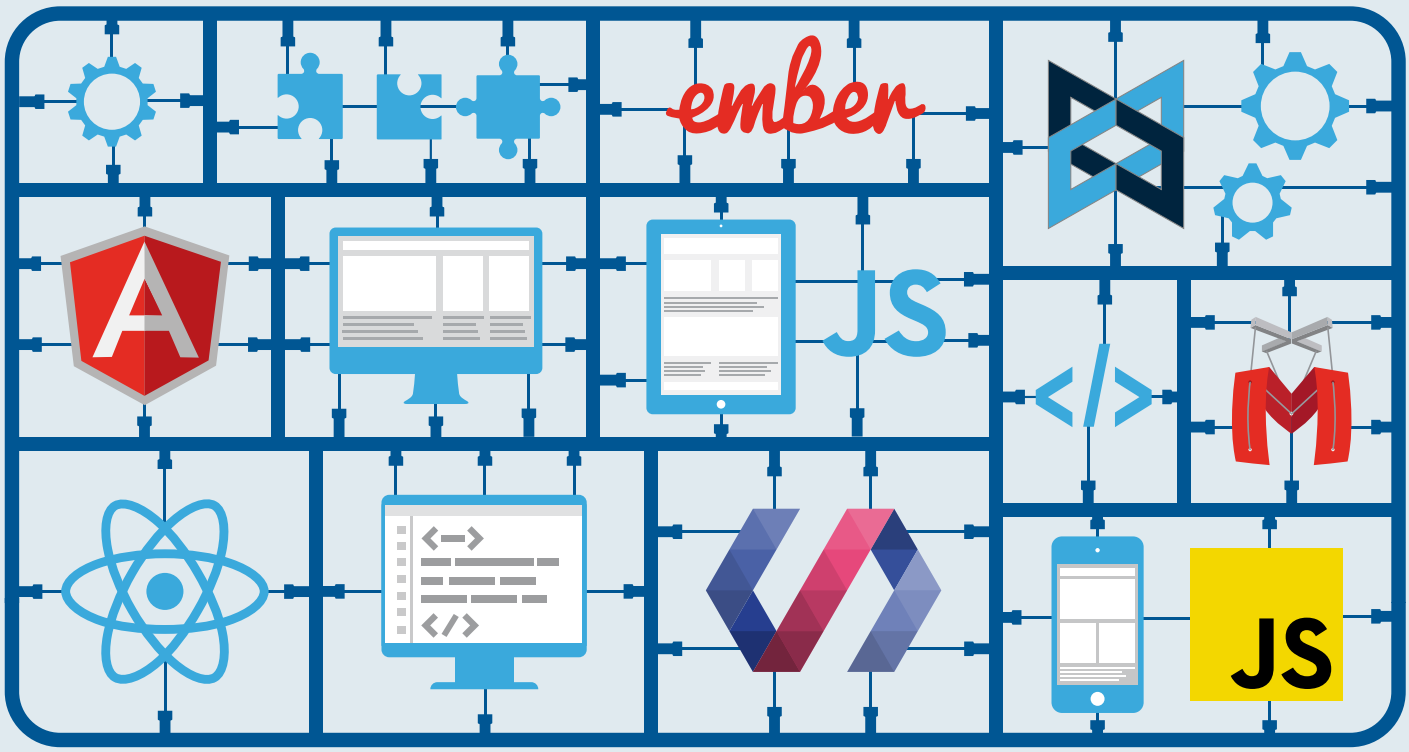
web & mobile DEVELOPER

webundmobile.de

JavaScript-Frameworks

Schneller und effektiver programmieren mit Frameworks S. 14



Auf der Heft-CD

Eine Sammlung interessanter JavaScript-Frameworks sowie eine Auswahl leistungsfähiger Tools für Entwickler S. 50

INFO-Programm gemäß § 14 JuSchG

Ausgabe 6/16

Deutschland 14,95 EUR

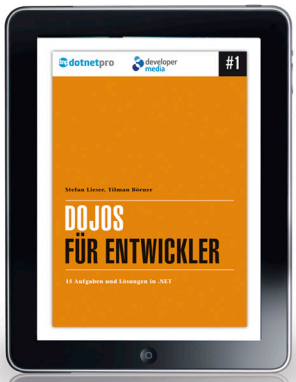
CH: 29,90 CHF
A, B, NL, L: 16,45 EUR

4 198255 014951 06

developer media Shop

Upgrades für Ihr Entwickler-Know-How

Ab sofort in unseren Shops erhältlich!



https://shop.dotnetpro.de

https://shop.webundmobile.de

Frameworks

Bei ihrer Arbeit können sich Entwickler auf leistungsstarke JavaScript-Frameworks stützen.

Erfahrene Entwickler möchten vorzugsweise auf ausgereifte, bewährte Technologien setzen, aber gleichzeitig neue Trends wie die bevorstehende groß angelegte Umstellung auf ECMAScript 6 (ES6) im Auge behalten, um sich darauf rechtzeitig vorzubereiten. Bei JavaScript ist die Aufgabe, den Überblick zu behalten, angesichts der zahllosen Frameworks und verschiedener Ansätze gar nicht so einfach. In unserem Schwerpunktthema erläutern ab Seite 14 unsere Autoren Anna Kobylinska und Filipe Martins am Beispiel einiger ausgewählter Frameworks, wohin der Trend geht.

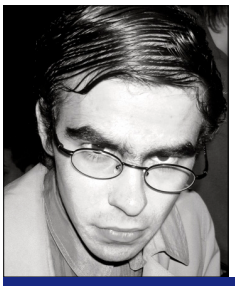
Node.js bringt zwar von Haus aus keinen Process Manager mit, das Modul PM2 schließt jedoch diese Lücke. Dabei handelt es sich um einen Process Manager für Node.js, über den sich Anwendungen starten, stoppen und neustarten lassen. Darüber hinaus verfügt PM2 über eine Reihe weiterer interessanter Features, die Philip Ackermann ab Seite 38 im Detail erläutert.

»Entwickler möchten vorzugsweise auf ausgereifte, bewährte Technologien setzen.«

Mit dem Release von iOS 8 schickte sich die neue Klasse UIAlertController an, die bisher für Alerts verwendeten Klassen UIAlertView und UIActionSheet abzulösen. Der neue UIAlertController dient als einheitliche Schnittstelle für Nutzermeldungen aller Art und soll den Umgang mit Alerts und Action-Sheets deutlich erleichtern. Thomas Sillmann erklärt ab Seite 52, wie man die eigenen Applikationen auf Apples neue Universalklasse umstellt.

Grafische Editoren zur Erstellung von Programmcode haben vor allem durch Scratch eine größere Verbreitung gefunden. Scratch kommt bevorzugt im Ausbildungsbereich zum Einsatz, um Kindern die Programmierung anhand von Spielen zu vermitteln. Eine Alternative zu Scratch stellt Blockly dar. Wobei der große Unterschied darin besteht, dass Scratch eine Anwendung ist und Blockly eine Programmbibliothek zur Erstellung von grafischen Editoren auf der Basis von Blöcken. Alles zum Einsatz von Blockly erfahren Sie von Dr. Markus Stäuble ab Seite 124.

Ihr Max Bold
chefredakteur@maxbold.de



Tam Hanna

erklärt die Spieleprogrammierung unter OpenGL (S. 69)

Daniel Basler

erläutert die Docker-Erweiterungen von Visual Studio (S. 106)

Katharina Sckommodau

sagt, was Formen auf der Fläche bewirken (S. 120)

INHALT

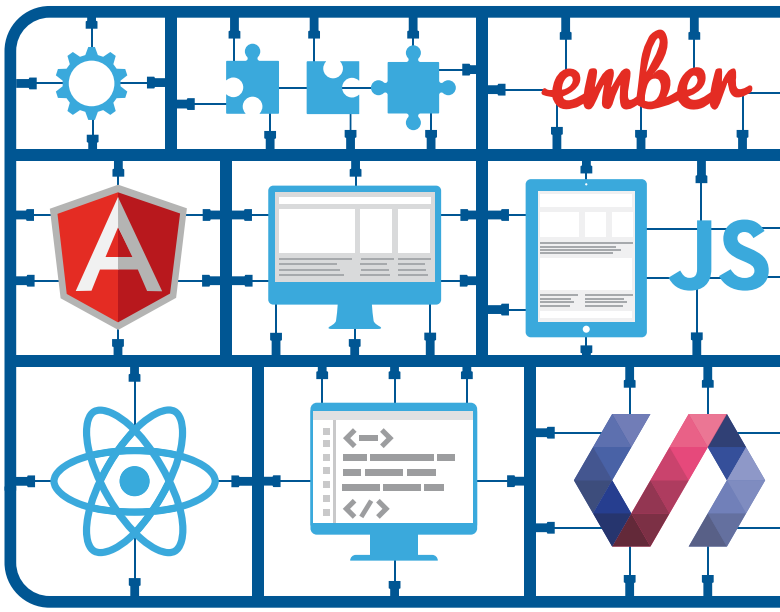
Update

Feature

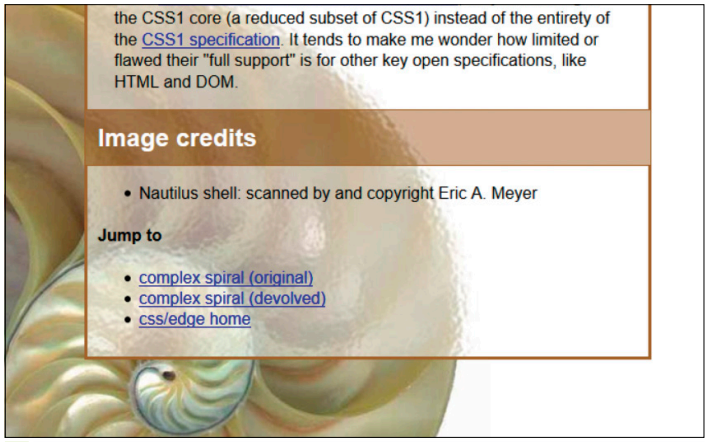
HTML, CSS & JavaScript

Mobile Development

Backend

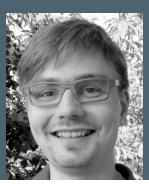


Die aktuellen JavaScript-Frameworks fallen in eine von zwei Kategorien: Innovatoren oder Traditionalisten **S. 14**

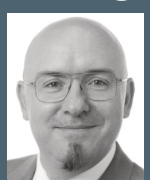


Webseiten sind scrollbar: Auf das Scrollverhalten kann man mit CSS3 auf verschiedene Arten Einfluss nehmen **S. 28**

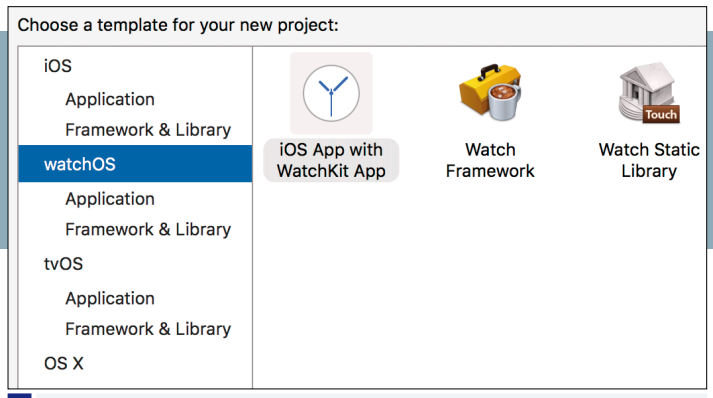
Experten in dieser Ausgabe



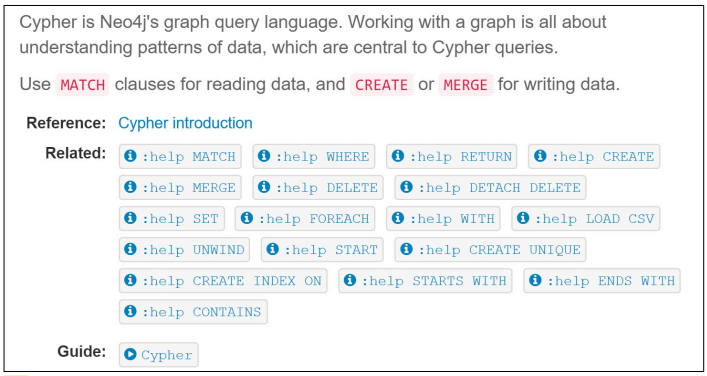
Thomas Sillmann erklärt den UIAlert-Controller in der iOS-Praxis. **52**



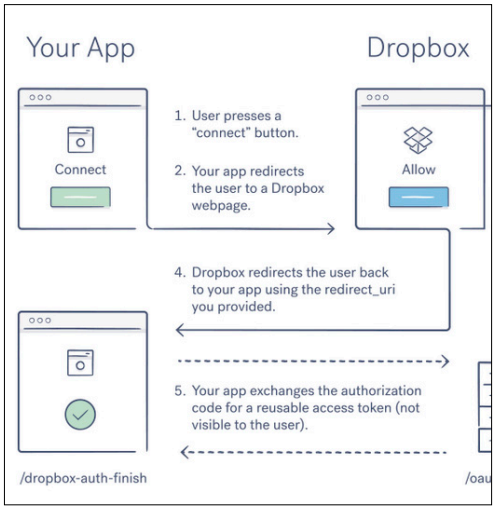
Patrick Lobacher erläutert das Angebot Deutsche Bahn Open Data. **102**



Im Vergleich zu Objective-C besitzt Swift zahlreiche Stärken: Es ist flexibler, geradliniger und unkomplizierter **S. 82**



Die Cypher Query Language stellt nicht nur eine reine Abfragesprache für die Daten der Graph-Datenbank dar **S. 88**



Der beliebte Dienst Dropbox bietet für alle gängigen Plattformen ausgereifte Clients an. Oft ist aber auch die Integration in Webanwendung gefragt. Zu diesem Zweck stellt Dropbox ein API bereit **S. 110**

BeyondDev

Standards

NEWS & TRENDS

AKTUELLE NEWS AUS DER ENTWICKLERSZENE

Fraunhofer IIS

Hackathon am Fraunhofer IIS

Das Fraunhofer-Institut für Integrierte Schaltungen IIS erwartet vom 30. Juni bis zum 2. Juli 2016 bis zu 50

Auf einem Hackathon sollen Multiplayer-Apps entwickelt werden

Software- und Spiele-Entwickler, 3D-Artists, Mediendesigner und qualifizierte Personen aus der Medienbranche, die im Rahmen des Projekts Holodeck 4.0 Multiplayer-Apps für Virtual-Reality-Anwendungen entwickeln werden.

Für die Programmierer wird im Rahmen des Holodeck-4.0-Hackathons auch das RedFIR-Tracking-system des Fraunhofer IIS genutzt.

Über seine präzise Positionsbestimmung erlaubt dieses System Nutzern, sich frei in virtuellen Welten zu bewegen und andere Nutzer, die sich ebenfalls in diesem Raum bewegen, als Avatare mit in die virtuelle Welt einzubinden und ihre Bewegungen wahrzunehmen.

www.holodeck40.de

Media Asset Management

Six launcht Update von SixOMC

Ein besonderer Fokus der Version SixOMC 9.7.5 liegt auf der Benutzerverwaltung und der Unterstützung von Administratoren. Das Highlight sind die neu eingeführten Benutzertemplates: Sie werden für eine Benutzergruppe erstellt, enthalten alle relevanten Einstellungen und können für beliebig viele neu hinzukommende Benutzer verwendet werden.

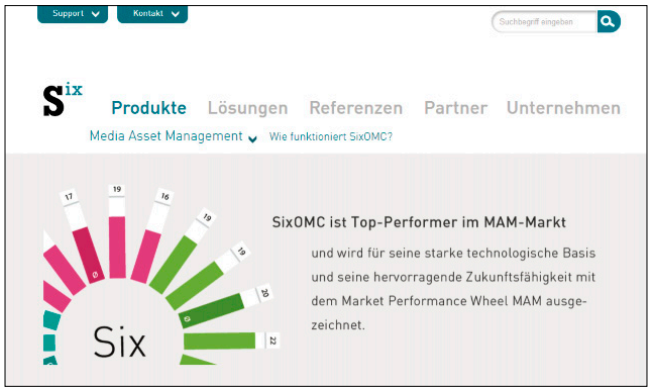
Weitere neue Funktionen sind die 24/7-Registrierung und die Authentifizierung eines Benutzers im Mixed-LDAP-Modus. Die Möglichkeit, Benutzerkonten künftig ohne Änderung des Passworts als der jeweilige Benutzer testen zu können, dürfte Administratoren freuen.

Einen wesentlichen Fortschritt für eine konfliktfreie Kollaboration liefert der Check-out-/Check-in-Prozess im Webclient. Insbesondere die Zusammenarbeit mit externen Dienstleistern wird durch das automatische Sperren von benutzten Dateien während der Bearbeitung unterstützt. Auch zum Thema Metadaten-Konsistenz wartet SixOMC 9.7.5 mit neuen Funktionen auf, wie zum Beispiel der optionalen Verschlagwortung bei Dateiimporten via Upload-Link.

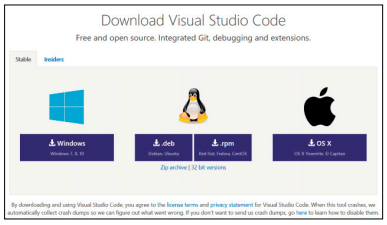
Auch Usability schreibt der Hersteller beim aktuellen Release groß. Im Dashboard sorgen neue Gadgets für mehr Übersichtlichkeit, die Performance wurde gesteigert und über individuelle Suchfilter kommt der Benutzer schnell an die relevanten Informationen.

Julia Hermel, Werkstudentin im Bereich Grafik und begeisterte Nutzerin des FatClients, über ihr Lieblings-Feature: »Die Stapelverarbeitung bei der Bearbeitung von Dateinamen ist sehr hilfreich! Ich habe oft mehrere Bilddateien mit kryptischen Namen und vielen Zahlen im Dateinamen. Für mehr Übersichtlichkeit bei der Weiterverarbeitung kann ich nun mit einem Klick zum Beispiel Zahlen entfernen oder wichtige Infos hinzufügen.«

www.six.de



Die neue Version 9.7.5 des Media-Asset-Management-Systems SixOMC ist da



Vom Open-Source-Editor Visual Studio Code gibt es eine neue Version

Visual Studio Code

Version 1.0 des Open-Source-Editors ist da

Ein Jahr nach der ersten Vorstellung hat Microsoft die Version 1.0 des Editors für Programmierer Visual Studio Code freigegeben. Inzwischen ist der für mehrere Plattformen verfügbare Code-Editor bereits auf rund zwei Millionen Rechnern installiert und er wird – so sagen es die von Microsoft veröffentlichten Zahlen – von mehr als 500.000 Entwicklern auch aktiv genutzt.

Auf dem Weg zur Version 1.0 hat Microsoft mit der Community zusammengearbeitet, um Hunderte von Bugs zu beseitigen, die Stabilität zu verbessern und eine annehmbare Performance zu erzielen.

Ursprünglich nur geplant für Entwickler, die Web-Apps mit JavaScript und TypeScript verfassen, unterstützt VS Code dank der über 1000 von der Community bereitgestellten Erweiterungen inzwischen über 100 Programmiersprachen.

VS Code 1.0 lässt sich nun komplett lokalisieren und wird in neun unterschiedlichen Sprachen ausgeliefert – darunter auch in deutscher Sprache.

https://code.visualstudio.com

Zahl des Monats

Laut einer Bitkom-Umfrage hat **fast jeder zweite Smartphone-Nutzer (47 %)** schon einmal Geld für Apps ausgegeben. Gut ein Drittel **(35 %)** hat dabei für die App an sich bezahlt, also den Kaufpreis beim Herunterladen der Anwendung. **14 %** haben außerdem für kostenpflichtige Angebote bei der Nutzung der App bezahlt (sogenannte In-App-Käufe).

Quelle: www.bitkom.org

WINT.global

Hosting-Pakete mit Website-Builder

WINT.global bietet seinen Kunden jetzt zu allen Hosting-Paketen der Wircon Interservices GmbH auch die Nutzung des Website-Builders Site.pro Premium an. Mit dieser Erweiterung können auch Menschen, die keine praktischen Computer-Erfahrungen haben, eine professionell aussehende Website binnen weniger Stunden einrichten.

WINT.global erweitert seine Hosting-Pakete um einen Website-Builder

»Unser Ziel ist es, unsere Kunden auf dem Weg zu Ihrer eigenen Website optimal zu beraten und zu begleiten. Mit dem Website-Builder Site.pro erweitern wir unser Angebot entsprechend der Bedürfnisse unserer Kunden«, erläutert Peter Vest, Geschäftsführer der Wircon Interservices GmbH. Der Website-Builder ermöglicht die freie Wahl von mehr als 170 Vorlagen. »Wir möchten unseren Kunden die Chance geben, mit wenig Aufwand einen professionellen und individuellen Internetauftritt zu realisieren, der im Gedächtnis bleibt.«

www.wint.global ▶

Facebooks Entwicklerkonferenz F8

Viele Neuerungen angekündigt

Während der F8 waren mehr als 2600 Teilnehmer vor Ort in San Francisco, und Millionen haben die Keynotes und Sessions per Livestream auf Facebook verfolgt. Dabei gab es eine Vielzahl von interessanten Ankündigungen für Entwickler, aber auch Neuerungen für die Milliardenschaft der Facebook-Nutzer.

Facebook hat einige seiner APIs aktualisiert und geöffnet sowie Entwicklern neue Möglichkeiten gegeben, ihre Apps zu verbreiten. Das Account-Kit ermöglicht den Nutzern auf Facebook, sich nur mit ihrer Telefonnummer oder ihrer E-Mail-Adresse in neuen Apps anzumelden, was Entwicklern dabei hilft, Anmeldehürden abzubauen und mehr Nutzer für ihre Apps zu erreichen. Ab sofort gibt es zusätzliche Funktionen für die mehr als 450.000 Apps, die Facebook Analytics for Apps einsetzen. Entwickler und Werbetreibende können ihr Geschäft durch tiefgehende Insights sowie Push- und In-App-Benachrichtigungen ausbauen.

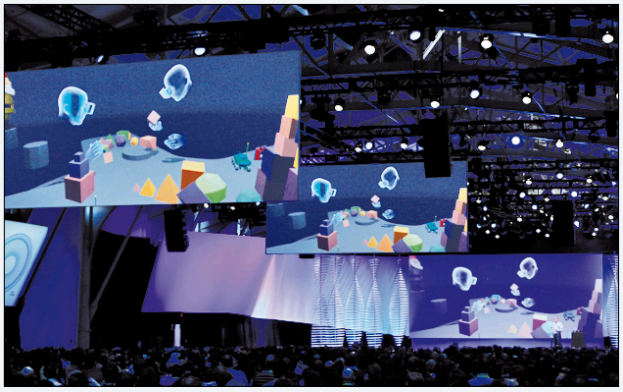
Mit der neuesten Version des Graph API werden neue Features eingeführt, die es Entwicklern ermöglichen, ihre Apps auf neue Arten mit dem Messenger, Facebook Live, Reactions, Video, dem Rights Manager und Facebook Anzeigen zu integrieren.

Am zweiten Tag der Konferenz sprach Facebooks CTO Mike Schroepfer über die VR- und AI-Technologien, die in Zukunft Menschen auf der ganzen Welt verbinden werden. Die Highlights seiner Ausführungen:

Connectivity Lab: Mit Terragraph und ARIES wurden zwei neue Systeme vorgestellt, die den Zugang zum Internet für Nutzer in strukturschwachen Regionen stabiler und schneller machen. Terragraph ist ein Funksystem für dicht besiedelte Städte. ARIES heißt Facebooks Machbarkeitsstudie zur Unterstützung von Funksignalen in Gebieten mit einer geringen Bevölkerungsdichte.

Artificial Intelligence: Facebooks Applied Machine Learning Team gab den Teilnehmern auf der F8 einen Einblick in die AI-Technologien, die im Hintergrund vieler Facebook-Produkte arbeiten.

Social VR: Facebook hat kürzlich ein Social VR Team zusammengestellt, das gerade an neuen VR-Erlebnissen arbeitet: Toybox und Sequencer. Toybox macht sich Oculus Touch zunutze und verbindet damit ein Gefühl der eigenen Präsenz im virtuellen Raum und die Möglichkeit, mit der Umwelt zu interagieren. Mit Sequencer kann der Nutzer sowohl mit seiner Körpersprache als auch über seine Stimme im virtuellen Raum kommunizieren.



Die Präsentationen auf der F8 haben einen Ausblick in die Zukunft von Facebook gegeben

React Native Ökosystem: React Native ist jetzt auch für Windows und Tizen – das Betriebssystem, das auf allen Samsung-Smart-TV-Geräten läuft – verfügbar. Außerdem gibt es jetzt ein Facebook SDK für React Native, das es Entwicklern einfacher und schneller ermöglicht, soziale Funktionen wie Anmelden, Teilen, App-Analytics und Graph APIs plattformübergreifend zu integrieren.

https://developers.facebook.com

Consol

Voraussetzungen für einen durchgängigen DevOps-Erfolg

DevOps-Ansätze reichen bis in die späten 1990er Jahre zurück, sie sind also prinzipiell nichts Neues. Dennoch führen viele Projekte nach wie vor nicht zum gewünschten Ergebnis. Der Münchner IT-Full-Service-Provider Consol zeigt auf der Basis eigener Erfahrungen aus zahlreichen DevOps-Kundenprojekten, welche fünf Punkte auf jeden Fall zu beachten sind:

- Kulturwandel aktiv angehen
- Investitionsvolumen berücksichtigen
- Simulationsumgebung nutzen
- Manuelle Prozessschritte eliminieren
- Systembrüche reduzieren

Joachim Hackmann ist Principal Consultant Software und Related Services beim Marktanalyse- und Beratungsunternehmen Pierre Audoin Consultants (PAC)

Das zur CXP Group gehörende Marktanalyse- und Beratungsunternehmen Pierre Audoin Consultants (PAC) mahnt auch vor dem Hintergrund wachsender Anforderungen an Flexibilität, Agilität und Schnelligkeit im Geschäftsumfeld zu einer engeren Verzahnung von Software-Entwicklung und IT-Betrieb. Joachim Hackmann, Principal Consultant Software & Related Services, führt aus: »Im Zuge der Digitalisierung werden kurze Innovationszyklen zu einem wettbewerbskritischen Faktor. Mit agilen Methoden wie Scrum oder Extreme Programming stehen zwar geeignete Mittel zur schnellen Software-Entwicklung zur Verfügung, doch der damit erzielte Geschwindigkeitsgewinn wird häufig vom schleppenden Übergang in den Betrieb wieder verworfen. Knapp 30 Prozent der von uns im Rahmen einer aktuellen DevOps-Studie befragten IT-Verantwortlichen beklagten beispielsweise Reibungsverluste, weil Test- und Zielumgebung nicht aufeinander abgestimmt sind. Hier kann eine verbesserte Automatisierung helfen. In der Tat planen oder diskutieren 58 Prozent der Nutzer mit DevOps-Erfahrungen Investitionen in Tools und Plattformen, die etwa das Testing verbessern.«

Henning von Kielpinski, Leiter Business Development bei Consol in München, ergänzt: »Beachtet man die angeführten fünf Prämissen, ist ein erster Schritt zu einem durchgängigen DevOps-Erfolg getan. Das können wir mit zahlreichen erfolgreich durchgeführten Projekten bei Unternehmen unterschiedlichster Branchen belegen. Natürlich darf man aber auch nicht die Augen vor der IT-Realität vor allem bei größeren Unternehmen verschließen. Hier verhindern oft die Strukturen eine nachhaltige Umsetzung von DevOps-Strukturen. Gründe sind zum Beispiel die Komplexität der Lieferantenbeziehungen oder die Auslagerung von IT-Services.«

www.consol.de

Build 2016

Die Xamarin-Tools sollen Open Source werden

Microsoft hat auf seiner Entwicklerkonferenz Build angekündigt, dass die Tools zur plattformübergreifenden Entwicklung von Apps für Desktop- und Mobilgeräte für alle Entwickler kostenlos angeboten werden sollen.

Auch Entwickler, die nur mit der kostenlosen Community Edition von Visual Studio arbeiten, sollen bald in den Genuss der Xamarin-Tools kommen – zumindest von einem Teil davon.

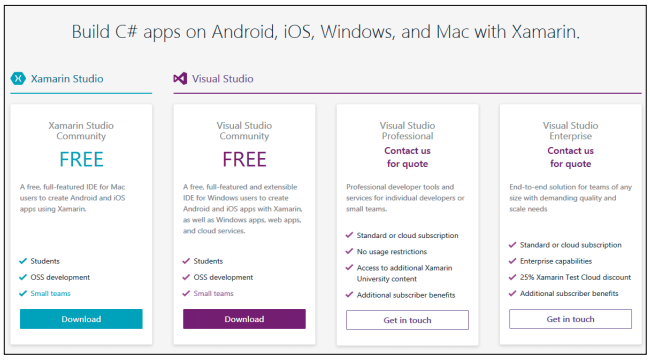
Bis zur Übernahme des Unternehmens Xamarin durch Microsoft Ende Februar war die Nutzung der Tools für Entwickler mit einer nicht unerheblichen jährlichen Zahlung verbunden. Künftig sollen die Tools nicht nur kostenfrei in Visual Studio integriert werden, sondern sogar als Open Source zur Verfügung stehen. Davon ausgenommen ist lediglich die Entwicklungsumgebung Xamarin Studio.

Die Freigabe der Quellen soll in den kommenden Monaten durchgeführt werden. Zum freigegebenen Open-Source-Paket gehört auch das Mono Framework, die .NET-Implementierung für Linux, Mac OS und Windows, also die Tools, die die Basis für Xamarin darstellen. Von der Ankündigung ausgenommen sind die grafischen Entwicklungswerkzeuge – was der Strategie bei Microsoft .NET entspricht, wo inzwischen alles außer Visual Studio Open Source ist.

Xamarin ist das Unternehmen, das vor ein paar Jahren das Mono-Projekt fortgeführt hat, eine Portierung von .NET für Linux und Mac. Mit der Gründung von Xamarin nutzte das Unternehmen sein Know-how, um leistungsfähige Werkzeuge zu kreieren, mit denen man mit einer C#/XAML-Codebasis sowohl Windows- als auch Linux- und Mac-Programme schreiben kann.

In den vergangenen Jahren wurden die Xamarin-Tools nach und nach auf Android und iOS ausgeweitet.

Seinen Store hat Xamarin bereits angepasst. Er verweist auf einen kostenfreien Download von Xamarin Studio Community for Mac sowie Visual Studio Community mit Xamarin.



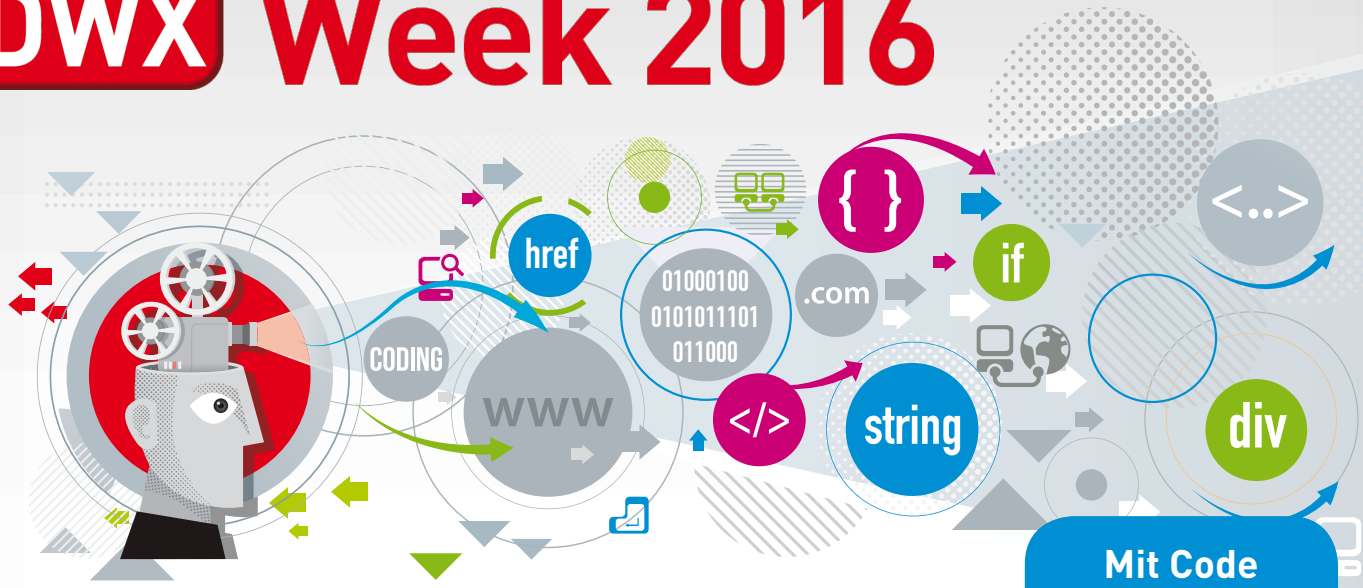
Der Xamarin-Store verweist schon auf einen kostenfreien Download von Xamarin Studio

Ein Teil der Xamarin-Features wird es offensichtlich nur mit den kostenpflichtigen Versionen von Visual Studio geben. Wie die Details dabei genau aussehen werden, ist noch unklar. Microsoft sprach bislang nur vage von »Additional subscriber benefits« und »Enterprise capabilities«.

http://xamarin.com ▶

DWX Developer Week 2016

20.-23. Juni 2016, Messe Nürnberg



Mit Code

DWX16wmd

€ 149,– sparen!

Das Event 2016 für .NET-, Web- & Mobile-Entwickler

Die Top-Experten der DWX 2016 (u.a.):


Christian Giesswein


Thorsten Kansy


Constantin Klein

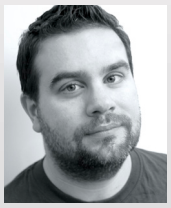

Peter Kröner


Stefan Lieser


Neno Loje


Bernd Marquardt


Pascal Precht


Golo Roden


Dr. Holger Schwichtenberg


Manfred Steyer


Ralf Westphal

developer-week.de | **DeveloperWeek**

Aussteller & Sponsoren:

adorsys · AIT · brainLight LIFE IN BALANCE · DevExpress · Fast Reports Reporting must be Fast! · GFU Cyrus AG · HeiReS · hemmersbach · JetBrains · MATHEMA · MAXIMAGO · Microsoft · Saxonia Systems So geht Software. · SDX · SHS VIVEON · SOS Software Service Value Added Distribution · Telerik · TEXT CONTROL · ThingWorx · hello, technology · UID

Veranstalter:

developer media
Neue Mediengesellschaft Ulm mbH

Präsentiert von:

dotnetpro
web & mobile DEVELOPER

Kaspersky Lab

Kampf gegen Ransomware

Die neue Version des Sicherheits-Tools Kaspersky Security for Windows Server beinhaltet jetzt auch eine Anti-Cryptor-Technologie gegen Ransomware.

Ransomware ist, wie die Beispiele Coinvault, Teslacrypt sowie jüngst auch Locky und Petya zeigen, kein neues Phänomen. Es breitet sich aktuell allerdings explosionsartig aus und bedroht weltweit Unternehmen und Organisationen.

Holger Suhl, General Manager DACH bei Kaspersky Lab

Das Bedrohungspotenzial ist erheblich, denn ein einzelner infizierter Rechner kann die Infektion binnen Minuten im gesamten Unternehmensnetz verbreiten und Schäden anrichten.

»Ransomware ist für Cyberkriminelle einfach zu entwickeln. Sie versprechen sich davon schnelles Geld. Zudem haben Opfer bei diesem unfairen Spiel keinerlei Garantie, verschlüsselte Daten durch die Zahlung eines Lösegelds jemals zurückzubekommen«, erklärt Holger Suhl, General Manager DACH bei Kaspersky Lab.
www.kaspersky.com/de

Developer Week 2016

Technologie und Networking

Seit der Build-Konferenz 2015 hat Microsoft sukzessive Technologien auf GitHub als Open Source freigegeben. Allem voran das .NET Framework. Die Verwaltung der Entwicklung liegt aber immer noch in den Händen der entsprechenden Abteilung bei Microsoft. Jedoch kann jeder nun Pull Requests an Microsoft schicken und um eine Übernahme seiner Änderungen in den Haupt-Branch bitten. Da Entwickler aber Teil dieser Welt sind, gilt für sie das Gleiche wie für Microsoft. Sie müssen offen sein für neue Technologien, die sie noch nicht kennen und aus dem Effeff beherrschen. Nur: Wie sollen sie sich diese neuen Techniken aneignen?

Hier kommt die Developer Week ins Spiel, die vom 20. bis 23. Juni 2016 in Nürnberg stattfindet. Als große Entwicklerkonferenz bietet sie genau diese Breite an Technologien an. Jeder Teilnehmer kann aus den rund 200 Sessions wählen und sich seinen persönlichen Lehrplan zusammenstellen.

Etwas mehr Datenbankzugriff gewünscht? Ein Deep Dive in C#? Oder doch eine große Portion Angular 2? Kein Problem. Sie bekommen auf der DWX all das. Kleine Häppchen in Form von einstündigen Konferenzsessions. Das ist die richtige Zeitspanne, um einen Überblick über ein Thema zu erhalten. Wenn es mehr Zeit erfordert? Genau an dem Punkt kommen die DevSessions ins Spiel: Zwei Stunden Zeit stehen für ein Thema zur Verfügung. Jeder Teilnehmer, der am Dienstag vor Ort ist, hat die Wahl zwischen zweimal zehn Themen. Hier geht es beispielsweise bei David Tielke um .NET Core oder bei Johannes Hoppe um einen Schnellstart mit Angular 2.

Wünschen Sie ein noch intensiveres Lernen, dann nehmen Sie doch an einem der Workshops teil. Hier erhalten Sie einen ganzen Tag Wissensvermittlung inklusive exklusiver Fragemöglichkeiten. Ein Dialog mit dem Referenten kann entstehen, der Sie auf neuem Terrain sicher führt. Wann immer etwas unklar ist – der kompetente Ansprechpartner ist im Raum.

Die Developer Week findet vom 20. bis 23. Juni 2016 in Nürnberg statt

Themen sind hier beispielsweise »Radikale Objektorientierung für die agile Softwareproduktion« mit Ralf Westphal. Aber Sie können auch lernen, wie Sie mit ECMAScript 2015 ganz modern entwickeln können. Marius Schulz zeigt es Ihnen.

Einen Blick über den Tellerrand sollen die beiden Keynotes bieten: in andere Themengebiete und in die Zukunft. Es freut das Programmkomitee ungemein, dass es dieses Jahr mit Jurgen Appelo und Scott Hanselman zwei versierte Vordenker als Sprecher gewinnen konnte, deren Thematiken nicht unterschiedlicher sein könnten.

Der Niederländer Jurgen Appelo hat sich als Management-Vordenker einen Namen gemacht. Moment mal: Die Developer Week ist eine Entwicklerkonferenz. Was haben die Teilnehmer mit Management am Hut? Management beginnt bei der Selbstorganisation des Tagesablaufs und hört beim Enterprise-Management nicht auf. »Managing for Happyness« lautet der Titel seiner Keynote. Glückliche Arbeiter sind produktiver. Glückliche Manager ebenso. In der Keynote vermittelt Appelo Tipps, die jeder sofort anwenden kann, damit seine Arbeit noch besser abläuft. Scott Hanselman, der Entwickler, Sprecher, Querdenker, Einmischer und Praktiker, gibt im Ausklang der Developer Week einen ganz anderen Ausblick. Was ist denn mit der Branche los, wenn es plötzlich Cross Compiler gibt, die aus LiveScript oder TypeScript JavaScript erzeugen? Wie kommt es, dass eine Sprache, die 20 Jahre alt ist, plötzlich so einen Auftrieb erhält? Hanselman wird zeigen, worauf das hinausläuft.

Wissen ohne die nötigen Werkzeuge reicht nicht. Und die Werkzeuge können Sie in der Ausstellung begutachten. In den Pausen oder auch mal während der Sessions haben Sie die Möglichkeit, sich Technologien und Tools an den Ständen der Partner zeigen lassen. Geben Sie Feedback oder fachsimpeln Sie einfach mit den Mitarbeitern der Firmen.
www.developer-week.de

Acquia

Vorteile eines Open-Source-CMS

Immer mehr Unternehmen kehren proprietären Lösungen den Rücken, weil sie Anwender viel zu stark an einen Hersteller binden. Bei Content-Management-Systemen heißt die Alternative Open Source. Acquia hat die größten Vorteile einer solchen Lösung in fünf Punkten zusammengefasst.

Bei einem Content-Management-System (CMS) steht heute nicht mehr die Webpräsenz eines Unternehmens im Mittelpunkt. Vielmehr geht es um eine Kombination von Content, Commerce und Communities, um eine einheitliche Brand Experience zu schaffen und Inhalte personalisiert, im richtigen Augenblick und über alle Kanäle auszuliefern. Unternehmen sind damit in der Lage, viel enger als bislang mit Kunden und Interessenten zu kommunizieren und die digitale Transformation voranzutreiben. Acquia, die Digital Experience Company, nennt die fünf wichtigsten Argumente, mit denen Open-Source-CMS wie Drupal überzeugen:

Michael Heuer ist Area Vice President and Country Manager – Central Europe (DACH und Benelux) bei Acquia

- Geschwindigkeit
- Anpassbarkeit
- Skalierbarkeit
- Community
- Sicherheit

»Proprietäre Systeme machen Unternehmen von einzelnen Herstellern abhängig, sie sind unterm Strich unflexibler, teurer, bieten weniger Features und lassen sich schwieriger in bestehende IT-Systeme integrieren«, erklärt Michael Heuer, Area Vice President and Country Manager – Central Europe (DACH und Benelux) bei Acquia in München. »Ein Open-Source-CMS wie Drupal ist weit mehr als ein statisches Tool für den Aufbau von Websites und Pages. Es ist einer der wich- ▶

Akamai-Umfrage

Deutsche Unternehmen als Opfer von Cyber-Attacken

Erschreckend viele deutsche Unternehmen waren schon einmal von einem Cyber-Angriff betroffen. Auch das Phänomen der Bitcoin-Erpressung ist hierzulande angekommen.

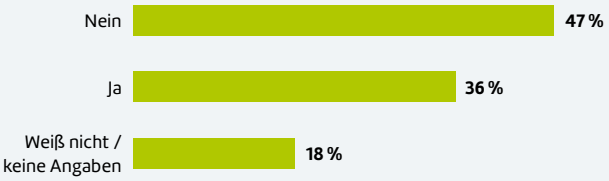
Mehr als ein Drittel der deutschen Unternehmen wurde schon einmal das Opfer einer Cyber-Attacke. Zu diesem Ergebnis kommt eine Umfrage, die Akamai Technologies GmbH auf der it-sa 2015 durchgeführt hat. Akamai hat dazu rund 200 Messebesucher zum Thema »Cyber-Security in deutschen Unternehmen« befragt. 36 Prozent der Befragten gaben an, dass ihr Unternehmen bereits von einem Cyber-Angriff betroffen war. Weitere 18 Prozent sagten, sie wüssten nicht, ob ihr Unternehmen schon einmal attackiert worden ist, oder wollten keine Angaben dazu machen.

Als neues Phänomen kristallisierten sich DDoS-Angriffe mit dem Ziel der Bitcoin-Erpressung heraus. Bis Mitte 2015 beobachtete das Prolexic Security Engineering & Response Team von Akamai beispielsweise Kampagnen, die auf das Konto der DD4BC-Gruppe gingen.

www.akamai.de

War Ihr Unternehmen jemals von Cyber-Attacken betroffen?

Aufgrund von Rundungen ergibt die Summe der Prozentzahlen nicht exakt 100



Umfrage: Cyber-Attacken aus deutsche Unternehmen.

web & mobile developer 6/2016 Quelle: Akamai

Stack Overflow

Der durchschnittliche Entwickler

Der Otto-Normalentwickler in Deutschland ist laut Stack-Overflow-Studie männlich, 27 Jahre alt und wohnt in Berlin.

Stack Overflow hat die Ergebnisse der fünften weltweiten Umfrage unter Entwicklern bekannt gegeben. Über 56.000 Entwickler aus 173 Ländern haben an der Umfrage teilgenommen. Global betrachtet ist der durchschnittliche Entwickler in der Studie männlich, lebt in den USA und ist zwischen 25 und 29 Jahre alt.

Ergebnisse der fünften weltweiten Stack-Overflow-Umfrage unter Entwicklern

In Deutschland ist der typische Entwickler ebenfalls männlich, zwischen 25 und 29 Jahre jung und lebt in Berlin. Er arbeitet in einem Unternehmen mit 20–99 Mitarbeitern und ist generell zufrieden mit seinem Job – aber offen für neue Herausforderungen. Die Arbeit macht ihm vor allem dann Spaß, wenn er viel Zeit mit Programmieren verbringt und er neue Technologien erlernt.

Besonders wichtig sind ihm dabei die Vereinbarkeit von Berufs- und Privatleben, ein gutes Gehalt und eine gute Unternehmenskultur.

http://stackoverflow.com

tigsten Innovationstreiber. Ergänzt um professionellen Support, wie ihn die Acquia Cloud Platform bietet, wird ein Open-Source-CMS zu einer hochflexiblen Infrastrukturplattform, auf deren Basis Unternehmen die digitale Transformation vorantreiben können.

Aufgrund seiner Erweiterbarkeit bietet ein solches Open-Source-CMS alles, was heute in Sachen Agilität, Personalisierung und Omni-Channel-Fähigkeit gefragt ist.«

www.acquia.com/de

DomainFactory

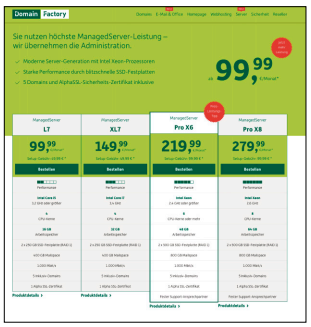
Managed-Server-Portfolio mit neuer Hardware

DomainFactory setzt für seine ManagedServer künftig auf eine neue Server-Generation mit Intel Core i5, i7 und Intel Xeon-Prozessoren. Für ein Plus an Performance sorgen zudem SSD-Laufwerke, die nun standardmäßig zum Angebot gehören. Mit dem Update gibt es für Kunden deutlich mehr Mailspace, fünf Domains und ein AlphaSSL-Sicherheitszertifikat inklusive.

»ManagedServer eignen sich aufgrund der dedizierten Ressourcen für langfristige Projekte, die Stabilität fordern. Mit dem Update bekommen Kunden noch mehr Leistung für ressourcenhungrige und anspruchsvolle Webanwendungen«, so Stephan Wolfram, Geschäftsführer beim Provider DomainFactory.

DomainFactory übernimmt die Wartung, Reparatur und Datensicherung mittels täglicher Backups. Als Betriebssystem kommt Linux Gentoo zum Einsatz.

Kunden haben aber außerdem die Möglichkeit, individuelle Anpassungen zum Beispiel bei Skript-Laufzeit, CPU-Sekunden und PHP-Einstellung



Die Leistungsstufen der ManagedServer von DomainFactory

vorzunehmen und können nach Bedarf eigene Dienste installieren. Dazu gehören ein eigener Apache und MySQL Server, PostgreSQL Server, JBoss und Jakarta Tomcat.

Wenn Geschwindigkeit zählt, kann für blitzschnelle Datenbankzugriffe optional MySQL-SSD-Storage gewählt werden. Für eine Spitzen-Performance von Skripts ist pro Domain FastCGI aktivierbar.

www.df.eu/de/server-hosting

Neueste Version von OpenStack

Besseres Nutzererlebnis

Die Version 13 von OpenStack zeichnet sich durch vereinfachte Handhabung, bessere Skalierbarkeit und optimiertes Nutzererlebnis aus.

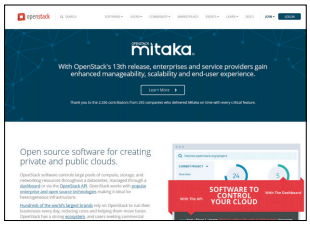
OpenStack Mitaka wurde von einer aus 2336 Entwicklern, Bedienern und Nutzern bestehenden internationalen Gemeinschaft aus 293 Organisationen entwickelt und implementiert. Da OpenStack nun seit fast sechs Jahren auf dem Markt ist, gilt sein Kern allgemein als ausgereift und stabil.

Deshalb hat sich die OpenStack-Community in letzter Zeit vor allem darauf konzentriert, den Einsatz sowie die Handhabung und Skalierung der Open-Source-Software zu vereinfachen.

Die Version 13 (Mitaka) enthält zahlreiche Weiterentwicklungen, mit denen die täglichen Aufgaben von Cloud-Bedienern und Administratoren erleichtert und verbessert werden.

Eines der Highlights ist die vereinfachte Konfiguration von OpenStack Compute (Nova): Durch zusätzliche Standardwerte müssen weniger Optionen manuell ausgewählt werden. Auch das Identity-Modul Keystone wurde deutlich vereinfacht.

Mitaka bietet auch zahlreiche Weiterentwicklungen bei der Skalierbarkeit von OpenStack Clouds. So kann zum Beispiel die in der Liberty-Version eingeführte Convergence Engine von Heat höhere Lasten und komplexere Aktionen für die



Release Mitaka: Bessere Skalierbarkeit und einfachere Benutzung

horizontale Skalierung verarbeiten und bringt zugleich bessere Leistung für zustandslose Vorgänge (Stateless Operations).

Mitaka zeigt deutlich das Engagement der Community, das Erlebnis für die Cloud-Benutzer zu verbessern, und zwar sowohl der Cloud-Bediener als auch der Endbenutzer, die Anwendungen auf Cloud-Systemen entwickeln oder implementieren. Beispielsweise bietet ein einheitlicher OpenStack-Client einen konsistenten Satz von Befehlen zum Erstellen von Ressourcen, sodass die Endbenutzer nicht die vielen Feinheiten jedes Dienst-APIs lernen müssen.

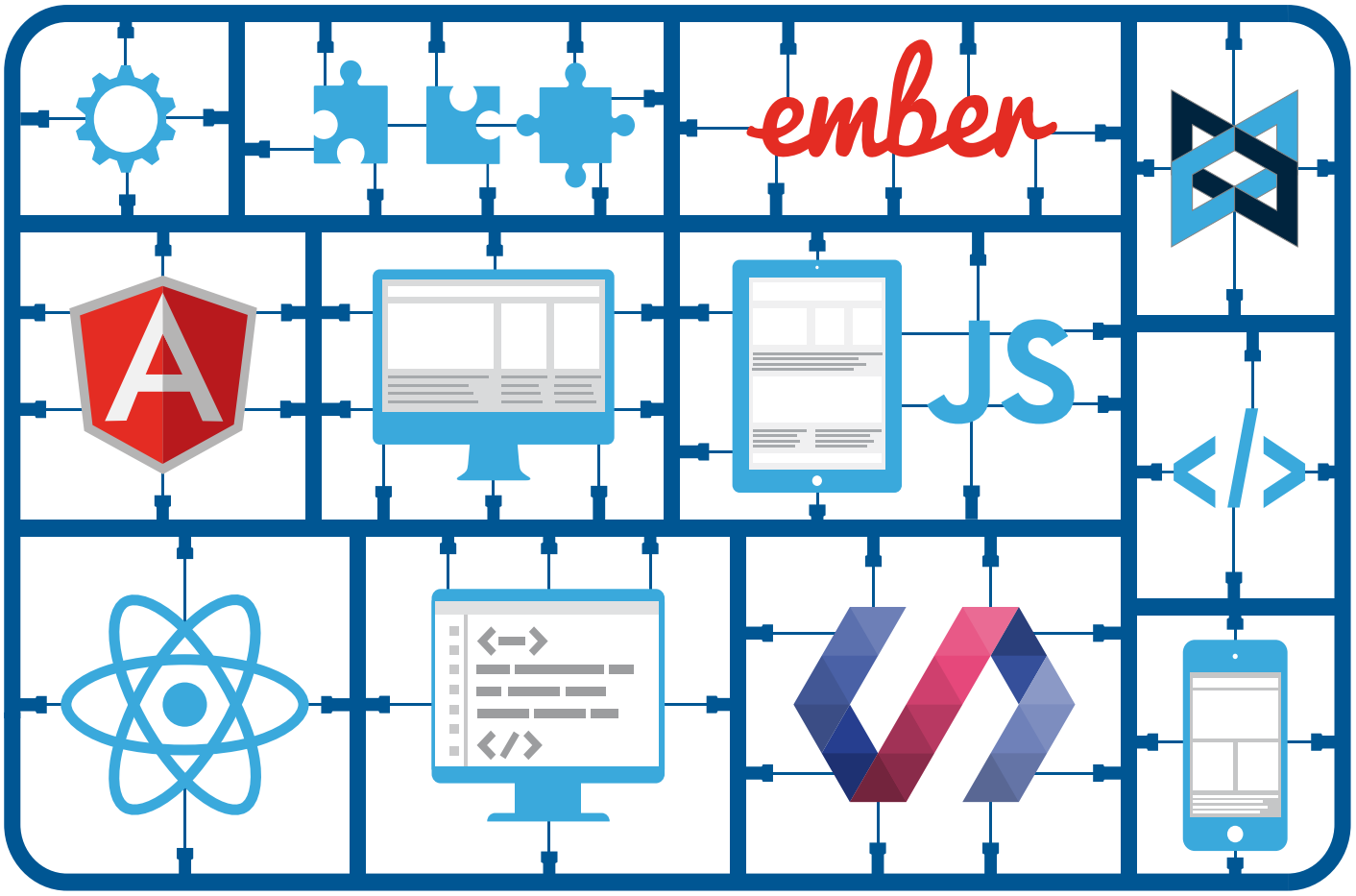
www.openstack.org

Jetzt kostenlos testen!

Das Fachmagazin für .NET-Entwickler

Testen Sie jetzt 2 kostenlose Ausgaben und erhalten Sie unseren exklusiven Newsletter gratis dazu.

www.dotnetpro.de/probeabo



Mathias Vietmeier

JAVASCRIPT-FRAMEWORKS IM VERGLEICH

Innovatoren vs. Traditionalisten

Leistungsstarke Frameworks mit unterschiedlichem Ansatz unterstützen JavaScript-Entwickler bei ihrer Arbeit.

Die aktuellen JavaScript-Frameworks fallen in eine von zwei Kategorien: Innovatoren oder Traditionalisten.

Erfahrene Entwickler möchten vorzugsweise auf ausgereifte, bewährte Technologien setzen und gleichzeitig neue Trends wie die bevorstehende groß angelegte Umstellung auf ECMAScript 6 (ES6) im Auge behalten, um sich darauf rechtzeitig vorzubereiten. Bei JavaScript ist die Aufgabe, den Überblick zu behalten, angesichts der zahllosen Frameworks und verschiedener Ansätze gar nicht so einfach.

Erfolgreiche Frameworks zeichnen sich durch eine hohe Qualität der Code-Basis, eine große, aktive Entwicklergemeinde und eine finanziell gesicherte Zukunft aus.

Das wohl wichtigste Kriterium, das zur Popularität eines JavaScript-Frameworks unter Web- und App-Entwicklern entscheidend beiträgt, ist der Status des jeweiligen Frameworks als quelloffene Software. Denn ein JavaScript-Framework ergibt erst dann wirklich Sinn, wenn auch hinreichend viele geübte Augen unter die Haube blicken dürfen, um gelegentlich auftretende Fehler aufzuspüren und Verbesserungen einzubringen. Abgesehen von den Kosten fördert das Open-Source-Lizenzmodell die Entstehung einer großen Entwicklergemeinde, die ihre Ideen und Wünsche beisteuert und so die weitere Entwicklung vorantreibt.

Selbstverständlich ist ein leistungsstarker Funktionsumfang in hochwertiger Qualität von großer Bedeutung, denn im Hinblick auf zeitsparende, innovative Features trennt sich schnell die Spreu vom Weizen. Ein gelungenes, durchdachtes Framework genießt gewöhnlich einen anhaltenden Zu-

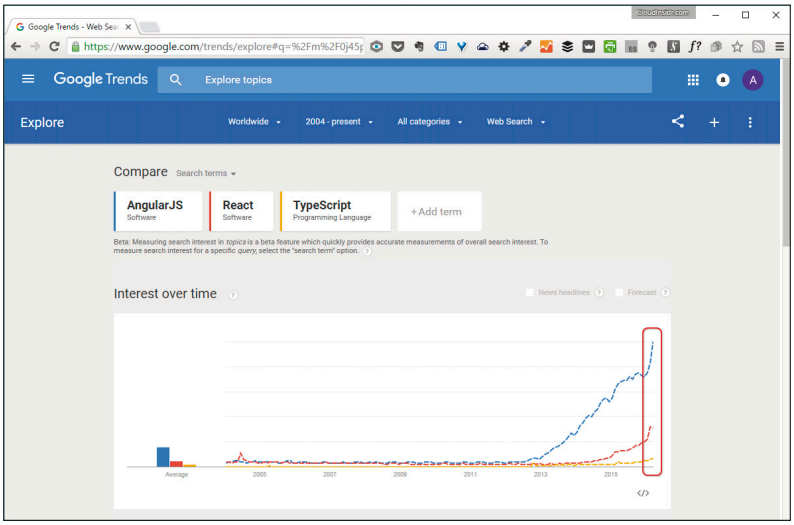
strom neuer Anwender, der wiederum der gesamten Gemeinde eine gewisse Zukunftssicherheit beschert.

Mit JavaScript am Puls der Zeit

Zu den traditionellen JavaScript-Frameworks sind in den vergangenen Monaten und Jahren viele neue hinzugestoßen. Einige beliebte Frameworks wurden inzwischen gründlich umgekrempelt. So ist etwa AngularJS in der vorliegenden neuen Version 2.0 nicht nur auf ECMAScript 6 hin optimiert, sondern zugleich in TypeScript von Grund auf neu entwickelt worden.

Die Gemeinde der JavaScript-Entwickler hat letztes Jahr mit der Standardisierung und Publikation von ECMAScript 6 einen enormen Durchbruch erzielt. Der Wachwechsel zwischen der Vorgängerversion ECMAScript 5 (ES5) und dem nun aktuellen ECMAScript 6 (kurz: ES6 oder ES2015) lag immerhin ganze sechs Jahre auseinander. Die finale Umstellung aktueller Webbrowser auf ES6 soll voraussichtlich im Lauf des Jahres 2016 abgeschlossen werden. Angesichts der Entschlossenheit und des hohen Entwicklungstempos, das die drei wichtigsten Browserhersteller Google, Mozilla und Microsoft an den Tag legen, könnte die hundertprozentige ES6-Kompatibilität aber schon im Sommer 2016 erreicht werden. Das Zeitfenster, um sich in ECMAScript 6 einzuarbeiten und den bestehenden Code zu optimieren, ist also vergleichsweise knapp.

Jeder der drei führenden Browserhersteller möchte als Erster die 100-prozentige Implementierung von ES6 für sich in Anspruch nehmen (Bild 1). Google Chrome kann ES6 in seiner Version 50 bereits zu 91 Prozent abdecken. Mozilla Firefox Version 45 und Microsoft Edge 13 liefern sich beim Wettstreit um den zweiten und dritten Platz ein Kopf-an-Kopf-Rennen.



Durchschlagend: ECMAScript 6 hat ein explosionsartiges Wachstum beim Interesse rund um JavaScript-Frameworks ausgelöst (Bild 2)

Mit 85 respektive 83 Prozent lässt sich zurzeit kein nennenswerter Unterschied ausmachen.

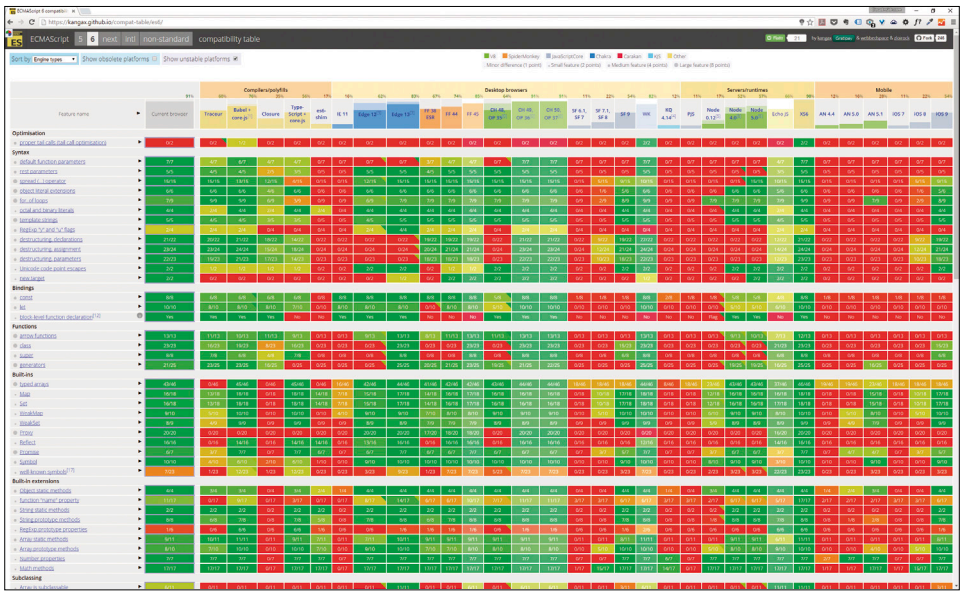
Ausschlaggebend wird aber nicht sein, wer zuerst werbewirksam die Hundertprozent-Marke erreicht, sondern wessen standardkonforme ES6-Umsetzung unter Alltagsbedingungen stabil und nicht zuletzt auch deutlich schneller läuft als die der Mitbewerber. Die JavaScript-Szene ist somit endlich wieder in Bewegung gekommen.

Wachablösung durch ECMAScript 6

Alle führenden JavaScript-Frameworks, die etwas auf sich halten, befassen sich zurzeit mehr oder weniger intensiv mit der bevorstehenden Umstellung auf ES6 (Bild 2).

Von ECMAScript 6 gibt es eine lange Liste neuer Features. Der Grad der Komplexität dieser Neuerungen variiert und das Einsatzspektrum reicht von einfachen Problemstellungen bis hin zu sehr anspruchsvollen Anwendungsszenarien. Es liegt also nahe, genauer unter die Haube von ECMAScript 6 zu schauen, um zu ermitteln, was in den modernen JavaScript-Frameworks auf Sie so alles zukommt.

In ES5 wurden Variablen üblicherweise mittels *var* deklariert. Nun können Sie in ES6 erstmals auch *let* verwenden. Der wichtigste Unterschied zwischen *var* und *let* besteht im beschränkten Geltungsbereich. Während sich beim Einsatz von *var* der Geltungsbereich auf die ganze Funktion erstreckt, in der die *var*-Deklaration eingeschlossen ist, beschränkt sich ▶



Fortgeschritten: Unterstützung für verschiedene Features von ECMAScript 6 durch JavaScript-Engines führender Compiler/Polyfills, Desktop- und mobiler Browser und Server-Runtimes (Bild 1)

der Geltungsbereich von *let* auf den jeweiligen Code-Block, in dem sich die Anweisung befindet, zum Beispiel:

```
if(true) {
  let x = 1; }
console.log(x); // undefiniert
```

Als Resultat der Verwendung von *let* anstelle von *var* wird der Code insgesamt lesbarer und verständlicher.

Konstanten in ES6

Möchten Sie Variablen mit beschränktem Gültigkeitsbereich deklarieren, so können Sie nicht nur *var* oder *let* nutzen, sondern auch *const*. Mittels *const* deklarieren Sie eine unveränderbare Referenz zu einem Inhaltselement.

Hierbei müssen Sie sich in ES6 umgehend auf einen Wert festlegen. Wenn Sie versuchen, den Wert der Konstante in diesem Beispiel zu ändern, würden Sie wie erwartet eine Fehlermeldung erhalten. Versäumen Sie es, der neu definierten Konstante einen Wert zuzuweisen, bekämen Sie ebenfalls eine Fehlermeldung.

Bei einer solchen Konstanten ist allerdings nur die Bindung unveränderlich, nicht der Wert an sich. Sie könnten nämlich Objekteigenschaften oder Feldelemente durchaus nachträglich ändern, zum Beispiel wie folgt:

```
const MEIN_OBJEKT =
{irgendwas: 1};
MEIN_OBJEKT.irgendwas = 'spitze';
```

Zu den Neuerungen in ES6 zählen die sogenannten Pfeil-Funktionen *(arrow functions)*. Diese erlauben es Ihnen, sehr gut lesbaren, kurzen und effizienten ES6-Code zu schreiben. Zum Beispiel:

```
let werke = [{titel: 'X', preis:
10}, {titel: 'Y', preis: 15}];
let titel = werke.map( item =>
item.titel );
```

Auffällig ist, dass die Syntax von Arrow-Funktionen kein *function*-Schlüsselwort beinhaltet. Die Funktion beinhaltet eine beliebige Anzahl von Argumenten (keines, eines oder auch mehrere Argumente), die Pfeil-Zeichenentsprechung => und den Funktionsausdruck. Die *return*-Anweisung versteht sich als implizit hinzugefügt. Falls Sie kein Argument oder aber mehrere Argumente verwenden, müssen Sie unbedingt Klammern verwenden.

Sollten Sie eine komplexere Logikabfrage benötigen oder schlicht die Lesbarkeit Ihres Codes verbessern wollen, sollten Sie den Funktionsausdruck in einem *({ … })*-Block einschließen.

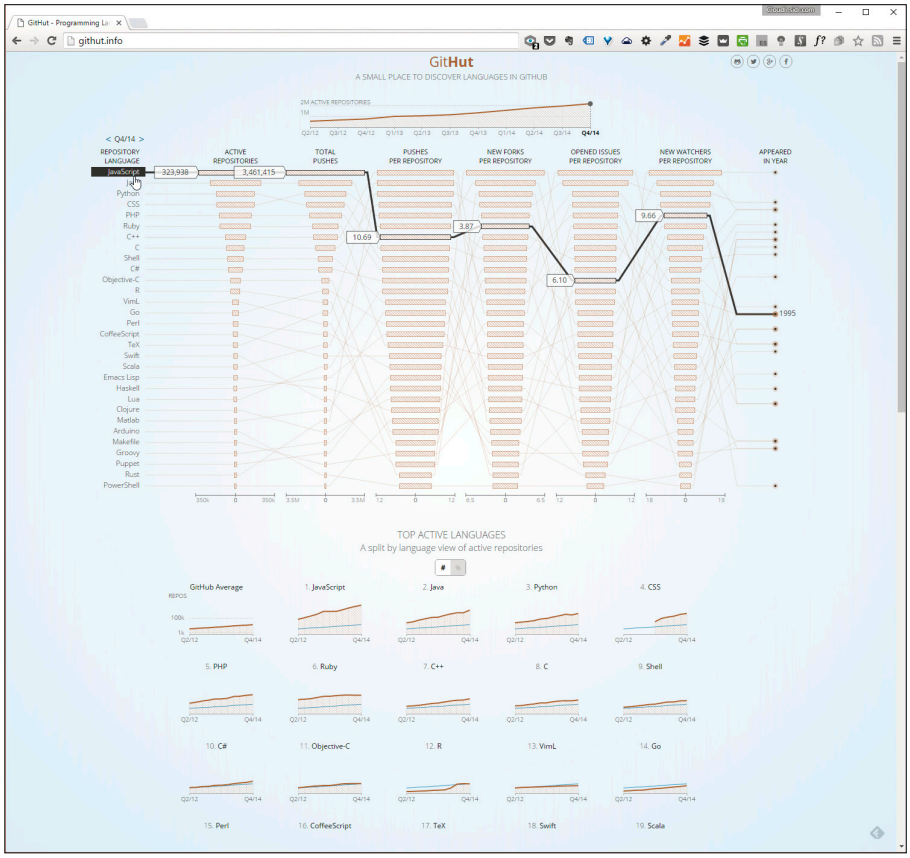
Zum String-Prototyp wurde eine Reihe komfortabler Methoden hinzugefügt. Bisher mussten Sie sich mittels der Methode *indexOf()* um die fehlende Funktionalität herumarbeiten. In ES6 können Sie schneller zu Werke schreiten, indem Sie Ihren Code zum Beispiel folgendermaßen aufbauen:

```
'meine Textzeichenkette'.startsWith('meine');
// true
'meine Textzeichenkette'.endsWith('my');
// false
'meine Textzeichenkette'.includes('Textzeichen');
// true
```

Auch Wiederholungen können Sie nun einfacher gestalten. Die Eingabe von: *,ES6 ,.repeat(3);* wiederholt den Textschnipsel dreimal.

Vorlagen-Literale sind eine ebenfalls praktische Art und Weise, um Textzeichenketten zu erzeugen und diese zu interpolieren. Die ES6-spezifische Syntax besteht ganz schlicht aus einem Dollar-Zeichen und einer geöffneten und geschlossenen geschweiften Klammer, also *${..}*. Template-Literale ES6 sind in Backticks eingeschlossen.

In ES6 können Sie nun beim Array-Objekt aus einigen neuen statischen Klassenmethoden und neuen Methoden des Array-Prototyps Nutzen ziehen.



Hyperaktiv: GitHut bescheinigt JavaScript den Nummer-eins-Status unter den beliebten Programmiersprachen auf GitHub (Bild 3)

So erzeugt *Array.from* eine Array-Instanz aus feldartigen und iterationsfähigen Objekten. Zu den passenden Praxisbeispielen für arrayartige Objekte zählen unter anderem Funktionsargumente, eine *nodeList*, die von *document.getElementsByTagName();* zurückgeliefert wird, und die neuen Map- und Set-Datenstrukturen.

Anstatt sich mit dem folgenden Workaround zu behelfen:

```
let items = Array.prototype.
slice.call(itemElements);
```

können Sie in ES6 besser den folgenden Code nutzen:

```
let itemElements = document.querySelectorAll('.items');
let items = Array.from(itemElements);
items.forEach(function(element) {
  console.log(element.nodeType)
});
```

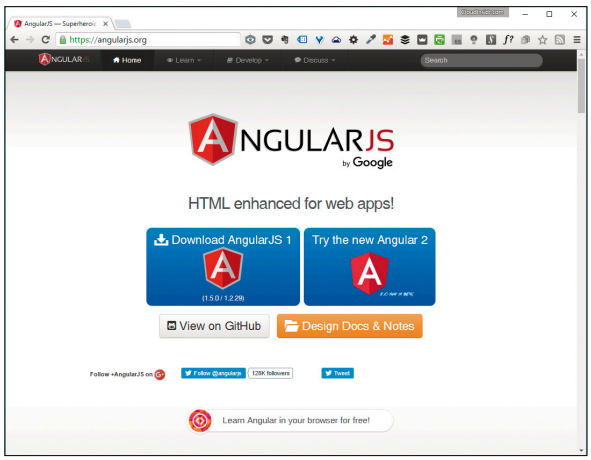
Einen praxisnahen Maßstab für die Relevanz – nicht unbedingt jedoch die Qualität – einer Programmiersprache beziehungsweise einer Technologie stellt die Anzahl von Beiträgen in einschlägigen Entwicklerforen wie StackOverflow dar. Für das JavaScript-Framework ReactJS gibt es vergleichsweise magere 6969 Beiträge. Backbone kommt auf 19.031 Beiträge. AngularJS spielt mit sage und schreibe 132.639 Beiträgen in einer eigenen Liga. JavaScript-getaggte Posts haben demgegenüber die 1-Million-Marke durchbrochen.

GitHut, ein Webdienst, der die Popularität von Programmier- und Skriptsprachen anhand von über zwei Millionen überwachter Repositories bewertet, stuft JavaScript als die wichtigste Sprache des Internets anhand der Gesamtzahl aktiver GitHub-Repositories (Platz 1) und Pushes (ebenfalls Platz 1) ein. Dieses hohe Aktivitätsniveau reflektiert sich in dem Tempo, mit dem der Fortschritt in diesem Bereich in den letzten zwei Jahren vorangeschritten ist (Bild 3).

Googles AngularJS

Bei Googles AngularJS handelt es sich um ein leistungsstarkes JavaScript-Framework für Webentwicklung und das beliebteste JavaScript-Projekt auf GitHub. Die in Kürze erscheinende zweite Generation des Frameworks hat die bevorstehende Umstellung der JavaScript-Engines auf ECMAScript 6 bereits vorweggenommen (Bild 4).

Angular 2 wurde für ECMAScript 6 von Grund auf neu entwickelt und zieht in aktuellen Browsern förmlich alle Register. Für die neue Version des Frameworks wurde eine bis zu achtfach höhere Performance beim Rendern von UI-Aktualisierungen gegenüber seinem Vorgänger nachgewiesen. Doch der wichtigste Vorteil von Angular 2 gegenüber AngularJS besteht nicht so sehr in der Leistung, als vielmehr in der Fähigkeit, erstmals nicht nur browserbasierte Webapplikationen, sondern auch mobile Apps für iOS und Android ins Leben zu rufen (Bild 5). Die Trennung des UI-Renderings vom DOM ermöglicht zudem die Umsetzung von Entwicklungsprojekten, die sich zuvor nur mit NativeScript von Telerik oder mit Facebooks React Native haben implementieren lassen (Bild 6).

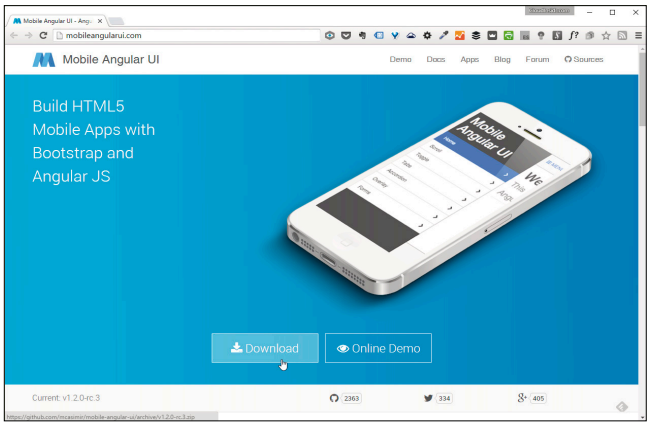


Umgekrempelt: Angular 2 ist nicht rückwärtskompatibel zu Angular 1.x (Bild 4)

Eine JavaScript-Applikation erzeugt normalerweise nur einen einzigen Thread. Google hat sich in Angular einen Workaround einfallen lassen: Angular-Code lässt sich innerhalb einzelner WebWorker ausführen, um so die Nebenläufigkeit quasi durch die Hintertür einzuführen.

Anders als die erste Generation des Frameworks, die auf reinem JavaScript basierte, entstand Angular 2 in Microsofts TypeScript. Google schuf damit quasi eine Abstraktionsebene zwischen Angular und ECMAScript. Die Umstellung hat es den Ingenieuren ermöglicht, unter anderem die Typüberprüfung und API-Auto-Vervollständigung in TypeScript zu nutzen und Microsofts leichtgewichtigen Visual Studio Code-Editor, VS Code, für Linux, Windows und OS X bereits in der Beta-Version einzuspannen. Dieses Endresultat war allerdings gar nicht geplant, sondern eher ein Nebeneffekt anderer Entscheidungen.

Als die Kernentwickler von Angular beschlossen hatten, das bereits etwas betagte Framework in seiner zweiten Generation an ES6 anzupassen, mussten sie sich zwangsweise auf den Traceur-Compiler von Google verlassen. Da ES6 zu diesem Zeitpunkt einige dringend benötigte Sprachkonstrukte fehlten, hat Google eine Erweiterung zu Traceur geschrieben und taufte diese auf den Namen AtScript. In der Zwischenzeit setzte auch Microsoft in TypeScript die Wei- ▶



Mobilisiert: Angular Mobile unterstützt die Entwicklung mobiler Apps (Bild 5)

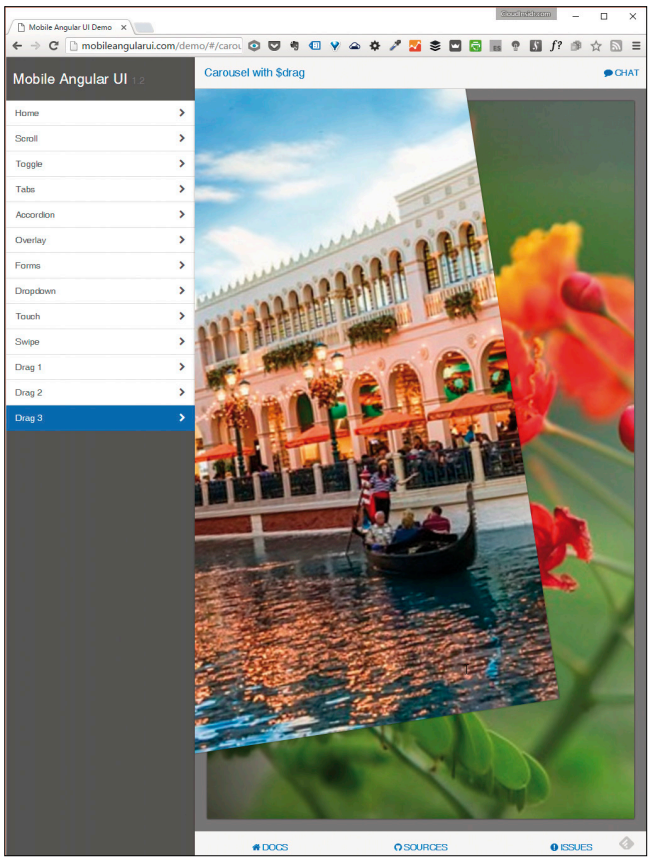
chen auf ES6. Google und Microsoft gingen daraufhin eine Zusammenarbeit ein, die Angular 2 maßgeblich prägen sollte. In einer Umfrage unter 2000 JavaScript-Entwicklern musste Google feststellen, dass nahezu jeder zweite Angular-Nutzer bereits ohnehin mit TypeScript entwickelte. Auch Teleriks NativeScript baut auf TypeScript auf, und somit lag die Entscheidung nahe, TypeScript als Unterbau für Angular zu verwenden. Google stellte die gesamte Codebasis von Angular von ES6 vollständig auf Microsofts TypeScript um; im Gegenzug hat Microsoft TypeScript in der Version 1.5 um die von Google benötigte AtScript-Funktionalität erweitert.

Die Zusammenarbeit von Google und Microsoft rund um die Weiterentwicklung von TypeScript und Angular fruchtete auch in zahllosen anderen Innovationen. Google und Microsoft arbeiten auch schon an Vorschlägen für ECMAScript-7-Erweiterungen rund um optionale statische Typen und Dekorateure zusammen. Google hat nebenbei zu der Entwicklung von Rx (Reactive Extensions) für .NET beigesteuert, einer Bibliothek für den Entwurf asynchroner ereignisbasierter Programme auf der Basis von beobachtbaren Sequenzen und LINQ-artigen Abfrageoperatoren.

Da ECMAScript 6 bisher noch nicht hundertprozentig mit aktuellen Browsern läuft, mussten sich die Angular-2-Entwickler eine Lösung einfallen lassen. Zurzeit erwarten die führenden Webbrowser immer noch das mittlerweile hoffnungslos antiquierte ES5. Den ES6-Code muss man also vorerst mit Hilfe eines sogenannten Transpilers in ES5-Code zurückwandeln. Googles Lösung ist der hauseigene Traceur-Compiler. Eine Alternative namens BabelJS von Sebastian McKenzie wächst zunehmend zu der von der Entwicklergemeinde bevorzugten Lösung heran (Bild 7).

Fortschrittlich: BabelJS.io, ein beliebter JavaScript-Compiler für ECMAScript 6 (Bild 7)

Damit der Transpiler zum Zuge kommen kann, müssen Sie allerdings einen Build-Schritt hinzufügen. TypeScript gibt reinen ES6-Code aus, den Sie gegebenenfalls in ES5 transpilieren müssen.



Flüssig: Drag and Drop von Bildern bietet Mobile Angular UI (Mobileangularui.com) (Bild 6)

Injektion in Angular 2

Google hat sich mit der Portierung von Angular nach ES6 in der 2.0-Version sehr viel Mühe gegeben, um noch hochwertigeren und nicht zuletzt noch schnelleren Code zu erzielen. Angular 2 erwartet aber auch vom Entwickler entsprechend mehr Disziplin.

In Angular 2 sei etwa der Einsatz von *forwardRef* auf ein absolutes Minimum zu reduzieren. Die Nutzung von *forwardRef* weist auf zyklische Abhängigkeiten oder Inkonsistenzen in der Deklaration der Angular-2-Dienste hin, wenn zum Beispiel ein abhängiger Dienst vor seiner eigener Abhängigkeit deklariert wird.

Offensichtlich sind sowohl zyklische Abhängigkeiten als auch Inkonsistenzen in der Deklaration schlechter Stil und sollten unbedingt vermieden werden. Sie können sich zum Beispiel mit einem Flow-Chart-Diagramm (zum Beispiel mittels *draw.io)* behelfen, um sich um solche Probleme herumzuarbeiten und einen vorhersehbaren Ablauf Ihrer Angular-2-basierten Apps zu gewährleisten.

Google rät außerdem davon ab, in Angular 2 globale Provider mit der *bootstrap*-Methode zu deklarieren, und empfiehlt, stattdessen auf sogenannte Top-Level-Komponenten zurückzugreifen. Provider, die in der *bootstrap*-Methode deklariert werden, sind von Google ausschließlich dafür vorgesehen, bestehende Provider zu überschreiben, und keinesfalls dafür, um Abhängigkeiten zu deklarieren.

Pipes zählen zu den besonders leistungsfähigen Konstrukten in Angular 2, die aber leider oft entweder missverstanden oder – zu Unrecht – ganz vermieden werden. Dabei könnten

Sie mit korrekter Anwendung von Pipes Ihre Entwicklungstechnik noch weiter verbessern:

```
@Pipe({
  name: 'sgTransformSomething'
})
class TransformSomethingPipe implements
PipeTransform {
  transform(input: any): any {
    //...
  }
}
```

Einbahnstraße: Das offizielle ng-forward-Logo spricht eigentlich eine klare Sprache (Bild 8)

Die vielen Vorteile, die Angular 2 gegenüber seinem Vorgänger aufweist, fordern jedoch einen hohen Preis: Um die fortschrittlichsten Features wie WebWorker-Einkapselung zu nutzen, müssen Sie Ihren bestehenden Code praktisch über Bord werfen und für Angular 2 umschreiben. Solange Sie sich allerdings mit nur einigen der gebotenen Vorteile begnügen können, stehen Ihnen zwei Upgrade-Wege zur Auswahl:

- **ng-upgrade:** Sie können Dienste und Bibliotheken von Angular 1 und 2 ohne Refactoring des bestehenden Codes sogar innerhalb einer einzigen View miteinander kombinieren, die Datenbindung zwischen den beiden Frameworks nutzen und Dienste injizieren;
- **forward:** Mit Hilfe dieser Bibliothek können Sie bereits in Angular 1.3+ im Stil von Angular 2 entwickeln (indem Sie zum Beispiel Ihren Code in Komponenten organisieren), sodass Sie jederzeit mit minimalem Aufwand das Framework wechseln können (Bild 8).

In beiden Fällen kommen Sie dennoch nicht umhin, Ihre gesamte Code-Basis früher oder später vollständig zu portieren.

Trotz der vielen Verbesserungen und der höheren Performance von Angular 2 werden Sie um das ausgiebige Testen Ihrer Lösungen nicht herumkommen. Zum Testen Ihres JavaScript-Codes können Sie Jasmine nutzen, ein verhaltensbezogenes Entwicklungsframework zum Testen von JavaScript-Code, das sich sowohl mit AngularJS 1.x und Angular 2 als auch mit anderen Dialekten sehr gut schlägt (http://jasmine.github.io).

Aurelia mit Zwei-Wege-Datenbindungs-Engine

Bei Aurelia handelt es sich um ein kostenfreies, quelloffenes JavaScript-Framework mit einer neuartigen adaptiven Zwei-Wege-Datenbindungs-Engine zum Anknüpfen von reinem JavaScript an DOM-Elemente einschließlich Web Components. Beim Binden an einen Ausdruck kann die adaptive Engine die passende Erfassungsstrategie – entweder objekt- oder eigenschaftsbezogen – eigenständig wählen. Das Framework kann sowohl *Object.observe* als auch *Array.observe*, *Getters* und *Setters* handhaben und bei Bedarf auf unsaubere Prüfungen zurückfallen.

Entwickler, die sich dafür entschieden haben, auf reines ES6 zu setzen, können dies in Aurelia tun. Der Chefentwickler des quelloffenen Frameworks Aurelia, Rob Eisenberg, ist vielen in der JavaScript-Gemeinde als ehemaliger Mitentwickler von AngularJS bereits gut bekannt. Nach seinem überraschenden Abschied von Google gründete er das Start-up Durandal und hob Aurelia aus der Taufe. Für 2016 erhofft sich Durandal vor allem aus der .NET-Entwicklergemeinde einen regen Zustrom für das Aurelia-Framework. Ausschlaggebend sollen dafür gemäß dem Anbieter vor allem zweierlei Gründe sein: Einerseits besteht Aurelia fast ausschließlich aus ES6-Klassen und andererseits ist das Referenzieren des Aurelia-Frameworks rein optional. Dies hilft, den gefürchteten Vendor-Lock-in bei diesem Anbieter zu vermeiden. Ob dem Team von Aurelia mit dieser fairen Strategie Erfolg beschert sein wird, wird aber erst die Zukunft erweisen. Verdient hätten sie es sicherlich.

Zurzeit befindet sich Aurelia 1.0 im Beta-Stadium. Sobald die finale Version 1.0 freigegeben wird, will sich das Aurelia-Team rund um Rob Eisenberg neue, anspruchsvolle Ziele setzen. Ganz oben auf der Liste steht das Erreichen einer nativen JavaScript-Performance, wie dies zurzeit zum Beispiel mit NativeScript oder React Native möglich ist. Qualitativ ist das Aurelia-Framework bereits heute sehr hochwertig, auch wenn es sicherlich nicht in die Kategorie Mainstream fällt.

Obwohl es sich bei Aurelia um ein quelloffenes JavaScript-Framework handelt, das sich kostenlos vom GitHub-Server herunterladen lässt, finanziert sich die Entwicklung primär über Unternehmen, die sich auf bezahlten Support verlassen.

Aurelia unterstützt zwei primäre Entwicklungsumgebungen: Visual Studio für Windows, und Node.js mit dem ▶

Korrekte Verwendung von Pipes in Angular 2

Beim Einsatz von Pipes in Angular 2 seien Ihnen folgende Merkregeln ans Herz gelegt.

Benennen Sie Ihre Pipe-Kontroller immer mit dem Suffix *Pipe*. Jede Pipe-Transformation sollte die allgemeinen Angular-2-Beschreibungsregeln einhalten *(KurzBeschreibung + Pipe)*.

Nutzen Sie beim Erstellen einer Pipe immer die PipeTransform-Schnittstelle. Auf diese Weise stellen Sie sicher, dass Ihre benutzerdefinierten Pipes auch die Anforderungen künftiger Angular-Versionen einhalten werden.

Benennen Sie Ihre Pipes in Binnenmajuskeln (in der englischsprachigen Angular-Dokumentation spricht man hier von *camelCase*).

Die sogenannten unreinen Pipes kommen nur dann zum Einsatz, wenn Sie einen Zustand der Transformation bewahren müssen.

Nutzen Sie für unreine Pipes immer das sogenannte On-Destroy-Interface.

Schwerpunkt auf Linux und andere Unix-Varianten. Außerdem können Sie Aurelia in einem Webbrowser auch ohne Installation einmal kurz auf die Probefahrt nehmen.

Je nachdem, für welche der beiden Programmiersprachen Sie sich bei der Konfiguration entscheiden sollten, verwenden Sie die entsprechend passenden Starter-Packages.Wenn Sie *index.html* in Visual Studio Solution Explorer laden, fährt der Microsoft IIS Express (Internet Information Server Express) hoch.

Die Konfiguration von Aurelia für Node.js kommt ohne jegliche grafische Oberfläche aus *(npm install http-server –g)*. Danach können Sie den Webserver mit dem Befehl *http-server -o -c-1* starten.

Wenn Ihnen zum bloßen Hineinschnuppern in Aurelia das Aufsetzen eines Webservers zu viel Aufwand bedeutet, können Sie auch schlicht einen Webbrowser verwenden, indem Sie darin einfach die Datei *index.html* aufrufen und mit den Entwickler-Tools inspizieren.

Aurelia-Apps bestehen aus HTML5, CSS und JavaScript. Wenn Sie einen Blick auf *index.html* werfen, sehen Sie eine HTML5-Seite, die im Grunde genommen als Vorlage für alle Aurelia-Apps dient. Im Bereich der *script*-Tags finden Sie *system.js*, den Modulloader, der sowohl alle benötigten Framework-Module als auch Ihren eigenen Code lädt. Danach wird der sogenannte *aurelia-bootstrapper* importiert.

Der Bootstrapper analysiert dann das HTML-5-Markup und sucht nach dem Vorkommen von *aurelia-app*-Attributen. In der Beispiel-App im Listing 1 wurde das *aurelia-app*-Attribut dem <*body*>-Element zugewiesen, sodass der Bootstrapper das *view-model* und die dazugehörige Ansicht *(view)* lädt. Die View befindet sich erwartungsgemäß in *app.html* und das View-Model in *app.js*; beide Quellen werden nun im DOM zu der resultierenden Aurelia-App kombiniert.

Das Benutzerinterface Ihrer Aurelia-App besteht immer aus eben diesen zwei Bestandteilen: der sogenannten View, die in HTML geschrieben ist und in das DOM hineingerendert wird, und dem View-Model, das bemerkenswerterweise bereits in ES6 geschrieben ist. Das View-Model liefert sowohl Daten als auch Verhaltenssteuerungsanweisungen an die View-Ansicht. Die sehr leistungsstarke Datenbindungs-Engine verknüpft dann die Elemente des Benutzer-Interfaces Ihrer App miteinander, sodass sich die Änderungen in den Daten sofort in der View reflektieren und umgekehrt.

Ein konkretes Beispiel für das View-Model mit einigen konkreten Daten illustriert Listing 2. Eingebunden in HTML könnte die View Ihrer Aurelia-App aussehen wie in Listing 3 dargestellt.

Alle View-Ansichten befinden sich gemäß den Konventionen der W3C-Spezifikation für Webkomponenten innerhalb des *template*-Tags. Mittels *value.bind="firstName"* entsteht eine Datenbindung zwischen dem Wert der Eingaben und der *firstName*-Eigenschaft im View-Model. Wann immer sich der Wert im Eingabeelement ändert, wird Aurelia dafür sorgen, dass diese Änderung sich in Ihrem View-Model automatisch reflektiert.

Die String-Interpolation *${fullName}* erzeugt eine Einweg-Datenbindung vom View-Model zur View, die automatisch in eine Zeichenkette konvertiert wird und in das Dokument hineininterpoliert.

Bei *submit.trigger="submit()"* handelt es sich wiederum um eine Ereignisbindung. Wann immer das *submit*-Ereignis des Formulars abgefeuert wird, wird die *submit*-Methode des View-Models ausgelöst.

Der Befehl *.bind* löst die standardmäßig vorgesehene Datenbindungsmethode für die jeweilige Eigenschaft aus, und zwar die Einweg-Datenbindung (von Model zu View) für alle Objekte außer allen Formularelementen (hier ist die Zwei-Wege-Datenbindung der Standard). Um die Standardmethode zu überschreiben, können Sie die Anweisungen *.one-way, .two-way* und *.one-time* nutzen, und für die Ereignisdelegation *.delegate* anstelle von *.trigger*.

Im Gegensatz zu Angular eignet sich Aurelia bei Weitem nicht für jede Problemstellung. Aurelia zählt zu den eleganteren und effizienteren, aber eben weniger umfangreichen JavaScript-Frameworks. Auch bricht es keine Rekorde der

Listing 1: index.html für Aurelia-Apps

```
<!doctype html>
<html>
  <head>
    <title>Aurelia</title>
    <link rel="stylesheet" href="styles/styles.css">
    <meta name="viewport"
    content="width=device-width, initial-scale=1">
  </head>
  <body aurelia-app>
    <script src="jspm_packages/system.js"></script>
    <script src="config.js"></script>
    <script>
      SystemJS.import('aurelia-bootstrapper');
    </script>
  </body>
</html>
```

Listing 2: app.js in Aurelia

```
export class Welcome {
  heading = 'Willkommen zu Aurelia!';
  firstName = 'Johann';
  lastName = 'Meier';

  get fullName() {
    return '${this.firstName} ${this.lastName}';
  }

  submit() {
    alert('Willkommen, ${this.fullName}!');
  }
}
```

Leichtgewichtheit. Wenn Ihnen primär ein schlankes, auf minimale Funktionalität abgespecktes JavaScript-Framework vorschwebt, dann sind Sie mit Backbone.js wahrscheinlich besser bedient.

Backbone.js

Webprogrammierer, die sich bei großen Projekten nicht von Angular beziehungsweise TypeScript vorschreiben lassen wollen, wie sie bestimmte Probleme lösen sollen, greifen lieber auf weniger allumfassende Frameworks zurück, um zusätzliche Freiheitsgrade zu haben, die sie sonst dem Programmierkomfort zuliebe aufgeben müssten.

Unter den Minimalisten erfreut sich Backbone.js einer großen Beliebtheit. Es verleiht Webapplikationen eine Struktur durch die Bereitstellung von Models, Collections und Views.

Backbone.js glänzt mit Features wie der Schlüssel-Wert-Bindung, nahezu beliebig definierbaren Ereignissen, deklarativer Ereignisauswertung in Views, einer Schnittstelle zu aufzählbaren Funktionen und einem JSON-basierten RESTful-API, durch das sich Backbone mit anderen Lösungen integrieren lässt.

Backbone.js liegt in zwei Editionen vor: in einer Entwicklerversion 1.2.3 inklusive vollständiger Kommentare und des ganzen Codes mit 69 KByte Umfang, und in einer Fassung für Produktionsumgebungen, die es erfreulicherweise gerade einmal auf magere 7 KByte bringt. Das Framework hat drei Abhängigkeiten: Underscore.js (ab Version 1.7.0), Backbone.View (für RESTful-konforme Persistenz und DOM-Manipulation) und jQuery (ab Version 1.11.0).

Bei der Entwicklung von Web-Apps in JavaScript zählt es zu den praxisbewährten Ansätzen, die Daten nicht an das DOM zu koppeln. Anfangs fällt es Ihnen vielleicht noch nicht auf, doch während Ihre JavaScript-App wächst, verwandelt sich der Code in einen verschachtelten Stapel von jQuery-Selektoren und -Callbacks, die alle angestrengt versuchen, das HTML-UI, Ihre JavaScript-Logik und die Datenbank auf dem Server um jeden Preis synchronisiert zu halten. Es ist meistens lediglich eine Frage der Zeit, bis sich die Synchronisation so weit verzögert, dass die Web-App sich nicht mehr flüssig nativ, sondern zunehmend langsam und träge anfühlt. Abhilfe können Sie hier nur mit einem strukturierten Ansatz schaffen.

Backbone repräsentiert Ihre Daten als Modelle, die erzeugt, validiert, gelöscht und auf dem Webserver gespeichert werden können. Jedes Mal, wenn eine Aktion im User-Interface ein Attribut eines Modells verändert, löst dieses ein Änderungsereignis *(change event)* aus. Entsprechend werden alle Ansichten, die den Zustand des Modells anzeigen, von der Veränderung benachrichtigt, sodass diese entsprechend reagieren und ein erneutes Rendern auslösen können.

Innovatoren haben Sponsoren

Nur die allerwenigsten Framework-Anbieter können von den Verkaufserlösen aus Premium-Support, kommerziellen Tools und sonstigen Dienstleistungen ihren Unterhalt bestreiten.

Neben Telerik hat es langfristig wohl nur Sencha so weit gebracht; in beiden Fällen liegen die Lizenzpreise im drei bis vierstelligen Bereich und damit außerhalb der Reichweite vieler talentierter Entwickler ohne eine große Agentur im Rücken.

Ein solider Sponsor für ein JavaScript-Framework klingt vielleicht auf den ersten Blick kontraproduktiv, weil Sie begründete Befürchtungen haben könnten, dass sich der Sponsor in strategische Entscheidungen einmischt. Prinzipiell kann man diesen begründeten Einwand nicht von der Hand weisen, aber im Fall des ReactJS-Frameworks, das bisher fast ausschließlich von Facebook entwickelt wurde, hat sich diese Sorge – zumindest bisher – als unbegründet erwiesen.

Entwickler von JavaScript-Frameworks sind in der Regel froh, einen großen Sponsor – und sei es auch nur ein einziger – zu finden, und wenn dieser noch das Kaliber von Facebook (ReactJS) oder Google (wie im Fall von Angular) hat, umso besser.

Sowohl bei Angular 2 als auch bei ReactJS hat sich das Sponsoring durch Google im Fall von Angular und Facebook im Fall von ReactJS positiv auf das Innovationstempo ausgewirkt. Beide Frameworks zeichnen sich durch eine solide und hochwertige Codebasis aus. Interessanterweise vertragen sich ReactJS und Angular.JS 2.0 erstaunlicherweise sehr gut miteinander, obwohl Google und Facebook sich sonst eher bitter bekämpfen.

Listing 3: app.html in Aurelia

```
<template>
  <section>
    <h2>${heading}</h2>

    <form submit.trigger="submit()">
      <div>
        <label>Vorname</label>
        <input type="text" value.bind="firstName">
      </div>
      <div>
        <label>Nachname</label>
        <input type="text" value.bind="lastName">
      </div>
      <div>
        <label>Vor- und Zuname</label>
        <p>${fullName}</p>
      </div>
      <button type="submit">Abschicken</button>
    </form>
  </section>
</template>
```

Modelle und Ansichten in Backbone.js

Zu den wichtigsten Vorteilen, die Ihnen Backbone.js bietet, zählt vor allem die saubere Trennung der Handlungslogik ▶

Ihrer Applikation von der Benutzerschnittstelle. Haben Sie die Handlungslogik und die Darstellung der Bedienoberfläche ineinanderfließen lassen, können Sie das eine nicht verändern, ohne das andere zu modifizieren und einen Domino-effekt von Anpassungen auszulösen.

Aus diesem Grund ist das sogenannte MVC-Paradigma (Model-View-Controller) entstanden, das die Handlungslogik (also die Programmsteuerung), die Benutzerschnittstelle (die Präsentation) und das Datenmodell jeweils sauber voneinander trennt. Backbone basiert auf einer Variante von MVC: dem Model-View-Presenter-Architekturmuster. In MVP wird jegliche Präsentationslogik in die Presenter-Schicht ausgelagert; bei der View handelt es sich um eine passive Schicht.

In vielen Programmiersprachen führt oft nur ein Weg ans jeweilige Ziel, sodass Umsteiger von anderen Sprachen oft ihre liebe Mühe damit haben, alte Gewohnheiten abzulegen. Backbone.js ist ein Beweis dafür, dass es bei der clientseitigen Programmierung durchaus auch anders und damit vor allem flexibler gehen kann.

Im Hinblick auf die Verwaltung von Referenzen zwischen Datenmodellen *(*Models) und Ansichten (Views) unterstützt Backbone.js ganz offiziell mehrere Ansätze. Vielleicht bevorzugen Sie direkte Zeiger, die 1-zu-1-Beziehungen *(model.view* und *vew.model)* aufbauen. Andere Programmierer ziehen es vielleicht vor, ein vermittelndes Kontrollerobjekt zu verwenden (das sogenannte *intermediate controller object)*, das für die Erzeugung und Verwaltung der Hierachieansichten verantwortlich zeichnet. Andere wiederum bevorzugen ein ereignisgesteuertes Konzept. Backbone.js unterstützt ganz offiziell alle diese drei recht unterschiedlichen Vorgehensweisen.

Stapelverarbeitung wird zwar sehr gerne auch auf Datenmodelle angewendet, aber die Handhabung kann in Abhängigkeit vom serverseitigen Aufbau durchaus äußerst verschieden ablaufen. Eine Möglichkeit stellt die Nutzung einzelner Ajax-Anfragen dar. Eine andere Möglichkeit besteht darin, RESTful-konforme Stapelverarbeitungsaufrufe zum Beispiel mit */notes/batch/destroy?ids=1,2,3,4* durchzuführen.

Sollte Ihnen weder die eine noch die andere Vorgehensweise zusagen, können Sie auch REST-Aufrufe mittels JSON (JavaScript Object Notation) tunneln, indem Sie *changeset*-Anfragen durchführen.

Bei Events handelt es sich um ein Modul, das einem beliebigen Objekt die Eigenschaft verleihen kann, benutzerdefinierte Ereignisse sowohl zu binden als auch auszulösen. So erzeugte Ereignisse müssen vor der Datenbindung nicht deklariert werden; auch die Übergabe von Argumenten lässt sich realisieren.

Binden einer Callback-Funktion an ein Objekt

Falls Sie in Backbone.js eine Callback-Funktion an ein Objekt binden möchten, gelingt dies mit *object.on(event, callback, [context])*. Die betreffende Callback-Funktion wird dadurch immer dann aufgerufen, wenn das zugehörige Ereignis auftritt. Wenn es darum geht, eine umfangreiche Anzahl von Ereignissen in den Griff zu bekommen, liegt es nahe, den Namespace unter Verwendung eines Doppelpunkts mit *poll:start* oder *change:selection* festzulegen.

Listing 4: Ereignisauswertung in Backbone

```
var Mensch = Backbone.Model.extend({
  defaults: {
    name: 'Julia',
    age: 21
  },
  initialize: function(){
    alert("Willkommen in Hamburg");
    this.on("change:name", function(model){
      var name = model.get("name"); // Petra
      alert("Mein neue Name ist " + name );
    });
  }
});
var human = new Human({ name: "Eva", age: 21});
human.set({name: 'Petra'});
// erst diese Operation löst das Ereignis aus
```

Um die Namespace-Konventionen in Backbone.js besser zu veranschaulichen, bietet sich ein schrittweises Vorgehen an. Zuerst einmal folgt ein Beispiel für gewöhnliche Namenskonventionen in Backbone.js:

```
Person
PersonView
PeopleCollection
PeopleView
```

Statt nun globale Funktionen zu deklarieren, wie im vorigen Beispiel, liegt es nahe, aus dem Namespacing Nutzen zu ziehen, sodass der Einsatz globaler Variablen auf ein absolutes Minimum reduziert wird. Jetzt bietet es sich an, den Code unter Nutzung der Namespace-Konventionen umzuschreiben:

```
App.Models.Person
App.Views.PersonView
App.Collections.PeopleCollection
App.Views.PeopleView
```

Die unter Verwendung der Namespace-Konvention von Backbone.js umgeschriebene Variante ist deutlich sicherer, aber leider unnötig redundant. Doch die Redundanz lässt sich leicht wie folgt beheben:

```
App.Models.Person
App.Views.Person
App.Collections.People
App.Views.People
```

Ein Datenmodell in Backbone bildet das Herzstück der JavaScript-Applikation; es beinhaltet interaktive Datenmuster sowie zugehörige Applikationslogik: die Transformationen, die

Validierungen, berechnete Eigenschaften und Zugriffsskontrollmechanismen. Um ein Datenmodell zu erstellen, genügt bereits:

```
var Mensch = Backbone.Model.extend({
  initialize: function(){
    alert("Willkommen in Hamburg");
  }
});
var mensch = new Mensch();
```

Datenmodelle, Collections und Views lassen sich in Backbone mit Hilfe der Funktion *initialize()* ins Leben rufen. Beim Initialisieren einer neuen Instanz des Datenmodells können Sie gleich auch einige Parameter daran übergeben, zum Beispiel, indem Sie die Variablendeklaration folgendermaßen modifizieren:

```
var mensch = new Mensch({ name: "Rudolf", age: 37});
```

Nachträglich lässt sich die Korrektur wie folgt umsetzen:

```
var mensch = new Mensch();
mensch.set({ name: "Rudolf", age: 37});
```

Die Übergabe eines JavaScript-Objekts an einen Konstruktor ist gleichwertig mit dem Aufruf von *model.set()*. Zum Abrufen der Attribute verwenden Sie die *model.get()*-Methode.

Alle Attribute eines Datenmodells können über Ereignis-Listener verfügen, die Änderungen der zugehörigen Werte aufspüren. In die Initialisierungsfunktion im Listing 4 ist ein Funktionsaufruf eingebunden, der Namensänderungen (in diesem Fall Abweichungen von den vordefinierten Standardwerten) überwacht und berichtet. Mittels *this.on("change", function(model){});* können Sie pauschal alle Attribute des betreffenden Datenmodells überwachen.

Backbone ermöglicht nebenbei auch die Validierung von Daten und Interaktionen mit einem Datenbankserver. Listing 5 illustriert das Erstellen eines neuen Benutzers in einer Datenbank auf dem Server, die über eine Tabelle namens *benutzer* verfügt und die Spalten *id, name, email* beinhaltet. Um mit dieser Tabelle zu interagieren, hat der Server den RESTful-URL */benutzer* erzeugt.

Da beim Erstellen des neuen Datenmodells keine ID an den Server übergeben wurde, tätigt der Server einen *POST*-Aufruf des URL */benutzer* und übergibt die Payload:

```
name:'Klaus', email: 'klaus@email.de'}
```

Der Server sichert daraufhin die Daten und gibt als eine Bestätigung die ID zurück.

Listing 5: Erstellen eines neuen Datenmodells

```
var UserModel = Backbone.Model.extend({
  urlRoot: ,/benutzer‘,
  defaults: {
    name: '''
    email: ' '
  }
});
var user = new UserModel();
// eine ID wird hier nicht gesetzt
var userDetails = {
  name: 'Klaus',
  email: 'klaus@email.de'
};
user.save(userDetails, {
  success: function (user) {
    alert(JSON.stringify(user));
  }
})
```

Flexible Rendering-Optionen in Backbone.js

Backbone.js zählt im Hinblick auf die verfügbaren Rendering-Optionen zu den flexibelsten JavaScript-Frameworks, denn es erlaubt Ihnen, die von Ihnen bevorzugte JavaScript-Vorlagenbibliothek zu nutzen, zum Beispiel:

- Underscore Templates (http://underscorejs.org/#template),
- Mustache.js (https://github.com/janl/mustache.js),
- Haml-js (https://github.com/creationix/haml-js),
- Eco (http://github.com/sstephenson/eco).

Backbone.js bietet zwar selbst keinen Templating-Mechanismus, doch die benötigte Funktionalität können Sie sich via *_.template* aus Underscore.js verschaffen. Underscore.js zählt ja ohnehin zu den wenigen harten Systemvoraussetzungen von Backbone.js. So können Sie ohne zusätzliche, speicherplatzverschwendende JavaScript-Templates auskommen.

Sollte Ihnen primär daran liegen, eine mobile App oder eine Webapplikation zu entwickeln, die in die Kategorie spartanisches Leichtgewicht fällt, stellt Backbone.js eine durchaus interessante Wahl dar.

Leider verträgt sich Backbone nicht mit EC6-Klassen. In EC6 lassen sich Eigenschaften nicht direkt einer Klasseninstanz hinzufügen, sondern lediglich Funktionen oder Methoden. Aufgrund der JavaScript-Vererbungsmechanismen können Eigenschaften einer Instanz bei der Initialisierung zugewiesen werden, während Methoden direkt auf einem Objektprototyp gesetzt und auf alle Instanzen angewendet werden. Das Anwenden von Eigenschaften direkt auf einen Prototyp würde sie allen Instanzen übergeben und eine Menge Probleme verursachen, sollte es sich bei der betreffenden Eigenschaft und ein Objekt mit veränderbarem Zustand wie ein Array handeln. Stattdessen ist es meistens sinnvoller, den Konstruktor der jeweiligen Klassen zu nutzen, um jeder neuen Instanz die benötigten Eigenschaften zuzuweisen. (Backbone fügt Eigenschaften direkt dem Prototyp zu.)

Den Konstruktor in der finalen Version von EC6 illustriert Listing 6. Der Aufruf von *super();* vor dem Referenzieren von *this* in dem Konstruktor einer Klasse, die eine andere Klasse erweitert, ist unverzichtbar. ▶

Listing 6: Ein EC6-Konstruktor

```
class DocumentRow extends Backbone.View {
  constructor() {
    super();
    this.tagName = "li";
    this.className = "document-row";
    this.events = {
      "click .icon":          "open",
      "click .button.edit":   "openEditDialog",
      "click .button.delete": "destroy"
    };
  }
}
```

Backbone führt auf der anderen Seite eine recht umständliche Initialisierung innerhalb des Konstruktors durch und erwartet, dass die Eigenschaften eines Datenmodells oder einer View vor dem Ausführen des Konstruktors Anwendung finden. Der Einsatz deklarativer Eigenschaften auf Subklassen, sodass diese durch den Konstruktor verarbeitet werden, ist in Backbone somit nicht möglich. Zwar ließen sich die Eigenschaften einer Instanz in Backbone als eine Methode definieren, aber dies widerspricht guten Programmierpraktiken. Auch andere Workarounds wie das erneute Ausführen desselben Konstruktors stoßen nicht auf Gegenliebe erfahrener Entwickler, da sie mehr Probleme verursachen als lösen.

NativeScript

Entwickler mobiler Apps können durch die Bereitstellung ihrer Apps auf iOS und Android OS stolze 97 Prozent des Marktes abdecken. Laut IDC hatte Android einen Marktanteil von 82,8 Prozent, während iOS auf immerhin 13,9 Prozent kam. Hierbei ist aber zu beachten, dass iOS-Apps trotz des absolut gesehen niedrigeren Marktanteils für viele Entwickler meist lukrativer sind (Bild 9). Entwickler können im Übrigen nicht nur mit bezahlten Apps, sondern auch mit kostenlosen Apps durch In-App-Verkäufe Geld verdienen. Android-Apps sind tendenziell günstiger als iOS-Apps und oft sogar kostenlos. Dadurch wird der hohe Marktanteil von Googles Android OS etwas relativiert. Als Konsequenz daraus müssen Sie sich als Entwickler mobiler Apps auf die sprichwörtliche Quadratur des Kreises einlassen und Cross-Plattform-Apps für iOS und Android OS anbieten, um finanziell rentabel zu arbeiten.

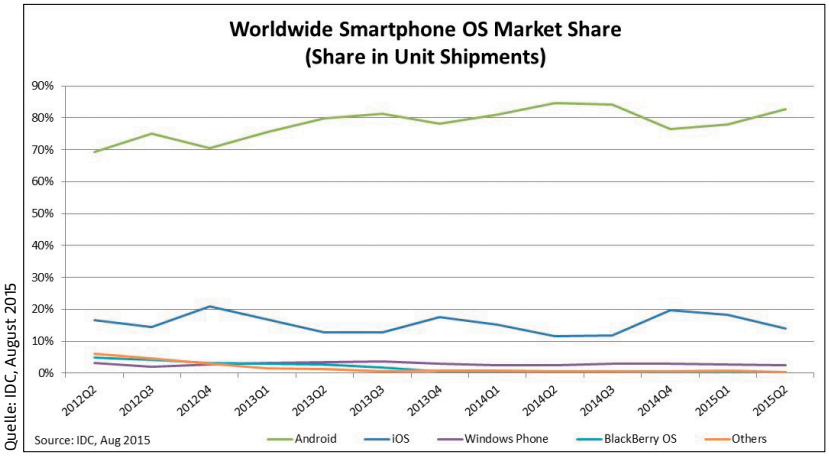
NativeScript ist eines dieser Cross-Plattform-Frameworks, das es Ihnen erlaubt, mobile Apps unter Verwendung einer gemeinsamen Code-Basis plattformübergreifend bereitzustellen und native App-Performance zu erzielen. NativeScript unterstützt sowohl iOS und Android als auch Windows Phone. Ob sich allerdings die Entwicklung für Windows Phone lohnt, sei angesichts eines Marktanteils von circa 2,6 Prozent einmal dahingestellt.

NativeScript erzeugt Apps mit einem nativen UI. Die Apps werden nicht im langsamen HTML-Rendering-Modus ausgeführt sondern können die beschleunigte JavaScript-Engine von iOS (Apples JavaScriptCore-Framework), Android OS und Windows Phone ansprechen und auf den nativen UI-Layer zugreifen.

NativeScript-Bibliotheken erzeugen eine mobile App in CSS, ECMAScript 5 und JavaScript. Beim JavaScript-Anteil der mobilen App verlässt sich Telerik auf das beliebte Node.js. Hierzu übersetzt NativeScript die plattformübergreifenden UI-Aufrufe in native Aufrufe der jeweiligen Zielplattform. Wenn Sie zum Beispiel eine UI-Komponente zu Ihrer mobilen App hinzufügen, greift NativeScript bei der Übersetzung auf die passenden nativen Plattform-APIs von iOS, Andoid OS oder Windows Phone zu.

Eigentlich könnte man bezüglich NativeScript von Telerik ja voll des Lobes sein, doch während es sich bei NativeScript selbst um ein quelloffenes, um damit kostenloses Framework handelt, sind die passenden (und zugegebenermaßen optionalen) Entwicklerwerkzeuge recht teuer: UI for iOS und UI for Android schlagen jeweils mit 499 US Dollar für eine lebenslange Lizenz zu Buche. Wenn Sie mit Teleriks NativeScript plattformübergreifend entwickeln möchten, müssen Sie sich also von 998 US Dollar trennen. Wenn Sie zudem eine angenehm schnelle Kompilierung Ihrer mobilen Apps in der Telerik-Platform-Cloud wünschen, werden weitere 39 US-Dollar pro Monat fällig. Einen Überblick aller von Telerik angeboten Entwicklerwerkzeuge finden Sie unter www.telerik.com/purchase.aspx.

Obwohl die Qualität der Telerik-Entwicklerwerkzeuge als hoch einzustufen ist, dürften die stolzen Preise vielen Entwicklern mobiler Apps eher sauer aufstoßen.

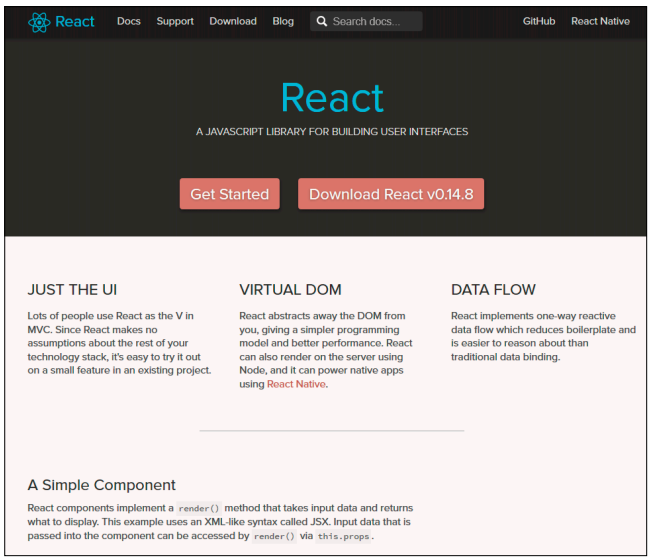


Quelle: IDC, August 2015

Aufgeteilt: Marktanteile von iOS und Android OS am Gesamtvolumen mobiler Endgeräte (Bild 9)

ReactJS

Das ReactJS-Framework ist ein weiteres modernes JavaScript-Framework, das sich großer Beliebtheit erfreut und auf vielen JavaScript-Konferenzen in aller Munde ist (Bild 10). Während Angular 2 den JavaScript-Unterbau für viele Websites und Web-Apps liefert, profiliert



ReactJS: Das Framework erfreut sich in Entwicklerkreisen großer Beliebtheit (Bild 10)

sich ReactJS als ein Visualisierungs-Framework nach dem MVC-Paradigma (Model-View-Controller).

Um ReactJS auf die Probefahrt zu nehmen, können Sie JSFiddle einspannen; so entfallen Download und Installation. Stattdessen können Sie in React gleich loslegen:

- React JSFiddle: https://jsfiddle.net/reactjs/69z2wepo
- React JSFiddle ohne den JSX-Präprozessor: https://jsfiddle.net/reactjs/5vjqabv3

React-Komponenten implementieren eine *render()*-Methode, die es Ihnen erlaubt, Eingabedaten einzulesen und daraufhin die gewünschten Daten anzuzeigen. In Listing 7 wird mittels JSX (einer XML-artigen Syntax für den JSX-Präprozessor) eine Begrüßung angezeigt. In Listing 8 kommt zum Vergleich für dieselbe Aufgabe reines JavaScript zum Einsatz. Die Eingabedaten, die an die React-Komponente übergeben werden, können von der Methode *render()* mittels *this.props* ausgelesen werden. Zusätzlich zum Einlesen der Eingabedaten (mittels *this.props*) kann eine React-Komponente auch einen internen Datenzustand verwalten, auf den Sie mittels *this.state* zugreifen können (Listing 9, Listing 10).

Sobald sich der Datenzustand einer Komponente ändert, wird das Markup neu gerendert, indem die Methode *render()* sowohl im JSX- als auch im JavaScript-Beispiel neu aufgerufen wird.

Mit dem nächsten logischen Schritt können Sie unter Verwendung der Methoden *state* und *props* eine kleine Zu-Erledigen-App erstellen. In dem Beispiel-Listing können Sie mittels *state* sowohl die Liste der zu erledigenden Aufgaben in Ihrer Mini-App verwalten als auch den Text erfassen, den der Benutzer eingegeben hat. Sowohl in der JSX- als auch in der JavaScript-Variante sollten Sie hierzu die Klasse *React.createClass* einsetzen (Listing 11).

ReactJS adressiert primär die Entwickler von Websites für Desktop- und Mobilgeräte.

Listing 7: JSX in React

```
var HelloMessage = React.createClass({
  render: function() {
    return <div>Hallo {this.props.name}</div>;
  }
});
ReactDOM.render(<HelloMessage name="Joachim!" />,
mountNode);
```

Listing 8: Reines JavaScript in React

```
"use strict";
var HelloMessage = React.createClass({
  displayName: "HelloMessage",
  render: function render() {
    return React.createElement(
      "div",
      null,
      "Hallo ",
      this.props.name
    );
  }
});
ReactDOM.render(React.createElement(HelloMessage, {
name: "Joachim!" }), mountNode);
```

Neue Funktionen in Apple Xcode 7

Apple Xcode 7.0, 7.1 und 7.2 bieten Unterstützung für neue Apple Hard- und Software.

Bei den 7.x.1-Versionen handelt es sich lediglich um Maintenance-Builds. Im Detail ließen sich die wichtigsten Neuerungen der verschiedenen Versionen wie folgt zusammenfassen:

- Xcode 7.0 beinhaltet das SDK für OS X Version 10.11, iOS 9, watchOS 2 und einige weitere kleinere Verbesserungen;
- Xcode 7.0.1 ist ein sogenannter Maintenance-Build;
- Xcode 7.1 bringt Support für neuere iOS-Geräte und für das neue Apple TV;
- Xcode 7.1.1 ist ein Maintenance-Build;
- Xcode 7.2 debütiert eine Fülle neuer Funktionalität, nicht zuletzt Unterstützung für iOS 9.2, OS X 10.11.2, tvOS 9.1 und watchOS 2.1;
- Xcode 7.2.1 ist ein weiterer Maintenance-Build.

React Native

Mobile Entwickler können auf React Native zurückgreifen, um native Cross-Plattform-Apps für iOS und Android OS in ReactJS und/oder JavaScript zu entwickeln (Bild 11). Mittels React Native können Sie die üblichen Standard-Plattform- ▶

Listing 9: Zustandsbehaftete Komponenten (JSX)

```
var Timer = React.createClass({
  getInitialState: function() {
    return {secondsElapsed: 0};
  },
  tick: function() {
    this.setState({secondsElapsed:
    this.state.secondsElapsed + 1});
  },
  componentDidMount: function() {
    this.interval = setInterval(this.tick, 1000);
  },
  componentWillUnmount: function() {
    clearInterval(this.interval);
  },
  render: function() {
    return (
      <div>Seconds Elapsed:
      {this.state.secondsElapsed}</div>
    );
  }
});
ReactDOM.render(<Timer />, mountNode);
```

Listing 10: Variante mit JavaScript

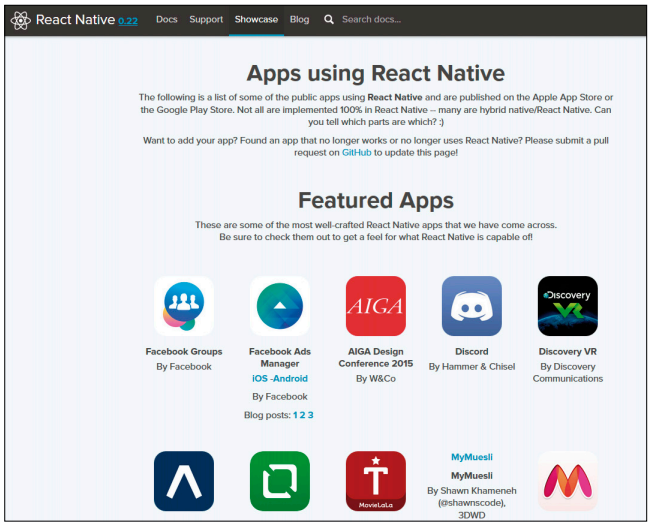
```
"use strict";
var Timer = React.createClass({
  displayName: "Timer",
  getInitialState: function getInitialState() {
    return { secondsElapsed: 0 };
  },
  tick: function tick() {
    this.setState({ secondsElapsed:
    this.state.secondsElapsed + 1 });
  },
  componentDidMount: function componentDidMount() {
    this.interval = setInterval(this.tick, 1000);
  },
  componentWillUnmount: function
  componentWillUnmount() {
    clearInterval(this.interval);
  },
  render: function render() {
    return React.createElement(
      "div",
      null,
      "Seconds Elapsed: ",
      this.state.secondsElapsed
    );
  }
});
ReactDOM.render(React.createElement(Timer, null),
mountNode);
```

komponenten wie zum Beispiel *UITabBar* in iOS oder Drawer in Android OS nutzen. Auf diese Weise gelingen Ihnen Cross-plattform-Apps, die dennoch ein konsistentes Look and Feel der jeweiligen Zielplattform und ihres Ökosystems konsequent einhalten. Diese Komponenten lassen sich unter Verwendung der passenden Komponentengegenstücke in React Native, also *TabBarIOS* für iOS und *DrawerLayoutAndroid* für Android OS, integrieren.

Die React-Native-Komponenten gleichen sich fast wie ein Ei dem anderen, damit Ihnen der Cross-Plattform-Ansatz wertvolle Zeit erspart. Die plattformspezifischen Unterschiede beschränken sich auf ein unbedingt nötiges Minimum.

Listing 11: To-do-List in React

```
var TodoList = React.createClass({
  render: function() {
    var createItem = function(item) {
      return <li key={item.id}>{item.text}</li>;
    };
    return <ul>{this.props.items.map(createItem)}
    </ul>;
  }
});
var TodoApp = React.createClass({
  getInitialState: function() {
    return {items: [], text: ''};
  },
  onChange: function(e) {
    this.setState({text: e.target.value});
  },
  handleSubmit: function(e) {
    e.preventDefault();
    var nextItems = this.state.items.concat(
    [{text: this.state.text, id: Date.now()}]);
    var nextText = '';
    this.setState({items: nextItems,
    text: nextText});
  },
  render: function() {
    return (
      <div>
        <h3>TODO</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit}>
          <input onChange={this.onChange}
          value={this.state.text} />
          <button>{'Add #' +
          (this.state.items.length + 1)}</button>
        </form>
      </div>
    );
  }
});
ReactDOM.render(<TodoApp />, mountNode);
```



React Native erlaubt die Entwicklung nativer Cross-Plattform-Apps für iOS und Android (Bild 11)

Alle Operationen zwischen dem JavaScript-Code und dem nativen Plattformcode laufen asynchron ab. Zudem können native Module von zusätzlichen Threads Gebrauch machen; so wird Ihr JavaScript plötzlich multi-threaded. Auf diese Weise können Sie zum Beispiel Bilder im Hauptthread decodieren, Daten im Hintergrund speichern, das Layout rendern, und trotzdem bleibt das User-Interface (UI) immer vollständig ansprechbar. Als Resultat daraus wirken React-Apps flüssig; sie reagieren immer unmittelbar auf die Benutzerinteraktionen. Die zeitnahe und damit gelungene Auswertung von Touch-Gesten im React-Native-Framework macht den gefühlten Unterschied zwischen drittklassigen Apps und solchen, die Ihnen ein natives Echtzeitgefühl bieten, aus. React Native bietet Ihnen ein leistungsfähiges System, das Ihnen die effiziente Auswertung von Touch-Gesten mit komplexen View-Hierarchien ermöglicht.

Links zum Thema

- Draw.io, ein Dienst für Flowchart-Diagramme
  https://www.draw.io
- Apples Bug-Reporter-Datenbank
  https://bugreport.apple.com
- Jasmine zum Testen von JavaScript
  https://jasmine.github.io
- AngularJS 1.x in ES5
  https://angularjs.org
- Angular 2, ein Framework in ES6
  http://angular.io
- Das Backbone.js-Framework
  http://backbonejs.org
- Das Aurelia-Framework
  http://aurelia.io
- ReactJS, eine JavaScript-Bibliothek von Facebook zum Entwickeln von Bedienoberflächen
  https://facebook.github.io/react
- React Native, ein JavaScript-Framework von Facebook zum Entwickeln von nativen mobilen Apps
  https://facebook.github.io/react-native

Tabelle 1: Die vorgestellten Frameworks auf einen Blick

Produkt	Anbieter	Web	Lizenz	Preis
AngularJS	Google	https://angularjs.org/	MIT-Lizenz	kostenfrei
Aurelia	Durandal Inc.	http://aurelia.io	MIT-Lizenz	kostenfrei
Backbone.js	Jeremy Ashkenas, Document Cloud	http://backbonejs.org	MIT-Lizenz	kostenfrei
NativeScript	Telerik	www.nativescript.org	Apache 2.0	kostenfrei
ReactJS	Facebook	https://github.com/facebook/react	Revidierte BSD-Lizenz	kostenfrei
React Native	Facebook	https://facebook.github.io/react-native	Revidierte BSD-Lizenz	kostenfrei

Fazit

Um JavaScript-Entwickler ist es wirklich sehr gut bestellt, denn es gibt sehr viele erstaunlich gelungene JavaScript-Frameworks für mobile Apps, Webapplikationen und schlichte Websites für Desktop- und Mobilgeräte (Tabelle 1). Sollte Ihnen vor allem an einem umfangreichen Programmierkomfort gelegen sein, entwickeln Sie wahrscheinlich bereits mit Angular. Sofern Sie bereit sind, Anpassungen an Ihrem Code vorzunehmen und Ihre gewohnten Workflows zu modifizieren, stellt Angular 2 eine verlockend leistungsfähige und vielseitige Lösung dar. Auch ReactJS und React Native kann man eine Menge an interessanten Features entlocken. ■

Filipe Pereira Martins und **Anna Kobylinska** sind international anerkannte IT-Berater mit Schwerpunkt auf Cloud-Lösungen. Sie stehen den Lesern der **web & mobile developer** gern per Twitter via **@D1gitalPro** und **@D1gitalInfo** zur Verfügung.

SCROLLEN MIT CSS3 BEEINFLUSSEN

Scrolleffekte

Ob und wie bestimmte Bereiche scrollbar sind, lässt sich durch eine Reihe neuer CSS3-Eigenschaften steuern.

An sich sind Webseiten scrollbar: Wenn es auf einer Seite mehr Inhalt gibt, als direkt angezeigt werden kann, stellt der Browser Scrollmechanismen zur Verfügung. Beim Scrollen bewegen sich dann standardmäßig alle Inhalte mit.

Auf das Scrollverhalten können Sie nun mit CSS auf verschiedene Arten Einfluss nehmen. Wie genau, das zeigt Ihnen dieser Artikel. Behandelt werden dabei klassische Angaben wie *background-attachment:fixed, overflow* oder *position: fixed*, aber es geht auch um die neueren Ergänzungen, also *background-attachment: local, position: sticky* oder *overflow-x* und *overflow-y*. Zudem gehen wir auf ganz neue Konzepte wie die Eigenschaft *scroll-behavior* oder die Snap Points ein, über die Sie Einrastpunkte beim Scrollen festlegen können (Tabelle 1).

Feste Hintergründe

Beginnen wir mit den Hintergrundbildern. Mit *background-attachment* steuern Sie das Verhalten von Hintergrundbildern beim Scrollen. Das Standardverhalten ist *scroll* – das heißt, die Hintergrundbilder bewegen sich beim Scrollen mit. Das lässt sich durch zwei Angaben ändern:

- *fixed:* Damit bleiben die Hintergrundbilder im Verhältnis zum Viewport stehen. Das heißt, auch wenn das Element selbst einen Scrollmechanismus hat, bleibt das fixierte Hintergrundbild unverändert.
- *local:* Hier bleibt das Element im Verhältnis zu seinem eigenen Kontext stehen. Wenn das Element selbst scrollbar ist, scrollt auch der Hintergrund mit.

background-attachment lässt sich bei mehrfachen Hintergrundbildern unterschiedlich definieren. So kann sich ein Hintergrundbild bewegen, während ein anderes stehen bleibt:

```
background-image: url(bild1.jpg), url(bild2.jpg);
background-attachment: scroll, fixed;
```

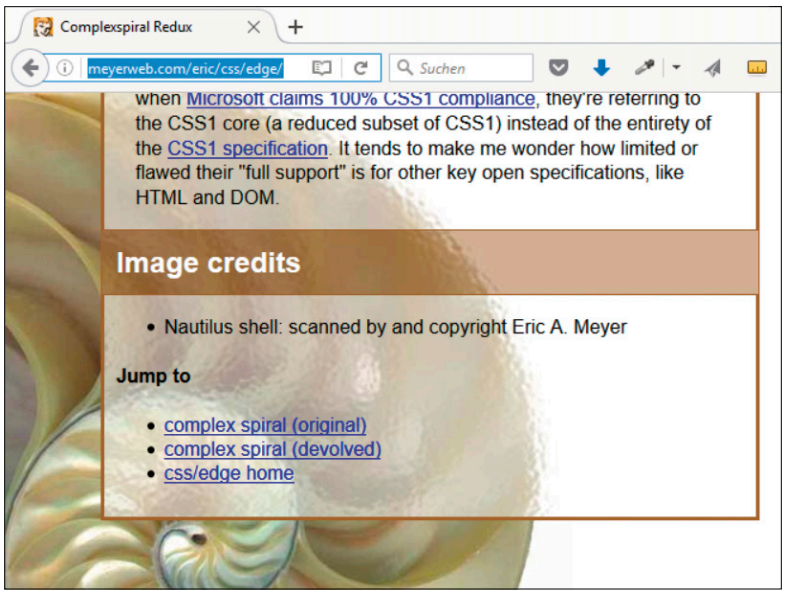
Durch diese Angabe bewegt sich *bild1.jpg* beim Scrollen mit und *bild2.jpg* bleibt stehen.

background-attachment: fixed gibt es bereits in CSS 2.1. Ursprünglich hat man es nur für *body* eingesetzt, aber es funktioniert bei allen Elementen, und darüber lassen sich interessante Effekte realisieren. Denn das Entscheidende an *background-attachment: fixed* ist, dass der Bezugspunkt für die Positionierung des Hintergrundbilds nicht das Element selbst ist, sondern der Viewport. Damit sind Effekte möglich,

Tabelle 1: Browserunterstützung der vorgestellten Eigenschaften

Feature	Anmerkung	Chrome/Opera	Firefox	Safari	IE/Edge
background-attachment: local	Mischung zwischen *background-attachment: scroll* und *background-attachment: fixed*	ja	ja	ja	ab 9
overflow-x	Wie *overflow*, aber beschränkt auf die Horizontale beziehungsweise Vertikale	ja	ja	ja	ja
overflow-y					
position: sticky	Mischung zwischen *position: relative* und *position: fixed*	–	ja	*-webkit-*	–
scroll-behavior	Mit *smooth* lässt sich ein sanftes, animiertes Scrollen auslösen	hinter Flag	ja	–	–
Snap Points	Einrastpunkte definieren	–	veraltete Version der Spezifikation	veraltete Version der Spezifikation mit *-webkit-*	veraltete Version der Spezifikation mit *-ms-*
*-webkit-overflow-scrolling: touch**	Natives Scrollen auch bei *overflow*-Boxen	–	–	ja	–

*kein Standard



Klassische Anwendung von *background-attachment: fixed* bei Body und einem weiteren Element: Die Hintergrundbilder liegen exakt übereinander (Bild 1)

wie sie Eric Meyer bei seinem Complexspiral-Distorted-Beispiel zeigt (Bild 1). Hier gibt es zwei Bilder, das normale Muschelbild wird *body* zugewiesen und als *fixed* deklariert:

```
body {
  background: white url(glassy-bg.jpg) 0 0 no-repeat
  fixed;
}
```

Innerhalb eines weiteren *#content*-Bereichs befindet sich die verzerrte Variante der Muschel als Hintergrundbild und ist ebenfalls *fixed* positioniert:

```
div#content {
  background: white url(glassy-ripple.jpg) 0 0 no-repeat
  fixed;
}
```

Das Hintergrundbild von *body* und das Hintergrundbild von *#content* befinden sich exakt übereinander – der Bezugspunkt für beide ist ja der Viewport. Der Effekt: Es sieht damit so aus, als würde der *#content*-Bereich das Muschelbild verzerren.

background-attachment: local ist eine Neuerung von CSS3. Eine Auswirkung hat es nur, wenn es bei einem Element angewandt wird, das selbst scrollbar ist, beispielsweise bei einem per *overflow: auto* definierten Element mit festen Ausmaßen. Wenn die Seite gesamt gescrollt wird, verhält sich *background-attachment: local* wie *background-attachment: scroll:* Alles bewegt sich mit. Wenn hingegen innerhalb des Element selbst gescrollt wird, bleibt das Hintergrundbild stehen.

Lea Verou zeigt, wie man *background-attachment: local* nutzen kann, um bei einem scrollbaren Bereich bei Bedarf einen kleinen Schatten oben und unten anzeigen zu lassen, der den Nutzer darauf hinweist, dass sich dort noch andere Inhalte verbergen.

Wenn es auf einer Webseite mehr Inhalt gibt, als im Viewport Platz haben, blendet der Browser automatisch Scrollbalken ein. Wie der Browser sich hingegen verhalten soll, wenn der Inhalt eines Elements größer ist als die definierten Ausmaße, können Sie über die Eigenschaft *overflow* steuern. Es sind folgende Werte möglich:

- *auto:* Inhalte werden beschnitten, können aber durch Scrollen erreicht werden. Scrollbalken sollen jedoch nur angezeigt werden, wenn es notwendig ist.
- *scroll:* Inhalte können durch Scrollen erreicht werden, Scrollbalken werden angezeigt.
- *visible:* Alle Inhalte sind sichtbar.
- *hidden:* Inhalte werden bei Bedarf beschnitten und sind nicht durch Scrollen erreichbar.

Das sind die klassischen, auch aus CSS 2.1 bekannten Angaben. CSS3 bietet jetzt Neuerungen: So können Sie mit *overflow-x* und *overflow-y* das Verhalten in der Horizontalen und in der Vertikalen einzeln steuern. Die andere Neuerung aus CSS3 ist, dass es einen weiteren Wert neben *auto, scroll, visible* und *hidden* gibt, nämlich *clip*. Das funktioniert im Prinzip wie *hidden*, macht aber jedes Scrollen unmöglich – bei *hidden* wäre hingegen noch ein Scrollen durch Programmierung möglich. Über die Details von *clip* wird jedoch noch diskutiert.

Mit Hilfe von *overflow-x* lässt sich beispielsweise einfach ein Karussell erstellen. Dabei sind die Bilder nebeneinander angeordnet und die nicht direkt sichtbaren können durch Scrollen erreicht werden. Für dieses Verhalten sorgt die Eigenschaft *overflow-x: scroll*.

Listing 1: HTML-Basis für Karussell

```
<ol class="karussell">
  <li><img src=
  „http://placehold.it/400x300/ff0000?text=1"></li>
  <li><img src=
  „http://placehold.it/400x300/00ff00?text=2"></li>
  <!-- weitere Bilder -->
</ol>
```

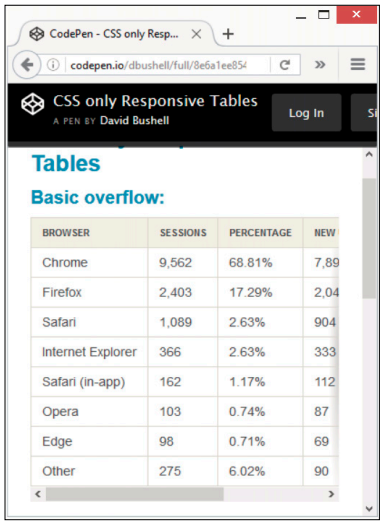
Listing 1 zeigt die HTML-Basis für ein Karussell – eine geordnete Liste, bei der in den einzelnen *li*-Elementen die Bilder platziert sind. Für die Karussell-Darstellung müssen wir die *li*-Elemente nebeneinander anordnen, was *display: inline-block* erledigt. Damit der Benutzer sich horizontal durch die Bilder scrollen kann, verwenden wir *overflow-x: scroll*. Die Details zeigt Listing 2.

Sehr nützlich ist für *overflow-x* auch bei der Realisierung einer responsiven Tabelle. Wenn zu wenig Platz zur Ver- ▶

fügung steht, sind nur die ersten Spalten zu sehen, die anderen sind durch Scrollen erreichbar. Das ist eine elegante CSS-only-Lösung für das Problem von Tabellen beim responsiven Webdesign (Bild 2).

Mit *position* bestimmen Sie die Art der Positionierung. Elemente, die eine andere Angabe als *position: static* haben, können dann über *top, left, right* oder *bottom* platziert werden. Worauf sich aber eine Angabe wie *top: 10em* bezieht, hängt von der jeweiligen Positionierungsart ab. Im folgenden Beispiel wird die relative Positionierung eingesetzt:

```
.box {
  position: relative;
  top: 0.5em;
  left: 1em;
}
```



Responsive Tabelle dank *overflow-x* (Bild 2)

Listing 2: CSS-Code für Karussell

```
.karussell li {
  display: inline-block;
  padding: 0;
  margin: 0;
  width: 400px;
  height: 300px;
}
ol.karussell {
  list-style-type: none;
  white-space: nowrap;
  overflow-x: scroll;
  overflow-y: hidden;
  width: 100%;
  height: 300px;
}
```

Damit wird das Element im Vergleich zur Position, die es ohne Positionierung hätte, um 0.5 em von oben und 1 em von links verschoben. Die auf *.box* folgenden Elemente verhalten sich so, als stünde *.box* noch an seiner ursprünglichen Position. Interessant im Hinblick auf das Verhalten beim Scrollen sind die Positionierungsarten *fixed* und *sticky*. Ein mit *position: fixed* versehenes Element bleibt beim Scrollen stehen – ähnlich wie *background-attachment: fixed*, nur dass sich *position: fixed* nicht auf Hintergrundbilder, sondern auf Elemente im Vordergrund bezieht.

```
.fixiert {
  top: 1em;
  left: 0;
  position: fixed;
}
```

Bei den *paged media*, also zum Beispiel beim Ausdruck, sollen mit *position: fixed* versehene Elemente auf jeder Seite wiederholt werden – Firefox und Internet Explorer machen das auch (Bild 3), Chrome hingegen nicht. *position: fixed* ist bereits in CSS 2.1 definiert, allerdings gibt es gerade bei älteren mobilen Geräten immer wieder Probleme bei der Unterstützung – über die Details dazu informiert Brad Frost.

Eine Mischung zwischen relativer und fixer Positionierung ist das neu vorgesehene *position: sticky*. Gehen wir von folgendem HTML-Code aus:

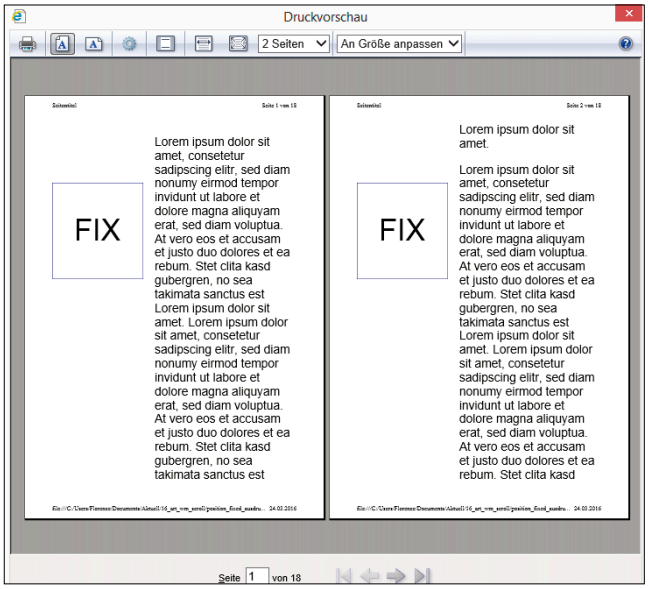
```
<p>Lorem ipsum <!-- längerer Absatz --></p>
<p class="sticky">position: sticky</p>
```

Außerdem gibt es folgenden CSS-Code, in dem als Positionierung *sticky* angegeben ist:

```
.sticky {
  top: 1em;
  left: 0;
  position: -webkit-sticky;
  position: sticky;
  background: rgba(255, 255, 255, 0.9);
}
```

Damit befindet sich das Element *.sticky* zuerst an seiner normalen Position unterhalb des vorher stehenden Absatzes. *.sticky* bewegt sich beim Scrollen mit wie die restlichen Elemente der Webseite.

Wenn die *.sticky*-Box jedoch 1 em vom oberen Rand des Browserfensters angekommen ist *(top: 1em)*, bleibt sie stehen und bewegt sich beim weiteren Scrollen nicht mehr mit – sie verhält sich in diesem Moment also so, als wäre *position: fixed* definiert (Bild 4).



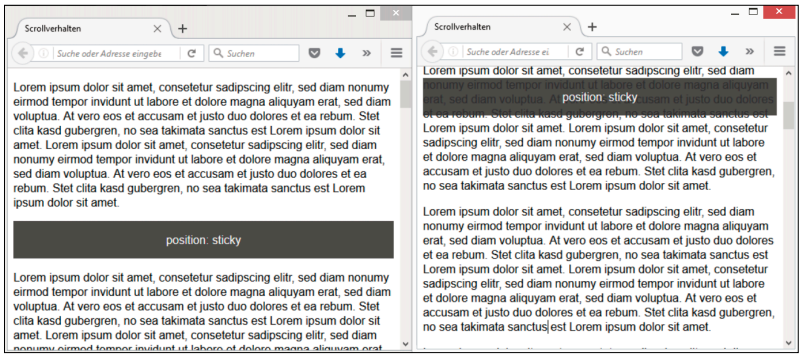
Beim Ausdruck sollen *fixed* positionierte Elemente auf jeder Seite wiederholt werden, wie hier im IE 11 (Bild 3)

Die Browserunterstützung für *position: sticky* ist allerdings noch durchwachsen. Es funktioniert in Firefox und mit dem *-webkit*-Präfix in Safari und iOS. In Chrome konnte es über ein Flag aktiviert werden, diese Möglichkeit wurde aber beim Wechsel zu Blink entfernt.

Die Gründe hierfür waren, dass die ursprüngliche Implementierung von *sticky* nicht gut mit dem Scrollen und dem Zusammenbau der Webseite (Compositing) zusammenspielte. Die Eigenschaft soll aber wieder neu eingeführt werden, wenn die geplanten Verbesserungen beim Scrollen und beim Compositing durchgeführt sind.

Bis die Browserunterstützung besser ist, haben Sie folgende Möglichkeiten, das Problem zu lösen:

- Sie nutzen *position: sticky* als Enhancement. Chrome ignoriert die Angabe *position: sticky*, damit bleibt das Element in seiner normalen Anordnung und der bei *top* definierte Wert für die Verschiebung hat keine Auswirkung.
- Außerdem gibt es zwei Polyfills, die *sticky* in nicht unterstützenden Browsern nachbessern – *stickyfill* von Oleg Korsunsky und *fixed-sticky* von der Filamentgroup. Beim echten Einsatz sollten Sie aber sicherstellen, dass die Performance nicht darunter leidet.
- Mehr Optionen als bei den bisher vorgestellten Lösungen haben Sie bei Sticky Elements von Waypoints – das ist eine Lösung, die vollständig auf JavaScript setzt, aber dann auch die Möglichkeit bietet, ein Element, das *sticky* ist, wieder in die normale Position zurückzuführen.



position: sticky: das Element bewegt sich beim Scrollen mit (links), bis es die bei *top* vorgegebene Position erreicht hat, und bleibt dann stehen (rechts) (Bild 4)

Prinzipiell gilt, dass manche Anpassungen an der Positionierungsart von Elementen nicht bei allen Viewport-Größen sinnvoll sind. Deswegen kann es bei Bedarf nützlich sein, für kleinere Screens auf die Standardpositionierung – *static* – zu wechseln. Über Media Queries lässt sich das einfach bewerkstelligen:

```
@media all and (max-width: 30em) {
  .fixed {
    position: static;
  }
}
```

Bei langen Seiten sind Nach oben-Buttons nach wie vor empfehlenswert. Standardmäßig wird dann direkt nach oben gesprungen. Das wirkt etwas unnatürlich, da der Benutzer selbst ja Zeit zum Scrollen gebraucht hat.

Sanftes Scrollen mit CSS

Schöner ist es, wenn der Weg nach oben spürbar wird, dann nimmt der Benutzer die Bewegung wahr und behält die Orientierung.

Für dieses sanfte Scrollen verwendet man üblicherweise JavaScript – es geht aber mit der neuen CSS3-Eigenschaft *scroll-behavior* auch über CSS. Der Standardwert von *scroll-behavior* ist *auto*, das heißt, es wird direkt nach oben gesprungen. Das sanfte Scrollen können Sie hingegen über den Wert *smooth* auslösen. Ein Beispiel: Am Anfang der Webseite steht ein Absatz mit einer ID. Ganz unten gibt es einen Link, um nach oben zu springen:

```
<p id="top"><strong>Ganz oben</strong> Lorem ...</p>
<!-- Weiterer Inhalt gekürzt -->
<a href="#top">smooth nach oben</a>
```

Das sanfte Scrollen brauchen Sie jetzt nur noch mit Hilfe der folgenden CSS-Notierung auszulösen:

```
body {
  scroll-behavior: smooth;
}
```

Die Eigenschaft *scroll-behavior* wirkt sich lediglich bei der Navigation oder bei CSSOM-Scrolling-APIs aus – das heißt, das normale Scrollen durch den Benutzer funktioniert weiter wie gehabt.

Wie schnell aber wird dann automatisch gescrollt? Das soll durch den Browser definiert sein und im Zweifelsfall dem entsprechen, was bei dem Betriebssystem üblich ist.

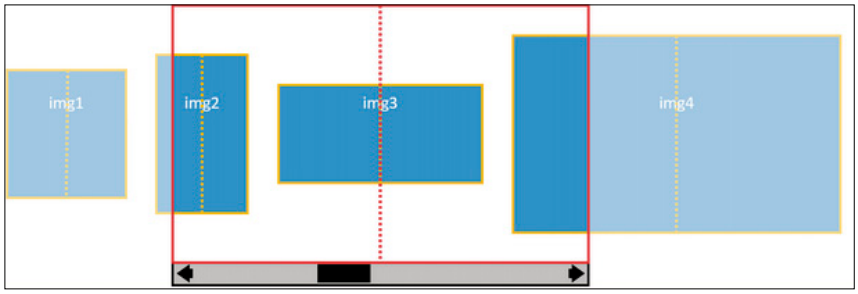
Progressive Enhancement

Die recht praktische Eigenschaft *scroll-behavior* funktioniert bereits in Firefox; außerdem lässt sie sich in Chrome aktivieren, wenn die Einstellung *Enable experimental web platform features* bei *chrome://flags* gewählt ist. Bis die Unterstützung besser ist, kann man *scroll-behavior* schon im Sinne eines Progressive Enhancement einsetzen.

Vorweg: Die nächste vorgestellte CSS-Angabe *-webkit-overflow-scrolling* ist keine Standardeigenschaft, sondern eine proprietäre, nur von Apple definierte Angabe; sie ist jedoch trotzdem sehr praktisch.

Nach einem Wischen etwa auf einem iPad bewegt sich die Webseite, bis die Bewegung langsam abklingt – das ist das Standardverhalten. Dies gilt jedoch üblicherweise nicht bei Bereichen, die über die Eigenschaft *overflow* oder Ähnliches scrollbar gemacht wurden. Dank *-webkit-overflow-scrol-* ▶

ling: touch lässt sich das ändern: Damit aktivieren Sie auf iOS-Geräten auch für diese Bereiche das intuitive native Scrollen.



Das Beispiel aus dem W3C-Entwurf zu den Snap Points demonstriert die Auswirkung einer zentrierten Anordnung am Ende des Scrollens (Bild 5)

CSS Scroll Snap Points

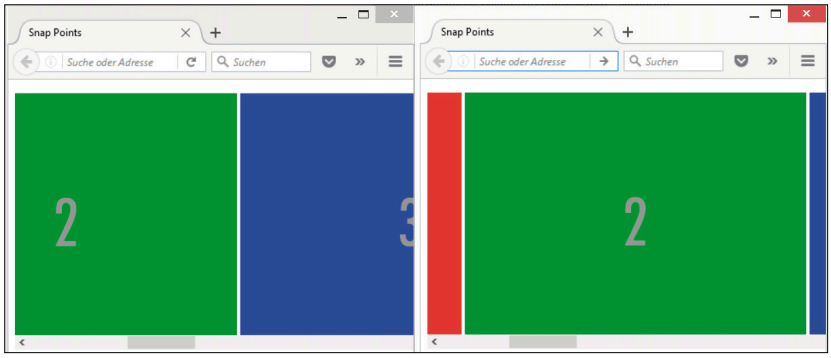
Kommen wir noch einmal zurück zu dem Karussell-Beispiel per *overflow-x*. Hier wäre es sinnvoll, dass am Ende einer beispielsweise auf einem Touchscreen durch das Wischen ausgelösten Bewegung nicht gerade zwei Bilder halb zu sehen sind, sondern jeweils ein Bild ganz und zentriert. Genau an dieser Stelle kommen die CSS Scroll Snap Points ins Spiel. Darüber können Sie genauer festlegen, wie weit bei einem Scrollen die Weiterbewegung erfolgt, und können Einrastpunkte definieren (Bild 5).

Die W3C-Spezifikation zu den Snap Points ist allerdings noch nicht endgültig; derzeit gibt es eine alte und eine neue Version. Das heißt, dass Ihnen im Web oder an anderer Stelle Beschreibungen zur alten Version begegnen können – und auch das Polyfill, das es für Snap Points gibt, verwendet derzeit noch die alten Eigenschaften. Interessant ist es aber trotzdem, sich heute schon mit Snap Points zu beschäftigen, um zu sehen, in welche Richtung die Entwicklung geht.

Erst einmal brauchen Sie die Eigenschaft *scroll-snap-type:* Diese Eigenschaft wird beim umfassenden Element spezifiziert, also beim Scrollcontainer. Hier können Sie zwei Werte angeben, die bestimmen, wie die Scroll Snap Points berücksichtigt werden sollen:

- *mandatory:* Obligatorisch. Auch wenn sich die Inhalte ändern, weil zum Beispiel neue ergänzt oder bestehende geändert werden, muss der Snap Point berücksichtigt werden.
- *proximity:* In der Nähe, das heißt, weniger streng. Die genaue Position bestimmt der User Agent anhand der Art des Scrollens, und es ist auch freigestellt, was passiert, wenn neue Inhalte hinzugefügt oder andere Änderungen durchgeführt werden.

Außerdem können Sie über *scroll-snap-padding* Innenabstände beim Scrollcontainer definieren. Weiterhin bestimmen Sie mit *scroll-snap-align* beim Element selbst, wo es sich befinden soll, wenn der Scrollvorgang stoppt. Mögliche Werte sind *start, end* oder *center*. Hier können Sie zwei Werte für die beiden Achsen (x und y) angeben. Schließlich lassen sich mit *scroll-snap-margin* Abstände beim Element definieren, die bei der Berechnung des Einrastpunkts berücksichtigt werden.



Scrollleiste: Wenn man zwischen zwei Bildern zu scrollen aufhört (links), bewegt der Browser die Scrollleiste so, dass das Bild im Zentrum ist (rechts) (Bild 6)

Karussell mit Snap Points

Ein Beispiel, bei dem der Einsatz von Snap Points sinnvoll sein könnte, ist das bereits in Listing 1 und Listing 2 erstellte Karussell. Sehen wir uns an, wie man das Karussell mit Snap Points verbessern kann. Dazu benutzen wir die alte Version der Snap-Points-Eigenschaften, weil diese teilweise schon von den Browsern unterstützt werden und es außerdem für die alte Version der Spezifikation ein Polyfill gibt. Die Angaben für die Snap Points werden beim Elternelement, also beim Scrollcontainer, geschrieben:

```
ol.karussell {
  scroll-snap-type: mandatory;
  scroll-snap-destination: 0% 400px;
  scroll-snap-points-x: repeat(400px);
}
```

Durch diesen Code wird ein Einrasten alle 400 px in der Horizontalen definiert. Für eine größtmögliche Browserunterstützung sollten Sie die Angaben zusätzlich mit den Präfixen *–ms-* und *–webkit-* schreiben. Wenn Sie das Beispiel im Firefox testen und den Scrollbalken bewegen, können Sie feststellen, dass beim Loslassen des Scrollbalkens immer zum nächstgelegenen Bildanfang gesprungen wird, sodass das Bild vollständig zu sehen ist (Bild 6).

Im Beispiel sind die ursprünglichen Eigenschaften eingesetzt, bei den neu diskutierten Angaben müsste man es umschreiben. Ein entscheidender Unterschied ist, dass etwa im neueren W3C-Entwurf die Position des Einrastpunkts nicht beim übergeordneten Element, sondern beim Kindelement angegeben wird:

```
ol.karussell {
  scroll-snap-type: mandatory;
```

```
}
.karussell li {
  scroll-snap-align: center none;
}
```

Snap Points lassen sich heute nur mit einem Polyfill einsetzen – es gibt noch keine Unterstützung für Chrome. Außerdem sollten Sie bereit sein, bei Bedarf den Code anzupassen, da die Browser derzeit nur die veraltete Version implementiert haben und die neuen Eigenschaften noch in der Diskussion sind.

Wenn man sich mit den Snap Points und ihren Einsatzmöglichkeiten beschäftigt, wird aber auch klar, dass man damit tief in die normale Benutzerinteraktion eingreifen kann – worüber der Benutzer nicht unbedingt begeistert ist. Das erklärt die teils sehr harsche Kritik in den Kommentaren zu einem Artikel bei CSS Tricks, in dem die Möglichkeiten der Snap Points vorgestellt werden. In diesen Kommentaren wird teilweise angemahnt, dass Webprogrammierer doch bitte das Scrollverhalten in Ruhe lassen sollen.

Custom-Scrollbar-Plug-in: Sieht schick aus, sollte aber vor dem Einsatz auf Benutzerfreundlichkeit getestet werden (Bild 7)

Wenn man das Scrollen beeinflusst, muss man darauf achten, nicht die normale Funktionalität auszuhebeln – vor allem auf Smartphones und Touchscreens kann es passieren, dass Seiten dann nicht mehr gut bedienbar sind. Das gilt besonders für weitergehende Eingriffsmöglichkeiten auf das Scrollverhalten, wie sie mit JavaScript möglich sind.

Auf den ersten Blick sehr faszinierend sind etwa die Effekte, die durch Plug-ins wie jQuery custom scrollbars (Bild 7) möglich sind. Mit diesem Plug-in können Sie nicht nur die Optik der Scrollleisten anpassen, sondern auch die Geschwindigkeit des Scrollens, und so können Sie etwa eine Verzögerung integrieren. Es ist aber fraglich, ob das bei den Benutzern immer gut ankommt – weil ihnen dadurch die normale Steuerungsmöglichkeit genommen wird.

Ein misslungenes Eingreifen in die Scrollmechanismen bezeichnet man auch als Scroll Hijacking. Allgemein versteht man darunter, dass die normale Scrollmöglichkeit zu anderen Zwecken missbraucht wird. Trent Walton vermittelt seine Erfahrungen damit sehr drastisch: »Anstatt die Seite zu benutzen, machte ich mir Sorgen, dass mein Trackpad kaputt sei, und darüber hinaus nervten mich die drei Sekunden Verzögerung, wenn ich einzelne Bereiche näher ansehen wollte.«

Dass etwas technisch möglich ist, heißt nicht, dass es immer sinnvoll ist. Prinzipiell ist es aber zu begrüßen, wenn es erweiterte Möglichkeiten gibt, das Scrollverhalten mit CSS zu steuern – und wenn man dafür nicht immer auf JavaScript zurückgreifen muss. Denn bei JavaScript besteht die zusätzliche Gefahr, dass dadurch die Performance leidet. Faszinierend ist auf jeden Fall, was heute alles mit CSS möglich ist – und da zeigen die Snap Points besonders, in welche Richtung die weitere Entwicklung geht. ■

Links zum Thema

- background-attachment:
  http://meyerweb.com/eric/css/edge/complexspiral/glassy.html
  http://lea.verou.me/2012/04/background-attachment-local
  http://people.opera.com/pepelsbey/experiments/bga
- position: fixed in mobilen Geräten
  http://bradfrost.com/blog/mobile/fixed-position
- CSS Overflow Module Level 3
  https://drafts.csswg.org/css-overflow-3
- Responsive Tabelle mit overflow-x
  http://dbushell.com/2016/03/04/css-only-responsive-tables
- position: sticky
  https://github.com/filamentgroup/fixed-sticky
  https://github.com/wilddeer/stickyfill
  http://imakewebthings.com/waypoints/shortcuts/sticky-elements
  https://bugs.chromium.org/p/chromium/issues/detail?id=389638
- Scroll Snap Points
  https://drafts.csswg.org/css-snappoints
  https://css-tricks.com/introducing-css-scroll-snap-points
  https://github.com/ckrack/scrollsnap-polyfill
- Scroll Hijacking
  http://trentwalton.com/2013/10/23/scroll-hijacking
- jQuery custom content scroller
  http://manos.malihu.gr/jquery-custom-content-scroller

Dr. Florence Maurice
ist Autorin, Trainerin und Programmiererin in München. Sie schreibt Bücher zu PHP und CSS3 und gibt Trainings per Video. Außerdem bloggt sie zu Webthemen unter:
http://maurice-web.de/blog

GEOLOCATION API UND GOOGLE MAPS API

Positionsermittlung

Das Geolocation API und das Google Maps API sind zwei APIs zur Positionsermittlung.

Mit Hilfe des Geolocation API lassen sich Standortinformationen des Nutzers ermitteln. Diese Informationen können anschließend beispielsweise dazu verwendet werden, um über das Google Maps API den Standort des Nutzers auf einer Karte anzuzeigen.

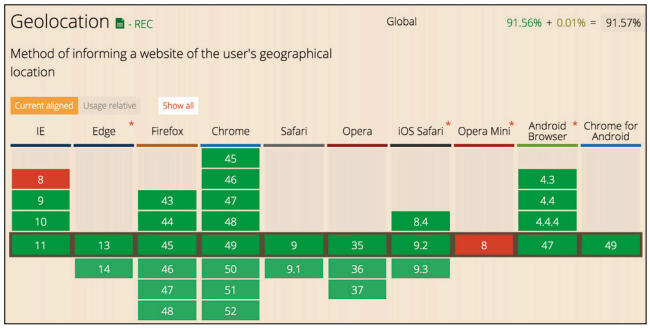
Ob das Geolocation API vom jeweiligen Browser unterstützt wird, kann bestimmt werden, indem man prüft, ob das navigator-Objekt die Eigenschaft *location* enthält. Momentan dürfte dies für alle namhaften Browser der Fall sein (Bild 1):

```
if ('geolocation' in navigator)
{
  // Geolocation API wird unterstützt
} else {
  // Geolocation API wird nicht unterstützt
}
```

Das Google Maps API dagegen ist – wie der Name bereits andeutet – ein proprietäres API vom Suchmaschinenriesen Google und muss folglich separat als JavaScript-Datei eingebunden werden.

Aktuelle Position des Nutzers

Um die aktuelle Position des Nutzers zu ermitteln, stellt das geolocation-Objekt (vom Typ Geolocation) die Methode *getCurrentPosition()* zur Verfügung. Da die Ermittlung der Position asynchron abläuft, übergibt man dieser Methode als erstes Argument eine Funktion als Callback-Handler, die wiederum dann aufgerufen wird, sobald die Position ermittelt wurde.



Das Geolocation API wird von den meisten aktuellen Browsern unterstützt (Bild 1)

Ermittlung der Position in Chrome (Bild 2)

Innerhalb des Callback-Handlers kann anschließend über den Parameter *position* auf verschiedene standortbezogene Informationen zugegriffen werden (Bild 2), beispielsweise auf Informationen bezüglich Längengrad *(longitude)* und Breitengrad *(latitude)*, Genauigkeit der Messung *(accuracy)* sowie – für den Fall, dass sich der Nutzer in eine bestimmte Richtung bewegt – Richtung *(heading)* und Geschwindigkeit *(speed)* (Listing 1).

Zu beachten ist Folgendes: Damit die Position des Nutzers und entsprechende Standortinformationen überhaupt ermittelt werden können, muss der Nutzer über einen entsprechenden Hinweisdialog seine Zustimmung geben. Für den Fall, dass keine Zustimmung erteilt wurde, kann dies über einen zweiten Callback-Handler der Methode *getCurrentPosition()* abgefangen werden (Listing 2).

Neben der Methode *getCurrentPosition()* dient die Methode *watchPosition()* dazu, die Position des Nutzers über einen längeren Zeitraum zu beobachten. Bezüglich der Parameter

Listing 1: Positionsermittlung mit dem Geolocation API

```
function getPosition() {
  if(navigator.geolocation) {
    navigator.geolocation.getCurrentPosition
    ((position) => {
      document.getElementById('latitude').value =
      position.coords.latitude;
      document.getElementById('longitude').value =
      position.coords.longitude;
      document.getElementById('accuracy').value =
      position.coords.accuracy;
      document.getElementById('altitudeAccuracy').
      value = position.coords.altitudeAccuracy;
      document.getElementById('heading').value =
      position.coords.heading;
      document.getElementById('speed').value =
      position.coords.speed;
      document.getElementById('timestamp').value =
      position.timestamp;
    });
  }
}
```

Listing 2: Fehlerbehandlung

```
function getPosition() {
  function successHandler(position) {
    document.getElementById('latitude').value =
    position.coords.latitude;
    document.getElementById('longitude').value =
    position.coords.longitude;
    document.getElementById('accuracy').value =
    position.coords.accuracy;
    document.getElementById('altitudeAccuracy').value
    = position.coords.altitudeAccuracy;
    document.getElementById('heading').value =
    position.coords.heading;
    document.getElementById('speed').value =
    position.coords.speed;
    document.getElementById('timestamp').value =
    position.timestamp;
  }
  function errorHandler(error) {
    switch(error.code) {
      case error.PERMISSION_DENIED:
        alert('Geolokalisierung durch Nutzer
          nicht erlaubt.');
          break;
        case error.POSITION_UNAVAILABLE:
          alert('Keine Lokalisierunginformationen
          verfügbar.');
          break;
        case error.TIMEOUT:
          alert('Timeout für die Lokalisierungsanfrage
          wurde überschritten.');
          break;
        default:
          alert('Unbekannter Fehler (#' + error.code +
          ': ' + error.message + ')');
          break;
      }
    }
    if(navigator.geolocation) {
      navigator.geolocation.getCurrentPosition
      (successHandler, errorHandler);
    }
}
```

verhält sich diese Methode gleich wie *getCurrentPosition()*, ruft aber im Gegensatz dazu den Callback-Handler mehrfach auf. Als Rückgabewert liefert die Methode zudem eine ID, mit Hilfe derer das Beobachten der Nutzerposition über die Methode *clearWatch()* zu einem späteren Zeitpunkt wieder abgebrochen werden kann.

Position auf der Karte anzeigen

Richtig interessant wird das Geolocation API erst, wenn man die ermittelten Informationen beispielsweise mit dem Google Maps API kombiniert.

Bevor das Google Maps API verwendet werden kann, muss allerdings, wie in Listing 3 zu sehen, die entsprechende JavaScript-Datei eingebunden werden.

Ein Codebeispiel, das das Geolocation API und das Google Maps API kombiniert, zeigt Listing 4. Das Ergebnis des Programms ist in Bild 3 zu sehen. Der Typ *Map* repräsentiert hierbei eine Karte, die auf Basis eines <*canvas*>-Elements erzeugt wird. Der Methode *setCenter()* können Längen- und Breitengrad in Form eines *LatLng*-Objekts übergeben werden, woraufhin die Karte sich genau an dieser Position zentriert. Über *InfoWindow* lassen sich zudem Hinweisfenster innerhalb der Karte erzeugen.

Das Google Maps API ist sehr mächtig und man kann als Entwickler und Gestalter von Webseiten viele nützliche Dinge damit realisieren.

Ein weiterer interessanter Anwendungsfall ist dabei das Anzeigen einer Anfahrtsbeschreibung, sowohl textuell neben der Karte als auch als Verbindungslinie innerhalb der Karte. Ein konkretes Anwendungsbeispiel für ein solches Szenario zeigt Listing 5. ▶

Listing 3: Einbinden des Google Maps API

```
<!DOCTYPE html>
<html>
  <head lang="en">
    <meta charset="UTF-8">
    <title>Google Maps API</title>
    <link rel="stylesheet" href="styles/main.css"
    type="text/css">
  </head>
  <body>
    <div id="map-canvas"></div>
    <script src="https://maps.googleapis.com/maps/
    api/js?v=3.exp&signed_in=true"></script>
    <script src="scripts/main.js"></script>
  </body>
</html>
```

Darstellung der Position mit Google Maps (Bild 3)

Listing 4: Position des Nutzers auf einer Karte

```
function getPosition() {
  function successHandler(position) {
    /* Hier der gleiche Code wie eben zum Befüllen
    der Eingabefelder */
    let mapOptions = {
      zoom: 6
    };
    let map = new google.maps.Map
    (document.getElementById ('map-canvas'),
    mapOptions);
    let googlePosition = new google.maps.LatLng(
      position.coords.latitude,
      position.coords.longitude
    );
    let infowindow =
      new google.maps.InfoWindow({
        map: map,
        position: googlePosition,
        content: 'Sie befinden sich hier.'
      });
      map.setCenter(googlePosition);
    }
    function errorHandler(error) {
      // Nutzer verweigert Positionsermittlung
    }
    if(navigator.geolocation) {
      navigator.geolocation.getCurrentPosition
      (successHandler, errorHandler);
    }
}
```

Listing 5: Anfahrtsbeschreibung

```
function getPosition() {
  function successHandler(position) {
    /* Hier der gleiche Code wie eben zum
    Befüllen der Eingabefelder */
    let directionsService = new
    google.maps.DirectionsService();
    let directionsRenderer = new
    google.maps.DirectionsRenderer();
    let mapOptions = {
      zoom: 6
    };
    let map = new google.maps.Map
    (document.getElementById('map-canvas'),
    mapOptions);
    let travel = {
      origin : new google.maps.LatLng
      (position.coords.latitude,
      position.coords.longitude),
      destination : "Alexanderplatz, Berlin",
      travelMode :
      google.maps.DirectionsTravelMode.DRIVING
    };
    let googlePosition =
      new google.maps.LatLng
      (position.coords.latitude,
      position.coords.longitude);
      map.setCenter(googlePosition);
      directionsRenderer.setMap(map);
      directionsRenderer.setPanel
      (document.getElementById(„map-directions“));
      directionsService.route(travel, function(result,
      status) {
        if (status === google.maps.DirectionsStatus.OK)
        {
          directionsRenderer.setDirections(result);
        }
      });
    }
    function errorHandler(error) {
      // Nutzer verweigert Positionsermittlung
    }
    if(navigator.geolocation) {
      navigator.geolocation.getCurrentPosition
      (successHandler, errorHandler);
    }
}
```

Fazit

Über das Geolocation API lässt sich die Position des Nutzers ermitteln. Die so erhaltenen Informationen lassen sich im Anschluss daran relativ einfach mit dem Google Maps API kombinieren, um beispielsweise den Standort auf einer Karte anzuzeigen oder Anfahrtsbeschreibungen ausgehend vom aktuellen Standort zu ermitteln. Dies kann die Benutzerfreundlichkeit beispielsweise einer E-Commerce-Seite mit direktem Kundenkontakt nachhaltig verbessern. ■

Philip Ackermann
arbeitet beim Fraunhofer-Institut für Angewandte Informationstechnologie FIT an Tools zum teilautomatisierten Testen von Web Compliance und ist Autor zweier Fachbücher über Java und JavaScript.
http://philipackermann.de

developer media

developer media Trainings

Updates für Ihr Know-how

„Die Faszination der Softwareentwicklung liegt darin begründet, dass nur die eigene Fantasie und Kreativität begrenzen, was möglich ist."

Golo Roden
Technologischer Visionär, Sprecher, Autor

Angular 2 mit TypeScript

Trainer: Johannes Hoppe, Gregor Woiwode

3 Tage, 23.-25.05.2016, Köln

Frühbucher: 2.199 EUR zzgl. MwSt.
Normalpreis: 2.399 EUR zzgl. MwSt.

Codequalität mit JavaScript

Trainer: Golo Roden

3 Tage, Termin & Ort nach Absprache

2.399 EUR zzgl. MwSt.

Agile Produktentwicklung

Trainer: Björn Schotte

2 Tage, Termin & Ort nach Absprache

1.999 EUR zzgl. MwSt.

PHPUnit erfolgreich einsetzen

Trainer: Sebastian Bergmann

2 Tage, Köln, Termin nach Absprache

1.999 EUR zzgl. MwSt.

Ihr Ansprechpartner:
Fernando Schneider – Key Account Manager – developer media
Telefon: +49 (0)89 74117-831 – E-Mail: fernando.schneider@developer-media.de

developer-media.de/trainings

PROCESS MANAGEMENT UNTER NODE.JS MIT PM2

Prozess-Manager

In Produktionsumgebungen können Process Manager das Steuern einzelner Prozesse beziehungsweise Anwendungen erleichtern.

Node.js bringt von Haus aus zwar keinen Process Manager mit, das Modul PM2 schließt jedoch diese Lücke.

Bei PM2 (http://pm2.keymetrics.io) handelt es sich um einen Process Manager für Node.js, über den sich Anwendungen starten, stoppen und neustarten lassen. Darüber hinaus verfügt PM2 über eine Reihe weiterer Features – unter anderem sind dies:

- Monitoring von Anwendungen,
- Clustering von Anwendungen,
- Load Balancing,
- Verwalten von Anwendungslogs,
- automatisches Neustarten von Anwendungen,
- programmatischer Zugriff per API.

PM2 kann nicht nur unter allen namhaften Betriebssystemen eingesetzt werden, darüber hinaus ist auch ein Einsatz auf Cloud-Plattformen wie Heroku, Google App Engine oder Azure möglich.

Um PM2 verwenden zu können, muss es zunächst, wie für Node.js-Module gewohnt, über den Node.js Package Manager NPM mit dem Befehl *npm install -g pm2* global installiert werden.

Tabelle 1: Übersicht über verschiedene PM2-Befehle

Befehl	Beschreibung
Fork-Modus	
pm2 start <app-name>	Starten einer Anwendung
pm2 start <app-name> --name <custom-name>	Starten einer Anwendung mit individuellem Namen
pm2 start <app-name> --watch	Starten einer Anwendung und Neustart bei Änderungen am Quelltext
Cluster-Modus	
pm2 start <app-name> -i 0	Starten von Anwendungen basierend auf der Anzahl zur Verfügung stehender CPUs
pm2 start <app-name> -i <number-of-instances>	Starten einer bestimmten Menge von Instanzen einer Anwendung
Übersicht und Monitoring	
pm2 list	Listet alle Prozesse inklusive des jeweiligen Status auf
pm2 jlist	Wie *pm2 list*, nur dass die Ergebnisse im JSON-Format ausgegeben werden
pm2 prettylist	Wie *pm2 jlist*, nur dass die Ergebnisse in formatiertem JSON ausgegeben werden
pm2 show <app-id>	Zeigt alle Informationen über einen bestimmten Prozess an
pm2 describe <app-id>	
pm2 monit	Startet das Monitoring aller Prozesse

Befehl	Beschreibung
Logs	
pm2 logs	Gibt die Logs aller Prozesse aus
pm2 flush	Leert alle Logs
pm2 reloadLogs	Lädt alle Logs neu
Weitere Aktionen	
pm2 stop all	Stoppt alle Prozesse
pm2 restart all	Startet alle Prozesse neu
pm2 reload all	Lädt alle Prozesse neu
pm2 gracefulReload all	Sendet zunächst eine Exit-Nachricht an die Prozesse und lädt diese erst dann neu
pm2 stop <app-id>	Stoppt einen bestimmten Prozess
pm2 restart <app-id>	Startet einen bestimmten Prozess neu
pm2 delete <app-id>	Entfernt einen bestimmten Prozess
pm2 delete all	Entfernt alle Prozesse
pm2 startup <platform>	Generieren eines Start-up-Skripts
Entwicklungsmodus	
pm2-dev run <app-name>	Starten einer Anwendung im Entwicklungsmodus
Module	
pm2 install <module-name>	Installiert oder aktualisiert ein Modul
pm2 uninstall <module-name>	Deinstalliert ein Modul
pm2 publish	Veröffentlicht ein neues Modul

Anschließend steht der Befehl *pm2* auf Kommandozeile zur Verfügung (Tabelle 1).

Nehmen wir als Beispiel nun eine einfache Anwendung, wie sie in Listing 1 gezeigt ist. Mit Hilfe des Standardmoduls *http* wird hier ein Webserver gestartet, der auf Port 8080 läuft und alle Anfragen mit der Zeichenkette *Hallo Welt* beantwortet. Ohne PM2 – nämlich klassisch direkt über Node.js – würde man diese Anwendung über den Befehl *node app.js* starten (vorausgesetzt, der Quelltext befindet sich in der Datei *app.js)*.

Um die Anwendung als Prozess über PM2 zu starten, verwendet man dagegen den Befehl *pm2 start*. Diesem Befehl übergibt man als Parameter den Pfad zu der Skriptdatei, die gestartet werden soll. Im vorliegenden Fall lautet der Befehl also *pm2 start app.js.*

Nach erfolgreichem Start gibt PM2 eine entsprechende Bestätigung aus (Bild 1). Man sieht: PM2 weist jeder Anwendung

```
[PM2] Spawning PM2 daemon
[PM2] PM2 Successfully daemonized
[PM2] Starting app.js in fork_mode (1 instance)
[PM2] Done.
```

App name	id	mode	pid	status	restart	uptime	memory	watching
app	0	fork	1178	online	0	0s	10.836 MB	disabled

Use `pm2 show <id|name>` to get more details about an app

Starten einer Anwendung (Bild 1)

unter anderem automatisch eine interne ID zu, über die sich im Folgenden die Anwendung ansprechen lässt, um sie beispielsweise wieder zu stoppen.

Optional lässt sich beim Starten einer Anwendung auch ein Name angeben, unter dem die Anwendung unter PM2 verwaltet wird, und zwar über das Flag *--name*. Gibt man an dieser Stelle keinen Namen an, bekommt die Anwendung stan-

Listing 1: Starten eines Webservers unter Node.js

```
'use strict';
let http = require('http');
let server = http.createServer(
  (request, response) => {
    response.writeHead(
      200,
      {
        'Content-Type': 'text/plain'
      }
    );
    response.end('Hallo Welt\n');
  }
);
server.listen(8080);
console.log('Server gestartet unter
http://localhost:8080/');
```

```
[PM2] Stopping app.js
[PM2] stopProcessId process id 0
```

App name	id	mode	pid	status	restart	uptime	memory	watching
app	0	fork	0	stopped	0	0	0 B	disabled

Use `pm2 show <id|name>` to get more details about an app

Stoppen einer Anwendung (Bild 2)

dardmäßig den Namen der Skriptdatei, die dem Befehl *pm2 start* übergeben wurde:

```
pm2 start app.js --name example-application
```

Standardmäßig werden alle Anwendungen im sogenannten Fork-Modus gestartet, das heißt, es wird nur eine einzelne Instanz der jeweiligen Anwendung erzeugt. Alternativ dazu lassen sich aber über den sogenannten Cluster-Modus auch direkt mehrere Instanzen einer Anwendung erzeugen, um dementsprechend eine höhere Ausfallsicherheit zu erreichen.

Weitere Aktionen

Analog zum Starten von Anwendungen steht für deren Stoppen der Befehl *pm2 stop* zur Verfügung. Als Parameter übergibt man diesem Befehl entweder die zuvor erwähnte ID, die PM2 der Anwendung vergeben hat, oder den Namen der Anwendung, beispielsweise *pm2 stop 0* oder *pm2 stop app.js*. Die Ausführung der Anwendung wird dadurch abgebrochen, aber weiterhin von PM2 verwaltet (Bild 2). Möchte man dagegen erreichen, dass die Anwendung nicht nur gestoppt, sondern auch nicht mehr von PM2 verwaltet wird, verwendet man den Befehl *pm2 delete*.

Um eine Anwendung neu zu starten, verwendet man den Befehl *pm2 restart*. Auch hier übergibt man als Parameter die ID oder den Namen der neuzustartenden Anwendung.

Detaillierte Informationen zu dem Status einer Anwendung lassen sich über die Befehle *pm2 show* und *pm2 describe* ermitteln (Bild 3), eine Übersicht über den Status aller Anwendungen über den Befehl *pm2 list* (Bild 4). Um sich die Logs anzeigen zu lassen, kann man den Befehl *pm2 logs* verwenden. Anschließend werden die Konsolenausgaben aller An- ▶

```
Describing process with id 0 - name app
```

status	online
name	app
node.js version	4.1.0
id	0
path	/Users/philipackermann/Documents/Synchronisierung/Entwicklung/Codebeispiele/NodeJS/PM2/app.js
args	N/A
exec cwd	/Users/philipackermann/Documents/Synchronisierung/Entwicklung/Codebeispiele/NodeJS/PM2
error log path	/Users/philipackermann/.pm2/logs/app-error-0.log
out log path	/Users/philipackermann/.pm2/logs/app-out-0.log
pid path	/Users/philipackermann/.pm2/pids/app-0.pid
mode	fork_mode
node v8 arguments	N/A
watch & reload	✘
interpreter	node
restarts	0
unstable restarts	0
uptime	16s
created at	2016-03-27T16:40:46.782Z

```
Process configuration
```

```
Probes value
```

Loop delay	1.49ms

Anwendungsdetails: So erfolgt die detaillierte Anzeige (Bild 3)

Tabelle 2: Übersicht über das programmatische API von PM2

Befehl	Beschreibung
Verbinden mit PM2 herstellen	*pm2.connect(function(error) {})*
Verbindung mit PM2 trennen	*pm2.disconnect(function(error, process) {})*
Starten einer Anwendung	*pm2.start(script_path | json_object | json_path, options, function(error, proc) {})*
Neustart einer Anwendung oder aller Anwendungen	*pm2.restart(proc_name | proc_id | all, function(error, process) {})*
Stoppen einer Anwendung oder aller Anwendungen	*pm2.stop(proc_name | proc_id | all, function(error, process) {})*
Löschen einer Anwendung oder aller Anwendungen	*pm2.delete(proc_name | proc_id | all, function(error, process) {})*
Liste aller Anwendungen	*pm2.list(function(error, list) {})*
Detailinformationen zur Anwendung	*pm2.describe(proc_name | proc_id, function(error, list) {})*

Befehl	Beschreibung
Neuladen einer Anwendung oder aller Anwendungen	*pm2.reload(proc_name | all, function(error, process) {})*
Neuladen einer Anwendung oder aller Anwendungen , nachdem Exit-Nachricht gesendet wurde.	*pm2.gracefulReload(proc_name | all, fn(error, process) {})*
Erstellen eines Dumps	*pm2.dump(function(error, ret) {})*
Löschen der Logs	*pm2.flush(function(error, ret) {})*
Neuladen der Logs	*pm2.reloadLogs(function(error, ret) {})*
Signal an Anwendung senden	*pm2.sendSignalToProcessName(signal,proc, function(error, ret) {})*
Generieren eines Start-Skripts	*pm2.startup(platform, function(error, ret) {})*
Beenden von PM2	*pm2.killDaemon(fn(error, ret) {})*

wendungen (entsprechend unterschiedlich gekennzeichnet) in Echtzeit auf die Konsole ausgegeben.

Weitere Features

Ein besonders während der Entwicklung einer Anwendung interessantes Feature ist der automatische Neustart bei Änderungen am Quelltext. Dazu verwendet man den Befehl *pm2-dev run* und startet dadurch die jeweilige Anwendung im Entwicklungsmodus: *pm2-dev run app.js*. Ändert man nun etwas an der Quelltextdatei, so wird die Anwendung von PM2 automatisch neu geladen. Ähnliches lässt sich erreichen, indem man dem Befehl *pm2 start* das Flag *--watch* übergibt. Möchte man, dass Anwendungen auch dann neu gestartet (beziehungsweise in ihren jeweiligen Zustand zurückversetzt) werden, wenn der Server, auf dem PM2 läuft, neu gestartet wird, lässt sich über den Befehl *pm2 startup* ein Startup-Skript für das jeweilige Betriebssystem erstellen.

Verwenden des API

PM2 kann nicht nur über die Konsole genutzt werden, sondern auch programmatisch per JavaScript. Dazu muss zunächst, wie in Listing 2 zu sehen, per *require('pm2')* das Modul eingebunden werden (vorher sollte natürlich *pm2* als Abhängigkeit in die jeweilige *package.json*-Datei hinzugefügt

App name	id	mode	pid	status	restart	uptime	memory	watching
app	0	fork	1509	online	0	2h	7.254 MB	disabled
app2	1	fork	2125	online	0	12s	23.836 MB	disabled
app3	2	fork	2133	online	0	5s	23.902 MB	disabled
app4	3	fork	2138	online	0	3s	23.680 MB	disabled
app5	4	fork	2144	online	0	0s	9.617 MB	disabled

Aufstellung: Die Liste aller Anwendungen (Bild 4)

```
> PM2 monitoring (To go further check out https://app.keymetrics.io)

● app                                                    [                ] 0 %
[0] [fork_mode]                                          [                ] 8.238 MB

● app2                                                   [                ] 0 %
[1] [fork_mode]                                          [                ] 23.754 MB

● app3                                                   [                ] 0 %
[2] [fork_mode]                                          [                ] 23.809 MB

● app4                                                   [                ] 0 %
[3] [fork_mode]                                          [                ] 23.813 MB

● app5                                                   [                ] 0 %
[4] [fork_mode]                                          [                ] 23.809 MB
```

Monitoring aller Anwendungen (Bild 5)

Listing 2: Verwendung des PM2-API

```
'use strict';
let pm2 = require('pm2');
pm2.connect((error) => {
  if (error) {
    console.error(error);
    process.exit(2);
  }
    pm2.start({
    script    : 'app.js',   // Zu startendes Skript
    exec_mode : 'cluster',  // Cluster-Modus an
    instances : 4,          // Anzahl an Instanzen
  }, (error, apps) =>
{
    pm2.disconnect();
  });
});
```

werden). Im Wesentlichen stehen über das API die eben vorgestellten PM2-Befehle zur Verfügung (Tabelle 2). Um die entsprechenden Methoden zu nutzen, muss allerdings durch einen Aufruf von *connect()* zunächst eine Verbindung zu PM2 hergestellt werden.

In der entsprechenden Callback-Funktion, die aufgerufen wird, sobald die Verbindung hergestellt wurde, wird in Listing 2 beispielsweise über *start()* die Anwendung *app.js* im Cluster-Modus mit vier Instanzen gestartet. Nachdem der Befehl ausgeführt wurde, ist es sinnvoll, die Verbindung von PM2 über *disconnect()* wieder zu trennen.

Fazit

PM2 erleichtert das Management von Anwendungen. Es ist hinsichtlich seiner Features wie Monitoring, Clustering, Load Balancing et cetera gegenüber Tools wie Forever (https://github.com/foreverjs/forever) um einiges umfangreicher (Bild 5). Durch den kommerziellen Dienst Keymetrics (https://keymetrics.io) wird PM2 zudem um eine Weboberfläche mit einem Dashboard erweitert, über das sich Anwendungen auch serverübergreifend verwalten und überwachen lassen.

Als ein ähnlich umfangreicher Process Manager ist der StrongLoop Process Manager (http://strong-pm.io) zu nennen, wobei sich ein Vergleich der verschiedenen Tools unter http://strong-pm.io/compare finden lässt. ■

Philip Ackermann
arbeitet beim Fraunhofer-Institut für Angewandte Informationstechnologie FIT an Tools zum teilautomatisierten Testen von Web Compliance und ist Autor zweier Fachbücher über Java und JavaScript.
http://philipackermann.de

FLOWTYPE – STATISCHE TYPEN IN JAVASCRIPT

Statische Typen

Seit TypeScript ist die Idee statischer Typen auch bei JavaScript nicht mehr ungewöhnlich.

Doch mit Flow hat Facebook eine Alternative geschaffen. Wie schlägt sich das neue System gegen TypeScript, und stellt es möglicherweise eine bessere Lösung dar?

Typen sind ein fundamentaler Baustein moderner Programmiersprachen. Auch JavaScript hat natürlich ein Typsystem. Jeder Wert besitzt einen Typ, und Operationen, die nicht dazu passen, lösen Laufzeitfehler aus. Damit verhält sich JavaScript genauso wie andere dynamische Programmiersprachen: Lisp, Smalltalk, Ruby, Python, Clojure – der Varianten gibt es viele.

Programmierer, die statische Sprachen wie C, Java oder C# gewohnt sind, haben oft Probleme mit der dynamischen Natur solcher Sprachen. Ein Compiler soll möglichst bereits beim Übersetzen naive Fehler erkennen: Dazu gehört auch das Übergeben eines Wertes mit falschem Typ an den Parameter einer Funktion. Derartige Typfehler sollten nach diesem Urteil so früh wie nur eben möglich erkannt werden und ein Programm im Idealfall nicht einmal vollständig übersetzbar sein.

Unterschiedliche Entwicklungsmethoden

Doch ist eine Typprüfung durch den Compiler tatsächlich der frühestmögliche Zeitpunkt? Hier zeigt sich schnell der grundsätzliche Unterschied bei den Entwicklungsmethoden zwischen dynamischen und statischen Programmiersprachen: Ein erfahrener Programmierer mit einer dynamisch getypten Programmiersprache hat jeden Ausdruck, der Teil seines Programms wird, bereits interaktiv ausprobiert. Implementation und Test vermischen sich in unzähligen kurzen Zyklen.

Es ist in Lisp oder Smalltalk üblich, dass neue Funktionen im laufenden Programm Baustein für Baustein ausprobiert werden und auf diese Weise organisch zu einem funktionierenden Ganzen kombiniert werden. Anders ausgedrückt: Was für den statischen Programmierer das Programm ist, das ist für den Lisp-Programmierer die Funktion. Diese wird in die REPL (Read-Eval-Print-Loop) eingegeben und ausprobiert, ob sie sich so verhält, wie sie soll. Eine starke Fokussierung auf funktionale Programmiermuster und pure Funktionen, die nur von ihren Parametern abhängen, tragen dazu bei, dass derartige Live-Tests oft wesentlich aussagekräftiger sind als ein paar einfache Typnotationen.

Aus diesem Grund ist es dort auch üblich, dass man jederzeit ein Image (Speicherabbild) des eigenen Programms abspeichern kann. Um ein Common-Lisp-Programm zur Auslieferung vorzubereiten, startet man die Laufzeitumgebung, lädt sämtlichen Code, den man benötigt, in das Image und speichert es in eine Datei, die dann als *.exe* aufgerufen werden kann.

Webentwicklung mit JavaScript gleicht diesem dynamischen Programmiermodell sehr stark: Der Webbrowser ist eine Laufzeitumgebung, in der eine Kommandozeile (REPL), ein Editor und diverse grafische Werkzeuge (DOM-Browser et cetera) zur Verfügung stehen. Es liegt also nahe, hier mit ähnlichen Methoden zu arbeiten, die auch schon bei Lisp oder Smalltalk funktioniert haben.

Doch auch wenn sich so die Wichtigkeit einer statischen Typisierung bei JavaScript relativiert, gänzlich verzichten muss man darauf nicht: Auch dynamische Programmiersprachen wie Common Lisp bieten an, optional statische Typen im Quelltext anzugeben. Der Compiler kann diese Information unter anderem dazu nutzen, schnelleren Maschinencode zu generieren oder eben frühzeitig auf Inkonsistenzen bei den verwendeten Typen hinzuweisen. Tatsächlich gibt es demnach nicht die Entscheidung, ob statisch oder dynamisch, sondern eher, wie viel statische Information man mit welchem Mehraufwand bei Compiler und Programmierer zulassen möchte. Denn eines muss klar sein – statische Information ist nicht kostenlos zu haben.

Statisch, dynamisch, graduell

Wenn man heutzutage über statische Typsysteme spricht, dann wird das Feld mittlerweile dominiert von Ansätzen, die fast alle auf dem sogenannten Hindley-Milner-Typsystem basieren. Vor allem funktionale Programmiersprachen wie Haskell, ML oder OCaml gehören dazu. Auch die Entwickler neuer Sprachen wie Mozillas Rust oder Apples Swift haben sich davon inspirieren lassen. In diesen Typsystemen stecken Jahrzehnte an Forschung und Entwicklung – es sind ausgereifte und erprobte Systeme. Es gibt zwar wesentlich ausdrucksstärkere Typsysteme, doch das Problem ist immer auch eine Abwägung aus effizienter Berechenbarkeit und Flexibilität. Was nützt es, wenn das Übersetzen eines Programms aufgrund der Typprüfung extrem lange dauern würde oder gar niemals endet?

Im Vergleich zu den Typsystemen, die man von C, C++, Java oder C# kennt, sind diese Typsysteme jedoch außerordentlich mächtig. Das statische Typsystem von ANSI C dient hauptsächlich dazu, dass zu jedem Datum exakt bekannt ist, wie viel Speicher es belegt. Das Typsystem lässt kaum Einschränkungen zu und kann leicht ausgehebelt werden. Das Problem: Nahezu jedes nichttriviale Programm in C erfordert es, das statische Typsystem der Sprache zu umgehen; gleichzeitig ist jedoch die unsichere Laufzeitumgebung derartig minimalistisch, dass es leicht zu Problemen kommt. Dangling Pointer, Fehler bei der (dynamischen) Speicherverwaltung – die Zahl der Herausforderungen ist groß.

Sprachen wie Java oder C# stellen hier durch ihre virtuellen Maschinen, automatische Speicherverwaltung und durch ein verbessertes Typsystem einen deutlichen Fortschritt dar. Diese Abstraktion hat aber ihren Preis: Bis heute haben sich diese Sprachen kaum als System-Entwicklungssprachen etabliert; als Anwendungsentwicklungssprachen hingegen umso mehr. Beide Sprachen wurden zu einer Zeit erfolgreich, als objektorientierte Programmierung als Allheilmittel angepriesen wurde. Ihnen fehlt zwar die flexible Dynamik der originären OOP-Sprachen wie Smalltalk, Self oder Common Lisp; dafür bieten sie mehr statische Typisierung.

Allerdings wiederum weniger flexibel als bei funktionalen Sprachen wie Haskell oder OCaml, sondern eher wie bei prozeduralen Sprachen wie Pascal. Durch die Mischung aus eingeschränktem OOP und einem primitiven statischen Typsystem befinden sich diese Sprachen auf einer stetigen Gratwanderung zwischen zugelassener Dynamik und absichernder Statik. Man kann sagen: Statische Typisierung in C# und Java dient insbesondere auch jenem Zweck, dass Entwicklungsumgebungen aus dem Quellcode möglichst viel Metainformationen ableiten können, um den Programmierer beim Arbeiten durch Features wie IntelliSense zu unterstützen.

Immer wieder taucht die Problematik auf, dass zwar statische Typprüfung gewünscht wird, jedoch auch kontrolliert umgangen werden soll. Dazu haben sich mitterweile die sogenannten graduellen Typsysteme herausgebildet. Damit sollen einerseits dynamische Typen realisiert, aber auch dynamische Typen zugelassen werden. Bei C# wurde dazu das Schlüsselwort *dynamic* eingeführt, und auch Swift bietet derartige Features, um besser mit dem dynamischen Objektsystem der Objective-C-Frameworks umgehen zu können. Doch nicht nur statische Typsysteme können so um dynamische Typen angereichert werden – auch dynamische Typsysteme können durch statische ergänzt werden. Genau dieser Aspekt findet nun bei JavaScript Anwendung.

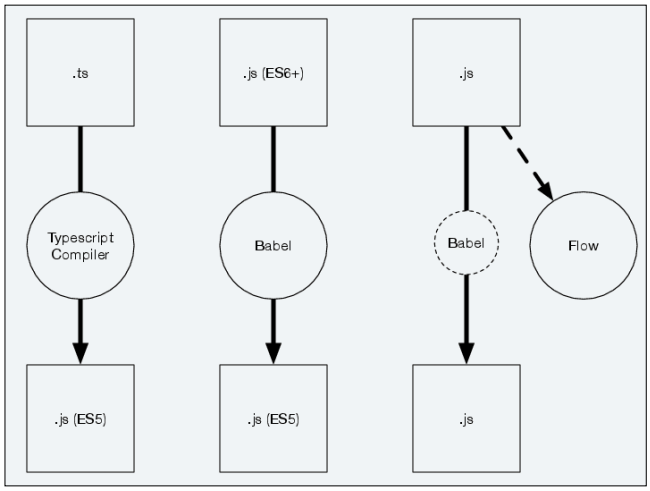
Zusammenfassend kann man vor allem drei Zwecke für die eingesetzten Typsysteme feststellen:

- **Festlegung einer Low-Level-Repräsentation der Daten:** Wie viel Speicher belegt ein Datum? Besitzt es Referenzen? Ist es veränderbar? Anhand von statischer Typinformation kann der Compiler effizienteren Code erzeugen oder Daten effizienter ablegen.
- **Dokumentation und Tool-Unterstützung:** Integrierte Entwicklungsumgebungen wie Visual Studio oder Delphi bieten dem Programmierer Hilfe per Autovervollständigung an. Welche Methoden besitzt ein Objekt? Sind alle Methoden einer Schnittstelle implementiert?
- **Absicherung und Prüfung von Merkmalen über Programme:** Manche statische Typsysteme ermöglichen es, bereits beim Übersetzen zu beweisen, dass es beispielsweise keine Fehler durch Null-Referenzen geben kann. Auch weitere Eigenschaften lassen sich so über komplexe Typen modellieren und bereits beim Übersetzen beweisen.

Eines der frühesten Projekte, um JavaScript um statische Typen zu erweitern, ist die Programmiersprache Haxe. Ursprünglich entstand sie 2003 als Projekt der Firma Motion Twin und als Compiler, der nach ActionScript übersetzt. Trotz der starken Ähnlichkeit zu JavaScript handelt es sich um eine eigenständige Programmiersprache.

Der ECMAScript-Standard selbst sah ursprünglich mit ES4 ebenfalls optionale statische Typen vor; neben Klassen, Interfaces, Packages und vielem mehr. Doch die Standardisierung von ES4 scheiterte einerseits an zu vielen Features und andererseits am mangelndem Interesse wichtiger Vertreter wie Microsoft. Das deutlich abgespeckte ES5 enthielt dann keinerlei Bestrebungen für statische Typen mehr.

Insofern stellt es wohl eine Ironie der Geschichte dar, dass gerade Microsoft, die bei ES4 so sehr bremsten, später mit TypeScript den nächsten Versuch wagten. TypeScript wird von Microsoft als Obermenge von ES2015 dargestellt. Jede JavaScript-Datei soll sich durch einfaches Umbenennen der Dateiendung auf *.ts* in eine TypeScript-Datei umwandeln lassen. In der Praxis gab es in der Vergangenheit hier und da Abweichungen, und bei ES5-Compile-Ausgabe werden nur circa 60 Prozent von ES2015 umgesetzt. Wer hauptsächlich einen ES2015-Transpiler sucht, der ist nach wie vor besser beraten, auf Babel zu setzen (http://babeljs.com). Wenn es aber um statische Typen in JavaScript geht, ist TypeScript die wohl momentan bekannteste Lösung. Ich werde es deshalb in diesem Artikel oft für Vergleiche zu Facebooks Flow nutzen.



Gegenüberstellung: Typescript, Babel und Flow im Vergleich (Bild 1)

Neue Sprache vs. Code-Prüfung

Der Ansatz von Facebooks Flow unterscheidet sich signifikant von TypeScript. Bild 1 zeigt TypeScript, Babel und Flow gegenübergestellt: Im linken Zweig sieht man, wie TypeScript-Dateien mit dem TypeScript-Compiler nach JavaScript (ES5) übersetzt werden. Der mittlere Zweig zeigt, wie man mit Babel beispielsweise JavaScript in Version ES6/ES2015 nach JavaScript in Version ES5 übersetzen kann. Der dritte Zweig zeigt, wo Flow einzuordnen ist: Flow ist nicht Teil eines Übersetzungsvorgangs. Es ist ein zusätzlicher, nebenher einzusetzender Prüfprozess – ganz ähnlich, wie man dies ▶

auch schon von Stil-Prüfern wie JSHint, JSLint oder ESLint kennt. Flow gibt beispielsweise Meldungen aus, wenn es Typfehler im aktuellen Code findet.

Damit Typen sich in JavaScript bequemer notieren lassen, gibt es jedoch auch eine (optionale) Erweiterung der Syntax. Diese ähnelt der Typannotationssyntax in TypeScript. Verwendet man diese Syntaxerweiterung, dann muss diese natürlich vor der Ausführung im JavaScript-Interpreter (Browser, Node.js …) entfernt werden. Dazu gibt es beispielsweise ein Plug-in für Babel. Ohne diese Syntaxerweiterung ist weder Babel noch irgendein anderer gearteter Übersetzungsschritt notwendig.

Unterschied zu TypeScript

Ein wichtiger Unterschied zu TypeScript ist bei Flow also: Es wird keine neue Programmiersprache als Obermenge von JavaScript definiert. Flow ist ein externes, unabhängiges Codeprüfungswerkzeug. Die notwendigen Typannotationen können entweder außerhalb der JavaScript-Dateien, in Inline-Kommentaren oder in einer an TypeScript angelehnten Syntaxerweiterung vorgenommen werden. Die Semantik der Programmiersprache ECMAScript selbst wird aber nicht angetastet.

TypeScript erweitert JavaScript auch um Konzepte wie Klassen, Module, public/private oder Enums, die ursprünglich in JavaScript nicht vorhanden waren. Da manche dieser Konzepte seit ES2015 auch in JavaScript realisiert wurden, musste TypeScript mehrmals an den fortschreitenden ECMAScript-Standard angepasst werden. TypeScript verbindet die Typprüfung mit einem Feature-Transpiler (wie Babel.js) und damit unterschiedliche Anwendungszwecke in einem nicht trennbaren Ganzen.

Das widerspricht eigentlich dem Vorsatz »Separation of Concerns« und schränkt den Entscheidungsspielraum ein. Durch Microsofts Ansatz einer neuen Sprache muss man einfach darauf hoffen, dass TypeScript und JavaScript niemals wirklich auseinanderlaufen.

Werden die TypeScript-Entwickler wirklich jedes kommende JavaScript-Feature implementieren? Wie viele zusätzliche Features wird TypeScript noch hinzufügen? Bislang gibt es dazu das Versprechen der Entwickler, doch was, wenn die Vorstellung von Teilnehmern in der TC39-Gruppe (die Task Group, die hinter dem ECMAScript-Standard steht) und Microsoft deutlicher auseinandergehen? Es wäre nicht das erste Mal.

Typnotation und Inferenz

Damit ein Typprüfer wie Flow die Typen von Variablen, Parametern, Rückgabewerten et cetera kennt, kann der Programmierer Informationen dazu hinterlegen – im einfachsten Fall durch eine Typnotation. In C# kann man eine Variable mit dem Typ Zeichenkette folgendermaßen deklarieren:

```
string v = "Hallo";
```

In Java, C, C++ und vielen anderen Sprachen sieht das sehr ähnlich aus. Die Syntax in Flow (und TypeScript) ähnelt eher jener, wie man sie auch aus den funktionalen Programmiersprachen kennt:

```
var v : string = "Hallo";
```

Der Typ steht also hinter den Bezeichnern und nicht davor. Auch bei Parametern in Funktionen kann man durch einen nachgestellten Doppelpunkt einen Typ notieren:

```
function foo (a : string) : number {
  return a.length;
}
```

Wie man sieht, wird der Rückgabewert der Funktion (hier *number)* durch einen Doppelpunkt nach den Parameterklammern angegeben. Da die Entwickler bei Facebook und Microsoft hier teilweise zusammengearbeitet haben, ist diese Typsyntax bislang sowohl bei TypeScript als auch bei Flow gleich. Möglicherweise wird eine solche Syntaxerweiterung einmal in einen zukünftigen ECMAScript-Standard einfließen – dies würde ermöglichen, dass man reine Typprüfer wie Flow auch gänzlich ohne Übersetzungsschritt mit bequemer Inline-Syntax einsetzen kann. Wer dennoch heute schon unbedingt auf den Übersetzungsschritt verzichten möchte (oder muss), der kann auch folgende Syntax verwenden:

```
function foo (a /*: string*/) /*: number*/ {
  return a.length;
}
```

Die Typnotation wird also in Kommentaren eingeschlossen und macht den Code somit zu gewöhnlicher ES5-JavaScript-Syntax. Eine Übersetzung mit Babel ist nicht mehr notwendig.

Typinferenz

Zu Typnotationen gibt es stark polarisierende Meinungen: Manche Entwickler möchten möglichst viele Codestellen mit solchen Informationen kennzeichnen, da sie sich so leichter tun, zu verstehen, was das Programm macht. Andere Entwickler möchten ihren Code nicht durch unnötige, in ihren Augen redundante Informationen aufblähen und verunstalten. Tatsächlich ist ein großer Teil der gängigen Typnotationen redundante Information. Die Variable *v* des obigen Beispiels wird mit einem Zeichenkettenliteral initialisiert – damit sollte der Typ eigentlich offensichtlich sein. Das erkennt nicht nur ein Mensch – das erkennt auch der Computer, und wenn Letzterer solche Schlüsse zieht, dann nennt man das Typinferenz. Lässt man die Typnotation weg, dann sieht das Beispiel so aus:

```
function foo (a) {
  return a.length
}
```

Tatsächlich haben mit diesem Code weder Flow noch TypeScript ein Problem. Wieso auch? Es handelt sich um ganz ge-

```
> flow; test $? -eq 0 -o $? -eq 2

simple_inference.js:4
  4:     return a.length;
                ^^^^^^ property `length`. Property not
  4:     return a.length;
                ^ Number


Found 1 error
macbookpro-2:FlowTypes js$
```

Fehlermeldung: So meldet Flow einen Typfehler (Bild 2)

wöhnlichen JavaScript-Code. Doch was passiert, wenn man nun statt der ursprünglichen Zeichenkette eine Zahl übergibt?

Flow liefert eine Fehlermeldung (Bild 2), nach der das Objekt im Parameter *a* der Funktion kein Attribut *length* besitzt. Diese Folgerung ist korrekt, denn Zahlen haben in JavaScript keine Eigenschaft *length*. Flow hat durch Typinferenz festgestellt, dass eine Zahl an die Funktion *foo* übergeben wird und dass diese nicht zu dem Zugriff dieses Attributs passt. Derselbe Code in TypeScript wird ohne Meldung übersetzt. Nicht einmal zur Laufzeit wird ein Fehler auftreten. Stattdessen wird *foo()* einfach *undefined* als Ergebnis liefern und ein schwer zu findender Bug ist geschaffen worden.

Das Beispiel zeigt, wie TypeScript ohne explizite Typen oft keine große Hilfe darstellt. Doch was ist das Problem? TypeScript besitzt eine eingeschränktere Typinferenz. Wenn der Typ des Parameters als Zeichenkette annotiert wird, dann zeigt auch Typescript bei *foo(42)* einen Fehler an.

Hier zeigt sich wieder ein methodischer Unterschied zwischen Flow und TypeScript: Flow ist ein Zusatzwerkzeug, das dem Programmierer schon bei unverändertem JavaScript viel helfen soll. Der Name Flow kommt von Facebooks Ansatz einer extensiven Datenflussanalyse. Dieser Ansatz findet nicht nur mehr Typinformation aus gewöhnlichem JavaScript-Code, er kann sogar die Typinformation einer Variablen zu unterschiedlichen Zeitpunkten im Programm unterschiedlich erkennen. TypeScript dagegen lebt davon, dass der Programmierer seine Programme umschreibt und händisch durch Typinformationen anreichert. Der TypeScript-Compiler kann infolgedessen auch einfacher implementiert sein.

Null-Referenzen und Maybe-Typen

Auch wenn in der abgesicherten Laufzeitumgebung von Sprachen wie JavaScript, C# oder Java keine Dangling Pointers entstehen können, so gibt es dennoch das Problem der sogenannten Null-Referenzen. Derartige Situationen entstehen, wenn beispielsweise auf Variablen vor ihrer Wertinitialisierung zugegriffen wird oder wenn die Attribute von Objekten noch nicht vollständig initialisiert sind. Manchmal sind auch API-Funktionen so konzipiert, dass eine Funktion statt eines Objekts bewusst eine Null-Referenz als Ergebnis liefert. Ein Beispiel dafür sind Datenbankabfragen, die keinen Wert zu einem gegebenen Schlüssel finden und stattdessen *null* liefern. Das Problem: Eine Null-Referenz ist kein gültiges Objekt. Man kann weder auf die Attribute des eigentlichen Objekts zugreifen noch dessen Methoden aufrufen – all dies führt üblicherweise zu Laufzeitfehlern:

```
let meineZahl : number = null;
meineZahl.toString();
```

In diesem Beispiel wird eine Variable mit dem Typ *number* deklariert. Statt einer Zahl wird jedoch die Null-Referenz *(null* in JavaScript) zugewiesen. In der nächsten Zeile wird versucht, die Methode *.toString()* mit der Zahl aufzurufen. Für Zahlen wäre dies legitim, für die Null-Referenz jedoch nicht:

```
2: var x : number = null;
   ^^^^ null. This type is incompatible with
2: var x : number = null;
   ^^^^ number
```

Flow erkennt das Problem und meldet, dass der Typ von *null* inkompatibel mit *number* ist. Derselbe Code wird in TypeScript wieder ohne Fehlermeldung übersetzt und führt erwartungsgemäß zu einem Laufzeitfehler. Bei TypeScript ist der Typ *null* in jedem anderen Typ enthalten – was es unmöglich macht, derartige Null-Referenzen zu erkennen.

Doch wie löst man eigentlich das Problem in Flow? Es kann ja vorkommen, dass eine Funktion entweder einen Wert oder *null* zurückliefert. Was ist zu tun? Man kann den Typ als sogenannten Maybe-Typ deklarieren, indem man ein Fragezeichen davorschreibt. Dann wird auch *null* als zulässiger Wert betrachtet:

```
let meineZahl : ?number = null;
meineZahl.toString();
```

Doch überraschenderweise zeigt Flow wieder einen Fehler an:

```
4: x.toString();
   call of method 'toString'. Method cannot be called on
   possibly null value
4: x.toString();
   ^ null
```

Nun darf zwar die Variable *meineZahl* auch Null-Referenzen enthalten, aber Flow erkennt korrekterweise, dass versucht wird, die Methode *.toString()* mit einem Wert aufzurufen, der *null* sein kann. Damit dieser Fehler behoben werden kann, muss man *x* noch prüfen:

```
var x : ?number = null;
if (x) {
  x.toString();
}
```

Nun ist die Typprüfung mit Flow zufriedengestellt, und zur Laufzeit tritt auch kein Fehler mehr auf. Flow zwingt den Programmierer somit, den Code so zu schreiben, dass Lauf- ▶

zeitfehler möglichst verhindert werden. Um zu demonstrieren wie gut die Typinferenz in Flow wirklich ist, kann man auch noch einmal sämtliche Typen entfernen und statt der direkten Nullreferenz *null* eine Funktion benutzen, die manchmal (also zufällig) Null-Referenzen liefert:

```
function someFun () {
  return (Math.random() < 0.5)? 42 : null;
}
var x = someFun();
x.toString();
```

Man beachte: Das ist völlig normaler JavaScript-Code, ohne irgendeine syntaktische Erweiterung. Flow liefert mit diesem Code folgende Meldung:

```
9: x.toString();
  ^^^^ call of method 'toString'. Method cannot be
  called on
  possibly null value
9: x.toString();
  ^ null
```

Selbst hier erkennt der Typ-Checker also das Problem mit der Null-Referenz und verlangt, dass der Programmierer beispielsweise mit einem *if* explizit eine Prüfung durchführt.

Nominal oder strukturell

So schön ausgereifte, fundierte Typsysteme auch sind: Der Nutzer muss sich mitunter mit unzähligen neuen Begriffen befassen. Eines dieser Konzepte ist die Unterscheidung in nominale und strukturelle Typen. Damit ist Folgendes gemeint:

```
class Person {
  name : string;
  alter: number;
}
class Auto {
  name: string;
  alter: number;
}
```

Die beiden ES2015-Klassen *Person* und *Auto* besitzen keine gemeinsame Basisklasse. Es sind unterschiedliche Klassen und sie repräsentieren auch unterschiedliche Konzepte. Dennoch besitzen beide Klassen den gleichen Aufbau. Die Attribute besitzen die gleichen Namen und Typen. Man sagt, dass die beiden Klassen zwar nominell unterschiedlich sind, jedoch strukturell gleich. Hier gibt es wieder einen Unterschied zwischen Flow und TypeScript. Flow bewertet Klassen nominell. Bei folgendem Code schlägt also die Typprüfung fehl:

```
var person : Person = new Auto();
```

Für Flow ist ein Auto eben keine Person und damit hat es sich. Da TypeScript jedoch bewusst auf strukturelle Typisierung von Klassen setzt, sind solche Verwirrungen damit legitim. Microsoft begründet diese Abweichung mit der Besonderheit, wie Objekte in JavaScript oft verwendet werden. Tatsächlich ist dieses auch Duck Typing genannte Prinzip bei JavaScript sehr gängig. Allerdings lässt sich das für ein Typsystem auch besser lösen: In Flow kann man statt einer Klasse beispielsweise eine Schnittstelle definieren – diese wird lediglich strukturell verglichen:

```
interface INameable {
  name : string;
}
var person : INameable = new Auto();
console.log(person.name)
```

In diesem Fall steht die Schnittstelle *INameable* sogar nur für einen Teil der beiden Klassen: das Attribut *name*. Der exakt selbe Code funktioniert auch in TypeScript, aber mit Flow hat der Programmierer die Wahl, ob er nominale oder strukturelle Typen prüfen will.

Bibliotheken deklarieren

Ein wichtiger Aspekt von TypeScript sind die sogenannten *.d.ts*-Dateien beziehungsweise Typings. Dabei handelt es sich um Dateien, deren einziger Zweck es ist, Typdeklarationen für Bibliotheken bereitzustellen. Denn der absolute Großteil der JavaScript-Bibliotheken ist nicht mit statischen Typ-Annotationen implementiert worden – weder für Flow noch für TypeScript. Microsofts JavaScript-Dialekt hat hier bislang die Nase vorn. Es gibt mittlerweile mehrere Versionen von Typing-Repositories, die teilweise miteinander konkurrieren. Die bekanntesten sind TSD von http://definitelytyped.org und der neuere TypeScript Definition Manager Typings (http://github.com/typings/typings).

Eines vorweg: Flow bietet noch keine wirklich vergleichbaren Repositories mit Typdefinitionen. Die Flow-Entwickler arbeiten unter anderem daran, die TypeScript-Definitions-Dateien auch durch Flow zu nutzen. Das ist bedingt durch einige Unterschiede in den Typsystemen nicht trivial. Eine erste Sammlung von Flow-Deklarationen gibt es unter https://github.com/marudor/flowInterfaces.

Das offizielle Projekt ist jedoch unter https://github.com/flowtype/flow-typed zu finden. Dazu soll es ein NPM-Paket mit einem Kommandozeilenwerkzeug *flow-typed* geben. Die momentan per NPM verteilte Version dieses Projekts ist veraltet und an der Version 2.0.0 wird noch aktiv gearbeitet.

Das äquivalente Feature zu TypeScripts *.d.ts*-Dateien wird in Flow Declarations genannt. Die Idee dazu ist, dass man zu vorhandenen Definitionen (Implementation) unabhängig davon Typ-Deklarationen erstellen kann, die Flow dann benutzt, anstatt sich den Code selbst anzuschauen. Damit ist es also auch möglich, eine Typdeklaration für Code zu schreiben, den der Flow-Compiler noch nicht erfolgreich checken kann.

Beispielsweise könnte die Implementierung einer Funktion zur Laufzeit zwar Werte verschiedener Typen ungeprüft akzeptieren, aber in einer Flow-Deklaration wird der Parametertyp bereits auf *string* festgelegt. Solange man die Implementierung zusammen mit der Deklaration mit Flow prüft,

kann man sich auf diese Weise absichern. Flow kennt zwei unterschiedliche Wege, um Deklarationen zu hinterlegen. Man kann in der *.flowconfig* des Projekts Verzeichnisse mit derartigen Deklarationsdateien konfigurieren, und man kann neben JavaScript-Dateien gleich benannte Dateien mit der Endung *.js.flow* legen.

FlowConfig-Deklarationen

Im Wurzelverzeichnis eines Projekts, das Flow einsetzt, gibt es eine Datei mit dem Namen *.flowconfig*. Ist diese noch nicht vorhanden, dann kann man sie durch den Aufruf *flow init* erzeugen. Diese Datei ist anfangs leer, doch sie kann unterschiedliche Sektionen enthalten, die jeweils durch eine Überschrift in eckigen Klammern erkannt wird. Die folgende Konfigurationssektion erklärt ein Verzeichnis mit dem Namen *decls* als Quelle für Deklarationen im Projekt:

```
[libs]
decls
```

In diesem Verzeichnis kann man beispielsweise eine Datei *core.js* anlegen und dort wie folgt Deklarationen erstellen:

```
declare var DEBUG: bool;
```

Diese Deklaration definiert, dass es eine Variable mit dem Namen *DEBUG* gibt und dass ihr Typ *bool* ist. Man beachte, dass eine solche Deklaration keine derartige Variable erzeugt, sondern nur die Aussage trifft, dass es sie gibt und von welchem Typ sie ist. Der Programmierer muss sich selbst darum kümmern, dass die Variable an der Verwendungsstelle dann auch wirklich vorhanden ist. Tut er das nicht, wird es zu einem Laufzeitfehler kommen.

Auch Funktionen und Klassen kann man deklarieren:

```
declare function concat (a: string, b: string) : string;
declare class View {
  el : HTMLElement;
  render(): HTMLElement;
}
```

Wie bereits erwähnt, besteht die zweite Möglichkeit, Typen extern zu deklarieren, darin, neben den JavaScript-Dateien jeweils Dateien mit der Endung *.js.flow* anzulegen. Sobald eine solche Datei vorhanden ist, wird Flow sie anstatt der eigentlichen JavaScript-Datei verarbeiten. Auf diese Weise können auch Dateien getypt werden, die noch nicht vollständig mit Flow prüfbar sind.

Die bisherigen Beispiele haben schnell gezeigt, dass Flow gegenüber TypeScript einige Vorteile im Typsystem und der Typinferenz bietet. Scheinbar waren manche dieser Themen den TypeScript-Entwicklern weniger wichtig. Das liegt wohl unter anderem daran, dass einer der gewichtigsten Existenzgründe für TypeScript die Integration in Entwickler-Tools wie Visual Studio Code ist.

Microsoft ist eben insbesondere auch Anbieter von Entwicklungswerkzeugen, und auf diese Weise ist TypeScript eher ein Produkt als ein Projekt, das eigenen Entwicklungszwecken dient wie Flow. Die statische Typinformation soll eben vor allem Informationen für IDEs bieten.

Doch auch für Flow ist dieser Anwendungsfall durchaus wichtig. Nicht umsonst bietet das Kommandozeilenwerkzeug *flow* ein eigenes Kommando *autocomplete,* mit dem man Flow nach den notwendigen Informationen fragen kann. Dazu müssen lediglich der Dateiname und die Position in der Datei übergeben werden. Der folgende Quelltext ist noch unvollständig. Der Cursor steht auf der letzten Stelle der Datei – direkt hinter *product.:*

```
class Product {
  name : string,
```

Listing 1: Aufruf von autocomplete

```
$ flow autocomplete <
autocomplete.js 12 9
--json
{"result":[
  {
   "name":"name",
   "type":"string",
   "func_details":null,
   "path":"-",
   "line":2,
   "endline":2,
   "start":12,
   "end":17
  },
  {
   "name":"price",
   "type":"number",
   "func_details":null,
   "path":"-",
   "line":3,
   "endline":3,
   "start":12,
   "end":17
  }
 ]
}
```

```
  price: number,
  constructor (name, price) {
    Object.assign(this,{name,price});
  }
}
let product = new Product("Kamera", 1234.99);
product.[]?Cursor
```

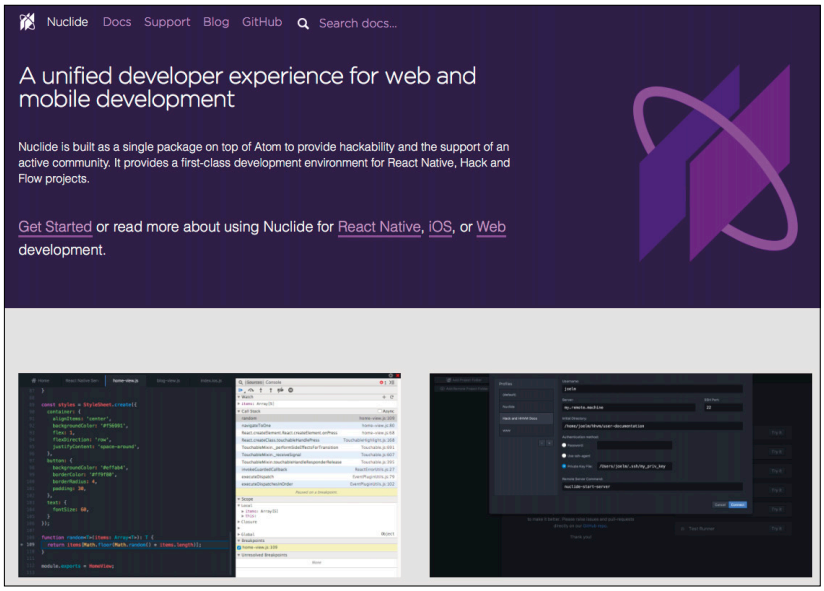
Der Editor zeigt an, dass sich der Cursor in Zeile 12 und Spalte 9 befindet. Auf der Konsole kann man nun folgendes Kommando ausführen:

```
$ flow autocomplete < autocomplete.js 12 9
name string
price number
```

Das Kommandozeilenwerkzeug *flow* wird mit dem Kommando *autocomplete* aufgerufen. Über die Standard-Eingabe wird die Datei in den Prozess geliefert und die Parameter *12* und *9* geben Zeile und Spalte in der Datei an. Flow liest die Datei in den Speicher, fügt an der angegebenen Cursor-Stelle einen Marker-Code ein und ermittelt dann die Autovervollständigung anhand der Position der Datei. In diesem Fall ▶

sind das die Attribute des Objekts in der Variablen *product*. Flow weiß, dass es sich um die Klasse *Product* handelt, und listet die Attribute der Klasse und deren Typen auf. Wer die Ergebnisse leichter weiterverarbeiten möchte, kann noch den Parameter *--json* übergeben (Listing 1).

In den JSON-Daten sind auch genauere Angaben über die Deklarationsstelle der jeweiligen Option vorhanden. Doch analysiert Flow tatsächlich bei jedem solchen Aufruf den gesamten Code? Nein. Flow startet im Hintergrund einen Dienst, der die Typinformation verarbeitet und für Anfragen bereithält. Sobald sich Dateien ändern, merkt der Dienst dies und passt seine Repräsentation der Daten entsprechend an.

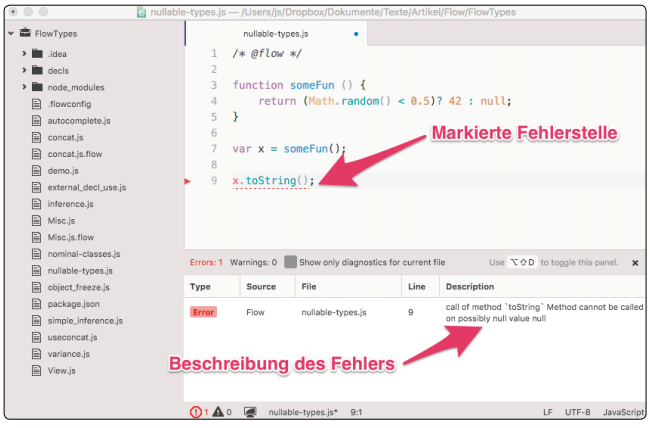


Nuclide: Die bislang ausgereifteste Entwicklungsumgebung mit Flow-Unterstützung (Bild 3)

Nuclide und andere IDEs

Die bislang ausgereifteste Entwicklungsumgebung mit Flow-Unterstützung ist ohne Zweifel und ohne Überraschung Nuclide von Facebook (http://nuclide.io) (Bild 3). Es handelt sich dabei um eine umfangreiche Sammlung von Plug-ins, die aus dem Open-Source-Editor Atom (http://atom.io) eine vollwertige IDE insbesondere für React, React Native und Hack realisieren. Nuclide bietet dafür viele integrierte Werkzeuge, Debugger und eben auch die Unterstützung für Autocompletion und Typprüfung mit Flow.

Bild 4 zeigt, wie Nuclide Fehler im Editor anzeigt, nachdem Flow sie gefunden hat: Die Fehlerstelle wird rot unterstrichen und im Gutter zeigt ein kleines Dreieck an, dass in dieser Zeile ein Fehler vorliegt. Ein aktivierbares Fenster listet alle gefundenen Fehler auf und zeigt detaillierte Beschreibungen dazu an. Auch durch ein rotes Icon in der Statusleiste wird die Anzahl der Fehler angezeigt. Über die Einstellungen der Entwicklungsumgebung kann man festlegen, ob die Prüfung ständig während der Eingabe oder aber erst beim Speichern der Datei erfolgen soll. Die ständige Prüfung ist dabei noch als experimentell gekennzeichnet, funktionierte aber im Test ohne Probleme.



Flow-Fehler: So zeigt Nuclide Fehler im Editor an (Bild 4)

Schwebt man mit dem Mauszeiger über dem Quellcode, so zeigt Nuclide Typinformationen zu darunterliegenden Codestellen an. Bild 5 zeigt, dass die Variable *result* in dieser *for*-Schleife den Typ *number* besitzt. In Bild 6 sieht man, wie man zu Funktionstypen noch weitere Details erhält.

Auch eine Autovervollständigung wird in Nuclide umgesetzt: Bild 7 zeigt dieselbe Code-Stelle an, die vorhin bereits per Kommandozeile untersucht wurde. Der Entwickler kann so wählen, ob er das Attribut *name* oder *price* einfügen will, und er sieht auch die Typen der Attribute sofort im Pop-up.

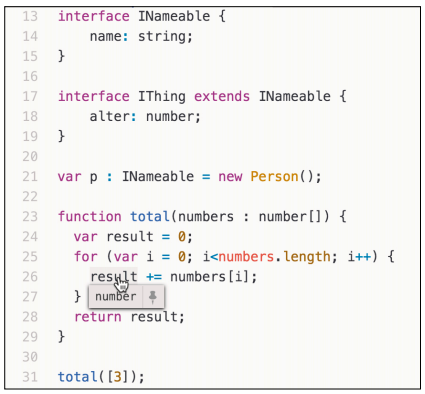
Für Visual Studio Code gibt es eine auf der Nuclide-Flow-Integration basierende Implementierung, die ebenfalls Syntax Coloring, IntelliSense, Go to Definition und Fehlermeldungen unterstützt.

Auch für Emacs gibt es diverse Pakete, um Flow zu integrieren: unter anderem *flycheck-flow,* mit dem man Flow im Flycheck-Emacs-Mode aktivieren kann. Für VIM-Fans gibt es das Plug-in *vim-flow,* und auch Fans von Sublime finden mit *SublimeLinter-flow* ein Plug-in für ihren Lieblingseditor.

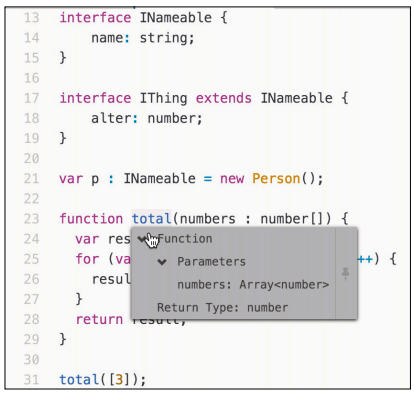
Die wohl momentan beliebteste kommerzielle Web-Entwicklungsumgebung ist Jetbrains WebStorm. Die IDE bietet in ihrer aktuellsten Version hervorragende Unterstützung von TypeScript, aber Flow wird bislang nur rudimentär unterstützt. Es gibt bislang keine richtige Integration des Typcheckers, und auch die Vervollständigung nutzt noch nicht die Möglichkeit, Flow direkt nach Typinformation zu fragen. Stattdessen versteht Webstorm bislang lediglich die Flow-Syntax und kann aus im Code vorhandenen Typinformationen Vorschläge anbieten.

Windows

In der Open-Source-Welt und insbesondere, wenn es um Werkzeuge für Webentwickler geht, stellt man fest, dass in erster Linie die Plattformen Mac OS X und Linux bedient werden. Ein großer Teil der Open-Source-Entwickler arbeitet un-



MausOver: Typinformation über den darunterliegenden Quellcode (Bild 5)



Detailliert: Details wie Parameter und Rückgabewert-Typen (Bild 6)

Autocomplete: Nuklide bietet auch eine Autovervollständigung an (Bild 7)

ter diesen Systemen, und es wird meist als notwendiges Übel betrachtet, die selbst geschaffenen Werkzeuge auch unter Windows verfügbar zu machen. Glücklicherweise hat sich hier in den letzten Jahren vieles zum Besseren gewendet. Node.js funktioniert gut unter Windows, und auch die Versionsverwaltung Git wird mittlerweile erstklassig unterstützt.

Leider gilt das bislang noch nicht für Facebooks Flow. Offensichtlich wird Windows in den Entwickler-Labors bei Facebook nicht wirklich genutzt, sodass Flow ursprünglich ausschließlich unter Mac OS X und Linux lief.

Flow ist in der funktionalen Programmiersprache OCaml implementiert. OCaml ist zwar eine sehr ausdrucksstarke und plattformübergreifend verfügbare Sprache, aber die Zahl der versierten OCaml-Entwickler ist dennoch überschaubar – vor allem, wenn es um systemnahe Programmierung geht.

Glücklicherweise haben sich die Entwickler von OCamlPro (http://ocamlpro.com) bereit erklärt, eine Portierung nach Windows zu unterstützen. Flow ist ihrer Meinung nach ein gutes Testfeld für den OCaml-Window-Compiler. Zum Zeitpunkt dieses Artikels war bei Flow gerade Version 0.22.1 aktuell und OCamlPro bot auf ihrer Website Window-Binaries der Version 0.19.1 an.

Fazit

Flow besitzt eine signifikant stärkere Typinferenz und kann damit wichtige Fehlerklassen in JavaScript-Code erkennen, die TypeScript teilweise nicht einmal durch aufwendige Typauszeichnung entdecken kann. Null-Referenzen sind eine große Fehlerklasse – insbesondere in JavaScript.

Es ist zu erwarten, dass Microsoft diese Mängel mit späteren Versionen von TypeScript zumindest zum Teil löst. So wie es im Moment aussieht, könnte man Variablen möglicherweise später explizit als *non-nullable* kennzeichnen. Dann wird jedoch weiterhin als Default überall *nullable* angenommen. Weiterhin wird Flow durch seine Fluss-Analyse in nicht annotiertem Quellcode mehr erkennen als TypeScript. Dies scheint ein fundamentaler Unterschied der beiden Lösungen zu sein.

Auch wenn mittlerweile aktuelle Builds der Flow-Binaries für Windows vorliegen, so müssen diese nach wie vor händisch heruntergeladen werden. Hoffentlich wird dieser Umstand möglichst bald beseitigt, denn an einem so einfachen Problem scheitert schlicht und einfach das bequeme schnelle Ausprobieren des Systems für eine beträchtliche Entwicklergemeinde, die auf Windows-Maschinen entwickelt. Der JavaScript-Port des Typecheckers könnte diese Situation ebenfalls verbessern oder gar neue Anwendungsmöglichkeiten in Webbrowsern ermöglichen.

Vermutlich hängt JetBrains bisherige Zurückhaltung in Sachen Windows auch mit der Situation des Windows-Ports zusammen. Denn im Business-Umfeld ist Windows eben dennoch weit verbreitet und viele der Kunden, die Webstorm einsetzen, könnten eine entsprechende Flow-Integration sonst nicht sinnvoll nutzen.

Auch die Sammlungen fertiger FlowType-Deklarationen scheinen sich bereits zu verbessern – aber auch da ist noch einiges zu tun. Nichtsdestotrotz ist der Ansatz sehr reizvoll: Ein unabhängiger Typprüfer anstelle eines JavaScript-Dialekts oder gar einer eigenen Sprache erfordert weniger Investition durch die Entwickler und sollte sich in bestehenden Projekten leichter gewinnbringend adaptieren lassen. ■

Links zum Thema

- Flow Website
  http://flowtype.org
- Flow im Browser ausprobieren
  http://tryflow.org

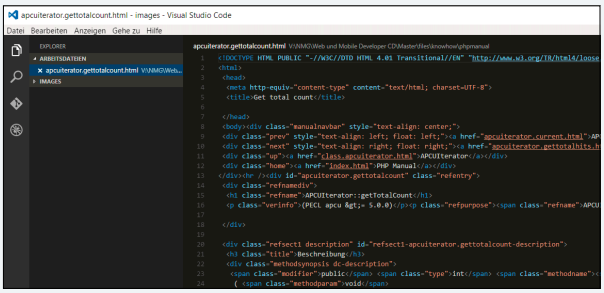
Jochen H. Schmidt
ist als Autor, Berater und Software-Entwickler tätig. Schwerpunkte seiner Aktivitäten sind Webentwicklung und Webtechnologien. Er ist Verfasser von bekannten Fachbüchern zum Thema Common Lisp.

ÜBERBLICK

CD-Highlights

Auf der Heft-CD finden Sie Werkzeuge für Webentwickler und nützliche Tools.

Microsoft Visual Studio Code 1.0.0



Mit dem kostenlosen Code-Editor sollen sich Anwendungen mit ASP.NET Framework und Node-Anwendungen schnell umsetzen lassen. Unterstützt werden unter anderem die Webtechnologien HTML, CSS, JSON, LESS, Sass, JavaScript und PHP. Da die Technik von Visual Studio Code in großen Teilen auf dem Atom-Editor von GitHub basiert, können Projekte als Git-Repository angelegt sein. Code nutzt eine angepasste Version der in Atom eingesetzten Electron Shell, die Node.js sowie Chromium zur Darstellung der Oberfläche verwendet. So lässt sich etwa die Unterstützung für verschiedene Programmiersprachen in wenigen Zeilen JavaScript umsetzen.

Google Web Designer 1.5.4

Mit dem Google Web Designer erstellen Sie dynamische Anzeigenformate und Webseiten in HTML5. In der Designansicht bietet der Webeditor verschiedene Zeichentools, mit denen Sie Texte und 3D-Objekte beliebig platzieren, anpassen und animieren. Außerdem gibt es eine Anpassung an verschiedene Bildschirmauflösungen, eine Codeansicht mit Syntax-Highlighting und eine Autovervollständigung. Unterstützt werden HTML, CSS, JavaScript und XML.

Weitere Highlights: Texteditoren

AkelPad 4.9.7

Die Freeware lässt sich mit Hilfe von Plug-ins funktional erweitern. AkelPad bietet eine volle Unicode-Unterstützung und kann im Mehrfenster-Modus mit Tabbed Browsing geöffnet werden. Gute Extras sind das Konvertieren von Schreibweisen sowie das Suchen/Ersetzen. Ein Plus von AkelPad ist seine Geschwindigkeit.

Bluefish 2.2.8

Das Programm bietet eine ganze Reihe von Features. Integriert sind eine Projektverwaltung und Assistenten zu CSS, JavaScript, WML und Metatags. Dazu gibt es Syntax-Highlighting, geschlossene Listen- und Paragraph-Tags, wählbare XHTML-Syntax, Einbindung externer HTML-Syntax-Checker und Spell-Checker.

BlueGriffon 1.8

Der Wysiwyg-Editor hält sich an die W3C-Webstandards und erstellt zu HTML 4, HTML 5, XHTML 1, XHTML 1.1 und XHTML 5 kompatible Seiten mit CSS 2.1 und 3. Über den Schalter am unteren Rand wechseln Sie zwischen Quelltext- und der Wysiwyg-Ansicht. BlueGriffon basiert auf der HTML-Rendering-Engine Gecko.

Notepad++ 6.9.1

Wollen Sie in Notepad++ etwa zwei Dokumente gleichzeitig editieren, lassen sich beide Texte in einer Split-Screen-Ansicht darstellen. Beim Editieren von HTML-Seiten oder Quelltexten bietet Notepad++ Syntax-Hervorhebung und -Gliederung mit Code-Collapsing. Wenn Sie Dateien öffnen, erkennt das Tool die Programmiersprache.

Synwrite 6.20.2212

Der vielseitige Quelltext-Editor unterstützt das Arbeiten im Quelltext durch das automatische Hervorheben der Syntax von über 80 Sprachen von ADA bis XML. Der Übersicht dient neben der üblichen Baumstruktur eine Minimap, für wiederkehrende Aufgaben gibt es einen Makrorekorder. Dazu gibt es einen integrierten FTP-Client.

Jetzt kostenlos testen!

2 x gratis!

Praxiswissen für Entwickler!

Testen Sie jetzt 2 kostenlose Ausgaben und erhalten Sie exklusiven Zugang zu unserem Archiv.

webundmobile.de/probelesen

DER UIALERTCONTROLLER IN DER PRAXIS

Einheitliche Schnittstelle

Der UIAlertController dient seit iOS 8 als einheitliche Schnittstelle für Nutzermeldungen.

Mit dem Release von iOS 8 schickte sich die neue Klasse *UIAlertController* an, die bisher für Alerts verwendeten Klassen *UIAlertView* und *UIActionSheet* abzulösen. Der neue UIAlertController dient als einheitliche Schnittstelle für Nutzermeldungen aller Art und soll den Umgang mit Alerts und Action-Sheets deutlich erleichtern.

UIAlertView und UIActionSheet sind seit iOS 8 deprecated (Bild 1)

So weit, so gut. Bis iOS 8 war das auch kein größeres Problem, da sowohl UIAlertController als auch UIAlertView und UIActionSheet problemlos nebeneinander verwendet werden konnten. Das hat sich aber mit iOS 9 geändert.

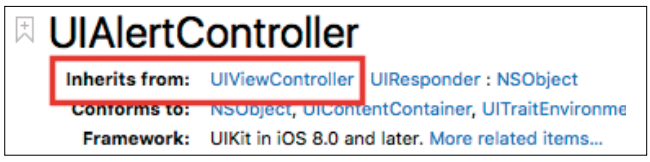
Ab iOS 8 sind die Klassen *UIAlertView* und *UIActionSheet* von Apple als deprecated gekennzeichnet (Bild 1). Im Klartext bedeutet das für App-Entwickler: Hände weg von diesen Klassen. Allerdings stellt es bei Apps, die noch wenigstens auf iOS 8 oder niedriger setzen, kein Problem dar, noch immer UIAlertView oder UIActionSheet zu verwenden. Ganz im Gegensatz zu iOS 9: Bei App-Projekten, in denen mindestens iOS 9 vorausgesetzt wird, versieht der Compiler jede Verwendung einer UIAlertView oder eines UIActionSheets mit einer Warnung und weist darauf hin, dass stattdessen doch eine Instanz der Klasse *UIAlertController* verwendet werden soll.

Spätestens dann wird es höchste Eisenbahn, auf Apples neue Universalklasse umzustellen und bisherige Alert-Views und Action-Sheets aus den eigenen Projekten zu verbannen. Wie stattdessen die Verwendung von UIAlertController aussieht und worauf Sie im Vergleich zu den beiden bisherigen View-Klassen achten müssen, erläutert dieser Artikel.

UIAlertController ist ein View-Controller

Der größte Unterschied zwischen den bisherigen Alert-View- und Action-Sheet-Klassen im Vergleich zu UIAlertController ist sicherlich der, dass UIAlertController ein View-Controller ist; er erbt direkt von *UIViewController* (Bild 2).

UIAlertView und *UIActionSheet* hingegen sind View-Klassen und erben beide direkt von *UIView* (Bild 3). Sie bringen jeweils passende Methoden mit, um angezeigt und eingeblendet zu werden. So wird eine UIAlertView beispielsweise mit Hilfe der Instanzmethode *show* innerhalb der aktuellen Ansicht angezeigt. UIActionSheet bringt mehrere verschiedener solcher Methoden mit, um beispielsweise ein Action-Sheet aus einer Tab-Bar oder Tool-Bar heraus anzuzeigen.



UIAlertController ist eine Subklasse von *UIViewController* (Bild 2)

Da der neue UIAlertController ein View-Controller ist, ist das Anzeigen eines solchen simpel und identisch zum Anzeigen beliebiger anderer View-Controller. So können Sie eine *UIAlertController*-Instanz beispielsweise mit Hilfe der Methode *presentViewController:animated:completion:* aus einem beliebigen anderen View-Controller heraus aufrufen und anzeigen. Sie müssen dabei lediglich den anzuzeigenden UIAlertController als Parameter übergeben.

Alles eine Sache des Stils

Der nächste Unterschied zu den bisherigen View-Klassen liegt darin, dass UIAlertController als Universal-Controller für alle Arten von Mitteilungen dient. Bisher können diese Mitteilungen auf die zwei bekannten Arten und Weisen präsentiert werden: als Alert-View oder Action-Sheet. Wie der Alert-Controller beim Anzeigen präsentiert wird, hängt davon ab, welche dieser beiden Optionen ihm als Stil zugewiesen wurde.

Zu diesem Zweck ist in der Klasse *UIAlertController* ein eigener Typ namens *UIAlertControllerStyle* definiert. Es handelt sich dabei um eine Enumeration, die aktuell über zwei Werte verfügt: *ActionSheet* und *Alert*.

Je nachdem, welcher Wert einer *UIAlertController*-Instanz zugewiesen wird, präsentiert diese sich beim Anzeigen entweder als Action-Sheet *(ActionSheet)* oder Alert *(Alert)*.

Diese grundlegende Änderung hat zur Folge, dass Apple in Zukunft einfach neue Werte für *UIAlertControllerStyle* definieren kann, um neue Ansichten für Mitteilungen bereitzustellen. Das ist deutlich komfortabler und übersichtlicher als mehrere verschiedene UIView-Subklassen, die alle anders implementiert sind und über unterschiedliche Methoden verfügen. Auf diese Art und Weise bleibt die grundlegende Ar-

beit mit *UIAlertController*-Instanzen immer gleich, während lediglich der verwendete Stil bestimmt, wie die jeweilige Mitteilung präsentiert wird.

Initialisierung eines UIAlertControllers

Mit diesen grundlegenden Informationen gewappnet betrachten wir nun einmal die Initialisierung eines neuen *UIAlertController*-Objekts. Dafür bringt die Klasse einen eigenen Convenience Initializer mit, den man typischerweise immer verwenden sollte, um neue Instanzen eines UIAlertController zu erstellen:

```
convenience init(title title: String?, message message:
String?, preferredStyle preferredStyle:
UIAlertControllerStyle)
```

Die Parameter *title* und *message* dienen dabei dazu, dem Nutzer Informationen zur jeweiligen Mitteilung anzuzeigen. Der Titel sollte dabei kurz und prägnant gehalten werden, während die Nachricht weitere Informationen und konkrete Details enthalten kann. Der Parameter *preferredStyle* vom Typ *UIAlertControllerStyle* legt dann die Art und Weise fest, wie die Mitteilung präsentiert wird.

Aktionen eines UIAlertControllers

Bevor ein so initialisierter UIAlertController aber genutzt werden kann, müssen ihm zuvor noch eine oder mehrere sogenannte Actions zugewiesen werden. Bildlich gesprochen entspricht eine Action einer Schaltfläche, die zusammen mit der jeweiligen Mitteilung angezeigt wird und bei Tap darauf eine bestimmte Aktion auslöst. Somit müssen also für alle Schaltflächen, die zusammen mit der Mitteilung angezeigt werden sollen, jeweils passende Actions erstellt werden.

Actions werden dabei über die Klasse *UIAlertAction* abgebildet und setzen sich aus drei Bestandteilen zusammen:

- Der *title* entspricht dem Titel der jeweiligen Action, der auf der entsprechenden Schaltfläche der Mitteilung angezeigt wird.
- Der *style* bestimmt das grundlegende Aussehen des Action-Buttons.
- Bei *handler* handelt es sich um einen Block, der als Parameter das zugrunde liegende *UIAlertAction*-Objekt übergibt. Innerhalb dieses Blocks wird die Aktion implementiert, die ausgeführt werden soll, wenn die entsprechende Action-Schaltfläche vom Nutzer betätigt wird.

Alle diese drei Parameter werden idealerweise bei der Initialisierung eines neuen *UIAlertAction*-Objekts gesetzt. Dazu bringt die Klasse einen passenden Convenience Initializer mit:

```
convenience init(title title: String?, style style:
UIAlertActionStyle, handler handler: ((UIAlertAction) ->
Void)?)
```

Ähnlich wie bei UIAlertController wird der Stil der Action-Buttons über einen eigenen Typ namens *UIAlertActionStyle* definiert, der Bestandteil der Implementierung von *UIAlertAction* ist. Auch hierbei handelt es sich um eine Enumeration, die aktuell über die folgenden drei Werte verfügt:

- *Default*: Die Standarddarstellung eines Action-Button.
- *Cancel*: Eine angepasste Darstellung für Action-Buttons, die zum Abbruch der geplanten Aktion führen.
- *Destructive*: Dieser Stil dient dazu, den Nutzer darauf hinzuweisen, dass das Betätigen eines solchen Action-Buttons zur Veränderung oder Löschung von Daten führen kann.

Der *handler*-Parameter ist optional und muss nur gesetzt werden, wenn das Betätigen des entsprechenden Action-Buttons auch tatsächlich eine Aktion nach sich ziehen soll. Abbrechen-Schaltflächen, die die Durchführung einer Aktion beenden, brauchen somit auch keinen leeren *handler*-Block zu setzen, sondern können stattdessen für den Handler einfach *nil* übergeben.

Etwas paradox ist die Deklaration des *title*-Parameters im Convenience Initializer von UIAlertAction. Zwar ist dieser Parameter als optional deklariert, in der Dokumentation weist Apple aber ganz klar darauf hin, dass dieser Parameter nicht *nil* sein darf (Bild 4). Entsprechend sollten Sie auch für jede Action einen passenden Titel mit übergeben. Um eine Action einem *UIAlertController* zuweisen zu können, stellt die Klas-

Bei UIAlertView und UIActionSheet handelt es sich um *UIView*-Subklassen (Bild 3)

se die Instanzmethode *addAction:* bereit, die im Folgenden deklariert ist:

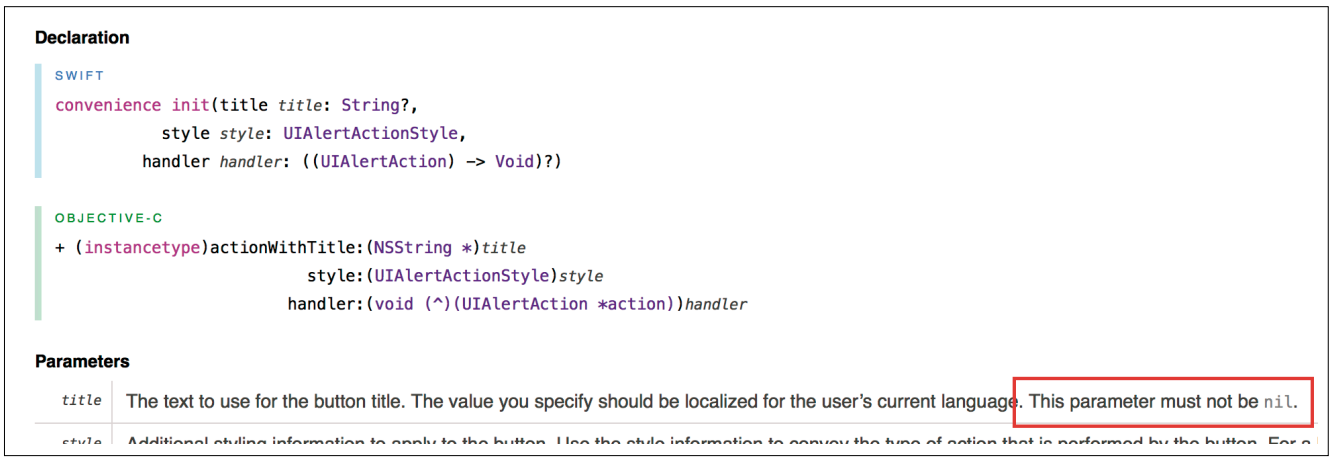
```
func addAction(_ action: UIAlertAction)
```

Alle Actions, über die ein UIAlertController verfügen soll, müssen nach und nach mit Hilfe dieser Methode einer entsprechenden Instanz der Klasse zugewiesen werden. Eine Möglichkeit, mehrere Actions auf einmal einem UIAlertController zuzuweisen, existiert zum jetzigen Zeitpunkt nicht.

Erstellen eines einfachen UIAlertControllers

Damit haben wir nun alle grundlegenden Informationen beisammen, um eine neue, voll funktionsfähige Instanz eines UIAlertControllers erstellen und konfigurieren sowie anschließend anzeigen zu können. Listing 1 zeigt einmal beispielhaft die Erstellung und Konfiguration eines einfachen UIAlertControllers, der anschließend angezeigt wird.

Das Listing besteht aus drei Teilen. Zunächst erfolgt innerhalb der Methode *showSimpleAlert* die Erstellung einer neuen *UIAlertController*-Instanz mit Hilfe des Convenience Initializers *title:message:preferredStyle:*. Dabei wird ein einfacher Alert erstellt. Im nächsten Schritt folgt die Initialisie- ▶



Declaration

SWIFT

```
convenience init(title title: String?,
           style style: UIAlertActionStyle,
         handler handler: ((UIAlertAction) -> Void)?)
```

OBJECTIVE-C

```
+ (instancetype)actionWithTitle:(NSString *)title
                          style:(UIAlertActionStyle)style
                        handler:(void (^)(UIAlertAction *action))handler
```

Parameters

title	The text to use for the button title. The value you specify should be localized for the user's current language. This parameter must not be `nil`.
style	Additional styling information to apply to the button. Use the style information to convey the type of action that is performed by the button. For a

Widersprüchlich: Auch wenn der *title*-Parameter als optional deklariert ist, darf er laut Doku nicht *nil* sein (Bild 4)

rung einer Action, die über den Titel *OK* verfügt und vom Typ *Default* ist. Sie führt beim Betätigen keine spezifische Aktion durch und sorgt damit lediglich dafür, dass der Alert verschwindet; aus diesem Grund ist der Parameter *handler* bei der Initialisierung der UIAlertAction gleich *nil*. Die so erzeugte Action wird im Anschluss daran unserem zuvor initialisierten UIAlertController mit Hilfe der Methode *addAction*: zugewiesen. Zu guter Letzt wird der Controller dann mittels der Methode *presentViewController:animated:completion:* der Klasse *UIViewController* auf dem Bildschirm angezeigt.

Listing 2 führt das Beispiel von eben noch einen Schritt weiter. Dieses Mal werden dem Alert-Controller drei verschiedene Actions zugewiesen, die jeweils mittels *print* eine Meldung auf der Konsole ausgeben. Je nachdem, welche der beiden Actions der Nutzer ausführt, wird das jeweils passende *print*-Statement ausgegeben.

Informationen eines UIAlertControllers

Die Klasse *UIAlertController* bringt drei Properties mit, um auf die Informationen des View-Controllers zugreifen und diese ändern zu können. Dabei handelt es sich um genau dieselben Eigenschaften, über die auch bereits der Convenience Initializer der Klasse verfügt: *title*, *message* und *preferredStyle*.

Dabei können Sie die Werte von *title* und *message* auch nach Initialisierung eines UIAlertControllers noch verändern, lediglich die Eigenschaft *preferredStyle* ist read-only und muss daher zwingend bei der Erstellung eines neuen Objekts wie gewünscht gesetzt werden.

Darüber hinaus steht Ihnen noch die Read-only-Property *actions* zur Verfügung. Diese liefert ein Array mit allen *UIAction*-Objekten zurück, die einem Alert-Controller mittels der Methode *addAction:* zugewiesen wurden.

Präferierte Actions

Ist ein UIAlertController mit dem Style *Alert* konfiguriert, haben Sie die Möglichkeit, eine sogenannte präferierte Action festzulegen. Das ist für diejenige Action eines Alerts gedacht, von der man ausgeht oder erwartet, dass diese in der Regel ausgelöst wird. Der UIAlertController hebt diese Action optisch ein wenig von den anderen ab. Ist das iOS-Gerät darüber hinaus mit einem externen Hardware-Keyboard verknüpft, führt das Betätigen der Return-Schaltfläche bei Anzeige eines solchen Alerts ebenfalls zur Ausführung der präferierten Action.

Um eine Action eines Alerts als präferierte Action festzulegen, müssen Sie diese der Property *preferredAction* des UIAlertControllers zuweisen:

```
var preferredAction: UIAlertAction?
```

Den Rest übernimmt dann das System.

Mit dem bisher Gezeigten sind Sie in der Lage, einfache Alerts mit einer oder mehreren Actions mittels UIAlertController zu erstellen und passend auf die vom Nutzer gewählte Schaltfläche zu reagieren.

Wie bei der UIAlertView haben sie darüber hinaus aber auch mit dem UIAlertController die Möglichkeit, in Ihrer Meldung zusätzliche Textfelder anzuzeigen, in denen der Nutzer weitergehende Informationen handhaben kann. Beispielsweise können Sie so einen Nutzer nach dem Namen eines neu zu erstellenden Dokuments direkt aus einem Alert heraus fragen und diesen dann weiterverarbeiten. Um einem Alert ein neues Textfeld hinzuzufügen, müssen Sie die ▶

Listing 1: Erstellen eines UIAlertControllers

```
class MyViewController: UIViewController {
  func showSimpleAlert() {
    let alertController = UIAlertController
    (title: „Alert“, message: „Alert message“,
    preferredStyle: .Alert)
    let alertAction = UIAlertAction(title:
    „OK“, style: .Default, handler: nil)
    alertController.addAction(alertAction)
    presentViewController(alertController,
    animated: true, completion: nil)
  }
}
```

Developer Newsletter

Top-Informationen für Web- und Mobile-Entwickler.
Klicken. Lesen. Mitreden.

Jetzt kostenlos anmelden:

webundmobile.de

twitter.com/webundmobile

facebook.de/webundmobile

gplus.to/webundmobile

Listing 2: UIAlertController mit verschiedenen Actions

```
class MyViewController: UIViewController {
  func showSimpleAlert() {
    let alertController = UIAlertController
    (title: "Alert", message: "Alert message",
    preferredStyle: .Alert)
    let firstAction = UIAlertAction(title:
    "First", style: .Default) { (alertAction:
    UIAlertAction) -> Void in
      print("First action.")
    }
    let secondAction = UIAlertAction(title:
    "Second", style: .Default) { (alertAction:
    UIAlertAction) -> Void in
      print("Second action.")
    }
    let thirdAction = UIAlertAction(title:
    "Third", style: .Default) { (alertAction:
    UIAlertAction) -> Void in
      print("Third action.")
    }

    alertController.addAction(firstAction)
    alertController.addAction(secondAction)
    alertController.addAction(thirdAction)
    presentViewController(alertController,
    animated: true, completion: nil)
  }
}
```

Methode *addTextFieldWithConfigurationHandler*: der Klasse *UIAlertController* aufrufen, die im Folgenden deklariert ist:

```
func addTextFieldWithConfigurationHandler
(_ configurationHandler: ((UITextField) -> Void)?)
```

Interessant ist hier der optionale Parameter *configurationHandler* – ein Block, der ein Objekt vom Typ *UITextField* übergibt. Sie können ihn nutzen, um das anzuzeigende Textfeld Ihren Wünschen entsprechend zu konfigurieren. Über den übergebenen Textfeld-Parameter des Blocks können Sie etwa einen Platzhalter setzen oder einen Standardtext einfügen. Wenn das für Sie aber nicht nötig ist, können Sie auch einfach *nil* für den *completionHandler*-Parameter übergeben. Mittels dieser Methode können Sie nach und nach alle benötigten Textfelder einem Alert hinzufügen. Zu beachten ist dabei, dass die so hinzugefügten Textfelder nur bei AlertControllern vom Typ *Alert* zur Verfügung stehen. Bei ActionSheets lassen sich Textfelder somit nicht nutzen und verwenden. Um auf alle erstellten Textfelder eines UIAlertControllers zugreifen zu können, steht Ihnen die Read-only-Property *textFields* zur Verfügung. Sie liefert ein Array mit allen Textfeldern des entsprechenden UIAlertControllers zurück.

Etwas problematisch an diesem Ansatz der Textfelder eines UIAlertControllers ist, dass Sie selbst dafür Sorge tragen müssen, das korrekte Textfeld auszulesen und zu verwenden. Letzten Endes greifen Sie dazu optimalerweise auf das eben vorgestellte *textFields*-Array von UIAlertController zu. Listing 3 zeigt ein weiteres einfaches Beispiel eines UIAlertControllers, der über ein einzelnes Textfeld verfügt, dessen Inhalt er bei Ausführung einer Action auf der Konsole ausgibt.

Listing 3: UIAlertController mit Textfeld

```
class MyViewController: UIViewController {
  func showSimpleAlert() {
    let alertController = UIAlertController
    (title: "Alert", message: "Alert message",
    preferredStyle: .Alert)
    alertController.
    addTextFieldWithConfigurationHandler(nil)
    let textAction = UIAlertAction(title:
    "First", style: .Default) { (alertAction:
    UIAlertAction) -> Void in
      let textField =
      alertController.textFields![0]
      print("Entered text: \(textField.text)")
    }
    alertController.addAction(textAction)
    presentViewController(alertController,
    animated: true, completion: nil)
  }
}
```

Fazit

Zunächst mag die Arbeit mit *UIAlertController* aufwendiger und komplexer erscheinen, schließlich hatte man mit *UIAlertView* und *UIActionSheet* bisher zwei einfache View-Klassen an der Hand, um dem Nutzer Mitteilungen anzuzeigen und auf eine Auswahl zu reagieren. Hat man allerdings einmal die Funktionsweise und das Verhalten eines UIAlertControllers verinnerlicht, lassen sich auch damit schnell und einfach verschiedene Arten von Alerts umsetzen. ■

Thomas Sillmann
ist iOS-App-Entwickler, Trainer und Autor. Freiberuflich tätig programmiert er für den App Store eigene Apps sowie Apps in Form von Kundenaufträgen. Er ist Autor eines erfolgreichen Fachbuchs und mehrerer Artikel in Fachzeitschriften.

DWX Developer Week 2016

**20.-23. Juni 2016,
Messe Nürnberg**

Unsere Leser sparen
€ 149,–
mit Code
DWX16wmd

Das Event 2016 für .NET-, Web- & Mobile-Entwickler

Keynote Speaker
Jurgen Appelo
Managing
for Happiness

Keynote und Workshop
Scott Hanselman
Java Script, The Web, Mobile and
the Rise of the New Virtual Machines

developer-week.de **DeveloperWeek**

Aussteller & Sponsoren:

adorsys · AIT · brainLight LIFE IN BALANCE · DevExpress · Fast Reports Reporting must be Fast! · GFU Cyrus AG · HeiReS · hemmersbach · JetBrains · MATHEMA · MAXIMAGO · Microsoft · Saxonia Systems So geht Software. · SDX · SHS VIVEON · SOS Software Service Value Added Distribution · Telerik · TEXT CONTROL · ThingWorx · hello, technology · UID

Veranstalter:

developer media
Neue Mediengesellschaft Ulm mbH

Präsentiert von:

dotnetpro
web & mobile DEVELOPER

DWX Developer Week 2016

Info

Grußwort des Wirtschaftsreferenten der Stadt Nürnberg

Dr. Michael Fraas, Wirtschaftsreferent der Stadt Nürnberg

Ein herzliches Willkommen allen Konferenzteilnehmerinnen und -teilnehmern auf der Developer Week 2016 in Nürnberg. Zum vierten Mal trifft sich die Entwicklerszene in Nürnberg, um über neue und zukunftsweisende Technologien und digitale Trends zu diskutieren. Der beachtliche Zuwachs der größten unabhängigen Softwareentwickler-Konferenz Europas bei Ausstellern und Besuchern bestätigt die Attraktivität des Themas und des Veranstaltungsortes.

Die Halbmillionenstadt Nürnberg ist das Herzstück einer Europäischen Metropolregion, die mit ihren 3,5 Millionen Menschen zu den zehn großen Wirtschaftszentren Deutschlands gehört. Nürnberg ist High Tech-, Industrie- und Dienstleistungsstandort. Der Standort entwickelt sich zu einer Innovations-Hauptstadt für Zukunftstechnologien, Forschung und Entwicklung.

Mit seinem starken Technologieprofil in Information und Kommunikation versteht sich Nürnberg als digitale Metropole. Mit mehr als 100.000 Beschäftigten ist die Informations- und Kommunikationstechnologie eine sehr bedeutende Branche in der Metropolregion Nürnberg. Knapp 10 % der Beschäftigten in Nürnberg arbeiten im IKT-Sektor – damit belegt Nürnberg einen Spitzenplatz unter den 20 größten deutschen Städten.

Für Ihren Aufenthalt in Nürnberg wünsche ich Ihnen interessante und anregende Diskussionen sowie viele neue inspirierende und ertragbringende Kontakte.

Ich hoffe, Sie finden neben dem vielfältigen Kongressprogramm ein wenig Zeit, Nürnberg näher kennenzulernen. Auch abseits des Kongressgeschehens bietet Nürnberg seinen Gästen viel Sehenswertes.

Ihnen Allen wünsche ich erfolgreiche Konferenztage und einen angenehmen Aufenthalt in Nürnberg.

Dr. Michael Fraas

Wirtschaftsreferent der Stadt Nürnberg

Conference Chairs:

Tilman Börner, Chefredakteur, dotnetpro

Tilman Börner ist seit 2003 Chefredakteur der dotnetpro. Er programmiert seit der Schulzeit in Basic, dBASE, Turbo Pascal, Visual Basic, Fortran, C, Delphi, PHP und C#.

Dr. Markus Stäuble, freier Berater und Autor

Dr. Markus Stäuble ist passionierter Informatiker, Conference Chair der Developer Week, Fachautor und Programmleiter Make beim Franzis Verlag. Neben Make beschäftigt er sich intensiv mit dem Thema Mobile und hat zu dessen Auswirkungen auf die Arbeitswelt promoviert.

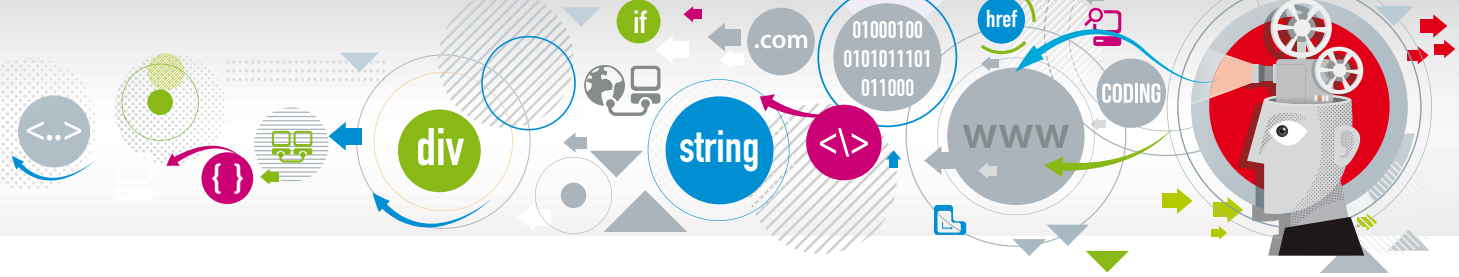
Advisory Board & Track Chairs:

Gregor Biswanger
Cross-Plattform

Robert Eichenseer
Cloud/Server

Robert Misch
Softskills

Björn Schotte
Agile



Programm-Übersicht Developer Week 2016

Montag, 20.06.2016	Dienstag, 21.06.2016	Mittwoch, 22.06.2016	Donnerstag, 23.06.2016
Konferenz	**Konferenz**	**Konferenz**	**Workshops**
■ Architektur ■ Datenzugriff ■ Best Practices ■ Frontend ■ DevOps ■ Projektpraxis	■ .NET Allgemein ■ Web ■ Funktional ■ Entity Framework ■ ALM ■ Cloud/Server	■ Softwarequalität ■ DevTest ■ Pattern ■ Tools ■ Softskills ■ Agile	■ Radikale Objektorientierung für die agile Softwareproduktion ■ Entwickeln für die Cloud ■ Parallelprogrammierung mit der TPL ■ Architektur für .NET-Projekte ■ Von den Anforderungen zum Code – Flüssige Softwareentwicklung ■ Angular 2: Einstieg in die komponentenbasierte Entwicklung von Single-Page-Anwendungen ■ Web Components ■ Schnelleinstieg UWP Apps unter Windows 10 ■ Team Foundation Server 2015: Ohne Stress zu häufigen Releases ■ Modernes JavaScript mit ECMAScript 2015
■ Industry 4.0 ■ JavaScript ■ DevOps ■ Projektpraxis	■ Angular 2 ■ Performance ■ ALM ■ Cloud/Server	■ UI/UX ■ Better Coding ■ Softskills ■ Agile	
■ Cross-Plattform ■ Testing ■ DevOps ■ Projektpraxis	■ Better Apps ■ Datenbanken ■ ALM ■ Cloud/Server	■ Responsive ■ Mobile Architekturen ■ Softskills ■ Agile	

Jan Fellien
ALM, Performance

Dr. Ronald Hartwig

Johannes Hoppe
Angular 2, JavaScript

André Krämer
Datenzugriff

Patrick Lobacher
Industry 4.0, Responsive

Hendrik Lösch
Best Practices, Design Pattern

Ulrike Stirnweiß
Frontend

David Thömmes
UI/UX

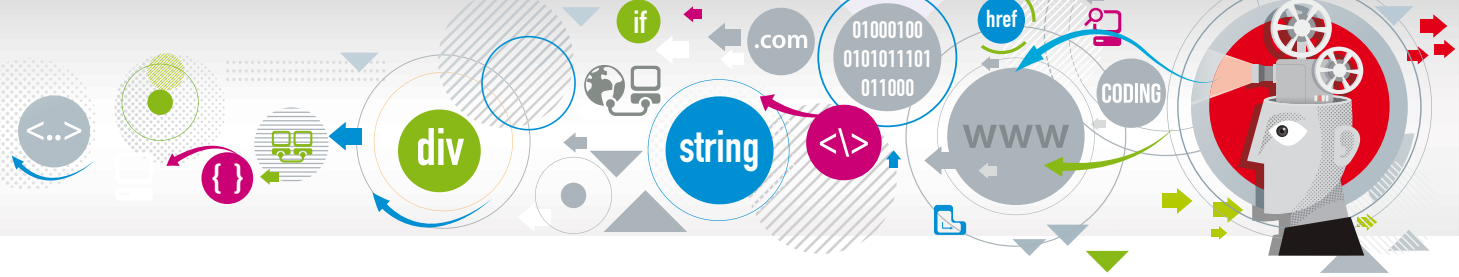
David Tielke
Architektur, Softwarequalität

Boris Wehrle
DevOps, DevTest

Holger Wendel
Better Coding, Tools

Programm – Tag 1: Montag, 20. Juni 2016

	Architektur	Datenzugriff	Best Practices	Frontend	DevOps
09.00 - 10.00	Keynote: Managing for Happiness *Jurgen Appelo*				
10.00 – 10.30	Kaffeepause				
10.30 – 11.30	Composite Components Architecture *David Tielke*	Performance trotz Entity Framework *André Krämer*	Was ist eigentlich eine Unit *Hendrik Lösch*	Datenbindung Deluxe – Deep Dive in das Binding von WPF *Christian Giesswein*	DevOps – Entwickelst du noch oder lieferst du schon? *Thomas Schissler*
11.30 – 11.45	Raumwechsel				
11.45 – 12.45	Flow Based Development – was war – was ist – was wird *Stefan Dirschnabel*	Daten vor Hackern schützen *N.N.*	KISS, im Prinzip ganz einfach *Jan Fellien*	The WPF First Aid Kit *David Würfel*	Entwurf und Bereitstellung von Software für DevOps-Organisationen *Michael Kaufmann*
12.45 – 14.15	Mittagspause				
14.15 – 15.15	Etwas zum Anfassen *Veikko Krypczyk*	C# + SQL = Big Data *Sascha Dittmann*	Legacy Code. Und jetzt? *Daniel Marbach*	One Code to rule 'em all – Die Windows Universal Platform *Marco Richardson*	Wenn jede Sekunde zählt – Unternehmen werden zu Software Firmen und Zeit ist hier bares Geld *Christian Karl-Heinz Nink*
15.15 – 15.30	Raumwechsel				
15.30 – 16.30	Modulares UI – MVVM mit Prism 6 *Christian Giesswein*	In-Memory Data Grids – supercomputing for the rest of us *Ralph Winzinger*	Automatisierte Erkennung der Top .NET und Web Performance Fehler in der CI *Andreas Grabner*	Von A wie Accelerometer bis X wie XmlLite – Das Windows 10 SDK für UWP Apps *Alexander Witkowski*	Monitoring 2.0: Alles im Lot? *Nico Orschel* *Marc Müller*
16.30 – 17.00	Kaffeepause				
17.00 – 18.00	Event Storming *Marco Heimeshoff*	SQL Server 2016 für Entwickler *Constantin Klein*	The Impact of Gherkin *Albert Schlotter, Maik Devrient*	Beyond XAML and HTML *David C. Thömmes*	Metrik gesteuertes DevOps und Continuous Delivery *Andreas Grabner*
ab 18.00	#nightone				
	Am ersten Abend der Developer Week gibt es ab 18.00 Uhr neben Speis und Trank auch ein buntes Programm für alle Konferenzbesucher. Diese Programmpunkte sind aktuell geplant: 19.00 Uhr: Thementische, 20.00 Uhr: Xamarin-Night, ab 20.00 Uhr: Ant.Me – C# spielerisch (Hendrik Lösch)				

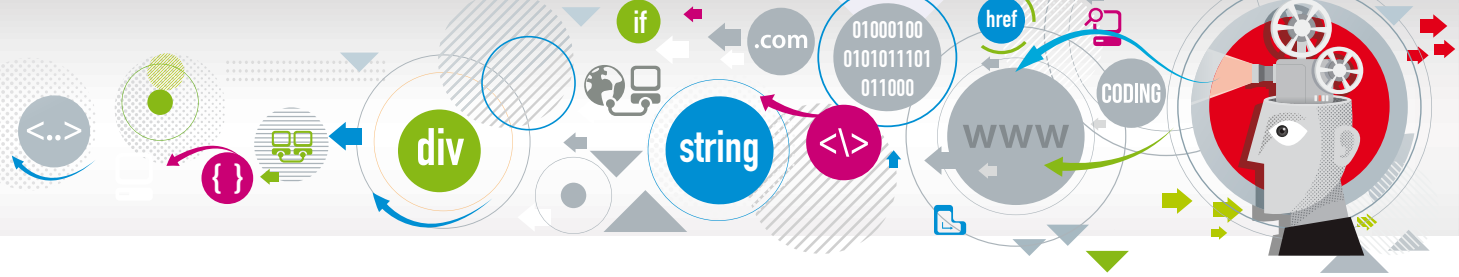


Projektpraxis	Industry 4.0	JavaScript	Cross-Plattform	Testing	
		Keynote: Managing for Happiness *Jurgen Appelo*			09.00 - 10.00
		Kaffeepause			10.00 – 10.30
Visualisierung von Software-Entwicklungsarbeit mit einem flexiblen Dashboard *Christian Ringler*	Smart Development in the Industry 4.0 – Best Practices and Prospects *Alexander Schulze*	ECMAScript 2015/2016: Neues aus der JavaScript-Welt *Marius Schulz*	Xamarin oder Cordova – das ist hier die Frage! *Ulrike Stirnweiß, Nina Hauer*	UI Testing Strategies *Daniel Kersting*	10.30 – 11.30
		Raumwechsel			11.30 – 11.45
Von Programmierer bis Projektleiter: Eine Sprache sprechen *Kurt Salman*	IoT Ultimate Edition *Damir Dobric*	State of Node.js *Sebastian Springer*	Aber schnell! Mehr Speed für Cross-Plattform-HTML5-Anwendungen *Gregor Biswanger*	Layout-Testing: Was geht? Was bringt`s? Wer braucht`s? *Christiane Helmchen, Bianca Niestroj*	11.45 – 12.45
		Mittagspause			12.45 – 14.15
Agile Projekte ohne Scrum *Ina Einemann, Ulf Mewe*	Innovationsmotoren für IoT *Patrick Lobacher*	Enhance applications with Location Data through the HERE JavaScript API *Richard Süselbeck*	Daten-Synchronisation für Apps – Offline und ohne Cloud *Christian Liebel, Marco Frodl*	Practical Mutation Testing *Sven Ruppert*	14.15 – 15.15
		Raumwechsel			15.15 – 15.30
Wettbewerbsvorteile durch den richtigen Umgang mit technischen Schulden *André Kolell*	Industrie 4.0-Lösungen für Ihre Kunden und sich selber liefern lassen *Lars Roith*	Testing unchained *Dominik Ehrenberg*	Ionic Apps mit Cloud Backend Services – Ein Dreamteam *Simon Martin Reimler*	Webtests automatisieren *N.N.*	15.30 – 16.30
		Kaffeepause			16.30 – 17.00
Work just simply smarter – für bessere Projekte und zufriedene Beteiligte *N.N.*	(UML) Modell-basiert Code generieren und debuggen – ja es funktioniert wirklich! *Daniel Sigl*	Domain-Driven Design mit JavaScript *Gregor Biswanger*	Introduction to NativeScript *Sebastian Witalec, Johannes Hoppe*	Test-Gap-Analyse: Erfahrungen aus drei Jahren Praxiseinsatz *Dennis Pagano*	17.00 – 18.00
		#nightone			ab 18.00
Am ersten Abend der Developer Week gibt es ab 18.00 Uhr neben Speis und Trank auch ein buntes Programm für alle Konferenzbesucher. Diese Programmpunkte sind aktuell geplant: 19.00 Uhr: Thementische, 20.00 Uhr: Xamarin-Night, ab 20.00 Uhr: Ant.Me – C# spielerisch (Hendrik Lösch)					

Programmänderung vorbehalten

Programm – Tag 2: Dienstag, 21. Juni 2016

	.NET Allgemein	Web	Funktional	Entity Framework	ALM
09.00 – 10.00	Visual Studio 2015 Geheimtipps *Manuel Meyer*	Web APIs mit ASP.NET MVC 6 *Manfred Steyer*	Einführung in die funktionale Programmierung mit F# *Carsten König*	Entity Framework hinter den Kulissen *André Krämer*	Docker 101 *Lars Kumbier*
10.00 – 10.30	Kaffeepause				
10.30 – 11.30	akka.net – Einführung in das Actor Model mit .NET *Ralph Waldenmaier*	NancyFX: Das Framework für die Web-Entwicklung *N.N.*	DDD mit Funktionalen Sprachen *Marco Heimeshoff*	Plattformunabhängiger Datenzugriff mit Entity Framework Core 1.0 *Dr. Holger Schwichtenberg*	Mit fünf Schritten effizienter Software entwickeln *Karsten Kempe*
11.30 – 11.45	Raumwechsel				
11.45 – 12.45	Continuous Delivery for .NET Projects at adidas *Paul Vassu, Daniel Eichten*	Enabling Plugins in your web application with MEF *Don Wibier*	Event Sourcing funktional *Carsten König*	Datenbasierte Services mit Entity Framework und Co. *Manfred Steyer*	Build und Release Automation mit TFS 2015 / VSTS *Marc Müller, Nico Orschel*
12.45 – 14.15	Mittagspause				
14.15 – 16.15 DevSessions	.NET Core *David Tielke*	Mit SignalR Datenverbindungen zwischen Server und Client herstellen *N.N.*	Funktionale und asynchrone Pipelines mit C# *Daniel Marbach*	Entity Framework: Tipps & Tricks *Dr. Holger Schwichtenberg*	Qualität von Anfang an - Best Practices zur Integration von agilem Testen aus realen Projekten *Holger Hofmeister*
16.15 – 16.45	Kaffeepause				
16.45 – 18.45 DevSessions	WPF Troubleshooting in Visual Studio 2015 *Manuel Meyer*	Datenlieferant: GraphQL mit .NET nutzen *Philip Jander*	Systematische Entwicklung mit funktionaler Programmierung *Michael Sperber*	Entity Framework und WPF *Christian Giesswein*	Schnellere Build-Prozesse mit IncrediBuild *Robin Sedlaczek*
ab 19.00	#community-night				
	Mitmachen und Spaß haben erwünscht! Am Abend des 21.06.2016 öffnet die DWX ab 19.00 Uhr ihre Türen und lädt alle Entwickler und Freunde ein zur #communitynight mit abwechslungsreichem Programm. Auch selbst mitmachen ist erwünscht. Das ausführliche Programm finden Sie auf der Webseite.				



Cloud/Server	Angular 2	Performance	Better Apps	Datenbanken	
„Backend vNext" mit Microsoervices und Actor Programming Model *Damir Dobric*	Speed-up AngularJS – Hochperformante Webanwendungen bauen *Timo Korinth*	HTTP/2: Die neue Generation für Performance *Sascha Schumann*	Mobile App-Entwicklung im Team: Tools und Techniken *Robert Virkus*	o3db – Datenbankanwendungen fürs Web in Scheme *Michael Köhne*	09.00 – 10.00
Kaffeepause					10.00 – 10.30
Connecting local data and the cloud – Hybride Lösungen mit Microsoft Azure *Sebastian Achatz*	Live-Coding mit Angular 2 *Johannes Hoppe*	Design the Priority: Performance und UX *Peter Rozek*	Mobile Friendly: Fluch oder Segen? *Thomas Kaiser, Paul Dougherty*	Automatisierte Datenbankänderungen – Vergleich der Open-Source-Tools Flyway und LiquiBase *N.N.*	10.30 – 11.30
Raumwechsel					11.30 – 11.45
Azure: ein Überblick über die Möglichkeiten der Cloud *N.N.*	Angular 2 Change Detection Explained *Pascal Precht*	JavaScript Performance Analyse *Sebastian Springer*	Ist Ihre App sicher? *Kerry W. Lothrop*	Database Lifecycle Management – die nächste Stufe für Ihre Datenbankentwicklung *Constantin Klein*	11.45 – 12.45
Mittagspause					12.45 – 14.15
In wenigen Minuten zur Serverlandschaft für meine Apps *Sascha Dittmann*	Schnellstart mit Angular 2 *Johannes Hoppe, Gregor Woiwode*	Performance & Load Testing mit Visual Studio richtig gemacht ... *Nico Orschel, Marc Müller*	UWP-Apps – das eierlegende Wollmilch-UI für Windows 10 *Peggy Reuter-Heinrich, Lars Heinrich*	SQL Server – Sicherheitkonzepte *Thorsten Kansy*	14.15 – 16.15 DevSessions
Kaffeepause					16.15 – 16.45
Containers, Service Fabric and Azure – Globally scalable, available architectures in Cloud Environments *Robert Eichenseer*	Angular 2 - Upgrade *Manfred Steyer*	Reaktive Web-Anwendungen mit RxJS *Michael Menzel*	Performancetuning von Apps *N.N.*	NoSQL mit PostgreSQL *Stephan Hochdörfer*	16.45 – 18.45 DevSessions
#community-night					ab 19.00

Mitmachen und Spaß haben erwünscht! Am Abend des 21.06.2016 öffnet die DWX ab 19.00 Uhr ihre Türen und lädt alle Entwickler und Freunde ein zur #communitynight mit abwechslungsreichem Programm. Auch selbst mitmachen ist erwünscht. Das ausführliche Programm finden Sie auf der Webseite.

Programmänderung vorbehalten

Programm – Tag 3: Mittwoch, 22. Juni 2016

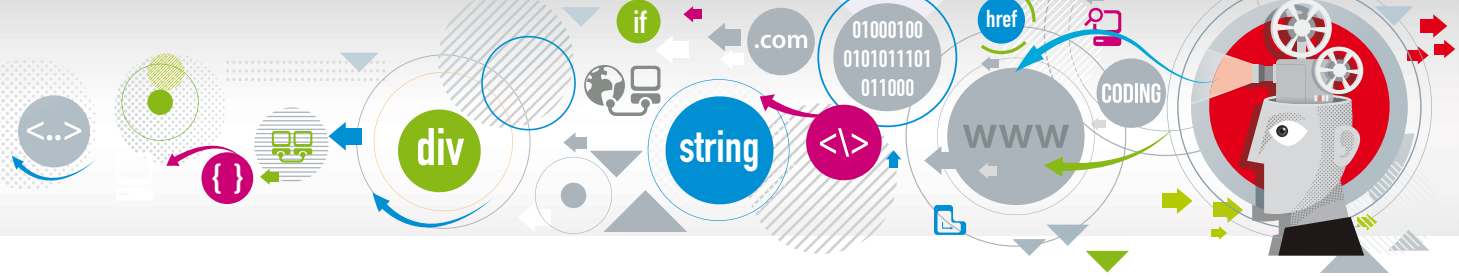
	Softwarequalität	DevTest	Pattern	Tools	Softskills
09.00 – 10.00	Softwarequalität *David Tielke*	Eigenschafts-basierendes Testen *Carsten König*	Lose gekoppelt wie nie: DI vs. IoC *Hendrik Lösch*	Visual Studio 2015 Extensions leichtgemacht *Christian Giesswein*	Innovativ, kreativ und begeisternd – aber wie? *Anja Schwarz*
10.00 – 10.30	Kaffeepause				
10.30 – 11.30	(Aus-)Kommentiert *Michael Wiedeking*	Testmanagement in der agilen Transition *Kay Grebenstein*	Async/Await. Die Würfel sind gefallen! *Daniel Marbach*	Zehn kostenfreie Visual Studio-Extensions, die Sie kennen sollten *Dr. Holger Schwichtenberg*	Von Ärzten, Piloten und Flugzeugen: Mehr Effizienz in der Softwareentwicklung *Christian Robert*
11.30 – 11.45	Raumwechsel				
11.45 – 12.45	Die S.O.L.I.D.-Prinzipien für C#-Entwickler *Thomas Claudius Huber*	Unit Testing Reifegrade *Frank Sons*	Microservices patterns with Spring Cloud *Paul Vassu, Daniel Eichten*	Ein Blick in meinen Werkzeugkasten für Desktop-anwendungen *Hendrik Lösch*	Einmal mit Profis arbeiten *Martin Walter*
12.45 – 14.15	Mittagspause				
14.15 – 15.15	.NET Native *David Tielke*	Testend Entwickeln – Entwickelnd Testen *Martin Uhlig, Michael Thiele*	Practical Proxy DeepDive *Sven Ruppert*	AMP – Eine Biblio-thek für einfachen Zugriff auf die GPU *Bernd Marquardt*	Auf der Suche nach dem perfekten Mitarbeiter oder vom T-Shape zum Team-Shape *Ulf Mewe*
15.15 – 15.30	Raumwechsel				
15.30 – 16.30	Planlos mit Plan – wie erhöhe ich die Verlässlichkeit der Planung in der Soft-wareentwicklung? *Frank Düsterbeck*	Entwicklerproduktivi-tät mit ALM Rangers Solutions für TFS & Visual Studio Real Live Scenario *Johannes Cosmin Dumitru*	Digitale Transformation *Golo Roden*	Office 365 – Die USB-Plattform von Microsoft *Alexander Tews*	Karriere in der IT und Informatik - Wie gelangen Fachkräfte an ihr Ziel? *Alexander Rabe*
16.30 – 17.00	Kaffeepause				
17.00 – 18.00	Abschluss-Keynote: JavaScript, The Web, Mobile, and the Rise of the New Virtual Machine *Scott Hanselman*				

Workshop

Workshop: ASP.NET Core 1.0 and .NET Core 1.0

Speaker: **Scott Hanselman** **Workshopdauer:** **09.00-16.00 Uhr**

Learn in this workshop how and why ASP.NET has been rebuilt from the ground up to be the best web development platform for Windows, Mac, and Linux. Get an introduction to the major advancements and an overview of what's changed and what's stayed the same.



Agile	UI/UX	Better Coding	Responsive	Mobile Architekturen	
Agil und Konzern - Das kannste schon so machen, aber...? *Dominik Ehrenberg, Sebastian Bauer*	User Experience Design und Usability Brandstifter *David C. Thömmes*	Continuous Quality – Gleichbleibende Qualitätsstandards durch (voll-)automatisierte Releaseprozesse sicherstellen *Thomas Rümmler*	Full Responsive *Sven Wolfermann*	Native Mobile Apps mit NativeScript und Angular 2.0 *Johannes Hoppe, Sebastian Witalec*	09.00 – 10.00
Kaffeepause					10.00 – 10.30
Alles wird gut – wir machen jetzt Scrum! *Julia Schmidt*	„Was würde Einstein sagen?" – Personas als Werkzeug zur Verbesserung der Gebrauchstauglichkeit von interaktiven Systemen *Armin Reuter*	Softwareentwicklung ohne Abhängigkeiten *Stefan Lieser*	Atomic Design – Die Einheit von Frontend und Design im RWD-Zeitalter *Patrick Lobacher*	MVVM und FRP: Perfekte Architekturen für komplexe mobile Anwendungen? *Dr. Wolfram Schroers*	10.30 – 11.30
Raumwechsel					11.30 – 11.45
Gerade genug Architektur vorneweg – Zur eigenen Architektur-Vision in 16 Minuten *Roland Mast*	Turn your ideas into reality – Prototyping in der Praxis *Ulrike Stirnweiß, Nina Hauer*	Demystifying automated testing in enterprise projects *Denis Biondic*	Responsive Design: Der Inhalt entscheidet, nicht die Auflösung. *Daniel Kersting*	Mobile Anwendungen synchronisieren mit Immutability *Michael Sperber*	11.45 – 12.45
Mittagspause					12.45 – 14.15
Agile Methoden in verteilten Teams – Was hat sich bewährt, was nicht? *Thomas Rümmler*	Sie sind Software Engineer? Dann sind Sie UX Designer! *Alexander Keller*	Asynchrone Programmierung 101 *Golo Roden*	Mobile First *N.N.*	Architekturen für das Mobile Enterprise *Thomas Künneth*	14.15 – 15.15
Raumwechsel					15.15 – 15.30
Lessons learned: Sprung von einer kleinen Firma in ein stark wachsendes Unternehmen *Sebastian Hoitz*	WPF und Parallelprogrammierung *Bernd Marquardt*	Refresh: Migration einer Legacy-Anwendung *Veikko Krypczyk*	Responsive Web Design for Developers *Don Wibier*	Advanced Mobile Cross-Platform Architecture *Jens Becker*	15.30 – 16.30
Kaffeepause					16.30 – 17.00
Abschluss-Keynote: JavaScript, The Web, Mobile, and the Rise of the New Virtual Machine *Scott Hanselman*					17.00 – 18.00

Programmänderung vorbehalten

DWX Developer Week 2016

Workshops

Workshops – Tag 4: Donnerstag, 23. Juni 2016

Workshop 1

Radikale Objektorientierung für die agile Softwareproduktion

Workshopleiter: **Ralf Westphal** **Workshopdauer:** **09.00-17.00 Uhr**

Die Objektorientierung hatte versprochen, die Entwicklung von Softwaresystemen zu verbessern. Aber Technical debt, brownfield, legacy code belasten die meisten Projekte – allerdings nicht trotz sondern eher wegen ihrer Objektorientierung. Im Workshop lernen Sie eine radikale Objektorientierung kennen. Radikal, weil sie zurückgeht an die Wurzeln. Dort findet sich nämlich das zentrale Merkmal für Objektorientierung. Messaging ist der Schlüssel zu vielen Tugenden guten Softwaredesigns wie Entkopplung, Testbarkeit, Kapselung, klaren Verantwortlichkeiten und sogar agilem Vorgehen. Als Teilnehmer lernen Sie die wesentlichen Prinzipien radikaler Objektorientierung kennen und wenden sie in mehreren Beispielprojekten an. Und Sie erfahren, wie ein Umgang mit Anforderungen aussehen muss, um die radikale Objektorientierung nahtlos in agiles Vorgehen einzubetten.

Workshop 2

Entwickeln für die Cloud

Workshopleiter: **Golo Roden** **Workshopdauer:** **09.00-17.00 Uhr**

Die Cloud ist die Zukunft, das steht außer Frage. Doch der Weg ist steinig. Sie benötigen REST-APIs, Streaming und Echtzeitdaten. Das alles skalierbar, verlässlich und effizient. Für welche Technologie entscheiden Sie sich? In diesem Workshop lernen Sie die Plattformen Node.js und io.js kennen, die JavaScript auf dem Server ausführen und so Ihren Aufwand verringern und Ihre Entwicklung beschleunigen.

Workshop 3

Parallelprogrammierung mit der TPL (Task Parallel Library)

Workshopleiter: **Bernd Marquardt** **Workshopdauer:** **09.00-17.00 Uhr**

Da die Taktfrequenzen unserer Prozessoren nicht ansteigen, ist die Parallelprogrammierung fast „zur Pflicht" geworden. Hier helfen moderne Parallel-Bibliotheken damit unsere Anwendungen durch die Parallelisierung nicht unendlich kompliziert werden. Sie lernen die Schleifenparallelisierung, die Task-orientierte Programmierung, die parallelen Container-Klassen, die Fehlerbehandlung und viele andere Themen aus der TPL kennen. Es wird auch auf die „unangenehmen" Eigenschaften der Parallelprogrammierung eingegangen. Hierzu gehören u.a. Data Races und Dead Locks. Viele Live-Beispiele, an denen das neue Wissen direkt beobachtet werden kann, lockern den Workshop auf.

Workshop 4

Architektur für .NET-Projekte

Workshopleiter: **David Tielke** **Workshopdauer:** **09.00-17.00 Uhr**

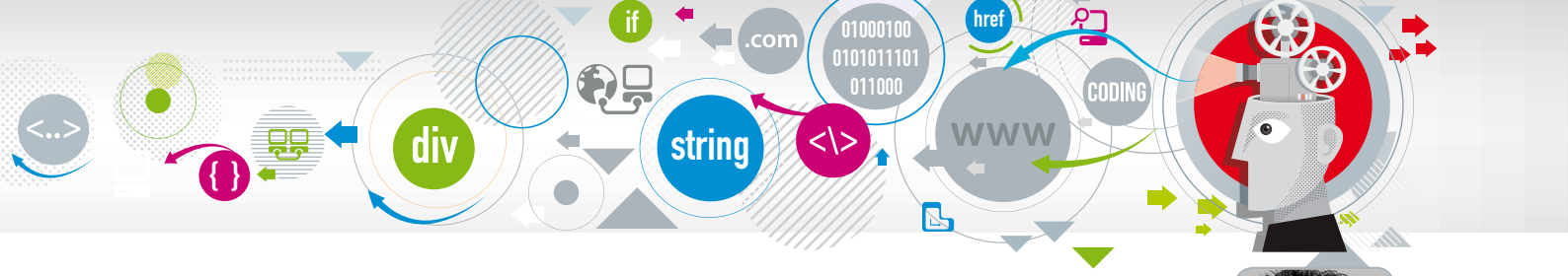
Ein Architekt sollte in jedem großen Softwareprojekt unter anderem für die Struktur der Anwendung sorgen und diese an den nichtfunktionalen Anforderungen ausrichten. Soweit so gut. Leider ist nicht jedes Projekt von solcher Größe und bei einem kleinen Team existiert nur selten ein dedizierter Softwarearchitekt. Zu oft wird die Notwendigkeit einer ordentlichen Softwarearchitektur in solchen Projekten nicht gesehen, oft mit fatalen Folgen: auf lange Sicht haben solche Projekte Probleme bei der Wartbarkeit, Weiterentwickelbarkeit und vielen anderen Eigenschaften. Wir schauen in diesem Workshop, wie Sie diese Aufgabe ohne Architekturzertifikat meistern.

Workshop 5

Von den Anforderungen zum Code – Flüssige Softwareentwicklung

Workshopleiter: **Stefan Lieser** **Workshopdauer:** **09.00-17.00 Uhr**

Softwareentwickler werden mit Anforderungen konfrontiert und müssen Code produzieren. Doch wie geht man als Softwareentwickler dabei vor? Der Workshop zeigt anhand zahlreicher Beispiele, wie Anforderungen so zerlegt werden, dass konkrete handhabbare Einheiten entstehen, die anschließend entworfen und umgesetzt werden können. Sie lernen, wie Anforderungen in konkrete umsetzbare Einheiten zerlegt werden. Wie Sie eine Lösung für die Anforderungen entwerfen und wie Sie die entworfene Lösung in Quellcode übersetzen. Ziel dieser Vorgehensweise ist ein flüssiger Produktionsprozess, der es erlaubt, Software als Einzelner oder auch im Team von mehreren Entwicklern zu produzieren. Die Werte der CCD-Initiative (Evolvierbarkeit & Korrektheit) werden in den Blick genommen.



Workshop 6

Angular 2: Einstieg in die komponentenbasierte Entwicklung von Single-Page-Anwendungen

Workshopleiter: **Johannes Hoppe, Gregor Woiwode** **Workshopdauer:** **09.00-17.00 Uhr**

AngularJS ist das populärste Framework für Single-Page-Anwendungen. Mit Angular 2 mischt Google die Webentwicklung gehörig auf. Das neue Framework setzt auf die Programmiersprache TypeScript, komponentenbasierte Entwicklung, eine neue Template-Syntax und ein verbessertes Tooling. Kurzum: alles ist neu! Unter Anleitung der beiden Angular-Experten lernen Sie folgende Themen kennen: Typescript, Setup einer Angular 2 App, Template-Syntax, Komponenten, Pipes & Direktiven, Navigation mit dem neuem Component-Router, Integration von Polymer & Kendo UI, Tooling, Unit-Test und Oberflächentests. Zum Ende des Workshops haben Sie die Grundlagen und fortgeschrittene Themen von Angular 2 gemeistert. Als Teilnehmer benötigen Sie einen Laptop mit vorinstallierter Software (Chrome, Atom Editor, Node.js) oder Virtual Box.


Johannes Hoppe


Gregor Woiwode

Workshop 7

Web Components

Workshopleiter: **Peter Kröner** **Workshopdauer:** **09.00-17.00 Uhr**

Das Webseiten-Modul der Zukunft ist kein jQuery-Plugin mehr, sondern ein selbstdefiniertes HTML-Element. Web Components werden die Entwicklung von Web-Frontend revolutionieren – und mit den richtigen Tools kann die Revolution bereits heute beginnen. Dieser Workshop gibt einen Einblick in die Webstandards hinter Web Components und zeigt Ihnen, wie Sie bequem eigene HTML-Elemente erfinden können. Außerdem führt der Workshop in Googles Polymer-Library für Web Components ein und gibt Ihnen in einem großen Praxisteil die Gelegenheit, erste Hands-On-Erfahrung mit Web Components zu sammeln. Für den Workshop werden ein Laptop mit modernen Browsern (Chrome, Firefox), einem beliebigen Code-Editor und einem beliebigen lokal installierten Webserver benötigt.

Workshop 8

Schnelleinstieg UWP Apps unter Windows 10

Workshopleiter: **Alexander Witkowski** **Workshopdauer:** **09.00-17.00 Uhr**

Der Workshop gibt einen Einstieg in die Entwicklung von Apps für Windows 10. Nur eine App mit nur einem SDK läuft auf den unterschiedlichsten Plattformen. Vom Raspberry Pi über das Telefon, das Tablet bis zum Desktop PC und der Xbox. Begonnen bei den notwendigen Werkzeugen wird, neben der Entwicklung selbst, ein Blick auf die Richtlinien und das verteilen der Anwendungen über den Store geworfen. Auch der Zugriff auf Sensoren, das Verwenden einer Datenbank und das MVVM Pattern kommen im Workshop zum Einsatz. Am Ende des Tages gehen Sie mit einer Universal Windows Plattform App nach Hause und wissen, wie Sie diese weiterentwickeln und in den Store bringen.

Workshop 9

Team Foundation Server 2015: Ohne Stress zu häufigen Releases

Workshopleiter: **Neno Loje** **Workshopdauer:** **09.00-17.00 Uhr**

In diesem interaktiven Workshop lernen Sie, wie Sie mit dem Team Foundation Server (TFS) den Prozess von der Anforderung, über Implementierung, Test bis hin zur Auslieferung effizient und schlank gestalten. Um häufige Releases zu ermöglichen bietet der TFS vielfältige Möglichkeiten der Automatisierung. In einer End-to-End-Demo wird auf die verschiedenen Aspekte eingegangen, um zeitnah auf Kunden und Marktanforderungen reagieren zu können. Freuen Sie sich auf einen lehrreichen und unterhaltsamen Tag zusammen mit dem TFS-Experten & MVP Neno Loje. Und: Bringen Sie gerne Ihre Fragen mit!

Workshop 10

Modernes JavaScript mit ECMAScript 2015

Workshopleiter: **Marius Schulz** **Workshopdauer:** **09.00-17.00 Uhr**

Mit ECMAScript 2015, früher ECMAScript 6, wurde JavaScript nach vielen Jahren erheblich ausgebaut. Die umfangreichste Erweiterung ist die Einführung eines nativen Modulsystems, mit dem sich auch größere Anwendungen sauber strukturieren lassen. Des Weiteren erleichtern diverse kleinere Features die Arbeit mit JavaScript. In dem Workshop werden u.a. die folgenden Sprachkonstrukte von ECMAScript 2015 behandelt: Natives Modulsystem, Konstanten und Block-Scoping, Arrow-Funktionen und lexikalisches „this"-Binding, Destrukturierung von Objekten und Arrays, Default-Werte für Parameter, Template-Strings und String-Interpolation, Rest- und Spread-Operator, Klassen und Vererbung, Iteratoren und Iteration. Zusätzlich lernen Sie einen Transpiler einzusetzen, der die neuen Sprachkonstrukte so umschreiben kann, dass sie von allen gängigen Browsern verstanden werden.

Die Referenten der Developer Week

Achatz, Sebastian
Appelo, Jurgen, Jojo Ventures BV
Bauer, Sebastian, Inovex GmbH
Becker, Jens, Incloud GmbH
Biondic, Denis, conplement AG
Biswanger, Gregor, CleverSocial.de
Devrient, Maik, DATEV eG
Dirschnabel, Stefan, Codelution
Dittmann, Sascha, Microsoft Deutschland GmbH
Dobric, Damir, Daenet GmbH
Dougherty, Paul, Forecheck LLC
Dumitru, Johannes C.
Düsterbeck, Frank, HEC GmbH
Ehrenberg, Dominik, Infineon Technologies AG
Eichenseer, Robert, Microsoft Corp.
Eichten, Daniel, Adidas-Salomon AG
Einemann, Ina, HEC GmbH
Fellien, Jan, devCrowd GmbH
Frodl, Marco, Thinktecture AG
Giesswein, Christian, Giesswein Creativ GmbH
Götz, Malte
Grabner, Andreas, Dynatrace, Austria
Grebenstein, Kay, Saxonia Systems AG
Hanselman, Scott, Microsoft Corp.
Hauer, Nina, conplement AG
Heimeshoff, Marco, Heimeshoff IT
Heinrich, Lars, Heinrich & Reuter Solutions GmbH
Helmchen, Christiane, punkt.de GmbH
Hochdörfer, Stephan, bitExpert AG
Hofmeister, Holger, conplement AG
Hoitz, Sebastian, komola GmbH
Hoppe, Johannes, HAUS HOPPE - ITS
Huber, Thomas, Trivadis AG
Jander, Philip, Jander IT
Kaiser, Thomas, cyberpromote GmbH
Kansy, Thorsten,
DotNetConsulting by Thorsten Kansy
Kaufmann, Michael,
Alegri International Service GmbH
Keller, Alexander, Centigrade GmbH
Kempe, Karsten, KWP Projekt-Vertriebs GmbH
Kersting, Daniel, User Interface Design GmbH
Klein, Constantin, Freudenberg IT GmbH & Co. KG
Köhne, Michael, o3db.com
Kolell, André,
FFG Finanzcheck Finanzportale GmbH
König, Carsten, Wiegand-Glas GmbH
Korinth, Timo, MAXIMAGO GmbH
Krämer, André, André Krämer - Software, Training & Consulting
Kröner, Peter, Brainfire Design
Krypczyk, Dr. Veikko, internationale Berufsakademie
Kumbier, Lars, Kumbier IT Consulting
Künneth, Thomas, MATHEMA Software GmbH
Liebel, Christian, Thinktecture AG
Lieser, Stefan
Lobacher, Patrick, +Pluswerk AG
Loje, Neno, NenoLoje.de
Lösch, Hendrik, Saxonia Systems AG
Lothrop, Kerry W., Zühlke Engineering GmbH
Marbach, Daniel, tracelight GmbH
Marquardt, Bernd, Bernd Marquardt
Mast, Roland, Sybit GmbH
Menzel, Dr. Michael, Senacor Technologies AG
Mewe, Ulf, HEC GmbH
Meyer, Manuel, Trivadis AG
Müller, Marc, 4tecture GmbH
Niestroj, Bianca, punkt.de GmbH
Nink, Christian, New Relic International Ltd.
Orschel, Nico, AIT GmbH & Co. KG
Pagano, Dr. Dennis, CQSE GmbH
Precht, Pascal, thoughtram GmbH
Rabe, Alexander, Gesellschaft für Informatik e.V. (GI)
Reimler, Simon Martin,
arvato Bertelsmann SE & Co. KGaA
Reuter, Armin, Heinrich & Reuter Solutions GmbH
Reuter-Heinrich, Peggy,
Heinrich & Reuter Solutions GmbH
Richardson, Marco, Microsoft Deutschland GmbH
Ringler, Christian,
GfK Retail and Technology GmbH
Robert, Christian, SapientNitro
Roden, Golo,
the native web ug (haftungsbeschränkt)
Roith, Lars, AIT GmbH & Co. KG
Rozek, Peter, Ecx.io germany GmbH
Rümmler, Thomas, AIT GmbH & Co. KG
Ruppert, Sven, Macros Reply GmbH
Salman, Kurt, CoMo Solutions GmbH
Schissler, Thomas, artiso solutions GmbH
Schlotter, Albert, DATEV eG
Schmidt, Julia, BERATUNG JUDITH ANDRESEN
Schroers, Dr. Wolfram,
Numerik & Analyse Schroers
Schulz, Marius, 69 Grad GmbH
Schulze, Alexander, Innotrade GmbH
Schumann, Sascha, Myra Security GmbH
Schwarz, Anja, SOPHIST GmbH
Schwichtenberg, Dr. Holger, www.IT-Visions.de
Sedlaczek, Robin, Fairmas GmbH
Siegl, Daniel, LieberLieber Software GmbH
Sons, Frank, code-qualiy.de
Sperber, Dr. Michael, Active Group GmbH
Springer, Sebastian, MaibornWolff GmbH
Steyer, Manfred, www.IT-Visions.de
Stirnweiß, Ulrike, conplement AG
Süselbeck, Dr. Richard, HERE
Tews, Alexander, conplement AG
Thiele, Michael, Saxonia Systems AG
Thömmes, David C., Shapefield UG
Tielke, David, david-tielke.de
Uhlig, Martin, Saxonia Systems AG
Vassu, Paul, Adidas Group
Virkus, Robert, Enough Software GmbH & Co.KG
Waldenmaier, Ralph,
PROGRESS SOFTWARE GmbH
Walter, Martin, Deutsche Welle
Westphal, Ralf, One Man Think Tank
Wibier, Don, Developer Express Inc.
Wiedeking, Michael, MATHEMA Software GmbH
Winzinger, Ralph, Senacor Technologies AG
Witalec, Sebastian, Telerik
Witkowski, Alexander, Marktjagd GmbH
Woiwode, Gregor, heco GmbH
Wolfermann, Sven, maddesigns
Würfel, David, Centigrade GmbH

SPIELEPROGRAMMIERUNG UNTER OPENGL (TEIL 2)

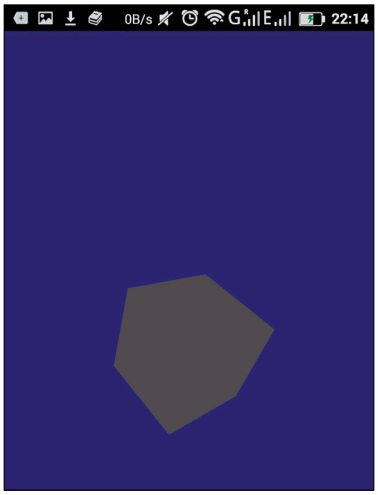
Shader-Programmierung

Einfarbige Objekte stellen den Einstieg in die Welt der Shader-Programmierung dar.

Bevor wir uns in die Programmierung vertiefen, sei ein kurzer Hinweis auf die Funktion des menschlichen Auges erlaubt. Das menschliche Auge arbeitet – anders als etwa das Radar eines Jagdflugzeugs – rein passiv: Es sieht nur das, was von der Umgebung reflektiert wird.

Das Feld des Raytracings versucht, dieses Verfahren zu modellieren. Per Raytracing lassen sich realistische Szenen errechnen. Leider ist es ob des immensen Rechenaufwands nicht möglich, die Verfahren zur Generierung von Grafiken für Computerspiele einzusetzen.

Als Alternative dazu haben sich Berechnungsmodelle etabliert, die das Einfallen des Lichts durch diverse mathematische Modelle nachzubilden suchen.



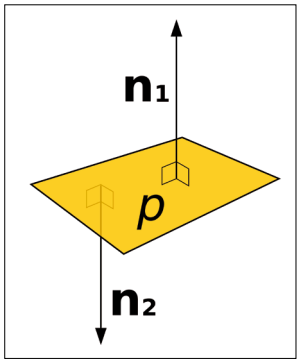
Ambient Light beleuchtet alle am Bildschirm befindlichen Elemente gleichermaßen (Bild 1)

Indirektes Licht

Wer die Umgebung seines Arbeitsplatzes sorgfältig betrachtet stellt fest, dass Licht aus diversen Quellen kommt. Der Monitor strahlt Licht ab, aus dem Fenster kommt ebenfalls etwas Sonnenlicht herein. Dieses Licht wird in der Welt der Shader als Ambient Light bezeichnet. Zur Simplifizierung geht man davon aus, dass es einfach immer da ist.

Mit diesem Wissen können wir uns an die Modifikation des Shaders aus dem vorangegangenen Artikel (**web & mobile developer** 5/2016, Seite 98) heranwagen. Der Korpus des Pixelshaders sieht nun folgendermaßen aus:

```
precision mediump float;
varying vec4 v_Color;
void main()
{
  gl_FragColor =
  vec4(0.1,0.1,0.1,1);
}
```



Wikimedia Commons / Elembis

Die Normale wächst aus dem jeweiligen Vertex heraus (Bild 2)

Wir modellieren Ambient Light durch die Annahme eines konstanten Helligkeitswerts, der im Rahmen des Pixelshaders jedem einzelnen Pixel der Objekte gleichermaßen eingeschrieben wird.

Da wir die nicht benötigten Parameter nicht entfernen, können wir den Code einfach zur Ausführung freigeben. Änderungen am Vertexshader sind ebenfalls nicht notwendig. Das Resultat ist das in Bild 1 gezeigte Negativ unseres Würfels. Für Ambient Light ist es definitionsgemäß irrelevant, in welchem Winkel die Szene zur Lichtquelle steht.

Lamberts Rache

Als nächste Quelle von Licht wollen wir uns jenen Lichtquellen zuwenden, die aus einer bestimmten Richtung in die Szene scheinen. Denken sie hier an eine beliebige Lampe, deren Licht – im Grunde genommen – als ein aus einer bestimmten Richtung kommendes Feld von Vektoren dargestellt werden kann.

Dieses in der Shaderfachsprache als Lambertian oder Directional Light bezeichnetes Licht ist für die Darstellung realistischer Grafikszenen von großer Bedeutung. Für seine Modellierung kommt unter anderem das Gesetz der Wellenbrechung zum Einsatz: Der Einfallswinkel eines Photons bei der Reflexion an einer Oberfläche entspricht jeweils dem Ausfallswinkel.

Zur effizienten Berechnung der diversen Lichtparameter benötigen wir ein als Normalvektor bezeichnetes Informationselement.

Es handelt sich dabei um einen Vektor, der – wie in Bild 2 gezeigt – aus der Fläche herauswächst und so für die Berechnung von Einfalls- und sonstigen Winkeln hilfreich ist.

Im Fall unseres per Open-GL realisierten Würfelmodells sehen die fehlenden Informationen des Würfels wie in Listing 1 aus. ▶

Listing 1: Würfelmodell

```
float[] cubeNormalData = {
  0.0f, 0.0f, 1.0f,
  0.0f, 0.0f, 1.0f,
  0.0f, 0.0f, 1.0f,
  0.0f, 0.0f, 1.0f,
  0.0f, 0.0f, 1.0f,
  0.0f, 0.0f, 1.0f,
  1.0f, 0.0f, 0.0f,
  1.0f, 0.0f, 0.0f,
  1.0f, 0.0f, 0.0f,
  1.0f, 0.0f, 0.0f,
  1.0f, 0.0f, 0.0f,
  1.0f, 0.0f, 0.0f,
  0.0f, 0.0f, -1.0f,
  0.0f, 0.0f, -1.0f,
  0.0f, 0.0f, -1.0f,
  0.0f, 0.0f, -1.0f,
  0.0f, 0.0f, -1.0f,
  0.0f, 0.0f, -1.0f,
  ...
```

cubeNormalData besteht – analog zu seinen Kollegen mit den Vertex- und Farbdaten – aus 36 dreiwertigen Elementen, die einen Normalvektor mit der jeweiligen Ausrichtung darstellen. Da die aus je sechs Punkten bestehenden Flächen des Quaders alle dieselbe Normale aufweisen, sind je sechs Elemente identisch.

Identische Elemente

Wie im vorangegangenen Teil besprochen, müssen wir auch diesmal einige Vorbereitungshandlungen treffen, um die Daten in ein für die Grafikpipeline bekömmliches Format zu bringen. Der FloatBuffer *myCubeNormals* ist hierbei als globaler Member der Rendererklasse deklariert und wird im Konstruktor folgendermaßen bevölkert:

```
myCubeNormals = ByteBuffer.
allocateDirect(cubeNormalData.length * 4).
order(ByteOrder.nativeOrder()).asFloatBuffer();

myCubeNormals.put(cubeNormalData).position(0);
}
```

Im Shader nehmen wir die Daten durch eine globale Variable entgegen, die im Header-Bereich des Shaders unter Nutzung des *attribute*-Systems angelegt wird:

```
uniform mat4 u_MVPMatrix;
attribute vec4 a_Position;
attribute vec4 a_Color;
attribute vec3 a_Normal;
```

Im Rendering-Pfad wird diese folgendermaßen bevölkert:

```
@Override
public void onDrawFrame(GL10 gl) {
  rotationStep+=1;
  ...
  int aNor = GLES20.glGetAttribLocation
  (myFinalProgramHandle, "a_Normal");

  myCubeNormals.position(0);
  GLES20.glVertexAttribPointer(aNor, 3,
  GLES20.GL_FLOAT, false, 0, myCubeNormals);
  GLES20.glEnableVertexAttribArray(aNor);
```

Nach diesen Vorbereitungshandlungen sind alle Daten am Platz und es ist an der Zeit, die Shaderpipeline zur Berechnung der neuen Lichtart aufzurüsten. Als Erstes wenden wir uns dem Vertexshader zu, dessen Code nun eine Gruppe weiterer Variablen aufweist:

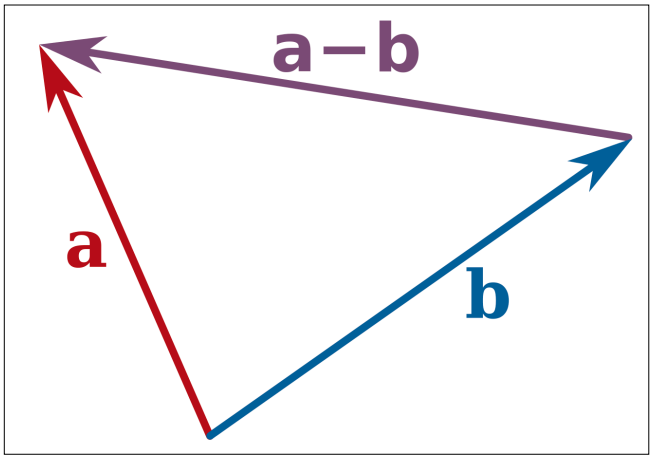
```
uniform mat4 u_MVPMatrix;
uniform mat4 u_MVMatrix;
attribute vec4 a_Position;
attribute vec4 a_Color;
attribute vec3 a_Normal;
varying vec4 v_Color;
varying vec3 v_Position;
varying vec3 v_Normal;
```

Neben der im vorigen Schritt besprochenen neuen Variablen mit Normaldaten legen wir ein weiteres Matrixfeld und zwei weitere Varying-Attribute an. Diese haben die Aufgabe, die an den Pixelshader weiterzuleitenden Positions- und Normaldaten entgegenzunehmen.

Weiterleitung an Pixelshader

Die für die eigentliche Grafikdarstellung zuständige Methode *main()* muss die angelieferten Informationen an den Pixelshader weiterleiten, der die eigentliche Berechnung für jedes Pixel neu durchführt:

```
void main()
{
  v_Color = a_Color;
  v_Position = vec3(u_MVMatrix * a_Position);
  v_Normal = vec3(u_MVMatrix * vec4(a_Normal,0.0));
```



Wikimedia Commons / bdesham

Vektormathematik ist ganz einfach (Bild 3)

```
  gl_Position = u_MVPMatrix * a_Position;
}
```

An dieser Stelle erklärt sich der Sinn der neu eingeführten Matrix *u_MVMatrix*. Die in *v_Position* und *v_Normal* vorliegenden Informationen müssen ja nicht dargestellt werden, weshalb die Multiplikation mit der Projektionsmatrix kontraproduktiv wäre – die Arbeit mit *MVMatrix* ist effizienter, weil sie das Element nur in den Kamera-Koordinatenraum transformiert.

Auf Seiten des Pixelshaders sind ebenfalls einige Programmänderungen im Bereich der Parameter notwendig. Die Aufgabe: Die angelieferten Informationen müssen entgegengenommen werden:

```
precision mediump float;
uniform vec3 u_LightPos;
varying vec3 v_Position;
```

```
varying vec4 v_Color;
varying vec3 v_Normal;
```

Neben den drei Datenfeldern legen wir hier ein neues *uniform*-Feld an, das die Position der Lichtquelle enthalten wird. Durch seine Einführung gewinnt der Shader an Flexibilität. Will der Designer die Position des Strahlers ändern, so muss der Shadercode nicht mehr verändert werden.

In der für die Lichtberechnung zuständigen Main-Methode findet sich nur wenig bekannter Code. Ob der hohen mathematischen Komplexität wollen wir den Code Zeile für Zeile durchsprechen. Nach der obligaten Deklaration der Genauigkeit finden sich neue *varying*- und ein neues *uniform*-Feld:

```
precision mediump float;
uniform vec3 u_LightPos;
varying vec3 v_Position;
varying vec4 v_Color;
varying vec3 v_Normal;
void main() {
  float distance = length(u_LightPos - v_Position);
  vec3 lightVector = normalize
  (u_LightPos - v_Position);
```

An erster Stelle steht hier die Errechnung der Distanz zwischen der Position der Lichtquelle und der Position des zu rendernden Vertex. Der Code macht sich dabei eine kleine Besonderheit der Vektorrechnung zunutze: Wer zwei Vektoren wie in Bild 3 voneinander subtrahiert, erhält einen die beiden Komponenten verbindenden dritten Vektor.

Multiplizierte Vektoren

Die Normalisierung dieses Längenvektors ermöglicht uns die eigentliche Ermittlung des Winkels zwischen den beiden Vektoren. Das Punkt beziehungsweise das Dotprodukt liefert einen Wert zurück, der das Verhältnis der beiden miteinander multiplizierten Vektoren beschreibt.

Das als Float vorliegende Resultat dieser Berechnung dient im nächsten Schritt zur Abschwächung eines vierwertigen Farbfelds. Bei maximaler Übereinstimmung der Vektoren wird hier *1* zurückgeliefert, maximale Devianz führt zu *0*. Da sich diese Operation von Haus aus auch auf den Transparenzwert *a* auswirkt, machen wir diesen Teil der Berechnung durch Zuweisen von *1.0* rückgängig:

```
float diffuse = max(dot(v_Normal, lightVector), 0.1);
vec4 colorBase=vec4(1,1,1,1) * diffuse;
colorBase.a=1.0;
```

GLSL betrachtet Zahlen ohne Kommastelle immer als Integers. Wer die Zuweisung der Intensität als *colorBase.a=1;* schreibt, wird vom Shadercompiler mit einem Syntaxfehler abgestraft. GLSL castet *ints* nicht automatisch in *floats*.

Nach der erfolgreichen Berechnung der lambertianischen Lichtkomponente müssen wir die im vorigen Abschnitt besprochene indirekte Belichtung anwenden. Unser Shader ▶

löst dies durch diskretes Addieren der drei Komponenten:

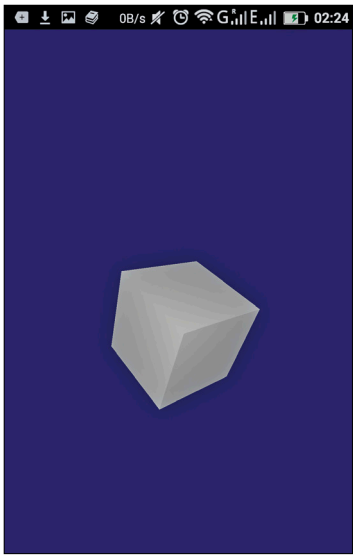
```
colorBase.r+=0.1;
colorBase.g+=0.1;
colorBase.b+=0.1;
```

Diese aus didaktischen Gründen interessante Vorgehensweise ist aus programmiertechnischer Sicht nicht notwendig: Die *max*-Methode liefert den kleineren der beiden an sie übergebenen Werte zurück. Jede Komponente des Ergebnisses von *max(dot(v_Normal, lightVector), 0.1)* ist also mindestens 0.1 groß.

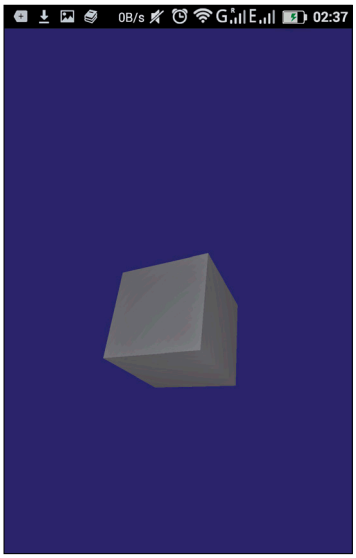
An diesem Codestück ist das Ansprechen der einzelnen Elemente des *vec4*-Felds interessant. GLSL unterstützt Entwickler durch das Anbieten der vier Subskripts *r, g, b* und *a*. Sie werden automatisch auf die vier Felder der *vec4*-Struktur gemappt.

Zu guter Letzt wandern die errechneten Resultate in den Grafikspeicher. Der Aufruf von *clamp()* sorgt dafür, dass die Maximal- und Minimalhelligkeitswerte nicht unter- oder überschritten werden. Manche GPUs reagieren darauf mit unschönen Renderingfehlern:

```
gl_FragColor =  clamp(colorBase,0.0,1.0);
}
```



Dank lambertscher Belichtung sieht der Würfel mehr als realistisch aus (Bild 4)



Dank Attenuation wirkt sich die Lichtquelle wesentlich schwächer auf den Würfel aus (Bild 5)

Auf Seiten des Shaders ist die Implementierung von lambertianischem Licht somit abgeschlossen. Wir können uns dem Java-Teil der Renderingpipeline zuwenden, der für das Bereitstellen der benötigten Komponenten zuständig ist. Die MV-Matrix lässt sich im Rahmen der Erzeugung der MVP-Matrix abgreifen, indem wir den Zwischenzustand durch Aufruf von *clone()* duplizieren:

```
Matrix.multiplyMM(myMVPMatrix, 0, myViewMatrix, 0,
myModelMatrix, 0);
myMVMatrix=myMVPMatrix.clone();
GLES20.glUniformMatrix4fv(aMV, 1, false, myMVMatrix, 0);
Matrix.multiplyMM(myMVPMatrix, 0, myProjectionMatrix, 0,
myMVPMatrix, 0);
```

Als Alternative dazu bietet sich das Aufrufen von *glUniformMatrix4fv* mit dem aktuellen Wert von *myMVPMatrix* an: Die Methode kombiniert die Werte sofort in Richtung des Shaderspeichers, wo sie von weiteren Veränderungen auf Java-Seite nicht tangiert werden.

Position der Lichtquelle

Das Bevölkern der Position der Lichtquelle erfolgt durch direktes Einschreiben eines konstanten Werts, der mit einer eigenen Weltmatrix an seinen Aufenthaltsort transformiert

Listing 2: Position der Lichtquelle

```
Matrix.setIdentityM(myLightModelMatrix, 0);
Matrix.translateM(myLightModelMatrix, 0, 3.0f,
3.0f, 15.0f);
Matrix.rotateM(myLightModelMatrix, 0,
rotationStep, 0.0f, 1.0f, 0.0f);
Matrix.multiplyMV(myLightInWorld, 0,
myLightModelMatrix, 0, myLightPosInModelSpace, 0);
Matrix.multiplyMV(myLightInEye, 0, myViewMatrix, 0,
myLightInWorld, 0);
GLES20.glUniform3f(aLiPos, myLightInEye[0],
myLightInEye[1], myLightInEye[2]);
GLES20.glUniformMatrix4fv(aMVP, 1, false,
myMVPMatrix, 0);
GLES20.glDrawArrays(GLES20.GL_TRIANGLES, 0, 36);
}
```

Listing 3: Intensität des Lichts

```
void main() {
  float distance = length(u_LightPos - v_Position);
  vec3 lightVector = normalize(u_LightPos -
  v_Position);
  float diffuse = dot(v_Normal, lightVector);

  diffuse = diffuse * (1.0 / (1.0 + (0.25 *
  distance * distance)));
  vec4 colorBase=vec4(1,1,1,1) * diffuse;
  colorBase.a=1.0;
  colorBase.r+=0.1;
  colorBase.g+=0.1;
  colorBase.b+=0.1;
  gl_FragColor =  clamp(colorBase,0.0,1.0);
}
```

wird (Listing 2). Damit ist das Programm abermals zur Ausführung bereit. Bild 4 zeigt das – durchaus realistisch aussehende – Resultat.

Intensität des Lichts

In manchen Situationen ist es wünschenswert, wenn die Intensität des Lichts mit steigendem Abstand zwischen Strahler und beleuchtetem Objekt absinkt. Dies lässt sich durch eine kleine Änderung im Pixelshader bewerkstelligen, die den errechneten Wert mit einer aus dem Abstand errechneten Konstante abschwächt (Listing 3).

Auf Seiten des Renderingpfads sind hier keine Änderungen notwendig. Bild 5 zeigt, wie sich unsere Szene bei aktivierter Attenuation verhält.

An dieser Stelle sei noch ein kleiner Trick zur effizienteren Realisierung von Ambient Lighting vorgestellt. Wer die Attenuation folgendermaßen ausprogrammiert, kann sich das manuelle Addieren der *r*-, *g*- und *b*-Komponenten sparen:

```
void main() {
  float distance = length(u_LightPos -
  v_Position);
  vec3 lightVector = normalize(u_LightPos -
  v_Position);
  float diffuse = max(dot(v_Normal,
  lightVector),0.1);
  vec4 colorBase=vec4(1,1,1,1) * diffuse;
  colorBase.a=1.0;
  gl_FragColor =  clamp(colorBase,0.0,1.0);
}
```

max() weist die GPU dazu an, das größere der beiden Werte zu retournieren. In unserem Fall ist durch das Übergeben von 0.1 sichergestellt, dass keine kleineren Ergebnisse entstehen.

Es handelt sich dabei zwar um keine perfekte Implementierung von Diffuse Lighting, die Einsparungen in Sachen Codeumfang sind allerdings durchaus interessant.

Farbwerte

Diffuses und lambertianisches Licht sind gut und schön – wirklich interessant werden die Szenen erst dann, wenn die in den Modellen vorhandenen Farbwerte Berücksichtigung finden.

Dies lässt sich durch eine kleine Änderung im Pixelshader bewerkstelligen. Wir haben die Ergebnisse der diversen Lichtberechnungen bisher direkt als Farbkomponente betrachtet: Die von 0 bis 1 laufenden Ergebnisse lassen sich auch zur Attenuation von an anderer Stelle im Programm vorhandenen Farbwerten nutzen:

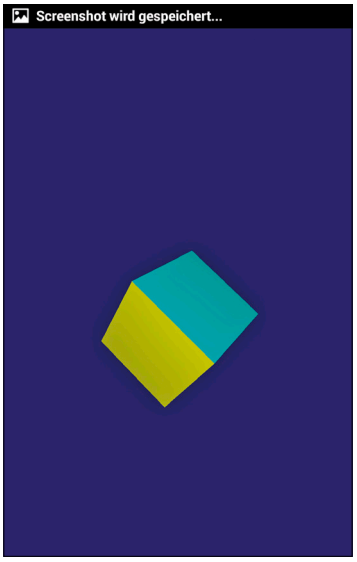
```
void main()
{
  ...
  vec4 colorBase=vec4(1,1,1,1) * diffuse;
  colorBase.a=1.0;
  colorBase.r+=0.1;
  colorBase.g+=0.1;
  colorBase.b+=0.1;
  gl_FragColor = clamp(v_Color*colorBase,0.0,1.0);
}
```

Die neue Version unseres Shaders nutzt *colorBase* zur Skalierung der in *v_Color* befindlichen Vertexfarben. Das Resultat davon ist eine mehr oder weniger stark abgeschwächte Farbfläche, deren Grundfarbe von den im Vertexfarbarray angelegten Informationen bestimmt wird.

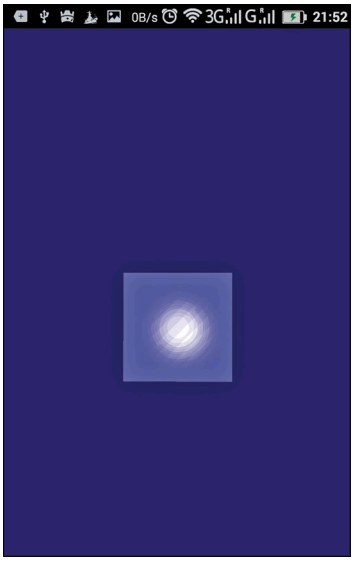
Damit ist unser Programm abermals zur Ausführung bereit. Bild 6 zeigt das – durchaus befriedigende – Resultat unserer Mühen.

Punktuelles Licht

Als Dritter im Bunde fehlen die als Specular Highlights bezeichneten Punktlichtquellen. Sie entsprechen im realen Leben einem gerichteten Spot, dessen Bestrahlung auf einem Zielobjekt ein kreisförmiges Highlight hervorruft. Auch sie lassen sich in Shadern mit vergleichsweise geringem Aufwand nachbilden. Die neue Version des Pixelshaders beginnt mit Bekanntem:



Farbe und Belichtung sorgen gemeinsam für Realismus (Bild 6)



Specular Lighting steigert den Realismus immens (Bild 7)

```
void main()
{
  float distance = length(u_LightPos - v_Position);
  vec3 lightVector = normalize(u_LightPos -
  v_Position);

  float diffuse = dot(v_Normal, lightVector);
```

Nach der bereits bekannten Berechnung der Intensität des diffusen Lichts wenden wir uns dem Lichtpunkt zu. Er ver- ▶

hält sich im Großen und Ganzen analog: In praktischen Shadern würde statt *u_LightPos* eine weitere Variable zur Positionierung des Punktlichts herangezogen werden:

```
vec3 toPointLight = vec3(u_LightPos)-
vec3(v_Position);
toPointLight = normalize(toPointLight);
float cosine = max(dot(normalize(v_Normal),
toPointLight), 0.0);
```

Die Besonderheit specularer Highlights findet sich in der Dotierung des errechneten Wertes. Angesichts der besonderen Wichtigkeit wollen wir die Zeile isoliert betrachten:

```
vec4 specularCol = vec4(1.0,1.0,1.0,1.0) *
pow(cosine,200.0);
```

pow ist die Potenzfunktion: Der im vorigen Schritt nach dem gewohnten Verfahren ermittelte Helligkeitswert wird mit einer als Shinyness Constant bezeichneten Zahl multipliziert, die die Breite des Lichtkegels bestimmt. Dabei gilt: Je höher der Faktor, desto kleiner der errechnete Kreis.

Durch Multiplikation mit einem Farbvektor entsteht der endgültige Helligkeitswert des jeweiligen Pixels – unser Beispiel übergibt hier *1/1/1/1*, um einen komplett weißen Highlight-Punkt zu erhalten.

Auch hier sei darauf hingewiesen, dass reale Shader davon profitieren würden, wenn Shinyness und Farbe als von außen setzbare Parameter ausgeführt sind. Das Mitführen mehrerer identischer Shader mündet häufig in Problemen.

Beim Zusammenfügen der verschiedenen Lichtkomponenten wartet eine weitere Falle auf Anfänger. Specular Light ist eine komplett eigene Lichtquelle, die mit diffusem und ambientem Licht nichts zu tun hat. Sie darf erst im letzten Schritt zur Gesamtberechnung addiert werden. Wer die Komponenten vor der Einfärbung zusammenfügt, verliert den Spotlight-Effekt:

```
vec4 colorBase=vec4(1,1,1,1) * diffuse;
colorBase.a=1.0;
colorBase.r+=0.1;
colorBase.g+=0.1;
colorBase.b+=0.1;
gl_FragColor =  clamp(v_Color*colorBase + specular-
Col,0.0,1.0);
}
```

Damit ist auch diese Variante des Programms einsatzbereit (Bild 7).

Listing 4: Laden der Textur

```
int[] textureArray = new int[1];
GLES20.glGenTextures(1, textureArray, 0);
if (textureArray[0] != 0){
  final BitmapFactory.Options options = new
  BitmapFactory.Options();
  options.inScaled = false;
  final Bitmap resBitmap = BitmapFactory.
  decodeResource(myApplication.
  getApplicationContext().getResources(),
  R.drawable.nmgtexture, options);
  GLES20.glBindTexture(GLES20.GL_TEXTURE_2D,
  textureArray[0]);
  GLES20.glTexParameteri(GLES20.GL_TEXTURE_2D,
  GLES20.GL_TEXTURE_MIN_FILTER,
  GLES20.GL_NEAREST);
  GLES20.glTexParameteri(GLES20.GL_TEXTURE_2D,
  GLES20.GL_TEXTURE_MAG_FILTER,
  GLES20.GL_NEAREST);
  GLUtils.texImage2D(GLES20.GL_TEXTURE_2D, 0,
  resBitmap, 0);
  resBitmap.recycle();
}
if (textureArray[0] == 0){
  Log.e("NMG",
  "Texturgenerierung fehlgeschlagen");
}
myTextureDataHandle=textureArray[0];
}}
```

Detail aus der Konserve

In der Praxis sind Würfel nur sehr selten komplett einfarbig: Materialien beeinflussen die Oberfläche; Bemalung und Lackierung sorgen für zusätzliche Details. Rein theoretisch spricht nichts dagegen, dies durch Einführung zusätzlicher Vertices mit konstanter Farbe zu realisieren.

In der Praxis würde der außer Kontrolle geratende Rechenleistungsbedarf der maximalen Detailfülle enge Grenzen auferlegen. Zudem ist die zusätzliche geometrische Information nur in den wenigsten Fällen notwendig. Die durch Farbauftrag entstehenden zusätzlichen Mikrometer spielen in der großen Ordnung der Dinge nur eine sehr geringe Rolle.

Texturierung stellt ein Alternativprogramm zur Umgehung dieses Problems dar. Jeder Vertex wird dabei mit einem zusätzlichen zweidimensionalen Koordinatenpaar versehen, das seine Position in einem als Textur bezeichneten Bitmap beschreibt. Zur Laufzeit ermittelt ein als Sampler bezeichnetes Modul die zum jeweiligen Vertex passenden Farbinformationen per linearer Interpolation.

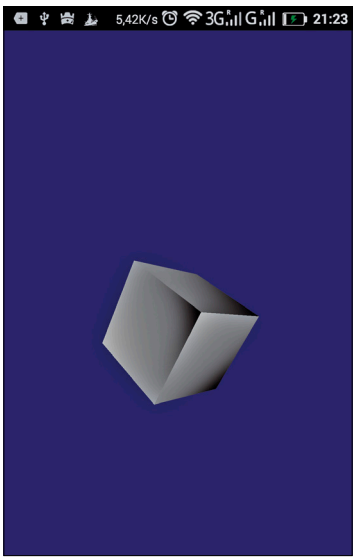
Wie im Fall der Normalen müssen wir auch hier mit dem Anlegen eines neuen Datenfelds beginnen. Texturkoordinaten sind per Definition zweiwertig und liegen immer im Wertebereich 0 bis 1 – die Korrelation zwischen Bitmap- und Texturkoordinaten erledigt die Hardware:

```
final float[] cubeTextureCoordinateData =
{
  0.0f, 0.0f,
  0.0f, 1.0f,
```

```
1.0f, 0.0f,
0.0f, 1.0f,
1.0f, 1.0f,
1.0f, 0.0f,
```

Aus Gründen der Bequemlichkeit recyceln wir die Texturkoordinaten für alle Seiten des Würfels. In einem mit einem 3D-Editor erzeugten Objekt wären komplexere Zuordnungen zwischen Textur und Vertices an der Tagesordnung. Die Konversion des Puffers in einen für die Hardware ansprechbaren FloatBuffer besprechen wir an dieser Stelle nicht weiter, da sie im Großen und Ganzen analog zur Handhabung der vorherigen Datenfelder ist.

Stattdessen wollen wir uns hier der Generierung der Textur zuwenden. Im Bereich der Grafikprogrammierung hat sich die Nutzung von Texturdimensionen etabliert, die Werte von 2^n sind. Manche (ältere) GPUs bringen in dieser Situation wesentlich mehr Leistung. Unser Beispiel nutzt eine 256 x 256 Pixel große PNG-Datei, die in das Drawables-Verzeichnis importiert wird.



Die Textur ist am Platz (Bild 8)

Laden der Textur

Als nächste Aufgabe folgt das Laden der Textur. Wir bringen den dazu notwendigen Code in *onSurfaceCreated* unter, wo er nach der Assemblierung des Hauptshaders abgearbeitet wird (Listing 4).

glGenTextures liefert uns einen Texture-Handle zurück, den wir im nächsten Schritt mit den im Bitmap vorliegenden Farbinformationen bevölkern. *MinFilter* und *MaxFilter* setzen die Interpolationsverfahren fest, die zum Einsatz kommen, wenn die Textur kleiner oder größer als der zu befüllende Bereich ist. In *onDrawFrame* ist ebenfalls etwas Arbeit erforderlich:

```
int aTexCoordID = GLES20.glGetAttribLocation
(myFinalProgramHandle, "a_TexCoordinate");
GLES20.glActiveTexture(GLES20.GL_TEXTURE0);
GLES20.glBindTexture(GLES20.GL_TEXTURE_2D,
myTextureDataHandle);
GLES20.glUniform1i(aTexCoordID, 0);
myCubeTextures.position(0);
GLES20.glVertexAttribPointer(aTexCoordID, 2,
GLES20.GL_FLOAT, false, 0, myCubeTextures);
GLES20.glEnableVertexAttribArray(aTexCoordID);
```

OpenGL erledigt die Bereitstellung von Texturen über Texureinheiten, die durch Nutzung von *glBindTexture* mit Texturdaten verbunden werden. Der Rest der Änderungen beschränkt sich auf die Zuweisung des Koordinatenarrays an den betreffenden Datenkanal.

Im Vertexshader müssen wir das angelieferte Koordinatenpaar entgegennehmen und in Richtung des Pixelshaders weiterschreiben. Der Rest des Shaders bleibt unverändert, wir drucken hier nur die neuen Teile ab:

```
attribute vec2 a_TexCoordinate;
varying vec2 v_TexCoordinate;
void main() {
  ...
  v_TexCoordinate = a_TexCoordinate;
```

Texturen stehen auf Shaderseite in Form eines Samplerobjekts zur Verfügung. Die Methode *texture2D* erlaubt das Auslesen des an einem bestimmten Ort stehenden Farbwerts, der im nächsten Schritt anstelle der Vektorfarbe in die Fragmentfarbberechnung wandert:

```
uniform sampler2D u_Texture;
varying vec2 v_TexCoordinate;
...
void main() {
  ...
  gl_FragColor =  clamp(texture2D(u_Texture,
  v_TexCoordinate)*colorBase + specularCol,0.0,1.0);
}
```

Damit ist auch diese Version des Programms fertig. Bild 8 zeigt, dass die Textur auf den Würfel aufgetragen wird.

Interaktion mit dem Nutzer

Unsere zweifelsohne nett anzusehenden Szenen leiden im Moment unter einer gravierenden Schwäche: Der Nutzer kann mit ihnen nicht interagieren. Das liegt daran, dass die von Haus aus implementierte GLSurfaceView keine Möglichkeit zur Entgegennahme von Touchscreen-Kommandos anbietet.

Als Lösung bietet sich die Realisierung einer eigenen Variante von GLSurfaceView an, die das von Haus aus als *nop* ausgelegte onTouchEvent durch eine aktive Implementierung ersetzt. Erstellen Sie eine neue Klasse namens *NMGGLView* (Listing 5). ▶

Listing 5: Klasse NMGGLView

```
public class NMGGLView extends GLSurfaceView {
  public NMGGLView(Context context) {
    super(context);
  }
  public NMGGLView(Context context,
  AttributeSet attrs) {
    super(context, attrs);
  }
  @Override
  public boolean onTouchEvent(MotionEvent e) {
  }
}
```

Neben dem Überschreiben der beiden obligatorischen Konstruktoren legen wir einen *onTouchEvent*-Handler an, dessen Verhalten aus der Entwicklung gewöhnlicher Android-Programme bekannt sein sollte.

Da *NMGGLView* eine perfekte Implementierung der GLView-Spezifikation darstellt, können wir sie in *onCreate* anstelle der vom Betriebssystem bereitgestellten Klasse laden:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  myGLView = new NMGGLView(this);
  myGLView.setEGLContextClientVersion(2);
  myGLView.setPreserveEGLContextOnPause(true);
  myGLView.setRenderer(new
  NMGRenderer(getApplication()));
  setContentView(myGLView);
}
```

Als nächstem Akt müssen wir uns der eigentlichen Verschiebung der Position des Quaders zuwenden. Dazu ist ein Event-Handler notwendig, der die Fingergesten in für die Engine verständliche Wertepaare umwandelt. Um erratisches Verhalten nach dem Programmstart zu verhindern, initialisieren wir *myPreviousX* und *myPreviousY* mit *-1*. Wenn die Variablen diesen Schlüsselwert enthalten, so aktualisieren wir ihre Inhalte und beenden die Abarbeitung von *onTouchEvent* (Listing 6). Unser Renderer berechnet die Matrix bei jedem Durchlauf von null auf neu. Aus diesem Grund müssen wir die Bewegungen über die gesamte Lebenszeit des Programms integrieren – eine Aufgabe, die die Variablen *myDeltaX* und *myDeltaY* übernehmen.

Beim Hochheben der Hand setzen wir die Previous-Werte auf *-1*, um das Programm in den Ruhezustand zu versetzen:

```
case MotionEvent.ACTION_UP:
    myPreviousX = -1;
    myPreviousY = -1;
    break;
  }
  myPreviousX = x;
  myPreviousY = y;
  return true;
}
```

Android führt den Renderer nicht am GUI-Thread aus. Ein Luxus, der onTouchEvent nicht zuteil wird. Aus diesem Grund müssen wir die für die Kommunikation verwendeten Variablen als *volatile* deklarieren, um den Compiler dazu zu animieren, bei jedem Durchlauf aktuelle Informationen aus dem Speicher zu holen und kein Caching zu verwenden:

```
public class NMGGLView extends GLSurfaceView {
  public volatile float myDeltaX=0;
  public volatile float myDeltaY=0;
```

Als letzte Änderung muss der Renderer dazu angewiesen werden, die Drehungswerte nicht mehr aus der automatisch inkrementierten Variablen zu entnehmen:

```
@Override
public void onDrawFrame(GL10 gl) {
  Matrix.translateM(myModelMatrix, 0, 0.0f, 0.0f, -7.0f);
  Matrix.rotateM(myModelMatrix, 0, myGLView.myDeltaX,
  0.0f, 1.0f, 0.0f);
  Matrix.rotateM(myModelMatrix, 0, myGLView.myDeltaY,
  1.0f, 0.0f, 0.0f);
```

Damit sind wir auch hier fertig. Führen Sie das Programm auf Ihrem Smartphone aus, um sich am frei beweglichen Quader zu erfreuen. Ein interessantes Detail ist, dass sich die Gesten nicht immer identisch verhalten: Je nach Gesamtausrichtung des Quaders kann es sein, dass Achsen plötzlich invertiert sind. Dies ist ein Problem unserer naiven Programmierweise – als Ideallösung bietet sich die Nutzung einer Modellklasse an, die die Transaktionen en bloc erledigt. ■

Listing 6: Verschiebung der Position des Quaders

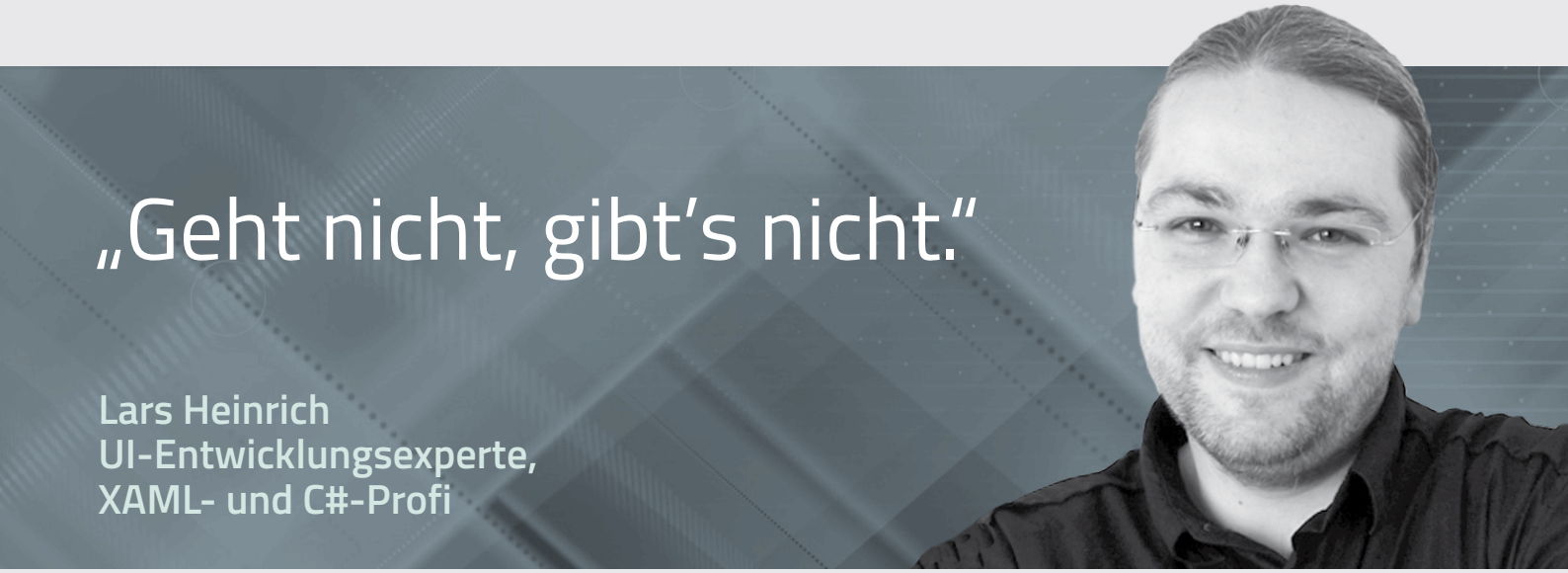
```
@Override
public boolean onTouchEvent(MotionEvent e) {
  if(myPreviousX==-1)
  {
    myPreviousX = e.getX();
    myPreviousY = e.getY();
    return true;
  }

  float x = e.getX();
  float y = e.getY();
  switch (e.getAction()) {
    case MotionEvent.ACTION_MOVE:
      float dx = x - myPreviousX;
      float dy = y - myPreviousY;
      if (y > getHeight() / 2) {
        dx = dx * -1 ;
      }
      if (x < getWidth() / 2) {
        dy = dy * -1 ;
      }
      myDeltaX+=dx;
      myDeltaY+=dy;
```

Tam Hanna
ist Autor, Trainer und Berater mit den Schwerpunkten Webentwicklung und Webtechnologien. Es lebt in der Slowakei und leitet dort die Firma Tamoggemon Holding k.s. Er bloggt sporadisch unter:
www.tamoggemon.com

developer media

developer media Trainings

Updates für Ihr Know-how



Entwicklung von Universal Windows Apps

Trainer: Lars Heinrich

2 Tage, 06.-07.06.2016, München

Frühbucher: 1.799 EUR zzgl. MwSt.
Normalpreis: 1.999 EUR zzgl. MwSt.

Webanwendungen mit HTML5, CSS3 und JavaScript

Trainer: David Tielke

2 Tage, 11.-12.07.2016, Köln

Frühbucher: 1.799 EUR zzgl. MwSt.
Normalpreis: 1.999 EUR zzgl. MwSt.

Webentwicklung mit ASP.NET, MVC und Web API

Trainer: David Tielke

3 Tage, 13.-15.07.2016, Köln

Frühbucher: 2.199 EUR zzgl. MwSt.
Normalpreis: 2.399 EUR zzgl. MwSt.

Cross-Plattform-Apps mit C# und Xamarin

Trainer: Sebastian Seidel

3 Tage, 12.-14.07.2016, Köln

Frühbucher: 2.199 EUR zzgl. MwSt.
Normalpreis: 2.399 EUR zzgl. MwSt.

Ihr Ansprechpartner:
Fernando Schneider – Key Account Manager – developer media
Telefon: +49 (0)89 74117-831 – E-Mail: fernando.schneider@developer-media.de

developer-media.de/trainings

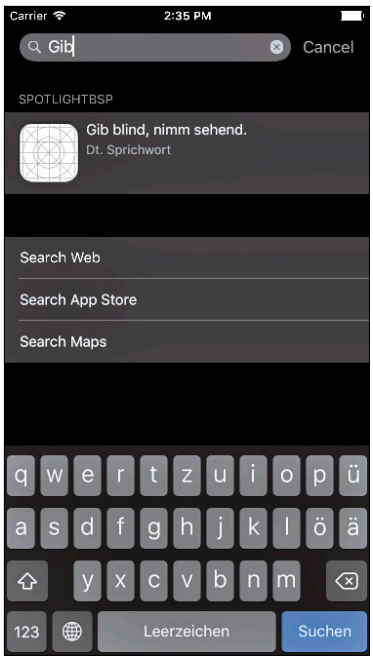
DAS CORE SPOTLIGHT FRAMEWORK

Spot on App

Mit dem Core Spotlight Framework ist es möglich, die Inhalte einer App innerhalb der Spotlight-Suche zu indizieren.

Bis zur Version 9 von iOS war der Inhalt von Apps für die Suchfunktion Spotlight von iOS eine Black Box. Mit Einführung der aktuellen iOS-Version hat Apple ein API eingeführt, das es dem Entwickler ermöglicht, den Inhalt einer App zu indizieren und somit über die Spotlight-Suche von iOS dem Anwender zugänglich zu machen. Wie aber wird der Inhalt nun verfügbar gemacht?

Zwei Komponenten werden hierzu benötigt: zum einen das Core Spotlight Framework und zum anderen die Klasse *NSUserActivity*, die mit iOS 8 eingeführt worden ist. Die Klasse *NSUserActivity* kennen Sie bereits aus dem Artikel »Das Handoff API« in der **web & mobile developer** 5/2016, Seite 94. In diesem Artikel wurde die Klasse bereits vorgestellt. Wer diese noch nicht kennt, hier nur kurz der Hinweis, dass diese mit dem Handoff API von Apple bereitgestellt wurde und den Austausch von Daten zwischen iOS- und OS-X-Geräten ermöglicht.



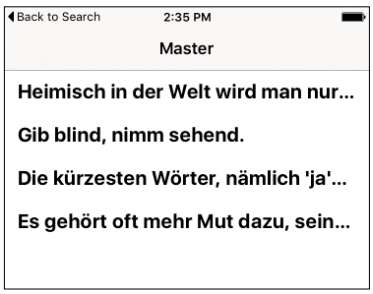
Eine App, die Zitate anzeigt, mit Spotlight-Indizierung (Bild 1)

Spotlight mit Beispiel

Eine App über Spotlight zu indizieren ist nur dann sinnvoll, wenn es innerhalb derselben auch etwas zu indizieren gibt. Texte bieten sich hierfür natürlich hervorragend an.

Aus diesem Grund erweitere ich ein bereits einmal verwendetes Beispiel. Es handelt sich dabei um eine App, die Zitate anzeigt und die um die Spotlight-Indizierung erweitert wurde (Bild 1). Die Zitate werden in der Übersicht in der Regel nicht komplett gezeigt, sondern nur angerissen (Bild 2).

Nach Auswahl eines Zitats wird dieses in der folgenden Detail-View komplett angezeigt und auch der Autor/Urheber wird nicht vergessen (Bild 3). Als Ausgangspunkt für die App wird eine neue Vorlage vom Typ *Master-Detail Application* verwendet. Das Wissen um die Funktionsweise der entsprechenden Vorlage setze ich voraus. Das erzeugte Storyboard wird nur in einer Kleinigkeit geändert: In der Detail-View wird ein zusätzliches Label-Control eingefügt, das später den Namen des Autors anzeigt. Außerdem wird das darüber liegende Label-Control so vergrößert, dass es die ganze Breite der View verwendet. Im ersten Schritt muss nun eine Datenstruktur zur Speicherung der Zitate angelegt werden. Dies geschieht in der Klasse *MasterViewController*.



Die Zitate-App soll indiziert werden (Bild 2)

Zur Speicherung eines Zitats in der App wird eine neue Struktur erzeugt. Diese enthält neben dem Text noch als ein weiteres Attribut den Urheber (Listing 1).

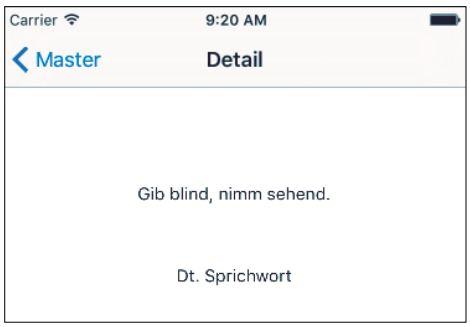
Die einzelnen Zitate werden in der Klasse *MasterViewController* in einem Array angelegt. Neben dem Aufbau des Arrays werden noch zwei weitere Variablen *detailViewController* und *restoreZitat* angelegt. Um ein ausgewähltes Zitat an die Detail-View zu übergeben, wird eine Instanz auf die View benötigt. Außerdem ist die Variable *restoreZitat* bei der Wiederherstellung einer *NSUserActivity* erforderlich.

Ein Zitat zerlegen

In der Methode *viewDidLoad* befindet sich der Code zum Zerlegen und Indizieren des Zitats (Listing 2). Im ersten Schritt wird der bereits deklarierten Variablen *detailViewController* die entsprechende View-Instanz zugewiesen. Anschließend wird das Zitat zur Indizierung zerlegt.

Zuerst wird ein Array vom Typ *CSSearchableItem* angelegt. Objekte dieser Klasse können indiziert und deren Inhalt dadurch innerhalb der Spotlight-Suche vom Anwender gefunden werden. In einer *for*-Schleife werden die im Array enthaltenen Zitate nun nacheinander durchlaufen. Innerhalb der Schleife wird im ersten Schritt eine Instanz der Klasse *CSSearchableItemAttributSet* erzeugt. Mittels der Instanz kann auf Eigenschaften eines *CSSearchItem*-Objekts zugegriffen werden. Im Beispiel wird der Eigenschaft *Text* das Zitat selbst zugewiesen und der Eigenschaft *contentDescription* der Urheber beziehungsweise dessen Name.

Nach diesen beiden Attributen beziehungsweise dessen Inhalt kann anschließend innerhalb der Spotlight-Suche ge-



Ein Zitat wird im Detail angezeigt (Bild 3)

sucht werden. Das Zitat wird hierfür in Schlüsselwörter zerlegt. Zur Trennung wird die Methode *componentsSeparatedByString* aufgerufen. Neben dem zerlegten Zitat wird der Variablen *keywords* außerdem der in der Eigenschaft *Urheber* des Zitat-Objekts enthaltene Inhalt zugewiesen. Zuletzt wird die Variable *keywords* dann der Eigenschaft des Objekts *attributeSet* zugewiesen. Den Abschluss bildet das Hinzufügen des *CSSearchableItem*-Objekts zum Array *searchableItems*. Dem Konstruktor der Klasse *CSSearchableItem* wird hierzu der Text des Zitats, dass *attributeSet*-Objekt sowie ein *DomainIdentifier* übergeben. Über einen Aufruf der Methode *append* wird dann das erzeugte Objekt *item* übernommen. Dann wird noch überprüft, ob beim Hinzufügen des Index ein Fehler aufgetreten ist.

Listing 1: Definition der Struktur

```
import UIKit
import CoreSpotlight
import MobileCoreServices

struct Zitat {
  var text: String
  var urheber: String
}

class MasterViewController: UITableViewController {
  let zitate = [
  Zitat(text:"Heimisch in der Welt wird man nur
  durch Arbeit. Wer nicht arbeitet, ist heimatlos.",
  urheber: "Berthold Auerbach"),
  Zitat(text:"Gib blind, nimm sehend.",
  urheber:"Dt. Sprichwort"),
  Zitat(text:"Die kürzesten Wörter, nämlich ‚ja‘
  und ‚nein‘, erfordern das meiste Nachdenken.",
  urheber:"Pythagoras von Samos"),
  Zitat(text:"Es gehört oft mehr Mut dazu, seine
  Meinung zu ändern, als ihr treu zu
  bleiben.",urheber:"Friedrich Hebbel")]

  var detailViewController:
    DetailViewController? = nil
  var restoreZitat: Zitat?
```

Bisher wurde nur vorgestellt, wie die Zitate innerhalb der App verwaltet und indiziert werden. Die Klasse *NSUserActivity* kam bisher nicht zum Einsatz. Das wird sich jetzt ändern. Die *NSUserActivity*-Instanz, mittels derer dann das entsprechende Zitat in der App aufgerufen werden kann, wird im *DetailViewController* erzeugt. Erst hierfür wird eine *NSUserActivity*-Instanz benötigt. Die Anlage einer entsprechenden Instanz beginnt in der Methode *prepareForSegue* der Klasse *MasterViewController*. Beim Aufruf wird festgelegt, wel- ▶

Listing 2: Funktion ViewDidLoad

```
override func viewDidLoad() {
  super.viewDidLoad()
  if let split =  self.
    splitViewController {
    let controllers =
      split.viewControllers
    self.detailViewController =
    (controllers[controllers.count-1]
      as! UINavigationController).
        topViewController as?
          DetailViewController
  }
  var searchableItems:
    [CSSearchableItem] = []
  for zitat in zitate {
    let attributeSet =
      CSSearchableItemAttributeSet(
        itemContentType: kUTTypeItem
          as String)
    attributeSet.title = zitat.text
    attributeSet.contentDescription =
      zitat.urheber
    var keywords = zitat.text.
      componentsSeparatedByString(" ")
    keywords.append(zitat.urheber)
    attributeSet.keywords = keywords
    let item = CSSearchableItem(
      uniqueIdentifier: zitat.text,
        domainIdentifier: "Zitate",
          attributeSet: attributeSet)
     searchableItems.append(item)
  }
  CSSearchableIndex.
    DefaultSearchableIndex().
      IndexSearchableItems(
        searchableItems) { (error) ->
        Void in
    if error != nil {
      print(error?.
        localizedDescription)
    }
  }
}
```

Links zum Thema

- NSUserActivity
  https://developer.apple.com/library/ios/documentation/Foundation/Reference/NSUserActivity_Class
- iOS-Seite von Apple
  www.apple.com/de/ios
- The Evolution of iOS from iOS 1 – iOS 8 (Infographic)
  https://blog.7dayshop.com/ios-timeline
- Entwicklerseite von Apple
  https://developer.apple.com

ches Zitat im Detail angezeigt werden soll (Listing 3). Innerhalb der Klasse *DetailViewController* gibt es die Eigenschaft *detailItem*, der das ausgewählte Zitat zugewiesen wird. Anschließend geht es innerhalb der Klasse weiter (Listing 4). Hier finden sich zunächst die üblichen Verdächtigen, etwa die benötigten Outlets und die Definition der Eigenschaft *detailtem*.

Der eigentlich interessante Code ist in der Methode *configureView* untergebracht. Darin wird zu Beginn überprüft, ob den Label-Controls ein Wert zugewiesen werden kann. Anschließend geht es an die Ableitung der *NSUserActivity*-Instanz. Im Konstruktor (Initalisierer) der Klasse wird der in der *Info.plist*-Datei definierte ActivityType übergeben (Bild 4).

Der Typ einer Activity ist für die Zuordnung zu einer App erforderlich. Für dessen Identifikation muss eine eindeutige Zeichenkette verwendet werden.

Als Nächstes werden der Eigenschaft *userInfo* der Inhalt der Eigenschaft *text* (enthält das Zitat) und der Eigenschaft *urheber* zugewiesen. Der Eigenschaft *userInfo* kommt eine zentrale Bedeutung zu. Es handelt sich hierbei um ein Dictionary, dem Informationen zugewiesen werden, um diese dann beispielsweise auf einem anderen Gerät weiterverwenden zu können. Dann wird abermals eine Variable *keywords* erzeugt. Auch an dieser Stelle wird der Inhalt des Zitats zur Indizierung der Eigenschaft *keywords* der *NSUserActivity*-Instanz zugewiesen.

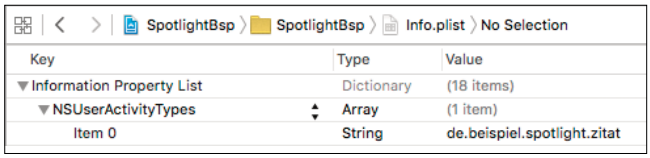
Warum passiert hier dasselbe wie schon in der Klasse *MasterViewController*? Möglicherweise wird die Activity zu einem anderen iOS-Gerät übertragen. Auch dort muss dann die Suche nach Schlüsselbegriffen möglich sein. Zuletzt werden

Listing 3: Methode prepareForSegue

```
override func prepareForSegue(segue:
UIStoryboardSegue, sender: AnyObject?) {
  if segue.identifier == "showZitat" {
    if let indexPath =
    self.tableView.indexPathForSelectedRow {
      let zitat = zitate[indexPath.row] as Zitat
      let controller =
      (segue.destinationViewController as!
      UINavigationController).topViewController as!
      DetailViewController
      controller.detailItem = zitat
      controller.navigationItem.leftBarButtonItem =
      self.splitViewController?.
      displayModeButtonItem()
        controller.navigationItem.
        leftItemsSupplementBackButton = true }
  } }
```

Listing 4: Klasse DetailViewController

```
class DetailViewController: UIViewController {
  @IBOutlet weak var detailDescriptionLabel:
  UILabel!
  @IBOutlet var laUrheber: UILabel!
  var detailItem: Zitat? {
    didSet {
      self.configureView()
    }
  }

  func configureView() {
    if let detail = self.detailItem {
      if let label = self.detailDescriptionLabel {
        label.text = detail.text
      }
      if let label2 = self.laUrheber {
        label2.text = detail.urheber
      }
      /* print (detail.text)
      print (detail.urheber)*/
      let activity = NSUserActivity(activityType:
      "de.beispiel.spotlight.zitat")
      activity.userInfo = ["text": detailItem!.text,
      „urheber“: detailItem!.urheber]
      activity.title = "Zitat"
      var keywords = detailItem!.text.
      componentsSeparatedByString(" ")
      keywords.append((detailItem?.urheber)!)
      activity.keywords = Set(keywords)
      activity.eligibleForHandoff = false
      activity.eligibleForSearch = true
       activity.eligibleForPublicIndexing = true
      activity.becomeCurrent()
    }
  }
  override func viewDidLoad() {
    super.viewDidLoad()
    self.configureView()
  }
  override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
  }
}
```



Definition eines ActivityTypes in der Datei *Info.plist* (Bild 4)

Listing 5: Prüfen, ob Activity vorhanden ist

```
func application(application: UIApplication,
continueUserActivity userActivity: NSUserActivity,
restorationHandler:([AnyObject]?) -> Void) -> Bool {
  if userActivity.activityType ==
  CSSearchableItemActionType {
    if let _ = userActivity.userInfo?
    [CSSearchableItemActivityIdentifier] as? String {
      return true
    }
  }
  let splitController = self.window?.
  rootViewController as! UISplitViewController
  let navigationController = splitController.
  viewControllers.first as! UINavigationController
    navigationController.topViewController?.
    restoreUserActivityState(userActivity)
  return true
}
```

Listing 6: Activity wiederherstellen

```
override func restoreUserActivityState(activity:
NSUserActivity) {
  if let stext = activity.userInfo?["text"] as?
  String,
  let urheber = activity.userInfo?["urheber"] as?
  String {
    let zitat = Zitat(text: stext, urheber:urheber )
    self.restoreZitat = zitat
    self.performSegueWithIdentifier
    ("showZitat", sender: self)
  }
  else
  {
    let alert = UIAlertController(title: "Fehler",
    message: "Informationen konnten nicht gelesen
    werden:\n\(activity.userInfo)",
    preferredStyle: .Alert)
    alert.addAction(UIAlertAction(title:
    "Hinweis", style: .Cancel, handler: nil))
    self.presentViewController(alert,
    animated: true, completion: nil)
  }
}
```

noch drei Eigenschaften konfiguriert. Mit *eligibleForHandoff* wird festgelegt, ob die Activity vom aktuellen iOS-Gerät auf ein anderes übertragen werden kann.

Um über die Suche gefunden zu werden, muss die Eigenschaft *eligibleForSearch* entsprechend gesetzt werden. Mit *eligibleForPublicIndexing* lässt sich festlegen, ob die Activity öffentlich verfügbar ist und ob sie von weiteren iOS-Nutzern verwendbar ist. Die Methode *becomeCurrent* wird aufgerufen, um die gerade erzeugte Activity zur aktuell vom Benutzer verwendeten zu machen.

Activity vorhanden?

Während des Starts der App wird bereits geprüft, ob ein aktives Objekt vom Typ *NSUserActivity* vorhanden ist. Diese Prüfung findet in der Klasse *AppDelegate* und dort innerhalb der Methode *application(_:continueUserActivity:restorationHandler:)* statt (Listing 5).

Los geht es innerhalb der Methode mit der Prüfung, ob die Eigenschaft *activityType* ein Objekt vom Typ *CSSearchableItemActionType* enthält. Wenn das der Fall ist, wird versucht, eine Ableitung zu erzeugen. Anschließend wird der NavigationController angewiesen, die Methode *restoreUserActivity* aufzurufen, um die Activity anzuzeigen. Mit der Verarbeitung geht es dann innerhalb der entsprechenden Methode der Klasse *MasterViewController* weiter.

Activity wieder herstellen

Die Methode *restoreUserActivityState* innerhalb der Klasse *MasterViewController* wird aufgerufen, wenn der Inhalt einer übergebenen UserActivity wieder gelesen werden soll (Listing 6). Im Beispiel werden die Inhalte *text* und *urheber* aus der Eigenschaft *userInfo* gelesen.

In der folgenden Zeile werden diese Informationen dem Initialisierer der Struktur *Zitat* übergeben. Die anschließend erzeugte Variable *zitat* wird nun der globalen Variablen *restoreZitat* der Klasse *MasterViewController* zugewiesen. Zur Übergabe wird dann die Methode *performSegueWithIdentifier* aufgerufen.

Fazit

Mit der Klasse *NSUserActivity* kann nicht nur die Handoff-Funktion von iOS umgesetzt, sondern seit Version 9 auch die Spotlight-Suche für den App-Inhalt verwendet werden. Apps mit textlastigem Inhalt, die diese Funktion unterstützen, können so von Anwendern via Spotlight bequem durchsucht werden. ■

Christian Bleske
ist Autor, Trainer und Entwickler mit dem Schwerpunkt Client/Server und mobile Technologien. Sein Arbeitsschwerpunkt liegt auf Microsoft-Technologien.
cb.2000@hotmail.de

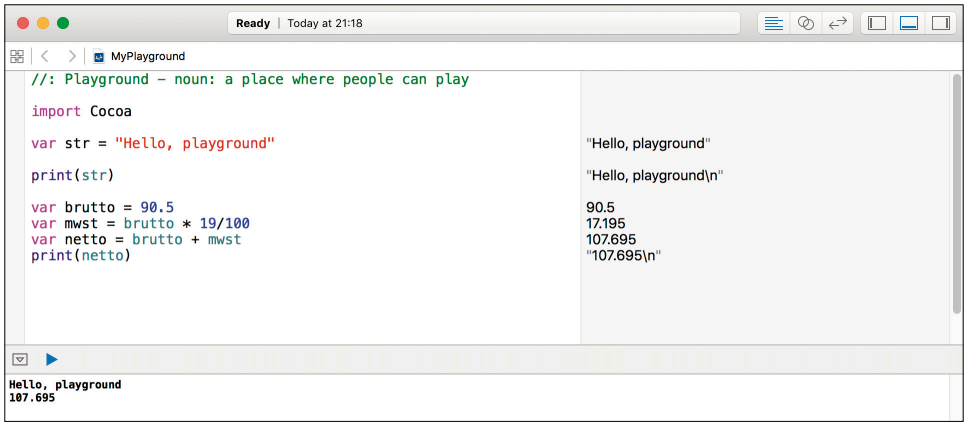
SPEZIALITÄTEN DER PROGRAMMIERSPRACHE SWIFT

Funktionale Aspekte

Eine Auswahl spezieller Konzepte der Programmiersprache Swift von Apple.

Ursprünglich ist Swift laut Apple zwar nicht dazu gedacht, Objective-C zu ersetzen, vielmehr sollte es eine Alternative darstellen. Im Vergleich besitzt Swift jedoch einige Stärken: Sie ist flexibler, geradlinig und nicht zuletzt erweisen sich viele Dinge als unkomplizierter als unter Objective-C, sodass vor allem auch Einsteiger mit schnellen Erfolgserlebnissen rechnen dürfen.

Die Programmierumgebung ist Xcode, das Sie kostenfrei über den Mac App Store installieren können. Die aktuelle Version von Xcode ist 7.3. Im iBooks-Store steht auch ein über 2000-seitiges E-Book von Apple über die aktuelle Sprachversion 2.2 von Swift zum kostenlosen Download bereit.



Xcode-Playground: Die ideale Lern- und Testumgebung bei der Entwicklung von OS-X- und iOS-Apps (Bild 1)

Die folgende Auswahl von Konzepten befasst sich vor allem mit den funktionalen Aspekten der Programmiersprache, denn gerade auf diesem Gebiet hat Swift einiges zu bieten.

Xcode-Playgrounds

Die Playgrounds sind zwar nicht unmittelbar Bestandteil der Programmiersprache Swift, sondern der Programmierumgebung Xcode. Dessen ungeachtet stellen sie gerade beim Erlernen der Sprache ein willkommenes Hilfsmittel dar.

Bei der App-Entwicklung fungieren die Playgrounds gewissermaßen als Notizblock, wenn man schnell einmal etwas ausprobieren oder Code testen möchte. Stellt sich heraus, dass alles einwandfrei funktioniert, kann man den getesteten Code eins zu eins in den Quellcode der App übernehmen. In einem Playground können Sie beliebigen Code eingeben und die Ausgaben sofort sehen. Das heißt, Sie brauchen den Code nicht erst zu kompilieren und das Ergebnis im Simulator zu betrachten. Ein kleines Beispiel zeigt Bild 1.

Der Code bis zur Anweisung *var str = "Hello, playground"* wird von Xcode automatisch eingefügt. In der rechten Spalte sehen Sie die von den aufgerufenen Funktionen ausgegebenen sowie die in den Variablen aktuell gespeicherten Werte. Eine separate Ausgabekonsole ist seit der Xcode-Version 7 im Debug-Bereich integriert.

Diesen können Sie einblenden, indem Sie in der linken unteren Ecke auf das Symbol mit der bei geschlossenem Debug-Bereich nach oben weisenden Pfeilspitze klicken. Sie können Playgrounds als Teil eines Projekts oder für sich alleine erstellen. Beim Anlegen entscheiden Sie, ob es sich um einen iOS- oder um einen OS-X- oder – neu in Xcode 7.3 – um einen tvOS-Playground handeln soll. Der OS-X-Playground fügt die Import-Anweisung für das Cocoa Framework ein, iOS-Apps arbeiten mit UIKit. Playgrounds werden als Datei gespeichert, sodass Sie jederzeit wieder auf den enthaltenen Code zurückgreifen können. Dafür, dass man die Ergebnisse des im Playground eingegebenen Swift-Codes sofort zu sehen bekommt, sorgt eine sogenannte Read-Eval-Print-Loop (REPL). Diese liest neu geschriebenen Code direkt beim Editieren ein und wertet ihn aus.

Benannte und unbenannte Parameter

Swift unterstützt benannte Parameter – »benannt« bedeutet in diesem Fall, dass die betreffenden Parameter beim Aufruf der Funktion mit Namen angegeben werden müssen. Diesbezüglich hat sich das Standardverhalten von Swift 2 gegenüber Swift 1 geändert. Während in der Sprachversion 1 keine Parameter zu benennen sind, trifft dies in Swift 2 nur für den ersten Parameter zu. Ab dem zweiten müssen alle Parameter mit Namen angegeben werden:

```
func multipliziere(zahl1: Int, zahl2: Int, zahl3: Int)
-> Int {
  return zahl1 * zahl2 * zahl3
}
var ergebnis = multipliziere(10, zahl2: 15, zahl3: 20)
print(ergebnis) // Ausgabe: 3000
```

Ein Aufruf der oben definierten Funktion *multipliziere()* mit *multipliziere(10, 15, 20)*, wie man es von anderen Programmiersprachen (und auch von Swift 1) gewohnt ist, würde dagegen eine Fehlermeldung nach sich ziehen. Wenn Sie möch-

ten, dass auch der erste Parameter benannt ist, schreiben Sie den Namen des formalen Parameters einfach zweimal hintereinander:

```
func multipliziere(zahl1 zahl1: Int, zahl2: Int, zahl3:
Int) -> Int {
  return zahl1 * zahl2 * zahl3
}
```

Beim Aufruf muss nun auch vor dem ersten Argument der Name des formalen Parameters, gefolgt von einem Doppelpunkt, angegeben werden:

```
var z1 = -11, z2 = 22, z3 = -33
var erg = multipliziere(zahl1: z1, zahl2: z2, zahl3: z3)
print(erg) // Ausgabe: 7986
```

Ähnlich kann man verfahren, wenn beim Funktionsaufruf für einen Parameter ein anderer Name als der des formalen Parameters verwendet werden soll. In diesem Fall schreibt man in der Definition den externen Namen vor den Namen des Formalparameters:

```
func calculate(zahl1: Int, mal zahl2:
Int, minus zahl3: Int) -> Int {
  return zahl1 * zahl2 - zahl3
}
```

Ein Aufruf der oben definierten Funktion zeigt sehr schön, wie benannte Parameter die Lesbarkeit des Quellcodes verbessern können:

```
var x = 13, y = 17
var erg = calculate(x, mal: y, minus:
100)
print(erg)  // Ausgabe: 121
```

Zu beachten ist, dass vor dem zweiten und dritten Argument der externe Parametername angegeben werden muss. Der Aufruf *calculate (x, zahl2: y, zahl3: 100)* wäre zum Beispiel fehlerhaft. Umgekehrt dürfen innerhalb der Funktion nur die internen Parameternamen verwendet werden. Um einen Parameter, der standardmäßig benannt ist, zu einem unbenannten zu machen, notiert man anstelle eines zweiten Namens einen Unterstrich:

```
func multipliziere(zahl1: Int, _ zahl2: Int, _ zahl3:
Int) -> Int {
  return zahl1 * zahl2 * zahl3
}
```

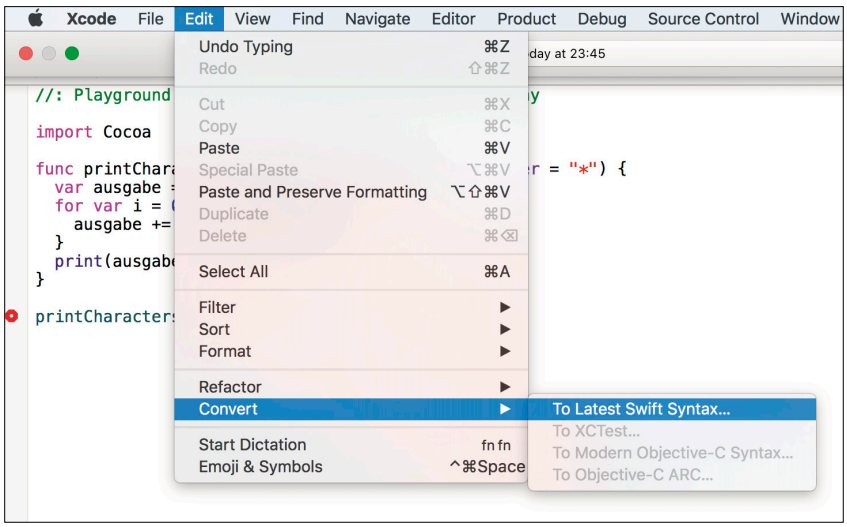
Die so definierte Funktion *multipliziere()* wird komplett ohne Parameternamen aufgerufen. Dies entspricht übrigens den meisten Standardfunktionen von Swift. Diese sind gewöhnlich so eingerichtet, dass keine Parameter zu benennen sind.

Eine Ausnahme bildet zum Beispiel der letzte Parameter von *print()*. Wenn Sie für diesen einen Leerstring angeben, hängt *print()* der Ausgabe kein Newline-Zeichen *(\n)* an.

Swift ist nicht abwärtskompatibel. Das bedeutet, dass Code, der in Swift 1 geschrieben wurde, in Xcode 7 nicht ohne Weiteres kompilierbar ist. Allerdings können Sie veralteten Code von Xcode konvertieren lassen. Wählen Sie dazu in der oberen Menüleiste den Befehl *Edit, Convert, To Latest Swift Syntax*. Die Konvertierungsfunktion ist nicht nur auf Projekte, sondern auch auf im Playground eingegebenen Swift-Code anwendbar (Bild 2).

Variadic-Parameter

Swift-Funktionen lassen sich so einrichten, dass man ihnen beim Aufruf beliebig viele Werte des gleichen Datentyps übergeben kann. Dahinter stehen sogenannte Variadic-Parameter (Variadics). Um einen solchen Parameter zu definieren, hängt man an das Schlüsselwort für den Datentyp drei Punkte an. Innerhalb der Funktion stehen die für den Variadic-Parameter übergebenen Werte in einem Array mit dem Namen des Parameters zur Verfügung. Die folgende Funktion berechnet Summe und Mittelwert der übergebenen Zahlen:



Älteren Code können Sie jederzeit von Xcode konvertieren lassen (Bild 2)

```
func mittelwert(zahlen: Int...)-> Double {
  if zahlen.isEmpty {
    return 0.0
  } else {
    var summe = 0
    for zahl in zahlen {
      summe += zahl }
    return Double(summe) / Double(zahlen.count)
  }
}
```

Mit den Eigenschaften *count* und *isEmpty* lässt sich ermitteln, wie viele Elemente im Array enthalten sind beziehungs- ▶

weise ob das Array leer ist. Letzteres ist hier gesondert zu prüfen, da für einen Variadic-Parameter beim Aufruf der Funktion nicht unbedingt ein Wert übergeben werden muss. Wird kein Wert übergeben, legt Swift für den Variadic ein leeres Array an. Der gewünschte Rückgabewert errechnet sich aus der Summe, geteilt durch die Elementzahl des Arrays.

Zu berücksichtigen ist auch, dass der Mittelwert von ganzen Zahlen Nachkommastellen enthalten kann. Daher muss man den Rückgabetyp der Funktion als *Double* festlegen und vor der Division die Summe und die Elementzahl von *zahlen* entsprechend konvertieren. Um zum Beispiel den Mittelwert der Zahlen von eins bis zehn zu erhalten, könnte man die obige Funktion *mittelwert()* wie folgt aufrufen.

```
var m = mittelwert(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
print("Mittelwert der Zahlen von 1 bis 10: \(m).")
```

Bei Variadic-Parametern existieren ein paar Einschränkungen: Eine Funktion kann nur einen einzigen Variadic-Parameter definieren, und einem Variadic-Parameter darf kein unbenannter Parameter mehr folgen.

Am besten, Sie verfahren so, wie es in Swift 1 noch vorgeschrieben war, und notieren Variadic-Parameter immer ganz am Ende der Parameterliste.

Tupel fassen mehrere Werte zusammen

Eine weitere Besonderheit der Programmiersprache Swift sind Tupel. Sie ermöglichen es, mehrere Werte auf einfache Weise zusammenzufassen und somit als Einheit zu betrachten. Um ein Tupel zu erstellen, notiert man die gewünschten Werte durch Kommas getrennt in runden Klammern:

```
let punkt = (17, -3)
let verlag = ("Neue Mediengesellschaft Ulm mbH", 80335,
"München")
```

Der Zugriff erfolgt wie bei den Arrays über einen nullbasierten Index. Die als Integer festgelegte Postleitzahl in dem oben definierten Tupel *verlag* wird zum Beispiel mit dem Index *1* in der Form *verlag.1* angesprochen:

```
print("Verlagsort: \(verlag.1) \(verlag.2)")
print("Der Punkt hat die Koordinaten \(punkt.0) und \
(punkt.1).")
```

Die Ausgaben der beiden *print()*-Aufrufe lauten »Verlagsort: 80335 München« sowie »Der Punkt hat die Koordinaten 17 und -3.«. Sie können die Werte von Tupeln aber auch benennen. In diesem Fall kann der Zugriff sowohl mit Index als auch mit Namen erfolgen:

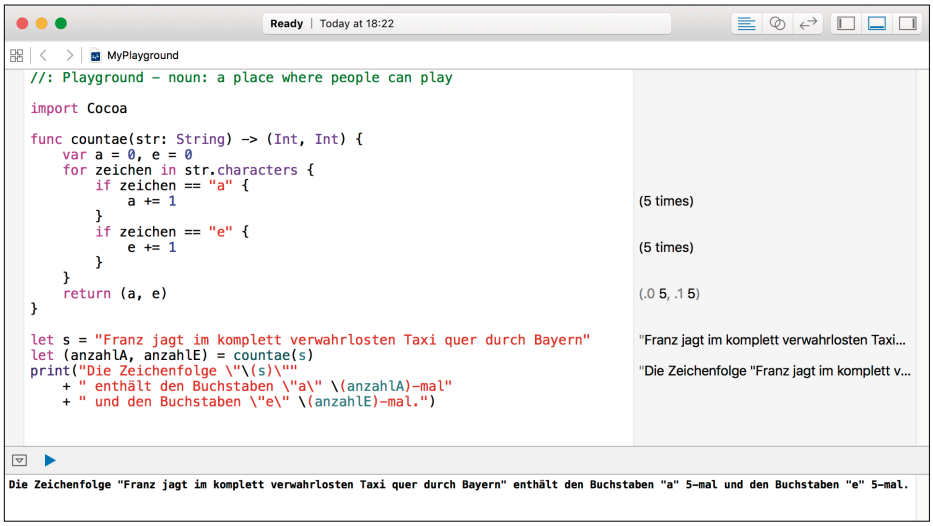
```
let zeitschrift = (name: "web & mobile developer", jahr:
2016, ausgabe: 6)
print("Aktuelle Ausgabe der Zeitschrift \(zeitschrift.
name): ", terminator: "")
print("\(zeitschrift.ausgabe)/\(zeitschrift.jahr)")
```

Die Ausgabe des obigen Listings lautet »Aktuelle Ausgabe der Zeitschrift web & mobile developer: 6/2016«. Die Werte von Tupeln, die mit dem Schlüsselwort *var* deklariert sind, können im Gegensatz zu den mit *let* deklarierten überschrieben werden. Hier ist es sogar möglich, den Datentyp eines Tupels im Voraus festzulegen:

```
var zeitschrift: (name: String, jahr: Int, ausgabe: Int)
zeitschrift = ("web & mobile developer", 2016, 6)
```

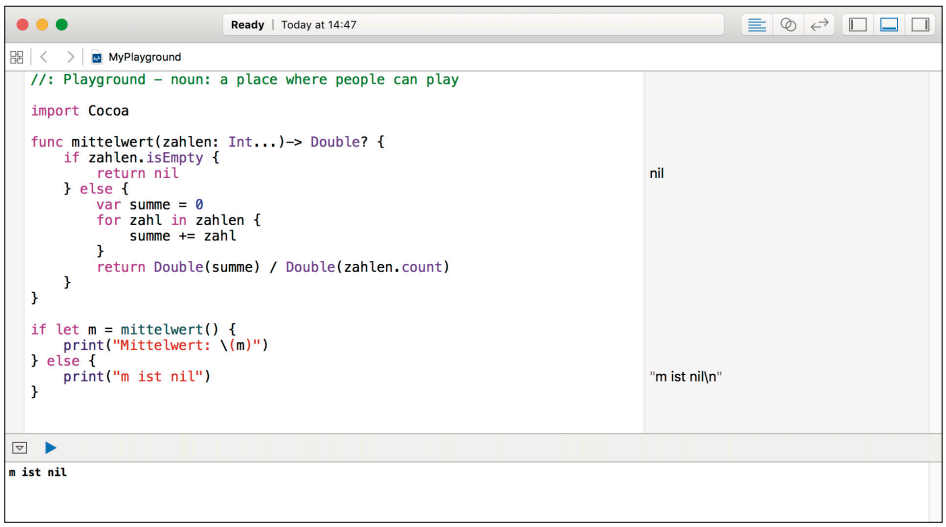
Dem Konzept der Tupel ist es zu verdanken, dass man in Swift Funktionen erstellen kann, die mehrere Werte zurückgeben. Die folgende Funktion ermittelt, wie oft in einer Zeichenfolge die Vokale a und e vorkommen. Zu übergeben ist der Funktion die zu durchsuchende Zeichenfolge:

```
func countae(str: String) -> (Int, Int) {
  var a = 0, e = 0
  for zeichen in str.characters {
    if zeichen == "a" {
      a += 1
    }
    if zeichen == "e" {
      e += 1
    }
  }
  return (a, e)
}
```



Die Funktion *countae()* besitzt zwei Rückgabewerte (Bild 3)

Statt eines einzelnen Wertes oder eines Ausdrucks, der nach Auswertung einen entsprechenden Wert ergibt, notiert man nach *return* ein Tupel mit den benötigten Rückgabewerten – hier die in den Variablen *a* und *e* gespeicherte Anzahl der Zeichen a und e. Zuvor gibt man im Kopf der Funktion



Der Rückgabewert der Funktion *mittelwert()* ist als optional festgelegt (Datentyp *Double?*). Falls die Funktion ohne Zahlenwerte aufgerufen wird, gibt sie *nil* zurück (Bild 4)

nach dem Zeichen -> alle Datentypen der Rückgabewerte in einer kommagetrennten Liste an – für die obige Funktion also *(Int, Int)*. Die oben definierte Funktion *countae()* könnte nun zum Beispiel so verwendet werden:

```
let s = "Franz jagt im komplett verwahrlosten Taxi quer
durch Bayern"
let (anzahlA, anzahlE) = countae(s)
print("Die Zeichenfolge \"\(s)\""
 + " enthält den Buchstaben \"a\" \(anzahlA)-mal"
 + " und den Buchstaben \"e\" \(anzahlE)-mal.")
```

Die Ausgabe sehen Sie in Bild 3. Die Anweisung *let (anzahlA, anzahlE) = countae(s)* speichert die Rückgabwerte der Funktion in zwei Variablen. Natürlich kann man die Rückgabewerte auch in einem Tupel speichern und mit diesem weiterarbeiten, zum Beispiel *let anzahlAE = countae(s)*.

Optionals

Optionals sind Variablen oder Konstanten, die einen gültigen Wert, aber auch *nil* enthalten können. Sie erleichtern damit die Aufgabe, im Programmcode mit unbrauchbaren Werten umzugehen.

Wie in anderen Programmiersprachen fungiert das Schlüsselwort *nil* zwar als Nullzeiger für Objekte, in Swift ist es aber nicht nur auf Referenztypen, sondern auch auf elementare Datentypen anwendbar – hier zeigt es an, dass eine Variable eben keinen richtigen Wert besitzt. Um eine Variable als optional zu kennzeichnen, notiert man ein Fragezeichen hinter dem Datentyp-Bezeichner:

```
var optVar: Int?
```

Der Unterschied zu nicht optionalen Variablen besteht vor allem darin, dass man mit Optionals sofort arbeiten, sie also zum Beispiel in einer *if*-Abfrage auf einen gültigen Wert hin überprüfen kann.

Man spricht im Zusammenhang mit Optionals auch von Wrappern, jedoch nicht im üblichen Sinne, dass Objekte – um sie mit erweiterter Funktionalität auszustatten – innerhalb anderer Objekte verpackt werden. Vielmehr packt Swift die Variable gewissermaßen in eine Blackbox. Aus dieser muss man den enthaltenen Wert, wenn man ihn benötigt, erst wieder herausholen. Swift stellt dafür mehrere Methoden zur Verfügung.

Die eleganteste Methode, den Wert einer optionalen Variablen zu entpacken, besteht in einer *if-let*-Anweisung. Dies geht mit der Prüfung einher. Der Wert wird nur dann in einer temporären Konstanten gespeichert, wenn er tatsächlich existiert:

```
var optZahl: Int?
optZahl = Int(textField.text!)
if let entpackterWert = optZahl {
  print("Entpackter Wert: \(entpackterWert)")
} else {
  print("Ungültige Eingabe")
}
```

Die Konstante *entpackterWert* ist nur innerhalb des *if*-Blocks gültig. Das Besondere an dem Konstrukt ist, dass der Ausdruck *let entpackterWert = optZahl* zu *true* auswertet, falls *optZahl* einen gültigen Wert enthält, andernfalls zu *false*. Schließlich können Sie optionale Werte auch automatisch entpacken lassen. In diesem Fall notieren Sie bei der Definition der Variablen statt eines Fragezeichens ein Ausrufezeichen hinter dem Schlüsselwort für den Datentyp.

Die Notation unterscheidet sich nach der Definition zwar nicht von der gewöhnlicher Variablen. Ungeachtet dessen handelt es sich nach wie vor um eine optionale Variable, die zu jeder Zeit *nil*, also keinen Wert, enthalten kann. In der ▶

Funktionen verschachteln

Wenn man Funktionen innerhalb anderer Funktionen einsetzt, verfolgt man in der Regel den Zweck, Funktionalität in separate Einheiten auszulagern.

Unter diesem Gesichtspunkt ist es nicht immer erwünscht, dass Funktionen, die eigentlich nur dazu bestimmt sind, andere Funktionen zu unterstützen, für sich alleine aufgerufen werden können. Um dies zu verhindern, erlaubt es Swift, Funktionen zu verschachteln. Das heißt, die »Hilfsfunktionen« werden einfach im Block der »Hauptfunktion« definiert.

Regel deklariert man eine Variable jedoch nur dann als *implicit unwrapped optional,* wenn man davon ausgehen kann, dass sie nach der ersten Wertzuweisung immer einen gültigen Wert besitzen wird.

Optionals können als Parameter und vor allem auch als Rückgabewerte von Funktionen fungieren. In Abänderung der Funktion *mittelwert()* des obigen Abschnitts über Variadic-Parameter gibt die Funktion aus Bild 4 nicht *0*, sondern *nil* zurück, wenn der Aufruf ohne Angabe von Zahlen erfolgt.

Funktionstypen

Auch Funktionen besitzen in Swift einen Datentyp. Eine Funktion, die einen Parameter vom Datentyp *String* und einen weiteren vom Datentyp *Int* erwartet und nach Ausführung einen booleschen Wert zurückgibt, besitzt zum Beispiel den Datentyp *(String, Int) -> Bool*. Der Datentyp einer Funktion, die ein Integer-Array übernimmt und keinen Wert zurückgibt, lautet *([Int]) -> ()*, und der Datentyp der in Bild 4 definierten Funktion *mittelwert() lautet (Int...) -> Double?*, wobei *Int...* für den Variadic-Parameter steht.

Funktionstypen werden praktisch genauso behandelt wie gewöhnliche Typen. So kann man zum Beispiel eine Variable mit einem Funktionstyp erstellen und ihr eine Funktion zuweisen. Im Weiteren kann die betreffende Funktion dann auch über die Variable aufgerufen werden:

```
var fktVar: (Int...) -> Double?
fktVar = mittelwert
var m = fktVar(1, 2, 3, 4, 5)
print(m!) // Ausgabe: 3.0
```

Der bloße Funktionsname steht für einen Verweis auf die Funktion. Der obigen Variablen *fktVar* könnte später eine andere Funktion des gleichen Typs zugewiesen werden, da sie mit *var* deklariert ist. Dagegen kann der Funktionsverweis bei Konstanten nicht mehr geändert werden. Daraus ergibt sich auch, dass Funktionen andere Funktionen als Parameter übernehmen und auch Funktionen zurückgeben können:

```
func calc(z1: Double, z2: Double, fktMath: (Double,
Double) -> Double) -> Double {
  return fktMath(z1, z2)
}
```

Die obige Funktion übernimmt zwei Double-Werte und führt mit diesen entsprechend der für den dritten Parameter übergebenen Funktion eine Berechnung durch. Existiert zum Beispiel eine Funktion *multiply()*, die zwei Double-Werte übernimmt und das Produkt dieser Werte zurückliefert, dann könnte *calc()* mit einem Verweis auf diese Funktion folgendermaßen aufgerufen werden:

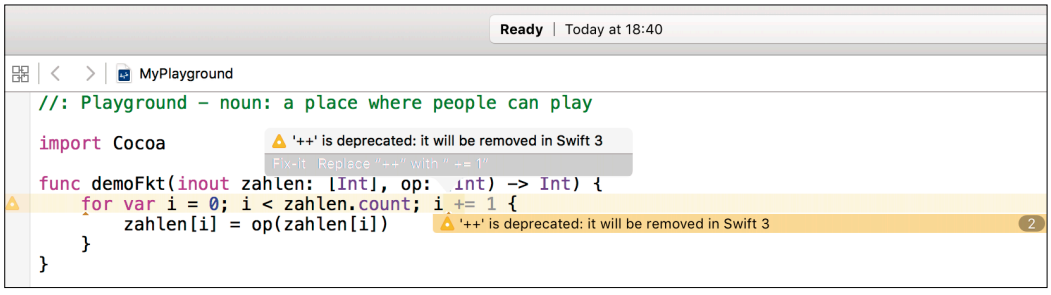
```
var erg = calc(118.98, z2: -217.5, fktMath: multiply)
print(erg) // Ausgabe: -25878.15
```

Die Angabe *fktMath: (Double, Double) -> Double* in der Parameterliste von *calc()* ist beim Aufruf der Funktion in etwa so zu interpretieren: »Definiere eine Konstante mit dem Namen *fktMath* und weise ihr einen Verweis auf eine Funktion zu, die zwei Double-Parameter übernimmt und nach Ausführung einen Double-Wert an den Aufrufer zurückgibt«.

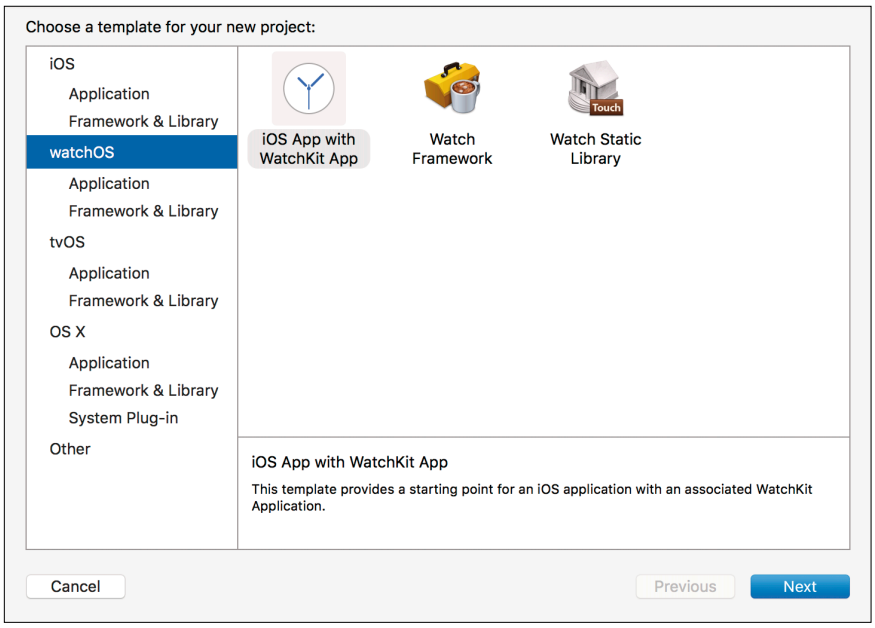
Für den dritten Parameter der Funktion *calc()* kann daher beim Aufruf grundsätzlich jede Funktion des Typs *(Double, Double) -> Double* übergeben werden. Entsprechend verhält es sich, wenn man aus einer Funktion einen Verweis auf eine andere Funktion zurückgeben will: Man notiert im Funktionskopf den passenden Funktionstyp und im Anweisungsteil nach dem Schlüsselwort *return* den Funktionsnamen. Dementsprechend gibt die folgende Funktion *waehleMath-*

Listing 1: Funktion demoFkt

```
func demoFkt(inout zahlen: [Int], op: (Int) -> Int)
{
  for i in 0 ..< zahlen.count {
    zahlen[i] = op(zahlen[i])
  }
}
var demoClosure = { (zahl: Int) -> Int in
  return zahl * zahl
}
var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
demoFkt(&numbers, op: demoClosure)
var ausgabe = ""
for elem in numbers {
  ausgabe += String(elem) + " "
}
print(ausgabe)

// Ausgabe: 1 4 9 16 25 36 49 64 81 100
```



Xcode 7.3 bietet bei Sprachelementen, die zwar aktuell noch gültig sind, aber in der Swift-Version 3 nicht mehr einsetzbar sein werden, eine automatische Korrekturhilfe an (Bild 5)



Xcode stellt mit der Version 7 nun auch Vorlagen mit Apple-Watch-Unterstützung (Betriebssystem watchOS) zur Verfügung (Bild 6)

Fkt() in Abhängigkeit eines zu übergebenden Zeichens eine der drei Funktionen *add(), subtract()* oder *multiply()* an den Aufrufer zurück:

```
func waehleMathFkt(rechenoperation: Character) ->
(Double, Double) -> Double {
  switch rechenoperation {
    case "s", "S":
      return subtract
    case "m", "M":
      return multiply
    default:
      return add
  }
}
```

Closure-Ausdrücke sind in sich abgeschlossene Codeblöcke, die bei Bedarf weitergereicht und ausgeführt werden können. Sie besitzen einen bestimmten Typ und können wie Funktionen Werte entgegennehmen und zurückgeben und auch Variablen oder Konstanten zugewiesen werden:

```
let einClosure: (Int, Int) -> Int
einClosure = { (z1: Int, z2: Int) -> Int in
  return z1 + z2 }
var summe = einClosure(77, -11)
print(summe) // Ausgabe: 66
```

Da die Konstante *einClosure* mit *let* deklariert ist, kann die Verbindung mit dem zugewiesenen Closure nicht mehr geändert werden. Für die Zuweisung an Variablen spielt es ansonsten keine Rolle, ob es sich um einen Closure-Ausdruck oder um eine Funktion handelt – ausschlaggebend ist allein der Datentyp. Ganz allgemein kann man Closure-Ausdrücke überall dort einsetzen, wo ein Funktionstyp erwartet wird, also auch als Parameter von Funktionen (Listing 1).

Die Funktion *demoFkt()* übernimmt den Verweis auf ein Integer-Array sowie einen Parameter vom Typ *(Int) -> Int*. Das Schlüsselwort *inout* bewirkt die Übergabe eines Verweises, Änderungen erfolgen also direkt am übergebenen Array des Aufrufers.

Die Art und Weise, wie das Array manipuliert wird, bestimmt allein der beim Aufruf übergebene Closure-Ausdruck, der hier in den einzelnen Elementen des Arrays jeweils das Quadrat des ursprünglichen Wertes speichert. Stattdessen könnte aber auch jeder andere Closure-Ausdruck oder eine Funktion des Typs *(Int) -> Int* mit einem ganz anderen Ergebnis an die Funktion *demoFkt()* übergeben werden.

Der eigentliche Vorteil von Closure-Ausdrücken gegenüber Funktionen besteht darin, dass Erstere anonym sind. Das heißt, man kann auf das Definieren von Funktionen und auch auf den Umweg über eine Variable verzichten, wenn Code nur einmal benötigt wird. Man notiert den Closure-Ausdruck in der Aufrufliste einfach an der Stelle, an der die Funktion den entsprechenden Parameter erwartet. Verzichtet man also in obigem Listing auf die Variable *demoClosure*, dann sieht der Aufruf von *demoFkt()* mit dem gleichen Ergebnis so aus:

```
demoFkt(&numbers, op: { (zahl: Int) -> Int in return
zahl * zahl })
```

Wenn der Closure-Parameter wie hier der letzte in der Parameterliste ist, darf man den Closure-Ausdruck sogar ohne Angabe des Parameternamens hinter die Funktionsklammern schreiben.

Fazit

Swift ist eine interessante und zudem äußerst flexible Programmiersprache, die mit modernen und teilweise völlig neuen Konzepten aufwartet (Bild 5, Bild 6). Sie bekommen zusammen mit Xcode alles in die Hand, was Sie für die erfolgreiche Entwicklung von iOS- und OS-X-Apps benötigen. ■

Walter Saumweber
hat langjährige Erfahrung als Entwickler, Berater und Dozent. Er ist Autor von zahlreichen Fachbüchern und Beiträgen in Computer-Fachzeitschriften.

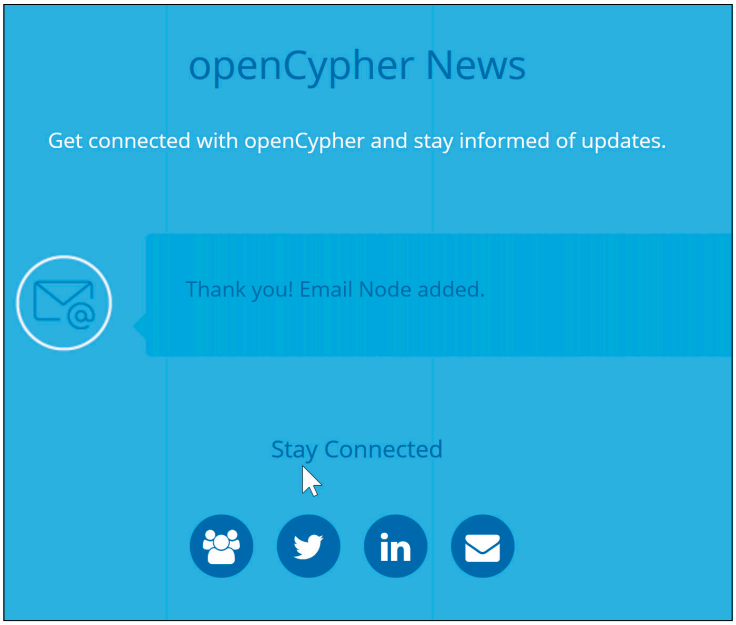
ERFOLGREICHE DATENHALTUNG MIT CYPHER

Mächtige Sprache

Im Zentrum von Neo4j steht CQL (Cypher Query Language) – eine mächtige Abfragesprache.

Den Erfolg einer Datenbankanwendung stellt nur eine ausreichende Flexibilität, Effizienz und Effektivität bei der Verarbeitung vorhandener Datenbestände sicher. Hierzu reicht die Implementierung von Maßnahmen und Verfahren des Datenmanagements alleine nicht aus. Vielmehr muss man gewährleisten, dass sich die Zugriffe auf Daten entsprechend den anfallenden Aufgaben und gemäß den tatsächlichen Anforderungen erledigen lassen. Insbesondere gilt es, die sogenannten Altdaten (Legacy Data, Stammdaten), also alle bereits vorhandenen Bestände in unterschiedlichen Datenhaltungssystemen, zu inkludieren. Zudem nimmt die NoSQL-Technologie für die zunehmende Anzahl an Web- und Mobile-Anwendungen eine maßgebliche Rolle ein.

Einen entscheidenden Schlüssel zum Erfolg bilden die seitens des Datenbank-Management-Systems (DBMS) implementierten Features. Diese müssen für die Bereitstellung, Verwaltung und Bearbeitung der Daten möglichst den maximalen Nutzen für das Unternehmen erzielen. Der Deckungsgrad der Digitalisierung ist anhand des Mehrwerts für die Geschäftsprozesse im Auge zu behalten. Speziell muss heutzutage die Unterstützung für das Internet stärker als bisher berücksichtigt werden. Ergänzend kommen Überlegungen zur Wirtschaftlichkeit bei Betrieb und Administration der Datenbank hinzu. Diese Ansprüche verdeutlichen den hohen und damit strategischen Stellenwert, den eine Auswahl des DBMS auf der Unternehmensebene – im Unterschied zur Abteilungsebene – einnimmt.



Nach Eintrag einer E-Mail-Adresse in die Newsliste des openCypher-Projekts erhält man automatisch Informationen zu Neuerungen (Bild 1)

Durch Open Source und Mainstream zum zukünftigen Sprachstandard

CQL, von Neo Technology eigens für Neo4j konzipiert, stellt analog zu SQL in der relationalen Welt nicht nur eine reine Abfragesprache für die Daten der Graph-Datenbank dar. Vielmehr unterstützt CQL auch die Definition von Graphen inklusive konkreter Daten. Darüber hinaus bietet CQL als weiteres Feature das Bearbeiten (Einfügen, Verändern, Löschen) von Elementen des Graphen an. Somit entspricht CQL einer vollwertigen Datenbanksprache, die speziell auf Graph-Datenbanken ausgerichtet ist.

Als Sprache basiert CQL auf der Verarbeitung von Mustern (Patterns) und rückt diese ins Zentrum. Zudem besitzt die um diese Muster aufgebaute Sprache eine relativ einfache Syntax, ist also leicht erlernbar, da an die englische Umgangssprache angelehnt, und gleichzeitig gut lesbar. Zusätzlich legte Neo Technology besonderen Wert auf die Berücksichtigung wichtiger, im Umfeld von Graphen notwendiger Algorithmen. Für diese Zwecke ergänzte Neo Technology CQL um zusätzliche Schlüsselwörter und Funktionen, beispielsweise für die Suche in Graphen oder für dessen Traversieren.

Neo Technology hat seit einiger Zeit den besonderen Stellenwert von Cypher für die Welt der Graph-Datenbanken erkannt und deshalb ein Open-Source-Projekt namens openCypher über das Web gestartet (Bild 1). Aufgrund der marktbeherrschenden Position von Neo4j kann Cypher als die am häufigsten eingesetzte Sprache für Graph-Datenbanken angesehen werden. Diese beiden Aspekte tragen maßgeblich dazu bei, mittels openCypher einen Sprachstandard für Graph-Datenbanken etablieren zu können.

Als Open-Source-Projekt liefert openCypher analog dem SQL-Standard verschiedene Arbeitsergebnisse:

- **Dokumentation der Sprache:** inklusive ausführlicher Beispiele und Tutorials;
- **Spezifikation von Cypher:** um die Datenbanksprache als Drittthersteller für die eigene Graph-Datenbank oder Datenwerkzeuge realisieren zu können;
- **Referenzimplementierung:** Freigabe zentraler Technologien der Sprache als Beispiele oder Grundlage für die eigene Umsetzung;
- **Testkit:** mehrere vorgegebene Tests für die Prüfung der Sprachkonformität einer konkreten Implementierung.

Die zunehmende Verbreitung von Neo4j als führende Graph-Datenbank und der Wunsch von Anwendern, Graphen in eigenen Anwendungen zu nutzen, rückte Cypher zur Problemlösung in den Fokus. Die Sprache eignet sich sowohl für Entwickler als auch für Endbenutzer mit wenig Programmiererfahrung. Diese Bedeutung von CQL bewog einige Neo4j-Anwender, im openCypher-Projekt mitzuarbeiten.

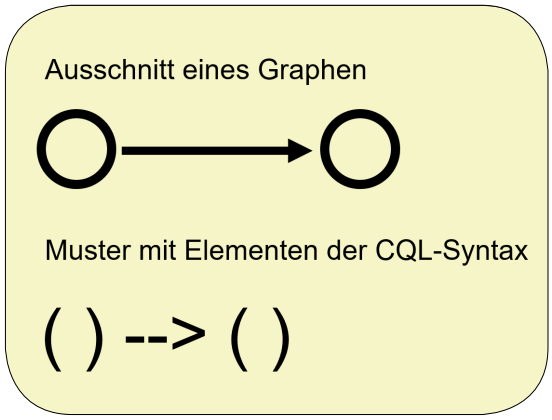
Zudem führte der Trend, Graph-Datenbanken – speziell Neo4j – in der Breite einzusetzen (Mainstream), weitere wichtige Firmen wie Datenbankhersteller und Softwarehäuser dazu, am Open-Source-Projekt teilzunehmen. Dazu gehören neben Neo Technology unter anderem Databricks (die Firma hinter Apache Spark), GraphAware, Linkurious, Oracle, Structr, Tableau und Tom Sawyer Software.

Grundlegende Bausteine der Cypher (CQL)-Syntax

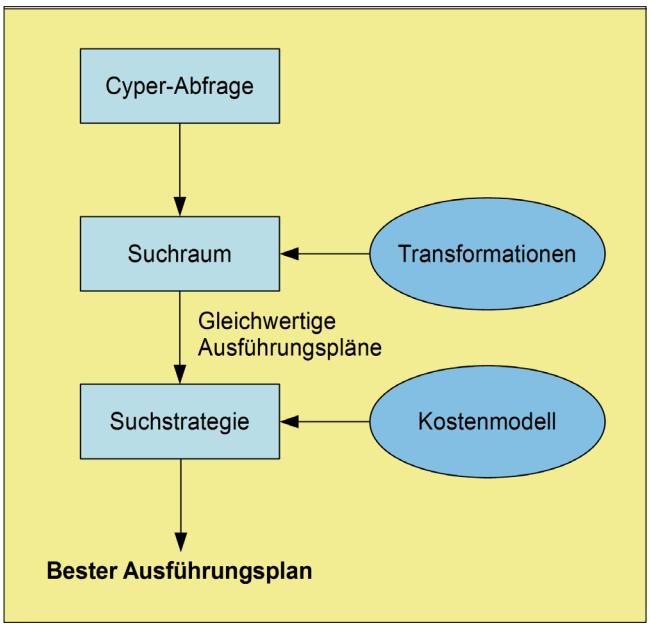
Eine Graph-Datenbank besteht aus Entitäten (Objekte der realen Welt), dargestellt als Knoten (Nodes), und aus Beziehungen (Relationships) als Kanten. Hinzu kommen an diesen Knoten und Kanten als Attribute eines Objekts oder einer Beziehung zusätzliche Eigenschaften (Properties). Ergänzend können Knoten, also Objekte der realen Welt, mit Labeln versehen werden. Aus diesen Elementen/Bestandteilen des Datenmodells eines Graphen leiten sich die nachfolgenden Symbole als grundlegende Bausteine der Syntax von Cypher ab:

- **( ) für ein Objekt (Entität):** stellvertretend für einen beliebigen Knoten (Node) des Graphen;
- **{ } für eine Eigenschaft/Attribut:** stellvertretend für eine beliebige Property (Eigenschaft/Attribut) eines Objekts oder einer Beziehung des Graphen;
- **– [ ] – für eine Beziehung:** stellvertretend für eine beliebige Kante des Graphen. Die Richtung einer Kante/Beziehung verdeutlicht das auf den – folgende Pfeil-Symbol > (Bild 2).
- **nameObjA:nameLabelX:nameLabelY ... für ein Objekt des Graphen mit Labeln:** stellvertretend für die Label mit dem Namen *nameLabelX* und *nameLabelY* beim Objekt (Entität/Kante) mit dem Namen *nameobjA*.

Um Ergebnisse einer Datenbankabfrage zu erhalten, kombiniert man diese vier Syntax-Symbole von Cypher auf verschiedene Art und Weise. So kann man beim Durchsuchen der Daten zusätzlich eine Eigenschaft hinzufügen oder Filter als weitere Kriterien verwenden, um die Anzahl beziehungsweise Größe der Ergebnismenge einzuschränken. Ein solcher Filter zur Beschränkung der Ergebnisse einer Abfrage lässt sich sowohl für Objekte als auch für Beziehungen definieren. Filter dienen nicht einfach nur dazu, die Ergebnismenge zu verkleinern. Vielmehr eignen sie sich speziell dazu, die tatsächlich relevanten Daten des Graphen zu ermitteln.



CQL definiert (mit ASCII-Zeichen als Bausteine) das gesuchte Muster als Ausschnitt des Graphen (Bild 2)



Ausgehend von einer Cypher-Abfrage generiert die Query-Engine von Neo4j einen Suchraum, bedient sich Transformationen, um zunächst gleichwertige, und anschließend eines Kostenmodells, um den besten Ausführungsplan zu bestimmen (Bild 3)

Die zuvor eingeführten Basisbausteine der CQL-Syntax hat Neo Technology um Symbole für Stellvertreter (Variable oder auch Alias genannt) erweitert. Erst dieses weitere Syntax-Element erlauben es, echt flexible Abfragen über den Graphen der Datenbank und dessen Datenbestände zu definieren. Verwendet man einen Alias, so erlaubt die Cypher-Anweisung eine mengenmäßige Verarbeitung des kompletten Abfrageergebnisses. Erläuterungen in einer CQL-Anweisung für einen Leser nimmt man über Kommentare, in Anlehnung an Programmiersprachen mittels zwei //-Zeichen vor. Alles, was in einer Zeile nach // folgt, stellt einen Kommentar dar und erklärt Formulierung und Aufbau der CQL-Abfrage.

Schlüsselwörter erweitern maßgeblich den Sprachumfang

Generell stellt CQL analog SQL eine deklarative Sprache dar. Eine CQL-Abfrage fokussiert sich auf das, was als Ergebnis erwartet wird, und nicht darauf, wie man zu diesem Ergebnis kommt. Man formuliert lediglich das Was und nicht das Wie in der Abfrage und entlastet damit den Benutzer, da sich dieser beim Schreiben der CQL-Anweisung ganz auf das erwartete Ergebnis konzentrieren kann. Zudem ermöglicht der deklarative Ansatz von Cypher (durch eine Beschreibung der ▶

gesuchten Eigenschaften die zu ermittelnde Ergebnismenge festzulegen) die Optimierung einer CQL-Abfrage. Den auszuführenden Algorithmus, um letztendlich zu den tatsächlichen Ergebnissen zu kommen, leitet Neo4j mittels einer Query-Engine direkt aus der CQL-Abfrage ab (Bild 3).

Zusätzlich besitzt Cypher auch imperative Sprachelemente (wesentlich mehr als SQL); sie beeinflussen den Algorithmus, um die Ergebnisse der Abfrage direkt zu bestimmen. So kann man einen Startpunkt vorgeben und die Beziehungen sowie ihre Richtung festlegen, über die eine Suche im Graphen erfolgen soll. Auch eine Sortierung, Gruppierung oder mengenmäßige Bearbeitung der Ergebnismenge einer CQL-Abfrage lässt sich spezifizieren. Allerdings bestimmt nur die Query-Engine, welche Optimierungsstrategien anzuwenden sind, und nimmt die geeignete Art der Suche im Graphen (Breadth-First, Depth-First und andere) vor. Der Sprachschatz besteht neben den grundlegenden Syntax-Bausteinen aus weiteren Schlüsselwörtern (Keywords), die sich stark an SQL orientieren:

- **Allgemeine Konstrukte:** zur Rückgabe von Ergebnissen *(RETURN)*, zum Sortieren *(ORDER BY)* oder zur Beschränkung von Ergebnissen *(SKIP, LIMIT)*;
- **Konstrukte zum/beim Lesen:** zum Erfüllen einer Bedingung *(MATCH)* oder *(OPTIONAL MATCH)*, zur Beschränkung einer Treffermenge *(WHERE)* oder zur Bildung von Datensammlungen (Aggregationen, Mengen) *(COUNT, COLLECT, DISTINCT, UNION)*;
- **Konstrukte zum/beim Aktualisieren, Bearbeiten, Erzeugen oder Schreiben:** Anlegen von Knoten und/oder Kanten eines Graphen *(CREATE, MERGE)*, Setzen von Eigenschaften/Attributen *(SET)*, Löschen von Knoten/Kanten oder ihrer Properties/Label *(DELETE, REMOVE)*.

Eine vollständige Liste aller Schlüsselwörter findet man in der Referenzkarte der Sprache. Zusätzlich hat Neo Technology die Mächtigkeit von CQL um Funktionen erweitert, die bestimmte Berechnungen, Prüfungen oder Verkettung von Zeichenketten oder Datensammlungen (Collections) durchführen. Bei der Formulierung von Anweisungen unterscheidet Cypher nicht zwischen Groß- oder Kleinschreibung bei den Schlüsselwörtern der Befehle. Dennoch sollte man innerhalb eines Projekts eine einheitliche Schreibweise festlegen und beispielsweise alle Schlüsselwörter von Cypher immer groß, alles andere klein schreiben. Diese Vorgehensweise unterstützt eine leichtere Lesbarkeit, Einarbeitung und Wartbarkeit von Cypher-Befehlsfolgen.

Die Semantik von Cypher zur Verarbeitung von Collections (Sammlungen) lehnt sich an Sprachen wie Haskell oder Python an. Generell baut die Struktur einer Abfrage auf der Verkettung von (im Wesentlichen) einfachen Konstrukten (Clauses) auf. Das erste Konstrukte ermittelt eine Ergebnismenge, die an die danach folgenden Konstrukte quasi weitergereicht wird, bis die komplette Abfrage mittels bestimmter Mechanismen (Pipes, Filter) abgearbeitet wurde. Allerdings bestimmt nur das DBMS den sogenannten Execution Plan, eine Menge an Start-Knoten oder -Kanten mit kombinierter Vorgehensweise, um anschließend mit ihnen eine Traversierung des Graphen für die Ermittlung der Ergebnismenge durchzuführen.

Überblick zu den CRUD-Operationen in Cypher

Das Akronym CRUD (manchmal auch CDUR in Anlehnung an die C-Dur-Tonleiter genannt) steht für die grundlegenden Datenbankoperationen: Create (Datensatz anlegen), Read/Retrieve (Datensatz lesen), Update (Datensatz aktualisieren) und Delete/Destroy (Datensatz löschen). Abhängig von der für Neo4j in der Programmierung eingesetzten Umgebung gibt es viele spezifische CRUD-Operationen; dieser Abschnitt beschränkt sich auf die von der Cypher Query Language (CQL) bereitgestellten Sprachelemente. Tippt man in der Befehlszeile des Neo4j-Browser *:help cypher* ein, so erscheint eine Zusammenfassung zu den wichtigsten CQL-Befehlen mit einem Verweis auf die Neo4j-Cypher-Dokumentation und den Cypher-Guide (Bild 4).

Der Cypher-Guide hilft beim Kennenlernen der Sprache; er entspricht einer Einführung in Cypher mit schrittweisen Anleitungen zum Einsatz der wichtigsten CQL-Befehle. Eine Eingabe von *:play cypher* in der Neo4j-Browser-Befehlszeile öffnet ohne Umwege direkt den Cypher-Guide. In CQL ste-

Pattern-Matching und Graph-Traversal

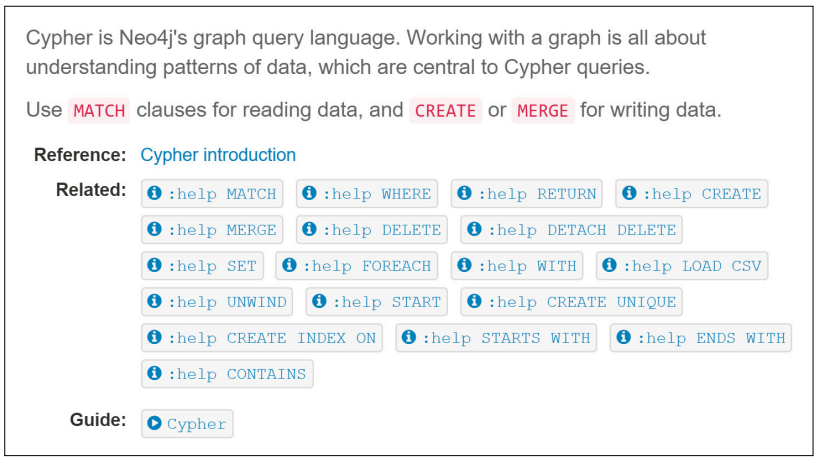
Für das Durchsuchen der Elemente eines Graphen gibt es im Wesentlichen zwei Ansätze.

- **Pattern-Matching (musterbasierte Suche)**: Gegeben ist ein Muster als Beispiel eines Graphen mit beliebigen Knoten und Kanten; gesucht sind alle Ausschnitte/Abschnitte des Graphen, die diesem vorgegebenen Muster entsprechen.
- **Graph-Traversal (Bestimmen eines Weges/einer Route):** Das Traversieren eines Graphen gehört zu den grundlegenden Aufgaben bei seiner Verarbeitung. Die Traversieren-Operation ist mit der Join/Verbund-Operation relationaler Datenbanken vergleichbar. Gegeben sind ein Start-Knoten und ein Algorithmus oder eine Vorgehensweise, die den Weg im Graphen beschreibt und so das Ergebnis der Abfrage bestimmt.

CQL ist leider eine mehrdeutige Abkürzung

Die Bedeutung einer Abkürzung in der Informatik ist häufig vom Kontext ihres Gebrauchs abhängig; das gilt leider auch für CQL.

- **Cypher Query Language** – Abfragesprache für die Graph-Datenbank Neo4j;
- **Contextual Query Language** (bis 2004 Common Query Language) – Sprache zur Formulierung von Anfragen an Bibliothekskataloge;
- **Cassandra Query Language** – die Abfragesprache der verteilten NoSQL-Datenbank Cassandra (Wide-Column Store).



Ein Klick auf einen der :help-Befehle erläutert dessen Syntax mit Anwendungsbeispiel und verweist auf weitere relevante Befehle (Bild 4)

hen für Knoten und Kanten die folgenden an SQL orientierten Schlüsselwörter als CRUD-Operationen zur Verfügung:

- **C(reate):** Der *CREATE*-Befehl für die Anlage eines Knotens oder einer Kante. Im Fall einer Kante müssen die beteiligten Knoten bereits in der Datenbank existieren. Knoten und Kanten ohne Namen können immer über eine interne, von Neo4j vergebene ID angesprochen werden. Zusätzlich besitzt noch der mit Neo4j Version 2.0 eingeführte *MERGE*-Befehl C(reate)-Character.
- **R(ead/Retrieve):** Der *MATCH*-Befehl, um mittels einer ASCII-Zeichenkette als Muster/Pattern alle dazu passenden Elemente in der Graph-Datenbank zu finden. Zusätzlich besitzt noch der mit Neo4j Version 2.0 eingeführte *MERGE*-Befehl R(ead/Retrieve)-Character.
- **U(pdate):** Die *SET*-Klausel in Zusammenhang mit einer Read/Retrieve-Operation. Mittels *SET* initialisiert oder ändert man die Werte vorhandener Properties (Attribute/Eigenschaften) von Knoten und Kanten/Beziehungen oder die Label der Knoten.
- **D(elete):** Die *DELETE*-Klausel im Zusammenhang mit einer Read/Retrieve-Operation. Nachdem man die gewünschten Knoten oder Kanten/Beziehungen bestimmt hat, entfernt eine direkt folgende *DELETE*-Klausel diese aus der Datenbank. Ein Knoten lässt sich nur löschen, wenn alle zu ihm gehörenden Kanten ebenso aus dem Graphen entfernt werden. Mittels *DETACH DELETE* löscht man einen Knoten und automatisch alle zu ihm gehörenden Kanten.

Eigenschaften/Attribute und/oder Label der Elemente eines Graphen löscht man mittels der *REMOVE*-Klausel – diese stellt ebenfalls eine D(elete)-Operation da. Die *REMOVE*-Klausel folgt wie *DELETE* einer Lese-Operation (Read/Retrieve), um zuvor die zu löschenden Elemente des Graphen zu bestimmen. Ein Stellvertreter einer definierten Variablen/Alias entfernt das zugehörige Attribut und/oder Label aus dem Graphen.

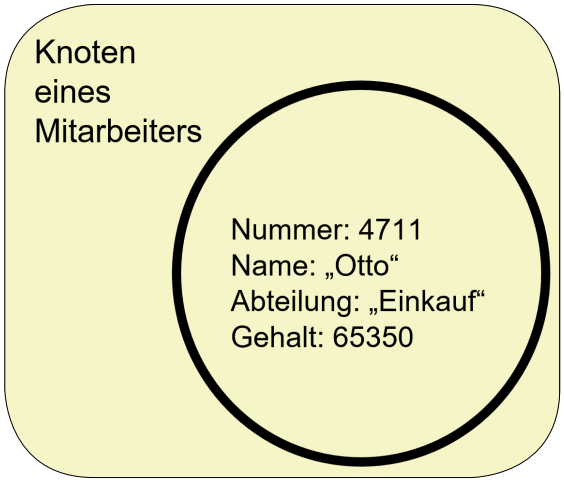
Dabei kennzeichnet ein Doppelpunkt vor dem Namen des Stellvertreters ein Label; während ein einzelner Punkt eine Property (Eigenschaft/Attribut) markiert. So löscht *REMOVE varX.nameA* die Eigenschaft mit der Bezeichnung *nameA* und *REMOVE varX:LabelA* das Label mit der Bezeichnung *LabelA* bei allen zuvor mittels einer Lese-Operation (Read/Retrieve) bestimmten Elementen des Graphen.

Neo4j-Datentypen für Eigenschaften/Properties

Das Datenmodell von Neo4j entspricht einem sogenannten Property-Graphen, dessen Knoten und Kanten man jeweils um Properties (Eigenschaften/Attribute) anreichern kann. Fügt man Properties/Attribute den Knoten und Kanten hinzu, so besitzt der Graph insgesamt eine höhere Semantik als ohne diese. Properties/Attribute stellen intern Key-Value- oder Schlüssel-Wert-Paare da. Beim Key/Schlüssel handelt es sich um eine Zeichenkette/String, dem ein Value/Wert zugeordnet ist (Bild 5).

Im Unterschied zu den CQL-Schlüsselwörtern spielt die Groß- oder Kleinschreibung bei den Key-Value/Schlüssel-Wert-Paaren eine Rolle. In der Theorie der Programmiersprachen bezeichnet man einen Wert gelegentlich auch als Literal. Ein Literal drückt aus, dass es sich um keine Variable, sondern um eine Zahl, Text oder anderweitige Information handelt, die direkt einen Wert darstellt. Ein solcher Wert nimmt einen beliebigen Neo4j-Datentyp an, dieser orientiert sich an den primitiven Datentypen der Programmiersprache Java:

- **boolean:** Stellt eines der beiden booleschen Literale *true* oder *false* dar. Neo4j speichert den booleschen Wert zwar in Kleinbuchstaben, toleriert aber Großbuchstaben bei Abfragen.
- **byte:** Repräsentiert eine 8-Bit ganze Zahl mit dem Wertebereich -128 bis 127 (inklusive).
- **short:** Repräsentiert eine 16-Bit ganze Zahl mit dem Wertebereich -32768 bis 32767 (inklusive). ▶



Der Knoten Mitarbeiter enthält vier Key-Value-Paare mit folgenden Neo4j-Datentypen: den Key *Nummer* mit einem *short*-Value, die beiden Keys *Name* und *Abteilung* mit jeweils einem *String*-Value sowie den Key *Gehalt* mit einem *long*-Value (Bild 5)

- **int:** Repräsentiert eine 32-Bit ganze Zahl mit dem Wertebereich -2147483648 bis 2147483647 (inklusive).
- **long:** Repräsentiert eine 64-Bit ganze Zahl mit dem Wertebereich 9223372036854775808 bis 9223372036854775807 (inklusive).
- **float:** Entspricht einer 32-Bit-Gleitkommazahl (Gleitpunktzahl/Fließkommazahl/Floating Point Number) gemäß der IEEE-Norm 754.
- **double:** Entspricht einer 64-Bit-Gleitkommazahl (Gleitpunktzahl/Fließkommazahl/Floating Point Number) gemäß der IEEE-Norm 754.
- **char:** Stellt ein 16-Bit-Unicode-Zeichen dar mit dem Wertebereich \u0000 (oder 0) als Minimum und \uffff (oder 65535) als Maximum.
- **String:** Repräsentiert eine Zeichenkette, die eine Folge aneinandergereihter Unicode-Zeichen darstellt.
- **Array/Feld:** Darunter versteht man eine Folge von Werten desselben Datentyps aus den obenstehenden Grundtypen. Neo4j unterstützt kein Array/Feld, das sich aus unterschiedlichen Datentypen zusammensetzt. Das erste Element einer solchen Folge von Werten legt den Datentyp des gesamten Arrays/Feldes fest.

Den Datentyp des Wertes eines konkreten Schlüssel-Wert-Paares legt man zwar beim erstmaligen Anlegen fest; er bleibt danach allerdings nicht fix, sondern passt sich bedarfsorientiert aufgrund des Wertes an. Dabei führt Neo4j für die Umwandlung des Datentyps automatisches Casting durch. Handelt es sich beim Wert eines Schlüssels um ein Array/Feld, so kommt auch abhängig vom ersten Element ebenfalls ein Casting zum Einsatz. Entspricht das erste Element eines Arrays/Felds einer Zeichenkette, so wandelt Neo4j alle danach folgenden Elemente ebenfalls in einen String um. Gemäß diesen Regeln nehmen beim Array/Feld mit den Elementen: *Otto, 4711, true, 65350* alle den Datentyp String ein.

Deklarative/deskriptive oder imperative Programme?

Imperative Programmierung sollte man nur anwenden, wenn das Problem sich nicht anders lösen lässt.

Beim deklarativen (im deutschen Sprachraum auch deskriptiv genannten) Programmierstil formuliert man Regeln und Zusammenhänge, die in ein ausführbares Programm überführt werden.

Die deklarative Programmierung konzentriert sich auf die Problemlösung und nicht auf die Umsetzung.

Deklarative Programme spezifizieren das Problem auf einer hohen Abstraktionsebene und sind damit wesentlich kürzer als imperative Programme.

Deklarative Programme sind zustandsfrei und eignen sich daher besonders für eine parallele Ausführung.

Als Entwickler arbeitet man mit deklarativen Programmen wesentlich produktiver, entsprechende Kenntnisse und Erfahrungen vorausgesetzt.

Neo4j kennt als Datentyp im Unterschied zur relationalen Datenbanken nicht den *NULL*-Wert; allerdings kann man bei einer CQL-Abfrage auf einen *NULL*-Wert prüfen: Dieser liegt genau dann vor, wenn der zugehörige Key/Schlüssel überhaupt nicht existiert. Analog verhält es sich mit dem Date-Datentyp zur Ablage eines Tagesdatums mit Uhrzeit. Für dessen Implementierung speichert man die Anzahl der Sekunden in einem *long*-Datentyp, alternativ in einer für den Menschen lesbaren Form der Art: *YYYY-MM-DD HH:mm:ss* oder durch Referenzen auf eine individuelle Baumstruktur. Legt man ein Tagesdatum mit Uhrzeit in einem *long*-Datentyp (gemäß dem Unix-Timestamp) ab, so kann man direkt Rechenoperationen mit ihm durchführen.

Schrittweiser Aufbau eines Graphen mit dem CREATE-Statement

Einen Graphen baut man immer beginnend bei den Knoten und anschließendem Hinzufügen der Kanten auf. In Neo4j kann es zwar leere Knoten ohne Inhalt geben, die keinen Namen, Eigenschaften oder Label besitzen; allerdings findet man für derartige Graphen in der Praxis keine gängigen Anwendungsfälle. Einen leeren Knoten ohne Property/Eigenschaften im Graphen erzeugt ein einfaches *CREATE (<node-name>:<label-name>)*-Statement. Um einen Knoten vom Typ Mitarbeiter zu erzeugen, greift man auf das Statement *CREATE (a:Mitarbeiter)* zurück. Dieser Cypher-Befehl fügt einen neuen Knoten über den Syntax-Grundbaustein des ()-Pattern/Musters der Graph-Datenbank hinzu. Zusätzlich versieht der *CREATE*-Befehl den neuen Knoten mit dem Label *Mitarbeiter*.

Label unterscheiden verschiedene Typen von Knoten; sie entsprechen in der Anwendungswelt einem Objekt/Entität des zu modellierenden Umweltausschnitts. Man kann von einem Knotentyp auch Untertypen bilden, beispielsweise *Feste_Mitarbeiter* und *Freie_Mitarbeiter*: *CREATE (a:Mitarbeiter:Fest)* oder *CREATE (a:Mitarbeiter:Frei)*. Um einem Knoten vom Typ *Mitarbeiter* Attribute hinzuzufügen, definiert man Properties/Eigenschaften und greift auf den CQL-Befehl. *CREATE (a:Mitarbeiter {Nummer: 4711, Name: "Otto", Abteilung: "Einkauf", Gehalt: 65350})* zurück. Hier setzt man erst-

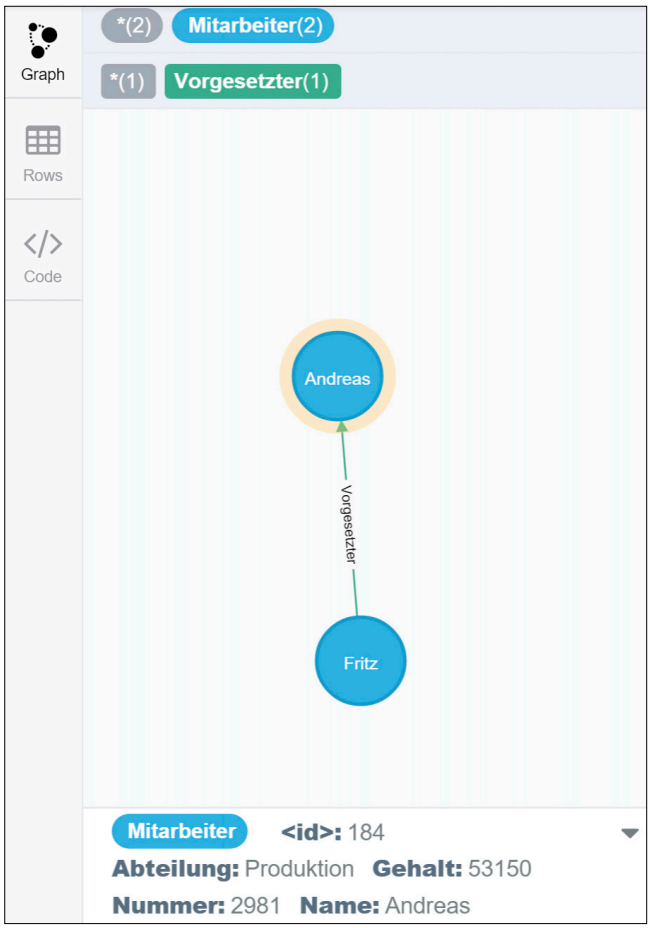
Cypher und Transaktionsmanagement

Generell führt Neo4j jede CQL-Anweisungsfolge innerhalb einer Transaktion aus.

Sollte noch keine Transaktion vorhanden sein, erzeugt Cypher vor Ausführung einer CQL-Anweisung automatisch den erforderlichen Kontext, um die Transaktion managen zu können.

Sollte bereits eine Transaktion bestehen, so führt Neo4j die neue CQL-Anweisung innerhalb des zugehörigen Kontextes durch.

Eventuelle Änderungen der Daten macht Neo4j erst bei erfolgreichem Abschluss der laufenden Transaktion in der Datenbank persistent.



Der Neo4j-Browser zeigt die mittels einer *RETURN*-Klausel übergebenen Knoten und Kanten als Graph oder, bei Klick auf das linke Rows-Symbol, in Zeilenform (Rows) an (Bild 6)

mals die erläuterten Grundbausteine der CQL-Syntax ( ) für Objekt/Entität und { } für eine Eigenschaft/Attribut gemeinsam ein. Mehrere Knoten erzeugt man innerhalb eines *CREATE*-Befehls:

```
CREATE    (a:Mitarbeiter {Nummer: 5761, Name: "Fritz",
Abteilung: "Produktion", Gehalt: 45200}),
(b:Mitarbeiter {Nummer: 3271, Name: "Thomas", Abteilung:
"Entwicklung", Gehalt: 55200}),
(c:Mitarbeiter {Nummer: 2981, Name: "Andreas",
Abteilung: "Produktion", Gehalt: 53150});
```

Einen kompletten Ausschnitt eines Graphen, das heißt alle seine zugehörigen Knoten zusammen mit ihren Kanten/Beziehungen, kann man direkt über ein einziges *CREATE*-Statement anlegen:

```
CREATE    (:Mitarbeiter {Nummer: 9871, Name: "Heinz",
Abteilung: "Verkauf", Gehalt: 65350})
  - [:Vorgesetzter] ->
(:Mitarbeiter {Nummer: 754, Name: "Karl", Abteilung:
"Verkauf", Gehalt: 87200});
```

Der *CREATE*-Befehl erzeugt zwei neue Knoten: den Mitarbeiter Heinz und seinen Vorgesetzten Karl. Beide sind über eine gerichtete Beziehung, die Kante *Vorgesetzter*, miteinander verbunden. Bei der Kante handelt es sich um eine gerichtete Beziehung zwischen Mitarbeiter und Vorgesetztem. In diesem Fall spielt die Richtung der Beziehung eine wichtige Rolle, sie muss auch bei einer Abfrage angegeben werden, da die Richtung festlegt, wer Vorgesetzter und wer Untergebener ist. Für die Definition der gerichteten Kante greift man auf das Pattern/Muster aus den Grundbausteinen – [ ] -> der CQL-Syntax zurück.

Um in der Datenbank bereits bestehende Knoten eines Graphen über eine Kante als Beziehung zu verknüpfen, muss man dem CQL-Befehl zunächst die beteiligten Knoten zugänglich machen. Dies erreicht man mittels einer *MATCH ()*-Abfrage, den ermittelten Knoten weist man jeweils eine Variable zu, um auf sie im anschließenden *CREATE*-Befehl bei der Neuanlage der Beziehung zugreifen zu können. Es handelt sich hierbei um eine aus zwei Teilen bestehende sogenannte Read-und-Write-Query, im ersten Teil liest man die Knoten (hierzu eignet sich jeder Lesebefehl: *START, MATCH, OPTIONAL MATCH, WITH*); der zweite Teil der Abfrage ▶

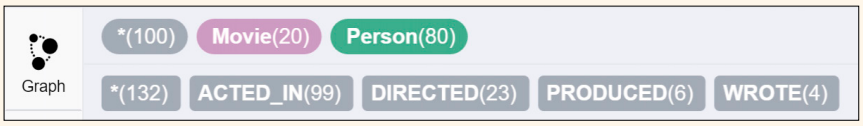
Visualisierung des Graphen im Neo4j-Browser

Nachfolgende Erklärungen helfen bei der Darstellung eines Graphen im Neo4j-Browser.

Im Graph-Fenster zeigt der Neo4j-Browser alle durch die letzte *RETURN*-Klausel spezifizierte Rückgabe-Elemente als Knoten oder Kanten an.

Das Aussehen (Farbe, Größe, Beschriftung) der Knoten oder Kanten im angezeigten Graph-Ausschnitt der Datenbank bestimmt ein sogenannter Style. Generell teilt der Neo4j-Browser den angezeigten Graph-Ausschnitt in die beiden Gruppen Knoten und Kanten ein.

Wählt man aus der vorherigen Gruppierung einen Knoten- oder einen Kantentyp aus, so erscheint unterhalb des Graphen in der letzten Zeile dessen Typ-Name mit den Bereichen *Color, Size, Caption*. Über diese drei Bereiche legt man die Farbe, Größe und die Bezeichnung als Style fest.

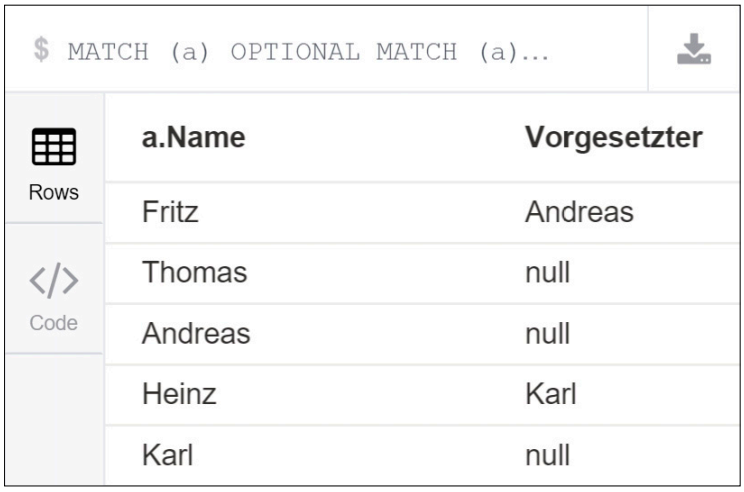


Neben dem Graph-Symbol zeigt der Neo4j-Browser die Summe der Knoten (100) und der Kanten (132) an

führt die Write-Operation durch (das Listing verwendet dazu den *CREATE*-Befehl):

```
MATCH    (a:Mitarbeiter {Nummer: 5761}),
(b:Mitarbeiter {Nummer: 2981})
CREATE    (a) - [r:Vorgesetzter] -> (b);
```

Führt man die diskutierten CQL-Statements im Neo4j-Browser aus, so gibt dieser lediglich positive Erfolgsmeldungen aus. Es erfolgt aber keine visuelle Anzeige des Graphen im Neo4j-Browser. Eine Visualisierung des Graphen im Neo4j-Browser nimmt die *RETURN*-Klausel vor, die es als Option beim *CREATE*- und *MATCH*-Befehl gibt. Ergänzt man beispielsweise die zuletzt vorgenommene Neuanlage einer Beziehung um die Zeile *RETURN a, r, b* –, so zeigt der Neo4j-Browser die beiden Knoten *a, b* sowie die zwischen ihnen neu definierte Kante *r* als Beziehung in Form eines Graphen an (Bild 6). Hätte man in der *RETURN*-Klausel lediglich auf *a, b* zurückgegriffen, so würden nur zwei Knoten, nicht aber die zwischen ihnen neu angelegte Kante *r* im Ausgabefenster des Neo4j-Browsers erscheinen.



Liste aller Mitarbeiter und ihrer Vorgesetzten; keinen Vorgesetzten haben Mitarbeiter mit *null*-Wert in der Vorgesetzten-Spalte (Bild 7)

Gezielte Abfragemöglichkeiten mit CQL

Für Abfragen der Elemente eines Graphen verfügt Neo4j über die beiden Statements *MATCH* und *START*. Anfangs stand im Mittelpunkt einer jeden CQL-Abfrage die *START*-Anweisung. Sie wurde ursprünglich als Ausgangspunkt für die Auswahl von Knoten oder Kanten in Anlehnung an den SQL-Befehlt *SELECT* konzipiert. Auf die *START*-Anweisung folgte als deren Bestandteil ein *MATCH*-Befehl, der ein zusätzliches Pattern/Muster als ergänzendes Suchkriterium definierte. Im Lauf der Weiterentwicklung von Cypher als Abfragesprache und dem Einsatz von Neo4j bemerkte man, dass in der Regel alle Anfragen von Anfang an direkt mit einer Pattern/Muster-Suche beginnen können.

Auf der Grundlage dieses Sachverhalts rückte der *MATCH*-Befehl als neuer Startpunkt ins Zentrum fast jeder Cypher-Abfrage. Direkte *START*-Zugriffe, die nicht auf einem Index basieren, hat Neo Technology mit der Cypher Version 2.0 abgeschafft; aus Kompatibilitätsgründen sind diese jedoch bis Version 2.1 von Neo4j erlaubt. Seit Version 2.2 von Cypher darf die *START*-Anweisung nur noch für Zugriffe auf bestehende Index-Strukturen (Index-Queries) benutzt werden. Demnach konzentriert man sich bei der Formulierung einer Datenbankabfrage auf den *MATCH*-Befehl, der eine optionale *WHERE*-Klausel besitzen kann, aber immer mit einer obligatorischen *RETURN*-Klausel verbunden ist.

Als Parameter übergibt man dem *MATCH*-Befehl eine Folge von Alias/Variablen, ergänzt um Pattern/Muster, dies können beispielsweise *<alias>*, *<node-name>:<label-name>*, *(a)-[:relationship-name]-(b)*, *(<alias>:<label-name>)-[:relationship-name]-(node-name)* oder eine Kombination der Pattern/Muster unter Einbezug von Bedingungen für Label oder Property/Eigenschaften sein. Die *WHERE*-Klausel in Cypher hat eine analoge Aufgabe wie die *WHERE*-Bedingung in SQL: Sie filtert, beziehungsweise schränkt die Abfrageergebnisse aufgrund vorgegebener Bedingungen ein. Dabei kann die *WHERE*-Klausel eine oder mehrere Bedingungen umfassen, die durch boolesche Operatoren *(AND, OR, XOR, NOT)* miteinander verknüpft sind.

Die *RETURN*-Klausel enthält eine beliebig lange Folge von Rückgabe-Parametern, die ein Alias/Variable oder *<node-name>:<property-name>* sein können. Dabei müssen sich die in der Folge angegebenen Rückgabe-Parameter auf die im *MATCH*-Befehl zuvor genannten Übergabe-Parameter beziehen. Das heißt, ein Rückgabe-Parameter der *RETURN*-Klausel muss stets ein Pendant in den Übergabe-Parametern des *MATCH*-Befehls besitzen. Als Ergänzung sind auch Berechnungen oder Konstanten in *RETURN*-Klauseln erlaubt. Die in Listing 1 gezeigten *MATCH*-Abfragen orientieren sich an den zuvor genannten Beispielen für Pattern/Muster und verdeutlichen den Einsatz des *MATCH*-Befehls.

Listing1: MATCH-Abfragen

```
<alias>                MATCH (n) RETURN n;
// Ermittelt den gesamten Graphen als Abfrage-Ergebnis
<node-name>:<label-name>        MATCH (n:Mitarbeiter) RETURN n;
// Ermittelt sämtliche Knoten mit dem Label (vom Typ) Mitarbeiter
(a)-[:relationship-name]-(b)    MATCH (x)-[:Vorgesetzter]->(y) RETURN y;
// Ermittelt sämtliche Vorgesetzte der Datenbank
MATCH (n:Mitarbeiter) WHERE n.Name="Andreas" RETURN n.Gehalt;
// Gibt das Gehalt aller Mitarbeiter mit dem Namen "Andreas" aus
```

SQL-Outer-Join über OPTIONAL MATCH realisiert

Relationale Datenbanken bieten vielfältige Operationen, um Tabellen miteinander zu verknüpfen. Die bekannteste Verknüpfung von Tabellen in einer SQL-Ab-

frage stellt der SQL-Join, die sogenannte Verbundoperation, dar. Im Lauf der Zeit erkannten die Hersteller relationaler Datenbanken die Notwendigkeit, unterschiedliche Typen/Arten von Joins bei der Verknüpfung von Tabellen zu unterstützen. Diese Anforderung resultierte aus dem Einsatz relationaler Datenbanken in der Praxis, um die gemäß einer Anfrage erwartete Ergebnistabelle systemseitig überhaupt bilden zu können. Der ANSI-Standard von SQL92 führte dazu eine spezielle *JOIN*-Syntax ein, um bereits den SQL-Befehl intuitiver und damit verständlicher zu formulieren.

Nach Ausführung eines Cypher-Befehls gibt der Neo4j-Browser dem Anwender immer eine Rückmeldung über die Befehlsausführung als kurze Zusammenfassung aus (Bild 8)

Folgen in einer Cypher-Abfrage mehrere *MATCH*-Befehle unmittelbar aufeinander, stehen diese also direkt hintereinander, so werden sie separat ausgeführt. Ein einzelner *MATCH*-Befehl liefert immer *true/false* zurück, abhängig davon, ob Neo4j das gesuchte Pattern/Muster im Graph finden konnte oder nicht. Neo4j verknüpft logisch aufeinander folgende *MATCH*-Befehle standardmäßig durch eine *AND*-Operation. Nur wenn das boolesche Gesamtergebnis aller *AND*-Operationen *true* liefert, führt Neo4j die nachfolgenden Klauseln aus und ermittelt die *RETURN*-Parameter. Um dieses Default-Verhalten bei der Verknüpfung mehrerer *MATCH*-Befehle von *AND* auf *OR* zu ändern, steht seit Neo4j Version 2.0 der Befehl *OPTIONAL MATCH* zur Verfügung.

OPTIONAL MATCH erzeugt aus einem nicht erfüllbaren Pattern/Muster einen *NULL*-Wert, verknüpft diesen mit dem vorausgegangenen *MATCH*-Befehl und übernimmt beide zusammen in die Ergebnismenge. Diese Art der Auswertung entspricht dem SQL-*OUTER JOIN*, den man von einer relationalen Datenbank kennt. Für einen *OPTIONAL MATCH*-Befehl stehen alle Pattern/Muster-Ausdrücke wie in einem gewöhnlichen *MATCH*-Befehl zur Verfügung. Wie das Wort *OPTIONAL* bereits semantisch ausdrückt, ist die Erfüllung des *MATCH*-Pattern/Muster nicht zwingend notwendig – es handelt sich vielmehr um eine *KANN*-Bedingung. Findet Neo4j kein Element im Graphen, welches das Pattern/Muster erfüllt, so greift es auf einen Platzhalter, den *NULL*-Wert, zurück. Anschließend führt Neo4j alle nachfolgenden Operationen des CQL-Statements mit diesem Platzhalter durch.

Bei der Problemstellung: »Ermittle eine Liste aller Mitarbeiter mit ihren Vorgesetzten« muss man berücksichtigen, dass es Mitarbeiter im Unternehmen gibt, die keinen Vorgesetzten haben. In der gewünschten Liste steht dann beim betroffenen Mitarbeiter ein *NULL*-Wert als Vorgesetzter. Anhand der bisherigen Graph-Datenbank erstellt man diese Liste mittels einer *OPTIONAL MATCH*-Abfrage (Bild 7):

```
MATCH (a) OPTIONAL MATCH (a)-[:Vorgesetzter]->(b) RETURN
a.Name, b.Name AS Vorgesetzter;
```

In einer bestehenden Graph-Datenbank können die Werte der Properties (Eigenschaften/Attribute) für alle vorhandenen Knoten (Objekte) und Kanten (Beziehungen) geändert oder ihnen neue Properties hinzugefügt werden. Vor der Durchführung einer gewünschten Änderung bestimmt man mittels eines *MATCH*-Befehls die betroffenen Elemente in der Graph-Datenbank. Der *MATCH*-Befehl muss so formuliert sein, dass er tatsächlich nur die zu ändernden Elemente der Graph-Datenbank bestimmt. Schließlich verwendet die auf den *MATCH*-Befehl folgende *SET*-Klausel sämtliche, das heißt alle Elemente der Treffermenge, um die gewünschte Änderung durchzuführen. Insofern sollte man den *MATCH*-Befehl vor Ausführung mit *SET* immer sorgfältig prüfen und ihn idealerweise vorher testen.

Die Verknüpfung des *MATCH*-Befehls mit der *SET*-Klausel für die Durchführung von Änderungen besitzt zwei wesentliche Vorteile. Mit dieser Konvention beschreibt man in Cypher nicht nur deklarativ, für welche Elemente des Graphen eine Änderung vorzunehmen ist, sondern eröffnet gleichzeitig der Update-Operation eine Arbeitsweise mit einem hohen Grad an Automatisierung. Die Änderung einer *SET*-Klausel erfolgt nämlich für alle durch den vorausgehenden *MATCH*-Befehl gefundenen Elemente der Graph-Datenbank. Damit erweitert man entscheidend die Mächtigkeit der Update-Operationen und auch die Einsatzbreite der *SET*-Klausel auf viele in der Praxis anzutreffende Fallbeispiele.

Wie das nachfolgende Beispiel zeigt, kann eine derartige Update-Operation sogar alle Knoten eines bestimmten ▶

Kompatibilitätslisten für einfachere Migrationspfade

Wie jede Sprache unterliegt auch Cypher Änderungen und Weiterentwicklungen, die man durch Versionsnummern unterscheidet.

Jede Version der Graph-Datenbank Neo4j unterstützt eine bestimmte Anzahl verschiedener Sprachversionen von Cypher: So unterstützt Neo4j Version 2.3 die Cypher-Versionen: 1.9, 2.2 und 2.3 – diese zusammengenommen nennt man Kompatibilitätsliste.

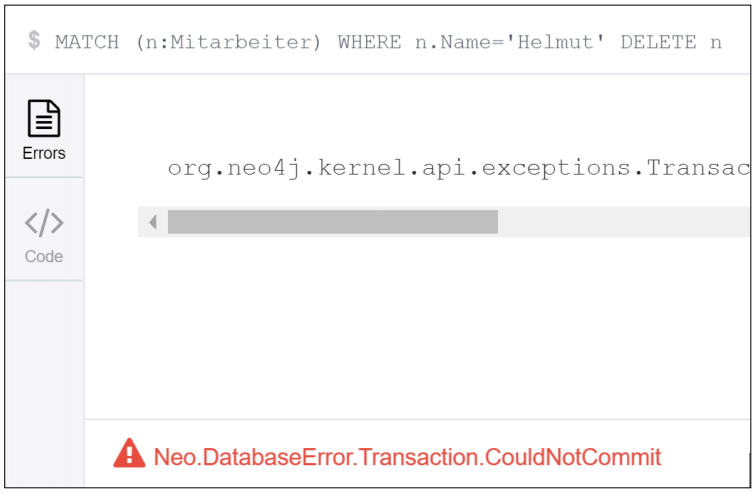
Diese Unterstützung mehrerer Sprachversionen im DBMS erleichtert die Migration von einer Neo4j-Version auf eine neuere, da man vorhandene Anwendungen schrittweise umstellen kann.

Zusätzlich kündigt Neo Technology rechtzeitig an, ab welcher Sprachversion bestimmte Cypher-Features deprecated (ungültig) werden.

Bei der Umstellung auf eine neue Neo4j-Version muss man also prüfen, ob die Kompatibilitätsliste die älteste in den Anwendungen verwendete Cypher-Version enthält.

Datentyps (Labels) im Graphen verarbeiten: Das Gehalt sämtlicher Mitarbeiter der bestehenden Graph-Datenbank soll um 5,7 Prozent auf die nächste ganze Zahl erhöht werden. Für diese Aktualisierung des Gehalts reicht das nachfolgende, recht einfache CQL-Statement aus:

```
MATCH (a:Mitarbeiter)
SET a.Gehalt = ROUND (a.Gehalt * 1.057)
RETURN a.Name, a.Gehalt;
// Die Funktion ROUND rundet auf die nächste ganze Zahl
```



Cypher gewährleistet über das Transaktions-Management von Neo4j automatisch die Integrität einer Graph-Datenbank (Bild 9)

Die *SET*-Klausel eignet sich auch vorzüglich zum Hinzufügen neuer Properties sowohl für Knoten als auch Kanten. Gleichzeitig kann man die Werte für diese neuen Properties initialisieren. Insbesondere erlaubt die *SET*-Klausel die Durchführung mehrerer hintereinander aufgeführter Operationen mit Knoten/Kanten, ihren Properties und Werten.

Eine Überprüfung, ob das *MATCH*-Statement tatsächlich alle betroffenen Knoten/Kanten verarbeitet hat, lässt sich über die Rückmeldung der Datenbank im Neo4j-Browser vornehmen (Bild 8).

Das Unternehmen im Fallbeispiel plant, erstmals Leiharbeiter an verschiedenen neuen Standorten einzusetzen und beides in der Datenbank ersichtlich zu machen. Man entscheidet, eine neue Property *Anstellung* mit den Werten *Fest* und *Leih* sowie ein weiteres Attribut *Ort* für den Standort des jeweiligen Mitarbeiters einzuführen. Alle bisherigen Mitarbeiter des Unternehmens sind fest angestellt und arbeiten in der Unternehmenszentrale in Frankfurt. Das nachfolgende Cypher-Statement aktualisiert die bereits vorhandene Graph-Datenbank komplett in einem einzigen Schritt:

```
MATCH (x:Mitarbeiter)
SET x.Anstellung = 'Fest', x.Ort = 'Frankfurt';
```

Eine Neo4j-Datenbank entspricht einem (im Bedarfsfall auch gerichteten) Property-Graphen, der vier grundlegende Elemente umfasst: Knoten (Entitäten), Beziehungen (Kanten), Attribute/Properties (Schlüssel-Wert-Paare) und Label. Cypher teilt diese vier Bestandteile der Graph-Datenbank über zwei Befehlsklauseln zum Löschen/Entfernen in zwei Gruppen ein:

- **REMOVE:** Entfernt Label und Properties von Entitäten (Knoten) oder Properties von Beziehungen (Kanten).
- **DELETE:** Löscht Knoten (Entitäten) oder Kanten (Beziehungen) aus der Datenbank.

Die englischen Bezeichnungen dieser beiden Befehlsklauseln verdeutlichen klar ihre Auswirkung auf die Graph-Datenbank. Während *DELETE* einem tatsächlichen Löschen des Objekts aus der Datenbank entspricht, entfernt *REMOVE* lediglich bestimmte Informationen aus dem Objekt, ohne das zugehörige Datenbankelement selbst zu löschen. Die *DELETE*-Klausel entfernt eine Entität oder Beziehung, auch dann, wenn diese selbst noch Label oder Properties besitzt; diese werden automatisch ebenfalls gelöscht. Allerdings lässt sich ein Knoten mittels *DELETE* nur löschen, wenn er in keiner Beziehung mehr über eine Kante eingebunden ist.

Beziehungsintegrität

Beim Verletzen dieser referenziellen Integritätsbedingung (Beziehungsintegrität) kommt es in Neo4j zu einer *DatabaseError*-Exception der zugehörigen Transaktion (Bild 9). Dieses Verhalten basiert auf der Konvention, dass der Neo4j-Browser jedes CQL-Statement innerhalb einer Transaktion ausführt. Generell gelten in Neo4j sogenannte Delete Semantics, dabei handelt es sich um Regeln, die sicherstellen sollen, die Integrität der Graph-Datenbank programmtechnisch über die Neo4j-Schnittstellen einhalten zu können. Das Handling zur Überprüfung der Datenbankintegrität erfolgt im Kontext des Transaktions-Managements. Nur diese Realisierung erlaubt beispielsweise die programmtechnische Verarbeitung eines

Behandlung des NULL-Werts in Cypher

Im Unterschied zu einer relationalen Datenbank kennt Neo4j den NULL-Wert nicht als Datentyp, unterstützt ihn aber bei der Verarbeitung von Cypher-Abfragen.

Eine Cypher-Abfrage kann als Ergebnis einen *NULL*-Wert erzeugen und auch zurückliefern.

Greift man innerhalb eines Cypher-Statements auf einen *NULL*-Wert zu, so erhält man als Rückgabewert wiederum den *NULL*-Wert.

Im Unterschied zu objektorientierten Programmiersprachen, die bei einer Referenzierung eines *NULL*-Werts eine Exception werfen, arbeitet Neo4j bei der Auswertung einer Cypher-Abfrage eigenständig mit dem *NULL*-Wert (aufgrund der zuvor genannten Regel) weiter.

bereits gelöschten Knotens oder einer gelöschten Beziehung innerhalb einer laufenden Transaktion.

Exceptions innerhalb einer Transaktion

Um Exceptions innerhalb einer Transaktion zu vermeiden, sollte man vor dem Löschen eines bestimmten Knotens alle zu ihm gehörenden Relationen löschen. Man erreicht dies recht einfach durch Lesen des Knotens mittels *MATCH*-Befehl, nachfolgendes Bestimmen aller eventuell zum Knoten gehörenden Beziehungen und abschließendes Löschen dieser Beziehungen mitsamt gewünschtem Knoten:

```
MATCH (x:<zu_loeschenden_Knoten_bestimmen>)
OPTIONAL MATCH (x) - [r] - ()
DELETE r, x
// OPTIONAL MATCH falls es Knoten ohne Beziehungen gibt
```

Als Besonderheit besitzen beide Klauseln, sowohl *DELETE* als auch *REMOTE,* ein sogenanntes idempotentes Verhalten: Nach mehrfacher Ausführung bleibt ihr Ergebnis gleich – es kommt seitens Cypher zu keinerlei Fehlermeldung, auch wenn das zu löschende/entfernende Element im Graph nicht (oder nicht mehr) existiert. Dieses Verhalten entspricht der bei einer parallelen oder verteilten Verarbeitung benötigten Fehlertoleranz, wie sie für Web/Internet- und Mobile-Anwendungen erforderlich ist. Diese Fehlertoleranz ermöglicht einer NoSQL-Datenbank, und zu dieser Klasse gehört Neo4j, besseren Durchsatz und höhere Verfügbarkeit. Gleichzeitig sinkt die Zahl der seitens der Programmierer zu behandelnden Fehlerfälle, die eigentlich gar keine Fehler darstellen.

Während der Entwicklung und Programmierung von Anwendungen führt man häufig Tests im Rahmen der Qualitätssicherung durch. Für diese Zwecke richtet man sich eine Testumgebung ein, die einer Produktionsumgebung möglichst weitgehend entspricht. Zusätzlich benötigt man für die Testdurchführung insbesondere für automatisierte Tests eine einheitliche Ausgangsbasis. Im Fall einer Datenbank muss man für automatisierte Tests immer dieselben Daten bereitstellen. In der Regel erreicht man dies über im Hintergrund laufende Prozesse, die anfangs alle in der Datenbank vorhandenen Daten löschen und anschließend die benötigten Daten für die ▶

Indexstrukturen für schnellere Zugriffe auf Daten

Indexe (Indizes) stellen eine von den eigentlichen Daten der Datenbank getrennte Datenstruktur dar.

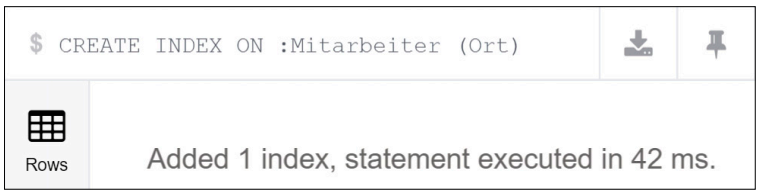
Mit der Einrichtung eines Index verfolgt man das Ziel, die Suche oder Sortierung der Daten einer Datenbank schneller durchführen zu können.

Ein Index beschleunigt das Zugriffsverhalten des DBMS erheblich gegenüber einer sequenziellen Suche.

Zudem erlaubt ein Index, ganze Teilmengen der Elemente einer Datenbank zu lokalisieren, die ein vorgegebenes Suchkriterium erfüllen.

Testdurchführung wieder bereitstellen. Um alle Daten in einer Graph-Datenbank zu löschen, eignet sich folgendes einfache Cypher-Statement:

```
MATCH (n)
OPTIONAL MATCH (n) - [r] - ()
DELETE n, r
```



Neo4j unterstützt die Neuanlage einer Indexstruktur für bereits in der Graph-Datenbank vorhandene Label-Attribut-Paare (Bild 10)

Seit Version 2 kennt Neo4j Index-Strukturen und Constraints (Integritätsbedingungen). Während die Query-Engine Indizes für ein schnelleres Durchsuchen des Graphen nutzt, dienen Constraints dem Schutz des Graphen vor fehlerhaften Daten. Bei einem Constraint handelt es sich um eine Integritätsbedingung, deren Einhaltung vor dem Hinzufügen oder Ändern eines Elements in der Datenbank überprüft wird. Im Unterschied zum Constraint, der nur sehr wenig zusätzlichen Speicherplatz in Anspruch nimmt, benötigt ein Index erheblich mehr Platz für seine eigenen Speicherstrukturen.

Einfügen und Ändern von Elementen

Zusätzlich beansprucht ein Index beim Einfügen und Ändern von Elementen wesentlich mehr Laufzeit, die eventuell vorzunehmende Änderungen in seiner Ablagestruktur verursachen. Allerdings wirkt sich die schnellere Performance beim Einsatz eines Index für Read/Retrieve-Operationen grundlegend positiv aus. Deshalb sollte man immer einen Index einsetzen, sobald seitens der Anwendung Engpässe beim Laufzeitverhalten zu erkennen sind. Sollte sich ein Index über die Lebenszeit einer Anwendung hinweg als nachteilig erweisen, kann er recht einfach wieder aus der Datenbank entfernt werden. Dieses Löschen erfolgt schnell, da lediglich der zugehörige Speicherbereich als ungültig zu erklären ist.

Indizes definiert man für Label-Attribut-Paare, daher heißen sie auch Label-Indizes; andersartige Indizes unterstützt Cypher derzeit noch nicht. Einen Index erstellt man mit dem Befehl *CREATE INDEX ON :<label_name> (<property_name>);* der Index selbst basiert weder auf dem Namen des Labels noch auf dem Attribut-Namen, sondern auf dem Wertebereich des zu beiden gehörenden Attributs (Property).

```
CREATE INDEX ON :Mitarbeiter (Ort)
```

Als Ergebnis dieses CQL-Befehls erhält man eine Erfolgsmeldung über die Anlage einer neuen Indexstruktur und die gesamte Ausführungszeit für deren Aufbau (Bild 10). Bei Neuanlage oder beim Ändern von Werten für das zugrunde liegende Label-Attribut-Paar übernimmt Neo4j automatisch die Pflege eines bereits vorhandenen Index. Zu viele Indizes beeinflussen die Performance der Datenbank in der Regel negativ. Indizes, die sich auf zu große Datenbestände beziehen und nur selten für Abfragen genutzt werden, sollten wieder gelöscht werden. Für diese Zwecke implementiert Cypher folgenden Befehl: *DROP INDEX ON :<label_name> (<property_name>)*.

Funktionen in Cypher

Neo Technology hat in Cypher eine Vielzahl von Funktionen implementiert, die immer ein Ergebnis zurückgeben.

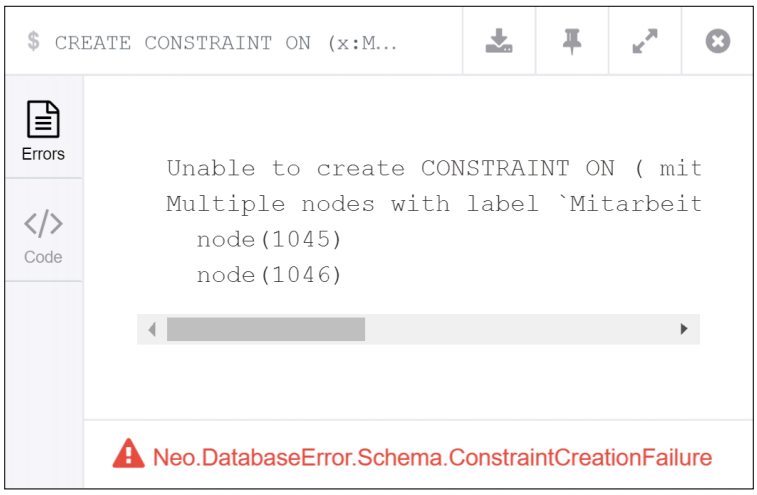
Sollte ein Eingabeparameter einen *NULL*-Wert annehmen, so liefern die meisten Funktionen auch *NULL* als Rückgabe zurück. Für einen besseren Überblick teilt man die Cypher-Funktionen in fünf verschiedene Gruppen oder Anwendungsgebiete ein:

- **Prädikate:** Prüfen aufgrund einer boolesche Funktion wie *ALL, ANY, NONE, SINGLE* oder *EXISTS* die Gültigkeit einer Aussage und liefern *true* oder *false*.
- **Skalare:** Geben, wie das Wort, ausdrückt einen eindimensionalen Wert (Zahl/Zeichenkette) zurück. Zu den Skalaren gehören: *SIZE, TYPE, ID, TIMESTAMP, TOINT, LENGTH* und weitere.
- **Collections/Sammlungen:** Als Rückgabe erhält man eine Collection zurück. Zu ihnen gehören: *NODES, RELATIONSHIPS, LABELS, KEYS, RANGE* und andere.
- **Mathematik:** Numerische, aus der Mathematik oder Physik bekannte Ausdrücke wie *ABS, ATAN, COS, DEGREES, EXP, LOG, ROUND, SIN, SQRT, TAN* und ähnliche.
- **Strings/Zeichenketten:** Sie verarbeiten analog den aus vielen Programmiersprachen bekannten Funktionen Zeichenketten (mit Ausnahme von *TOSTRING*) (*LEFT, RIGHT, TRIM, LOWER, UPPER* und andere).

Integrität der Daten – Datenkonsistenz (Fehlerfreiheit)

Datenkonsistenz soll die Korrektheit der gespeicherten Daten im Sinne einer widerspruchsfreien und vollständigen Abbildung der Realität gewährleisten. Eine Datenbank kennt verschiedene Integritätsebenen oder -stufen.

- **Wertebereichsintegrität:** Prüft, ob die Werte eines in die Datenbank einzufügenden Elements korrekt sind. Man spricht auch von der Integrität auf der Datenfeldebene.
- **Datenintegrität:** Stellt sicher, dass beim Hinzufügen, Ändern oder Löschen von Elementen aus der Datenbank keine Zustände in der Datenbank auftreten, die der Realität der Anwendungswelt widersprechen.
- **Referenzielle Integrität:** Verhindert das Auftreten von Inkonsistenzen für alle in der Datenbank vorhandenen Beziehungen; man nennt dies auch Beziehungsintegrität.



Unique Constraints von Cypher gewährleisten immer eindeutige Werte für das zugrunde liegende Label-Attribut-Paar (Bild 11)

Neo4j kennt bisher nur Unique Constraints für Label-Attribut-Paare, dabei handelt es sich um Integritätsbedingungen, die Eindeutigkeit in den Datenbeständen gewährleisten. Parallel mit der Anlage eines Constraints erzeugt Neo4j im Hintergrund automatisch einen Index für die Werte des eindeutig zu haltenden Attributs. Im Bedarfsfall nutzt Neo4j für Queries selbstständig derart angelegte Indexstrukturen. Einen Constraint legt man mit dem Cypher-Befehl *CREATE CONSTRAINT ON (alias:<label_name>) ASSERT alias.<property_name> IS UNIQUE* an.

Bei bereits vorhandenen Datenbeständen müssen die Werte des zugehörigen Label-Attribut-Paars eindeutig sein, ansonsten kommt es zu einer Fehlermeldung (Bild 11) und die Eindeutigkeit muss durch Bereinigen der Datenbestände manuell hergestellt werden. Auch darf noch kein Index für das Label-Attribut-Paar des Constraints vorhanden sein. Ein eventuell vorhandener Index muss vor Anlage des Constraints gelöscht werden. Analog zum Löschen von Indizes erfolgt das Löschen eines Constraints mit dem CQL-Befehl: *DROP CONSTRAINT ON (alias:<label_name>) ASSERT alias.<property_name> IS UNIQUE*. Diese Anweisung löscht den Constraint und gleichzeitig automatisch den mit ihm verbundenen Index.

Anders als CQL-Schlüsselwörter unterscheiden reguläre Ausdrücke zwischen Groß- und Kleinbuchstaben. Soll dieses Verhalten in einem Cypher-Statement keine Rolle spielen, also im regulären Ausdruck die Groß- oder Kleinschreibung des Zeichenmusters nicht relevant sein, so erreicht man das durch *(?i)* als Zeichenfolge. Stehen diese vier Zeichen *(?i)* vor einem regulären Ausdruck, so besitzt dessen Schreibweise – ob klein oder groß – keine Bedeutung mehr. Ferner verfügt CQL noch über zusätzliche Zeichenfolgen, um die Verarbeitung des regulären Ausdrucks zu beeinflussen.

Als weitere Notation für die Mengenorientierung steht in der *WHERE*-Klausel das *IN*-Prädikat zur Verfügung; es befindet sich vor der Angabe einer Menge oder einer Liste. Das *IN*-Prädikat erweitert die Semantik der *WHERE*-Bedingung und erleichtert wesentlich das Lesen und Formulieren der gesamten CQL-Anweisung. Generell prüft das *IN*-Prädikat in einer CQL-Abfrage, ob der auf die *WHERE*-Bedingung folgende Parameter als Element in der nach *IN* stehenden Menge/Liste enthalten ist. Liefert das *IN*-Prädikat *true* zurück, so ist der zugehörige Teil der *WHERE*-Bedingung erfüllt:

```
MATCH (x:Mitarbeiter)
WHERE x.Anstellung IN ['Fest', 'Frei']
RETURN x.Name, x.Anstellung;
```

Neo4j kennt als Datentypen für die Definition von Properties (Attribute/Eigenschaften) nur boolesche Werte, gängige Zahlentypen, Zeichen und Felder. Die Definition eigener Datentypen in Anlehnung an den SQL-Standard relationaler DBMS unterstützt weder Neo4j noch Cypher. Zusätzlich verfügt die 4GL-Sprache Cypher jedoch über Maps und Collections als weitere Sprachkonstrukte. Bei beiden handelt es sich in Anlehnung an die Programmiersprache Java um Behälter, also Ansammlungen von Objekten/Elementen. Anders als Java unterstützt Cypher nur literale Maps, die man innerhalb der 4GL als Ausdrücke verwenden kann. Für den Einsatz von Collections bietet Cypher ein wesentlich breiteres Einsatzspektrum mit weiteren Features an.

Neo4j Version 2 führte beide Sprachkonstrukte, sowohl Maps als auch Collections, in Cypher ein. Maps kommen vor allem innerhalb eines Cypher-Statements wie *CREATE* zum Einsatz. Literal Maps eigen sich insbesondere als Parameter für die Programmierung über das REST-Java-Interface oder Import-Tool. So initialisiert beispielsweise ein *CREATE-* ▶

Neo4j-Schnittstellen für den Einsatz von Cypher

Neo4j besitzt für den direkten Einsatz von Cypher verschiedene Schnittstellen.

Neo4j-Shell: versteht CQL-Befehle und führt sie direkt aus; des Weiteren stehen Befehle zur Verfügung, um Informationen über die Datenbank zu erhalten oder durch den Graphen zu navigieren.

- **Neo4j-Browser:** Auch Neo4j-Webinterface genannt, stellt das Werkzeug für den Einstieg in Neo4j dar. Über einen gängigen Browser kann man die Datenbank per CQL abfragen, ihre Struktur und Elemente verändern. Der Browser kennt verschiedene Anzeigemodi wie die Visualisierung der Elemente einer Datenbank als Graph oder in Zeilenform.
- **Import-Tool:** Ein Neo4j-Werkzeug, um Massendaten zu importieren oder zu verarbeiten.
- **REST-API:** Diese Schnittstelle erlaubt es, über HTTP und JSON mit einer Neo4j-Datenbank zu kommunizieren, um CQL-Befehle auszuführen.
- **Native Java-API:** Neo4j besitzt ein eigenes Java-API; über diese Schnittstelle kann man Cypher-Abfragen als String-Parameter an Neo4j übergeben.

Befehl über eine Map als Key-Value-Paar die Eigenschaften einer Kante oder eines Knotens und fügt so der Graph-Datenbank ein neues Element hinzu:

```
CREATE    (u:Mitarbeiter {Name: "Georg", Nummer: 4711,
Ort: "Ulm"}),
(u)-[:Vorgesetzter]->(:Mitarbeiter {Name: "Peter", Ort:
"Augsburg"}),
(u)-[:Vorgesetzter]->(:Mitarbeiter {Name: "Jörg", Ort:
"Berlin"});
```

Über den Key erreicht man innerhalb einer Map gültige Wert-Einträge, während ein Zugriff auf einen nicht existierenden Key eine Fehlermeldung erzeugt. Um mehrere Elemente mit einem CQL-Befehl anzulegen, übergibt man ein Array (Feld) oder eine Collection von Maps an das *CREATE*-Statement.

Das Schachteln von Literal Maps und Collections ist erlaubt, dadurch entstehen Nested Maps/Collections. Für Collections kann man Prädikate wie *ALL*, *ANY*, *NONE* oder *SINGLE* anwenden, die abhängig davon, ob sie erfüllt sind oder nicht, den entsprechenden booleschen Wert zurückliefern:

- **ALL:** Liefert *true*, wenn das Prädikat für alle Einträge der Collection erfüllt ist.
- **ANY:** Liefert *true*, sobald ein Element der Collection das Prädikat erfüllt.
- **NONE:** Erfüllt kein Element der Collection das Prädikat, so liefert es *true* zurück.
- **SINGLE:** Liefert *true*, wenn genau ein und kein weiteres Element der Collection das Prädikat erfüllt.

Besonders produktiv arbeitet man beim Einsatz von Collections mit der *FOREACH*-Klausel und dem *UNWIND*-Befehl. Während *FOREACH* bereits mit Neo4j Version 1.8 eingeführt wurde, existiert der *UNWIND*-Befehl erst seit Version 2.1.1. Die allgemeine Syntax der *FOREACH*-Klausel lautet: *FOREACH (variable IN collection | befehl);* dabei reicht die vor *IN* stehende Variable ihre Werte an den durch | verknüpften Befehl weiter. Somit iteriert *FOREACH* mittels dieser Variable über sämtliche Elemente der durch *IN* übergebenen Collection und führt anschließend mit jedem Element der Collection den auf | folgenden Befehl aus:

```
FOREACH (wert IN ['Renate', 'Herbert', 'Susanne'] |
CREATE (:Mitarbeiter {Name: wert}))
```

Der *UNWIND*-Befehl wandelt die Elemente einer beliebigen Collection in einzelne Zeilen um. Eine derartige Umwandlung einer Collection in einzelne Zeilen ist besonders für zeilen- oder satzorientierte Verarbeitung nützlich. Der *UNWIND*-Befehl übergibt die Elemente einer Collection quasi einzeln an die nach *AS* stehende Variable und reicht deren Wert anschließend sequenziell an die nachfolgende Befehlsfolge weiter:

```
UNWIND ['Augsburg', 'Ulm', 'Berlin'] AS ort
RETURN ort
```

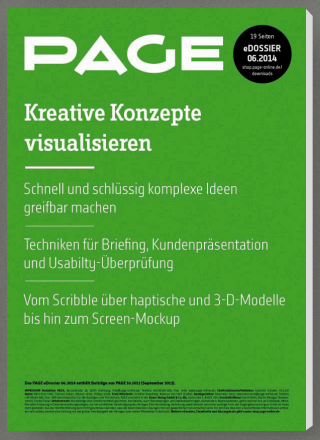
Ergänzend gibt es für Collections noch spezielle Funktionen wie die Länge einer Collection *(LENGTH)*, das erste *()* oder das letzte Element *(LAST)*, einen Filter *(FILTER)*, der über ein Prädikat eine Vorauswahl für bestimmte Elemente einer Collection trifft, *KEYS*, die zu einem Element alle Properties ermitteln, *NODE* bestimmt alle Knoten eines übergebenen Pfades, *RELATIONSHIPS* alle Beziehungen innerhalb eines Pfades oder *LABELS* alle Label eines Knotens. ■

Links zum Thema

- Homepage der Graph-Datenbank Neo4j
  www.neo4j.com
- Download-Seite von Neo4j
  http://neo4j.com/download
- Webseite mit den Neo4j Release Notes
  http://neo4j.com/release-notes
- Homepage der Neo4j-Dokumentation
  http://neo4j.com/docs
- Homepage der aktuellen Cypher Refcard
  http://neo4j.com/docs/stable/cypher-refcard
- Homepage des Open-Source-Projekts openCypher
  www.opencypher.org
- Website zum Arbeiten mit Neo4j und CQL über eine Neo4j-Console
  http://console.neo4j.org
- Überblick zu den primitiven Datentypen von Java
  https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html
- Zusammenfassung der regulären Ausdrücke für Java
  http://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html
- Website mit vielen Informationen und Tools rund um reguläre Ausdrücke, auch in Verbindung mit Programmiersprachen
  www.regular-expressions.info
- Website zum Prüfen regulärer Ausdrücke für verschiedene Programmiersprachen
  www.regexplanet.com
- Anwendungsbeispiele für reguläre Ausdrücke
  www.regexplanet.com/cookbook

Frank Simon
arbeitet in der Software-Entwicklung mit den aktuellen Arbeitsgebieten Entwicklung, Programmierung, Test und Debugging von Cloud-, Rich-Internet-, Mobile- und Web-Anwendungen inklusive deren System-Management.
web_mobile_developers@gmx.eu

Downloaden, aufschlauen!



PAGE eDossiers – Best-of-Kompilationen aus PAGE und WEAVE im Originallayout: PDFs einfach und jederzeit runterladen in unserem Online-Shop **shop.page-online.de/downloads**

PAGE
Das Magazin der Kreativbranche

Ein Magazin der Verlagsgruppe Ebner Ulm • www.ebnerverlag.de

DEUTSCHE BAHN OPEN DATA

Daten von der Bahn

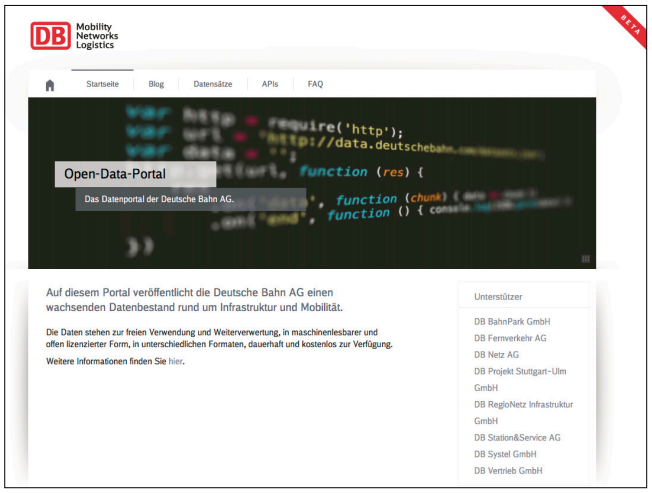
Vorbei sind die Zeiten, in denen man an die Daten der Deutschen Bahn nur per Hacks oder eigener Erfassung gekommen ist.

Seit Ende 2015 betreibt die Bahn eine Open-Data-Schnittstelle, über die per API beziehungsweise über Datensätze auf zahlreiche Daten ganz offiziell zugegriffen werden kann (Bild 1). Momentan noch als Beta gekennzeichnet, stellt die Open-Data-Schnittstelle der Bahn bereits jetzt zahlreiche Datensätze als XML/JSON-basierte REST-Schnittstelle einerseits und als Download (JSON, Excel oder XML) andererseits zur Verfügung. Geplant ist zusätzlich die stetige Erweiterung der Daten – so wurden am 25. Februar 2016 beispielsweise die Daten des Fernverkehrs integriert.

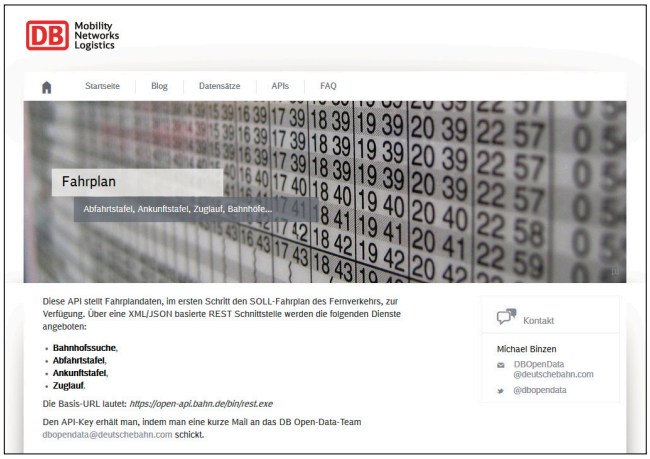
Stand heute gibt es bereits die folgenden Dienste innerhalb des API: Der Dienst Fahrplan bietet eine Bahnhofssuche, eine Abfahrts- und eine Ankunftstafel sowie Zugläufe (Bild 2). Über den Dienst Parkplätze kann man Bahnhöfe, Parkräume sowie Belegungsdaten ausgewählter Parkräume abrufen. Die Daten dieser APIs werden unter der Lizenz Creative Commons Attribution 4.0 International (CC BY 4.0) bereitgestellt. Tabelle 1 zeigt, welche Datensätze zurzeit zur Verfügung stehen. Die Datensätze werden auf der Seite http://data.deutschebahn.com/datasets/ referenziert. So kann man beispielsweise eine Liste aller Haltestellen über den Link

```
http://data.deutschebahn.com/datasets/haltestellen/
D_Bahnhof_2016_01_alle.csv
```

herunterladen und bekommt die in Listing 1 gezeigten Daten geliefert. In den veröffentlichten Daten sind zu jeder Haltestelle beispielsweise die in Tabelle 2 dargestellten Informationen enthalten.



DB Mobility Networks Logistics stellt ein Open Data Portal zur Verfügung (Bild 1)



Der Dienst Fahrplan bietet Entwicklern Informationen zu Zugläufen (Bild 2)

Aufbau des Fahrplan API

Das API wird durch einen Aufruf eines URL mit dem folgenden Aufbau angesprochen:

```
https://open-api.bahn.de/bin/rest.exe/<<Version>>/
<<Dienst>>?<<Parameter>>
```

Dabei ist die Version zurzeit immer 1.0. Parameter gibt es je nach Dienst unterschiedliche. Man benötigt aber in fast allen Fällen einen Parameter *authKey*, den man derzeit per formloser E-Mail an dbopendata@deutschebahn.com anfordern kann. Das Format wählt man über den Parameter *format*. Per Default wird hier XML ausgegeben; wählt man als Wert *json*, so wird auf das JSON-Format umgestellt. Dabei wird der JSON-Content durch Umwandlung des XML-Outputs per Regel generiert.

Bei der Umwandlung werden Element-Namen zu Objekt-Eigenschaften und Text (PCDATA) wird zu Objekt-Eigenschaften mit *$*, also beispielsweise wird *<a>foo* zu *{„a“:{„$“:“foo“}}*. Verschachtelte Elemente werden zu verschachtelten Eigenschaften. Zum Beispiel wird *<a>foo<c>foo</c>* zu *{„a“:{„b“:{„$“:“foo“},“c“:{„$“:“foo“}}}*. Gibt es mehrere Elemente mit demselben Namen, erzeugt JSON hierfür ein Array. Aus *<a>foo1foo2* wird also *{„a“:{„b“:[{„$“:foo1“}*.

Das REST-API unterstützt zudem verschiedene Sprachen – diese können über den Parameter *lang=<code>* selektiert werden. Default ist hierbei immer Englisch *(en)*. Weitere Sprachen sind Deutsch *(de)*, Dänisch *(da)*, Spanisch *(es)*, Französisch *(fr)*, Italienisch *(it)*, Holländisch *(nl)* und Polnisch *(pl)*. Jeder Service wird zudem von einer XSD (XML Schema Definition) definiert, die auch ohne Authentifizierung mit

```
http://open-api.bahn.de/bin/rest.exe/v1.0/
xsd?name=hafasRestLocation.xsd
```

angerufen werden kann. Listing 2 zeigt ein Beispiel einer solchen XML Schema Definition. Die Bahnhofssuche liefert zum angefragten Suchstring

```
https://open-api.bahn.de/bin/rest.exe/v1.0/
location.name?authKey=<<AuthKey>>&lang=de&format=
json&input=München+Hbf
```

(anzugeben mit dem Parameter *input*) eine Liste von Bahnhöfen mit ihren jeweiligen API-spezifischen Bahnhofs-IDs. Anstelle von *<<AuthKey>>* tragen Sie den individuellen Key ein (Listing 3). Hier erhält man als Antwort sowohl den Namen als auch die Geodaten des Bahnhofs. Ebenfalls mitgeliefert wird die ID, die in späteren Abfragen als Referenz benötigt wird.

Die Abfahrtstafel (Service-Name: *departureBoard)* liefert zu einer gegebenen Bahnhofs-ID (anzugeben mit *id)* und optional zu Datum *(date)* sowie Uhrzeit *(time)* die planmäßig nächsten Abfahrten:

```
https://open-api.bahn.de/bin/rest.exe/departureBoard?
authKey=<<AuthKey>>&lang=de&id=008000261&date=2016-04-
04&time=09%3a00&format=json
```

Im Beispiel in Listing 4 wird der Bahnhof München am 4. April 2016 ab 9:00 Uhr angefragt. Hier erhält man nun die Abfahrtsdaten zurück, die zum Beispiel den Namen des Zuges, Zugtyp, Zeit, Datum und den Zielbahnhof enthalten. Als *JourneyDetailRef* wird ein Link zurückgegeben, der über das API die Reisedaten des Zugs zurückgibt. ▶

Tabelle 1: Datensätze

Datensätze	Details
Aufzugsdaten	Equipment, Ort, Wirtschaftseinheit, Hersteller, Baujahr, Antriebsart, Kabinenhöhe, Förderhöhe, Tragkraft, Koordinaten
Bahnsteigdaten	Bahnhofsnummer, Bahnsteignummer, Nr. der Bahnsteigkante, Gleis, Bahnsteiglänge, Höhe Bahnsteigkante
Bahnsteigdaten (RNI)	Bahnhofsnummer, Bahnsteignummer, Bahnsteiglänge, Höhe Bahnsteigkante
Betriebsstellen-verzeichnis	Bahnhöfe, Anschluss-, Ausweichanschluss-, Abzweig-, Überleitstellen, Haltepunkte, Blockstellen, Streckenwechsel
Haltestellen	EVA-Nr, DS 100, Name Haltestelle, Koordinaten
Netzradar	LTE, Mobilfunk, Konnektivität, GeoJSON, Mobilfunkanbieter, O2, E-Plus, Vodafone, T-Mobile
Reisezentren	Reisezentrum, Öffnungszeiten, Verkaufsstellen-nummer, Koordinaten
Serviceeinrich-tungen	Bundesland, Regionalbereich, Betriebsstelle, Gleis-Nr, Funktionskategorie, Weiche, Gleislänge, Oberleitungslänge, Nutzlänge, DS100
Stationsdaten	Adresse, Bahnhof, Station, DS 100, Kategorie, Aufgabenträger, Nahverkehr, Fernverkehr
Stationsdaten (RNI)	Bahnhof, Station, Kategorie, Aufgabenträger, Nahverkehr, Fernverkehr
Streckennetz	Streckennetz, Geodaten, INSPIRE, Bauwerke, Tunnel, Bahnübergänge, Betriebsstellen, Bahnhöfe, Haltestellen
Stuttgart 21 (Geodaten)	Gleistrassen, Umrisse, Webcam-Standorte, Gleisanlagen, Stuttgart 21

Listing 1: Download-Liste der Haltestellen

```
EVA_NR;DS100;NAME;VERKEHR;LAENGE;BREITE;;
8000001;KA;Aachen Hbf;FV;6.091499;50.7678;;
8070704;KASZ;Aachen Schanz;RV;6.07384;50.769862;;
...
8004128;MMDN;München Donnersbergerbrücke;
RV;11.536537;48.142623;;
8004168;MFHM;München Flughafen Terminal;
RV;11.78597;48.353728;;
8004129;MHAB;München Hackerbrücke;
RV;11.548543;48.141915;;
8000261;MH;München Hbf;FV;11.558335;48.140232;;
8098263;MHT;München Hbf
(tief);RV;11.560493;48.141172;;
...
```

Tabelle 2: Informationen zu Haltestellen

Begriff	Erläuterung
EVA_NR	Nummer der Haltestelle
DS100	Verweis auf Betriebsstelle
NAME	Name der Haltestelle
VERKEHR	Kann folgende Werte annehmen FV (mit Fernverkehr), RV (nur Regionalverkehr) oder nur DPN (nur Regionalverkehr von privaten Eisenbahnunterneh-men).
LAENGE	Longitude der Haltestelle in WGS84
BREITE	Latitude der Haltestelle in WGS84

Die Ankunftstafel (Service-Name: *arrivalBoard)* liefert zu einer gegebenen Bahnhof-ID und optional zu Datum sowie Uhrzeit die planmäßig nächsten Ankünfte:

```
https://open-api.bahn.de/bin/rest.exe/arrivalBoard?
authKey=<<AuthKey>>&lang=de&id=008000261&date=2016-04-
04&time=09%3a00&format=json
```

Im Beispiel in Listing 5 wird der Bahnhof München am 4. April 2016 ab 9:00 Uhr angefragt.

Listing 2: Aufruf einer Schema-Datei

```
<!--  ========================================== -->
<!--  Version 1.0.0                 -->
<!--  ========================================== -->
<!--  XSLT   -->
<xs:schema xmlns:xs="http://www.w3.org/2001/
XMLSchema" elementFormDefault="qualified">
<xs:element name="LocationList">
<xs:annotation>
<xs:documentation>
The location list contains either named coordinates
or stops/stations with name and id as a result of a
location request. The data of every list entry can
be used for further trip or departureBoard requests.
</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:choice minOccurs="0" maxOccurs="unbounded">
...
```

Listing 3: location.name

```
{
"LocationList":{
  "noNamespaceSchemaLocation":"http://open-api.bahn.
  de/bin/rest.exe/v1.0/xsd?name=hafasRestLocation.
  xsd",
  "StopLocation":[{
    "name":"München Hbf",
    "lon":"11.558338",
    "lat":"48.140228",
    "id":"008000261"
    },{
    ...
    "name":"Warnemünde",
    "lon":"12.091273",
    "lat":"54.176853",
    "id":"008013236"
    }]
  }
}
```

Die Reiseinfos – offiziell im Bahnjargon Zuglauf genannt – (Service-Name: *journeyDetail)* liefern den Reiseverlauf eines Zuges (Bild 2). Sie werden jeweils beim Request von *departureBoard* und *arrivalBoard* mit angegeben:

```
http://open-api.bahn.de/bin/rest.exe/v1.0/journeyDetail?
ref=991341%2F334059%2F976116%2F157611%2F80%3Fdate%
3D2016-04-04%26station_evaId%3D8000261%26station_type%
3Ddep%26authKey%3D<<AuthKey>>%26lang%3Dde%26format%3
Djson%26
```

Im Beispiel wird der Bahnhof München (ID aus der vorherigen Abfrage) am 4. April 2016 ab 9:00 Uhr angefragt.

Parkplätze

Auch Informationen über Park & Ride-Plätze stehen zum Abruf zur Verfügung. Parkplätze werden über die DB BahnPark-OpenData-Schnittstelle für Parkrauminformationen angezeigt. Hierfür kann man sich zunächst die Liste der Bahnhöfe mit Parkplatzmöglichkeiten anzeigen lassen (Service: *stations)* (Listing 6).

Die Parkräume selbst kann man über den Service *sites* ermitteln lassen. Für bestimmte Parkräume kann sogar die aktuelle Belegung angezeigt werden (Service: *occupancy)* – dafür wird die ID des Parkraums angegeben, die bei der vorherigen Abfrage als *parkraumid* übermittelt worden ist.

Listing 4: departureBoard

```
{
„DepartureBoard“:{
  „noNamespaceSchemaLocation“:“http://open-api.bahn.
  de/bin/rest.exe/v1.0/xsd?name=
  hafasRestDepartureBoard.xsd“,
  „Departure“:[{
    „name“:“ICE 630“,
    „type“:“ICE“,
    „stopid“:“8000261“,
    „stop“:“München Hbf“,
    „time“:“09:05“,
    „date“:“2016-04-04“,
    „direction“:“Bremen Hbf“,
    „track“:“14“,
    „JourneyDetailRef“:{
      „ref“:“http://open-api.bahn.de/bin/rest.exe/
      v1.0/journeyDetail?ref=659778%2F223538%2F35676
      2%2F41545%2F80%3Fdate%3D2016-04-04%26station_
      evaId%3D8000261%26station_type%3Ddep%26authKey
      %3D<<AuthKey>>%26lang%3Dde%26format%3Djson%26“
      }
    },
    ...
    ]
  }
}
```

Listing 5: arrivalBoard

```
"ArrivalBoard":{
"noNamespaceSchemaLocation":"http://open-api.bahn.
de/bin/rest.exe/v1.0/xsd?name=hafasRestArrivalBoard.
xsd",
  "Arrival":[{
    "name":"ICE 521",
    "type":"ICE",
    "stopid":"8000261",
    "stop":"München Hbf",
    "time":"09:04",
    "date":"2016-04-04",
    "origin":"Köln Hbf",
    "track":"23",
    "JourneyDetailRef":{
      "ref":"http://open-api.bahn.de/bin/rest.exe/
      v1.0/journeyDetail?ref=958983%2F322758%2F73933
      6%2F50007%2F80%3Fdate%3D2016-04-04%26station_
      evaId%3D8000261%26station_type%3Darr%26authKey
      %3DDBhackFrankfurt0316%26lang%3Dde%26format%3D
      json%26"
      }
    },
    ...
    ]
  } }
```

Dabei hat die Zahl im Feld *category* folgende Belegung:

- *1* bedeutet 0 – 10 freie Stellplätze
- *2* bedeutet 10 – 30 freie Stellplätze
- *3* bedeutet 30 – 50 freie Stellplätze
- *4* bedeutet mehr als 50 freie Stellplätze

Alle diese Belegungen bekommt man mit dem generischen Aufruf http://opendata.dbbahnpark.info/api/beta/occupancy. Selbstverständlich gibt es mittlerweile auch eine Reihe von Clients, um das API direkt abzufragen (siehe Kasten).

Fazit

Die Bahn veröffentlicht seit November 2015 schrittweise Daten – und ist hiermit noch lange nicht fertig. Dabei ist ihr auch der Austausch mit Entwicklern wichtig – auch durch Wettbewerbe und Hackathons. Feedback kann so jederzeit entweder in den API-Dokumenten oder per E-Mail unter dbopendata@deutschebahn.com eingereicht werden. ■

Patrick Lobacher
ist Digital Native, Entwickler, Berater, Coach und Autor zahlreicher Fachbücher und Fachartikel. Er ist Vorstandsvorsitzender der Pluswerk AG, die digitale Kommunikationslösungen konzipiert, umsetzt und betreut.

Listing 6: stations

```
{
  "type" : "result",
  "version" : 1,
  "totalCount" : 145,
  "count" : 145,
  "offset" : 0,
  "limit" : 0,
  "results" : [ {
    "type" : "stationCms",
    "bahnhofsNummer" : 4,
    "cityTitle" : "Aalen",
    "evaNummer" : 8000002,
    "isDbBahnpark" : true,
    "isPublished" : true,
    "mainPicId" : "308",
    "mainPicId_en" : "8018",
     "station" : "Aalen"
  },
  ...
  ]
 }
}
```

Links zum Thema

- Deutsche Bahn Open Data Portal
  http://data.deutschebahn.com
- Fahrplan-API-Dokumentation
  http://data.deutschebahn.com/apis/fahrplan/Fpl-API-Doku-Open-Data-BETA-0_81_2.pdf
- Parkplätze-API-Dokumentation
  http://data.deutschebahn.com/apis/parkplatz/opendata_dbbahnpark_08.pdf
- Timetable im GTFS-Format
  https://github.com/fredlockheed/db-fv-gtfs
- Twitter-Account des Projekts
  https://twitter.com/dbopendata
- JavaScript-Client
  https://github.com/pbock/fahrplan
- Java-Clients
  https://github.com/highsource/db-fahrplan-api
  https://github.com/marcusschiesser/openbahn-api
- Ruby-Client
  https://github.com/gasman/bahn
- ReactJS-Client
  https://github.com/brakmic/OpenLok

MICROSOFT ASP.NET 5 UND DOCKER

Docker mit Microsoft Azure

Wie Sie die Docker-Erweiterungen von Visual Studio nutzen, um in Azure eine ASP.NET-5-Anwendung zu installieren und zu hosten.

Das Open-Source-Tool Docker ist eine Container-Technologie, die eine leichtgewichtige virtuelle Maschine darstellt und auf Prozesslevel-Ebene arbeitet. Hierbei vereinfacht es auch das Management von Linux-Containern.

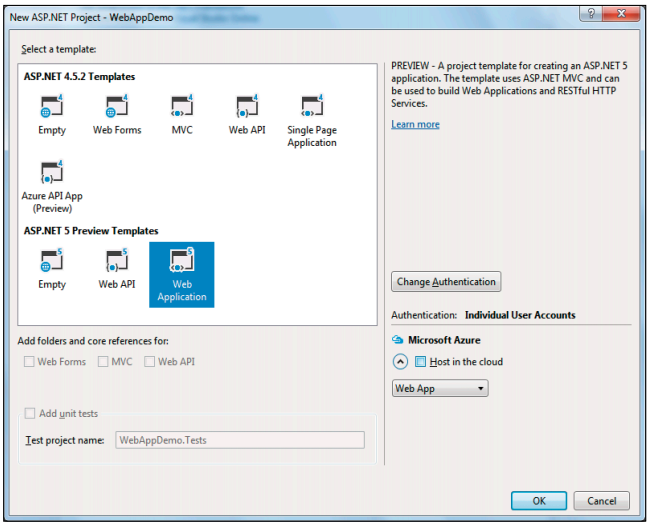
Das heißt, Docker ermöglicht es, Anwendungen in Containern auszuführen. Diese Container können aufeinander aufbauen und miteinander kommunizieren. Docker verpackt Anwendungen mit ihren Abhängigkeiten zu einer kompakten Einheit und stellt so eine Anwendungsvirtualisierung dar. Docker nutzt spezielle Kernelfunktionalitäten, um einzelne Prozesse in Containern voneinander zu isolieren. Für Prozesse, die mit Containern gestartet werden, scheint es so, als würden sie auf einem eigenen System laufen.

Docker stellt aber keine virtuelle Maschine dar, ist also definitiv nicht dasselbe wie Virtualisierung, sondern setzt auf einen Linux-Container (LXC) auf und erweitert diesen mit einer umfangreichen API für das Containermanagement.

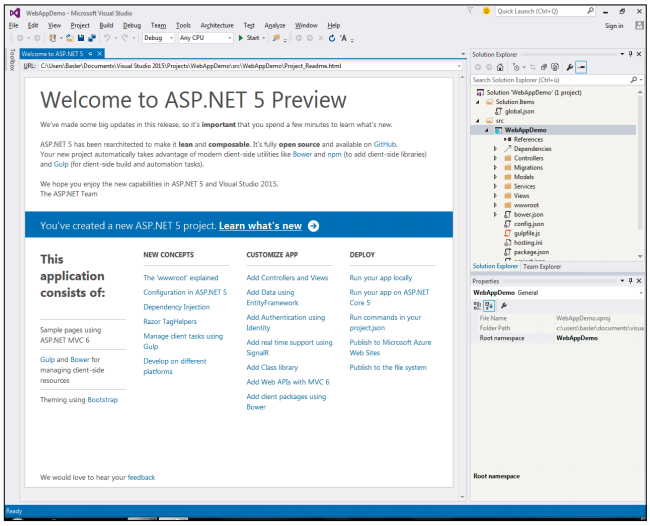
Dei Technik bietet somit die Möglichkeit, Software innerhalb und für Software-Container auf einer Server-Virtualisierung zu entwickeln, wobei Docker-Container völlig unabhängig vom Betriebssystem oder Infrastruktur arbeiten.

Unterschied zu Virtualisierung

Der große Unterschied zu einer virtuellen Maschine liegt darin, dass nicht jeder Container ein komplettes Betriebssystem bereitstellen muss, sondern nur die wichtigsten Daten für die Applikation enthält.



Auswahl des Projekts Web Application (Bild 1)



Beispiel: Die Beispielanwendung WebAppDemo (Bild 2)

Somit fungiert Docker als Container für die Anwendung und löst damit auch ein wenig die Zusammenstellung der Open-Source-Installation von Apache, Datenbank, Perl oder PHP, kurz XAMPP, in der Nutzung ab. XAMPP kann bei der heutigen Vielzahl von Entwicklungsumgebungen und Versionen einfach nicht mehr überall mithalten.

Docker kann mit Hilfe von VirtualBox auch auf OS X und Windows verwendet werden. Dies ist für die Entwicklung von Webanwendungen besonders nützlich. So kann bei der Webentwicklung die Serverumgebung, auf der später die Webapplikation laufen wird, mit Hilfe eines Containers genau auf dem eigenen Entwicklungsrechner nachgebildet werden.

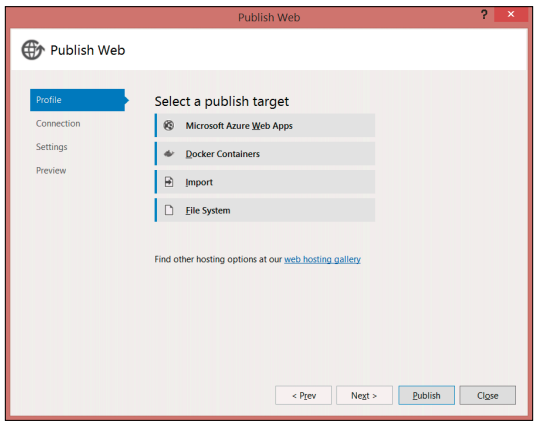
Das Verteilen der Docker-Container findet über einen sogenannten Docker-Hub statt. Aus dem fertig erstellten und für die Applikation konfigurierten Container wird dann ein Image des Containers in den zentralen Docker-Hub hochgeladen. Dort ist es dann zugänglich und kann benutzt werden. Docker-Container sind also konfigurierbare, abgeschlossene Einheiten, in denen Anwendungen ausgeführt werden können.

Funktionsweise

Wie schon beschrieben, stellt das Image ein Template für die Erstellung eines Containers da. Somit ist der Container eine Instanz des Images, in dem sich die Spezifikation befindet. Ein vorhandenes oder erstelltes Image kann jederzeit modifiziert und als neues Image gespeichert werden.

Hierfür können Sie die gewünschten Anpassungen in der Konfiguration in der Datei *Dockerfile* vornehmen oder bei einem gestarteten Container über eine Shell die Modifikationen durchführen und den Container als neues Image abspeichern.

Um Docker dann zu verwenden, muss der Docker-Daemon gestartet werden; dieser kennt den Docker-Hub und wird verwendet, um die lokalen Docker-Images und Container zu verwalten. Möchten Sie Docker unter Windows nutzen, müssen Sie einen kleinen Umweg in Kauf nehmen. Es gibt noch keine offizielle native Windows-Version für und mit Docker. Somit gibt es für Sie drei Alternativen für den Einsatz von Docker auf Windows.

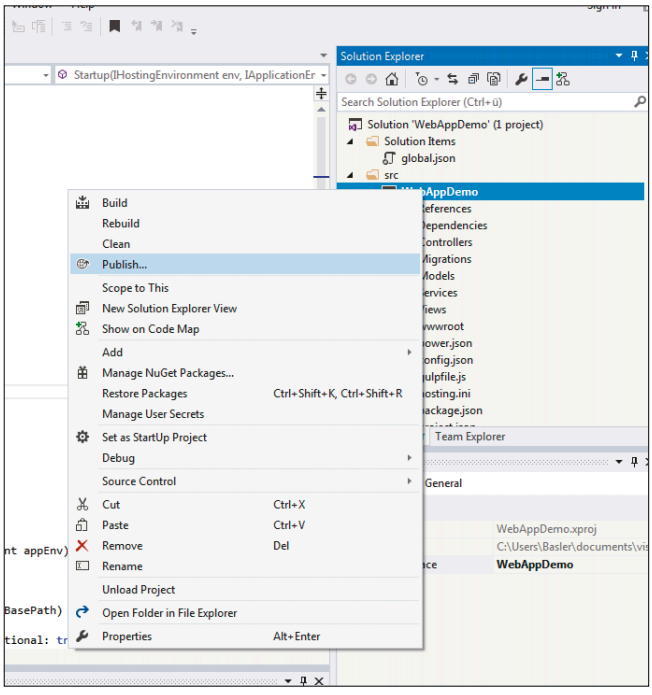


Docker-Container: Den Container auswählen (Bild 4)

Der erste Weg ist die Installation von Linux in einer virtuellen Maschine und der Installation von Docker. Hierbei wird die Docker-Maschine benutzt, um eine virtuelle Maschine zu starten, in der dann der Docker Daemon läuft.

Eine weitere Vorgehensweise bietet die Installation des Projekts Boot2Docker. Mit dessen Hilfe wird auf dem eigenen Windows eine virtuelle Maschine mit einer Basis-Docker-Tiny-Linuxversion installiert. Boot2Docker wird inklusive aller benötigten Abhängigkeiten heruntergeladen und installiert. Dabei wird sowohl Docker als auch VirtualBox zusammen mit GIT und einem SSH-Client eingerichtet.

Der einfachste Weg, eine Docker-Testumgebung einzurichten, ist die Verwendung einer virtuellen Maschine in der Cloud von Microsoft Azure.



Publish: Die Veröffentlichung aufrufen (Bild 3)

Microsoft Azure stellt seit Kurzem eine fertige VM-Extension für Docker als VM-Vorlage zur Verfügung. Diese eignet sich sehr gut für das erste Kennenlernen von Docker im Azure-Portal. Über das Shell-Kommando steht Ihnen hier das für Docker benötigte Scripting zur Verfügung. Eine ausführliche Anleitung mit den Titel »Using the Docker VM Extension from the Azure Command-line Interface« von Ralph Squillace finden Sie unter dem Download-Link https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-docker-with-xplat-cli.

Ein weiterer Vorteil bei dieser Vorgehensweise ist, dass die benötigten Zertifikate für HTTPS von Azure selbst erstellt und in das Verzeichnis *~/.docker* abgelegt werden.

Weitere Informationen zur Verwendung der Docker VM-Erweiterung für Linux in Azure finden Sie im Tutorial von Alp Balkan unter https://blogs.msdn.microsoft.com/webdev/2015/01/14/running-asp-net-5-applications-in-linux-containers-with-docker. Hier erfahren Sie, wie sie ein Container-Image erzeugen, anpassen und starten.

Systemanforderung

Sie benötigen für diesen Workshop Visual Studio 2015 mit Update 1 mit den dazugehörigen Visual Studio 2015 Tools für die Docker-Erweiterung (DockerVSTools.VS140). Den entsprechenden Download finden Sie unter https://visualstudiogallery.msdn.microsoft.com/0f5b2caa-ea00-41c8-b8a2-058c7da0b3e4.

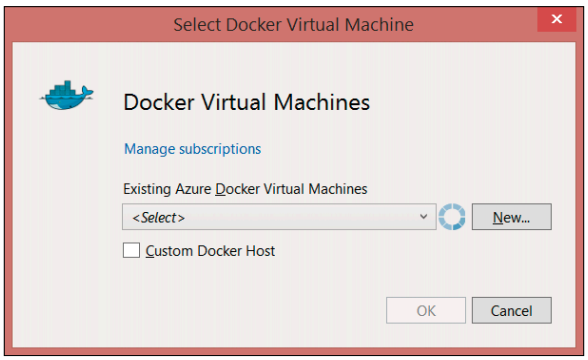
Weiterhin ist auch die Docker-Toolbox sehr hilfreich. Sie stellt eine Virtuelle Maschine und einen entsprechenden Docker-Client für Windows zur Verfügung (https://www.docker.com/products/docker-toolbox).

In dem Beispiel soll eine ASP.NET-5-Webanwendung in Docker gehostet werden. Hierfür benutzen Sie einfach die Microsoft-ASP.NET-Webvorlage. Sie können aber auch alternativ eine vorhandene ASP.NET-Applikation verwenden.

Bevor Sie mit Visual Studio starten, installieren Sie die Docker-Tools für Visual Studio 2015. Beachten Sie, dass es sich hierbei um ein Pre-Release handelt. Folgen Sie zur Installation einfach den Anweisungen des Assistenten. Ist die Installation abgeschlossen, so starten Sie Visual Studio 2015 und erstellen ein Projekt vom Typ *ASP.NET Web Application*.

Die Projektvorlage hierfür finden Sie unter *Start* und *New Project*. Unterhalb des Visual-C#-Knotens steht die Projektvorlage *ASP.NET Web Application* zur Verfügung. Wählen Sie die Projektvorlage aus und vergeben Sie als Vorbereitung für unsere Beispielanwendung den Namen *WebAppDemo*. Schließen Sie das Dialogfenster *New Project* über *Ok* ab. Daraufhin öffnet sich das ASP.NET-Projekttyp-Dialogfenster zur näheren Spezifikation der zu erstellenden ASP.NET-Webapplikation. ▶

Wählen Sie dort für das Beispiel die Vorlage *Web Application* aus den ASP.NET 5 Preview Templates aus und deaktivieren Sie das Häkchen bei *Host in the cloud* (Bild 1). Visual Studio erstellt anschließend automatisch aus dem Template eine bereits lauffähige Webapplikation auf Basis des MVC-6-Entwurfsmusters. Bild 2 zeigt die erstellte Webanwendung mit der entsprechenden Projektstruktur. Über die Taste [F5] oder das Startsymbol lässt sich die Webseite ausführen. Visual Studio startet hierfür im Hintergrund automatisch einen Webserver (IIS Express).



Anlegen einer Docker Virtual Machine (Bild 5)

Veröffentlichung

Nachdem Sie eine ASP.NET-5-Webanwendung erstellt haben, können Sie die Anwendung veröffentlichen. Wählen Sie hierfür im Kontextmenü Ihres ASP.NET-Projekts den Punkt *Publish* aus (Bild 3). Es öffnet sich der Publish Web Assistent. Im Dialogfenster klicken Sie auf den Eintrag *Docker Containers* (Bild 4). Über das Dialogfenster *Select Docker Virtual Machine* wählen Sie den benötigen Docker-Host aus. Sie können hier auch einen neuen Docker-Host erstellen oder einen vorhandenen virtuellen Computer auswählen, der in Azure oder an einer anderen Stelle gehostet wird.

Erstellen Sie für das Beispiel einen neuen Docker-Host (Bild 5). Klicken Sie auf *New…* und folgen Sie den Anweisungen des Assistenten. Schließen Sie das Dialogfenster über *Ok* ab. Danach öffnet sich das Dialogfenster *Create virtual machine on Microsoft Azure*. Tragen Sie hier die benötigten Informationen ein. Legen Sie einen eindeutigen DNS-Namen fest und wählen Sie als Server-Image für das Beispiel *Ubuntu Server 14.04 LTS* aus. Vergeben Sie einen Benutzernamen und ein Kennwort. Wählen Sie für die Location die Einstellung für Ihr Gebietsschema. Zum Schluss legen Sie noch den benötigten Docker-Server-Port fest.

Durch das Häkchen *Auto-generate Docker certificates* werden die erforderlichen Docker-Zertifikate installiert. Nach dem Klick auf *Ok* erhalten Sie eine Meldung, dass die Erstellung des virtuellen Computers beginnt. Visual Studio erstellt jetzt automatisch eine Azure-Ressourcen-Manager-Vorlagendatei (ARM), eine Parameterdatei und ein Power-Shell-Skript, damit die Befehle später erneut ausgeführt werden können.

Wurde der Docker-Host vollständig in Azure bereitgestellt, können Sie die Webapplikation jetzt über die Registerkarte *Connection* veröffentlichen. Geben Sie hier den Server-URL mit der entsprechenden Portnummer und den Image-Namen an und veröffentlichen Sie die Anwendung über den Button *Publish*. Das Beispielprojekt *WebAppDemo* wird daraufhin veröffentlicht und im Docker-Host zur Verfügung gestellt. Über die Eingabeaufforderung können Sie mit dem Befehl

```
Set DOCKER_HOST=tcp://<NameAzureVM>.cloudapp.net:2280
Set DOCKER_TLS_VERIFY=1
```

jederzeit prüfen, ob der Docker-Host vorhanden ist und funktioniert. Eine Liste der verfügbaren Docker-Befehle erhalten Sie einfach über den Befehl *docker*. Über Docker lassen sich somit Images in einem zentralen Repository für Kollegen oder sogar öffentlich über den Docker Hub anbieten. Dies erlaubt eine einfache und effektive Wiederverwendung von Docker-Images und macht das Entwickeln von Anwendungen einfacher, da alle Beteiligten immer den gleichen Entwicklungsstand benutzen beziehungsweise auf diesem testen können.

Fazit

Docker bietet noch viel mehr Funktionen zum Erstellen und Pflegen von Anwendungsvirtualisierung. Leider steht Docker im Moment noch nicht für Windows zur Verfügung. Auch die Docker-Erweiterungen für Visual Studio sind noch in der Beta-Phase. Es zwickt und zwackt noch an der ein oder anderen Stelle, zeigt aber jetzt schon die vielfältigen Möglichkeiten von Docker. ■

Links zum Thema

- Using the Docker VM Extension from the Azure Command-line Interface (Azure CLI)
  https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-docker-with-xplat-cli
- Docker-Erweiterung für virtuelle Linux-Computer in Azure
  https://blogs.msdn.microsoft.com/webdev/2015/01/14/running-asp-net-5-applications-in-linux-containers-with-docker
- Docker-Erweiterung für Visual Studio 2015
  https://visualstudiogallery.msdn.microsoft.com/0f5b2caa-ea00-41c8-b8a2-058c7da0b3e4
- Docker-Toolbox für Windows
  https://www.docker.com/products/docker-toolbox

Daniel Basler
ist Senior Consultant für Microsoft-Technologien und beschäftigt sich darüber hinaus mit Datenbanken und Compiler-Bau.

Der ultimative Leitfaden zur Entwicklung des Cross-Border Verkaufs in Europa

So skalieren Sie Ihre internationale Entwicklung und erzielen einen schnellen ROI

GRATIS Whitepaper

Lesen Sie im kostenlosen Whitepaper von Lengow

- Wichtige Fakten, Zahlen und Trends im E-Commerce
- Der Aufbau einer soliden internationalen Strategie
- Bestimmung der Zielgruppe
- Gesetzliche Vorschriften
- Versand- und Rückgaberegelung
- Zahlungsabwicklung
- Zollgebühren und Steuern
- Leitfaden für einzelne Länder
- Die Skalierung der Internationalisierung
- Skalierung der Reichweite durch verschiedene Absatzwege
- Vorstellung von Textmaster
- Vorstellung von Lengow
- Die Partnerschaft Lengow-Textmaster

Jetzt kostenlos downloaden:
http://digital.internetworld.de/whitepaper-cross-border-wump/

Ein Service von:
INTERNET WORLD Business
Ihr Whitepaper bei uns?
Anfragen an sales@nmg.de
oder Tel. 089 / 7 41 17 – 124.

DROPBOX-STEUERUNG MIT PHP UND SHELL-SKRIPTS

PHP auf Wolke sieben

Dropbox ist ein beliebter Online-Speicher mit einer guten Software-Unterstützung.

Das Ablegen von Dateien in der Cloud kann auf viele Arten nützlich sein. Es erlaubt das gemeinsame Speichern von Dokumenten bei verteilt arbeitenden Teams, bietet einen sicheren Datenspeicher und vieles mehr. Der beliebte Dienst Dropbox bietet für alle gängigen Plattformen ausgereifte Clients an, die sich um die Synchronisation der Daten zwischen dem Dienst und lokalen Verzeichnissen kümmern. Oft ist aber auch die Integration in Webanwendungen gefragt.

Zu diesem Zweck bietet Dropbox ein API und liefert auch gleich eine passende Client-Library für alle namhaften Programmiersprachen. Darüber können Sie alle wichtigen Funktionen des Dienstes steuern. Zu den unterstützten Sprachen gehört auch PHP. Es gibt in GitHub zwar noch weitere Client-Bibliotheken, die sich desselben Themas annehmen, aber die Variante von Dropbox selbst ist die ausgereifteste. Die aktuelle Version 2 des API wird von PHP zwar noch nicht unterstützt, aber das ist keine große Einschränkung, da schon die Funktionen der ersten Version sehr mächtig sind (Bild 1).

Die Library können Sie auf zwei Arten nutzen. Bei einer Variante laden Sie sich das ZIP-File aus der Online-Doku herunter und kopieren den Inhalt des Ordners *lib* in Ihr Projekt. Dann aktivieren Sie den enthaltenen Autoloader:

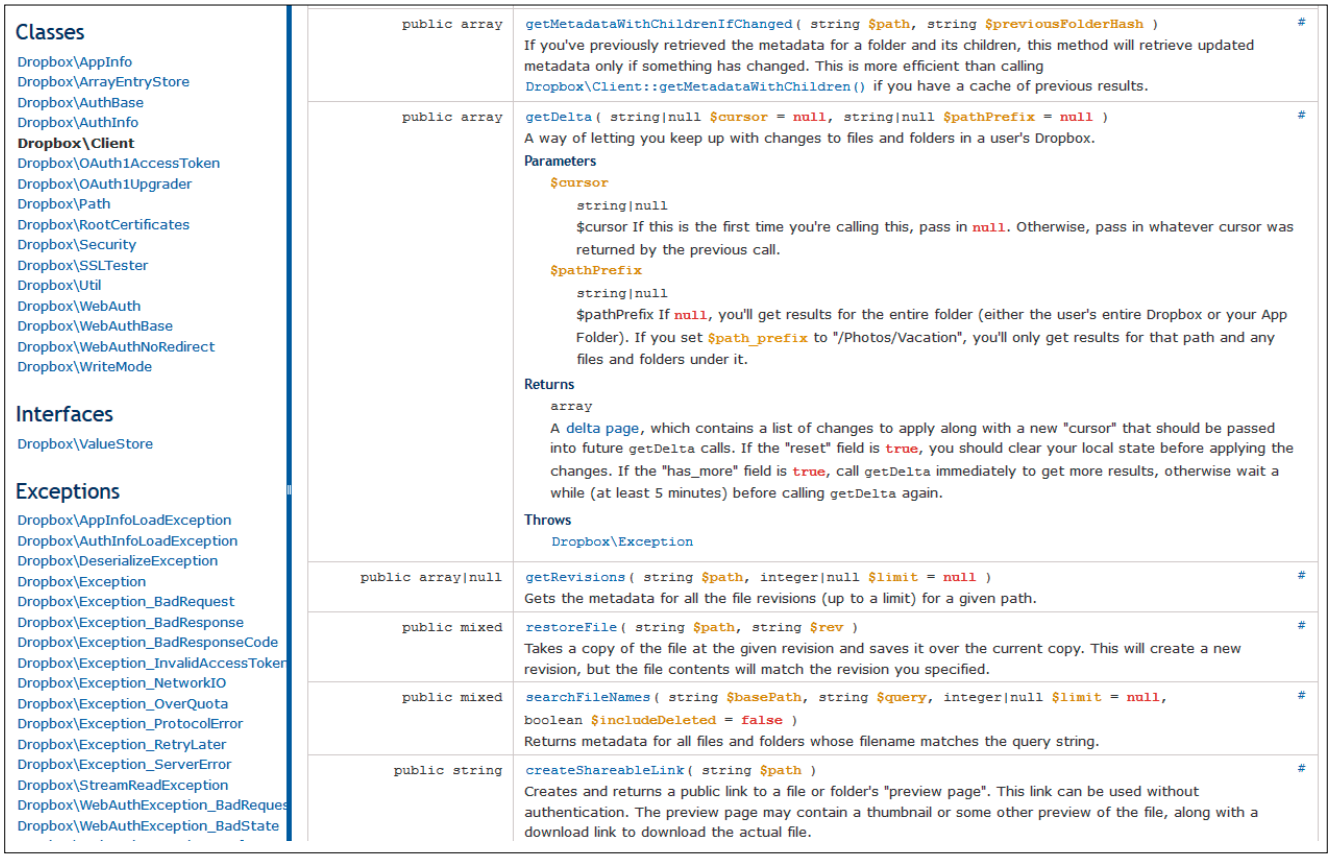
```
require_once "lib/Dropbox/autoload.php";
```

Falls Sie bereits Composer nutzen, um die Abhängigkeiten in Ihrem Projekt zu verwalten, dann haben Sie es leichter: Integrieren Sie die Bibliothek einfach über Composer:

```
composer require dropbox/dropbox-sdk
```

Der zugehörige Autoloader aktiviert dann die Dropbox-Bibliothek gleich mit.

Die ausführliche Methode für die Generierung einer Zugangsberechtigung hat Ihre Berechtigung. Wenn Sie aber direkt und schnell loslegen wollen, gibt es einen einfacheren



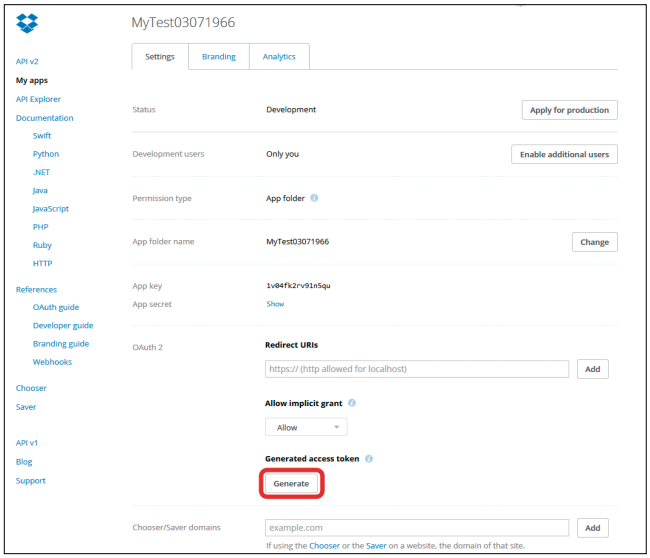
Während die allgemeine Dokumentation generell die Möglichkeiten des Zugriffs auf das Dropbox-API zeigt, bietet die Klassenreferenz für PHP konkretere Hilfe für Webprogrammierer (Bild 1)

Weg, Ihren PHP-Skripts den Zugang zur eigenen Dropbox zu erlauben. Dazu rufen Sie im Browser den URL https://www.dropbox.com/developers/apps auf und klicken auf den Button *Create app*.

Wählen Sie dann das Dropbox-API als Ziel und entscheiden Sie sich für den Gültigkeitsbereich der Erlaubnis: Mit *App Folder* erhält die Anwendung als Wirkungskreis lediglich ein Unterverzeichnis unterhalb des Ordners */Apps*, während ihr mit *Full Dropbox* die gesamte Ordnerstruktur Ihrer Dropbox offensteht.

Zum Schluss müssen Sie der Anwendung noch einen Namen geben. Da diese Namen nicht nur für Ihr Konto, sondern in einem globalen Raum eindeutig sein müssen, verwenden Sie am besten ein aussagekräftiges Präfix plus eine zufällige Kombination, wie *MyPhpTest47114321*, damit nicht der Fehler erscheint: *Name wurde bereits vergeben*.

Auf der nächsten Seite klicken Sie auf den Button unterhalb von *Generate Access Token* und kopieren den angezeigten Schlüssel. Diesen können Sie dann in den PHP-Anwendungen direkt als einzig notwendige Zugangskennung verwenden (Bild 2).



Um zum Ausprobieren des Dropbox-API die aufwendige Authentifizierung zu umgehen, nutzen Sie in der Verwaltungsoberfläche den Button, der ad hoc einen Zugangstoken generiert (Bild 2)

Die ersten Schritte mit dem Client

Als erste Aktion lassen wir uns Informationen zu Ihrem Konto ausgeben und prüfen so, ob der Zugriff überhaupt klappt.

Listing 1: Informationen des Verzeichnisses

```
Array (
[hash] => b542b8d057ddb524a3542b5f862d6d29
[thumb_exists] =>
[bytes] => 0
[path] => /
[is_dir] => 1
[icon] => folder
[root] => app_folder
[contents] => Array (
  [0] => Array
    (
    [rev] => 1a468a1bef
    [thumb_exists] =>
    [path] => /test.txt
    [is_dir] =>
    [client_mtime] => Fri,...
    [icon] => page_white_text
    [bytes] => 12
    [modified] => Fri,...
    [size] => 12 bytes
    [root] => app_folder
    [mime_type] => text/plain
    [revision] => 26
    )
  )
  [size] => 0 bytes
)
```

Geben Sie im Beispiel der Variablen *$token* den Code, den Ihnen die Dropbox-Website im vorigen Schritt generiert hat.

```
<?php
include "vendor/autoload.php";
use \Dropbox as dbox;
$token = "<Ihr Token>";
$dboxClient = new dbox\Client($token, "PHP-Beispiel");
$info = $dboxClient->getAccountInfo();
print_r($info);
```

Sie sehen, dass die Nutzung keine großen Hürden beinhaltet: Das Objekt mit dem Dropbox-Client wird mit dem Token und einem beliebigen Bezeichner initialisiert. Das Objekt bietet dann eine Reihe von Methoden, wie eben *getAccountInfo()*, was dann ein Array mit Informationen wie Ihre E-Mail-Adresse oder das Speicherplatzkontingent zurückliefert:

```
Array (
  [display_name] => Markus Schraudolph
  [uid] => 4093xx
  [locale] => de
  [country] => DE
  [email] => markus@schraudolph.de
  [quota_info] => Array (
    [datastores] => 0
    [shared] => 577012506
    [quota] => 3489660928
    [normal] => 1817161844
  )
  ...
)
```

Wenn das Skript keinen Fehler wirft, wissen Sie schon einmal, dass der Zugang funktioniert. ▶

Eine erste wirklich nützliche Aktion wäre das Hochschieben einer existierenden Datei in die Cloud. Dazu nehmen wir an, es existiere im aktuellen Verzeichnis die Datei *test.txt*. Die soll nun ins oberste Verzeichnis der Dropbox kopiert werden. Für diese Operation verwenden Sie die Methode *uploadFile()*. Hier der dazu notwendige Code, wobei wir uns die ersten Zeilen der Initialisierung gespart haben:

```
$fh = fopen("test.txt", "rb");
$md = $dboxClient->uploadFile(
  "/test.txt",
  dbox\WriteMode::add(),
  $fh);
fclose($fh);
print_r($md);
```

Der erste Parameter ist dabei der gewünschte Name der Datei, hier inklusive Pfadpräfix. Danach kommt der Schreibmodus. Der gewählte Modus *add()* bedeutet: Wenn bereits eine Datei mit dem Namen existiert, dann wird die hochgeladene Datei als *test (1).txt* abgespeichert.

Es gibt die Alternative *force()*, die in einem solchen Fall die alte Datei einfach überschreibt. Der dritte Parameter ist die lokale Datei, die hochgeladen werden soll. Allerdings erwartet die Methode nicht etwa den lokalen Dateinamen, sondern einen geöffneten File Handle. Zurückgeliefert wird dann ein Array mit Metainformationen zur hochgeladenen Datei:

```
Array (
  [revision] => 11
  [bytes] => 132
  [thumb_exists] =>
  [rev] => b468a1bef
  [modified] => Thu, 31 Mar 2016 14:51:12
  [mime_type] => text/plain
  [path] => /test.txt
  [is_dir] =>
  [size] => 132 bytes
  [root] => app_folder
  [id] => id:kszC0Q6p1rAAAAAAAAAACA
  [client_mtime] => Thu, 31 Mar 2016 14:51:12
  [icon] => page_white_text
)
```

Falls die Datei wegen eines Namenskonfliktes beim Hochladen umbenannt worden ist, könnten Sie das über den zurückgelieferten Wert von *path* feststellen.

Der normale Upload-Mechanismus versucht, die Datei in einem Rutsch in die Cloud hochzuschieben. Für große Dateien gibt es mit *UploadFileChunked()* eine Alternative, die die lokale Datei in Häppchen von 4 MByte aufsplittet. Treten dann Probleme auf, wird automatisch der aktuelle Chunk nochmals übertragen. Ab einer Dateigröße von 150 MByte müssen Sie diese Variante wählen, da der normale Upload hier an seine Grenzen stößt.

Listing 2: Funktionsprinzip der Delta-Abfrage

```
<?php
session_start();
include "vendor/autoload.php";
use \Dropbox as dbox;
$token = "<ihr token>";
$dboxClient = new dbox\Client($token,
"PHP-Beispiel");
// vorherigen Cursor holen, falls er existiert
if (array_key_exists("cursor", $_SESSION)){
  $cursor =  $_SESSION['cursor'];
} else {
  $cursor = null;
};
// Delta abrufen
$delta = $dboxClient->getDelta($cursor);
if (count($delta["entries"])==0) echo "Keine
Änderungen<br>";
  foreach ($delta["entries"] as $entry) {
  list($path, $meta) = $entry;
  if ($meta === null) {
    echo "Entfernt: $path<br>";
  } else {
    echo "Neu/geändert: $path<br>";
  }
}
// aktuellen Cursor abspeichern
$_SESSION['cursor'] = $delta["cursor"];
?>
```

Dateien herunterladen

Um eine Datei von der Cloud zum lokalen System zu kopieren, verwenden Sie die Methode *getFile()*, die auch wieder mit einem File Handle arbeitet:

```
$f = fopen("test_copy.txt", "w+b");
$md = $dboxClient->getFile("/test.txt", $f);
fclose($f);
print_r($md);
```

Auch hier enthält das zurückgelieferte Array die Metadaten der Dropbox-Datei. Hat in der Zwischenzeit kein anderer Client die Datei *test.txt* geändert, entsprechen diese Metadaten exakt den Werten, die der Upload ergab.

Inhaltsverzeichnis abrufen

Um die Dateien in einem bestimmten Verzeichnis abzurufen, dient die Methode *getMetadataWithChildren()*:

```
$md = $dboxClient->getMetadataWithChildren("/");
print_r($md);
```

Das Ergebnis enthält die Informationen des genannten Verzeichnisses und seiner Kinder – also der darin enthaltenen Dateien – unter *contents* (Listing 1).

Signalisierung aus der Cloud

Wollen Sie in einer Anwendung erfahren, ob ein anderer Client Dateien verändert hat, können Sie – wie im Beitrag gezeigt – einen Startzustand abrufen und durch API-Aufrufe immer wieder prüfen, ob sich Änderungen ergeben haben.

Es geht aber auch ohne dieses ständige Polling, indem Sie den Spieß umdrehen und den Dropbox-Server Ihre Anwendung benachrichtigen lassen, sobald sich etwas tut.

Dazu gibt es sogenannte Webhooks. Das ist eine Webadresse, die Sie in der Dropbox-Verwaltungsoberfläche im Register *Settings* bei einer App eintragen. Sobald sich eine Änderung ergibt, ruft Dropbox dann die von Ihnen in diesem Webhook festgelegten Adresse auf.

Damit Sie dort nur einen gültigen und unter Ihrer Obhut stehenden URL eintragen können, wendet die Oberfläche von Dropbox bei der Eingabe dieser Webadresse eine einfache Prüfung an. Es wird über den *GET*-Parameter *challenge* ein zufälliger Code übertragen und erwartet, dass er als Antwort zurückkommt.

Sie schreiben also zuvor in der zugehörigen Skript-Datei einfach den PHP-Befehl *<?php echo $_GET['challenge']; ?>* und können nach der erfolgten Verifikation diesen Part einfach aus dem Skript wieder entfernen, weil es sich um eine einmalig durchzuführende Prüfung handelt.

Tritt eine Änderung in der Dropbox ein, ruft der Server übrigens diesen URL ohne weitere Parameter auf. Ihr Programm muss also selbst durch API-Aufrufe herausfinden, was sich genau getan hat.

In Anwendungsfällen, in denen ein Webhook das Problem nicht löst, gibt es eine weitere Möglichkeit, ohne dauerndes Polling ein Update der Dropbox-Inhalte feststellen zu können. Dazu bietet das API eine Funktion an, die Long-Polling unterstützt. Damit ist eine HTTP-Anfrage gemeint, bei der die Rückmeldung erst dann kommt, wenn es tatsächlich etwas zu übertragen gibt – hier also eine Änderung an den Dropbox-Inhalten. Zwischen Anfrage und Antwort bleibt die Verbindung also offen, es herrscht aber Funkstille.

Ein Beispiel zu Long-Polling mit PHP und der Dropbox finden Sie in den Links. In einem Webserver-Umfeld ist diese Methode allerdings nicht immer optimal. Denn für jeden Apache-Thread, der aktiv ist, werden beispielsweise mehr als 15 MByte an Speicher verwendet. Hat Ihre Anwendung dann Kontakt zu vielen verschiedenen Dropbox-Instanzen gleichzeitig, kann das schnell den Hauptspeicher des Servers sprengen.

Eine zentrale Idee bei Dropbox ist ja, dass gleichzeitig mehreren Geräte und auch verschiedene User Zugriff haben können. Da ist die Frage nach dem Abgleich der Daten natürlich elementar. Der offizielle Client löst das so, dass er durch Synchronisation immer dafür sorgt, dass alle Beteiligten den gleich Stand haben.

Ihre Anwendung kann am leichtesten über das Metadaten-Feld *rev* feststellen, ob sich eine Datei geändert hat. Das ist die eindeutige Kennung der aktuellen Dateiversion. Überschreiben Sie (oder ein anderer Client) die bisherige Dateiversion mit einem neuen Inhalt, dann ändert sich dieser Wert.

Indem Sie also *rev* abfragen, können Sie feststellen, ob das noch die Version ist, die Sie erwarten, und entsprechende Maßnahmen ergreifen.

Das API bietet Ihnen auch einen speziellen Schreibmodus, der die Versionsnummer berücksichtigt. Angenommen, Ihre Anwendung hat eine Datei zur Bearbeitung heruntergeladen und sich die Version in der Variablen *$rev* gemerkt. Nach dem Ändern wollen Sie die Datei in der Dropbox speichern, allerdings nur, wenn in der Zwischenzeit kein anderer Client daran etwas verändert hat. Dann verwenden Sie als Schreibmodus nicht *add()* wie im ersten Upload-Beispiel, sondern *update()*, der als Parameter die vorherige Version erwartet:

```
$md = $dboxClient->uploadFile(
  "/test.txt",
  dbox\WriteMode::update($rev),
  $fh);
```

Ist die Datei online noch in der alten, erwarteten Version, wird sie einfach ersetzt. Gab es dagegen eine unerwartete Änderung von anderer Seite, legt Dropbox die neue Datei *test (conflicted copy).txt* an.

Durch die Auswertung des zurückgelieferten Arrays, das im Wert von *path* diese Namensänderung reflektiert, können Sie in Ihrem Programm auf diesen Umstand reagieren. Das Feld *revision* sollten Sie für den Zweck des Versionsvergleichs übrigens nicht nutzen. Dieser einfache Zähler ist veraltet und kann in Zukunft entfallen.

Inhalt des Verzeichnisses

Auch auf Verzeichnisebene gibt es eine ähnliche Einrichtung: Der Wert *hash* im Ergebnis von *getMetadataWithChildren()* bietet einen Fingerabdruck über den Inhalt des Verzeichnisses. Ihre Anwendung kann diesen Hash speichern und später die spezielle Variante *getMetadataWithChildrenIfChanged()* damit als Parameter aufrufen:

```
list($changed, $newMd)=
  $dboxClient->
    getMetadataWithChildrenIfChanged(
    "/",
    $savedHash
  )
```

Hat sich keine einzige Datei im genannten Verzeichnis geändert, dann ist das Ergebnis von *$changed* ein *false* und *$newMd* beinhaltet ein leeres Array. Ansonsten erhalten Sie *true* und die Metadaten aller Dateien zurück – also auch der unveränderten. Der Aufruf der Methode *getMetadataWithChildrenIfChanged()* ist für viele Anwendungen sicher nützlich, hat aber auch klare Nachteile. So ist die Information ▶

Nützliches abseits der PHP-Welt

Über die CLI-Variante von PHP können Sie zwar die im Beitrag gezeigten Möglichkeiten auch aus einem Shellskript oder anderen Programmen heraus nutzen, es geht aber einfacher.

Mit dem Paket Dropbox-Uploader von Andrea Fabrizi stehen Ihnen – anders, als der Name suggeriert – alle grundlegenden Aktionen für die Dropbox per Kommandozeile zur Verfügung, also auch das Herunterladen oder Löschen. Sie können damit durch einen Befehlsaufruf schnell und unkompliziert etwas aus der Drobox downloaden oder sich etwa einen Cronjob einrichten, der nächtlich Ihre wichtigsten Backups anlegt und in die Dropbox-Cloud hochschiebt. Auf dem Linux-System müssen lediglich wget und cURL installiert sein, damit Sie damit arbeiten können. Root-Rechte sind nicht erforderlich. Geben Sie auf der Shell folgende beiden Zeilen ein:

```
wget bit.ly/1Cqqn4T -O dropbox_uploader.sh
chmod a+x dropbox_uploader.sh
```

Den Umweg über den URL-Kürzer Bit.ly haben wir gewählt, weil der originale URL etwas sperrig ist. Sie können diesen aber natürlich genauso verwenden. Anschließend starten Sie das Skript mit:

```
./dropbox_uploader.sh
```

Das Programm begleitet Sie dann durch den Prozess der Authentifizierung bei Dropbox per oAuth. Dazu rufen Sie die Dropbox-Website auf, loggen sich ein und legen ein Schlüsselpaar für den Zugriff der Anwendung an. Dazu gehört auch die Vergabe eines Namens für die App.

Auf der nächsten Seite können Sie dann einige weiteren Einstellungen vornehmen. Wichtig sind hier aber nur die beiden Codes *App Key* und *App Secret*, die Sie herauskopieren und in das Setup-Skript auf der Konsole übertragen.

Im nächsten Schritt generiert das Skript dann einen weiteren URL, mit dessen Hilfe die API-Anforderung bei Dropbox final bestätigt werden muss. Wenn Sie als Zugangsmodell die Beschränkung auf einen Unterordner innerhalb des App-Verzeichnisses verwenden, wird nach dieser Bestätigung der freigegebene Ordner automatisch erzeugt. Sein Name entspricht dem der App-Bezeichnung.

```
Dropbox Uploader v0.16
Andrea Fabrizi - andrea.fabrizi@gmail.com

Usage: ./dropbox_uploader.sh COMMAND [PARAMETERS]...

Commands:
        upload   <LOCAL_FILE/DIR ...>  <REMOTE_FILE/DIR>
        download <REMOTE_FILE/DIR> [LOCAL_FILE/DIR]
        delete   <REMOTE_FILE/DIR>
        move     <REMOTE_FILE/DIR> <REMOTE_FILE/DIR>
        copy     <REMOTE_FILE/DIR> <REMOTE_FILE/DIR>
        mkdir    <REMOTE_DIR>
        list     [REMOTE_DIR]
        share    <REMOTE_FILE>
        saveurl  <URL> <REMOTE_DIR>
        info
        unlink

Optional parameters:
       -f <FILENAME> Load the configuration file from a specific file
       -s            Skip already existing files when download/upload. Default: Overwrite
       -d            Enable DEBUG mode
       -q            Quiet mode. Don't show messages
       -h            Show file sizes in human readable format
       -p            Show cURL progress meter
       -k            Doesn't check for SSL certificates (insecure)

For more info and examples, please see the README file.
```

Mit dem dem Kommandozeilen-Client Dropbox-Uploader docken Sie von Linux aus schnell an Ihre Dropbox an

Die Kommandos des Dropbox-Clients sind an die Befehle einer üblichen Shell angelehnt. Um eine lokale Datei in die Dropbox zu kopieren, geben Sie beispielsweise ein:

```
./dropbox_uploader.sh upload backups/
```

Ähnlich funktionieren das Herunterladen oder Verschieben. Zu den Besonderheiten gehört das Kommando *share*, das zur Weitergabe von Dateien an andere Personen dient. Dabei geben Sie als Parameter den Pfad einer bereits in der Dropbox liegenden Datei an. Als Antwort erhalten Sie beispielsweise:

```
> Share link: https://db.tt/FYuIZSHX
```

Dieses Link können Sie dann jemandem geben, der sich damit die genannte Datei herunterladen kann, ohne Dropbox-Mitglied zu sein. Die andere Besonderheit ist das Kommando *saveurl*. Es funktioniert sinngemäß so wie *wget*, aber mit einer Speicherung in der Dropbox statt lokal.

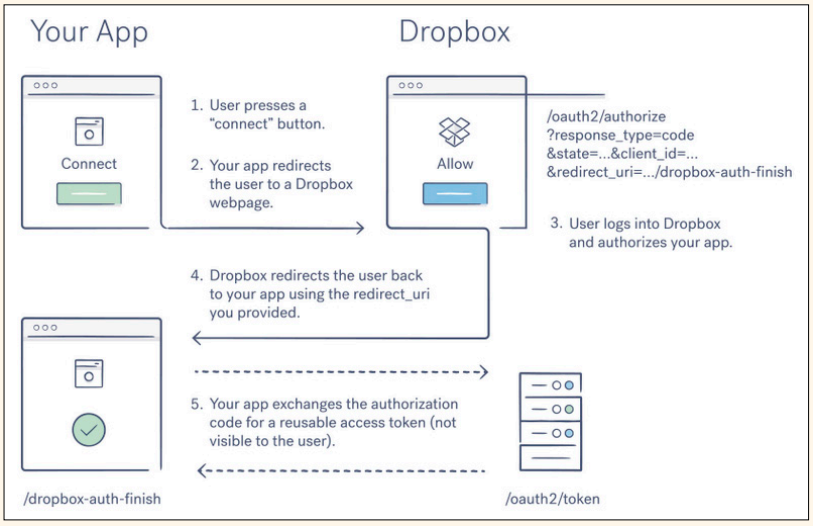
Zugang erlangen

Dropbox verwendet das Prinzip oAuth, um Anwendungen Zugriff auf die Dateien eines Users zu gewähren.

Dabei erhält die Anwendung nicht etwa die Zugangsdaten des Users, sondern es handelt sich um einen mehrstufigen Prozess, bei dem spezielle Zugangsdaten für diese eine Anwendung generiert werden.

Als letzter Schritt in diesem Ablauf muss der Eigentümer des Dropbox-Kontos explizit im Browser bestätigen, dass er den Zugriff erlauben möchte.

Neben dem grundsätzlichen Sicherheitsgewinn hat diese Methode den entscheidenden Vorteil, dass Sie Ihre Anwendung nicht auf die Verwendung mit Ihrem eigenen Dropbox-Konto beschränken müssen. Sie können sie auch so auslegen, dass diese im Namen anderer deren Dropbox-Dateien anfassen darf und dabei nicht Kenntnis vom Passwort des jeweiligen Users haben muss. Die Online-Dokumentation erklärt im Detail die Schritte für die Authentifizierung nach diesem Prinzip, und Sie finden darüber hinaus im Ordner *examples* auch ein detailliertes Beispiel, wie das in einer PHP-Datei realisiert werden kann.



Die Online-Anleitung zum Dropbox-API erläutert ausführlich die Schritte, um eine Authentifizierung mittels oAuth durchzuführen

einerseits eingeschränkt, weil immer nur auf ein Verzeichnis bezogen, und andererseits zu unspezifisch, weil bei einer minimalen Änderung einfach die Metadaten des kompletten Directorys geliefert werden.

Änderungen gezielt abfragen

Ausgefeilter arbeitet da die Methode *getDelta()*. Ohne Parameter aufgerufen liefert sie einen Cursor zurück. Das können Sie wie einen Fingerabdruck für den Zustand der gesamten Dropbox-Hierarchie verstehen, die die App unter sich hat. Dabei handelt es sich genauso um einen codierten String wie bei den Hashes. Rufen Sie nun *getDelta()* erneut auf und liefern den vorigen Cursor mit, dann erhalten Sie ein Array aller Dateien und Verzeichnisse, die sich seit dem letzten Abruf geändert haben. Jeder Eintrag besteht aus dem Namen selbst und dem Array der Metadaten für diese Datei oder das Verzeichnis.

Dabei erscheinen dann auch gelöschte Dateien. Zur Unterscheidung von den neu hinzugekommen oder modifizierten Dateien besteht deren Eintrag lediglich aus den Dateinamen selbst. Listing 2 zeigt ein Skript, das diese Methode nutzt, um die Einträge darzustellen, die sich seit dem letzten Aufruf geändert haben. Dazu speichert es den Cursor in einer Session-Variablen ab und holt ihn beim nächsten Aufruf wieder ab.

Fazit

Dropbox ist einer der beliebtesten Online-Speicher mit toller Software-Unterstützung für Desktops und mobile Geräte. Mit nur wenig Nachhilfe durch eine Library steht auch Ihren Webanwendungen der Weg in die Dropbox-Datenwelt offen. ■

Links zum Thema

- Startseite für die Vergabe von App-Berechtigungen, mit allgemeiner Dokumentation und Tutorials zu den einzelnen Client-Programmiersprachen
  https://www.dropbox.com/developers/apps
- Referenz der PHP-Klassen für die Nutzung des API
  http://dropbox.github.io/dropbox-sdk-php/api-docs/v1.1.x
- Erklärung des Prinzips oAuth für die Zugangsfreigabe
  www.dropbox.com/developers/reference/oauth-guide
- Beispiel für Statusabfragen mittels Long-Polling
  http://github.com/panique/php-long-polling

Markus Schraudolph
ist Journalist und Programmierer. Er schreibt seit 16 Jahren Bücher und Artikel für Fachzeitschriften. Seine Schwerpunktgebiete als Programmierer sind die Webprogrammierung und Datenbanken.

Bild: Shutterstock / NicoElNino

ANGRIFFE AUF LEGITIME FUNKTIONEN IN WEBANWENDUNGEN

Automatisierter Funktionsmissbrauch

Das OWASP Automated Threat Handbook stellt eine Ontologie von Angriffen im Internet dar, die keine Sicherheitslücken ausnutzen.

Öffentlich zugängliche Internetseiten und Webapplikationen sind dem permanenten Risiko ausgesetzt, angegriffen zu werden. Hacker versuchen immer wieder, Websites zu kompromittieren, sensible Daten abzugreifen oder höhere Berechtigungen zu erlangen, als ihnen eigentlich zustehen. Dies sind typische Szenarien, mit denen Webentwickler und Systemadministratoren bei ihrer täglichen Arbeit konfrontiert werden.

Angreifer versuchen hierbei in der Regel, bekannte Schwachstellen auszunutzen, die sich häufig durch Programmier-, Konfigurations- oder konzeptionelle Fehler in die Webanwendungen eingeschlichen haben.

Die OWASP Top Ten ist eines der bekanntesten Projekte, die in den letzten Jahren die am häufigsten festgestellten Sicherheitsprobleme dokumentieren und ausführlich erklären. Die letzte Version der Top Ten wurde im Jahr 2013 veröffentlicht und zählt die typischen Schwachstellen in Software auf: Injections (zum Beispiel SQL Injections), Fehler in Authentifizierung und Session-Management, Cross-Site Scripting (XSS), unsichere direkte Objektreferenzen und sicherheitsrelevante Fehlkonfiguration, um nur die ersten fünf der Schwachstellen zu nennen.

Viele Sicherheitslücken werden heutzutage in automatisierten Angriffen ausgenutzt. Hierfür existiert zahlreiche Software, die den Angriff völlig selbstständig ausführt und nicht eine bestimmte Website aufs Korn nimmt, sondern einfach das Internet nach potenziellen Zielen durchsucht. In diesen Fällen spricht man von nicht gezielten Angriffen. Diese Tools suchen nach verwundbaren Servern oder Websites und scannen innerhalb weniger Minuten eine Vielzahl von IP-Adressen beziehungsweise Servern.

Missbrauch gültiger Funktionen

Es gibt aber noch eine andere Kategorie von automatisierten Angriffen, die in der gängigen Literatur bisher kaum Beachtung findet: Anstatt typische Schwachstellen auszunutzen (wie zum Beispiel SQL Injections), konzentrieren sich diese Angriffe auf den Missbrauch von legitimen Funktionen. Die-

se führen unter normalen Umständen einfache Aufgaben aus und weisen keine technischen Fehler auf.

Um ein Beispiel vorwegzunehmen: Gestohlene Kreditkarten, bei denen dem Angreifer das Ablaufdatum und/oder der Sicherheitscode fehlt, sind ein typischer Fall. Dem Angreifer geht es hierbei nicht darum, die Kreditkarte wirklich zu benutzen, sondern es geht darum, die fehlenden Daten zu ermitteln. Weist ein Payment Gateway keine entsprechenden Sicherheitsvorkehrungen auf, um das automatisierte Durchprobieren dieser Daten zu unterbinden, kann eine gültige Funktion dazu missbraucht werden, die fehlenden Daten zu ermitteln.

Angriffe bleiben häufig unbemerkt

Auch wenn es sich hierbei nicht direkt um einen Angriff auf das Payment Gateway selbst handelt, wird klar, dass keine technische Schwäche wie beispielsweise SQL Injections oder Cross-Site Scripting (XSS) ausgenutzt wird.

Da es sich um die Ausführung von gültigen Funktionen handelt, und nicht um Programmierfehler oder Schwachstellen, stellt dieses Szenario ein ganz neues Problem dar: Es findet derzeit keine Beachtung in der OWASP Top Ten oder ähnlichen Listen.

Tabelle 1: Automatisierte Angriffe

OAT-020	Account Aggregation
OAT-019	Account Creation
OAT-003	Ad Fraud
OAT-009	CAPTCHA Bypass
OAT-010	Card Cracking
OAT-001	Carding
OAT-012	Cashing Out
OAT-007	Credential Cracking
OAT-008	Credential Stuffing
OAT-015	Denial of Service
OAT-006	Expediting
OAT-004	Fingerprinting
OAT-018	Footprinting
OAT-005	Scalping
OAT-011	Scraping
OAT-016	Skewing
OAT-013	Sniping
OAT-017	Spamming
OAT-002	Token Cracking
OAT-014	Vulnerability Scanning

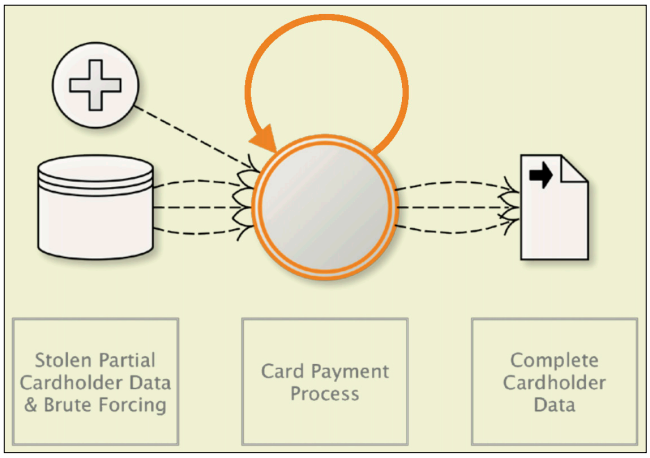


Diagramm zum Ereignis Card Cracking (Bild 1)

Das raffinierte Ausnutzen von gültigen Funktionen anstatt von Sicherheitslücken wird von Website-Betreibern nur selten bemerkt. Ein übermäßiger Missbrauch von Funktionen von Webapplikationen wird hierbei nicht selten fälschlicherweise als ein DoS-Angriff (Denial of Service) gewertet. In Wirklichkeit ist jenes allerdings nur ein Nebeneffekt des eigentlichen Angriffs. HTTP-Flooding kann hier typischerweise erwähnt werden.

Beim zuvor genannten Beispiel bemerken die Website-Betreiber unter Umständen lediglich eine kurzzeitige Zunahme von HTTP-Anfragen an das Payment Gateway. Da aber kein direkter Schaden auf ihren Systemen entstanden ist, bleibt der wahre Hintergrund der Aktion unbemerkt und wird in den meisten Fällen höchstens als versuchter DoS abgetan.

Um Maßnahmen zur Abwehr und zur Prävention von Sicherheitsproblemen zu entwickeln, ist es notwendig, diese zuvor eindeutig zu identifizieren. Ohne ein gemeinsames Vokabular ist es für Sicherheitsexperten und Entwickler schwierig, gemeinsam an Lösungen zu arbeiten.

Das OWASP Automated Threat Handbook

Colin Watson hat sich dieses Themas angenommen und leitet ein Projekt, dessen Hauptziel es ist, Informationen und Ressourcen zur Verfügung zu stellen, um Verteidigungsstrategien gegen diese Art von Bedrohung zu entwickeln und später verifizieren zu können (Tabelle 1).

Die erste Projektphase konzentrierte sich auf die Erarbeitung einer grundsätzlichen gemeinsamen Sprache und die Dokumentation von Terminologien. Genau das stellt das Mitte 2015 veröffentliche Handbuch dar: eine Ontologie über automatisierte Bedrohungen gegen Webapplikationen aus der Sicht der Verteidiger.

Das OWASP Automated Threat Handbook listet derzeit auf knapp 70 Seiten insgesamt 20 Bedrohungen auf und erklärt ausführlich ihre Hintergründe und Zusammenhänge. Die Bedrohungen werden als Ereignisse (Events) bezeichnet.

Eine komplette Vorstellung aller Ereignisse würde den Rahmen dieses Artikels sprengen, daher stelle ich im Folgenden sechs ausgewählte vor, von denen den meisten Lesern sicherlich das ein oder andere bereits begegnet ist. ▶

Das zuvor beschriebene Beispiel mit den gestohlenen Kreditkartendaten fällt unter den Begriff Card Cracking (OAT-010): hierbei handelt es sich um einen Brute-Force-Angriff gegen eine Anwendung, die Kreditkartendaten verifiziert. Das Ermitteln des Gültigkeitsdatums beziehungsweise des Sicherheitscodes steht hierbei im Vordergrund, jedoch nicht, einen Bezahlvorgang zu tätigen (Bild 1).

Das Ablaufdatum der Gültigkeit einer Kreditkarte ist immer als Monat und Jahr definiert (zum Beispiel 03/17 für März 2017). Um sämtliche Monate der nächsten drei Jahre durchzuprobieren sind nur 36 Versuche notwendig. In Verbindung mit dem dreistelligen Sicherheitscode (000 bis 999) wächst die Anzahl auf ein paar Tausend Versuche.

Beim Card Cracking werden allen Kombinationen so lange durchprobiert, bis das Payment Gateway die Kreditkartennummer, ein Gültigkeitsdatum und einen Sicherheitscode akzeptiert. Natürlich sind nicht nur Kreditkarten betroffen, sondern sämtliche Karten im allgemeinen Sinn.

Scalping

Als Scalping (OAT-005) bezeichnet man den Missbrauch einer Anwendung, die zum Erschleichen von Dienstleistungen oder dem unrechtmäßigen Erhalt von Waren führt, die eigentlich nur begrenzt verfügbar sind oder die ein normaler User durch manuelle Nutzung des Systems nicht erhalten kann.

Mit dieser Methode hat häufig die Ticket-Industrie zu tun, wenn etwa sehr gefragte Tickets massenhaft erworben und von den Angreifern später für einen teureren Preis wieder verkauft werden. Auch kann es zu einem benutzerbezogenen DoS-Fall kommen, wenn durch den Angriff Waren oder Dienstleistungen sehr schnell nicht mehr verfügbar sind.

Skewing

Durch den massenhaft wiederholten Klick auf bestimmte Links, den fortlaufenden Besuch von Seiten oder die mehrfache Absendung von Formularen lassen sich unter gewissen Umständen Metriken beeinflussen. Werden diese Metriken für Auswertungen herangezogen, beispielsweise zur Bestimmung der Popularität einer Ware oder Aussage, für Benutzungsstatistiken oder Ähnliches, so wird der Missbrauch als Skewing (OAT-016) bezeichnet.

Praktische Beispiele sind ReTweets bei Twitter, die Manipulation von Download- oder Zugriffszählern, die Bewertung der Aktivität eines Benutzers oder Mitarbeiters et cetera. Wer auf einem virtuellen Marktplatz eine Software anbietet und diese damit bewirbt, sie sei angeblich schon millionenfach heruntergeladen worden, beeinflusst durch diese Tatsache möglicherweise Interessenten.

Solche Metriken haben zum Beispiel Einfluss auf das Ansehen eines Benutzers, können zu Ruhm führen (beziehungsweise das Gegenteil bei einem Konkurrenten herbeiführen) oder das Kaufverhalten von potenziellen Interessenten beeinflussen. Eine ähnliche Angriffsmethode ist das Ad Fraud, bei der es um die Manipulation von Werbeanzeigen geht, wie im Folgenden beschrieben.

Ad Fraud

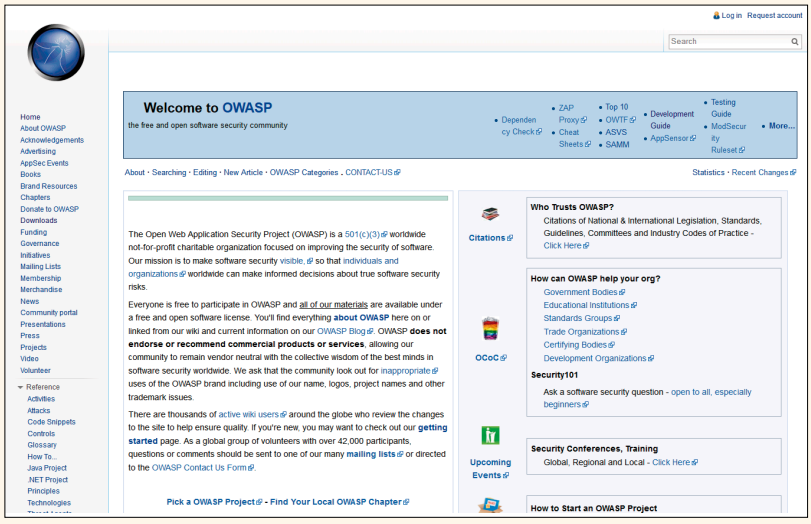
Werbeanzeigen auf Websites kennt heutzutage jeder. Grundsätzlich erhalten Website-Betreiber für zwei unterschiedliche Möglichkeiten Werbeeinnahmen: entweder für die Anzahl, wie oft eine Anzeige dargestellt wurde, oder für die Anzahl, wie häufig Besucher darauf klickten (Click-through). Durch die Beeinflussung einer oder beider dieser Metriken kann ein

Über OWASP

Das im Jahr 2001 von Mark Curphey gegründete Open Web Application Security Project (OWASP) hat sich zum Ziel gesetzt, die Sicherheit von Anwendungen und Diensten im World Wide Web zu verbessern.

Durch frei verfügbare Informationsmaterialien, Methoden, Werkzeuge und Technologien soll eine Transparenz geschaffen werden, die es Endanwendern, Unternehmen und Organisationen erlaubt, fundierte Entscheidungen über Sicherheitsrisiken in Software-Applikationen treffen zu können.

Die heutige OWASP Foundation ist eine in den USA registrierte gemeinnützige Organisation. Sie unterhält eine Vielzahl von verschiedenen Projekten rund um das Thema Sicherheit im Internet und in Anwendungen. Konferenzen auf der ganzen Welt mit namhaften Sprechern runden das Angebot von OWASP ab. Knapp 50 Organisationen und Firmen, darunter beispielsweise Akamai, Rackspace, Salesforce.com und Symantec, zählen zu den aktuellen Mitgliedern der Foundation. Zusätzlich wird sie derzeit von über 2400 Privatpersonen finanziell unterstützt.



Die Projektseite der OWASP Foundation

Das Buch gibt es als Download und als Print-on-Demand-Variante (Bild 2)

Website-Betreiber versuchen, vom Werbetreibenden mehr Einnahmen zu erschleichen, als ihm eigentlich zustehen. Gleichzeitig kann ein Konkurrent auch versuchen, durch eine eindeutige und auffällige Manipulation das Verhältnis zwischen Werbenden und dem Website-Betreiber dermaßen zu stören, dass der Werbende lieber bei ihm wirbt. Diese Angriffe auf digitale Werbeanzeigen im Internet bezeichnet man laut dem OWASP Automated Threat Handbook als Ad Fraud (OAT-003).

Sniping

Auktionen im Internet sind nicht zuletzt durch Ebay in den letzten zwei Jahrzehnten zur Normalität geworden. Online-Auktionshäuser müssen sich seit jeher gegen Käufer wehren, die versuchen, ihre Kontrahenten mit unzulässigen Mitteln zu überbieten. Bei Verkäufen, in denen ein Angebot so kurz vor Ablauf der Auktion (automatisch) abgegeben wird, dass kein Mitbieter die Chance hat, jenes zu überbieten, wird diese unerlaubte Aktion als Sniping (OAT-013) bezeichnet. Hierbei muss es sich nicht unbedingt um Waren handeln, die erschlichen werden, sondern es können auch Dienstleistungen oder virtuelle Güter sein.

Neben Online-Auktionshäusern ist vor allem die Online-Spieleindustrie ein typisches Ziel dieser Methode.

Scraping

Als Scraping (OAT-011) bezeichnet man das automatische Abgreifen von Inhalten im Internet, um diese an anderer Stelle zu verwenden. Das können relativ einfache Angriffe sein, bei denen Spammer Websites nach E-Mail-Adressen durchsuchen, um an diese später Spam-Mails zu senden.

Es existieren komplexere Szenarien, bei denen ein Drittanbieter Interesse an bestimmten Informationen hat und der Angreifer diese Daten bei anderen Websites abgreift (Namen, Adressen oder Inhalte, wie beispielsweise Bilder, Artikel oder Informationen aller Art).

Fazit

Die vorgestellten Beispiele machen deutlich, um welche Art von Angriffen es sich handelt: Hacker benutzen Webapplikationen lediglich als ihre Werkzeuge. Das Angriffsziel sind nicht die Websites, sondern es werden auf ihnen legitime Funktionen missbraucht, die eine Applikation bereitstellt. Die Websites funktionieren weiterhin (sofern sie dem DoS als möglichen Nebeneffekt standhalten), und selbst registrierte Benutzer dieser Websites sind selten die eigentlichen Opfer.

Auch ist wichtig festzuhalten, dass hier keine Implementierungsfehler ausgenutzt werden. Die Software arbeitet fehlerfrei, und wenn sogar einige Hunderttausend HTTP-Anfragen innerhalb weniger Sekunden verarbeitet werden können, wahrscheinlich sogar ausgesprochen gut.

Das Problem sind häufig Designfehler, die diese Angriffe erlauben und für die es bisher noch keine anerkannte Terminologie gab. Mit dem OWASP Automated Threat Handbook soll diese Situation verbessert werden.

Die Tatsache, dass das Handbuch eine Pflichtlektüre für jeden Sicherheitsexperten ist, steht außer Frage. Im Vergleich zur OWASP Top Ten konzentriert es sich allerdings eher auf konzeptionelle Aspekte, jedoch in einem technischen Jargon. Auch nichttechnisches Personal, das mit Begriffen wie SQL/Code Injection, Cross-Site-Scripting und unsichere direkte Objektreferenzierung nicht viel anfangen kann, wird beim Durchlesen der Beschreibungen von Card Cracking oder Ad Fraud schnell verstehen, was damit gemeint ist. Somit können Mitarbeiter abschätzen, ob möglicherweise die Software betroffen ist, die in ihrer Firma entwickelt wird.

Neben den sechs in diesem Artikel vorgestellten Ereignissen beinhaltet die aktuelle Version des OWASP Automated Threat Handbook 14 weitere. Jedes einzelne Ereignis wird ausführlich erklärt, ist mit einem Diagramm zum besseren Verständnis versehen und enthält außerdem eine Liste von Begriffen, unter denen der Angriff ebenfalls bekannt ist.

Uns Webentwicklern kommt es gelegen, dass sich die aktuelle Ausgabe ausschließlich auf Webanwendung konzentriert. Das Handbuch liegt derzeit nur in englischer Sprache vor, wurde unter der Creative Commons Attribution-ShareAlike 3.0 Lizenz veröffentlicht und kann kostenlos von der Internetseite des OWASP heruntergeladen werden (Bild 2). Eine gedruckte Version ist ebenfalls erhältlich: Diese kann als Print-on-Demand-Buch bestellt werden, ist allerdings kostenpflichtig. ■

Michael Schams
lebt seit 2008 in Australien und arbeitet als Projekt Manager und IT Security Consultant. Er ist zertifizierter TYPO3 Integrator, Project Leader des offiziellen TYPO3 Security Guides und berät Kunden weltweit.
@MichaelSchams

Links zum Thema

- OWASP-Website
  http://owasp.org
- Vortrag von Colin Watson (2015)
  https://www.youtube.com/watch?v=JZZn2HfClEs

GRAFIK FÜR ENTWICKLER

Formen wahrnehmen

Ein stabiles Quadrat, das richtungsweisende Dreieck oder der in sich ruhende Kreis – was Formen auf der Fläche bewirken.

Bilder und Grafiken erzeugen beim Betrachter eine größere Aufmerksamkeit als Textelemente. Dabei spielt es keine Rolle, ob es sich um ein Printprodukt oder um ein digitales Erzeugnis handelt. Zudem nehmen die meisten von uns die Aussage der gezeigten Bilder deutlich schneller wahr als Geschriebenes: Text müssen wir zunächst in die dazugehörigen Bilder umrechnen, visuelle Reize hingegen können wir ohne Umwege verarbeiten.

Eine sehr reduzierte Grafik, etwa eine einfache geometrische Form, lässt einen größeren Freiraum an Interpretationsmöglichkeiten als eine noch so einfache Zeichnung, die bereits die Formen eines bestimmten Objekts andeutet. So verknüpfen wir mit den meisten Formen eine bestimmte Eigenschaft oder weisen ihr eine Funktionalität zu. Gerade bei digitalen Anwendungen übernehmen manche Formen oft eine bestimmte Funktion. So vermittelt hier etwa ein Rechteck mit abgerundeten Ecken und Text darauf, dass es eine Schaltfläche ist (Bild 1).

Chefkoch.de: Die grünen, leicht abgerundeten Rechtecke signalisieren bereits durch ihre Form, dass es sich um Schaltflächen handelt (Bild 1)

Google Maps: Auf der Karte legt der Punkt die genaue Position der Städte fest (Bild 2)

Linien und ihre Wirkung

Der Punkt stellt die kleinste Gestaltungseinheit dar. Wobei der »echte« Punkt nicht darstellbar ist; vielmehr handelt es sich um die kleinstmögliche Fläche. So sind für uns auch die kleinen Kreise, die die Position der Städte auf der Karte anzeigen, Punkte, obwohl sie eigentlich kleine runde Flächen mit einem schwarzen Rand sind (Bild 2).

Ein Punkt signalisiert die Mitte, beendet einen Satz, oder er ist als Pixel Teil einer jeden Darstellung auf dem Bildschirm.

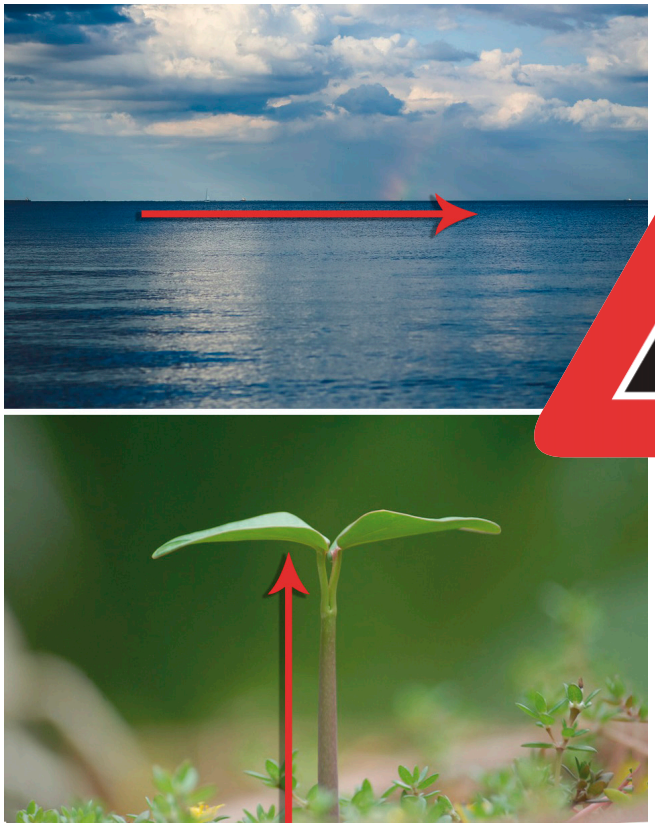
Die Linie wird durch zwei Punkte definiert, die in einer beliebigen Entfernung voneinander liegen. Die Platzierung von Anfangs- und Endpunkt definiert die Richtung der Linie, die zusammen mit der Farbe, der Stärke und anderen optischen Merkmalen deren Wirkung bestimmt. So vermittelt eine horizontale Linie Weite und Sicherheit. Sie liegt und befindet sich somit in einer sehr stabilen Position. Horizontalen erinnern – wie der Name bereits erkennen lässt – beispielsweise an den Horizont. Eine vertikale Linie zeigt die Richtung des Wachstums, etwa bei einer Pflanze. Somit wirkt die Vertikale deutlich dynamischer als eine Waagerechte (Bild 3).

Verläuft die Linie diagonal, wirkt sie eher unruhig. Ihre Position ist instabil, hier ist eine Entwicklung zu erwarten. So sehen wir eine Linie, die von links unten nach rechts oben verläuft, als steigend. Diese positive Wirkung machen sich viele Grafiker bei der Entwicklung eines Firmenlogos zunutze, beispielsweise bei dem Logo der Deutschen Bank oder dem des Computerherstellers Dell.

Abfallende Linien, also Diagonalen, die von links oben nach rechts unten zeigen, weisen den Weg nach unten, sie dienen somit nicht unbedingt dazu, beim Betrachter positive Gefühle zu erzeugen. Einer solchen Linie begegnen wir beispielsweise bei dem Verkehrswarnschild vor einem Gefälle (Bild 4).

Ist die Linie gekrümmt, wird sie zur Kurve, beziehungsweise zum Bogen. Hier wird mindestens ein weiterer Punkt auf der Linie zum Scheitelpunkt, der die Stärke und die Richtung der Krümmung bestimmt.

Kurven wirken weicher als gerade Linien, sie vermitteln Spontanität, Bewegung und Offenheit, da sie noch keine ge-

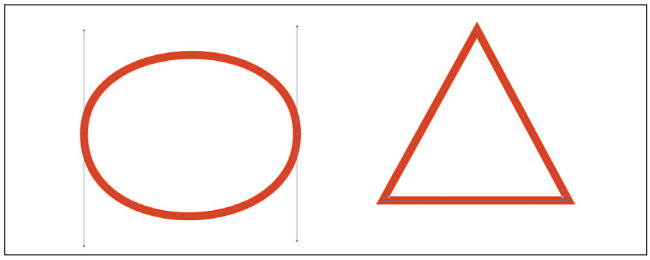


Die Horizontale (oben) vermittelt Sicherheit, die senkrechte Linie zeigt die Richtung von Wachstum an (Bild 3)

Das Warnschild zeigt anhand einer geneigten Horizontlinie, dass es auf dem Weg nach unten geht (Bild 4)

Wella – Intel – Nike: Kurven zeigen sehr unterschiedliche Wirkungen, die durch die entsprechende Farbe unterstützt werden (Bild 5)

schlossene Form bilden. Kurven können in sehr unterschiedlichen Ausführungen vorkommen und können daher vielfältige Gefühle beim Betrachter erzeugen. Hier spielt auch die Farbe der Grafik eine entscheidende Rolle. Im Bild 5 vermitteln die gezeigten Logos sehr unterschiedliche Werte: Wella, der Hersteller für Haarpflegemittel, verknüpft in seinem Logo Text und Bild. Drei Kurven stehen für die Haare, die mit den Produkten von Wella zu dynamischen Haar-Wellen werden. Die beiden Kurven im Logo des Halbleiterherstellers Intel hingegen zeigen sich wesentlich weniger verspielt. Hier verknüpft der Betrachter die Formen eher zu einem Ganzen – wie eben auch elektronische Bauteile in ihrer Kombination zu einem Produkt werden. Wie bei Dell zeigt auch dieses Logo ein kühles Blau. Der Sportartikelhersteller Nike kommt mit einer Kurve aus, die Bewegung nach vorne signalisiert. Zudem ist die Kurve aufsteigend, also in der Richtung positiv behaftet. Das Logo ist in neutralem Schwarz gehalten, zeigt aber auf jedem Artikel eine passende Farbe.



Eine ovalrunde Form, gekennzeichnet durch zwei Scheitelpunkte, und ein Dreieck, bestimmt durch drei Eckpunkte (Bild 6)

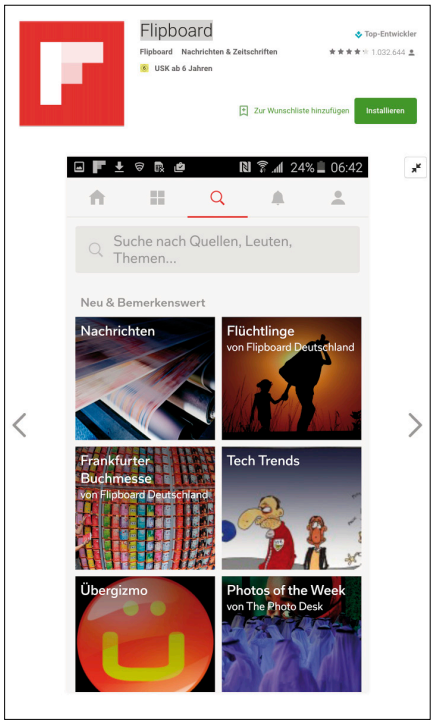
Flächen und ihre Wirkung

Werden mehr als zwei Punkte miteinander verbunden und entspricht der Startpunkt dem Endpunkt, entsteht eine Fläche, die sich im zweidimensionalen Raum befindet. So werden für die Bestimmung einer geschlossenen Form nur zwei Scheitelpunkte einer Kurve benötigt. Handelt es sich um Eckpunkte, bilden drei davon ein Dreieck (Bild 6).

Das Rechteck ist die Form, die uns in allen Medien am häufigsten begegnet. So füllt Text in der Regel einen rechteckigen Rahmen aus, der in Spalten unterteilt werden, kann sodass mehrere Rechtecke entstehen. Bilder belegen ebenfalls meistens rechteckige Flächen. Zudem – und diese Tatsache ist mit entscheidend für jegliche Formwahl innerhalb eines Layouts – ist nahezu jede Grundfläche, die es zu befüllen gilt, ein Rechteck.

Wie auch immer das Seitenverhältnis eines Rechtecks sein mag, es liegt in der Regel sehr stabil auf der Grundfläche. Dadurch wirkt diese Form schnell etwas träge, besonders wenn die Horizontalen länger sind als die senkrechten Begrenzungslinien. Je länger die Senkrechten im Verhältnis zu den Waagerechten sind, desto instabiler wirkt das Rechteck und desto mehr gewinnt es an Dynamik. Somit zeigt das Rechteck dann eine Richtung, ähnlich wie die Linie.

Ebenso statisch und dabei besonders stabil wie das Rechteck wirkt das Quadrat mit seinen vier gleichen Seiten. Diese Sonderform des Rechtecks erinnert beispielsweise an Bausteine; mehrere Quadrate der gleichen Größe lassen sich problemlos auf einer beliebigen Fläche anordnen und austauschen, ohne dass ungleiche Lücken entstehen. Ein solches Raster aus Quadraten verwendet zum Beispiel die Magazin-App Flipboard. Bereits das Logo zeigt die symmetrische ▶



Die Oberfläche der App Flipboard ist in Quadrate geteilt, die sich wie in einem Baukastensystem beliebig austauschen lassen (Bild 7)

Ein gekipptes Quadrat kommt als Verkehrsschild zum Einsatz (Bild 8)

Die Website von Vodafone verwendet im oberen Bereich angeschnittene Dreiecke, deren Spitzen zum Zentrum zeigen (Bild 9)

Anordnung. Neben dem Quadrat in der Mitte bilden bei den beiden anderen Rechtecken die Längsseiten ein Vielfaches der kürzeren Seiten. Somit ergeben sie, zusammen mit der Grundfläche, ein symmetrisches Raster (Bild 7).

Wird das Quadrat um 45 Grad gedreht, steht es auf einer Ecke und zeigt sich deutlich instabiler. Die vier Begrenzungslinien liegen nunmehr diagonal im Raum, wodurch die Form, die zu jeder Zeit in eine Richtung kippen könnte, an Dynamik gewinnt. Diese Form ist beispielsweise beim Verkehrsschild für Vorfahrtsstraßen zu finden (Bild 8).

Ein Dreieck ist richtungsweisend: Liegt ein Schenkel auf der Horizontalen und weist eine Spitze nach oben oder nach unten, führt es den Blick des Betrachters in die jeweilige Richtung. Dabei zeigt sich die letztere Variante deutlich instabiler; ähnlich wie beim gekippten Quadrat. Beide Formen sind bei verschiedenen Verkehrsschildern zu finden, wobei hier die dreieckigen Schilder zu den Warn- und Gefahrzeichen zählen. Zudem verknüpfen wir intuitiv Bilder aus unserer Erfahrungswelt mit der jeweiligen Form. Liegt der horizontale Schenkel beispielsweise unten, wird das Dreieck zur Pyramide oder zum Schutz bietenden Dach eines Hauses.

Mit verschiedenen, zueinander angeordneten Dreiecken arbeitet das Design der Website von Vodafone: Im oberen Bereich liegt links ein angeschnittenes rotes Dreieck, dessen Spitze auf eine Slider-Bildershow zeigt. Hier stoßen die Spitzen von drei weiteren, ebenfalls angeschnittenen Dreiecken in der Mitte zusammen und führen den Blick in das Zentrum der Slider-Fläche, in der das Angebot steht. Die wichtigen Bildteile befinden sich dabei in den stabileren Dreiecken unten und rechts. Das obere, von der Wirkung instabilere Dreieck, dessen Spitze nach unten zeigt, ergänzt lediglich die Zusammenstellung aller Formen und ist in einem neutralen hellen Grau belassen (Bild 9).

Der Kreis ist das Symbol der Geschlossenheit und der Ewigkeit. Anders als bei allen eckigen Formen und durch seine gleichmäßige Krümmung gibt es hier keine Richtung, der Kreis ruht in sich selbst. Ein Kreis kann die Sonne oder den Mond darstellen, kann also selbst zum Zentrum werden oder um ein solches kreisen. Zudem bildet die Ansicht eines Balls auf einer zweidimensionalen Fläche einen Kreis, ebenso eine Münze, das Rad oder eine Scheibe. Ein sehr kleiner Kreis wird zum Punkt. Im Gegensatz zu den eher maskulin wirkenden eckigen Formen wirkt der Kreis dagegen feminin.

Beispiel Mona-net: Das Online-Netzwerk für junge Frauen zeigt ein Layout, das der Zielgruppe gerecht wird. Gehalten in warmen Rottönen, gibt es hier kaum eckige Formen. So sind alle Rechtecke abgerundet und der Farbbalken im oberen Bereich zeigt eine weiche Kante. Als Hintergrund wurde

Mona-net.at: Das Layout in warmen Rottönen und mit runden Formen richtet sich eindeutig an eine weibliche Zielgruppe (Bild 10)

Obwohl die Ecken nicht miteinander verbunden sind, ergänzt das Auge die vier Ecken zu einem Rechteck (Bild 11)

Vier Rechtecke bilden in dieser Anordnung eine kreuzförmige Struktur (Bild 12)

Das Quadrat oben rechts sorgt für Spannung in der sonst etwas problematischen Anordnung der Formen (Bild 13)

ein Muster aus verschiedenen Kreisen gewählt (Bild 10).

Form und Grundfläche

Steht eine Form nicht für sich alleine, wie etwa bei einem Verkehrsschild, muss sie in Bezug zur Grundfläche stehen. Befinden sich mehrere Formen auf dieser Grundfläche, stehen auch sie zueinander in Beziehung. Ein gutes Layout gelingt, wenn alle Formen zusammen mit der Grundfläche ein harmonisches Ganzes bilden. Verschiedene Überlegungen helfen hierbei.

Bei der Anordnung sollte die Blickführung berücksichtigt werden. Diese orientiert sich an der Leserichtung, sie geht also in unserem Kulturraum von links nach rechts und von oben nach unten.

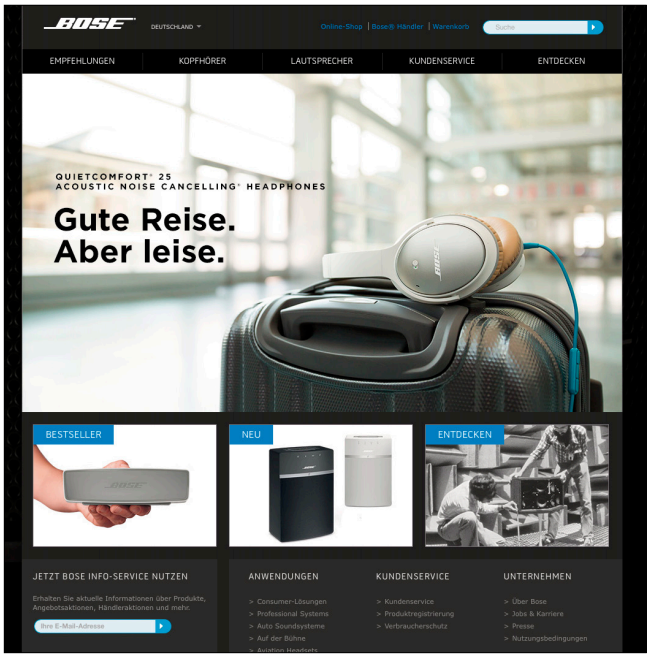
Befinden sich verschiedene optische Zentren auf der Fläche, tastet das Auge zunächst die wichtigsten Elemente ab, um sich dann den optisch weniger reizvollen Flächen zu widmen, etwa längeren Textblöcken. So sollten besonders auffällige Elemente nicht am Rand platziert werden, da so der Blick aus der Fläche herausgelenkt wird.

Zu berücksichtigen sind hierbei auch die gedachten Linien. Stehen zwei oder mehrere Flächen oder Linien in Bezug zueinander, werden sie durch unsichtbare Linien zu einem Ganzen ergänzt (Bild 11).

Problematisch wird es, wenn die verschiedenen Objekte auf der Fläche eine kreuzförmige Struktur bilden. So entstehen dabei einerseits vier gleiche Bereiche, die wenig Spannung bieten und eine gezielte Blickführung erschweren. Zudem haftet einer solchen Komposition unterschwellig die Symbolik des Kreuzes an (Bild 12).

Die optische Spannung der Fläche leidet häufig auch dann, wenn durch die Anordnung der Elemente eine Treppe entsteht. Eine weitere Form kann hier jedoch für ein Gegengewicht sorgen und so wieder Spannung in das Layout bringen (Bild 13).

Einer guten Flächenaufteilung liegt immer ein Gestaltungsraster zugrunde. So teilt etwa das Framework 960.gs die Fläche in wahlweise 12 oder 16 Spalten auf, die einen definierten Abstand zueinander haben. In diese Spalten werden die verschiedenen Designelemente gesetzt. Als Hilfe gibt es für den Anwender die entsprechenden Grids zum Download unter http://960.gs für alle gängigen Anwendungen wie InDesign, Illustrator, Photoshop oder auch Gimp und Inkscape.



Die Webseite Bose.de zeigt drei nebeneinanderliegende Designelemente (Bild 14)

Auch Gestaltungsregeln aus der Fotografie können beim Festlegen eines Layouts helfen. So etwa die Drittel-Regel: Dabei wird das Bild durch zwei senkrechte und zwei waagerechte imaginäre Hilfslinien in neun gleich große Flächen geteilt. Das Motiv liegt dann auf einer der Linien oder auf einem Schnittpunkt. Generell ist die drei eine spannende Zahl; diese Tatsache mag wohl darin begründet sein, dass zu einer Gruppe mindestens drei Objekte gehören. So finden sich in einem Layout oft Dreiergruppen, etwa die Textspalten im Magazinbereich. Auch Websites zeigen oft eine Aufteilung der Fläche in drei nebeneinanderliegende Designblöcke (Bild 14).

Fazit

Jede Grundform liefert dem Betrachter eine Vielzahl von Informationen, die er teilweise nur unbewusst wahrnimmt. Die richtige Wahl der Formen spielt also eine wichtige Rolle beim Planen eines guten Layouts für Websites und Applikationen. Zudem sollte das Zusammenspiel der einzelnen Formen auf seine Wirkung überprüft werden. Hilfreich ist in jedem Fall ein Gestaltungsraster.

■

Katharina Sckommodau
arbeitet als freiberufliche Autorin, Grafikerin und Dozentin, unter anderem für die Akademie der Bayerischen Presse und für Macromedia. Sie veröffentlicht regelmäßig Beiträge in Fachzeitschriften und Buchprojekten.

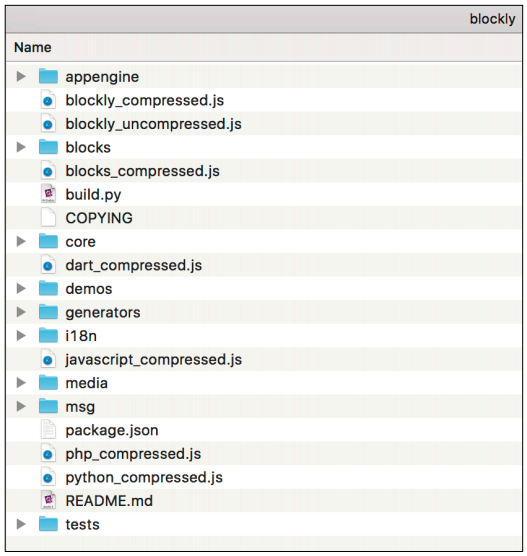
GRAFISCHE EDITOREN FÜR CODE, DSL-SPRACHEN UND KONFIGURATIONEN

Block statt Code

Für die Programmierung muss man nicht zwingend eine Syntax erlernen. Es reicht auch völlig die Bedienung eines grafischen Editors, der dann lauffähigen Code erzeugt.

Grafische Editoren zur Erstellung von Programmcode haben nicht zuletzt durch Scratch eine größere Verbreitung gefunden. Scratch kommt vor allem im Ausbildungsbereich zum Einsatz, um Kindern die Programmierung anhand von Spielen zu vermitteln. Eine Alternative zu Scratch stellt Blockly dar. Wobei der große Unterschied darin besteht, dass Scratch eine Anwendung ist und Blockly eine Programmbibliothek zur Erstellung von grafischen Editoren auf der Basis von Blöcken.

Blockly steht unter der Apache 2.0 License zur Verfügung. Die erste Version von Blockly wurde im Jahr 2012 von Google veröffentlicht und wurde entwickelt, um OpenBlocks in App Inventor zu ersetzen. Auf der Projektseite ist ein Blockly-Editor direkt integriert und kann dort für einen ersten Eindruck direkt getestet werden (Bild 1).



Nach dem Klonen des Projekts stehen alle Verzeichnisse des Blockly-Projekts lokal zur Verfügung (Bild 2)

Installation von Blockly

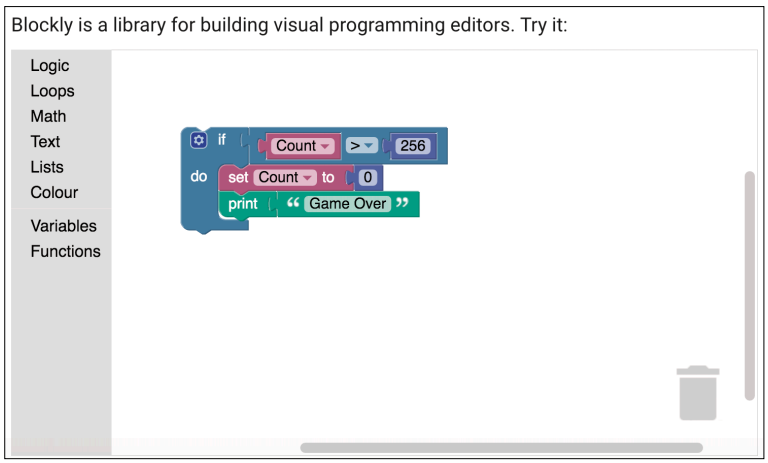
Da es sich bei Blockly um eine JavaScript-Bibliothek handelt, besteht die Installation lediglich aus der Einbindung der richtigen Datei. Hierfür muss zunächst der Quellcode von Blockly aus dem GitHub-Repository heruntergeladen werden:

```
git clone https://github.com/
google/blockly
```

Nach Ausführung des obigen Befehls wird im aktuellen Verzeichnis ein Unterverzeichnis mit dem Namen *blockly* erstellt (Bild 2).

Alternativ kann der Quellcode auch als ZIP- oder Tarball-Archiv heruntergeladen werden. Um sicherzustellen, dass die aktuellste Version heruntergeladen wurde, empfiehlt sich jedoch die Nutzung von Git.

Nach dem Herunterladen des Quellcodes sollte geprüft werden, ob Blockly auch in der lokalen Variante auf dem Entwicklungsrechner funktioniert. Hierfür stehen im Verzeichnis *demos* fertige HTML-Seiten zur Verfügung. Am besten öffnen Sie im Browser die Datei *demos/fixed/index.html* (Bild 3). Alternativ kann auch die Datei *demos/index.html* aufgerufen werden. Diese Seite ist die Navigationsseite für alle mitgelieferten Demoprogramme.



Auf der Projektseite kann Blockly direkt getestet werden (Bild 1)

Integration in eine HTML-Seite

Der einfachste Weg, um Blockly in einer Webanwendung zu nutzen, ist die Definition eines eigenen <*div*>-Blocks für den Editor. Der Editor kann dabei eine feste Größe haben oder sich auch der Größe des Browserfensters anpassen. Unabhängig von der gewählten Variante müssen zunächst die entsprechenden JavaScript-Dateien eingebunden werden. Für die produktive Nutzung steht Blockly in einer minifizierten Variante zu Verfügung; um das Debugging in der Entwicklung zu erleichtern, steht auch eine nicht minifizierte Variante bereit. Die minifizierten Dateien sind mit *_compressed* gekennzeichnet und die nicht minifizierten Dateien mit *_uncompressed:*

```
<script src = "blockly_compressed.js">
</script>
```

Listing 1: Blockly mit leerer Toolbox

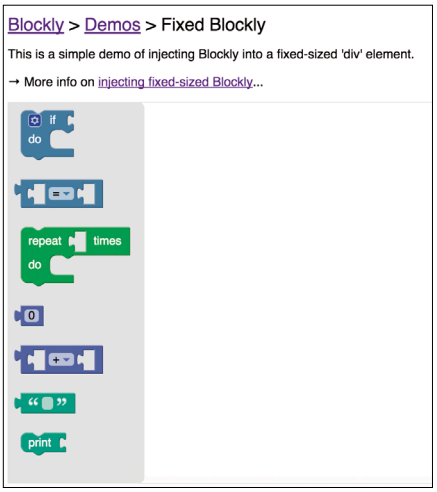
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <!-- Minifizierte Variante nutzen -->
    <script src="../../blockly_compressed.js">
    </script>
    <script src="../../blocks_compressed.js">
    </script>
    <!-- Deutsche Oberflaeche -->
    <script src="../../msg/js/de.js"></script>
  </head>
  <body>
    <h1>Blockly mit fester Gr&ouml;&szlig;e</h1>
    <div id="blocklyDiv" style="height: 480px;
    width: 600px;"></div>
    <!-- Definition einer leeren Toolbox -->
    <xml id="toolbox" style="display: none"></xml>
    <script>
      var workspace = Blockly.inject('blocklyDiv',
      {media: '../../media/', toolbox:
      document.getElementById('toolbox')});
    </script>
  </body>
</html>
```

```
<script src="blocks_compressed.js"></script >
<!-- Sprachdatei nicht minifiziert -->
<script src = "msg/js/de.js" > </script>
```

Für alle hier gezeigten Beispiele wurde die aktuellste Version von Blockly aus dem GitHub-Repository (https://github.com/google/blockly) genutzt. Die Beispieldateien nutzen als Basisverzeichnis *demos/webmobdev*. Als Browser für die Tests wurde Chrome in Version 49.0.2623.87 unter Mac OS X 10.11.3 eingesetzt. Alle hier gezeigten Beispiele stehen in folgendem GitHub-Repository bereit: https://github.com/mstaeuble/webmobdev-blockly. Bei jedem Listing, das im genannten Repository liegt, ist der Dateiname angegeben.

Für die Nutzung von Blockly müssen mindestens drei JavaScript-Dateien eingebunden werden: die Basisbibliothek *(blockly_compressed.js* oder *blockly_uncompressed.js)*, die Basisblöcke *(blocks_compressed.js* oder *blocks_uncompressed.js)* und eine Sprachdatei für die Oberfläche.

Die Oberfläche steht im Repository in mehreren Sprachen zur Verfügung, derzeit insgesamt über 40 Sprachen.



Viele Demos helfen beim Einstieg in Blockly (Bild 3)

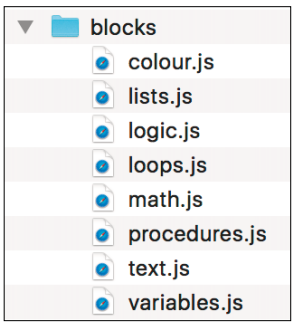
Die einzelnen Dateien sind im Verzeichnis *msg* abgelegt. Für die deutsche Oberfläche muss *msg/js/de.js* eingebunden werden, für die englische Version *msg/js/en.js*.

Nun muss noch ein *div*-Block für Blockly definiert werden. Für dieses Beispiel soll Blockly mit einer festen Größe verwendet werden:

```
<div id="blocklyDiv"
style="height: 480px;
width: 600px;"></div>
```

Um den Arbeitsbereich von Blockly sichtbar zu machen, muss auch eine Toolbox für die Blöcke definiert werden. Diese wird im ersten Schritt als leere Box festgelegt:

```
<xml id="toolbox"
style="display: none"></xml>
```



Wer an Details einzelner Blockdefinitionen interessiert ist, findet diese im Verzeichnis *blocks* (Bild 4)

Block mit Rückgabewert Test, erkennbar an der linken Lasche (Bild 5)

Nach der Definition aller nötigen Elemente kann Blockly über die JavaScript-Funktion *Blockly.inject* eingebunden werden. In Listing 1 ist die komplette HTML-Datei abgebildet.

Bei der Integration von Blockly in eigene HTML-Seiten ist es wichtig, den Doctype zu setzen. Ohne die Festlegung des Doctypes funktionieren einige Funktionen von Blockly nicht, wie zum Beispiel das Ausblenden des Beschreibungsfelds bei den Funktionen.

Blocksprache

Die Basis von Blockly sind Blöcke, aus denen das Programm erstellt wird. Für die Erstellung stehen viele unterschiedliche Blöcke zur Verfügung. Die Definitionen für die einzelnen Blöcke liegen im Verzeichnis *blocks* (Bild 4). Es gibt zwar ein Wiki zu den Blocks, doch leider ist dieses nicht vollständig. Somit muss man sich die Block-Definitionen entweder selbst anhand des Codes oder über die zahlreichen Demos erarbeiten.

Das Prinzip der Blocksprache ist, dass Blöcke zu einem Programm zusammengesetzt werden. Die Blöcke haben dabei Laschen und öffnende Verbinder, ähnlich einem Puzzle. Dabei gibt es Blöcke, die einen Rückgabewert haben, erkennbar an der linken Lasche (Bild 5). Dieser Wert kann entweder per Pulldown-Menü (Bild 6) oder per Eingabe (Bild 7) festgelegt werden. Es gibt aber auch Blöcke, bei denen der ▶

Rückgabewert berechnet wird und somit keine manuelle Festlegung des Werts möglich ist, wie bei einem Vergleich.

Die Rückgabewerte können als Wert für andere Blöcke verwendet werden. Blöcke die einen Wert aufnehmen können, besitzen einen öffnenden Verbinder (Bild 8).

Es gibt Blöcke, die weitere Optionen anbieten. Diese Optionen werden durch blaue Icons im Block dargestellt (Bild 9). Eine der Optionen ist das Kommentarfeld, visualisiert durch ein Fragezeichen (Bild 10). Zusätzlich kann ein Block unterschiedliche Varianten anbieten, die durch Klick auf das Zahnrad angezeigt werden (Bild 11).

Auch die Verschachtelung von Blöcken ist möglich. Bei Blöcken, bei denen dies möglich ist, steht eine Rundung zur Verfügung, wie zum Beispiel bei der *if*-Abfrage (Bild 12).

Konfiguration der Toolbox

Neben den Blöcken selbst ist das wichtigste Element im Editor die Toolbox zur Aufnahme der Blöcke. Die Toolbox bietet reichhaltige Optionen an. Welche Blöcke in einem Editor zur Verfügung stehen, wird bei der Konfiguration der Toolbox festgelegt. Ein einzelner Block wird über das Element *block* zur Toolbox hinzugefügt. Die Reihenfolge der Anordnung ergibt sich durch die Reihenfolge in der Definition. Die verfügbaren Blöcke kann man sich im Verzeichnis *blocks* ansehen.

Im nachfolgenden Listing werden zwei Blöcke definiert, ein *if*- und ein *while-until*-Block. Das Ergebnis ist in Bild 13 dargestellt. Standardmäßig sind alle hinzugefügten Blöcke in der Toolbox aktiv und können vom Anwender genutzt werden. Über den Parameter *disabled* können einzelne Blöcke deaktiviert werden und zum Beispiel nur für Anwender mit bestimmten Rechten freigeschaltet werden:

```
<xml id="toolbox" style="display: none">
  <block type="controls_if"></block>
  <block type="controls_whileUntil"></block>
</xml>
```

Bei der Nutzung von vielen Blöcken in der Toolbox kann der Platz knapp werden. Um trotzdem mehr Blöcke zu nutzen, können Sie entweder auf das in der Toolbox integrierte Scrolling setzen oder die Blöcke in Kategorien aufteilen. Bei größeren Programmen kann der Arbeitsbereich schnell zu klein werden. Hierfür kann ein Scrolling aktiviert werden, dies ist bei der Nutzung von Kategorien direkt automatisch aktiviert. Die Aktivierung erfolgt über den Parameter *scrollbars* bei *Blockly.inject:*

```
var workspace = Blockly.inject('blocklyDiv',
{media: '../../media/', toolbox: document.
getElementById('toolbox'), scrollbars: true});
```

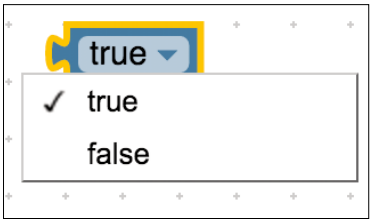
Bei der Nutzung von Kategorien wird im Arbeitsbereich auch direkt ein Mülleimer angezeigt. Elemente können vom Arbeitsbereich dort hineingezogen werden, um sie zu löschen. Möchte man den Mülleimer auch in einem definierten Editor ohne Kategorien anzeigen, so muss beim Aufruf von *Blockly.inject* der Parameter *trashcan* auf *true* gesetzt werden:

```
var workspace = Blockly.inject('blocklyDiv',
{media: '../../media/', toolbox: document.
getElementById('toolbox'), trashcan: true});
```

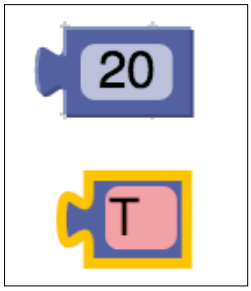
Blockly im Einsatz

Neben der Projektseite von Blockly gibt es noch weitere Seiten, auf denen Blockly produktiv eingesetzt wird.

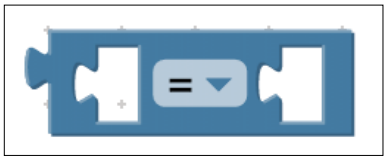
Ähnlich wie Scratch wird auch Blockly im Bereich der Spiele eingesetzt, um damit Programmierkenntnisse zu vermitteln. Eine sehr unterhaltsame Seite ist Blockly Games. Auf dieser Seite befinden sich Spiele, die mit Blockly umgesetzt wurden. Wer selbst Spiele mit Blockly erstellen möchte, sollte sich die Seiten Gameblox und Gamefroot ansehen. Auch LEGO Mindstorms lässt sich mit Blockly programmieren. Hier ist die Webseite zu Open Roberta die passende Anlaufstelle. Blockly kann nicht nur zur Vermittlung von Programmierkenntnissen eingesetzt werden, sondern auch, um Anwendern ohne Programmierkenntnisse die Möglichkeit zur Programmierung zu geben. Im Bereich von Internet of Things macht dies die Seite Scriptr;. Dort werden Webservices mittels Blockly erstellt.



Rückgaben lassen sich bei einigen Blöcken per Pulldown-Menü festlegen; im Beispiel kann zwischen *true* und *false* gewählt werden (Bild 6)

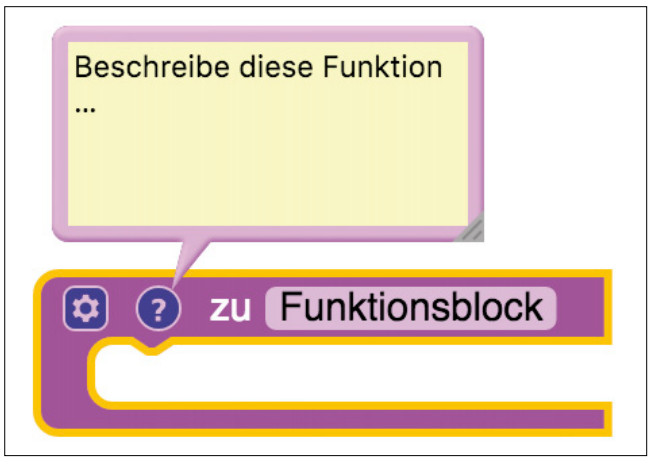


Der Rückgabewert wird per Eingabe festgelegt. Die Eingabe wird direkt validiert (Bild 7)

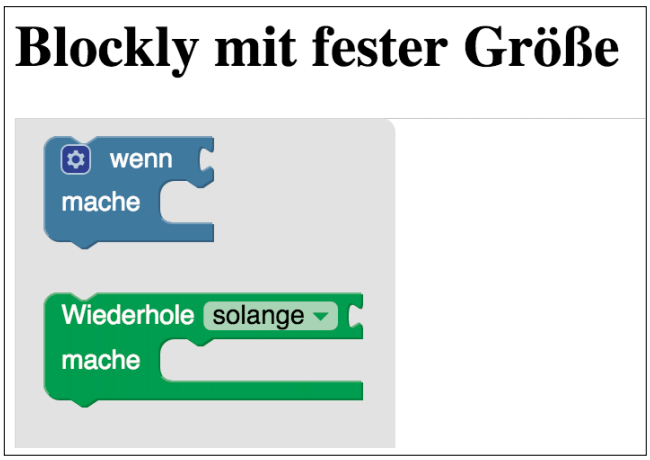


Rückgabewerte werden in einen öffnenden Verbinder eingeklinkt wie ein Puzzlestück (Bild 8)

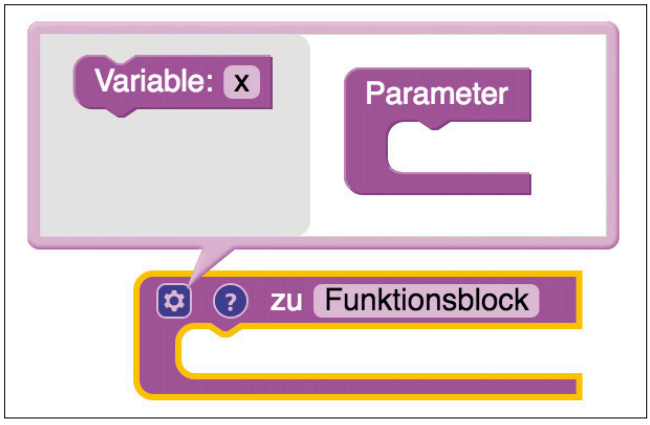
Block mit Möglichkeit zur Eingabe eines Kommentars und Auswahl von weiteren Varianten (Bild 9)



Das Kommentarfeld kann eine beliebige Größe haben und lässt sich durch Klick auf das Fragezeichen ausblenden (Bild 10)



Toolbox mit zwei Elementen. Die Reihenfolge ist die gleiche wie in der Definition (Bild 13)



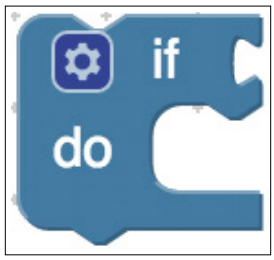
Ein Block kann in zusätzlichen Varianten auftreten. Über das Zahnrad werden diese Varianten angezeigt und lassen sich per Klick auswählen (Bild 11)

Toolbox mit zwei Kategorien: Die Kategorie *Bedingungen* ist in der Abbildung selektiert (Bild 14)

Eine Kategorie wird über das Element *category* eingeleitet. Der Parameter *name* definiert den Namen, der angezeigt wird. Im folgenden Beispiel werden die beiden Kategorien *Bedingungen* und *Logik* definiert. Das Ergebnis ist in Bild 14 dargestellt.

Kategorien können auch verschachtelt werden. Eine Kategorie kann sowohl Unterkategorien als auch einzelne Blöcke enthalten. Standardmäßig wird beim Start der Anzeige eine Kategorie zugeklappt dargestellt. Um die Kategorie bereits von Beginn an ausgeklappt anzuzeigen, muss der Parameter *expanded* auf *true* gesetzt werden:

```
<xml id="toolbox"
style="display: none">
  <category
  name="Bedingungen">
    <block type="controls_
    if"></block>
    <block type="controls_whileUntil"></block>
  </category>
  <category name="Logik">
    <block type="logic_compare"></block>
    <block type="logic_operation"></block>
    <block type="logic_boolean"></block>
  </category>
</xml>
```



Rundungen bilden einen eigenen Programmblock (Bild 12)

Um die Strukturierung durch Kategorien zu unterstützen, existieren Trennlinien. Diese werden über das Element *sep* definiert. Durch dessen Definition zwischen zwei Kategorien wird eine Trennlinie gezeichnet (Bild 15):

```
<xml id="toolbox" style="display: none">
  <category name="Bedingungen">
    <block type="controls_if"></block>
    <block type="controls_whileUntil"></block>
  </category>
  <sep></sep>
  <category name="Logik">
```

```
    <block type="logic_compare"></block>
    <block type="logic_operation"></block>
    <block type="logic_boolean"></block>
  </category>
</xml>
```

Zwei besondere Kategorien sind Funktionen und Variablen. Diese müssen nicht zwingend einzeln aufgelistet werden, sondern werden direkt vom System aufgefüllt. Hierfür muss der Parameter *custom* gesetzt werden: für Variablen auf den Wert *VARIABLE* und für Funktionen auf den Wert *PROCEDURE*:

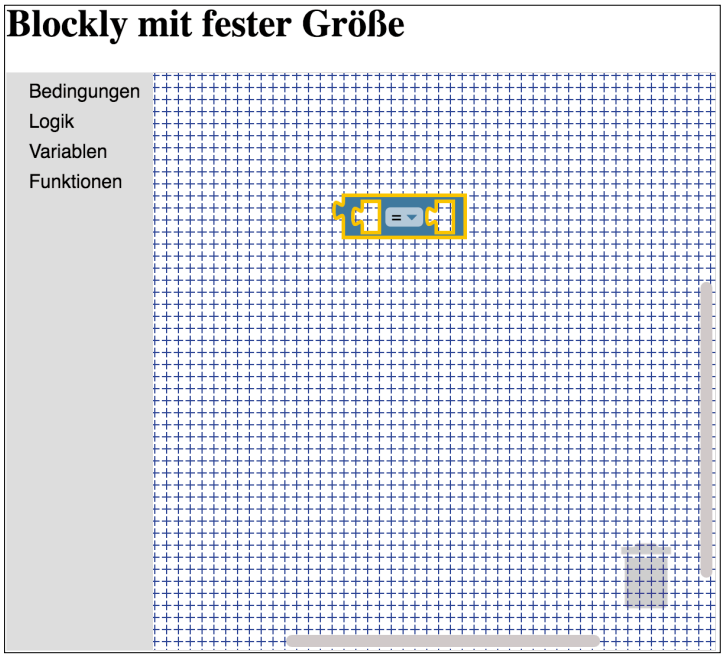
```
<xml id="toolbox" style="display: none">
  <category name="Bedingungen">
    <block type="controls_if"></block>
    <block type="controls_whileUntil"></block>
  </category>
  <category name="Logik">
    <block type="logic_compare"></block>
    <block type="logic_operation"></block>
    <block type="logic_boolean"></block>
  </category>
  <category name="Variablen" custom="VARIABLE">
  </category>
  <category name="Funktionen" custom="PROCEDURE">
  </category>
</xml>
```

Neben der Toolbox bietet der Arbeitsbereich noch weitere Funktionen wie zum Beispiel den Mülleimer. Per Rechtsklick im Arbeitsbereich wird ein Kontextmenü aktiviert (Bild 16). Sollte dabei ein Block im Arbeitsbereich selektiert sein, so wird ein erweitertes Kontextmenü angezeigt (Bild 17).

Blöcke sind innerhalb des Arbeitsbereichs frei positionierbar. Um auch noch in größeren Projekten den Überblick zu behalten, steht ein einblendbares Gitter (Grid) für den Arbeitsbereich zur Verfügung. Dies wird über den Parameter grid in *Blockly.inject* eingeblendet.

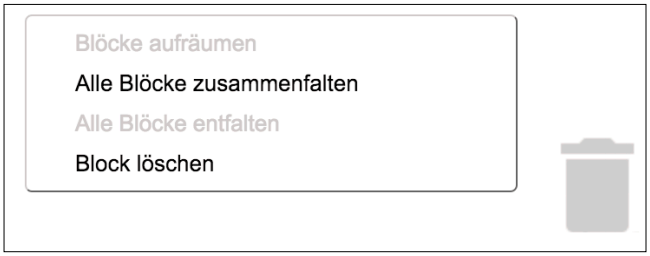
Das Gitter besteht dabei aus farbigen Kreuzen und hat insgesamt vier Parameter: *spacing, length, colour* (Achtung: britische Schreibweise) und *snap*. Über den Parameter *spacing* wird der Abstand der einzelnen Kreuze definiert. Bei einem Wert von *0* wird kein Gitter gezeichnet. Mit *length* wird die Länge der Kreuze angegeben, über *colour* die Farbe, und über *snap* kann angegeben werden, ob sich ein Block am Gitter automatisch ausrichten soll oder nicht.

Mit der folgenden Definition wird ein Gitter mit blauen Kreuzen, einer Kreuzgröße von *8* und einem Abstand von *10* definiert. Verwendete Blöcke richten sich bei dieser Definition am Gitter aus. Das Ergebnis ist in Bild 18 zu sehen:

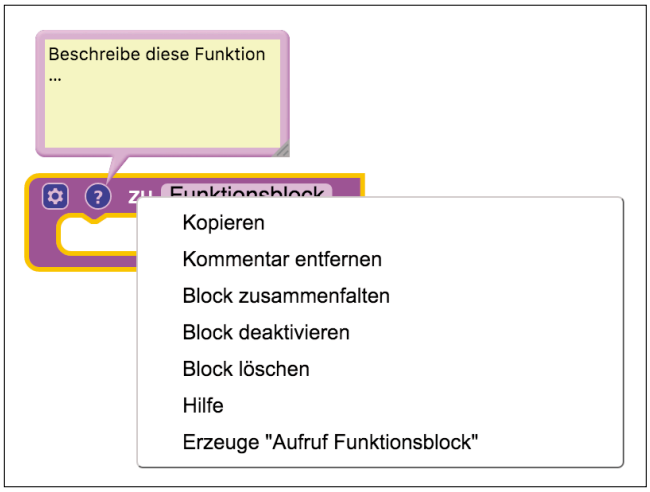


Definiertes Gitter, mit einem platzierten Block und Mülleimer (Bild 18)

Trennlinie nach der Kategorie *Bedingungen* (Bild 15)



Kontextmenü im Arbeitsbereich, ohne Selektion eines Blocks (Bild 16)



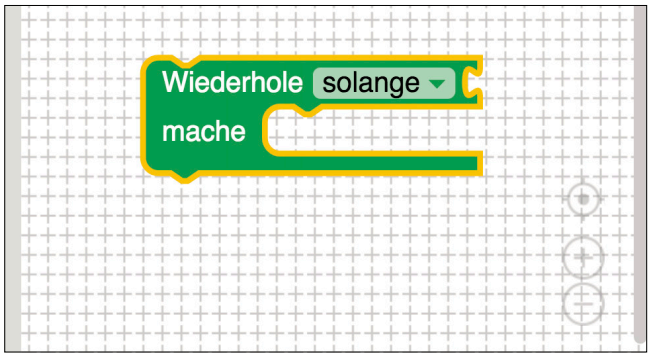
Kontextmenü im Arbeitsbereich, mit Selektion eines Blocks (Bild 17)

```
var workspace = Blockly.inject('blocklyDiv',
{
  media: '../../media/',
  toolbox: document.getElementById('toolbox'),
  grid:
    {spacing: 10,
      length: 8,
      colour: '#0000ff',
      snap: true}
    }
);
```

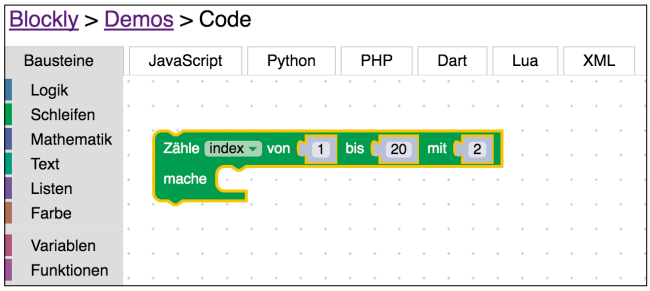
Über das Einblenden des Gitters hinaus kann der Arbeitsbereich auch skaliert werden. Hierfür steht für Blockly.inject der Parameter *zoom* bereit.

Beim Zoomen kann ein fester initialer Skalierungswert über den Parameter *startScale* angegeben werden, und zusätzlich kann man dem Anwender die Möglichkeit geben, die Skalierung über eingeblendete Schaltflächen zu beeinflussen. Die Schaltflächen werden über den Parameter *controls* eingeblendet, dabei werden drei Elemente dargestellt: größer, kleiner und Reset.

Das Reset setzt den Skalierungswert auf den Wert von *startScale* zurück und stellt den Arbeitsbereich mittig dar. Mit *wheel* kann das Mausrad für die Steuerung zugelassen werden. Über *maxScale* wird der maximale Skalierungsfaktor angeben und über *minScale* der minimale Skalierungsfaktor. Der Skalierungsschritt wird über *scaleSpeed* beeinflusst, dabei wird der aktuelle Wert mit *scaleSpeed* multipliziert oder dividiert. Dabei ist ein maximaler Skalierungswert von *3* angegeben, mit einem Skalierungsfaktor von *1.2* (Bild 19):



Skalierter Arbeitsbereich mit Grid und den drei Elementen für die Skalierung (Bild 19)



Der entsprechende Quellcode wird über die sechs Reiter eingeblendet. Das Blockly-Programm wird beim Klick auf den Reiter *Blocks* wieder angezeigt (Bild 20)

Listing 2: XML-Repräsentation

```
<xml xmlns="http://www.w3.org/1999/xhtml">
  <block type="controls_for" id=")3Xh,
  9JRZE-EWPe,MoAa" x="-462" y="-137">
    <field name="VAR">index</field>
    <value name="FROM">
      <shadow type="math_number"
      id="l8S%6F#Nk:C%??TOWJst">
        <field name="NUM">1</field>
      </shadow>
    </value>
    <value name="TO">
      <shadow type="math_number"
      id="uS@7!CSj^ZYxN3f+=X]Z">
        <field name="NUM">20</field>
      </shadow>
    </value>
    <value name="BY">
      <shadow type="math_number"
      id="W,Yj)=tQia%Vmh3p)[v1">
        <field name="NUM">2</field>
      </shadow>
    </value>
  </block>
</xml>
```

```
var workspace = Blockly.inject('blocklyDiv', {
  media: '../../media/',
  toolbox: document.getElementById('toolbox'),
  grid:
    {spacing: 10,
     length: 8,
     colour: '#ccc',
     snap: true},
  zoom:
    {controls: true,
     wheel: true,
     startScale: 1.0,
     maxScale: 3,
     minScale: 0.3,
     scaleSpeed: 1.2
    }
  }
);
```

Die vorhandenen Blöcke können bereits vorkonfiguriert in der Toolbox platziert werden. Dafür müssen die entsprechenden Werte beim Aufbau der Toolbox festgelegt werden. Dies kann man manuell tun, oder man kann dafür die vorhandene Demoanwendung Code *(demos/code/index.html)* nutzen. ▶

Diese Anwendung hat alle Blöcke integriert und kann aus dem zusammengestellten Blockly-Programm einen Quellcode erzeugen. Das gewünschte Format wird dabei jeweils über den Reiter am oberen Rand des Arbeitsbereichs ausgewählt (Bild 20).

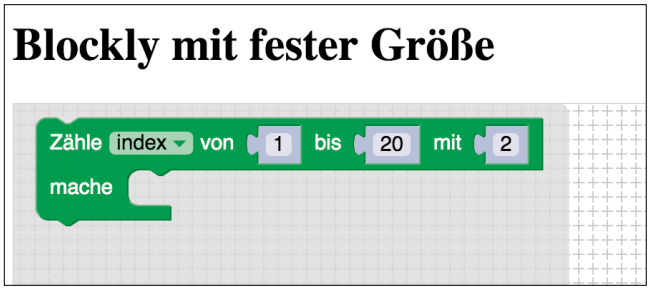
Unter den Formaten, die erzeugt werden können, ist auch XML. Dieses XML kann für die Definition in der Toolbox genutzt werden. Der in Bild 20 abgebildete Zähl-Block wurde mit einem Startwert von *1* und einem Endwert von *20* vorbelegt. Als Schrittweite wurde *2* festgelegt. Die entsprechende XML-Repräsentation ist in Listing 2 abgebildet.

Das XML enthält *id*-Felder; diese werden für die Definition innerhalb der Toolbox nicht benötigt und können beim Übertragen gelöscht werden. Die Koordinaten des Blocks (Felder x und y) können ebenfalls gelöscht werden. Die Toolbox mit übernommener Konfiguration ist in Listing 3 dargestellt. Das Ergebnis im Editor sehen Sie in Bild 21.

Eigene Blöcke definieren

Blockly bietet bereits eine große Anzahl an vorhandenen Blöcken. Für die Umsetzung eines Editors für eine spezielle Anwendung oder als Editor für eine spezielle DSL (Domain Specific Language) mögen die vorhandenen Blöcke nicht ausreichend sein. Für diesen Zweck lassen sich eigene Blöcke definieren.

Der einfachste Weg für die Erstellung von neuen Blöcken ist die Nutzung der Block Factory. Dabei handelt es sich um eine Webanwendung mit einem Blockly-Editor zur Definition neuer Blockly-Blöcke. Die Block Factory steht online zur Verfügung oder kann über *demos/blockfactory/index.html* auch lokal ausgeführt werden. Die Online-Block-Factory hat den Vorteil, dass definierte Blockdefinitionen abgespeichert und



Das vorkonfigurierte Element in der Toolbox (Bild 21)

für eine erneute Änderung wieder geöffnet werden können. Für die nun folgenden Beispiele wurde die Online-Variante gewählt.

Fenster der Block Factory

Das Fenster der Block Factory ist in insgesamt vier Bereiche aufgeteilt. Links befindet sich der Blockly-Editor für die Erstellung des Blocks. Im Bereich *Preview* rechts daneben wird der Block direkt dargestellt. Darunter im Bereich *Language Code* ist der JavaScript-Code enthalten, um den Block darzustellen. *Generator stub* schließlich enthält JavaScript-Code, um aus dem Block JavaScript-, Python-, PHP- oder Dart-Code zu erzeugen (Bild 22). Der Quellcode ist dabei nicht komplett, sondern enthält lediglich die Felder. Die auszuführende Logik muss natürlich selbst ergänzt werden.

Der Editor für die Erstellung des Blocks hat vier Kategorien: *Input, Field, Type* und *Colour.* Über *Colour* kann die Farbe eingestellt werden. Dabei sind in der Kategorie bereits einige vordefinierte Farben enthalten; diese können bei Klick auf das Eingabefeld nochmals angepasst werden (Bild 23). Um

Listing 3: Toolbox mit vorkonfiguriertem Zählblock

```
<xml id="toolbox" style="display: none">
  <block type="controls_for">
    <field name="VAR">index</field>
    <value name="FROM">
      <shadow type="math_number">
        <field name="NUM">1</field>
      </shadow>
    </value>
    <value name="TO">
      <shadow type="math_number">
        <field name="NUM">20</field>
      </shadow>
    </value>
    <value name="BY">
      <shadow type="math_number">
        <field name="NUM">2</field>
      </shadow>
    </value>
  </block>
</xml>
```

Listing 4: Definition der 3er-Multiplikation

```
Blockly.Blocks['webmobdev_block_multi'] = {
  init: function() {
    this.appendDummyInput()
        .appendField("Multiplikation:");
    this.appendValueInput("Value_1")
        .setCheck("Number")
        .appendField("Wert 1");
    this.appendValueInput("Value_2")
        .setCheck("Number")
        .appendField("Wert 2");
    this.appendValueInput("Value_3")
        .setCheck("Number")
        .appendField("Wert 3");
    this.setInputsInline(true);
    this.setOutput(true, "Number");
    this.setColour(210);
    this.setTooltip('');
    this.setHelpUrl('http://www.example.com/');
  }
};
```

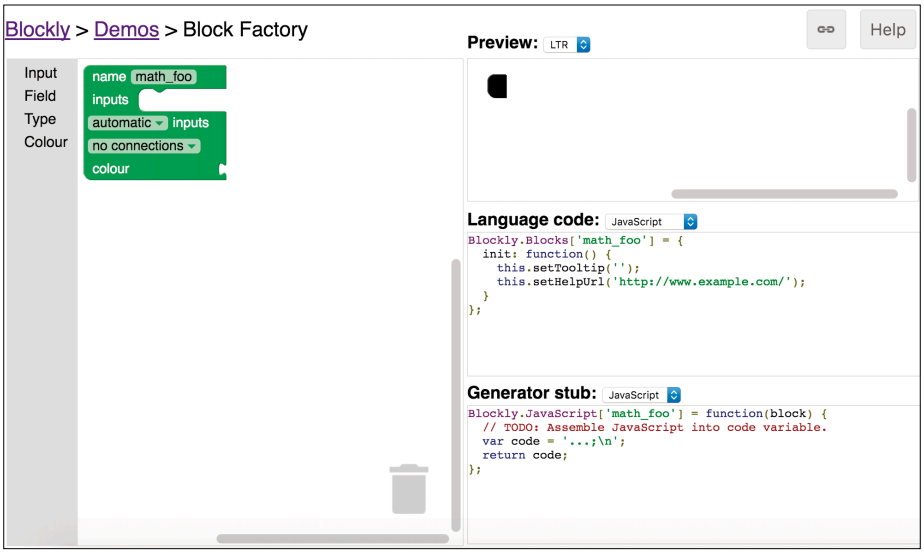
den Block später verwenden zu können, sollte im Feld *name* ein aussagekräftiger und vor allem eindeutiger Name vergeben werden.

Um die Funktionalitäten der Block Factory zu zeigen, werden zwei Beispiele umgesetzt. Zuerst soll ein Block für die Multiplikation dreier Werte erstellt werden. Der Name des Blocks soll *webmobdev_multiplikation* lauten; dieser Name wird im Feld *name* eingetragen.

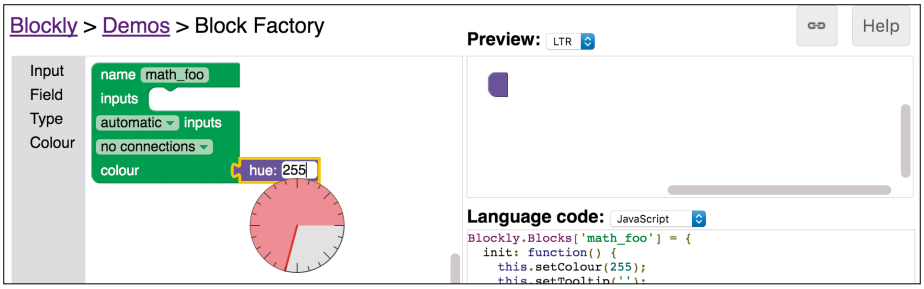
Der Block benötigt drei Eingabewerte, die im Bereich *input* platziert werden. Hierfür wird aus der Kategorie *Input* der Block *value input* ausgewählt und innerhalb von *inputs* platziert. Für die Identifizierung im Code wird das Feld mit *Value_1* benannt.

Um das Feld auch an der Oberfläche eindeutig zuordnen zu können, wird noch eine Beschriftung hinzugefügt. Hierfür wird aus der Kategorie *Field* das Feld *text* ausgewählt und im Bereich *fields* von *Value_1* eingehängt. Als Wert für das Feld wird Wert *1* eingegeben.

Bei einem Eingabefeld kann zusätzlich der Typ des Eingabewerts festgelegt werden. Da es sich bei dem Feld um eine Multiplikation handeln soll, können wir Texteingaben ausschließen und legen den Eingabewert als numerisch fest. Hierfür wählen wir aus der Kategorie *Field* das Feld *Number* und verbinden es mit dem Feld *type* von *Value_1*. Das Ergebnis der Multiplikation ist auch eine Zahl, und somit können wir den Ausgabewert über *output type* auch als *Number* festlegen. Insgesamt werden drei Eingabefelder benötigt, eines davon ist schon platziert. Die restlichen zwei Felder müssen nicht mehr aus der Toolbox zusammengestellt werden, sondern können aus dem bereits vorhandenen Element erstellt werden, entweder durch Anklicken des Elements und Auswahl von *Duplicate* aus dem Kontextmenü oder über die Tastatur mit [Strg C] und [Strg V].



Die Block Factory hat auch eine Preview auf den gestalteten Block (Bild 22)



Anpassung der Farbe erfolgt über einen Wert von *0* bis *360* (Bild 23)

Darstellung der Eingabefelder

Die beiden kopierten Teilblöcke werden unterhalb des bereits eingehängten Blocks platziert. Nun werden noch die internen Namen auf *Value_2* und *Value_3* geändert und die sichtbaren Bezeichnungen auf *Wert 2* und *Wert 3*. Die Darstellung der Eingabefelder kann auf zwei Arten erfolgen: als externe Verbindung oder als interne Verbindung. Die Art der Verbindung wird über das Feld *inputs* festgelegt. Für dieses Beispiel wählen wir *inline* aus.

Nun fehlt noch die Verbindung, um den Block mit anderen Blöcken zu verbinden. Hier wählen wir *left output*. Zuletzt soll der Block noch durch eine Überschrift beschrieben werden. Hierfür wird als erster Block, über dem ersten Eingabefeld, ein *dummy input* aus der Kategorie *Input* hinzugefügt und mit einem Feld *text* versehen. Der Wert für das Feld soll *Multiplikation*: sein.

Um den Block in einem Blockly-Editor zu verwenden, muss der erzeugte JavaScript-Code kopiert und in die eigene Webanwendung integriert werden. Für die Darstellung ist der Code im Feld *Language Code* verantwortlich (Listing 4).

Komplexer Block

Zum Abschluss soll nun noch ein komplexerer Block erstellt werden: *If-then-else*. Der Name des Blocks soll *webmobdev_block_if_else* lauten. Für den *if*- und *else*-Block wird ein jeweils ein Block *statement input* aus der Kategorie *Input* eingefügt.

Dieses Feld kann mehrere Blöcke aufnehmen. Die beiden Blöcke werden noch entsprechend benannt: mit *if_clause* und *else_clause*. Der Start des Blocks bildet die *if*-Abfrage. Diese wird innerhalb eines *dummy input* aufgebaut. Der Wert, der überprüft werden soll, wird als Variable angelegt; diese wird aus der Kategorie *Field* eingefügt und mit *item* benannt.

Für den Vergleich wird ein Dropdown mit den Werten *true* und *false* eingefügt. Das entsprechende Feld findet sich in der Kategorie *Field* und hat den Namen *dropdown*. Für die ▶

Verbindung mit anderen Blöcken wird anschließend noch *top+bottom connections* gesetzt (Listing 5).

Erzeugung von Quellcode

In den gezeigten Beispielen wurde nicht auf die Erzeugung von Quellcode eingegangen. Hierbei ist es wichtig, den Generatorcode zu implementieren. Im Verzeichnis *generators* liegen die Funktionen für die mitgelieferten Blöcke.

Soll ein Block mit der Block Factory erzeugt werden, so kann das Gerüst aus dem Bereich Generater *stub* genutzt werden. Die Logik muss aber selbst implementiert werden. Ein gutes Beispiel für die Erzeugung von Quellcode ist die Anwendung unter *demos/code/index.html*.

Links zum Thema

- Apache 2.0 License
  www.apache.org/licenses/LICENSE-2.0
- App Inventor
  http://appinventor.mit.edu/explore
- Block Factory
  https://blockly-demo.appspot.com/static/demos/blockfactory/index.html
- Blockly Block Wiki
  https://github.com/google/blockly/wiki
- Blockly Games
  https://blockly-games.appspot.com
- Blockly – GitHub
  https://github.com/google/blockly
- Blockly – Projektwebseite
  https://developers.google.com/blockly
- Blockly – Zip-Archiv
  https://github.com/google/blockly/archive/master.zip
- Gameblox
  https://gameblox.org
- Gamefroot
  http://make.gamefroot.com
- Open Roberta
  http://lab.open-roberta.org
- Quellcode zu den Beispielen
  https://github.com/mstaeuble/webmobdev-blockly
- Scratch
  https://scratch.mit.edu
- Scriptr;
  https://www.scriptr.io
- Snap!
  http://snap.berkeley.edu
- StarLogo
  http://education.mit.edu/portfolio_page/starlogo-tng

Listing 5: Defintion von if-then-else

```
Blockly.Blocks['webmobdev_block_if_else'] = {
  init: function() {
    this.appendDummyInput()
      .appendField("if")
      .appendField
      (new Blockly.FieldVariable("item"),
      "NAME")
      .appendField(new Blockly.FieldDropdown
      ([["is true", "v_TRUE"], ["is false",
      "v_false"]]), "bool_values")
      .appendField("else");
    this.appendStatementInput("if_clause")
      .setCheck(null);
    this.appendDummyInput()
      .appendField("else");
    this.appendStatementInput("else_clause")
      .setCheck(null);
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour(65);
    this.setTooltip('');
    this.setHelpUrl('http://www.example.com/');
  }
};
```

Fazit

Wer Anwendern eine Schnittstelle zur eigenen Anwendung anbieten möchte, um zum Beispiel Konfigurationen anzupassen oder gar in den Programmfluss einzugreifen, sollte über die Verwendung von Blockly nachdenken. Dem Anwender steht damit ein leicht zu erlernender Editor zur Verfügung, der auch noch fehlerfreien Code erzeugt.

Die Integration in die eigene Webanwendung ist sehr einfach. Insbesondere die Möglichkeit, neue Blöcke mit der Block Factory grafisch zu erzeugen, macht es dem Einsteiger noch leichter.

Wem die Funktionalität der Block Factory nicht ausreicht, der kann auf das funktionskräftige API zurückgreifen. Wobei ich hier vor allem die vielen Beispiele empfehlen möchte – diese gleichen die teilweise fehlende Dokumentation wieder aus. Insgesamt ist Blockly eine tolle Bibliothek für Webentwickler. ■

Dr. Markus Stäuble
ist Informatiker, Conference Chair der Developer Week und Programmleiter Make beim Franzis Verlag. Neben Programmiersprachen beschäftigt er sich mit dem Thema Mobile und hat zu dessen Auswirkungen auf die Arbeitswelt promoviert.
@staeuble

Jetzt kostenlos testen!

Das Fachmagazin für IT-Entscheider

2 Ausgaben kostenlos testen. Mit exklusivem Zugang zu unseren Digitalausgaben. Business-Newsletter inklusive.

www.com-magazin.de/gratis

RASPBERRY PI 3

Raspberry Pi zur Vierten

Die vierte Auflage des Raspberry bringt Verbesserungen im Bereich der Rechenleistung und ein integriertes WLAN-/Bluetooth-Modul.

Der am 29. Februar erstmals vorgestellte Raspberry Pi 3 wurde von der Presse angesichts der gesteigerten Konkurrenzsituation nicht mehr so enthusiastisch aufgenommen wie die Vorgänger. Trotzdem sollte man den Neuling nicht aus den Augen verlieren. Neben dem neuen Hauptprozessor gibt es nun auch ein integriertes Kombinationsfunkmodul, das per WLAN, Bluetooth und Bluetooth LE Kontakt zu externer Hardware aufnimmt.

Wenn man die beiden Planaren direkt nebeneinanderlegt (Bild 1), fällt auf, dass sich aus mechanischer Sicht nichts geändert hat. Gehäuse lassen sich weiterverwenden, wenn die geänderte Position der Status-LEDs keine Probleme macht.

Der einzige ärgerliche Unterschied findet sich auf der Unterseite: Der MicroSD-Slot basiert nun nicht mehr auf einer Schiebemechanik: Speicherkarten werden nun mit mechanischer Kraft in den Slot geschoben und dort per Reibschluss gehalten. Wie sich diese Kosteneinsparung auf die praktische Zuverlässigkeit des Prozessrechners auswirken wird, muss sich im Lauf der Zeit zeigen.

Der Prozessor des Neulings ist 64-Bit-fähig. Die Foundation nutzt diese Umstellung zum Erzwingen des Wechsels auf die neue Raspbian-Version, die auf Debian Jessie basiert. Wer – versehentlich oder absichtlich – eine Karte mit einem älteren Betriebssystem in einen Raspberry Pi 3 schiebt, bekommt nur einen Regenbogenbildschirm zu sehen. Andere Nutzer berichten von Blinksignalen der grünen LED – klar ist nur, dass sich auf Wheezy basierende Versionen von Raspbian auf einem Raspberry Pi 3 nicht starten lassen.

Kein einfacher Umstieg

Zur Behebung dieses Problems genügt es, das unter https://www.raspberrypi.org/downloads/raspbian bereitstehende Image herunterzuladen. Die Raspberry Pi Foundation bietet dort neben einer Lite-Version auch eine Vollversion an, die mehr Programme enthält.

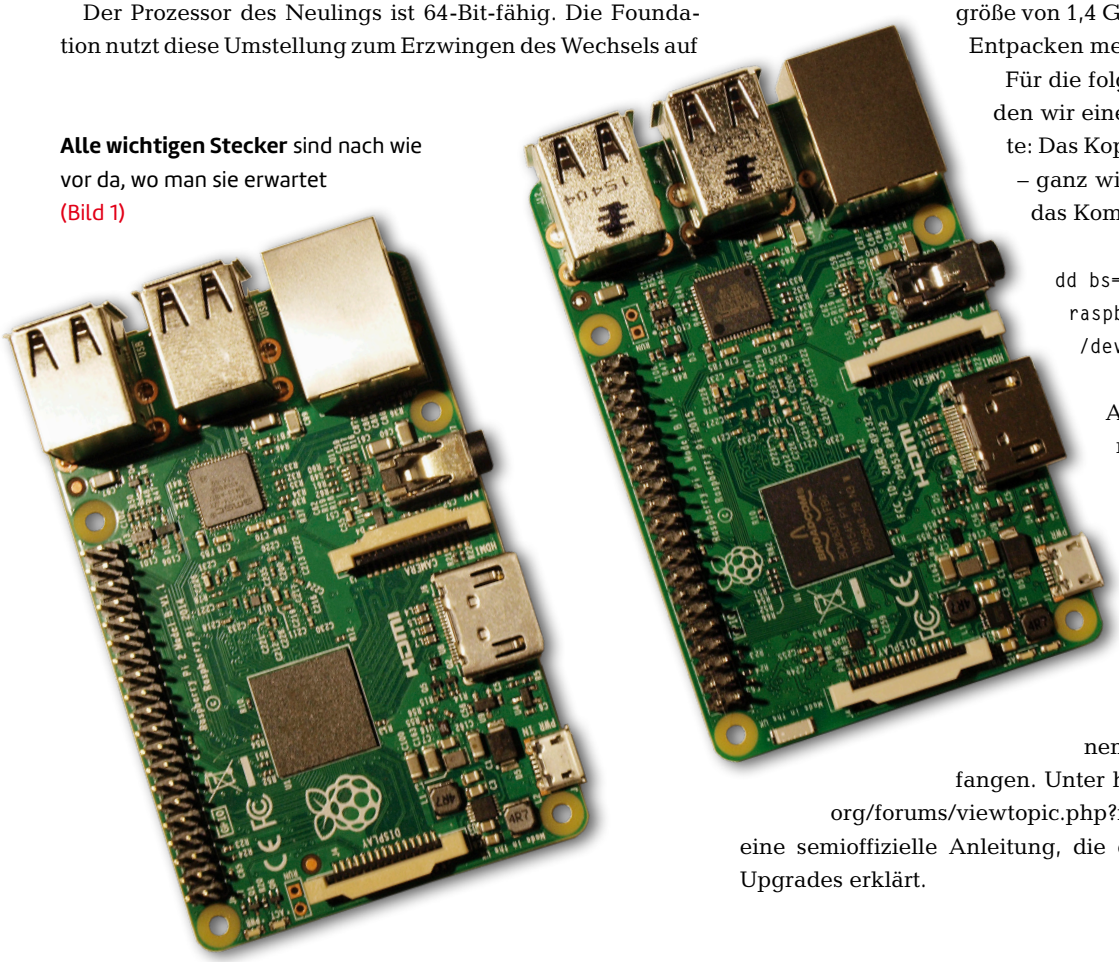
Die unterschiedlichen Datei- und Downloadgrößen sehen so aus: Die Version 2016-03-18-raspbian-jessie-lite.zip hat beispielsweise eine Downloadgröße von 298,2 MByte (nach dem Entpacken 1,4 GByte), während die Version 2016-03-18-raspbian-jessie.zip eine Downloadgröße von 1,4 GByte aufweist (nach dem Entpacken mehr als 4 GByte).

Für die folgenden Schritte verwenden wir eine 8 GByte große SD-Karte: Das Kopieren des Images erfolgt – ganz wie in UNIX üblich – über das Kommando

```
dd bs=4M if=2016-03-18-
 raspbian-jessie.img of=
 /dev/sdd
```

Achten Sie lediglich darauf, dass Sie nicht versehentlich ein anderes Laufwerk überschreiben.

Wer mit Wheezy-basierten Images arbeitet, soll nach dem Willen der Foundation mit einem frischen Image neu anfangen. Unter https://www.raspberrypi.org/forums/viewtopic.php?f=66&t=121880 gibt es eine semioffizielle Anleitung, die die Durchführung eines Upgrades erklärt.



Alle wichtigen Stecker sind nach wie vor da, wo man sie erwartet (Bild 1)

Alternativ dazu bietet sich die Nutzung des Disks-Applets an, das den Installationsprozess grafisch vereinfacht. Klicken Sie die extrahierte IMG-Datei dazu rechts an und wählen Sie das betreffende Programm. Der Fehler zur Thematik des zu kleinen Images kann bedenkenlos ignoriert werden.

Angemerkt sei, dass es sich hierbei nicht um ein komplettes 64-Bit-Betriebssystem handelt. Die Raspberry Pi Foundation setzt stattdessen auf ein 32-bittiges Userland. Dies wurde von Eben Upton im Rahmen eines Statements folgendermaßen begründet: »At launch, we are using the same 32-bit Raspbian userland that we use on other Raspberry Pi devices; over the next few months we will investigate whether there is value in moving to 64-bit mode.«

Alles grafisch

Für den ersten Start sind diesmal sowohl Maus als auch Tastatur empfehlenswert: Auf Jessie basierende Versionen von Raspbian starten sofort in den grafischen Desktop durch. Mein Testexemplar zeigte an dieser Stelle charakteristisches Schaltreglerfiepen – die vom Vorgänger bekannte Lichtempfindlichkeit ist ebenfalls mit von der Partie, betrifft nun aber einen anderen Chip.

Die bisher unter *raspi-config* findbaren Konfigurationseinstellungen sind nun in einem grafischen Werkzeug dupliziert, das im Menü unter *Preferences* bereitsteht.

Achten Sie darauf, beim Setup die Tastatur richtig einzustellen und das Dateisystem auf die gesamte SD-Karte zu expandieren (Bild 2).

WLANs lassen sich über ein in der oberen rechten Bildschirmkante eingeblendetes Menü verbinden. Leider ist die Leistungsfähigkeit der Antenne eng begrenzt. Im Labor des Autors ließen sich Verbindungen zu WLANs nur mit Mühen aufbauen, die Beschaffung von DNS-Informationen per DHCP gelang während der Testphase nie. Da andere Nutzer von Erfolgen berichten, könnte es sich hier auch um einen Einzelfall handeln.

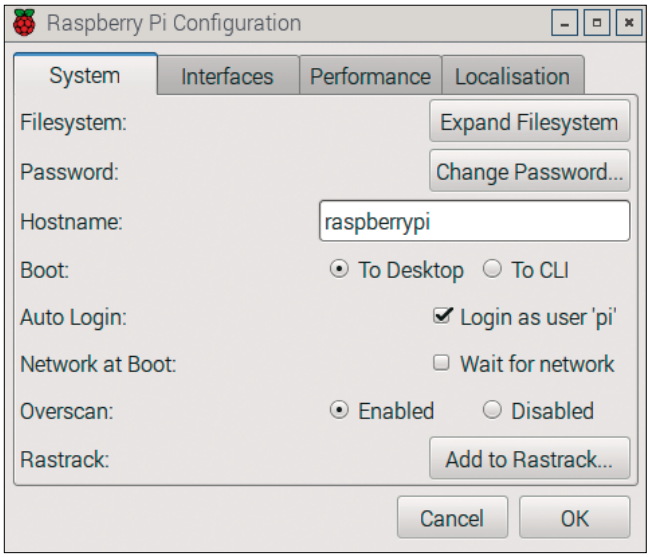
CPU im Blick

Der mit 1,2 GHz arbeitende Vierkernprozessor basiert nun auf der ARM-v8-Architektur. Die mit Abstand wichtigste Neuerung ist die Unterstützung von 64-bittigen Instruktionen. Schon aus Gründen der Vergleichbarkeit mit vorhergehenden Tests in der **web & mobile developer** wollen wir den ersten Test mit dem durch Eingabe von *sudo apt-get install sysbench* herunterladbaren Benchmarkprogramm durchführen.

Im Einzelkernbetrieb ergeben sich die in Tabelle 1 gezeigten Werte. Tatsächlich funktionierte übrigens hierbei ein- und dieselbe SD-Karte gleichermaßen in allen drei Varianten des Einplatinencomputers.

Da sowohl Raspberry Pi Zero als auch der Ur-RPI nur einen einkernigen Prozessor aufweisen, bringt die Durchführung eines Tests mit mehreren Kernen auf dieser Plattform nichts. Der Raspberry Pi 3 ist seinem vierkernigen Vorgänger nicht wirklich überlegen, was an der hohen Varianz liegt.

Interessanterweise kam es unter Jessi – wohl wegen des von Haus aus aktivierten GUI – zu wesentlich stärkeren Varianzen: insbesondere im Einzelkerntest gab es immer wieder Situationen, in denen ein Durchlauf die Form eines massiven Ausreißers annahm.



Die Konfiguration via *raspi-config* ist grafikfähig geworden (Bild 2)

Im Bereich der GPU hingegen gibt es keine interessanten Neuigkeiten. Auch im Raspberry Pi 3 findet sich die hinreichend bekannte VideoCore-GPU, die nun aber statt mit 250 MHz mit bis zu 400 MHz takten darf. Ihre Leistung reicht für das Decodieren von Full-Hd-Videos mal besser, mal schlechter aus. Andere Einplatinencomputer bieten ihrer Nutzerschaft wesentlich mehr Grafikleistung für dasselbe Geld.

Während die Foundation für den Raspberry Pi 2 die Nutzung eines 2-Ampere-Netzteils empfahl, ist beim Raspberry Pi 3 sogar von 2,5 A die Rede. Das zusätzliche Strombudget soll neben der Speisung von WLAN und Bluetooth auch bei der Versorgung leistungsstarker USB-Geräte hilfreich sein.

Da die Arbeit mit den von Telefon und Co. bekannten Ladegeräten immer Unsicherheiten birgt – der Widerstand von MicroUSB-Kabeln variiert von Hersteller zu Hersteller –, erledigte der Autor dieser Zeilen die Stromversorgung durch ein Hochleistungsnetzteil vom Typ Agilent 6624A. Der zur Lieferung von bis zu 5 Ampere befähigte Kanal 1 wurde über eine Kabelgruppe direkt mit den GPIO-Ports verbunden, um so die 5-V-Spannungsschiene direkt zu versorgen.

Mehr Energie

Als erster Test wurde das Netzgerät zur Berschränkung des Stromes auf 0,4 A angewiesen: Der Raspberry Pi begann zwar zu booten, schaltete sich aber beim Anwerfen des GUI reproduzierbar aufgrund von Energiemangel ab. Mit 0,5 A trat das Problem ebenfalls auf; ab 0,6 A war trotz angeschlossener Maus und Tastatur ein normaler Bootvorgang beobachtbar. Der Raspberry Pi 2 erwies sich hier als sparsamer – er kam auch schon mit 0,5 A ins GUI. Der Raspberry Pi 1 war noch genügsamer und startete auch mit nur maximal 400 mA hoch.

Das Versorgen einer externen Festplatte ist ein klassischer Quältest – auch bei voll deaktivierter Strombegrenzung (der Raspberry Pi durfte 5 A ziehen) gelang es nicht, die Platte ▶

Benchmark-Ergebnisse

Raspberry Pi 1		Raspberry Pi 2		Raspberry Pi 3	
Test execution summary					
total time:	513.3189s	total time:	291.0999s	total time:	212.6383s
total number of events:	10000	total number of events:	10000	total number of events:	10000
total time taken by event execution:	2052.7249	total time taken by event execution:	291.0849	total time taken by event execution:	212.6281
per-request statistics					
min:	166.09ms	min:	29.02ms	min:	18.29ms
avg:	205.27ms	avg:	29.11ms	avg:	21.26ms
max:	376.23ms	max:	44.07ms	max:	44.61ms
approx. 95 percentile:	231.08ms	approx. 95 percentile:	29.11ms	approx. 95 percentile:	36.78ms
Threads fairness					
events (avg/stddev):	2500.0000/0.71	events (avg/stddev):	10000.0000/0.00	events (avg/stddev):	10000.0000/0.00
execution time (avg/stddev):	513.1812/0.04	execution time (avg/stddev):	291.0849/0.00	execution time (avg/stddev):	212.6281/0.00

eines EliteBook 2450p in einem externen Gehäuse anzuwerfen. Interessanterweise lag der Strombedarf dabei nie über 0,5 A. Zur Kompensation von Spannungsabfällen in der Schaltung wurde die Spannung durch die Kelvinleitungen automatisch erhöht – und siehe da, die Festplatte lief bei einem Gesamtstromverbrauch von rund 0,8 A an.

Externer Hub für USB

Daraus folgt, dass der Raspberry Pi 3 – zumindest in der Theorie – in der Lage ist, als Standalone-NAS samt angeschlossener Festplatte zu agieren. Angesichts der Ansprüche an die Kabelqualität ist ein externer Hub nach Ansicht des Autors trotzdem vorzuziehen.

Der Hauptprozessor zeichnet sich sehr deutlich durch hohen Stromverbrauch aus. Bei Nutzung aller vier Kerne stieg der Stromverbrauch von 290 mA im Leerlauf auf bis zu 880 mA, bei einem Kern genehmigte sich der Raspberry Pi 3 immerhin rund 450 mA. Der Raspberry Pi 2 kommt bei voller Auslastung eines Kerns mit 320 mA aus, mit vier Kernen zieht er 445 mA aus dem Agilent 6624A. In dieser Disziplin ist er dem Raspberry Pi 1 überlegen, dessen einzelner Kern unter Last 380 mA verbrennt.

Nicht außer Acht gelassen werden darf, dass der Raspberry Pi 3 thermisch wesentlich knapper kalkuliert ist. Der angelsächsische Nachrichtendienst Phoronix berichtet bei Benchmarks von Maximaltemperaturen im Bereich um 100 Grad Celsius: Wer mehrere dieser Computerchen auf engem Raum kombiniert, sollte zu Kühlkörper und Lüfter greifen.

Was passiert mit den Vorgängern?

Ob des geringen Preises und der hohen Verfügbarkeit hielten Prozessrechner aus dem Hause Upton in diversen Systemen Einzug: Die Produkte dienen sowohl als MSR- als auch als Anzeigerechner. Im Embedded-Bereich ist Konstanz wegen der immens langen Produktzyklen besonders wichtig. Wer seine Kunden durch Abkündigung von Angeboten verärgert, kann sein Unternehmen binnen kurzer Zeit schließen.

Eben Upton ist sich dieser Situation bewusst. Die Raspberry Pi Foundation wird sowohl den Raspberry Pi 1 als auch den Raspberry Pi 2 auf absehbare Zeit weiterproduzieren. Laut einer offiziellen Verlautbarung der Foundation werden die Modelle B1+ und B2 weiterhin für 25 beziehungsweise 35 US-Dollar erhältlich sein, solange ausreichend Interesse vonseiten der Kundschaft besteht. Das gilt auch für das Modell A1, dem in nicht allzu ferner Zeit eine schnellere Variante mit der im B3 verwendeten CPU folgen wird.

Fazit

Die immer stärker werdende Konkurrenz beginnt damit, Luft aus dem übergroßen Ego der Truppe um Upton abzulassen. Für Entwickler ist dies erfreulich: Die vierte Generation des Raspberry Pi bietet hilfreiche Neuerungen, die die Hardwarekomplexität reduzieren. Das Fiepen des Schaltreglers ist allerdings nervtötend, zudem könnte die enge thermische Auslegung im Clusterbetrieb für Ärger sorgen. ■

Tam Hanna
ist Autor, Trainer und Berater mit den Schwerpunkten Webentwicklung und Webtechnologien. Es lebt in der Slowakei und leitet dort die Firma Tamoggemon Holding k.s. Er bloggt sporadisch unter:
www.tamoggemon.com

Impressum

Verlag
Neue Mediengesellschaft Ulm mbH
Bayerstraße 16a,
80335 München
Telefon: (089) 741 17-0,
Fax: (089) 741 17-101
(ist zugleich Anschrift aller Verantwortlichen)

Herausgeber
Dr. Günter Götz

Chefredakteur
Max Bold
– verantwortlich für den redaktionellen Teil –
E-Mail: redaktion@webundmobile.de

Schlussredaktion
Ernst Altmannshofer

Redaktionelle Mitarbeit
Philip Ackermann, Daniel Basler, Christian Bleske, Tam Hanna, Anna Kobylinska, Bernhard Lauer, Patrick Lobacher, Filipe Martins, Florence Maurice, Daniel Reyhe, Michael Rohrlich, Thomas Rudin, Walter Saumweber, Michael Schams, Jochen Schmidt, Markus Schraudolph, Katharina Sckommodau, Thomas Sillmann, Frank Simon, Markus Stäuble

Art Directorin
Maria-Luise Sailer

Grafik & Bildredaktion
Alfred Agatz, Dagmar Breitenbauch, Hedi Hefele, Manuela Keller, Simone Köhnke, Cornelia Pflanzer, Petra Reichenspurner, Ilka Rüther, Christian Schumacher, Mathias Vietmeier

Anzeigenberatung
Jens Schmidtmann, Anzeigenleiter
Klaus Ahlering, Senior Sales Manager
Telefon: (089) 741 17-125
Fax: (089) 741 17-269
E-Mail Anzeigenberatung: sales@nmg.de

Anzeigendisposition
Dr. Jürgen Bossmann
Telefon: (089) 741 17-281
Fax: (089) 741 17-269
E-Mail: sales@nmg.de

Leitung Herstellung/Vertrieb
Thomas Heydn
Telefon: (089) 741 17-111
E-Mail: thomas.heydn@nmg.de

Leserservice
Hotline: (089) 741 17-205
Fax: (089) 741 17-101
E-Mail: leserservice@nmg.de

Kooperationen
Denis Motzko
Telefon: (089) 741 17-116
E-Mail: kooperationen@nmg.de

Druck
L.N. Schaffrath Druckmedien
Marktweg 42–50
47608 Geldern

CD-Produktion
Stroemung GmbH

Vertrieb
Axel Springer Vertriebsservice GmbH
Objektvertriebsleitung Lothar Kosbü
Süderstraße 77
20097 Hamburg
Telefon: (040) 34724857

Bezugspreise
web & mobile developer ist das Profi-Magazin für Web- und Mobile-Entwickler und erscheint zwölfmal im Jahr. Der Bezugszeitraum für Abonnenten ist jeweils ein Jahr. Der Bezugspreis im Abonnement beträgt 76,20 Euro inklusive Versand und Mehrwertsteuer im Halbjahr, der Preis für ein Einzelheft 14,95 Euro. Der Jahresbezugspreis beträgt damit 152,40 Euro.

In Österreich sowie im übrigen Ausland kostet das Abonnement 83,70 Euro im Halbjahr. Der Jahresbezugspreis beträgt somit 167,40 Euro. In der Schweiz kostet das Abonnement 152,00 Franken im Halbjahr. Der Jahresbezugspreis in der Schweiz beträgt 304,00 Franken.

Das Abonnement verlängert sich automatisch um ein Jahr, wenn es nicht sechs Wochen vor Ablauf der Bezugszeit schriftlich beim Verlag gekündigt wird.

Studenten erhalten bei Vorlage eines Nachweises einen Rabatt von 50 Prozent.

ISSN: 2194-4105

ZUSAMMENSPIEL VON CMS UND E-COMMERCE-SYSTEMEN

Content- und Shop-Welt

Mehr als je zuvor müssen E-Commerce-Händler ihre Produkte heute im Kontext relevanter und informativer Inhalte präsentieren.

Ein Shop beziehungsweise ein dort angebotenes Produkt erscheint nur dann ganz oben in den Suchmaschinen, wenn das Content-Umfeld stimmt. Viele Unternehmen betrachten die Themen Content Management einerseits und E-Commerce andererseits jedoch immer noch als zwei ganz unterschiedliche Bereiche.

Das liegt oft nicht zuletzt auch daran, dass die zugrunde liegenden IT-Systeme nicht optimal miteinander integriert sind. Ein reibungsloses Zusammenspiel der Content- und der Shop-Welt ist im E-Commerce allerdings zunehmend erfolgsentscheidend. Wie dies gelingen kann, zeigen die folgenden zehn Hinweise:

Projektteam organisieren

In den meisten Unternehmen ist das Marketing für Website, Content und CMS zuständig, während der E-Commerce-Bereich meist Vertrieb/Sales zugeordnet ist. Diese Trennung sollten Sie organisatorisch überwinden, um Ihre CMS- und E-Commerce-Welt erfolgreich miteinander zu verknüpfen. Planen Sie ein entsprechendes Projekt, müssen Verantwortliche aus beiden Bereichen gleichberechtigte Mitglieder des Projektteams sein.

Es gibt Systeme auf dem Markt, die von ihren Herstellern als Alleskönner positioniert werden. Empfehlenswerter ist in der Regel jedoch der Best-of-Breed-Ansatz: Wählen Sie die Systeme aus, die unter Berücksichtigung der eigenen Anforderungen die besten sind.

Ein spezialisiertes System deckt die jeweiligen Anforderungen in einem bestimmten Bereich am besten ab. Im Hinblick auf den großen Trend Content-Commerce – also dem Anliegen, Kunden durch hochwertigen Content in den eigenen Shop zu holen und Produkte im Rahmen serviceorientierter, informativer und unterhaltsamer Themenwelten zu präsentieren – ist ein hochprofessionelles CMS daher unverzicht-

bar. Als spezialisiertes System bietet es Ihnen mehr Möglichkeiten, Content-Einheiten zu pflegen oder Drittanbietern wie beispielsweise Marktplätzen bestimmten Content zur Verfügung zu stellen.

Kompatibilität prüfen

Idealerweise verfügen Content-Management-Systeme und E-Commerce-System bereits über passende Konnektoren. Wollen Sie beide Systeme neu einführen, sollten Sie dies bei der Auswahl als wichtiges Kriterium beachten. Ist bereits ein System vorhanden, etwa das CMS, lohnt es sich, zunächst festzustellen, zu welchen E-Commerce-Lösungen Konnektoren angeboten werden.

Diese Lösungen sollten Sie sich dann näher anschauen. Grundsätzlich gilt: Überprüfen Sie die CMS- und E-Commerce-Systeme, die für Sie in Frage kommen, auf ihre Kompatibilität – sowohl miteinander als auch im Hinblick auf die bestehende Infrastruktur. Unterstützen die Systeme die Schnittstellen der vorhandenen Systemwelt? Ist eine Integration ohne hohen Programmieraufwand möglich?

Wenn Sie sich für Systeme entscheiden, die nicht über passende Konnektoren verfügen, kann es sinnvoll sein, eine Zusatzkomponente einzusetzen – beispielsweise ein Product Information Management (PIM) System oder einen Suchindex. Der Vorteil eines zwischengeschalteten dritten Systems ist die höhere Flexibilität: Produktdaten lassen sich so in der Regel schneller zwischen Content Management- und E-Commerce-System übertragen.

Frontend-System bestimmen

Angesichts der wachsenden Bedeutung von Content im E-Commerce rückt das CMS inzwischen zunehmend in den Vordergrund – schließlich werden rund um die Produkte immer mehr Inhalte angeboten. Das CMS ist von vornherein für die optimale Anzeige von Content ausgelegt und hat hier deshalb seine Stärken, auch was die Darstellung auf unterschiedlichen Endgeräten angeht.

Das Shop-System dient in diesem Fall dann unter anderem als weitere Datenquelle für das CMS. Es geht aber auch andersherum, indem die Shop-Lösung als führendes System definiert wird. Dann ist das CMS nur für die reine Content-Pflege verantwortlich. Die Entscheidung, welches System im Vordergrund steht, kann auch davon abhängig sein, ob Sie bereits über ein bestehendes System verfügen, das beibehalten und ergänzt werden soll.

Produktstrategie des Herstellers

Blicken Sie bei der Software-Auswahl nicht nur auf den Status quo oder in die Vergangenheit, sondern auch in die Zukunft. Prüfen Sie daher genau, wie die Roadmap des Herstellers aussieht, und analysieren Sie dessen künftige Produktstrategie. Wie gestaltet sich die Update-Strategie? Gibt es überhaupt regelmäßige Updates? Und passt der Update-Pfad zur Arbeitsweise im Unternehmen? Vermutlich möchten Sie nicht in zwei Jahren wieder ein komplett neues System einführen. Daher ist es unverzichtbar, bei einem Softwareprojekt über die aktuellen Anforderungen hinauszudenken.

Eine gute Planung erleichtert den Implementierungsprozess erheblich, ganz besonders, wenn viele Komponenten über Schnittstellen angebunden werden müssen. Gerade beim Thema Schnittstellen lauern zahlreiche Stolperfallen – achten Sie daher von Anfang an darauf, dass diese offen und kompatibel sind. Schnittmengen der Systeme und Datenmodelle sollten Sie ebenso vorab definieren wie den gesamten Release-Prozess.

Wenn Sie mit der Implementierung beginnen: Setzen Sie die Systeme logisch getrennt auf, zum Beispiel auf zwei unterschiedlichen Servern. Dadurch profitieren Sie von mehr Flexibilität und können die Systeme bei Bedarf unkompliziert weiter ausbauen. Zudem kann es sinnvoll sein, ein identisches Test-System aufzusetzen.

Ihr Shop kann nur erfolgreich performen, wenn jederzeit aktuelle Produktdaten auf allen relevanten Systemen zur Verfügung stehen. Für die Synchronisation dieser Daten gibt es unterschiedliche Lösungsansätze: zum einen den sogenannten Master-Slave-Ansatz, wobei von einem Hauptsystem aus die Daten im nachgelagerten System überschrieben werden. Zum anderen gibt es die Option, dass die Systeme gleichberechtigt sind und Daten permanent synchronisiert werden. Sie können aber auch ein PIM als führendes drittes System einsetzen und festlegen, dass sämtliche Daten nur dort aktualisiert werden.

Mobile-First-Strategie

Immer mehr Nutzer rufen Online-Inhalte immer häufiger über mobile Geräte auf. Überlegen Sie daher schon bei der Planung eines Online-Projekts, ob eine Mobile-First-Strategie sinnvoll ist, da dies sich bereits auf die Auswahl der Systeme auswirkt: Nicht nur die benötigten Content-Elemente unterscheiden sich erheblich voneinander, es ergeben sich auch unterschiedliche Anforderungen an die Pflegeprozesse. Ein CMS beispielsweise, das bei der Vorschau nicht verschiedene Ausgabekanäle gleichzeitig darstellen kann, wird dann schnell zum Bremsklotz.

Nehmen Sie bei solchen IT-Projekten das Know-how eines erfahrenen Implementierungspartners in Anspruch. Dieser kennt die Vor- und Nachteile unterschiedlicher Lösungen und Implementierungsoptionen, kann Sie bedarfsgerecht beraten und unterstützen. Ein qualifizierter Partner kann Ihnen dabei helfen, die richtigen Entscheidungen zu treffen. Und in fachmännische Beratung und Unterstützung zu investieren ist letztendlich oft günstiger, als Fehlinvestitionen in eine unpassende Software zu spät zu bemerken. ■

Thomas Rudin ist Senior Consultant bei der diva-e Digital Value Enterprise GmbH. **Daniel Reyhe** ist Senior Architect bei der diva-e Digital Value Enterprise GmbH.
www.diva-e.com

EUROPÄISCHE DATENSCHUTZGRUNDVERORDNUNG (DGSVO)

Datenschutz in der EU

Alles bleibt anders – die EU hat sich auf ein neues Datenschutzrecht geeinigt.

Die gute Nachricht vorweg: Das Datenschutzrecht wird nicht komplett umgekrempelt, viele Grundzüge bleiben erhalten. Allerdings gibt es durchaus praxisrelevante Änderungen, nicht zuletzt auch für Betreiber von Internetseiten.

Inkrafttreten 2018

Ab Mitte 2018 soll die europäische Datenschutzgrundverordnung (DGSVO) in Kraft treten (Bild 1). Sie wird dann unmittelbar in allen Mitgliedstaaten der Europäischen Union Gültigkeit erlangen, es bedarf keiner gesonderten Umsetzung in nationales Recht.

Die DGSVO wird die deutschen Gesetze, insbesondere das Bundesdatenschutzgesetz (BDSG), zwar nicht aufheben. Allerdings wird sie vorrangig anzuwenden sein, daher ist de facto mit einer Anpassung der deutschen Gesetzeslage zu rechnen. Ob und wann das geschieht und wie es konkret aussehen wird, bleibt abzuwarten.

Es sei vorweggeschickt, dass auch zukünftig die durch die Datenerhebung betroffene Person (Betroffener) sowie das datenerhebende Unternehmen (verantwortliche Stelle) geben wird – sowohl terminologisch als auch inhaltlich.

Fakt ist, dass die DGSVO nicht das bestehende Datenschutzrecht von Grund auf verändern wird. Fakt ist aber auch, dass es Änderungen geben wird, die Auswirkungen für alle Bereiche unternehmerischen Handelns bewirken werden. Dies gilt online wie offline, es sind also letztlich alle Unternehmen betroffen, nicht nur der Sektor E-Commerce.

Folgende Grundsätze des Datenschutzrechts bleiben erhalten beziehungsweise werden teilweise neu geregelt:

- Zweckbindung
- Datensparsamkeit
- Datenvermeidung
- Datenrichtigkeit
- Bestimmung der Löschfristen durch Unternehmen
- IT-Sicherheit
- Transparenz
- Rechenschaft

Was die essenziellen Grundsätze angeht, wird sich also nicht sonderlich viel ändern. Allerdings bleibt abzuwarten, wie viel Zündstoff in den Detailanpassungen tatsächlich steckt.

Einwilligung

Das grundlegende Prinzip, nämlich das Verbot mit Erlaubnisvorbehalt, bleibt ebenfalls erhalten. Dies besagt, dass personenbezogene Daten, wie bisher auch schon, nur verarbeitet werden dürfen, wenn die Einwilligung der betroffenen Person vorliegt oder eine gesetzliche Grundlage dafür besteht.

EUROPÄISCHE KOMMISSION

Brüssel, den 25.1.2012
KOM(2012) 11 endgültig

2012/0011 (COD)

Vorschlag für

VERORDNUNG DES EUROPÄISCHEN PARLAMENTS UND DES RATES

zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr (Datenschutz-Grundverordnung)

(Text von Bedeutung für den EWR)

{SEK(2012) 72 endgültig}
{SEK(2012) 73 endgültig}

Die Europäische Union arbeitet an einer neuen Datenschutzgrundverordnung (DGSVO) (Bild 1)

Die Anforderungen an eine korrekte Einwilligung sind (und bleiben) die folgenden:

- Freiwilligkeit: Der Betroffene darf »nicht vor vollendete Tatsachen« gestellt werden und es muss eine realistische Möglichkeit zum Widerruf seiner Einwilligung geben; insbesondere darf die Möglichkeit zum Vertragsschluss nicht von der Preisgabe solcher persönlicher Daten abhängig gemacht werden, die nicht zur Durchführung des Vertrages benötigt werden (sogenanntes Kopplungsverbot).
- Information: Der Betroffene muss vor Erteilung seiner Zustimmung über die verantwortliche Stelle, über die Art der Daten, über den Zweck der Erhebung und über alle sonstigen wichtigen Informationen aufgeklärt werden.
- Erkennbarkeit: Die Einwilligungserklärung kann prinzipiell auch zusammen mit anderen Erklärungen verbunden werden. Allerdings muss sie dann hervorgehoben dargestellt werden.
- Alter: Einwilligungen von Minderjährigen müssen durch die Erziehungsberechtigten genehmigt werden.
- Zeitpunkt: Die Einwilligungserklärung muss vor Beginn der Datenerhebung beziehungsweise -verwendung abgegeben werden.
- Schriftformerfordernis: Die Einwilligungserklärung muss grundsätzlich in schriftlicher Form erfolgen, kann jedoch ausnahmsweise auch, insbesondere im Rahmen von elektronischer Kommunikation (zum Beispiel bei Webshops), zum Beispiel wie E-Mail abgegeben werden.
- Widerrufsmöglichkeit: Die abgegebene Einwilligungserklärung muss widerrufbar sein. Auf diese Möglichkeit muss der Betroffene auch ausdrücklich hingewiesen werden.

Links zum Thema

- Video-Trainings des Autors
  www.video2brain.com/de/trainer/michael-rohrlich
- Blog des Autors zum Thema Online-Recht für Webmaster
  http://webmaster-onlinerecht.de
- Blog des Autors zum Verbraucherrecht online
  http://verbraucherrechte-online.de
- Weitergehende Informationen zum Thema E-Commerce
  http://rechtssicher.info

- Besonderheit: Bei der Verarbeitung besonderer personenbezogener Daten (zum Beispiel Religion, Gewerkschaftszugehörigkeit et cetera) muss sich die Einwilligung auf die konkrete Verarbeitung beziehen.

Auch in diesem Punkt wird es kaum nennenswerte Änderungen durch die DSGVO geben.

Neue Regelungen

Bestimmte Pflichten werden neu geregelt, allerdings werden sich die praktischen Auswirkungen für viele Unternehmen wohl in Grenzen halten.

Gibt es bislang etwa die Pflicht zur Führung eines sogenannten Verfahrensverzeichnisses, so wird die DSGVO ab 2018 die Verpflichtung zur Erstellung beziehungsweise Vorhaltung eines Verzeichnisses von Verarbeitungstätigkeiten einführen. Sprachlich wird es zwar eine Änderung geben, inhaltlich ähnelt die neue Dokumentationspflicht aber den bisherigen Vorgaben. Weiterhin sind dann darin die essenziellen Informationen der Datenverarbeitung zusammengefasst. Es wird jedoch die Trennung zwischen öffentlichem – als auf Antrag jedermann bereitzustellendem – und internem Teil aufgehoben. Das Verzeichnis der Verarbeitungstätigkeiten muss zukünftig nur noch den Datenschutzaufsichtsbehörden auf Anfrage zur Verfügung gestellt werden.

Auch in puncto Vorabkontrolle werden Änderungen zu verzeichnen sein. Aktuell ist eine Vorabkontrolle dann durchzuführen, wenn besonders sensible Daten betroffen sind oder die Persönlichkeit des Betroffenen bewertet wird. Das zukünftige Verfahren wird inhaltlich ähnlich gelagert sein, allerdings einen neuen Namen erhalten, nämlich Folgenabschätzung. Diesbezüglich sind die Datenschutzaufsichtsbehörden gehalten, in Zukunft für ihren Zuständigkeitsbereich eine Liste der Verarbeitungsvorgänge bereitzustellen.

In der DSGVO finden sich einige Mindestvorgaben für die Inhalte der Datenschutz-Folgenabschätzung:

- systematische Beschreibung der geplanten Verarbeitungsvorgänge sowie des Zwecks der Verarbeitung,
- gegebenenfalls Angabe der berechtigten Interessen,
- Bewertung der Notwendigkeit sowie der Verhältnismäßigkeit der Verarbeitung,
- Risikobewertung in Bezug auf Rechte und Freiheiten der Betroffenen,
- geplante Abhilfemaßnahmen, also etwaige Sicherheitsvorkehrungen und spezielle Verfahren zur Vermeidung der bestehenden Risiken (in Bezug auf die Rechte und berechtigten Interessen der Betroffenen).

Außerdem ist, sofern ein Datenschutzbeauftragter durch das Unternehmen bestellt wurde, stets dessen Rat im Hinblick auf die Folgenabschätzung einzuholen.

Sonstiges

Weitere Änderungen gibt es in zahlreichen anderen Bereichen. Bislang mussten zum Beispiel Betroffene beziehungsweise die Aufsichtsbehörden bei Datenpannen nur benachrichtigt werden, wenn es um besonders sensible Daten ging (zum Beispiel Gesundheits- oder Kontodaten). Zukünftig sollen alle Datenschutzverletzungen »ohne unangemessene Verzögerung« (möglichst innerhalb von 72 Stunden) grundsätzlich an die zuständige Aufsichtsbehörde gemeldet werden; von dieser Pflicht wird es nur wenige Ausnahmen geben.

Parallel zur Meldepflicht müssen zukünftige Datenschutzverletzungen fortlaufend dokumentiert werden. Unterbleibt eine Meldung an die Aufsichtsbehörde, ist ebenfalls schriftlich der Grund dafür festzuhalten.

Die exakten inhaltlichen Anforderungen an zukünftige Datenschutzhinweise auf Websites sind derzeit noch nicht in Gänze absehbar. Die wesentlichen Grundzüge werden sich nicht ändern, allerdings wird die Datenschutzerklärung eher umfangreicher als kürzer.

Bislang gelten bestimmte Schwellenwerte für die verpflichtende Bestellung eines Datenschutzbeauftragten (Faustregel: bei mehr als neun mit der Datenverarbeitung betreuten Personen im Unternehmen).

Zukünftig ist ein Datenschutzbeauftragter zu bestellen, wenn die Kerntätigkeit der verantwortlichen Stelle »in der Durchführung von Verarbeitungsvorgängen besteht, welche aufgrund ihrer Art, ihres Umfangs und/oder ihrer Zwecke eine umfangreiche regelmäßige und systematische Beobachtung von betroffenen Personen erforderlich machen«, oder die Kerntätigkeit besonders sensible Daten betrifft (zum Beispiel Gesundheitsdaten).

Im Hinblick auf das Auslagern von IT- und sonstigen Dienstleistungen, die im Zusammenhang mit der Verarbeitung personenbezogener Daten stehen, wird es ebenfalls Änderungen geben. Diese werden jedoch weniger die Auftraggeber, also die verantwortliche Stelle, sondern den Auftragnehmer, also den Dienstleister, betreffen. ■

Michael Rohrlich
ist Rechtsanwalt und Fachautor aus Würselen. Seine beruflichen Schwerpunkte liegen auf dem Gebiet des Online-Rechts und des gewerblichen Rechtsschutzes.
www.rechtssicher.info

ARBEITSMARKT

TRENDS UND JOBS FÜR ENTWICKLER

Monatliches Ranking

Projektmarkt 2016

Der Projektmarkt zeigt sich im laufenden Jahr durchwachsen im Hinblick auf Stundensätze und persönliche Erwartungen, so berichtet eine aktuelle Marktstudie von Solcom.

Das vergangene Jahr war für die Mehrheit der befragten Freiberufler überwiegend erfolgreich. Allerdings hat sich auch der Anteil der Umfrageteilnehmer erhöht, die vergangenes Jahr nur eine sehr schwache Auslastung hatten. Zwar erwartet mehr als die Hälfte der Befragten in diesem Jahr ein Wachstum, allerdings befürchtet fast jeder Zehnte – und damit deutlich mehr als in 2015 – einen schrumpfenden Markt.

Bei den Stundensätzen ist der Ausblick in diesem Jahr weniger optimistisch. So erwarten weit weniger Umfrageteilnehmer eine bessere Entlohnung. Im Gegensatz dazu werden jedoch die persönlichen Chancen für das laufende Jahr positiver eingeschätzt, ein Drittel sieht hier bessere Möglichkeiten als im vergangenen Jahr.

Der Megatrend Big Data ist nun vollständig im Projektmarkt angekommen. Mit deutlichem Zuwachs sehen mittlerweile fast die Hälfte der Befragten das Sammeln und Auswerten großer Datenmengen als das Thema mit den größten Marktchancen.

Jobs für Entwickler

Die Gesamtzahl der Treffer der Recherche in der Datenbank von Jobkralle.de nach Jobangeboten für Webentwickler in den 15 größten deutschen Städten ist in diesem Monat um 7,6 Prozent zurückgegangen. Die meisten Jobs werden derzeit für München, Berlin und Hamburg ausgeschrieben (Bild 1). Auch in Köln, Frankfurt, Stuttgart und Düsseldorf herrscht noch eine rege Nachfrage. Bei der Verteilung der Angebote auf die Bundesländer lag Bayern vor Nordrhein-Westfalen und Baden-Württemberg.

Bei den Entwicklern von Apps für Mobilgeräte ist die Gesamtzahl der Treffer lediglich um ein knappes Prozent geschrumpft. Im Ländervergleich liegt hier ebenfalls Bayern vorn (Bild 2). Betrachtet man die deutschen Großstädte, so findet man die meisten Angebote wieder in München, Berlin und Hamburg.

Technologien

In Tabelle 1 finden Sie die in Stellenanzeigen geforderten Technologiekenntnisse. Dafür wird die Datenbank der Meta-Suchmaschine Jobkralle.de nach ausgewählten Stichworten abgefragt. Insgesamt lieferte die Abfrage diesmal 3,3 Prozent mehr Treffer als im Vormonat.

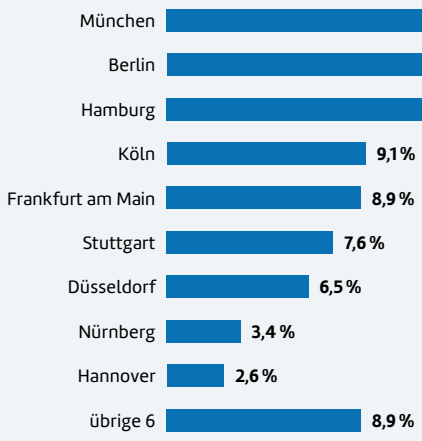
Am häufigsten ist in den Stellenanzeigen immer noch das Stichwort Cloud (über 6.900 Treffer), das seinen Anteil noch einmal auf 18,4 Prozent steigern konnte (Vormonat: 17,4 Prozent). Auch die am häufigsten genannte Smartphone-Technologie Android hat einen deutlichen Auftrieb erhalten.

Tabelle 1: Technologie

Rang	Technologie	Anteil *
1	Cloud	18,4 %
2	MySQL	12,0 %
3	HTML5	9,8 %
4	Big Data	7,7 %
5	Android	6,8 %
6	SharePoint	6,8 %
7	iOS	6,1 %
8	Microsoft SQL Server	5,9 %
9	Responsive Web	5,5 %
10	AngularJS	4,7 %
11	CSS3	4,3 %
12	Windows 10	3,5 %
13	WPF	2,6 %
14	NoSQL	2,5 %
15	Azure	1,8 %
16	WCF	1,6 %

* Prozentualer Anteil der Treffer

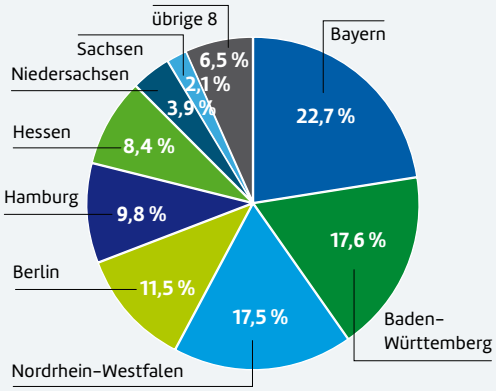
Jobs für Webentwickler



In München, Berlin und Hamburg werden derzeit am meisten Webentwickler gesucht (Bild 1)

web & mobile developer 6/2016 Quelle: Eigene Erhebungen, Jobkralle.de

Jobs für Mobile-Entwickler



Auf die Top 5 der Bundesländer entfallen nahezu 80 Prozent der Jobangebote (Bild 2)

web & mobile developer 6/2016 Eigene Erhebungen, Jobkralle.de

Zahl des Monats

Unverzichtbare Freelancer: Für **83 Prozent** der größeren Unternehmen haben **IT-Freelancer** eine eher große, große oder sehr große Bedeutung. **23 Prozent** des Projektvolumens in diesen Unternehmen wird von IT-Freelancern erbracht. Quelle: Bitkom

Adzuna

Wie viel bin ich wert?

Unter dem Markennamen ValueMyCV bietet Adzuna einen neuen Service an, mit dem jeder seinen Wert auf dem Arbeitsmarkt ermitteln kann. Hierzu lädt man einfach den eigenen Lebenslauf auf http://try.adzuna.de/value-my-cv und erhält binnen Sekunden eine statistische Einschätzung des eigenen Marktwerts in Form des Bruttojahresgehaltes – anonym und kostenlos. Die Grundlage bildet laut Adzuna eine Datenbank von 50.000 Lebensläufen inklusive dazugehöriger Gehaltsdaten, die auf Daten von deutschen Arbeitnehmern basieren. Bei der Analyse werden die aufgeführte Ausbildung, Berufserfahrung und Fähigkeiten mit ähnlichen Profilen in der Datenbank abgeglichen, um den Marktwert zu generieren. Daneben wird auch die Lage sowie das Geschlecht der Nutzer berücksichtigt.

http://try.adzuna.de

Bewerbungsgespräch

Was Firmen verbessern können

Als eines der größten Ärgernisse während des Bewerbungsprozesses empfinden viele Entwickler (15,4 Prozent) das Bewerbungsgespräch, das ergab die Stack-Overflow-Umfrage 2016 unter deutschen Entwicklern. Unternehmen können ihre Bewerbungsgespräche deutlich verbessern, wenn sie die Kandidaten dem Team vorstellen, was sich 55,6 Prozent aller Entwickler wünschen.

48,3 Prozent wollen darüber hinaus ihren Arbeitsplatz gezeigt bekommen, und 35,9 Prozent möchten schon im Bewerbungsgespräch den Live-Code sehen, mit dem sie zukünftig arbeiten sollen (Bild 3).

32,3 Prozent gaben an, dass sie vorab wissen möchten, wen sie im Bewerbungsgespräch treffen, und 26,4 Prozent wünschen sich weniger Quiz in der Bewerbung.

http://stackoverflow.com

Der Weg zum Job

Von einem Freund vorgestellt

Dass es für Unternehmen nicht leicht ist, Entwickler für sich zu gewinnen, zeigt die aktuelle Entwicklerumfrage von Stack Overflow auf eine ungewöhnliche Art und Weise. Der größte Anteil der Befragten ist zum jetzigen Arbeitsplatz nämlich nicht über Stellenanzeigen gelangt, sondern sie wurden von einem Freund vorgestellt, der bereits im Unternehmen arbeitet (24,8 Prozent). Zählt man diejenigen hinzu, die wussten, dass sie bei diesem Unternehmen arbeiten wollten, und sich direkt beworben haben, sind das immerhin 38,1 Prozent. Von firmeneigenen Recruitern angesprochen wurden dagegen lediglich 8,1 Prozent der Befragten (Bild 4). Soziale Medien wurden kaum als Antworten genannt. Facebook und Twitter kamen zusammen nur auf 0,9 Prozent.

http://stackoverflow.com

Evans Data

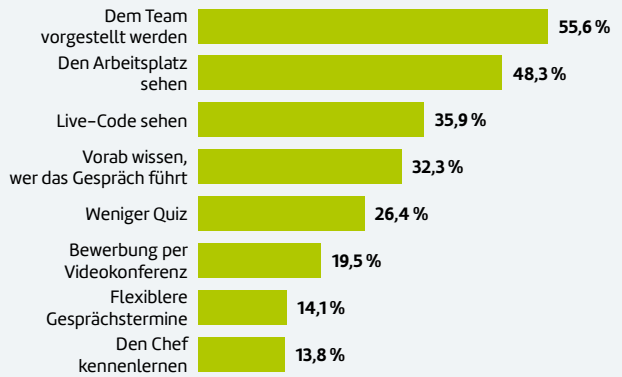
Entwickler fürchten die KI

Künstliche Intelligenz (KI) durchdringt immer weitere Bereiche in der Wirtschaft. Intelligente Maschinen in der Cloud werden schon jetzt eingesetzt, um Großunternehmen bei Entscheidungen zu unterstützen. In Zukunft wird so manche heute gut bezahlte Tätigkeit von KI-Software erledigt werden. Und wer wüsste besser, was mit Software möglich ist, als die Entwickler selbst. Evans Data hat sie befragt und das Ergebnis der Studie veröffentlicht: Entwickler fürchten, dass KI-Software sie in absehbarer Zeit überflüssig machen könnte.

Gefragt nach der am besorgniserregendsten Aussicht in ihrer Karriere wählte die größte Zahl der Befragten (29,1 Prozent) den Punkt »Ich und meine Entwicklerleistungen werden durch Künstliche Intelligenz ersetzt«.

www.evansdata.com

Bewerbungen verbessern



Die Hälfte der Entwickler will im Bewerbungsgespräch Team und Arbeitsplatz kennenlernen (Bild 3)

web & mobile developer 6/2016 Quelle: Stack Overflow Entwicklerumfrage 2016

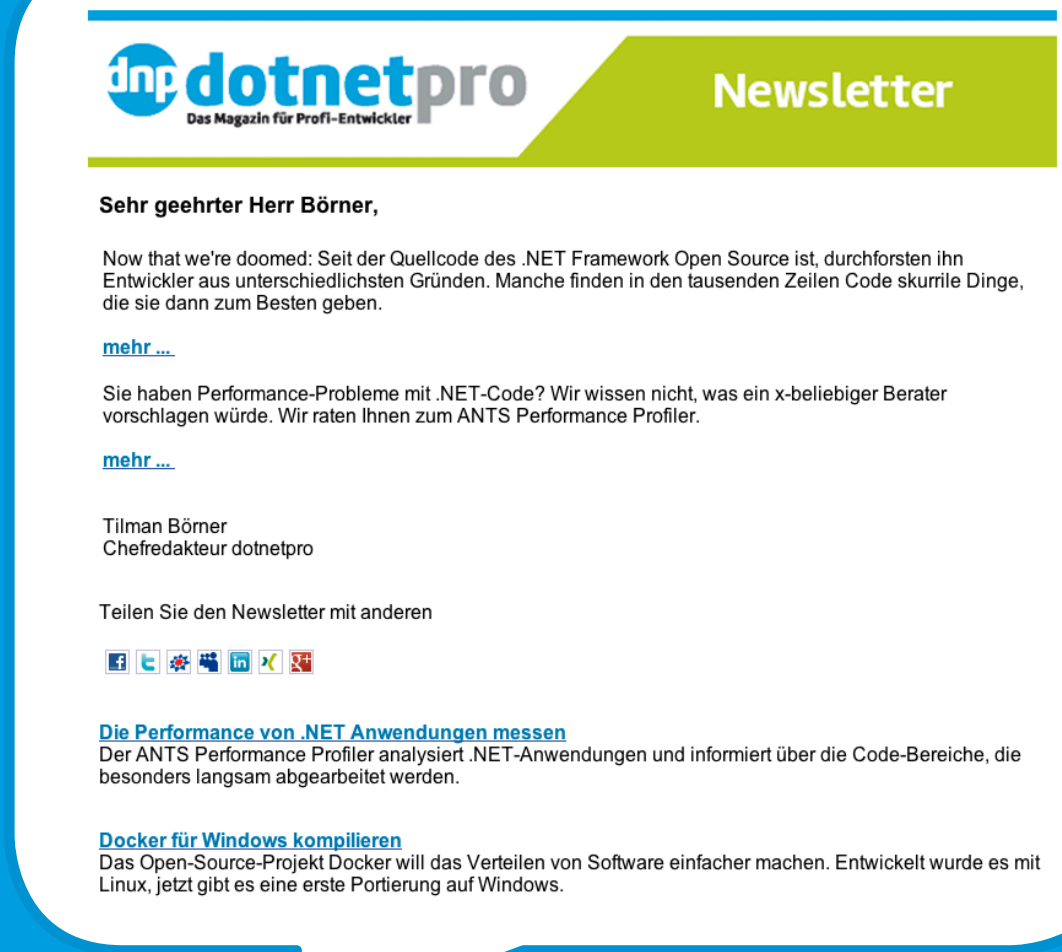
Wie Entwickler ihren Job finden

Jobseiten stehen nicht ganz oben auf der Liste der Wege, auf denen Entwickler ihren neuen Job finden (Bild 4)

web & mobile developer 6/2016 Quelle: Stack Overflow Entwicklerumfrage 2016

dotnetpro Newsletter

Top-Informationen für den .NET-Entwickler.
Klicken. Lesen. Mitreden.


dotnetpro
Das Magazin für Profi-Entwickler
Newsletter

Sehr geehrter Herr Börner,

Now that we're doomed: Seit der Quellcode des .NET Framework Open Source ist, durchforsten ihn Entwickler aus unterschiedlichsten Gründen. Manche finden in den tausenden Zeilen Code skurrile Dinge, die sie dann zum Besten geben.

mehr ...

Sie haben Performance-Probleme mit .NET-Code? Wir wissen nicht, was ein x-beliebiger Berater vorschlagen würde. Wir raten Ihnen zum ANTS Performance Profiler.

mehr ...

Tilman Börner
Chefredakteur dotnetpro

Teilen Sie den Newsletter mit anderen

Die Performance von .NET Anwendungen messen
Der ANTS Performance Profiler analysiert .NET-Anwendungen und informiert über die Code-Bereiche, die besonders langsam abgearbeitet werden.

Docker für Windows kompilieren
Das Open-Source-Projekt Docker will das Verteilen von Software einfacher machen. Entwickelt wurde es mit Linux, jetzt gibt es eine erste Portierung auf Windows.

Jetzt kostenlos anmelden:

dotnetpro.de

twitter.com/dotnetpro_mag

facebook.de/dotnetpro

gplus.to/dotnetpro

Anbieterverzeichnis

für Deutschland, Schweiz und Österreich.

Consulting / Dienstleister

ANEXIA Internetdienst-leistungs GmbH
Feldkirchner Straße 140
9020 Klagenfurt / AUSTRIA
T +43-50-556
F +43-50-556-500
info@anexia-it.com

ANEXIA wurde im Juni 2006 von Alexander Windbichler als klassischer Internet Service Provider gegründet. In den letzten Jahren hat sich ANEXIA zu einem stabilen, erfolgreichen und international tätigen Unternehmen entwickelt, das namhafte Kunden rund um den Globus mit Standorten in Wien, Klagenfurt, München, Köln und New York City betreut. ANEXIA bietet ihren Kunden hochwertige und individuelle Lösungen im Bereich Web- und Managed Hosting, sowie Individualsoftware und App Entwicklung.

---

prodot GmbH
Schifferstraße 196
47059 Duisburg
T: 0203 – 346945 – 0
F: 0203 – 346945 – 20
info@prodot.de
https://prodot.de

Intelligente Software für internationale Konzerne und mittelständische Unternehmen: prodot stärkt Kunden im weltweiten Wettbewerb – mit effizienten, stabilen und kostensenkenden Lösungen.
Durch das Zusammenspiel aus Know-how, Kreativität und Qualitätsmanagement leisten wir einen Beitrag zum langfristigen Erfolg unserer Auftraggeber.
Seit über 15 Jahren vertrauen uns deshalb Marktführer wie Aldi Süd, Microsoft und Siemens.
prodot – People. Passion. Performance..

eCommerce / Payment

Payone GmbH & Co. KG
Fraunhoferstraße 2-4
24118 Kiel
T: +49 431 25968-400
F: +49 431 25968-1400
sales@payone.de
www.payone.de

PAYONE ist einer der führenden Payment Service Provider und bietet modulare Lösungen zur ganzheitlichen Abwicklung aller Zahlungsprozesse im E-Commerce. Das Leistungsspektrum umfasst die Zahlungsabwicklung von allen relevanten Zahlarten mit integriertem Risikomanagement zur Minimierung von Zahlungsausfällen und Betrug. Standardisierte Schnittstellen und SDKs erlauben eine einfache Integration in bestehende IT- und mobile Systemumgebungen. Über Extensions können auch E-Commerce-Systeme wie Magento, OXID eSales, Demandware, Shopware, plentymarkets und viele weitere unkompliziert angebunden werden.

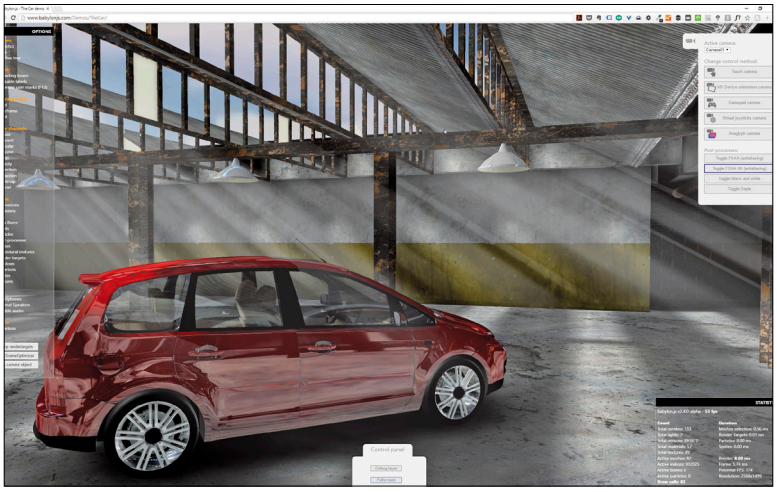
Web- / Mobile-Entwicklung & Content Management

digitalmobil GmbH & Co. KG
Bayerstraße 16a, 80335 München, T: +49 (0) 89 7 41 17 760, info@digitalmobil.com, www.digitalmobil.com

In allen Fragen rund um das Dienstleisterverzeichnis berät Sie Frau Roschke gerne persönlich!
Juliane Roschke ■ 089 / 7 41 17 – 283 ■ juliane.roschke@nmg.de

Die Ausgabe 7/2016 erscheint am 9. Juni 2016

Canvas: Die besten Grafik-Frameworks und -Bibliotheken



Das moderne Web möchte den Benutzer nicht nur mit Grafik und Animationen verwöhnen, sondern durch ausgefallene Interaktivität engagieren. Anzeigen wollen angeklickt werden oder noch besser: den Besucher zu einer unmittelbaren Handlung motivieren. So ist es auch kein Wunder, dass interaktive 3D-Modelle von Produkten oder gar virtuelle Erlebniswelten im Web immer öfter anzutreffen sind, sei es als ein Blickfang oder als Bestandteil eines Webshops. Noch nie zuvor standen Webentwicklern so viele verschiedene Technologien zur Verfügung, um diese Ziele auch wirklich zu erreichen. Visuell ansprechend und engagierend: Eine Webapplikation kann heute alle diese Merkmale besitzen, ohne den Browser in die Knie zu zwingen.

Rendering-Performance

Bei Performance-Verbesserung denkt man üblicherweise an Techniken, die für eine schnellere Übertragung sorgen. Wichtig ist allerdings auch die die Darstellungsperformance. Der Artikel behandelt grundlegende Konzepte, Techniken und Tools, die bei deren Überprüfung helfen. Im Einzelnen geht es darum, die JavaScript-Ausführung zu optimieren, CSS zu vereinfachen sowie darum, aufwendige Stiländerungen zu vermeiden.

Neuerungen von Swift 2.2

Mit Swift 2.2 legt Apple den Grundstein für die kommende Version 3 und lenkt die Sprache in die richtige Richtung. Was die Entwickler-Community betrifft, so ist Swift für Apple sicherlich aktuell das wichtigste Projekt. Die Sprache wurde nach ihrer erstmaligen Vorstellung auf der WWDC 2014 sehr positiv von den Entwicklern aufgenommen, und Apple arbeitete seitdem fortlaufend an weiteren neuen Features und Verbesserungen der Sprache.

OpenStack und seine Dienste

Openstack stellt eine Plattform für den Aufbau und Betrieb von virtualisierten Systemen und Clouds zur Verfügung. Ein Artikel erläutert den Aufbau und Nutzen dieses neuen Konzepts zur Virtualisierung. Alle heute genutzten IT-Dienste brauchen immer drei zentrale Ressourcen: die CPU mit Arbeitsspeicher, um die Programme abzuarbeiten, Netzwerke zur Kommunikation und Massenspeicher zur Ablage von Daten oder Programmen.

dotnetpro

Ausgabe 6/2016 ab 19. Mai am Kiosk

Ja, es gab sie: die Zeit vor dem Internet. Damals musste man alles selbst anschaffen. In Zeiten der Cloud kann das aber der Cloud angehören. Denn dank Internet gibt es alles auch im Netz, vor allem nützliche Dienste für Entwickler.
www.dotnetpro.de

Unsere digitalen Angebote

Wöchentlicher Newsletter
webundmobile.de/newsletter

Shop
shop.webundmobile.de

YouTube
youtube.com/user/developermedia

Facebook
facebook.com/webundmobile

Google +
gplus.to/webundmobile

Twitter
twitter.com/webundmobile

Stellenmarkt

dotnetpro + web & mobile Developer

- 25.800 Exemplare Gesamtauflage
- 25.300 Newsletter-Empfänger
- 66.600 PI'S

.NET Architektur HTML5/JavaScript iOS/Android

Kontakt:
Jens Schmidtmann, Klaus Ahlering • Tel. 089/74117-125 • sales@nmg.de

Ihr Partner für mehr Online-Wachstum

Wir planen, entwickeln und steuern
Websites, **Apps** und **Kampagnen**.

Unsere Ziele sind Ihre Ziele:

- **Mehr Sichtbarkeit**
- **Mehr Traffic**
- **Mehr Leads**
- **Mehr Conversions**

- **Mehr Kunden**

Besuchen Sie uns unter
www.digitalmobil.com

digitalmobil